LLNL-PROP-404138-DRAFT

RFP Attachment 2

DRAFT STATEMENT OF WORK

May 21, 2008

ADVANCED SIMULATION AND COMPUTING (ASC)

Comment [MKS1]: http://www.shannontech.com/ParkVision/Sequoia/Sequoia.html

The sequoia, which sprouts exclusively from seeds, continues to grow throughout its life. It usually dies only when toppled by wind or other catastrophic event. The trees are virtually impervious to disease; the oldest specimen on record lived approximately 3200 years. Their cinnamon color is an effect of the presence of tannin in the thick bark and heartwood which also contributes to their resistance to fire, insects, fungus, and decay.

Muir described the trees as "colonnades" along the edges of meadows, an comparison which can easily be understood from the images below.

B563020

LAWRENCE LIVERMORE NATIONAL SECURITY, LLC (LLNS)
LAWRENCE LIVERMORE NATIONAL LABORATORY (LLNL)
LIVERMORE, CALIFORNIA

Table of Contents

1.0 Introduction - 10 -
1.1 NNSA's Stockpile Stewardship Program and Complex 2030 - 10 -
1.2 Advanced Simulation and Computing (ASC) Program Overview - 11 -
1.3 ASC Applications Overview - 15 -
1.3.1 Current IDC Description - 17 -
1.3.2 Petascale Applications Predictivity Improvement Strategy - 20 -
1.3.3 Code Development Strategy - 22 -
1.4 ASC Software Development Environment - 23 -
1.5 ASC Applications Execution Environment - 29 -
1.6 ASC Sequoia Operations - 30 -
1.6.1 Sequoia Support Model - 33 -
1.7 ASC Dawn and Sequoia Simulation Environment - 34 -
1.8 Sequoia Timescale and High Level Deliverables - 38 -
2.0 Sequoia High-Level Hardware Requirements - 40 -
2.1 Sequoia System Peak (MR) - 41 -
2.1.1 Sequoia System Performance (TR-1) - 41 -
2.2 Sequoia Major System Components (TR-1) - 41 -
2.2.1 IO Subsystem Architecture (TR-1) - 41 -
2.3 Sequoia Component Scaling (TR-1) - 42 -
2.4 Sequoia Node Requirements (TR-1) - 43 -
2.4.1 Node Architecture (TR-1) - 43 -
2.4.2 Core Characteristics (TR-1) - 44 -
2.4.3 IEEE 754 32-Bit Floating Point Numbers (TR-3) - 44 -
2.4.4 Inter Core Communication (TR-1) - 44 -
2.4.5 Node Interconnect Interface (TR-2) - 44 -
2.4.6 Hardware Support for Low Overhead Threads (TR-1) - 45 -
2.4.7 Hardware Support for Innovative node Programming Models (TR-2) - 45 -
2.4.8 Programmable Clock (TR-2) - 45 -
2.4.9 Hardware Interrupt (TR-2) - 45 -
2.4.10 Hardware Performance Monitors (TR-1) - 45 -
2.4.11 Hardware Debugging Support (TR-1) - 46 -
2.4.12 JTAG Infrastructure - 46 -
2.4.13 No Local Hard Disk (TR-1) - 46 -
2.4.14 Remote Manageability (TR-1) - 46 -
2.5 I/O Node Requirements (TR-1) - 47 -
2.5.1 ION Count (TR-1) - 47 -
2.5.2 ION IO Configuration (TR-2) - 47 -
2.5.3 ION Delivered Performance (TR-2) - 47 -
2.6 Login Node Requirements (TR-1) - 48 -
2.6.1 LN Count (TR-1) - 48 -
2.6.2 LN Locally Mounted Disk and Multiple Boot (TR-1) - 48 -
2.6.3 LN IO Configuration (TR-2) - 48 -
2.6.4 LN Delivered Performance (TR-2) - 49 -

2.7 Service Node Requirements (TR-1) - 49 -
2.7.1 SN Scalability (TR-1) - 49 -
2.7.2 SN Communications (TR-1) - 49 -
2.7.3 SN Locally Mounted Disk and Multiple Boot (TR-1) - 49 -
2.7.4 SN IO Configuration (TR-2) - 50 -
2.7.5 SN Delivered Performance (TR-2) - 50 -
2.8 Sequoia Interconnect (TR-1) - 50 -
2.8.1 Interconnect Messaging Rate (TR-1) - 50 -
2.8.2 Interconnect Delivered Latency (TR-1) - 50 -
2.8.3 Interconnect Off-Node Aggregate Delivered Bandwidth (TR-1) - 51 -
2.8.4 Interconnect MPI Task Placement Delivered Bandwidth Variation (TR-2) - 51 -
2.8.5 Delivered Minimum Bi-Section Bandwidth (TR-2) - 52 -
2.8.6 Broadcast Delivered Latency (TR-2) - 52 -
2.8.7 All Reduce Delivered Latency (TR-2) - 52 -
2.8.8 Interconnect Hardware Bit Error Rate (TR-1) - 53 -
2.8.9 Global Barriers Network Delivered Latency (TR-2) - 53 -
2.8.10 Cluster Wide High Resolution Event Sequencing (TR-2) - 54 -
2.8.11 Interconnect Security (TR-2) - 54 -
2.9 Input/Output Subsystem (TR-1) - 54 -
2.9.1 File IO Subsystem Performance (TR-1) - 55 -
2.9.2 LN & SN High-Availability RAID Arrays (TR-1) - 57 -
2.9.3 LN & SN High IOPS RAID (TR-2) - 57 -
2.10 Management Ethernet Infrastructure (TR-1) - 57 -
2.11 Early Access to Sequoia Technology (TR-1) - 58 -
2.12 Sequoia Hardware Options - 58 -
2.12.1 Sequoia Enhanced IO Subsystem (TO-1) - 58 -
2.12.2 Sequoia Half Memory (TO-1) - 58 -
2.12.3 Sequoia14 System Performance (MO) - 58 -
2.12.4 Sequoia14 Enhanced IO Subsystem (TO-1) - 58 -
2.12.5 Sequoia14 Half Memory (TO-1) - 58 -
3.0 Sequoia High-Level Software Requirements (TR-1) - 60 -
3.1 LN, ION and SN Operating System Requirements - 60 -
3.1.1 Base Operating System and License (TR-1) - 60 -
3.1.2 Function Shipping From LWK (TR-1) - 60 -
3.1.3 Remote Process Control Tools Interface (TR-1) - 61 -
3.1.4 OS Virtualization (TR-3) - 61 -
3.1.5 Multi-Boot Capability (TR-1) - 61 -
3.1.6 Pluggable Authentication Mechanism (TR-1) - 61 -
3.1.7 Node Fault Tolerance and Graceful Degradation of Service (TR-2) - 61 -
3.1.8 Networking Protocols (TR-1) - 62 -
3.1.9 OFED IBA Software Stack (TR-1) - 62 -
3.1.10 IBA Upper Layer Protocols (TR-1) - 62 -
3.1.11 Local File Systems (TR-2) - 62 -
3.1.12 Operating System Security (TR-2) - 63 -
3.2 Light-Weight Kernel and Services (TR-1) - 63 -
3.2.1 LWK Livermore Model Support (TR-1) - 63 -

3.2.2 LWK Supported System Calls (TR-1) ..... - 64 -
3.2.3 LWK Job Launch (TR-1) ..... - 65 -
3.2.4 Diminutive Noise LWK (TR-1) ..... - 65 -
3.2.5 LWK Application Remote Debugging Support (TR-1) ..... - 65 -
3.2.6 LD_PRELOAD Mechanism (TR-2) ..... - 65 -
3.2.7 LWK Limitations (TR-1) ..... - 65 -
3.2.8 RAS Management (TR-1) ..... - 66 -
3.2.9 LWK 64b HPM Support (TR-1) ..... - 66 -
3.2.10 Application Checkpoint and Restart (TR-2) ..... - 66 -
3.2.11 LWK "RAM Disk" Support (TR-2) ..... - 67 -
3.3 Distributed Computing Middleware ..... - 67 -
3.3.1 Kerberos (TR-1) ..... - 67 -
3.3.2 LDAP Client (TR-1) ..... - 67 -
3.3.3 NFSv4.1 Client (TR-1) ..... - 68 -
3.3.4 Cluster Wide Service Security (TR-1) ..... - 68 -
3.4 System Resource Management (SRM) (TR-1) ..... - 68 -
3.4.1 SRM Security (TR-1) ..... - 69 -
3.4.2 SRM API Requirements (TR-1) ..... - 69 -
3.4.3 Node Reboot API (TR-1) ..... - 69 -
3.4.4 Network Topology API (TR-1) ..... - 69 -
3.4.5 Job Manipulation Commands and API (TR-1) ..... - 69 -
3.4.6 Job Signaling API (TR-1) ..... - 69 -
3.4.7 User Task Launch API (TR-1) ..... - 70 -
3.4.8 User Task Connectivity API (TR-1) ..... - 70 -
3.4.9 SRM STDIO (TR-1) ..... - 70 -
3.4.10 System Initiated Checkpoint API (TR-3) ..... - 70 -
3.4.11 Predicting Failed Nodes (TR-2) ..... - 70 -
3.5 Integrated System Administration Tools ..... - 70 -
3.5.1 Single Point for System Administration (TR-1) ..... - 70 -
3.5.2 System Admin (TR-1) ..... - 71 -
3.5.3 System Debugging and Performance Analysis (TR-2) ..... - 71 -
3.5.4 Scalable Centralized Resource Data Base (TR-2) ..... - 71 -
3.5.5 User Maintenance (TR-2) ..... - 72 -
3.5.6 Login Load Balancing Service(TR-2) ..... - 72 -
3.6 Parallelizing Compilers/Translators ..... - 72 -
3.6.1 Baseline Languages (TR-1) ..... - 72 -
3.6.2 Baseline Language Optimizations (TR-1) ..... - 72 -
3.6.3 Baseline Language 64b Pointer Default (TR-1) ..... - 72 -
3.6.4 Baseline Language Standardization Tracking (TR-1) ..... - 73 -
3.6.5 Common Preprocessor for Baseline Languages (TR-2) ..... - 73 -
3.6.6 Base Language Interprocedural Analysis (TR-2) ..... - 73 -
3.6.7 Baseline Language Compiler Generated Listings (TR-2) ..... - 73 -
3.6.8 C++ Functionality (TR-2) ..... - 73 -
3.6.9 Cray Pointer Functionality (TR-2) ..... - 73 -
3.6.10 Baseline Language Support for the "Livermore Model" (TR-1) ..... - 73 -
3.6.11 Baseline Language and GNU Interoperability (TR-1) ..... - 75 -

3.6.12 Runtime GNU Libc Backtrace (TR-2) - 75 -
3.6.13 Debugging Optimized Applications (TR-2) - 75 -
3.6.14 Floating Point Exception Handling (TR-2) - 75 -
3.7 Debugging and Tuning Tools - 76 -
3.7.1 Petascale Code Development Tools Infrastructure (TR-1) - 76 -
3.7.2 Debugger for Petascale Applications (TR-1) - 79 -
3.7.3 Stack Traceback (TR-2) - 82 -
3.7.4 User Access to A Scalable Stack Trace Analysis Tool (TR-2) - 82 -
3.7.5 Lightweight Corefile API (TR-2) - 82 -
3.7.6 Profiling Tools for Applications (TR-1) - 83 -
3.7.7 Event Tracing Tools for Applications (TR-1) - 83 -
3.7.8 Performance Statistics Tools for Applications (TR-1) - 84 -
3.7.9 Scalable Visualization of Trace Data (TR-1) - 84 -
3.7.10 Timer API (TR-2) - 84 -
3.7.11 Valgrind Infrastructure and Tools (TR-1) - 84 -
3.8 Applications Building - 84 -
3.8.1 LN Cross-Compilation Environment for CN and ION (TR-1) - 85 -
3.8.2 Linker and Library Building Utility (TR-1) - 85 -
3.8.3 GNU Make Utility (TR-1) - 85 -
3.8.4 Source Code Management (TR-2) - 85 -
3.8.5 Dynamic Processor Allocation (TR-2) - 85 -
3.9 Application Programming Interfaces (TR-1) - 85 -
3.9.1 Optimized Message-Passing Interface (MPI) Library (TR-1) - 86 -
3.9.2 Low Level Communication API (TR-1) - 87 -
3.9.3 User Level Thread Library (TR-1) - 87 -
3.9.4 Link Error Verification Facilities - 87 -
3.9.5 Graphical User Interface API (TR-1) - 87 -
3.9.6 Visualization API (TR-2) - 87 -
3.9.7 Math Libraries (TR-2) - 88 -
3.9.8 Hardware Debugging API (TR-2) - 88 -
3.10 Compliance with DOE Security Mandates (TR-1) - 88 -
3.11 On-Line Document (TR-2) - 88 -
3.12 Early Access to Sequoia Software Technology (TR-1) - 88 -
4.0 Dawn High-Level Hardware Requirements - 89 -
4.1 Dawn 0.5 petaFLOP/s System (MR) - 90 -
4.2 (4.3) Dawn Component Scaling (TR-1) - 90 -
4.3 (4.12) Dawn Hardware Options - 90 -
4.3.1 (4.12.1) Dawn Enhanced IO Subsystem (TO-1) - 90 -
4.3.2 (4.12.2) Dawn Double Memory (TO-1) - 90 -
4.3.3 (4.12.2) Dawn Double ION/LN Memory (TO-2) - 91 -
5.0 Dawn High Level Software Requirements - 92 -
6.0 Integrated System Features (TR-1) - 93 -
6.1 System RAS (TR-1) - 94 -
6.1.1 Hardware Failure Rate Impact on Applications (TR-1) - 94 -
6.1.2 Mean Time Between Failure Calculation (TR-1) - 94 -
6.1.3 Failure Protection Methods (TR-1) - 94 -

6.1.4 Data Integrity Checks (TR-1) - 95 -
6.1.5 Interconnect Reliability (TR-1) - 95 -
6.1.6 Link-Level Errors (TR-1) - 95 -
6.1.7 Capability Application Reliability (TR-1) - 96 -
6.1.8 Power Cycling (TR-3) - 96 -
6.1.9 Hot Swap Capability (TR-2) - 96 -
6.1.10 Production Level System Stability (TR-2) - 96 -
6.1.11 System Down Time (TR-2) - 96 -
6.1.12 Scalable RAS Infrastructure (TR-1) - 97 -
6.1.13 System Graceful Degradation Failure Mode (TR-2) - 98 -
6.1.14 Node Processor Failure Tolerance (TR-2) - 99 -
6.1.15 Node Memory Failure Tolerance (TR-2) - 99 -
6.2 Hardware Maintenance (TR-1) - 99 -
6.2.1 On-site Parts Cache (TR-1) - 99 -
6.2.2 Secure FRU Components (TR-1) - 100 -
6.3 Software Support (TR-1) - 100 -
6.4 On-site Analyst Support (TR-1) - 100 -
7.0 Facilities Requirements - 102 -
7.1 Power & Cooling Requirements (TR-1) - 104 -
7.1.1 Rack Power and Cooling (TR-1) - 104 -
7.1.2 Rack PDU (TR-1) - 104 -
7.2 Floor Space Requirements (TR-1) - 104 -
7.2.1 Dawn Floor Space Requirement (TR-1) - 105 -
7.2.2 Sequoia Floor Space Requirement (TR-1) - 105 -
7.3 Rack Height and Weight (TR-1) - 105 -
7.4 Rack Seismic Protection (TR-2) - 105 -
7.5 Installation Plan (TR-2) - 106 -
8.0 Project Management - 107 -
8.1 Performance Reviews (TR-1) - 109 -
8.2 Detailed Sequoia Plan Of Record (TR-1) - 109 -
8.2.1 Full-Term Project Management Plan (TR-1) - 109 -
8.2.2 Full-Term Hardware Development Plan (TR-1) - 111 -
8.2.3 Full-Term Software Development Plan (TR-1) - 111 -
8.2.4 Detailed Year Plan (TR-1) - 113 -
8.3 Project Milestones (TR-1) - 113 -
8.3.1 Full-Term Sequoia Plan of Record (TR-1) - 114 -
8.3.2 FY09 On-Site Support Personnel (TR-1) - 114 -
8.3.3 CY09 Plan and Review – Jan 2009 - 114 -
8.3.4 Dawn Demonstration – Feb 2009 (TR-1) - 115 -
8.3.5 Dawn Acceptance – March 2009 (TR-1) - 115 -
8.3.6 GFY10 On-Site Support Personnel – Oct 2009 (TR-1) - 115 -
8.3.7 GFY10 Dawn Support – Oct 2009 (TR-1) - 115 -
8.3.8 CY10 Plan and Review – Dec 2009 (TR-1) - 115 -
8.3.9 Sequoia Prototype Review – June 2010 - 115 -
8.3.10 GFY11 On-Site Support Personnel – Oct 2010 (TR-1) - 116 -
8.3.11 GFY11 Dawn Support – Oct 2010 (TR-1) - 116 -

8.3.12 CY11 Plan and Review – Dec 2010 (TR-1) - 116 -
8.3.13 Sequoia Build – March 2011 (TR-1) - 116 -
8.3.14 Sequoia Demonstration – June 2011 (TR-1) - 116 -
8.3.15 Sequoia Acceptance and LA – Sept 2011 (TR-1) - 117 -
8.3.16 GFY12 On-Site Support Personnel – Oct 2011 (TR-1) - 117 -
8.3.17 GFY12 Dawn Support – Oct 2011 (TR-1) - 117 -
8.3.18 Sequoia Production General Availability – Dec 2011 (TR-1) - 117 -
8.3.19 GFY13 On-Site Support Personnel – Oct 2012 (TR-1) - 117 -
8.3.20 GFY13 Dawn Support – Oct 2012 (TR-1) - 118 -
8.3.21 GFY13 Sequoia Support – Oct 2012 (TR-1) - 118 -
8.3.22 GFY14 On-Site Support Personnel – Oct 2013 (TR-1) - 118 -
8.3.23 FY14 Dawn Support – Oct 2013 (TR-1) - 118 -
8.3.24 GFY14 Sequoia Support – Oct 2013 (TR-1) - 118 -
8.3.25 GFY15 On-Site Support Personnel – Oct 2014 (TR-1) - 118 -
8.3.26 GFY15 Sequoia Support – Oct 2014 (TR-1) - 118 -
8.3.27 GFY16 On-Site Support Personnel – Oct 2015 (TR-1) - 118 -
8.3.28 GFY16 Sequoia Support – Oct 2015 (TR-1) - 118 -
9.0 Performance of the System - 119 -
9.1 Benchmark Suite - 120 -
9.1.1 Sequoia Marquee Benchmarks - 121 -
9.1.2 Sequoia Tier 2 Benchmarks - 124 -
9.1.3 Sequoia Tier 3 Benchmarks - 126 -
9.2 Benchmark System Configuration (TR-1) - 127 -
9.3 Sequoia Marquee Benchmark Test Procedures (TR-1) - 127 -
9.4 Performance Measurements (TR-1) - 129 -
9.4.1 Modifications - 131 -
9.4.2 Sequoia Execution Requirements - 132 -
10.0 Appendix A Glossary - 133 -
10.1 Hardware - 133 -
10.2 Software - 137 -

This document was prepared as an account of work sponsored by an agency of the United States government. Neither the United States government nor Lawrence Livermore National Security, LLC, nor any of their employees makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States government or Lawrence Livermore National Security, LLC. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States government or Lawrence Livermore National Security, LLC, and shall not be used for advertising or product endorsement purposes.

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Requirements Definitions

Particular paragraphs of this Statement of Work (SOW) have priority designations, which are defined as follow.

(a) Mandatory Requirements designated as (MR)

Mandatory Requirements (designated MR) in the Statement of Work (SOW) are performance features that are essential to LLNS requirements, and an Offeror must satisfactorily propose all Mandatory Requirements in order to have its proposal considered responsive.

(b) Mandatory Option Requirements designated as (MO)

Mandatory Option Requirements (designated MO) in the SOW are features, components, performance characteristics, or upgrades whose availability as options to LLNS are mandatory, and an Offeror must satisfactorily propose all Mandatory Option Requirements in order to have its proposal considered responsive. LLNS may or may not elect to include such options in the resulting subcontract(s). Therefore, each MO shall appear as a separately identifiable item in Offeror's proposal.

(c) Technical Option Requirements designated as (TO-1, TO-2 and TO-3)

Technical Option Requirements (designated TO-1, TO-2, or TO-3) in the SOW are features, components, performance characteristics, or upgrades that are important to LLNS, but which will not result in a nonresponsive determination if omitted from a proposal. Technical Options add value to a proposal. Technical Options are prioritized by dash number. TO-1 is most desirable to LLNS, while TO-2 is more desirable than TO-3. Technical Option responses will be considered as part of the proposal evaluation process; however, LLNS may or may not elect to include Technical Options in the resulting subcontract(s). Each proposed TO should appear as a separately identifiable item in an Offeror's proposal response.

(d) Target Requirements designated as (TR-1, TR-2 and TR-3).

Target Requirements (designated TR-1, TR-2, or TR-3), identified throughout the SOW, are features, components, performance characteristics, or other properties that are important to LLNS, but which will not result in a nonresponsive determination if omitted from a proposal. Target Requirements add value to a proposal. Target Requirements are prioritized by dash number. TR-1 is most desirable, while TR-2 is more desirable than TR-3. TR-1s and Mandatory Requirements are of equal value. The aggregate of MRs and TR-1s form a baseline system. TR-2s are goals that boost a baseline system, taken together as an aggregate of MRs, TR-1s and TR-2s, into the moderately useful system. TR-3s are stretch goals that boost a moderately useful system, taken together as an aggregate of MRs, TR-1s, TR-2s and TR-3s, into the highly useful system. Therefore, the ideal ASC Dawn and Sequoia systems will meet or exceed all MRs, TR-1s, TR-2s and TR-3s requirements. MOs are alternative sizes of the system that may be considered for technical and/or budgetary reasons. Technical Option Requirements may also affect LLNS perspective of the ideal ASC Dawn and Sequoia systems, depending on future ASC Program budget considerations. Target Requirement responses will be considered as part of the proposal evaluation process.

1.0 Introduction

Offeror may replace this section in its technical proposal(s) response with an overview of the proposed Dawn and Sequoia systems, technology development, project plan and build strategy.

1.1 NNSA's Stockpile Stewardship Program and Complex 2030

The National Nuclear Security Administration (NNSA) Advanced Simulation and Computing (ASC) computational resources are essential to enable nuclear weapon scientists to fulfill stockpile stewardship requirements through simulation in lieu of underground testing. Modern simulations on powerful computing systems are key to supporting our national security mission. As the nuclear stockpile moves further from the nuclear test base through either the natural aging of today's stockpile or introduction of modifications, the realism and accuracy of ASC simulations must further increase through development of improved physics models and methods requiring ever greater computational resources.

Problems at the highest end of this computational spectrum have been, and will continue to be, a principal driver for the ASC Program as highly predictive codes are developed (as outlined in the ASC Roadmap[1] and the evolving Predictive Capability Framework[2]) between 2008 and 2020. Predictive simulation of nuclear weapons performance requires rigorous assessment of margins and quantification of uncertainties. To be predictive, these uncertainties must be small enough to allow the certification of nuclear warheads without resorting to underground nuclear tests. Predictive simulation eliminates the technical need for future nuclear tests.

Reducing uncertainties sufficiently for predictive simulation requires advances in the fidelity of physics models, the accuracy of numerical algorithms, and their resolution and the ability to assess uncertainty – all ASC Program Roadmap goals. These in turn are dependent on the level of computing that can be brought to bear. The ASC Program requires an appropriate mix of platforms to quantify uncertainties and to predict with confidence. Capability, together with capacity and advanced architecture, systems are components of the balanced triad necessary for success in weapons simulation, as described in the ASC Platform Plan. The ASC Platform Plan describes the need for new computing resources to support uncertainty quantification (UQ) and reduction in phenomenology (i.e., replacing calibrated models with physics-based models).

As part of the Stockpile Stewardship Program Plan[3], the National Nuclear Security Administration (NNSA) Defense Programs (DP) recently set forth a goal for transforming the nuclear weapons complex into a responsive, modern infrastructure over the next two decades, while continuing to address needs in the enduring national nuclear weapons stockpile, as warheads age and move further from the test base. A modern, responsive weapons complex demands a balanced and predictive simulation infrastructure, including powerful systems like Sequoia to support Uncertainty Quantification (UQ), improving the physical models in the design codes, and more effective use of 3D models.Accomplishing this effectively will require performance at least 24 times the delivered performance of design codes today on Purple and 20 times improvement over BlueGene/L (BG/L)

[1] http://www.sandia.gov/ASC/pubs_pres/pubs/ASC%20RdMap1206r.pdf

[2] http://www.sandia.gov/NNSA/ASC/univ/psaap/kusnezov.pdf

[3] http://www.nnsa.doe.gov/docs/Stockpile_Overview_November_13_2006.pdf

for underlying materials studies. The preceding performance measures represent characterizing requirements for the Sequoia system.

The critical importance of UQ for all of these mission elements stems from its systematic approach to quantifying margins and uncertainty and hence improve confidence in the predicted weapons performance. Uncertainties that are accurately quantified can be risk managed. Responsibly managed risks allow NNSA's highest level weapons certification processes to continue with confidence.

The fundamental benefits from successful implementation of Sequoia are agile design and responsive certification infrastructure, increased accuracy in material property data, improved models for understood physical processes that are known to be important, meeting programmatic requirements for uncovering missing physics, and improving the performance of complex models and algorithms in the design codes. All of these are necessary to achieve predictive simulation in support of NNSA's modern-responsive weapons complex.

1.2 Advanced Simulation and Computing (ASC) Program Overview

The Accelerated Strategic Computing Initiative (ASCI) was established in 1995 as a critical element to help shift from test-based confidence to science- and simulation-based confidence. Specifically, ASCI was a focused and balanced program that accelerated the development of simulation capabilities needed to analyze and predict the performance, safety, and reliability of nuclear weapons and certify their functionality—far exceeding what might have been achieved in the absence of a focused initiative.

To realize its vision, ASCI created simulation capabilities based on advanced, 3D weapon codes coupled with functional, scalable high-performance computing. The result are simulations that enable assessment and certification of the safety, performance, and reliability of nuclear systems, in both 2D, and entry-level 3D simulations. The left panel of Figure 1-1 depicts the initial goals of the first ten years of ASCI. These simulation capabilities also help scientists understand weapons aging, predict when components will have to be replaced, and evaluate the implications of changes in materials and fabrication processes to the design life of the aging weapon systems. This science-based understanding is essential to ensure that changes brought about through aging or remanufacturing will not adversely affect the enduring stockpile.

In 2000, ASCI transitioned from an initiative to a program with an enduring mission; renamed the Advanced Simulation and Computing (ASC) Program. The establishment of the ASC Program affirmed simulation and modeling as key decision-making tools and cemented their long-term role as integral components of the Stockpile Stewardship Program (SSP). The middle panel of Figure 1-1 depicts the predictive simulation goals of SSP for the ASC Program during the lifetime of the Sequoia platform. Overall, the SSP through ASC Program:

Allows the U.S. to continue an underground nuclear test moratorium and still maintain a reliable nuclear weapons stockpile.

Ensures that all aspects of nuclear weapons stockpile operations are safe and secure— from design and engineering through dismantlement.

Generates a large return on investment by providing cost-effective, simulation-based solutions (without testing) to issues facing the nuclear weapons stockpile.

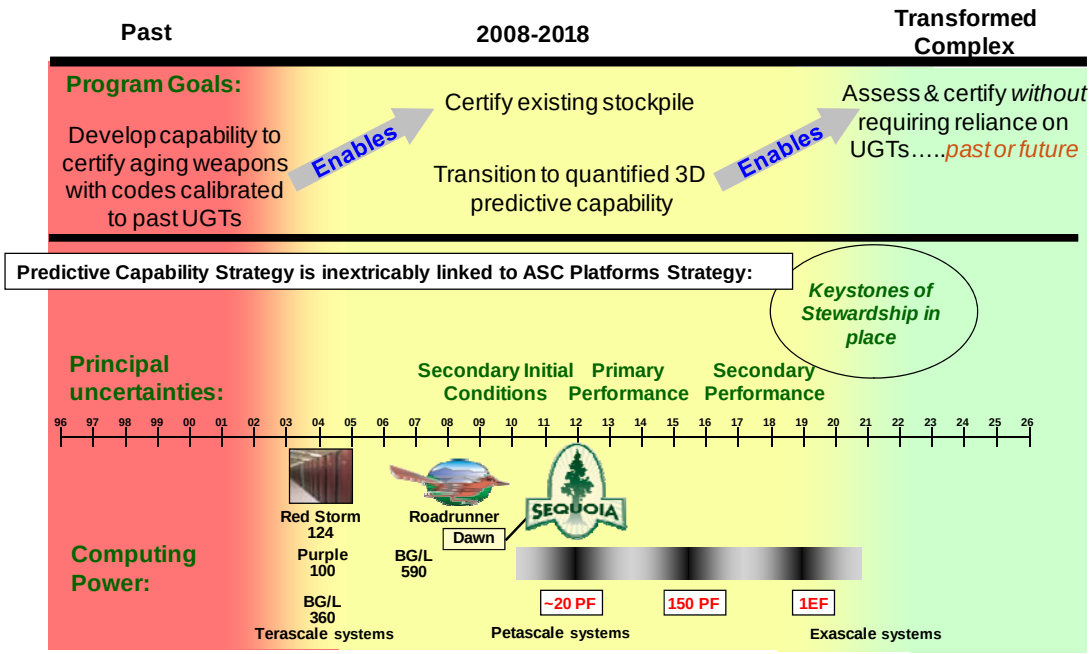


Figure 1-1: Simulation is key to eliminating the technical requirement for nuclear testing.

Lastly, as the US maintains its moratorium on underground nuclear tests, the Complex cannot continue to base its simulation and modeling efforts solely on data that are increasingly removed from the reality of the aged-weapons performance. Previously, both the limited computational tools and the near-term commitments to support the stockpile necessitated this approach. Now, however, the ASC Program has a development path for the needed software and hardware tools to move towards a quantified predictive capability (Figure 1-2). The ASC Roadmap focuses the ASC Program's efforts over the next decade on providing new levels of predictive capability to the SSP. It defines focus areas and supporting goals and targets required to achieve predictive capability in modeling and simulation, and it articulates a sequential, priority-based approach to achieving a new level of fidelity, adding confidence to SSP decisions and supporting a capability-based nuclear deterrent into the future.

Computer simulation is, and will continue to be, the only means to responsively address emerging issues related to systems under nuclear conditions. This continued capability is crucial to the nation's commitment to cease underground nuclear tests. The ASC Program is following two paths that allow it to maintain the testing moratorium: the traditional path of calibrating models to underground test data and performing simulations in regimes that are minimally removed from the applicable parameter space, and the rigorous, science-based path intended to address a diverse portfolio of current and future nuclear applications.

As Figure 1-2 illustrates, aging and refurbishment push nuclear weapons behavior into an area where the uncertainty associated with traditional approaches becomes progressively larger. To credibly address this space and predict performance further from the as-tested configurations, the ASC Program must create modern physical models with capabilities enabling confident calculation in these new and more applicable regimes.

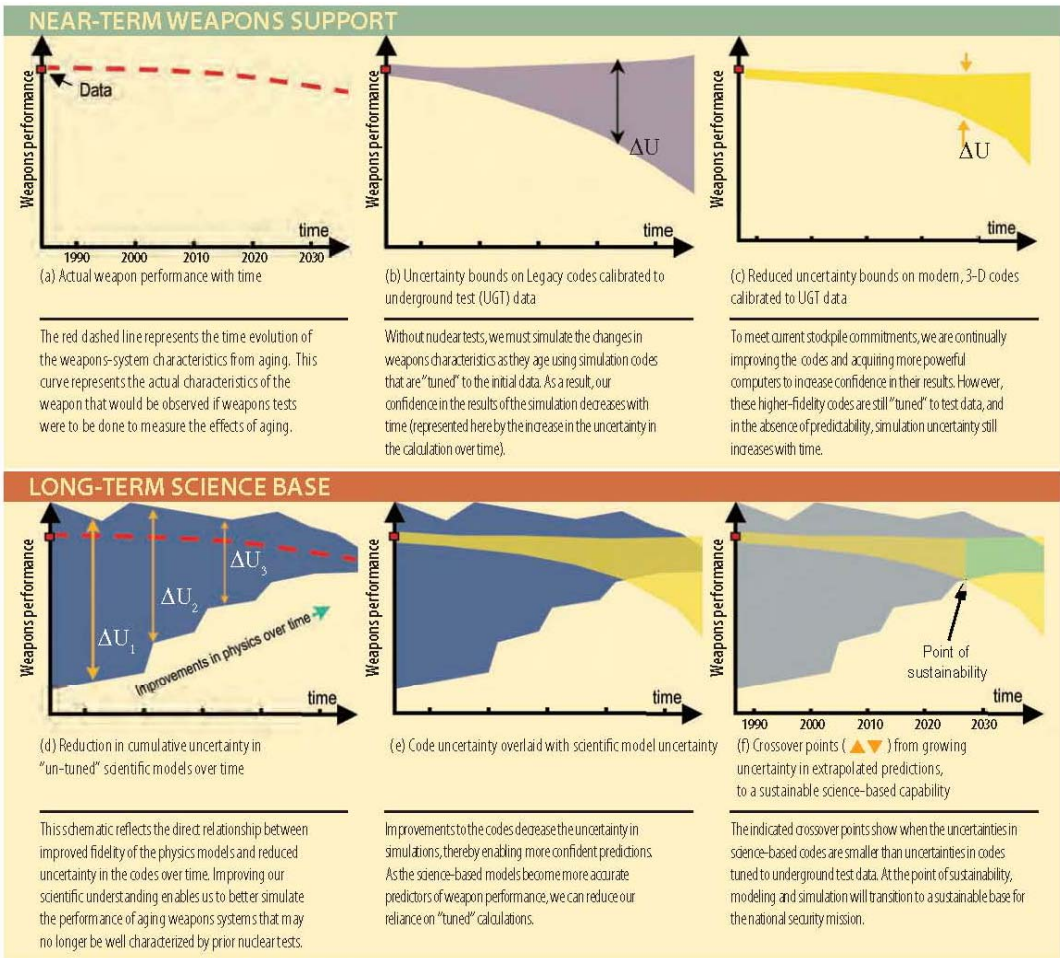


Figure 1-2: Near-term weapons support and long-term science base.

The ASC Program has aggressively pushed computational capabilities and enhanced simulation tools to meet the needs of the SSP in the near term,. Code developers and designers have used test data to calibrate models to build effective computer representations that probe scenarios at and near the area of test experience. The process of calibration allowed for credible interpolation between different nuclear tests and for small extrapolations to untested conditions. However, this same process conceals the unknown science issues through possibly compensating errors in various approximations that mask reality.

There are several clear advantages to the replacement of calibrated models with credible scientific models:

Improved confidence in ASC Program predictions over time.

Confirmation, rather than calibration of ASC Program simulation predictions through existing nuclear test data.

Creation of a robust, responsive, and versatile simulation tool that provides uncertainty bounds with predictions.

The ASC Program has, in fewer than ten years, produced results that may well make it the most successful high-performance computing program in U.S. history. Three of the top ten systems on the June 2007 Top 500[4] list of the world's fastest computers are the ASC BlueGene/L and ASC Purple at Lawrence Livermore National Laboratory, and ASC RedStorm at Sandia National Laboratories. These systems have been instrumental in first-time, 3D simulations involving components of a nuclear weapon during an explosion. Such accomplishments are based on the successes of other elements of ASC Program research, such as scalable algorithms, programming techniques for thousands of processors, and unparalleled visualization capabilities. This history offers confidence that the challenging goals and objectives facing the ASC Program can be achieved.

As an integral and vital element of the SSP, the ASC Program provides the integrating simulation and modeling capabilities and technologies needed to combine new and old experimental data, past nuclear test data, and past design and engineering experience into a powerful tool for future design assessment and certification of nuclear weapons and their components. ASC Program capabilities are needed to model prior manufacturing processes for weapon components and define new, cost-effective, safe, and environmentally compliant manufacturing processes that will provide for consistent nuclear weapon performance, safety, and reliability in the future.

The simulation and modeling tools have already made impacts on the assessment of stockpile issues. Weapon designers, scientists, and engineers are applying ASC Program simulation and modeling capabilities and technologies to assess changes occurring in stockpile nuclear weapons due to natural aging and introduction of modifications.

The recent ASC Roadmap has provided the programmatic justification for petascale and later exascale computing requirements. The ASC Platform Roadmap responded to these programmatic drivers with a platforms roadmap that tasks the ASC Program to delivered petascale computational requirements. The present Sequoia and Dawn systems procurement is intended to deliver on this roadmap, subject to ASC Program and budgetary constraints.

As part of the ASC Roadmap, the ASC Program developed, in conjunction with the overall SSP, a set of eight High Level (Level 1) milestones (Table 1-1) for the FY07 through FY20 timeframe. These milestones are reportable to the U.S. Congress to demonstrate progress towards predictive simulation and support of the overall NNSA 2030 transition strategy. ASC Sequoia, and the Dawn initial delivery system, will be the Production computing engine used by the program to deliver on these milestones during the lifetime of these systems.

[4] http://top500.org/lists/2007/06

ASC Level 1 Milestone and Title	Responsibility	End Date	Program Stakeholders
1. Develop a 100 teraFLOP/s platform environment supporting Tri-Lab Directed Stockpile Work (DSW) and Campaign simulation requirements.	HQ, LLNL	FY07 Q1	C11
2. Develop, implement, and apply a suite of physics-based models and high-fidelity databases to enable predictive simulation of the initial conditions for secondary performance.	HQ, LLNL, LANL, SNL	FY09 Q4	C11, C4
2a: Develop, implement, and validate a suite of physics-based models and high-fidelity databases in support of Full Operational Capability in DTRA's National Technical Nuclear Forensics program.	HQ, LLNL, LANL	FY09 Q4	C11, C1, C4, NA-22, DTRA
3. Baseline demonstration of UQ aggregation methodology for full-system weapon performance prediction	HQ, LLNL, LANL	FY10 Q4	C11, C1, C4, DSW
4. Develop, implement, and apply a suite of physics-based models and high-fidelity databases to enable predictive simulation of the initial conditions for primary boost.	HQ, LLNL, LANL	FY12 Q4	C11, C1, C2
5. Capabilities for SFI response improvements	HQ, LLNL, LANL, SNL	FY13 Q4	C11, DSW
6. Develop, implement, and apply a suite of physics-based models and high-fidelity databases to enable predictive simulation of primary boost	HQ, LLNL, LANL, SNL	FY15 Q4	C11, C1, C2, C10
7. Develop predictive capability for full-system integrated weapon safety assessment	HQ, LLNL, LANL, SNL	FY16 Q4	C11, C1, C2, DSW
8. Develop, implement, and apply a suite of physics-based models and high-fidelity databases to enable predictive simulation of secondary performance	HQ, LLNL, LANL	FY20 Q4	C11, C4, C2, C10

Table 1-1: Proposed ASC Level 1 Milestone List from ASC FY07 Program Plan.

1.3 ASC Applications Overview

ASC Program applications codes perform complex time-dependent two- and three-dimensional simulations of multiple physical processes, where often the processes are tightly coupled and will require physics models linking micro-scale phenomena to macroscopic system behavior. These simulations are divided into two broad categories; integrated design codes (IDC) containing multiple physics simulation packages, and science codes that are mostly single physics process simulation codes. In Figure 1-3, IDC codes are used in the two rightmost regimes, and science codes in the two leftmost regimes.

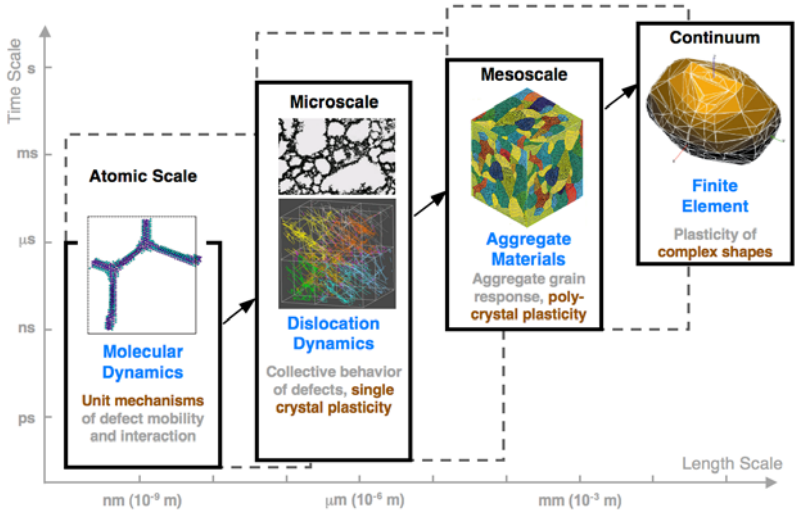


Figure 1-3: Time and space scales for ASC Science Codes (predominately in the Atomic Scale and Microscale regimes) and Integrated Design Codes (predominately in the Mesoscale and Continuum regimes).

The term integrated design codes designates a general category of codes that simulate complex systems where a number of physical processes occur simultaneously and interact with one another. Examples of IDCs include codes that simulate inertial confinement fusion (ICF) laser targets, codes that simulate conventional explosives, and codes that simulate nuclear weapons. ICF codes include packages that model laser deposition, shock hydrodynamics, radiation and particle transport, and thermonuclear burn. Conventional explosives codes include modeling of high explosives chemistry and shock hydrodynamics. All that can be described of the physics modeled in nuclear weapons codes in an unclassified setting is it may include hydrodynamics, radiation transport, fission, thermonuclear burn, high explosives burn, and instabilities and mix. In support of stockpile stewardship IDC codes of all these types and others are required to run on ASC platforms.

ASC science codes are used to resolve fundamental scientific uncertainties that limit the accuracy of the IDC codes. These limitations include material properties, such as strength, compressibility, melt temperatures, and phase transitions. Fundamental physical processes of interest addressed by science codes include mix, turbulence, thermonuclear burn, and plasma physics. The collection of science codes model conditions present in a nuclear weapon, but not achievable in a laboratory, as well as conditions present in stockpile stewardship experimental facilities such as NIF and ZR. These facilities allow scientists to validate the science codes in regimes accessible experimentally giving confidence of their validity in nuclear weapons regimes.

In December 2005 a Tri-lab, Level-1 Milestone effort reported the results of an in-depth study of the needs for petascale simulation in support of NNSA programmatic deliverables. Table 1-2 below contains an unclassified summary of simulations needed to support certification for what has now become recognized as a changing stockpile. This table contains both design and science simulations.

Application	Desired run time (days)	PF needed

Nuclear weapon physics simulation A (3D)	14	0.214
1-ns shocked high explosives chemical dynamics	30	1.0
Nuclear weapon physics simulation B (3D)	14	1.24
Nuclear weapon physics simulation C (3D)	14	1.47
Nuclear weapon physics simulation D (3D)	14	2.3
DNS turbulence simulation (near-asymptotic regime)	30	3.0
Model NGT design	7	3.7
Nuclear weapon physics simulation E (3D)	48	10.2
LES turbulence simulation (far asymptotic regime)	365	10.7
Classical MD simulation of Plutonium process	30	20.0

Table 1-2: Petascale computing requirements for simulations in support of the stockpile stewardship program.

Traditionally, IDC simulations have been divided into two *size* classes; capability runs, that use all of the largest available computer systems, and smaller "capacity" runs, that can be performed on commodity Linux clusters, albeit large clusters. NNSA Defense Programs and the ASC Program are now working to make rigorous a methodology of uncertainty quantification (UQ) as a way of strengthening the certification process and directing the efforts to remove calibrated models in the design codes. This methodology relies on running large suites of simulations that establish sensitivities for all physics parameters in the codes. As executed presently, this suite consists of 4,400 separate runs. This has led to a third class of design code runs called the "UQ class", and for the Sequoia / Dawn procurement it has been characterized as "capacity at the ASC Purple capability level". That is, each individual UQ run requires computing resources with a peak of about 100 teraFLOP/s.

To be useful to Tri-Laboratory Stockpile Stewardship weapons designers and code developers, all of these 4,400 "UQ" runs need to be completed in about one month. Once the number of runs is set, and the time period in which they must complete is set, the maximum spatial resolution is fixed for simulations in both 2D and 3D simulations. Achievable today on ASC Purple and BlueGene/L in the 2006-2008 timeframe is standard resolution 2D UQ and high-resolution 2D or standard resolution 3D capability runs. In the 2011-2015 timeframe, 2D UQ studies must be performed at high-resolution and in 3D standard resolution. In addition, 3D capability runs are required at high-resolution, 2D at ultra-high resolution. These drive the requirements of the Sequoia system.

1.3.1 Current IDC Description

IDC codes model multiple types of physics, generally in a single (usually monolithic) application, in a time-evolving manner with direct coupling between all simulated processes. They use a variety of computational methods, often through a separation or "split" of the various physics computations and coupling terms. This process involves doing first one type of physics, then the next, then another, and then repeating this sequence for every time step. Some algorithms are explicit in time while others are fully implicit or semi-implicit and typically involve iterative solvers of some form. Some special wavefront "sweep" algorithms are employed for transport. Each separate type of physics (e.g., hydrodynamics, radiation transport) is typically packaged up as a separate set of routines and maintained by a different set of code

physicists and computer scientists and is called a physics package. A code integration framework, such as Python, is used to integrate these packages into a single application binary and provide consistent, object oriented interfaces and a vast set of support methods and libraries for such things as input parsing, IO, visualization, meshing and domain decomposition.

An example unclassified ICF code, called Kull, that uses this structure and code management paradigm is shown in Figure 1-6. Kull is an unstructured, massively parallel, object-oriented, multi-physics simulation code. It is developed using multiple languages, C++, C, FORTRAN90, and Python, and MPI and OpenMP for parallelism. Extensive wrapping of the C++ infrastructure and physics packages with the SWIG and Pyffle wrapping technologies exposes many of the C++ classes to Python, enabling users to computationally steer their simulations. While the code infrastructure handles most of the code parallelism, users can also access parallel (MPI) operations from Python using the PyMPI extension set.

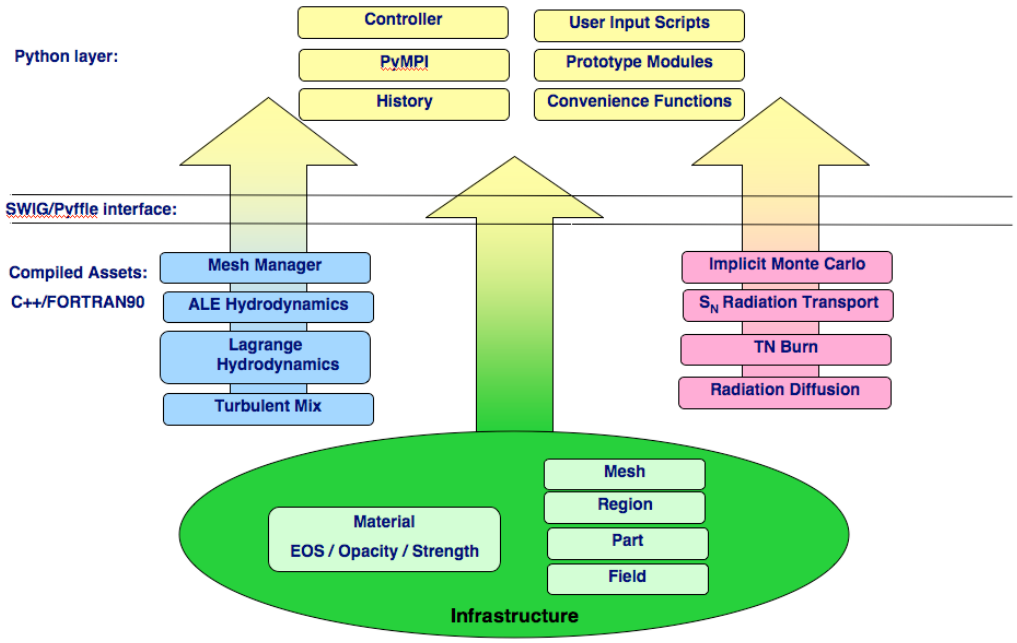


Figure 1-4: Code integration technology and architecture for Kull.

IDC calculations treat millions of spatial zones or cells, with an expected requirement for many applications to use about a billion zones. The equations are typically solved by spatial discretization. Discretization of transport processes over energy and/or angle, in addition, can increase the data space size by 100 to 1,000 times. In the final analysis, thousands of variables are associated with each zone. Monte Carlo algorithms treat millions to billions of particles distributed throughout the problem domain. The parallelization strategy for many codes is based upon decomposition into spatial domains. Some codes use decomposition over angular or energy domains, as well, for some applications.

Currently, almost all codes use the standard Message Passing Interface (MPI) for parallel communication, even between processes running on the same symmetric multi-processor (SMP).

Some applications also utilize OpenMP for SMP parallelism. The efficiency of OpenMP SMP parallelism depends highly on the underlying compiler implementation (i.e., the algorithms are highly sensitive to OpenMP overheads). Also, it is possible in the future that different physics models within the same application might use different communication models. For example, an MPI-only main program may call a module that uses the same number of MPI processes, but also uses threads (either explicitly or through OpenMP). In the ideal system, these models should interoperate as seamlessly as possible. Mixing such models mandates thread-safe MPI libraries. Alternative strategies may involve calling MPI from multiple threads with the expectation of increased parallelism in the communications; such use implies multi-threaded MPI implementations as well.

Because of the memory footprint of the many material property databases used during a run, the amount of memory per MPI process effectively has a lower limit defined by the size of these databases. Although there is some flexibility, IDC codes on ASC Purple strongly prefer to use at least 2 GB per MPI task, and usually more. In most cases, all MPI processes use the same databases and once read in from disk, do not update the databases during a run. A memory saving possibility is to develop a portable method of allowing multiple MPI processes on the same node to read from a single copy of the database in shared memory on that node. For future many-core architectures that do not have 2GB of memory per core, IDC codes will be forced to use threading inside an MPI task in some form. Idling cores is tolerated for occasional urgent needs, but is not acceptable as the primary usage model for Sequoia.

Current codes are based on a single program multiple data (SPMD) approach to parallel computing. However, director/worker constructs are often used. Typically, data are decomposed and distributed across the system and the same execution image is started on all MPI processes and/or threads. Exchanges of remote data occur for the most part at regular points in the execution, and all processes/threads participate (or appear to) in each such exchange. Data are actually exchanged with individual MPI send-receive requests, but the exchange as a whole can be thought of as a "some-to-some" operation with the actual data transfer needs determined from the decomposition. Weak synchronization naturally occurs in this case because of these exchanges, while stronger synchronization occurs because of global operations, such as reductions and broadcasts (e.g., MPI_Allreduce), which are critical parts of iterative methods. It is quite possible that future applications will use functional parallelism, but mostly in conjunction with the SPMD model. Parallel input-output (I/O) and visualization are areas that may use such an approach with functional parallelism at a high level to separate them from the physics simulation, yet maintain the SPMD parallelism within each subset. There is some interest in having visualization tools dynamically attach to running codes and then detach for interactive interrogation of simulation progress. Such mixed approaches are also under consideration for some physics models.

Many applications use unstructured spatial meshes. Even codes with regular structured meshes may have unstructured data if they use cell-by-cell, compressed multi-material storage, or continuous adaptive mesh refinement (AMR). In an unstructured mesh, the neighbor of zone (i) is not zone (i+1), and one must use indirection or data pointers to define connectivity. Indirection has been implemented in several codes through libraries of gather-scatter functions that handle both on-processor as well as remote communication to access that neighbor information. This communication support is currently built on top of MPI and/or shared memory. These scatter-gather libraries are two-phased for efficiency. In phase one, the gather-

scatter pattern is presented and all local memory and remote memory and communication structures are initialized. Then in phase two, the actual requests for data are made, usually many, many times. Thus, the patterns are extensively reused. Also, several patterns will coexist simultaneously during a timestep for various data. Techniques like AMR and reconnecting meshes can lead to pattern changes at fixed points in time, possibly every cycle or maybe only after several cycles.

Memory for arrays and/or data structures is typically allocated dynamically, avoiding the need to recompile with changed parameters for each simulation size. This allocation requires compilers, debuggers, and other tools that recognize and support such features as dynamic arrays and data structures, as well as memory allocation intrinsics and pointers in the various languages.

Many of the physics modules will have low compute–communications ratios. It is not always possible to hide latency through non-blocking asynchronous communication, as the data are usually needed to proceed with the calculation. Thus, a low-latency communications system is crucial.

Many of the physics models are memory intensive, and will perform only about one 64b FLOP per load from memory. Thus, performance of the memory sub-system is crucial, as are compilers that optimize cache blocking, loop unrolling, loop nest analysis, etc. Many codes have loops over all points in an entire spatial decomposition domain. This coding style is preferred by many for ease of implementation and readability of the physics and algorithms. Although recognized as problematic, effective automatic optimization is preferred, where possible.

The multiple physics models embedded in a large application as packages may have dramatically varying communication characteristics, i.e., one model may be bandwidth-sensitive, while another may be latency-sensitive. Even the communications characteristics of a single physics model may vary greatly during the course of a calculation as the spatial mesh evolves or different physical regimes are reached and the modeling requirements change. In the ideal system, the communications system should handle this disparity without requiring user tuning or intervention.

Although static domain decomposition is used for load balancing as much as possible, dynamic load balancing, in which the work is moved from one processor to another, is definitely also needed. One obvious example is for AMR codes, where additional cells may be added or removed during the execution wherever necessary in the mesh. It is also expected that different physical processes will be regionally constrained and, as such, will lead to load imbalances that can change with time as different processes become “active” or more difficult to model. Any such dynamic load balancing is expected to be accomplished through associated data migration explicitly done by the application itself. This re-balancing might occur inside a time step, every few timesteps, or infrequently, depending on the nature of the problem being run. In the future, code execution may also spawn and/or delete processes to account for the increase and/or decrease in the total amount of work the code is doing at that time.

1.3.2 Petascale Applications Predictivity Improvement Strategy

Until recently, supercomputer system performance improvements were achieved by a combination of faster processors and gradually increasing processor counts. Now processor clock speed is effectively capped by power constraints. All processor vendors are increasing performance of successive generations of processors by adding cores and threads geometrically with time according to Moore’s Law and only incremental improvements in clock rate. Thus, to

sustain the 12x improvement over ASC Purple on IDC and 20x improvement over BlueGene/L on Science Codes in 2011-2015, millions of processor cores/threads (i.e., cores *or* threads) will be needed, regardless of the processor technology. Few existing codes will easily scale to this regime, so major code development efforts will be needed to achieve the requisite scaling, regardless of the base processor technology selected. In addition, more is required than just porting and scaling up the codes.

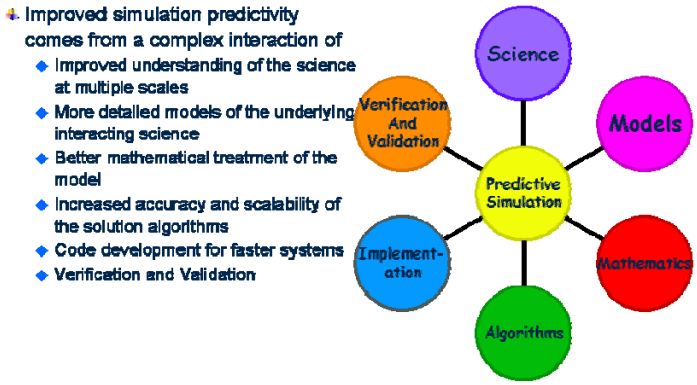


Figure 1-7: In order to improve the simulation predictivity, ASC petascale code development strategy includes improving all aspects of the simulation.

Typically codes scale up utilizing weak scaling by keeping the amount of work per MPI task roughly the same and adding more MPI tasks. To do this, the grid is refined or more atoms are added, or more Monte Carlo particles are added, etc. However, to obtain more predictive and hence useful scientific and engineering results (the difference between busyness and progress), the scientific and engineering capability itself must be scaled up. Increasing the scientific and engineering capability requires improved physical models that remove phenomenologically based interpolative models, as opposed to models based on the actual underlying physics or chemistry. For example, going from *ad hoc* burn models to chemical kinetics models for high explosive detonation. The physical models must be improved with more accurate mathematical abstractions and approximations. The solution algorithms must be improved to increase accuracy and scaling of the resulting techniques. In addition, higher accuracy in material properties (e.g., equation of state, opacities, material cross-sections, strength of materials under normal and abnormal pressure and temperature regimes) are essential. As solution algorithms are developed for the mathematical representations of the physical models, higher resolution spatial and temporal grids are required. The input data sets must increase in resolution (more data points) and the accuracy of the input data for measured data must increase. The physical implementation or code must accurately reflect the mathematics and scalable solution algorithms, mapped onto the target programming model.

Each of these predictive simulation capability improvements require greater computing capability and combined demand petascale computing for the next level of scientific advancement. Improvements in each of these areas requires substantial efforts. For example, better sub-grid turbulence models for general hydrodynamics codes are required for improved prediction of fluid flows. However, these sub-grid turbulence models can only be developed by better understanding of and physical models for the underlying turbulence mechanisms. Better

understanding of turbulence hydrodynamics requires petascale computing. In addition, developing these improved models, verification and validation of the models, algorithms and codes requires similar levels of computational capability. A supercomputer with the target sustained rate of Sequoia will dramatically improve the fidelity of the simulated results and lead to both quantitative and qualitative improvements in understanding. This will again revolutionize science and engineering in the Stockpile Stewardship and ASC communities.

ASC Program's actual experience with transitioning multiple gigascale simulation capabilities to 100's of teraFLOP/s scale suggests that getting ASC IDC and science codes to the petascale regime will be just as hard as building and deploying a petascale computer. To make this problem more acute, some portion of the ASC IDC scientific capability must be deployed commensurate with the petascale platform. This is true, no matter what petascale platform is chosen. Obviously some platform architectures will make this effort more or less problematic. The ASC Program strategy includes three key elements to solve this extremely hard problem: 1.) pick a platform that makes the code scalability more tractable; 2.) take multiple steps to get there; and 3.) tightly couple the ecosystem component development efforts so that they learn from one another and progress together.

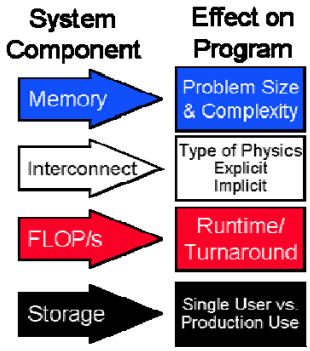
The ASC Program petascale applications strategy includes two significant steps for increasing the ASC IDC and science codes simulation capability.

Increase the node count and node memory on the existing BlueGene/L at Livermore. This enhanced BG/L system can immediately be used to incentivize ASC IDC and science codes research and development efforts to start ramping up their simulation efforts in 2008 rather than in 2010 or later.

A sizable prototype scalable system will be deployed two years before the petascale system. Called Dawn, the prototype bridges the gap (on a log scale) between the BG/L systems and Sequoia. Dawn will provide substantial capability to ASC Program and Stockpile Stewardship researchers to evaluate new models and improve other required simulation components.

Thus, a close collaboration with the selected Offeror will be required during the build of Sequoia and during the deployment of Dawn and Sequoia. At every step, staff and researchers will be supported to transform existing applications and develop new ones that scale to the petascale regime.

1.3.3 Code Development Strategy



The prospect of scaling codes, with improved scientific models, databases, input data sets and grids, to O(1M) way parallelism is a daunting task, even for an organization with successful scaling experience up to 131,072 way parallelism with BlueGene/L. The fundamental issue here is how to deal with the geometric increase in cores/threads on a processor within the lifetime of Sequoia. Simply scaling up the current practice of one MPI task per core, as described in Section 1.3.1, has serious known difficulties.

These difficulties are summarized by the fact that obtaining reasonable code scaling to O(1M) MPI tasks will require that the serial work in all physics packages in an IDC be reduced to 1 in O(1M). Given that code

development tools will not have the resolution to differentiate the 1 in O(1M) differences in subroutine execution times, let alone the problem of workload balancing to that level, this leads one to consider that scaling to this number of MPI tasks may be an insurmountable obstacle. These considerations among others leads one to consider using multiple cores/tasks per MPI task.

The ASC codes require at least 1GB per MPI task (not per core, not per node) and would significantly benefit from 2GB per MPI task. This is **a *critical*** platform attribute. An application mapping of one MPI task per core would lead to a platform with aggregate memory requirement on the order of 1-2PB, which is not affordable. It is also not practical (due to MTBAF and power considerations) in the 2010-2011 timeframe. This also leads one to consider using multiple cores/threads per MPI task.

If one considers the second ***critical*** system attribute for ASC codes, the ASC Program requires >2 million messages per second per MPI task. Again mapping one MPI task per core onto a multicore processor per socket and one or more sockets per node with each node having one or multiple interconnect interfaces, the resulting interconnect requirements make the overall system either too expensive or too specialized to be general purpose or too high risk or a combination of all three. This again leads one to consider using multiple cores/threads per MPI task.

By considering using a reasonable amount of cores/threads per MPI task (i.e., SMP parallelism within the MPI node code), one has effectively divided an impossible problem (scaling to O(1M) MPI tasks) into one that is doable (scaling to O(50-200K) MPI tasks) and another one that is just hard (adding effective SMP parallelism to the MPI node code). Thus, the ASC Program is starting to focus its efforts within the Tri-Laboratory community on scaling the IDC and science codes to O(50-200K) way MPI parallelism now with an extension to the BlueGene/L platform with more memory.

In addition, the ASC Program understands that multiple researchers in industry are working on novel techniques to conquer SMP parallelism (e.g., Transactional Memory and Speculative Execution) for desktop applications in order to enable compelling applications for mainstream Windows and Linux desktop and laptop users. The ASC Program intends to ride this industry trend with a close collaboration with the selected Offeror.

However, ASC Program codes must remain ubiquitously portable, which means any innovation on back end and hardware technology for solving the concurrency problem must have open runtime and operating interfaces and be comprised of incremental changes in the existing C, C++ and Fortran standard language specifications.

1.4 ASC Software Development Environment

The following provides some of the major characteristics of the software development environment for Sequoia in an ideal scenario.

A high degree of code portability and longevity is a major objective. ASC codes must execute at all three ASC sites located at Lawrence Livermore National Laboratory, Sandia National Laboratories and Los Alamos National Laboratory. Development, testing and validation of 3D, full-physics, full system applications requires four to six years. The productive lifespan of these codes is at least ten years. Thus these applications must span not only today's architectures but any possible future system. Codes will be developed in standards-conforming languages so non-standard compiler features are of little interest unless they can be made transparent. The use of Cray Pointers in

Fortran is an exception to our reliance on standard features. It is also highly desirable that C++ compilers accept syntax conventions as implemented in the Gnu C++ compiler. The ASC Program also will not take advantage of any idiosyncratic features of optimization, unless they can be hidden from the codes (e.g., in a standard library). Non-standard "hand tuning" of codes for specific platforms is antithetical to this concept.

A high-performance, low-latency MPI environment that is robust and highly scalable is crucial to the ASC Program. Today applications are utilizing all the features of MPI 1.2 functionality. Many features of MPI-2 functionality are also in current use. Hence, a full, robust and efficient implementation of MPI-2, except for dynamic tasking, including a fully operational message queue debug interface, is of tremendous interest. To execute the petascale code development strategy described in section 1.3.3, we require robust and flexible multi-core/thread node programming environments that allow programmers to construct MPI parallel applications with a unified nested node concurrency model. In this "Livermore Model" a single MPI parallel application is made of multiple packages, each with potentially different node parallelism style within the MPI tasks. Since packages may call each other (from the master thread), these node parallelism styles must nest and allow helper threads to be repurposed very efficiently. At a minimum a POSIX compliant-thread environment is also crucial and a Fortran03-threads interface is also important. All libraries must be thread-safe. In addition, a low overhead implementation of OpenMP style parallelism should be implemented in baseline languages. The ASC Program needs close collaboration with the selected Offeror to develop incremental advances in the baseline languages compilers for Sequoia that would take advantage of any leading edge concurrency hardware enablement such as Transactional Memory (TM) and Speculative Execution (SE). MPI should be thread-safe with the MPI Init thread function able to set the thread support level (e.g., MPI THREAD SINGLE, MPI THREAD FUNNELED and MPI_THREAD_MULTIPLE). The ASC Program should not have to tune the MPI-runtime environment for different codes or different problem sizes and different MPI task counts. In ASC's estimation, there are several basic MPI characteristics that must be optimized. Link bandwidth as a function of MPI task count per multi-core/thread node and link ping-pong latency are obvious ones. In addition, messages processed per second per MPI task (adapter messaging rate) needs to be large and grow as the number of MPI tasks per node increases. Further, the real world delivered MPI bisection bandwidth of the machine should be a large fraction of the peak bisection bandwidth and collectives (MPI_Barrier, MPI_Allreduce, MPI_Broadcast) should be fast and scalable. As a node parallelism exploitation risk reduction fall back strategy, the ASC Program must be able to run applications with one MPI task per core over significant portions of Sequoia. Since this involves systems with millions of cores/threads, it is vitally important that the MPI implementation scale to the full size of the system, and that sub-communicators within MPI support efficient use of available hardware capabilities in the system. This scaling is both in terms of efficiency (particularly of the MPI_Allreduce functionality) as well as the efficient use of buffer memory. ASC applications are carefully programmed so that MPI receive operations are usually posted before the corresponding send operation, which allows for minimal (and hence scalable) MPI buffer space allocations.

ASC applications require the ability for each MPI task to access all physical memory on a node. The large memory sizes of MPI tasks requires that all of our applications are completely 64b by default.

The ASC Program expects the compilers to do the vast majority of code optimization through simple easy-to-use compiler switches (e.g. -On) and compiler directives and possible language extensions for exploitation of leading edge concurrency hardware support (e.g., TM and SE). Also, the ASC

Program expects the compilers to have options to range check arrays and under debug mode detect silent NaN's, and to trap all floating point exceptions including underflow, overflow and divide by zero.

Parallelization through the OpenMP constructs is of particular interest and is expected for the baseline languages. OpenMP parallelization must function correctly in programs that also use MPI. OpenMP Version 2.5 support for Fortran03 and C/C++ is required while OpenMP 3.0 support is highly desired in the time frame of Dawn and required for Sequoia. OpenMP performance will be critical for effective use of the Sequoia system. It is desirable that OpenMP barrier performance be 200 clock cycles or better, and that overhead for OpenMP parallel FOR/DO be 500 cycles or less in the case of static scheduling with NCORE OpenMP threads. Automatic parallelization is of great interest, if it is efficient, utilizes advanced concurrency hardware (e.g., TM and SE) and does not drive compile times to unreasonable lengths and yields binaries over a wide range of ASC applications and problems sizes that actually run faster when utilizing this form of parallelization. Detailed diagnostic information the compiler can provide about the optimizations performed is essential. Compiler parallelism has to work in conjunction with MPI. All compilers must be fully ANSI-compliant.

The availability of standard, platform-independent tools is necessary for a portable and powerful development environment. Examples of these tools are GNU software (especially the GNU build system with transparent configuration support for cross-compilation environment, but others, such as binutils, libunwind and gprof as well), the TotalView debugger (the current debugger on all ASC Program platforms), dependency builders (Fortran USE & INCLUDE as well as #include), preprocessors (CPP, M4), source analyzers (lint, flint, etc), hardware counter libraries (PAPI), communications profilers (mpiP, OpenTraceFormat writers, and the VAMPIR trace viewer), and performance analysis toolsets (Open|SpeedShop). Tools that work with source code should fully support the most current language standards. Standard APIs to give debuggers and performance analyzers access to the state of a running code would allow the ASC Program to develop its own tools and/or to use a variety of tools developed by others. The MPIR automatic process acquisition interface (based on the interface described at http://www-unix.mcs.anl.gov/mpi/mpi-debug/) with tool daemon bulk launch support is a well-established public domain API that meets portions of this need; the process control interfaces like the /proc interface and ptrace are another; MRNet (the Multicast Reduction Network), the StackWalker API, the Dynamic Probe Class Library (DPCL) and Dyninst are public domain APIs that meet still others. These performance and debugging tools must not require privileged access modes for installation or execution, such as root user, nor compromise the security of the runtime environment. Documentation for tools and APIs must be fully installed on the delivered machine without recourse to an internet connection.

The ASC Program must have parallel symbolic debuggers that allow debugging of parallel applications within a node and that permit large, complex application debugging of parallel applications utilizing multiple nodes. This includes MPI-only as well as mixed MPI + explicit threads and/or OpenMP codes. Some of the ASC Program applications have a huge number of symbols and a large amount of code and may run with 100K to 1M MP tasks, so application job launch under control of the debugger is a major scalablity issue that must be solved for the Sequoia system. In the best of all possible worlds, the debugger would allow effective debugging of jobs using every core/thread on the system. Practical use of a large fraction of the machine by an application under the control of the debugger requires that the debugger be highly scalable and

integrated into the system initiated parallel checkpoint/restart. Some specific features of interest follow.

- breakpoints, and barriers and watchpoints with compiled expression system
- fast conditional breakpoints,
- fast conditional watchpoints on memory locations,
- single-stepping at various control levels,
- a save-restore state for backing up via checkpoint/restart mechanism,
- complex selections for data display including user-programmable GUI data display,
- support for array statistics (min, max, etc),
- attaching/detaching to all or a subset of the processes in starting or running jobs,
- support for MPI-2 dynamic tasking,
- an effective user-defined process group capability,
- an initialization file that knows where the sources are and what options we want etc., and
- a command-line interface in addition to a GUI (e.g. for script driven debugging).
- LD_PRELOAD-based memory debugging,
- the ability to display all kinds of Fortran descriptor-based data,
- the ability to display OpenMP THREADPRIVATE common data,
- the ability to display a C++ vector< vector<T> > in 2D array format,
- the ability to show/hide elements of aggregate objects,
- automatic display of important variables, e.g., those on the current line, or a user-defined set per function.
- changed values highlighted with color,
- important-variable timestamped trace and replay,
- exclusion of non-rank processes from view and interference,
- sufficient debugger status feedback to give the user positive control continuously,
- convenient MPMD debugging,
- a facility for relative debugging,
- a facility to record debugger commands for later automating reruns,

The capability to examine slices and subsets of multidimensional arrays visually is a feature that has proven useful. The debugger should allow complex selections for data display to be expressible with Fortran03 and C language constructs and features. It should support applications written in a mixture of the baseline languages (Python, Fortran03, C and C++), support Cray-style pointers in Fortran77, and be able to resolve C++ templated symbols and perform complex template evaluation in C++. It should be able to debug compiler-optimized codes since problems sometimes disappear

with non-optimized codes, although progressively less symbolic and contextual information will be available to the debugger at higher levels of optimization. The ASC Program build environment involves accessing source code from NFS and/or NFSv4 mounted file systems with likely compiling and linking of the executable in alternate directories. This process may have implications, depending on how the compiler tells the debugger to find the source code. To meet the challenges of petascale debugging that involves O(1M) threads of control, it is crucial for key debugging features to be scalable. For example, the performance of subset debugging must scale according to the number of processes in the control subset, not the number of processes in the target job. The debugger currently used in the Tri-Laboratory ASC applications development environment is the TotalView debugger from TotalView Technologies, LLC. (see URL: http://www.totalviewtech.com/index.htm) . This debugger requires that the O.S. provide a POSIX 1003.1-2004-compliant kill -s KILL system call.

Many ASC Program applications use Python for package integration within a single application binary; to provide a convenient input dataset syntax; implement data object abstraction and extensibility and enable runtime application steering. Thus, it is essential that the system includes support for running Python based applications. This support includes, but is not limited to, dynamic linking and loading. The debugger must also support these features so as to allow efficient debugging of the entire application.

Because most ASC Program codes are memory-access intensive, optimizing the spatial and temporal locality of memory accesses is crucial for all levels of the memory hierarchy. To tune memory distribution in a NUMA machine, it is necessary to be able to specify where memory is allocated. To use memory optimally and to reuse data in cache, it is also necessary to cause threads to execute on CPUs that quickly access particular NUMA regions and particular caches. Expressing such affinities should be an unprivileged operation. Threads generated by a parallelizing compiler (OpenMP or otherwise) should be aware of memory-thread affinity issues as well.

Other ramifications of the large memory footprint of ASC Program codes is that they require full 64b support in **all** supplied hardware and software. This includes the seamless ability to specify through the compiler that all unmodified integer declarations are 64 bit quantities. In addition, because these memory-access intensive codes have random memory access patterns (due to indirect addressing or complex C++ structure and method dereferencing brought about from implementing discretization of spatial variables on block structured or unstructured grids) and hence access thousands to millions of standard UNIX™ 4KiB VM pages every timestep, “large page support” in the operating system for efficient utilization of the microprocessor virtual to real memory translation functionality and caches is required for efficient use of the hardware. This is due to the fact that hardware TLBs have a limited number of entries (although caching additional entries in L1 cache helps but does not solve the problem) and having, say, 2GiB page size would significantly reduce the number of TLB entries required for large memory-access ASC code VM to real memory translations. Since TLB misses (that are not cached in L1) are very expensive, this feature can significantly enhance ASC application performance.

Many of the ASC Program codes could benefit from a high-performance, standards-conforming, parallel I/O library, such as MPI-I/O. In addition, low latency GET/PUT operations for transmission of single cache lines is viewed as essential for domain overloading on a single SMP or node. However, many implementations of the MPI-2 MPI_Get/MPI_Put mechanisms do not have lower latency than MPI_Send/MPI_Recv, but do allow for multiple outstanding MPI_Get/MPI_Put operations to be active at a time. This approach, although appealing to MPI-2 library developers,

puts the onus of latency hiding on the applications developer, who would rather think about physics issues. Future ASC applications require a very low latency (as close to the SMP memory copy hardware latency as possible) for GET/PUT operations.

Effectively tuning an application's performance requires detailed information on its timing and computation activities. On a node, a timer should be provided that is consistent between threads or tasks running on different cores/threads in that same node. The timer should be high-resolution (10 microseconds or better) and low overhead to call. In addition, other hardware performance monitoring information such as the number of cache misses, TLB misses and floating-point operations, can be very helpful. All modern microprocessors contain hardware counters that gather this kind of information. Additionally, network performance counters should be accessible to the user. The data in these counters should be made available separately for each thread or process (as selected by the user) through tools or programming libraries accessible to the user. For portability, ASC Program tools are targeting the PAPI library for hardware counters (http://icl.cs.utk.edu/projects/papi/). To limit instrumentation overhead, the potential Offerors should provide a version of their tools that support sampling and multiplexing of hardware counters, and sampling of instructions in the pipeline. Note that this facility requires that the operating system context switch these counters at process or heavy weight (OS scheduled) thread level and that the POSIX or OpenMP runtime libraries context switch the counters on light-weight (library scheduled) thread level. Furthermore, these counters must be available to users that do not have privileged access, such as the root user. Per-thread OS statistics must be available to all users via a command line utility as well as a system call. One example of such a feature is the `kstat` facility: a general-purpose mechanism for providing kernel statistics to users. Both hardware and counter statistics must provide virtualized information, so that users can make the correct attribution of performance data to application behaviors.

The ASC Program needs early access to new versions of system and development software, as well as other supplied software. Software releases of the various products should be synchronized with operating system releases to ensure compatibility and interoperability. Documentation needs to be provided in a timely manner, and documentation of system API's needed to support OpenSource and OpenAPI tools such as Valgrind must be provided.

Code development will be done directly on Dawn and Sequoia. This means that it must be possible to configure, compile, build, load and execute large scale applications on a portion of the machine (front-end) and cross compile effectively and transparently to the set of nodes that run parallel applications (back-end). A key component of this code development environment is the ability to run AUTOCONF where the applications are compiled, but transparently target the back-end that will actually run the parallel application. That is, ASC Program code developers want to be able to configure the large scale ASC applications build process with AUTOCONF and cross configure and build applications on the front-end to execute on the back-end. Careful attention must be paid to any operating system and/or processor hardware difference between the nodes where the AUTOCONF and compilations are performed (front-end) and where the application is run (back-end). This difference in front-end/back-end hardware and software environments should be as transparent to the applications developers as possible (e.g., handled via AUTOCONF configuration or compiler options).

1.5 ASC Applications Execution Environment

The following provides some major characteristics of the ASC Program ultra-scale applications execution environment.

It is crucial to be able to run a single parallel job on the full system using all resources available for a week or more at a time. This is called a "full-system" or "capability" run. Any form of interruption should be minimized. The capability for the system and application to "fail gracefully" and then recover quickly and easily is an extremely important issue for such calculations. The ASC Program expects to run a large number of jobs on thousands to hundreds of thousands of nodes each for hundreds of hours. These would require significant system resources, but not the entire system. The capability of the system to "fail gracefully," so that a failure in one section of the system would only affect jobs running on that specific section, is important. From the applications perspective, the probability of failure should be proportional to the fraction of the system utilized. A failed section should be repairable without bringing down the entire system.

A single simulation may run over a period of months as separate restarted jobs in increments of days running on varying numbers of nodes with different physics packages activated. Output and checkpoint files produced by a code on one set of nodes need to be efficiently accessible by another set of processors, or possibly even by a different number of processors, to restart the simulation. Thus an efficient cluster wide file system is essential. Ideally, file input and output between runs should be insensitive to the number of nodes before and after a restart. It should be possible for an application to restart across a larger or smaller number of nodes than originally used, with only a slight difference in performance visible.

ASC applications write many restart and visualization dumps during the course of a run. A single restart dump may be about the same size as the job's memory resident set size, while visualization dumps may be perhaps from 1 to 10 % of that size. Restart dumps would typically be scheduled based on wall clock periods, while visualization dumps are scheduled entirely on the basis of internal physics simulation time. The ASC Program usually creates visualization dumps more frequently than restart dumps. System reliability will have a direct effect on the frequency of restart dumps; the less reliable the system is, the more frequently restart dumps will be made and the more sensitive the ASC Program will be to I /O performance. The ASC Program has observed, on previous generation ASC platforms, that restart dumps comprise over 75% of the data written to disk. Most of this I/O is wasted in the sense that restart dumps are overwritten as the simulation progresses. However, this I/O must be done so that the simulation is not lost to a platform failure. This leads to the notion that cluster wide file system (CWFS) I/O can be segregated into two portions: productive I/O and defensive I/O. Productive I/O is the writing of data that the user needs to do science (visualization dumps, traces of key physics variables over time, etc.). Defensive I/O is done to manage a large simulation run over a period of time much larger than the platform MTBF. Thus, one would like to minimize the amount of resources devoted to defensive I/O and computation lost due to platform failure.

System (hardware and software) failure modes should be clear and unambiguous. Supplied software should detect hardware and system software failures, report the error in a clear and concise manner to user as well as system administrator as appropriate, and recover with minimal to no impact to applications whenever possible.

Operationally, applications teams push the large restart and visualization dumps (already described) off to HPSS tertiary storage within the wall clock time between making these dumps. The disk

space mentioned elsewhere in this document is insufficient to handle ASC applications long-term storage needs. HPSS is the archive storage system of ASC and compatibility with it is needed. Thus, a highly usable mechanism is required for the parallel high speed transport of 100's of TB to 10's of PB of data from the CWFS to HPSS.

The ASC Program plans to use the MOAB job scheduler (www.clusterresources.com/moab) and SLURM (www.llnl.gov/linux/slurm/) resource manager that manages with all aspects of the system's resources, not just nodes and time allocations. It is essential for this resource manager-scheduler to handle both batch and interactive execution of both serial and parallel programs supporting the "Livermore model "of mixed MPI and threaded modes of parallelization in the same binary from a single node to the full system. The MOAB/SLURM manager-scheduler provides a way to implement policies on selecting and executing various problems (problem size, problem run time, timeslots, preemption, users' allocated share of machine, etc). Also, methods are provided for users to connect to executing batch jobs to query or change problem status or parameters. ASC Program codes and users benefit from a robust, globally visible, high-performance, parallel file system called Lustre. It is essential that all Offeror provided hardware and software IO infrastructure allow LLNS provided file systems and software support to support a full 64b address space. A 32b address space is clearly insufficient.

1.6 ASC Sequoia Operations

The Sequoia and Dawn systems should be designed to minimize floor space, power, and cooling requirements.

The ASC Program plans to operate the systems 24 hours per day, 7 days per week, including holidays. The prime shift will be from 8 AM to 5 PM, Pacific Time Zone. LLNL local and remote (e.g., LANL and SNL) users would access the system via the 1 and 10 Gigabit Ethernet local-area network (LAN). For remote users, the Sequoia 1 and 10 Gigabit Ethernet infrastructure will be switched to the DisCom2 wide-area network (WAN) which will be OC-48/ATM/ POS connections.

The prime shift period will be devoted primarily to interactive applications development, interactive visualization, relatively short large core/thread count (e.g., over half the system cores/threads), high priority production runs and extremely long running, routine core/thread count (e.g, 10K-100K), lower priority production runs. Yes that's right, 10K-100K will be routine for Sequoia. Night shifts, as well as the weekend and holiday periods, will be devoted to extremely long-running jobs. Checkpointing and restarting jobs will take place as necessary to schedule this heterogeneous mix of jobs under dynamic load and job priorities on Sequoia. Because the workload is so varied and the demands for compute time oversubscribe the machine by several factors, the checkpoint/restart facility to **dynamically terminate** jobs and save their state to disk on Sequoia and later restart them is an essential production requirement. In addition to system initiated checkpoint/restart, ASC applications have the ability to do application based restart dumps. These interim dumps, as well as visualization output, would be stored on HPSS-based archival systems or sent to the CSSE PPPE visualization corridors via the system-area network (SAN) and external "Jumbo Frame" 1 and 10 Gigabit Ethernet interfaces. Depending upon system protocol support, IP version 4, IP version 6, and lightweight memory-to-memory protocol (e.g., Scheduled Transfer) traffic will be carried in this environment.

Hardware maintenance services may be required around the clock, with two hour response time during the hours of 8:00 a.m. through 5:00 p.m., Monday through Friday (excluding Laboratory

holidays), and less than four hours response time otherwise. The following are holidays currently observed at LLNL:

- New Year's Day
- Martin Luther King, Jr., Day (third Monday in January)
- President's Day (third Monday in February)
- Memorial Day (last Monday in May)
- Fourth of July
- Labor Day
- Thanksgiving Day
- Friday following Thanksgiving Day
- December 24 (or announced equivalent)
- Christmas Day
- December 31 (or announced equivalent)
- One administrative holiday (in March or April; the Monday following Easter)

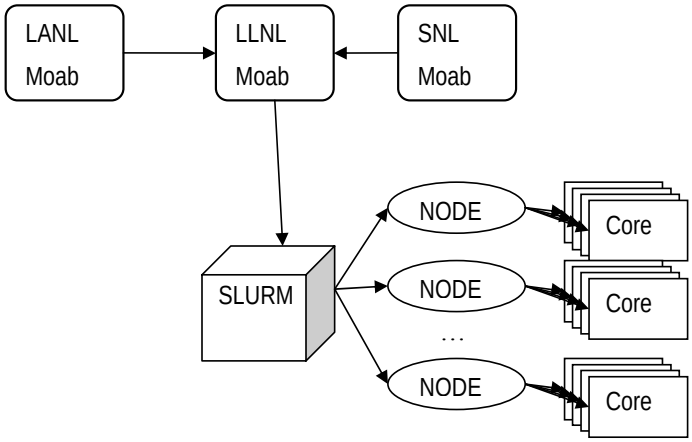
A single point of system administration may allow the configuration of the entire system from a common server. The single server may control all aspects of system administration in aggregate. Examples of system administration functions include modifying configuration files, editing mail lists, software, upgrades and patch (bug fix) installs, kernel parameter changes, file system-disk manipulation, reboots, user account activities (adding, removing, modifying), performance analysis, hardware changes, and network changes. A hardware and software configuration management system that profiles the system hardware and software configuration as a function of time and keeps track of who makes changes is essential.

Due to the large size of Sequoia, it is anticipated that the selected Offeror's System Test facility may not always be able to test software releases and bug fixes at scale. Although it is expected that a rigorous and intelligent testing methodology will always be employed by the selected Offeror prior to delivery of system releases or bug fixes, the final step in scaling and performance testing might, at times, have to be accomplished on Sequoia. Although this use of the system by the selected Offeror should be kept to an absolute minimum, there will be times when new releases and or patches will need to be installed on an interim basis on Sequoia. To this end the ASC Program requires a multi-boot capability on the system so that the known good, production quality software environment is not disrupted by the new releases and or bug fixes and different types of kernels or system configurations can be tested. This multi-boot capability should be sufficient to bring the system to the new software level quickly and return the system to the previous state quickly after the debug shot. This of course engenders a requirement for fast and reliable full system reboot as it does not make sense to most sentient beings to have a four hour debug shot and an eight to sixteen hour period for the minimum of two required system reboots (one to boot the test system and one to boot the production system, assuming each reboot is successful on the first attempt).

The ability to dynamically monitor system functioning in real time and allow system administrators to quickly diagnose hardware, software (e.g., job scheduler) and workload problems and take

corrective action is also essential. Due to the anticipated large size of Sequoia, these monitoring tools must be fast, scalable and display data in a hierarchal schema. The overhead of system monitoring and control will necessarily need to be low in order to not destroy large job scalability (performance).

At the highest level, the workload will be managed by Moab. Moab will control the use of the resources for both interactive and batch usage from a single core/thread to all cores/threads in compute node in the system. Users are organized within programmatic hierarchies that define relative rights to access the resources. The Moab system will distribute resources to groups of users by political priorities in accordance with established allocations and their recent usage. Under the constraints of political and other scheduling priorities, the Moab system must be capable of considering the resource needs and requests of all jobs submitted to it, and of making an intelligent mapping of the job needs to the resources available.



The LLNL supplied SLURM system will be able to manage the various system components that comprise the entire environments, including, but not limited to, development, production, dedicated benchmarking, a mix of single-node jobs, a mix of multi-node parallel jobs, and jobs that use the entire system resource. This capability will be flexible enough to allow a rapid transition from one run-time environment to another. It will be able to configure run-time environments on an automated basis, such as by time and day of week. It will manage this transition in a graceful manner with no loss of jobs during the transition.

Production control of the Moab/SLURM will span the entire system. That is, a job is an object that may be targeted to run on the entire system or a subset of the system. The resource management system will globally recognize a job throughout the system. A job will use 64b MPI libraries to span up to the complete system.

Jobs will be accounted for and managed as a single entity that included all its associated processes and memory. The Moab/SLURM system will be able to dynamically collect and maintain complete information on the status of all the resources under its control at all times, so that the current pool of unused resources is known at all times.

It is anticipated that LLNL will port Moab/SLURM to quickly and reliably launch jobs, shepherd jobs through the system and accurately account for their system resource usage on an interval basis (not end of job accounting). The overhead of job management and accounting will necessarily need to be low in order to not destroy large job scalability (performance).

1.6.1 Sequoia Support Model

The ideal system will have reliability, availability, and serviceability (RAS) features integral to its design up to, and including, the full system. It will support such features as hot-swapping of components, graceful degradation, automatic fail-over, and predictive diagnostics. LLNS will supply level-one hardware and software support. Offeror may need to provide additional field engineering support to provide more comprehensive hardware and software support should the need arise. The diagnostic tools the hardware support team employs will make possible the accurate diagnosis of problems to the field replaceable unit, thereby minimizing time-to-repair and repeated changing of parts hoping against all common sense that the next FRU replacement will be the actual failing unit. A sufficiently large on-site parts cache and hot-spare nodes should be available to the hardware support team so that nodes can be quickly repaired or replaced and brought back on-line. Target typical hardware failure to fix times are as follows: four hour return to production for down nodes or other critical components during the day; and eight hours during off peak periods, is a strong requirement. A problem escalation procedure may be in place and will be invoked when necessary. Hardware and software development personnel will be available for solving particularly difficult problems as a backup to the Offeror field engineering support. There will be a high degree of cooperation among the hardware engineers, field software analysts, LLNS personnel, and third-party suppliers. Engineering problems will be promptly reported to the appropriate engineering staff for immediate correction by an interim hardware release as well as in subsequent normal releases of the hardware. Appropriate testing and burn-in of the system components prior to delivery would also reduce the number of component "dead-on-arrival" and infant mortality problems.

In order to provide adequate support and interface back to the selected Offeror's development and support organization, on-site (i.e., resident at LLNL), Q-cleared personnel are needed. These selected Offeror employees need to be able to remotely use Offeror's web sites and other IT facilities for support, education and communication functions. Ideally, this staff will include one highly capable systems analyst and one highly-capable applications performance and scalability analyst. These staff will be available on-site at LLNL as consultants to, and resident area experts for, both the LLNS Sequoia support staff and Tri-Laboratory end-users.

The systems analyst should be available to support LLNS Sequoia system administration activities. Ideally, this analyst should be a hybrid systems programmer and systems administrator with capability to write and debug OS, drivers, installation scripts, etc. This Q-Cleared staff will provide LLNS the ability to provide Offeror hands-on access to the classified Sequoia system to assist in hardware and software problem root cause analysis activities. Smooth operation of Sequoia and interfacing to Offeror's support organization will depend heavily on this individual.

The applications analyst should be available to support code development activities of the major ASC code projects directly. Ideally, this analyst should have a background in physics, numerical mathematics, or other computational science discipline and possess parallel computing and scientific applications development experience. The analyst should be closely associated with the software development organization and, in a sense, be an extension of the Tri-Lab ASC code development teams. Our experience has been that such analysts are critical to our ability to

make progress on applications codes on complex ASC scale systems. The importance of their role cannot be overemphasized.

1.7 ASC Dawn and Sequoia Simulation Environment

The ASC Program petascale ecosystem components at LLNL where the Sequoia system will be integrated, incorporates a single enterprise wide file system to which multiple computational, visualization and archival resources read and write simulation, visualization and checkpoint/restart data. There are strong incentives for an enterprise-wide file system as it is prohibitive in cost and performance to move and/or copy multi-petabyte file sets that are created in the simulation phase for subsequent processing, as for post processing and visualization. The goal of LLNS with the development of Sequoia is to integrate this system into an existing Secure Computing Facility (SCF) simulation environment based on the Lustre[5] enterprise wide file system. This simulation environment at LLNL will be based on 1 and 10 Gb/s Ethernet and possibly InfiniBand™ technology.

[5] wiki.lustre.org

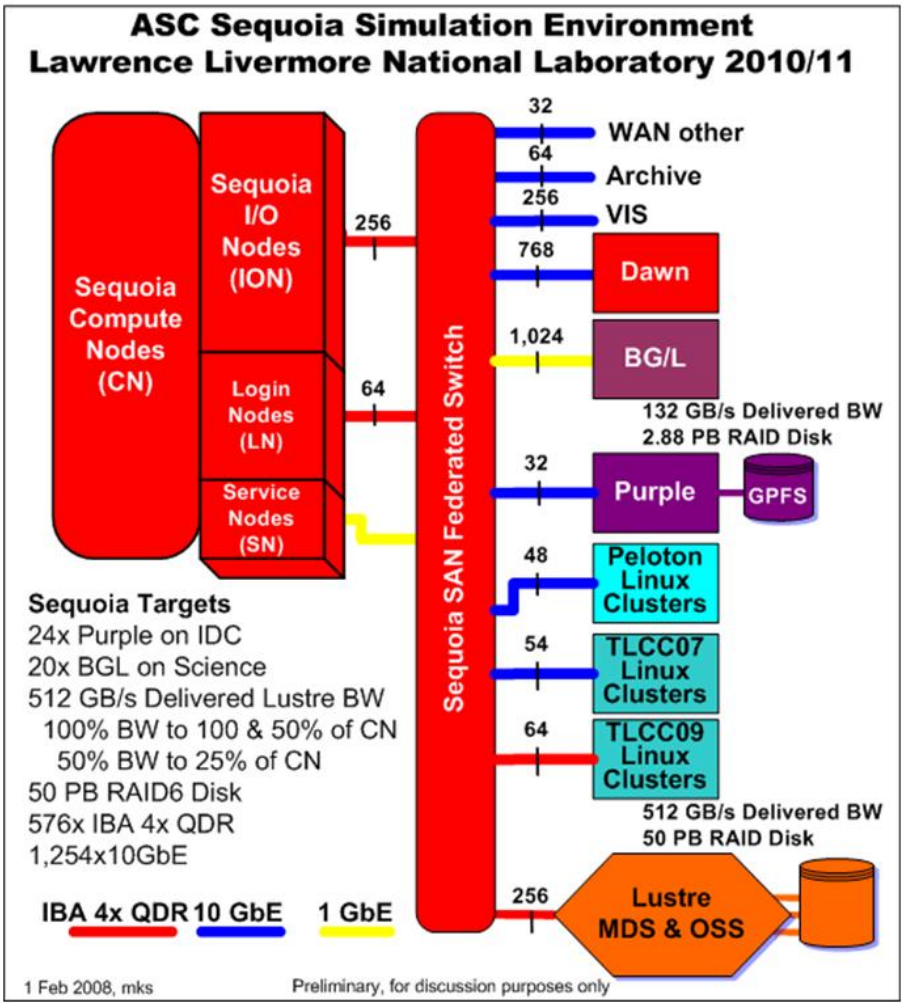


Figure 1-5: The Sequoia simulation environment at LLNL includes access to the Lustre enterprise wide file system, Login Nodes (LN), Service Nodes (SN) and control management network, visualization cluster (VIS), archive and WAN resources.

A schematic of the Sequoia simulation environment at LLNL is depicted in Figure 1-5. This SOW includes the Sequoia back-end of compute nodes (CN) and I/O nodes (ION), the login nodes (LN), the control management network, and the Service Nodes (SN). Other existing and future compute, visualization and storage resources are part of the overall LLNL classified simulation environment. A Lustre based enterprise-wide file system and 1/10 Gigabit Ethernet and possibly IBA federated Storage Area Network (SAN) switch are LLNS furnished government property (LFGP).

In this Sequoia target architecture, CN are a set of nodes that run end-user scalable MPI and SMP parallel scientific simulation applications and are scaled to meet the overall peak petaFLOP/s and delivered application performance metrics in section 2.1. ION provide Lustre IO function shipping

capability and high-bandwidth access to the Lustre based OSS and MDS resource for Lustre parallel file system access to applications running on the CN. The number of ION are scaled to meet the delivered IO performance requirements in sections 2.3 and 2.9.1. In addition, ION provide IP routing from the CN to the SAN. LN provide nodes for users to login (via ssh and associated tools) and interact with the system to perform code development activities, run and interact with interactive jobs and manage (launch, terminate and status) batch jobs. The number of LN are scaled to meet the number of active users and compilations required in section 2.6.1. SN are a set of nodes that provide *all* scalable system administration and RAS functionality. The number of required SN is determined by Offeror's scalable system administration and RAS architecture and the overall size of the system.

The diagram explicitly shows the interconnection by SAN switch of the back-end of Sequoia, the front-end nodes, service node, and the Lustre file system. This configuration provides for the addition of future services via connection to the SAN switch.

The login nodes are the interactive resources on which users login to access Sequoia. Users will edit, configure and compile codes, create job control files, launch jobs on Sequoia, post process output, and perform other interactive activities. System administrators will also utilize the front-end nodes to control and configure Sequoia.

A large federated 1/10 Gigabit-Ethernet and possibly IBA switch is the main communications path from Sequoia to the outside world. This switch is designed to provide high-speed connectivity to the Lustre file system which is the main disk storage for Sequoia. This switch also gives other resources access to the files on the Lustre file system. Interactive users on the front-end nodes will have ready access to the files on Lustre, as well as visualization servers, archive services, and other resources on the SCF network.

A control and management network (CMN), shown in yellow in Figure 1-5, provides system administrators with a separate command and control path to Sequoia. This private network is not available to unprivileged users.

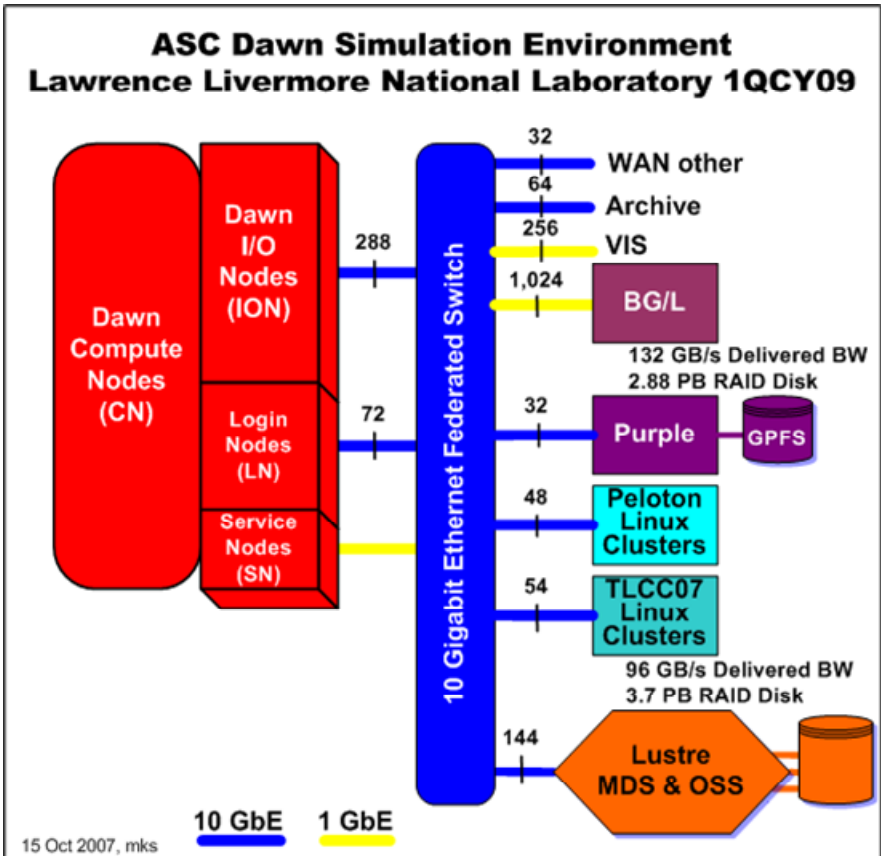


Figure 1-6 ASC Dawn Simulation Environnent.

The ASC Program intends to integrate the Dawn system into the existing SCF 1/10 Gb/s Ethernet federated switch Storage Area Network (SAN) currently in use at LLNL for classified computing, see Figure 1-6. The ASC Program will augment this SAN and Lustre file system with the necessary networking and RAID disk resources to provide an appropriately scaled Lustre file system for Dawn and the other computing resources connected to the SCF simulation environment. Therefore, it is essential that the I/O subsystem for connections for Dawn be based on a SAN technology that can interoperate in this heterogeneous environment. At this time, it appears the leading contenders for this SAN technology are: Infiniband™ 4x QDR, 1000Base-SW and 10GBase-SW.

In addition, the ASC Program expects TCP/IP off-load engines (TOEs) to be available for these competing SAN technologies. These TOEs will allow extremely fast TCP/IP communications that don't burden the cores/threads on the Dawn and Sequoia nodes originating the traffic. Thus the ideal Dawn and Sequoia systems will have outboard (to the IO nodes) TOE devices that interface the SAN to the external networking environment.

External networking I/O to LAN, WAN, and SAN networks in the ideal system would support multiple protocols, perform channel striping, and have sufficient bandwidth to be in balance with the other elements of the system. Depending upon system protocol support, IP version 4 and IP version 6 traffic will be carried on the LAN and WAN. These circuits will support either IP over 1000Base-SW or 10 Gb/s Ethernet.

The operating environment shall conform to DOE security requirements. Software modifications must be made in a timely manner to meet changes to those security requirements.

1.8 Sequoia Timescale and High Level Deliverables

Building and delivering a petascale computing resource of this scale is a daunting task. The successful completion of this project will require close collaboration between the selected Offeror and LLNS. It requires careful planning and coordination of these efforts within the selected Offeror and LLNS partnership. To this end, LLNS anticipates that the project will take on several critical stages: 1) formation of the selected Offeror / LLNS partnership; 2) Dawn Demo; 3) Deployment of the Dawn system for ASC application code development and scaling; 4) demonstrate the Sequoia prototype hardware and software capabilities with Sequoia benchmarks; 5) LLNS decision on the size of Sequoia system to build (Sequoia or Sequoia14, see section 2.12); 6) demonstrate a peak (petaFLOP/s) plus weighted sustained performance (application specific figure-of-merit) of at least forty (40.0) on the five ASC Sequoia Marquee application benchmarks; 7) Sequoia deployment to the program as the ASC Tri-Laboratory capability platform; 8) Sequoia deployment to the ASC Program as a general purpose production resource; and 9) final retirement of the Sequoia platform after five years of use from the time of acceptance. The table below gives general progress metrics for the successful completion of the Sequoia subcontract(s). These metrics include target dates based on ASC programmatic requirements and anticipated fiscal year funding. These target dates are not mandatory and can be modified to more closely match an Offeror's product roadmap. However, there is a significant value to LLNS and the ASC Program to timely delivery of the proposed system and computing capability.

#	Target Date	Event	Metrics
1	Dec 2008	Partnership Formation	Contract award and development of initial overall project plan
2	Mar 2009	Dawn Demo	Demonstration of Dawn hardware and software prior to system shipment.
3	June 2009	Dawn Acceptance	Delivery, stabilization and acceptance of Dawn system. Five year Dawn maintenance clock starts after Dawn acceptance.
4	2Q CY 2010	Sequoia Prototype Demo	Demonstration of key Sequoia hardware and software technology for applications scalability and system effectiveness with Sequoia Benchmarks.
5	4Q CY 2010	Sequoia Build Size Decision	Offeror notified that LLNS elects to exercise the Sequoia or Sequoia14 system build.

#	Target Date	Event	Metrics
6	2Q CY 2011	Sequoia Demo and Delivery	Demonstration of Sequoia peak plus sustained performance on Sequoia marquee benchmarks performance. Delivery to LLNL.
7	3Q CY 2011	Sequoia Deployment	Acceptance of Sequoia. Sequoia stabilization. Start of limited availability. Start of five year maintenance clock.
8	4Q CY 2011	Sequoia Production	Migration to heavy QU workload and change in hardware/software maintenance. Start of general availability.
9	3Q CY 2016	Sequoia End of Life	Planned useful lifespan of Sequoia is five years after acceptance.

End of Section 1.0

2.0 Sequoia High-Level Hardware Requirements

The end product of the selected Offeror's ASC Sequoia development and engineering activity will be a balanced compute resource 24 times more powerful than ASC Purple on the ASC Integrated Design Codes (IDC) and 25-50 times more powerful than BlueGene/L (65,536 node configuration) on ASC Scientific Applications currently available within the ASC Tri-Laboratory community. It will be focused on solving the critical stockpile stewardship problems, that is, the large-scale application problems at the edge of the ASC Program's understanding of weapon physics. This fully functional Sequoia system must be useful in the sense of being able to deliver a large fraction of peak performance to a diverse scientific and engineering workload. It must also be useful in the sense that the code development and production environments are robust and facilitate the dynamic workload requirements.

The specifications below define a Sequoia scalable system with peak of at least 20 petaFLOP/s. Offeror should provide an estimate of the proposed Sequoia system sustained performance on ASC marquee benchmarks (measured as a weighted average of the figure-of-merit for these codes) based on the performance of the marquee demonstration codes identified in Section 9.1.1. The physics and numerical analysis algorithms and coding styles of these codes are indicative of key portions of the overall stockpile stewardship workload. Obviously, Offeror may necessarily have to estimate the efficiency of the marquee applications on the proposed system in order to determine what to actually bid, price and ultimately deliver to meet the mandatory requirement identified in Section 2.1. If the delivered performance of the marquee applications on the proposed system is below the Offeror's estimates, then more than 20.0 petaFLOP/s of peak computational resources will be required, and scaled as defined in Section 2.3. In LLNS' view, this issue will motivate additional Offeror innovation during subcontract performance.

Due to the classified ASC programmatic requirements both the Sequoia and Dawn systems will be initially deployed in the unclassified (BLACK) network environment and, once accepted and stabilized, migrated to, and be gainfully employed in, the classified (RED) network environment.

Development of the Dawn and Sequoia systems may comply with the requirements identified in section 8.0, Project Management.

There is only one mandatory requirement for Sequoia, Section 2.1 "Sequoia System Peak". There is only one mandatory option requirement for Sequoia, Section 2.12.3 " Sequoia14 System Performance." The specific hardware and software Target Requirements the Sequoia system may meet are delineated in Sections 2.0 and 3.0, respectively, with (TR-1, TR-2 and TR-3) designation with TR-1 being highest priority and TR-3 being lowest. Target Options (TO-1, TO-2) are specific hardware configurations described in Section 2.12 that LLNS has identified as options that may be advantageous for the ASC program. In addition to the highest priority hardware and software targets or options, the Offeror may deliver any Target Requirements (TR-2 and TR-3) for the Sequoia system, and any additional features consistent with the objectives of this project and Offeror's Research and Development Plan, which the Offeror believes will be of benefit to the project.

Offeror's technical proposal Section 2 will contain a detailed description of the proposed Sequoia System. It may include a detailed discussion of how all of the Baseline Characteristics (MR, MO, TRs, and TOs) will be met, as well as a discussion of LLNS and Offeror identified Value-Related Characteristics included in the technical solution.

2.1 Sequoia System Peak (MR)

The Sequoia baseline system performance shall be at least 20.0 petaFLOP/s (20.0×10^{15} floating point operations retired per second).

2.1.1 Sequoia System Performance (TR-1)

The Sequoia system performance may be at least $M = P + S = 40.0$. Where *P* is the peak of the system as defined in section 2.1 and *S* is the weighted figure of merit for five applications and is defined in Section 9.4.2.

2.2 Sequoia Major System Components (TR-1)

Offeror's proposed Sequoia system will include the following major system components (see Figure 1-5): 1) the Compute Nodes (CN) and I/O Nodes (ION), the Login Nodes (LN), the Service Nodes (SN), and the management Ethernet. Not shown in the figure is the interconnect network(s) that provide high speed, low latency RDMA and MPI communications between the nodes in the system. The remaining components in Figure 1-5, including the Storage Area Network (SAN), Lustre OSS and OSS resources will be supplied by LLNS and integrated with the proposed Sequoia system by the selected Offeror and LLNS in partnership.

Offeror's technical proposal will include a concise description of the Sequoia system architecture that includes the following.

- System architectural diagram showing all nodes, networks, external network connections and their proposed functions.
- Detailed architectural diagram of each node type showing all major components (e.g., processor cores and their functional units, caches, memory, system interconnect interfaces, DMA engines, etc.) and data pathways along with latency and bandwidth to move data between them.
- Detailed architectural diagram showing all management networking components, connections to Sequoia system, and connections to the front-end nodes.
- Number of nodes required or recommended by the Offeror for system functions (e.g., cluster wide file system operation, switch operation and management, RAS and other system management systems, user login) may be indicated and clearly denoted as NOT part of the compute nodes.
- Describe each subsystem and its system architectural requirements including bandwidths and latencies into, out of, and through each component.
- Clearly indicate the known, anticipated and suspected I/O performance limiters and bottlenecks.

2.2.1 IO Subsystem Architecture (TR-1)

The CN IO data path for file IO to the LLNS supplied Lustre file system may be between the CN to the ION over the system interconnect where the IO operations are handled by the Lustre client and then over the Offeror provided SAN interface to LLNS supplied SAN infrastructure to the LLNS supplied Lustre MDS and OSS. The CN IO data path for IP based communications to the LLNS SAN based IP devices may be between the CN to the ION over the system interconnect where the IP packets are routed to the Offeror provided SAN interface to LLNS supplied SAN infrastructure to the LLNS supplied IP based devices. The LN IO data path for both file IO to

the Lustre file system and SAN based IP devices is over the external networking interfaces on the LN.

The Sequoia target architecture (see Figure 1-5) provides for a static allocation of CN to ION. This provides for scalable IO bandwidth proportional to job size (number of CN and ION utilized by a job) with full system jobs running on 100% of the CN achieving at least 100% of the IO delivered bandwidth, half system jobs running on 50% of the CN achieving at least 50% of the full system IO delivered bandwidth and quarter system job running on 25% of the CN achieving at least 25% of the full system IO delivered bandwidth. This target architecture also allows for a distributed and scalable system software infrastructure by utilizing ION to perform some of the processing in parallel.

As a separately priced option specified in Section 2.12.1, Offeror may propose an enhanced IO subsystem that allows smaller jobs to achieve 2x of the IO file system bandwidth of the baseline system.

2.3 Sequoia Component Scaling (TR-1)

In order to provide maximum flexibility to Offerors in meeting the goals of the ASC Program, the exact configuration of the Sequoia scalable system is not specified. Rather, the Sequoia configuration is given in terms of lower bounds on component attributes relative to the peak performance of the proposed configuration. The Sequoia scalable system configuration may meet or exceed the following parameters:

- Memory Size (Byte:FLOP/s) ≥ 0.08
- Memory Bandwidth (Byte/s:FLOP/s) ≥ 0.2
- Node Interconnect Aggregate Link Bandwidth (Bytes/s:FLOP/s) ≥ 0.15
- Node Interconnect Minimum Bi-Section Bandwidth (Bytes/s:FLOP/s) ≥ 0.0025
- System Sustained SAN Bandwidth (GB/s:petaFLOP/s) ≥ 25.6
- High Speed External Network Interfaces (GB/s:petaFLOP/s) ≥ 6.4

The foregoing parameters will be computed as follows:

- Peak FLoating point OPeration per second (FLOP/s) rate computation: Maximum number of floating point arithmetic **operations** (chained multiply add counts as two) that can be completed per core cycle per compute node times the number of compute nodes in the system. Peak FP arithmetic operation per second rate is measured in petaFLOP/s = 10^{15} FLOP/s.
- Memory Size computation: Number of bytes of main memory directly addressable with a single LOAD/STORE instruction (but not caches nor ROM nor EPROM) of each compute node times the number of compute nodes in the system. Memory is measured in petiByte (PiB) = 2^{50} Bytes.
- Memory Bandwidth/Peak FP Instructions (Byte/s:FLOP/s) computation: maximum number of bytes per second that some or all of the cores in a node can simultaneously move between main memory and processor registers (node memory bandwidth) in the compute nodes times the total number of compute nodes in the system divided by the peak FLOP/s of the system.

- Node Interconnect Aggregate Link Bandwidth computation: Intra-cluster network link bandwidth is peak speed at which user data can be moved bi-directionally to/from a compute node over a single active network link. It is calculated by taking the MHz rating of the link time the width in bytes of that link minus the overhead associated with link error protection and addressing. The node interconnect aggregate link bandwidth is the sum over all active compute node links in the system of the node interconnect link bandwidths. Passive standby network interfaces and links for failover may not be counted.
- Node Interconnect Minimum Bi-Section Bandwidth computation: A bi-section of the system is any division of the compute nodes that evenly divides the total system into two equal partitions. A bi-section bandwidth is the peak number of user payload bytes per second that could be moved bi-directionally across the high speed interconnect network between compute nodes summed over each compute node in one partition communicating to one other compute node in the other partition. The Node Interconnect Network Minimum Bi-Section Bandwidth is the minimum over all bi-sections of the bi-section bandwidths.
- System Sustained SAN Bandwidth (GByte/s:petaFLOP/s) computation: The system sustained filesystem bandwidth is the measured rate an application can read or write data to/from LLNS supplied Lustre filesystem from all CN through the ION and LLNS supplied SAN to the Lustre OSS. Note that the SAN connects to the Sequoia ION. The methodology for measuring this metric is specified in Section 2.9.1. Note that Section 2.12.1 enhances this SAN bandwidth requirement with a separately priced Technical Option that configures the Sequoia system to deliver 2x this bandwidth to applications running on 50% and 25% subdivisions of the system.
- High Speed External Network Interfaces (GB/s:petaFLOP/s) computation: The high speed external network interface link bandwidth (in GB/s) is the HW rated link uni-directional bandwidth. This is the data rate, so it is 4.0 GB/s for IniniBand 4x QDR and 1.25 GB/s for 10GbE (IEEE 802.3ba) Ethernet. The cluster high speed external network interfaces bandwidth is the sum of over all the external network interface link bandwidths. Note that the External Network connects to the Sequoia LN.

Example: For a 20.0 petaFLOP/s peak system, Section 2.3 specifies that the system may have at least 1.6 PiB of memory, 4.0 PB/s memory bandwidth, 2.0 PB/s node interconnect network aggregate link bandwidth, 50 TB/s intra-cluster networking bi-sectional bandwidth, and 512 GB/s system sustained SAN bandwidth and 128 GB/s peak external networking bandwidth.

2.4 Sequoia Node Requirements (TR-1)

The following requirements apply to all Sequoia system node types except where superseded in subsequent sections.

2.4.1 Node Architecture (TR-1)

The Shared memory Multi-Processor (SMP) nodes may be a set of processor cores sharing random access memory within the same memory address space. The cores may be connected via a high speed, extremely low latency mechanism to the set of hierarchical memory components. The memory hierarchy consists of at least processor registers, cache and memory. The cache may also be hierarchical. If there are multiple caches, they may be kept coherent automatically by the hardware. The main memory may be a Uniform Memory Access (UMA) architecture. The access mechanism to every memory element may be the same from every core. More

specifically, all memory operations may be accomplished with load/store instructions issued by the core to move data to/from registers from/to the memory.

2.4.2 Core Characteristics (TR-1)

Each node may be an aggregate of homogeneous general purpose computing cores consisting of high-speed instruction issue, arithmetic, logic units, and memory reference execution units integrated together with the necessary control circuitry and interprocessor communications mechanism(s) and caches. All functional units and data paths may be at least 64b data path plus error detecting and correcting codes. Virtual memory data pointers may be at least 64b with at least 42b physical addresses. Each may execute fixed and IEEE 754 floating-point arithmetic, logical, branching, index, and memory reference instructions. A 64-bit data word size may directly handle IEEE 754 floating-point numbers whose range is at least 10^{-305} to 10^{+305} and whose precision is at least 14 decimal digits. The cores and memory hierarchy may provide an appropriate mechanism for interprocessor communication, interrupt, and synchronization. The core may contain built in error detection and fault isolation for all core components and in particular for the floating-point units, all caches, TLB entries. All storage elements not limited to registers, caches, TLB entries, memory may be at a minimum SECDED protected.

2.4.3 IEEE 754 32-Bit Floating Point Numbers (TR-3)

The cores may have the ability to operate on 32-bit IEEE 754 floating-point numbers whose range is at least 10^{-35} to 10^{+35} and whose precision is at least 6 decimal digits, for improved memory utilization and improved execution times.

2.4.4 Inter Core Communication (TR-1)

The cores may provide sufficient atomic capabilities (e.g., test-and-set or load-and-clear) along with some atomic incrementing capabilities (e.g., test-and-add or fetch-and-increment/fetch-and-decrement) so that the usual higher level synchronizations (i.e., critical section, barrier, etc.) can be constructed. These capabilities may allow the construction of memory and execution synchronization that is extremely low latency (<70 core cycles in the contention free case). As the number of user threads can be large in a Sequoia node, special hardware mechanism may be provided that allows multiple threads to simultaneously register with a barrier/lock device with a total latency of less than 350 processor clocks for the maximal case of all hardware threads trying to register with the barrier on the same cycle. This corresponds to a latency of effectively less than 4 processor cycles per thread when utilizing this on-chip hardware barrier/lock mechanism. Hardware support may be provided to allow for DMA to be coherent with the local node memory. Additionally, these synchronization capabilities or their higher-level equivalents will be directly accessible from user programs.

The atomic instructions API overhead, in the absence of contention, may be less than or equal to one micro-second (1.0×10^{-6} seconds).

2.4.5 Node Interconnect Interface (TR-2)

Each node may be configured with a high speed, low latency interconnect (section 2.8) interface. This interface may allows all cores in the system to simultaneously communicate synchronously or asynchronously with the high speed interconnect. The asynchronous communications mechanisms may employ a DMA engine or equivalent that does not require the core to physically move the data. This interface may be capable of delivering either full memory bandwidth to the interconnect or all interconnect off node links simultaneously, whichever is less.

2.4.6 Hardware Support for Low Overhead Threads (TR-1)

The nodes may be configured with hardware mechanisms for spawning, controlling and terminating low overhead computational threads. This published and well documented hardware thread interface support may include a low overhead locking mechanism and a highly efficient fetch and increment operation for memory consistency among the threads. Offeror supplied OpenMP and POSIX thread implementations for all provided compilers may use these hardware mechanisms to implement a highly efficient programming models for node parallelism. Offeror may fully describe this hardware facility and limitations and the potential benefit to ASC applications for exploiting OpenMP and POSIX threads node parallelism within MPI tasks.

2.4.7 Hardware Support for Innovative node Programming Models (TR-2)

The Sequoia nodes may be configured with hardware support for innovative node programming models such as Speculative Execution (SE) or Transactional Memory (TM) that allow the automatic extraction and execution of parallel work items where sequential execution consistency is guaranteed by the hardware, not the programmer nor compiler. These facilities may allow the correct execution of multiple work items that have infrequent load/store conflicts. These hardware facilities may allow ASC applications to utilize all SMP programming techniques (e.g., OpenMP, POSIX Threads, SE or TM) within a single application, with the restriction that within a given subroutine only one style of node parallel programming will be active within a thread at a time. These hardware facilities and the Low Overhead Threads may be combinable allowing the programming models to be nested within an application call stack. Offer may fully describe these hardware facilities and limitations and the potential benefit to ASC applications for exploiting innovative programming models for node parallelism within MPI tasks.

2.4.8 Programmable Clock (TR-2)

There may be a real-time clock per core capable of causing a hardware interrupt after a preset interval (i.e., a programmable clock). The clock frequency may be at least one megahertz and the preset interval may be capable of being set in increments of 10 microseconds or less. There may be at least 16 seconds allocated for the time interval. This clock may have at least 24 bits.

2.4.9 Hardware Interrupt (TR-2)

The nodes may have hardware support for interrupting given subsets of cores based on conditions noted by the operating system or by other cores within the subset executing the same user application.

2.4.10 Hardware Performance Monitors (TR-1)

The cores may have hardware support for monitoring system performance. This published and well documented hardware performance monitor (HPM) interface will be capable of separately counting hardware events generated by every thread executing on every core in the node. This HPM may count at least the following: instructions (FP/INT/BR) per cycle with or without loads/stores; cache hits and misses and prefetches for all levels of the data and instruction cache hierarchy; TLB misses for all levels of the data and instruction cache hierarchy; branch mis-predictions; snoop requests; snoop hits; load miss penalty in cycles; number of pipeline flushing operations (e.g. sync). Countable events from the Floating Point Unit or Units (FPU) may include floating point scalar and SIMD (or vector) add/subtract, mult, fused mult add, divide, double load, quad load, double store and quad store events. The available FPU events may be

completely inclusive of all FPU activity, and may allow accurate calculation of total floating point performance (FLIN/s and FLOP/s) of a core when the counters are configured to count the floating point events for that core. In addition, the node interconnect interfaces may have hardware support for monitoring message passing performance on all proposed networks. If hardware support for LOT (section 2.4.5) or SE/TM (section 2.4.7) are proposed, then the HPM may count relevant events to determine parallel programming (in)efficiencies. This HPM may have 64b counters and the ability to notify the node OS of counter wrapping. This data will be made available directly to applications programmers and to code development tools (Section 3.7.8).

2.4.11 Hardware Debugging Support (TR-1)

The cores may have hardware support for debugging of user applications, and in particular, hardware that enables setting regular data watch points (e.g., debug registers and hardware interrupts on read/write to a specific virtual memory location) and break points as well as fast versions of them via fast trap mechanisms (e.g., fast trap instructions that allow the application to trap into an exception handler without having to notify the debugging process). If hardware support for TM/SE (Section 2.4.7) is proposed, hardware support may also be proposed to allow tools to trace, debug, and analyze TM/SE threads in-depth (e.g., hardware support for fine-grain memory conflict detection). The Offeror may fully describe the hardware debugging facility and limitations. These hardware features may be made available directly to applications programmers in a published and supported API (Section 3.9.8) and utilized by the code development tools including the debugger (Sections 3.7.2, 3.7.2.6).

2.4.12 JTAG Infrastructure

The Sequoia system nodes may be accessible over a out of band JTAG interface. This interface may be accessible over the service Ethernet network. This interface may allow system RAS and system administration functions an out of band (off the system interconnect) path.

2.4.13 No Local Hard Disk (TR-1)

The Sequoia system nodes may be configured **without** a hard disk drive.

2.4.14 Remote Manageability (TR-1)

All nodes may be 100% remotely manageable, and all routine administration tasks automatable in a manner that scales up to the full system size. In particular, all service operations under any circumstance on any node must be accomplished without the attachment of keyboard, video, monitor, and mouse (KVM). Areas of particular concern include remote console, remote power control, system monitoring, and node BIOS or firmware.

Offeror will fully describe all remote manageability features, protocols, APIs, utilities and management of all node types bid. Any available manuals (or URLs pointing to those manuals) describing node management procedures for each node type will be provided with the proposal.

All remote management protocols, including power control, console redirection, system monitoring, and management network functions must be simultaneously available via the system management Ethernet. Access to all hardware system functions within the nodes, must be made available at the OS level so as to enable complete system health verification.

2.4.14.1 Remote Console and Power Management (TR-3)

Offeror may provide a hardware interface to the console port of every node in the system. The console interfaces will be aggregated on the Management Ethernet. Offeror may provide

a scalable and reliable hardware interface to change the power state (up/down/reset) and querying the power state of every node or node aggregates, racks, etc., that use power. The provided power interfaces may be aggregated in a power control device on the Management Ethernet. The power control infrastructure may be able to reliably power up/down all nodes or groups of nodes in the system simultaneously. Reliable here means that 1,000,000 power state change commands may complete with at most one failure to actually change the power state of the target nodes.

2.5 I/O Node Requirements (TR-1)

The following requirements are specific to the I/O Nodes (ION) and augment the general node requirements (Section 2.4) above. As defined in Section 2.1, ION do not contribute to the system peak performance. The CN data path for file IO to the LLNS supplied Lustre file system may be between the CN and the ION over the system interconnect where the IO operations are handled by the Lustre client and then over the Offeror provided SAN interfaces to the LLNS supplied SAN infrastructure to the LLNS supplied Lustre MDS and OSS. The CN IO data path for IP based communications to the LLNS supplied SAN based IP devices may be between the CN to the ION over the system interconnect where the IP packets are routed to Offeror's provided SAN interfaces to the LLNS supplied IP based devices.

2.5.1 ION Count (TR-1)

Offeror will configure the system with sufficient ION to meet the IO file system performance requirements in Section 2.9.1. If the delivered system does not meet these performance requirements, then Offeror will, at no additional expense to LLNS, add additional ION to meet these requirements.

2.5.2 ION IO Configuration (TR-2)

The ION may incorporate one or more SAN interfaces with PCIe2 x8 or faster busses each with a single slot filled with single PCIe2 x8 or faster InfiniBand 4x QDR or faster interface with one 40 Gb/s SFP+ or smaller interface to short range (SR) multi-mode fiber (MMF) optics capable of driving optical cables of at least 40m length or PCIe2 x8 40 or 100 Gb/s IEE 802.3ba compliant Ethernet interface with one SFP+ or smaller interface to short range (SR) multi-mode fiber (MMF) optics capable of driving optical cables of at least 40m length. The Sequoia ION is the data path from the CN to the SAN based resources (Lustre MDS and OSS, external 1 and 10 Gb/s Ethernet networks). The Offeror's proposed ION configuration may carefully balance the *delivered* ION Interconnect bandwidth with the *delivered* PCIe2 x8 bus/slot bandwidth and with the *delivered* SAN network card bandwidth. In addition, the Offeror's proposed ION configuration may carefully balance these delivered IO rates with the delivered integer processing performance and delivered memory bandwidth of the ION.

2.5.3 ION Delivered Performance (TR-2)

The proposed ION PCIe2 x8 slots may *deliver* 3.6 (3.6+0 or 0+3.6) GB/s uni-directional and 7.2 (3.6+3.6) GB/s bi-directional bandwidth. The proposed InfiniBand or Ethernet interface may *deliver* at least 90% LNET Self Test with 1MB transfers. These interfaces may *deliver* an aggregate of 90% of bidirectional peak to TTCP benchmark. ION configurations with more than one PCIe2 x8 bus/slot and populated network interfaces may *deliver* an aggregate bandwidth of at least 90% of the *delivered* individual bus/interface bandwidth times the number of populated network interfaces.

Comment [MKS2]: PCIe2 is 5.0GT/s. x8 is 40+40Gb/s bandwidth or 5.0+5.0 GB/s. This is raw line rate. 8b/10b clock encoding reduces the overall peak bandwidth by 0.8. Peak is 4+4 GB/s.

Example: An ION with two PCIe2 x8 buses and InfiniBand cards may *deliver* at least 0.9*2*0.90*7.2 = 11.67 GB/s bidirectional bandwidth with the LNET SelfTest with 1MB transfers. This ION would require at least 13 GB/s *delivered* interconnect bandwidth and sufficient memory bandwidth to route packets to/from the interconnect from/to the InfiniBand.

2.6 Login Node Requirements (TR-1)

The following requirements are specific to the Login Nodes (LN) and augment or supersede the general node requirements (Section 2.4) above. As defined in Section 2.1, LN do not contribute to the system peak performance. The LN are one or more node(s) that provide the hardware necessary for end-users of the system to login to the system and perform normal user activities of code development, job launch, job management and data movement. Local disk on the Login node may be used for OS configuration and high speed local (/tmp and /var/tmp) temporary file systems to accelerate editing, compiling and loading large applications. The LN data path for both file IO to the Lustre file system, local RAID disk and external networking based IP devices (such as NFS servers and user login via ssh) are all over the SAN interface.

2.6.1 LN Count (TR-1)

Offeror will configure the system with sufficient LN configured with external networking interfaces to meet the high speed external networking interfaces requirement in Section 2.3. The Offeror will configure the system with sufficient number of LN with appropriate memory and local disk resources to support 100 active users executing 50 simultaneous compiles. The LN may have a similar SAN interface as the ION defined in Section 2.5.

2.6.2 LN Locally Mounted Disk and Multiple Boot (TR-1)

The LN may have sufficient disk resources in aggregate to allow the storage of: 1) at least 10 system software images for each type of node (if not managed on the SN); and 2) 50 TB of local disk for /tmp and /var/tmp. These disk resources may be packaged with the node (i.e., physically local) or packaged remotely, but locally mounted. Each system software image of each type of node may have sufficient disk space for operating system, code development tools and other system binaries, swap and local tmp, NFSv4 cache. The LN locally mounted disk may be configured with High Availability, High IOPS RAID 5 (or better) arrays of hard disks as specified in Sections 2.9.2 and 2.9.3.

The LN may have the capability to boot up to 10 different versions of the operating system and all associated software (i.e., ten completely separate and independent software releases or patch levels). Switching to a new boot device will be accomplished by the root user issuing commands at the shell prompt and will not require recabling any hardware.

2.6.3 LN IO Configuration (TR-2)

The LN may have one or more interfaces with PCIe2 x8 or faster busses each with a single slot filled with single PCIe2 x8 or faster InfiniBand 4x QDR or faster interface with one 40 Gb/s SFP+ or smaller interface to short range (SR) multi-mode fiber (MMF) optics capable of driving optical cables of at least 40m length or PCIe2 x8 40 or 100 Gb/s IEE 802.3ba compliant Ethernet interface with one SFP+ or smaller interface to short range (SR) multi-mode fiber (MMF) optics capable of driving optical cables of at least 40m length.

Offeror's proposed LN configuration may carefully balance the *delivered* LN local iSER RAID file system bandwidth with the *delivered* PCIe2 x8 bus/slot bandwidth and with the *delivered* SAN network card bandwidth. In addition, Offeror's proposed LN configuration may carefully

balance these *delivered* IO rates with the *delivered* integer processing performance and *delivered* memory bandwidth of the LN.

2.6.4 LN Delivered Performance (TR-2)

The proposed LN PCIe2 x8 slots may *deliver* 3.6 (3.6+0 or 0+3.6) GB/s uni-directional and 7.2 (3.6+3.6) GB/s bi-directional bandwidth. The proposed InfiniBand or Ethernet interface may *deliver* at least 90% LNET Self Test with 1MB transfers. These interfaces may *deliver* an aggregate of 90% of bidirectional peak to TTCP benchmark. LN configurations with more than one PCIe2 x8 bus/slot and populated network interfaces may *deliver* an aggregate bandwidth of at least 90% of the *delivered* individual bus/interface bandwidth times the number of populated network interfaces. These interfaces utilizing iSER or iSCSI may *deliver* an aggregate of at least 1,024 4KiB transactions per second with 50% read and 50% write balance.

Comment [MKS3]: PCIe2 is 5.0GT/s. x8 is 40+40Gb/s bandwidth or 5.0+5.0 GB/s. This is raw line rate. 8b/10b clock encoding reduces the overall peak bandwidth by 0.8. Peak is 4+4 GB/s.

2.7 Service Node Requirements (TR-1)

The following requirements are specific to the Service Nodes (SN) and augment the general node requirements (Section 2.4) above. As defined in Section 2.1, SN do not contribute to the system peak performance. SN are a set of nodes that provide *all* scalable system administration and RAS functionality. The number of required SN is determined by Offeror's scalable system administration and RAS architecture and the overall size of the system.

2.7.1 SN Scalability (TR-1)

The Service Nodes (SN) are the one or more system node(s) that provide *all* the scalable hardware necessary to completely manage the system. The SN may have sufficiently scalable hardware to boot the entire system in less than 15 minutes per Section 3.5.2.1.

2.7.2 SN Communications (TR-1)

The SN cluster may communicate directly with ION and LN with the interconnect and the SAN defined in Section 2.5 and management Ethernet defined in Section 2.4.14.

2.7.3 SN Locally Mounted Disk and Multiple Boot (TR-1)

The SN may have sufficient disk resources in aggregate to allow the storage of: 1) at least 10 system software images for each type of node; and 2) six months of system RAS information. These disk resources may be packaged with the node (i.e., physically local) or packaged remotely, but locally mounted. Each system software image of each type of node may have sufficient disk space for operating system, code development tools and other system binaries, swap and local tmp, NFSv4 cache. The SN locally mounted disk may be configured with High Availability, Hi IOPS RAID 5 (or better) arrays of hard disks as specified in Sections 2.9.2 and 2.9.3.

The SN may have the capability to boot up to 10 different versions of the operating system and all associated software (i.e., two completely separate and independent software releases or patch levels). Switching to a new boot device will be accomplished by the root user issuing commands at the shell prompt and will not require recabling any hardware.

If Offeror's bid configuration shares the RAID 5 disk resources between LN and SN, then the SN IO capacity requirements in this section are additive to the LN aggregate IO capacity requirement in Section 2.6.2.

2.7.4 SN IO Configuration (TR-2)

The SN may have one or more interfaces with PCIe2 x8 or faster busses each with a single slot filled with single PCIe2 x8 or faster InfiniBand 4x QDR or faster interface with one 40 Gb/s SFP+ or smaller interface to short range (SR) multi-mode fiber (MMF) optics capable of driving optical cables of at least 40m length or PCIe2 x8 40 or 100 Gb/s IEE 802.3ba compliant Ethernet interface with one SFP+ or smaller interface to short range (SR) multi-mode fiber (MMF) optics capable of driving optical cables of at least 40m length.

Offeror's proposed SN configuration may carefully balance the *delivered* SN local iSER or iSCSI RAID file system bandwidth with the *delivered* PCIe2 x8 bus/slot bandwidth and with the *delivered* SAN network card bandwidth. In addition, Offeror's proposed SN configuration may carefully balance these *delivered* IO rates with the *delivered* integer processing performance and delivered memory bandwidth of the SN.

2.7.5 SN Delivered Performance (TR-2)

Offeror's SN configuration may have sufficient processing power, memory capacity and bandwidth, number of interfaces and delivered bandwidth to/from the Management Network interfaces and local disk capacity and bandwidth to effectively manage the entire system. In particular, the local disk interface may have sufficient random IOPS performance so that the RAS database transaction rate is sufficient to allow the system installation, reconfiguration, reboot, or job launch time targets. See RAS Section 6.1.12.

2.8 Sequoia Interconnect (TR-1)

A physical network or networks for high-performance intra-application communication is required for Sequoia. The Sequoia interconnect may connect all Compute (CN), IO Nodes (ION), Login Nodes (LN) and Service Nodes (SN) in the system.

2.8.1 Interconnect Messaging Rate (TR-1)

The Sequoia CN messaging rate may be measured from a single reference CN node with N MPI tasks ($1 \leq N \leq NCORE$) on that CN sending/receiving messages of a size that optimizes system performance (e.g., with MPI_SEND/MPI_RECV or MPI_ISEND/MPI_IRECV pairs for measuring uni-directional bandwidth, and for bi-directional bandwidth with MPI_SENDRECV or MPI_ISEND/MPI_IRECV pairs) to/from a set of N MPI tasks with one MPI task per CN. In other words the reference CN with N MPI tasks on it communicating with N other CN each with 1 MPI task per node.

The CN interconnect messaging rate may be at least 3.2 mM/s/MPI (3.2 million messages per second per MPI task) for 1, 2 and 4 MPI tasks on the reference node and an aggregate rate of 12.8 mM/s (12.8 million Messages per second) for 5 through NCORE MPI task counts on the reference node.

Every CN in the Sequoia system will deliver this interconnect messaging rate.

2.8.2 Interconnect Delivered Latency (TR-1)

The interconnect latency is measured by the time for sending a minimum length MPI message from user program memory on one CN to user program memory on any other CN in the system and receiving back an acknowledgment divided by two (standard MPI user-space ping-pong test). Nearest neighbor interconnect latency is the interconnect latency between two CN that are

separated by at most one interconnect routing hop. The maximum interconnect latency is the maximum of the interconnect latency over all pairs of CNs in the system.

The maximum interconnect delivered latency when measured with one MPI task per CN or one MPI task per core on each CN will be less than 5.0 microseconds (5.0×10^{-6} seconds). The nearest neighbor interconnect delivered latency with one MPI task per CN or one MPI task per core on each CN will be less than 2.5 microseconds (2.5×10^{-6} seconds).

2.8.3 Interconnect Off-Node Aggregate Delivered Bandwidth (TR-1)

The CN interconnect off-node aggregate delivered bandwidth may be measured from a single reference CN node with N MPI tasks ($1 \le N \le$ NCORE) on that CN sending/receiving messages of a size that optimizes system performance (e.g., with MPI_SEND/MPI_RECV or MPI_ISEND/MPI_IRECV pairs for measuring uni-directional bandwidth, and for bi-directional bandwidth with MPI_SENDRECV or MPI_ISEND/MPI_IRECV pairs) to/from a sufficient number and placement of MPI tasks on other CNs to maximize performance.

The CN interconnect aggregate delivered bandwidth will be at least 80% of the CN aggregate link bandwidth. Specifically, the CN interconnect aggregate delivered bandwidth is targeted to deliver over 80% of all links simultaneously.

Every CN in the Sequoia system will deliver this all-connect off-node aggregate delivered bandwidth.

2.8.4 Interconnect MPI Task Placement Delivered Bandwidth Variation (TR-2)

Let N = NCORE * NCN be the number of MPI tasks. Let the MPI tasks be mapped to each core on all the CN in the system linearly (i.e., task 1 to task NCORE on the first CN, task NCORE+1 to task 2*NCORE on the second CN, etc.). Choose neighbors for each MPI task in this fixed MPI task layout indicative of a 3D mesh 27 point differencing stencil in the following manner. For each neighbor choice, k, let the neighbor list for task j ($1 \le j \le N$), S(k,j), be chosen so that: 1) each task has 26 neighbors; 2) every task has a unique set of neighbors; 3) every task is on a unique node; and 4) every task is chosen as a neighbor 26 times. Let K be the maximum number of possible unique sets S(k,j), ($1 \le k \le K$) for the proposed system. For each neighbor choice k, define the aggregate delivered MPI bandwidth B(k) as the sum over all tasks of the aggregate delivered MPI task bandwidth. The aggregate delivered MPI task bandwidth for task j is the sum of the uni-directional bandwidth sending messages of a size that maximizes performance to the 26 S(k,j) neighbors with MPI_ISEND or MPI_SEND or MPI_ALLGATHER with all tasks sending data to (and receiving from) their neighbors simultaneously.

Let $b = \min_k B(k)$ be the minimum over all neighbor choices of the aggregate delivered MPI bandwidth B(k) and $B = \max_k B(k)$ be the maximum. Then $D = (B - b)/b$ is a measurement of the delivered aggregate MPI bandwidth variation depending on where neighbors are placed in the system. The CN interconnect task placement delivered bandwidth variation target may be less than 12 ($D \le 12$).

Offeror may provide the D, B and b values and at least two neighbor choices S(k,j) that achieve B and b with a technical description fully explaining the rational for or measurement of these values and the corresponding neighbor choices with the proposal submission. Part of that

explanation may contain several other neighbor choices and resulting aggregate MPI task bandwidth B(k) with $b < B(k) < B$.

2.8.5 Delivered Minimum Bi-Section Bandwidth (TR-2)

The minimum bi-section bandwidth measurement may be the minimum delivered MPI bi-directional bandwidth over all possible bisections. For a bisection (one half of the nodes in Sequoia communicating with the other half of the nodes in two-node pairs) the aggregate delivered MPI bi-directional bandwidth computation is the sum over the two-node pairs of the delivered MPI bi-directional bandwidth in each pair with NCORE MPI tasks on each node. The delivered two-node MPI bandwidth is defined as the total number of bytes of user data sent from the two nodes in the pair in the test divided by the time globally elapsed during the sending and receiving operations on any node in the test. The minimum delivered aggregate MPI message bandwidth available to/from all nodes may be at least 80% of the interconnect minimum bi-section bandwidth (i.e., 80% when sending/receiving messages of a size that optimizes system performance with MPI_SENDRECV or MPI_ISEND/MPI_IRECV pairs).

2.8.6 Broadcast Delivered Latency (TR-2)

The MPI_BCAST delivered latency may be measured with one MPI task per core on all CN utilizing a user defined communicator (i.e., not MPI_COMM_WORLD). The data type for this measurement may be 64-bit floating point and the number of elements to be broadcast (i.e., the MPI_BCAST "length" parameter) may be 8,192. This measurement may be repeated for each subdivision of the tasks that maximizes the number of nodes in each subcommunicator into multiple subcommunicators of equal size (1, 2*1/2, 4*1/4, etc) down to an odd number of tasks per subcommunicator. When running with multiple subcommunicators the measurements may be contemporaneous. Within each subcommunicator, the MPI_BCAST elapsed wall clock time is measured by timing the operation start on the broadcasting core and the end as the last core in the subcommunicator to receive all the data. The broadcast delivered latency is the maximum over all subcommunicators of the MPI_BCAST elapsed wall clock.

The MPI_BCAST delivered latency on any above subcommunicator may be less than the ping-pong latency with message length 8,192*8 = 65,536 bytes on that set of MPI tasks.

2.8.7 All Reduce Delivered Latency (TR-2)

The MPI_ALLREDUCE sum, min and max with MPI_COMM_WORLD operation may be measured with the following methodology: for a given partition, iterate 10^3 times over the MPI_ALLREDUCE operation utilizing MPI_COMM_WORLD communicator and copies of MPI_COMM_WORLD. The MPI_ALLREDUCE latency for each MPI task is the wall clock time for that MPI task to perform this loop divided by 10^3. The maximum MPI_ALLREDUCE latency, for that MPI_ALLREDUCE operation, may be measured with one MPI task per core per CN on the partition. The datatype for this measurement may be 64-bit floating point and the number of vector elements to be reduced per core (i.e., the MPI_ALLREDUCE "count" parameter) may be 2^k, k=0, 1, 2, …, 16. This measurement may be repeated for each subdivision of the machine into two subpartitions of equal size (1, 2*1/2, 4*1/4, etc) subject to the partitioning restrictions, up to the minimum partition size. When running on multiple partitions the measurements may be contemporaneous with multiple copies of the benchmark. The MPI_ALLREDUCE latency is measured as the amount of wall clock time each MPI task takes to perform the MPI_ALLREDUCE operation. The maximum MPI_ALLREDUCE latency is the maximum over all MPI tasks of the individual MPI task MPI_ALLREDUCE latencies. The

interconnect MPI_ALLREDUCE latency utilizing MPI_COMM_WORLD communicator or copies of MPI_COMM_WORLD will be less than $10.0+0.002*2^k$ micro-seconds ($(10.0+0.002*2^k) \times 10^{-6}$ seconds) for sum, min, and max MPI_ALLREDUCE operations on vectors with length 2^k elements per MPI task.

The MPI_ALLREDUCE sum, min and max operation with user defined (i.e., not MPI_COMM_WORLD or copies of MPI_COMM_WORLD) communicators may be measured with the following methodology: for a given partition, iterate 10^3 times over the one MPI_ALLREDUCE operation per communicator utilizing a logical 3D Torus MPI task layout with each X-Plane, Y-plane and Z-Plane of MPI tasks in the logical layout utilizing a separate communicator. The MPI_ALLREDUCE latency for each MPI task is the wall clock time for that MPI task to perform this loop divided by 3×10^3. The maximum MPI_ALLREDUCE latency, for that MPI_ALLREDUCE operation, may be measured with one MPI task per core per CN on the partition. The datatype for this measurement may be 64-bit floating point and the number of vector elements to be reduced per core (i.e., the MPI_ALLREDUCE "count" parameter) may be 2^k, k=0, 1, 2, …, 16. This measurement may be repeated for each subdivision of the machine into multiple subpartitions of equal number of nodes (1, 2*1/2, 4*1/4, etc) subject to the partitioning restrictions, up to the minimum partition size. When running on multiple partitions the measurements may be contemporaneous with multiple copies of the benchmark. The MPI_ALLREDUCE latency is measured as the amount of wall clock time each MPI task takes to perform the MPI_ALLREDUCE operation. The maximum MPI_ALLREDUCE latency is the maximum over all MPI tasks of the individual MPI task MPI_ALLREDUCE latencies. The interconnect MPI_ALLREDUCE latency utilizing a user defined communicator will be less than $10.0+0.002*2^k$ micro-seconds ($(10.0+0.002*2^k) \times 10^{-6}$ seconds) for sum, min, and max MPI_ALLREDUCE operations on vectors with length 2^k elements per MPI task.

2.8.8 Interconnect Hardware Bit Error Rate (TR-1)

The Sequoia full system Bit Error Rate (BER) for non-recovered errors in the CN interconnect is targeted to be less than 1 bit in 1.25×10^{20}. This error rate applies to errors that are not automatically corrected through ECC or CRC checks with automatic resends. Any loss in bandwidth associated with the resends would reduce the sustained interconnect bandwidth and is accounted for in sustained bandwidth for the Sequoia interconnect.

2.8.9 Global Barriers Network Delivered Latency (TR-2)

The MPI_BARRIER operation may be measured with the following methodology: for a given partition, iterate 10^3 times over the MPI_BARRIER operation utilizing MPI_COMM_WORLD and copies of MPI_COMM_WORLD. The MPI_BARRIER latency for each MPI task with one MPI task per core per CN in the partition is the wall clock time for that MPI task to perform this loop divided by 10^3. The maximum MPI_BARRIER latency is the maximum over all individual MPI task MPI_BARRIER latencies. This measurement may be repeated for each subdivision of the machine into multiple subpartitions of equal number of CN (1, 2*1/2, 4*1/4, etc) up to the minimum partition size. When running on multiple partitions the measurements may be contemporaneous with multiple copies of the benchmark. This benchmark may be run under conditions matching those of the general workload (i.e., special calls requiring root access that perform task binding to cores or changing thread/process/task priorities is specifically disallowed) with normal system daemons running under normal operating conditions. However the benchmark may not checkpoint during testing. The maximum MPI_BARRIER latency will be under 5.0 microseconds (5.0×10^{-6} seconds).

The maximum MPI_BARRIER latency when utilizing user defined communicators will be under 10 microseconds (1.0×10^{-5} seconds)

2.8.10 Cluster Wide High Resolution Event Sequencing (TR-2)

The Sequoia system is targeted to include hardware support for a cluster-wide real-time clock or other hardware mechanism for cluster-wide event sequencing. The resolution of this mechanism may be less than 1 microsecond (1×10^{-6} seconds) within a single partition. This resolution of event sequencing is not required between partitions. This facility would be used for parallel program debugging and performance monitoring. This objective will be measured in software by measuring the latency of the global interrupt network. All the real-time clocks in the system are synchronized using the global interrupt network. Thus measuring the skew of the global interrupt network across all nodes will closely approximate the skew in the clocks.

This estimate and a transitive closure argument can be applied to show approximate upper bound of clock synchronization is within the target objective. This methodology will be used to demonstrate this requirement.

This resolution of event sequencing is not required between partitions. This facility would be used for parallel program debugging and performance monitoring. The API overhead for obtaining the current clock reading from a user program on any node may be less than one micro-seconds (1×10^{-6} seconds).

2.8.11 Interconnect Security (TR-2)

The interconnect hardware and supporting software interfaces may segregate user application jobs so that one user job may not be able to read/write packets from/to another job.

2.9 Input/Output Subsystem (TR-1)

The Input/Output subsystem for Sequoia has two major components: 1) an Offeror provided interface to the LLNS provided SAN; and 2) local RAID storage for LN and SN. User login and other external network TCP/IP services access to LN are supported over the SAN interface on the LN. External network TCP/IP services access to CN are supported by the SAN interfaces on the ION. By the time Sequoia is delivered, Infiniband™ SAN and attached storage and networking solutions should be widely available. This solution is highly desired because it offers the opportunity to share disk and external networking resources between multiple platforms within the Livermore Computing High Performance SAN environment (e.g., capacity computing clusters, data manipulation engines, visualization engines, archival storage). The architectural picture Figure 1-5 shows the preferred system layout for the I/O subsystem.

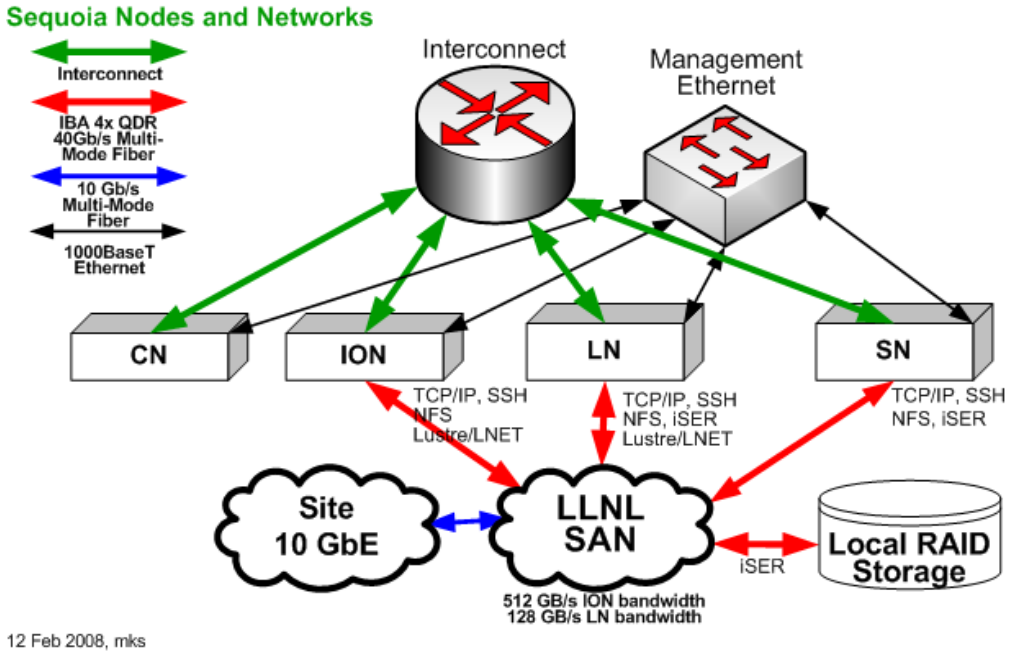


Figure 2-1: Offeror provided IO Subsystem components include SAN interface and local RAID storage for LN and SN.

2.9.1 File IO Subsystem Performance (TR-1)

Offeror may propose sufficient ION and SAN interfaces to provide 100% of the required delivered Lustre IO bandwidth (as defined in Section 2.2.1) for jobs running on 100% of the CN and to provide 50% of the required delivered Lustre IO bandwidth for jobs running on 50% of the CN using 50% of the ION and 25% of the required delivered Lustre IO bandwidth for jobs running on 25% of the CN using 25% of the ION. Note that Section 2.12.1 requires an option that doubles the delivered bandwidth to applications using only a portion of the compute nodes. These performance numbers may be measured with IOR_POSIX over a minimum of 8 hour test period. IOR_POSIX may be configured for writing and then reading files from Lustre using standard POSIX IO calls under the following benchmarking conditions:

Launch: IOR_POSIX may have one MPI task per node. Number of threads within the MPI task can be changed to maximize delivered IO performance.

Create: each IOR_POSIX MPI task may create one file with zero size.

Write: each MPI task may write 35% of node memory size data to the file and close the file.

Verify: each MPI task may open and read in all data from the file of another MPI task (shift) and verify the data was written correctly and close the file.

Read: Each MPI task may open and read the data in the file it originally wrote and close and delete the file.

Terminate: IOR_POSIX job will terminate.

Each run of IOR_POSIX may execute steps 2-5 above 4 times. IOR_POSIX prints out the read and write rate for each iteration. The figure of merit for IOR_POSIX is the minimum of the read rate and write rate. The figure of merit for IOR_POSIX file IO subsystem performance is the minimum of the four read and four write rates. The overall file IO subsystem I/O rate (R_p) is defined as

$$R_p = \underset{k=1}{\overset{N}{MIN}}(R_k)$$

Where N is the number of IOR_POSIX runs completed in the 8 hour test period.

2.9.1.1 File IO Function Ship Performance (TR-1)

Offeror provided hardware and software may deliver at least 95% of the SAN interface peak link unidirectional bandwidth (0.95*3.2 GB/s = 3.04 GB/s per IBA 4x QDR interface) to a user application performing file IO running on all CN associated with an ION to a Linux tempfs file system using 50% of the ION memory. This may be measured with the "IOR_POSIX" benchmark or equivalent running on each CN utilizing one file per IOR_POSIX instance in stonewalling mode. Note that the number of IOR_POSIX instances per CN is not specified, but all output files must be of the same size and fit on the tmpfs. These performance numbers may be measured with IOR_POSIX over a minimum of 30 minute test period. IOR_POSIX may be configured for writing and then reading files from ION tmpfs using standard POSIX IO calls under the following benchmarking conditions:

Launch: At least one IOR_POSIX instance per CN.

Create: each IOR_POSIX instance task may create one file with zero size.

Write: each IOR_POSIX instance may write all data to the file and close the file.

Read: Each IOR_POSIX instance may open and read the data in the file it originally wrote and close and delete the file.

Terminate: IOR_POSIX job will terminate.

Each run of IOR_POSIX may execute steps 2-4 above 4 times. IOR_POSIX prints out the read and write rate for each iteration. The figure of merit for IOR_POSIX function ship test is the minimum of the four read and four write rates. The overall function ship I/O rate (R_p) is defined as

$$R_p = \underset{k=1}{\overset{N}{MIN}}(R_k)$$

Where N is the number of IOR_POSIX runs completed in the 30 minute test period.

2.9.1.2 ION to ION Link RDMA Performance (TR-1)

Offeror provided hardware and software may deliver at least 95% of the SAN interface peak link bidirectional bandwidth (0.95*6.4 GB/s = 6.08 GB/s per IBA 4x QDR interface) to the OFED perf test RDMA bandwidth or equivalent test with the message length and number of messages chosen to maximize delivered bandwidth between two ION. For messages of 1MB length (transfer size use by Lustre), message bandwidth may be at least 90% of the SAN interface peak link bidirectional bandwidth (0.90*6.4 GB/s = 5.75 GB/s).

2.9.1.3 ION to Lustre OSS Performance (TR-1)

Offeror provided hardware and software may not inhibit LLNS from utilizing the Lustre LNET self-test from any ION to any LLNS supplied Lustre OSS over the LLNS provided SAN from achieving 85% of the Offeror provided SAN interface peak link bidirectional bandwidth (0.85*6.4 GB/s = 5.45 GB/s). Offeror may work with LLNS to identify any performance bottlenecks or bugs in Offeror provided hardware and software to enable the correct functioning and achieve the performance requirements of this test.

2.9.2 LN & SN High-Availability RAID Arrays (TR-1)

All disk resources for the LN and SN local IO may be RAID5 (or better) controller active-active (as opposed to active-passive) fail-over pair and disk arrays. RAID parity may be calculated on reads as well as writes and the RAID parity read in from disk verified against calculated RAID parity on the data read in from disk. The RAID units and disk enclosures may have high availability characteristics. These may include no single point of failure architecture, dual data paths between the RAID controllers and each disk, redundant fail-over power supplies and fans, at least one hot spare disk per eight RAID chains, hot swappable disks, the capability to run in degraded mode (one disk/RAID string failure), and the capability to rebuild a replaced disk on the fly with a delivered raw I/O performance impact of less than 30% on that RAID chain. There may be system diagnostics capable of monitoring the function of the RAID units and detecting disk or other component failure and monitoring read or write soft failures.

2.9.3 LN & SN High IOPS RAID (TR-2)

The LN and SN RAID5 (or better) arrays will *deliver* at least 500 MB/s aggregate large block read/write bandwidth from the Linux EXT3 filesystem mounted on /tmp and /var/tmp on each LN. The RAID5 arrays will *deliver* at least 640 IOPS to an IO workload randomly reading and writing 4,096B blocks with 50% read and 50% write balance from the Linux EXT3 filesystem mounted on each partition on each LN. Note that the aggregate RAID controller pair and disk arrays performance is 500 MB/s times the number of LN and SN for large block IO and 640 IOPS times the number of LN and SN for 4KiB random IO.

2.10 Management Ethernet Infrastructure (TR-1)

Offeror will propose a management Ethernet (100 BaseT or faster, 1000Base-TX (copper) is preferred) for the system. The management Ethernet infrastructure will provide access to every externally manageable hardware entity (e.g., node, chassis, PDU or rack). In the case of failure in the system interconnect, the management network will be used to boot the entire system. The management Ethernet will be aggregated with high quality, high reliability Ethernet switches with full bandwidth backplanes and provide a single 1000Base-TX (copper) Ethernet (or faster) uplink(s) to the SN. The management Ethernet cables will be bundled within the rack in such a way as to not kink the cables, nor place strain on the Ethernet connectors. All management Ethernet connectors will have a snug fit when inserted in the management Ethernet port on the nodes and switches. The management Ethernet cables will meet or exceed Cat 5E specifications for cable and connectors. Cable quality references can be found at: (http://www.integrityscs.com/index.htm) and (http://www.panduitncg.com:80/whatsnew/integrity_white_paper.asp).

If 1000Base-TX (copper) is offered, then Offeror will provide CAT6 or equivalent management cables. A suggested source of this quality cable is Panduit corporations Powersum+ tangle free patch cords, Part# UTPCI10BL for a 10' cable. The URL for this product is:

(http://www.panduitncg.com/solutions/copper_category_5e_5_3.asp). Management Ethernet reliability is specified in Section 6.1.12.1.

2.11 Early Access to Sequoia Technology (TR-1)

Offeror may propose mechanisms to provide LLNS early access to Sequoia hardware technology for hardware and software testing that includes other steps before inserting the technology into Sequoia. Small additional early access systems are encouraged.

2.12 Sequoia Hardware Options

Offeror shall propose the following MOs, and may propose each of the following TOs, as separately priced options. Offeror may technically describe, in the following sections of its technical proposal(s), how the options will be effected, if exercised by LLNS.

2.12.1 Sequoia Enhanced IO Subsystem (TO-1)

Offeror may propose an enhanced IO subsystem for Sequoia that provides for double the baseline IO performance for jobs spanning 50% of the machine and 25% of the compute nodes. That is, the enhanced IO subsystem proposed may deliver at least 100% of the full system IO delivered bandwidth to jobs using 100% of the CN and may achieve 100% of the full system IO delivered bandwidth for jobs using 50% of the CN and may achieve 50% of the full system IO delivered bandwidth for jobs using 25% of the CN.

2.12.2 Sequoia Half Memory (TO-1)

Offeror may propose Sequoia CN with half the memory of the baseline Sequoia system. In this option, the ION/LN memory may remain consistent with Section 2.3. That is, the memory size component scaling B:F ratio for this CN (only) memory option may meet or exceed:

Memory Size (Byte:FLOP/s) ≥ 0.04

2.12.3 Sequoia14 System Performance (MO)

Offeror shall propose, as a separately priced option, a Sequoia system configuration called Sequoia14 with 70% performance of the baseline. That is, the Sequoia14 system configuration performance may have a peak performance of at least 14.0 petaFLOP/s (14.0×10^{12} floating point operations per second.

2.12.4 Sequoia14 Enhanced IO Subsystem (TO-1)

Offeror may propose an enhanced IO subsystem for Sequoia14 that provides for double the Sequoia14 IO performance for jobs spanning 50% of the machine and 25% of the compute nodes. That is the enhanced IO subsystem proposed may deliver at least 100% of the full system IO delivered bandwidth to jobs using 100% of the CN and may achieve 100% of the full system IO delivered bandwidth for jobs using 50% of the CN and may achieve 50% of the full system IO delivered bandwidth for jobs using 25% of the CN.

2.12.5 Sequoia14 Half Memory (TO-1)

Offeror may propose Sequoia14 CN with half the memory of the baseline Sequoia14 system. In this option, the Sequoia14 ION/LN memory may remain consistent with Section 2.3. That is, the memory size component scaling B:F ratio for this CN (only) memory option may meet or exceed:

Memory Size (Byte:FLOP/s) ≥ 0.04

End of Section 2.0

3.0 Sequoia High-Level Software Requirements (TR-1)

The ASC Sequoia software model and resulting requirements are described from the perspective of a highly scalable system consisting of CN numbering in the range of 30K-60K, ION numbering in the range of 128-1,024 and LN numbering in the range of 4-64 and SN numbering in the range of 1-8. Thus, the scalability and functionality requirements for these classes of nodes are vastly different. In addition, key software model architectural choices must be hierarchal and scalable. Scalability and reliability dictates less is more on the CN with function shipping of complex OS functions to an ION. Conversely, RAS infrastructure requires accurate and timely information about the hardware, software and applications from every component in the system. Thus, the Sequoia system model is required as a Light-Weight Kernel (LWK) with minimal functionality with extremely low noise on the compute nodes and a full function "Linux-like" OS on the ION, LN and SN with additional, possibly unique additional system services on each. A full featured "Linux-like" OS on the CN is a possible alternative, only if the additional functionality does not destroy the overall system scalability, reliability and performance (from an application perspective).

All offered software components may be Open Source.

3.1 LN, ION and SN Operating System Requirements

The following requirements apply only to the Sequoia system LN, ION and SN.

3.1.1 Base Operating System and License (TR-1)

Offeror may provide a standard multiuser Linux standards base specification V3.1 or later compliant interactive operating system (http://www.linux-foundation.org/spec/). The base operating system is designated as BOS and may provide at least a basic kernel that supports system services and multiprocessing applications. Fully supported kernel-level implementation, as defined by the POSIX 1003.4 (or later) working group standard of thread operations in shared address spaces may also be provided (within six months of standardization or at Sequoia delivery). The operating system may provide mechanisms to share memory between user processes and to run OS threads within a single user process on multiple cores and/or hardware threads from a core or multiple cores simultaneously. This may include provision of right-to-use license for an unlimited number of users, including unlimited concurrent usage, of the operating system, daemons, and associated utilities. LLNS will accept the Offeror's self-certification for POSIX compliance.

3.1.1.1 Base OS Compliance (TR-2)

The proposed operating system will have the Linux Standards Base (LSB) 3.1 or later certification. The Offeror will deliver a copy of the certificate of compliance with the system delivery.

3.1.2 Function Shipping From LWK (TR-1)

The BOS on the ION may support function shipped OS calls from the LWK as described in Section 3.2.2. If BOS function ship IO support includes buffered IO, then this feature will have system administrator configurable buffer lengths. BOS will automatically flush all user buffers associated with a job upon normal completion or explicit call to "abort()" termination of the job. BOS will also support for job invoked flushing of all user buffers.

3.1.3 Remote Process Control Tools Interface (TR-1)

As part of the petascale code development tools infrastructure described in Section 3.7.1, the BOS proposed for the ION, may provide a secure Remote Process Control code development Tools Interface (RPCTI) that enables a code development tool daemon to control processes and threads running on their associated CN. This interface may model a well-known serial process control interface such as *ptrace* or */proc*. Alternative to an interface implemented as system or library calls, a message passing style is also acceptable in which a tool daemon exchanges process control messages with an ION system daemon in a compact binary format. In either case, however, the latency of the interface may be low.

3.1.4 OS Virtualization (TR-3)

If Offeror proposes to virtualize the operating system or services into multiple OS images on a node, then the virtualization mechanism may support the allocation of the node resources so that all IO devices, sockets, cores and physical memory can either be virtualized and shared among all OS images or statically allocated to a specific OS image and made invisible to the other OS images. In addition, booting of individual OS images may be independent. Each node may be able to have different versions or patch levels of the OS and other supplied software running in a virtual environment.

3.1.5 Multi-Boot Capability (TR-1)

The node operating system may have the ability to boot from at least ten different environments. Switching between the ten boot environments may be accomplished by the root user issuing commands from the shell prompt and rebooting the node. No manual hardware reconfiguring may be required to switch boot environments. Once running a boot environment, it may be possible to apply system installs, patches and configuration changes to both the active and the non-active boot environment. The supplied operating system may share (reuse) the swap and local /tmp space in each of the ten boot environments. Other required file systems (e.g., /) may be replicated.

3.1.6 Pluggable Authentication Mechanism (TR-1)

Offeror may provide a service programming interface (SPI) that allows the replacement of the standard authentication mechanism with a LLNS provided pluggable authentication mechanism (PAM). The SPI may be supported by all Offeror supplied login utilities and authentication APIs. The purpose of this PAM is to allow LLNS to meet changes in DOE security requirements and LLNS to implement stronger authentication (e.g., one-time password authentication).

3.1.7 Node Fault Tolerance and Graceful Degradation of Service (TR-2)

The node operating system may have the ability to detect, isolate and manage hardware or software faults in a way that minimizes the impact on overall system availability. When system (hardware or software) components fail, the node software resources may provide degraded system availability. Under most circumstances, it may be possible to take hardware and software components off-line or bring them back on-line without operating system rebooting. The probability that a job will fail (due to hardware or software faults) should be proportional to the amount of resources consumed by the job, not Sequoia system size.

3.1.8 Networking Protocols (TR-1)

The operating system may support the Open Group (C808) Networking Services (XNS) Issue 5.2 (http://www.opengroup.org/pubs/catalog/c808.htm), or later, standard networking protocol suite over the network interfaces described in requirement 2.9.2. In particular, over these interfaces the IPv4 (http://www.ietf.org/rfc/rfc0791.txt), IPv6 (http://www.ietf.org/rfc/rfc2460.txt, http://www.ietf.org/rfc/rfc4213.txt), TCP/IP, UDP, NIS, NFSv4 (client and server, http://www.ietf.org/rfc/rfc3530.txt), RIP, telnet, ssh, and ftp protocols may be supported. If selected, Offeror may need to provide a rational basis for claiming IPv6 compliance and interoperability with IPv4. Meeting IPv6 Ready branding is sufficient.

3.1.9 OFED IBA Software Stack (TR-1)

Offeror may provide and support a fully compliant InfiniBand Architecture (IBA) V1.2 (http://www.infinibandta.org/specs) software stack. Offeror's IBA software stack may be fully functional, stable and scale to the SAN size LLNS will provide. Offeror may supply and support OpenFabrics Enterprise Distribution (OFED) version 1.3, or then current, IBA compliant software stack. The supplied and supported OpenFabrics software stack may be certified as OpenFabrics compliant after successfully passing the OpenFabrics compliance test suite and being released by the OpenFabrics Alliance. (PUT COMPLIANCE URL HERE)

Offeror may contribute all modifications to the OFED software stack to the OpenFabrics Alliance throughout the lifetime of this procurement. Offeror may document and track all their OFED software stack bugs in the OpenFabrics Alliance bugzilla bug tracking system (https://bugs.openfabrics.org/).

3.1.10 IBA Upper Layer Protocols (TR-1)

Offeror's provided and supported OFED stack releases may also include the following Upper Layer Protocols (ULP):

- IPoIB, http://www.ieft.org/html.charters/OLD/ipoib-charter.htmlhttp://www.datcollaborative.org/kdapl.html
- SRP, http://www.t11.org/t10/drafts/srp/srp-r16a.pdf
- iSCSI, http://www.ietf.org/rfc/rfc3720.txt
- iSER, http://www.rdmaconsortium.org/home
- NFS-RDMA, http://www.ietf.org/rfc/rfc3010.txt
- IPoIB connected mode, http://www.ietf.org/internet-drafts/draft-ietf-ipoib-connected-mode-00.txt

These protocols may fully implement and conform to the above specifications. Offeror's OFED ULPs may successfully pass all relevant tests in the OpenFabrics compliance test suite.

3.1.11 Local File Systems (TR-2)

The BOS local file system may have a POSIX interface that is 64b by default and will support individual files of at least ten (10.0) GB in size. The local file system may support individual file systems of at least eight (8.0) TB in size. The file systems may support increased reliability and fast reboots (e.g., reduce the FSCK time via a journal implementation). That is, the file system may be designed and implemented so that any file system initialization that delays system reboots or file system restarts/mounts may have at most logarithmic complexity in the number of

devices and files/directories. The aggregate file system initialization and file system restarts/mounts may be less than five (5.0) minutes for all the proposed node local file systems. The local file system may have a logical volume manager that allows the striping of all local file systems (including the root or /, /swap, /usr and /var) across multiple disks in order to maximize performance. The logical volume manager may be able to migrate directory structures and associated files to different physical devices and add/subtract disk blocks to a file system. The local file system may support multi-boot capability (section 3.1.5) by being able to mount all the partitions of the other boot environment. The provided file system and logical volume manager may have a utility that will scan the file system metadata and data disk blocks and repair damage to the file system while the file system is mounted for normal usage.

3.1.12 Operating System Security (TR-2)

Offeror may provide security functionality where access to the system may be controlled by identifying and authorizing the user or by checking the validity of forwarded credentials. All users may be authenticated before access is permitted. Successive logon attempts may be controlled by denying access after multiple (maximum of 5) unsuccessful logon attempts by the same user.

3.1.12.1 Login Information (TR-2)

Users may be notified upon successful login of the following information: date and time of last successful login; and where the operating system provides the capability, number of unsuccessful attempts.

3.1.12.2 Audit Capability (TR-1)

A record of each user login and logoff may be maintained. In addition, the following information may be maintained as an audit record: use of authentication changing procedures; unsuccessful logon attempts; and blocking of a user, and the reason for the blocking.

3.2 Light-Weight Kernel and Services (TR-1)

The following requirements apply only to the operating system kernel running on system CN. The purpose of the CN Light-Weight Kernel (LWK) is to implement the “Livermore Model” for petascale applications with an extremely reliable, diminutive runtime overhead and OS noise environment to enable highly scalable MPI applications running on a large number of CN with multiple styles of concurrency within each MPI task. Therefore, the LWK may feature minimal complexity, without support for any more services than necessary to implement the required functionality.

3.2.1 LWK Livermore Model Support (TR-1)

The proposed LWK (with support from the IONK) may support the “Livermore Model” for petascale applications as follows. An application is a set of binaries launched as a single (Multiple Program, Multiple Data) MPMD job on a specified and fixed number of CN, and with a specified and fixed number between 1 and NCORE MPI tasks per CN. Specifically, different CN may run different binaries executables, specified at launch time.

MPI (and any exposed native packet transport libraries) are the only means of inter-CN task to task communication within a job.

A job may specify the LWK kernel, or kernel version, to boot and run the job on the CN.

All tasks on a single CN may be able to dynamically allocate memory regions that are addressable by them all. Allocation may be a collective operation among a subset of the tasks on a CN. A shared memory region may be freed by each task separately: the region is deallocated and available for reallocation when all tasks that allocated it have freed it.

An MPI task may be threaded, but with restrictions designed to prevent the need for any pre-emptive thread scheduling in the kernel.

Each MPI task is statically associated with one or more cores of a node. The task's kernel threads run only on the cores the task is associated with.

There is a fixed maximum number of active kernel threads per task, but no more than the number of hardware threads supported by the cores associated with the task.

Each kernel thread is statically associated with a particular core, and no more threads are associated with any core than the number of hardware threads the core is designed to support.

An MPI task can dynamically load libraries via `dlopen()` and related library functions.

An MPI task can freely alternate among several threading models, particularly at call and return points in the code.

Single threaded: The MPI task may be single threaded; in this case, the single thread is permitted to make MPI communication and synchronization calls.

Pthreads: The Pthread interface should be supported, but with a cap on the number of threads that can be created that is consistent with the 4th item, above. MPI calls are permitted from each Pthread, with the programmer responsible for making sure only one thread calls `MPI_INIT()` and `MPI_FINALIZE()`.

OpenMP: OpenMP threading must be supported, again in a manner consistent with the above. MPI calls are permitted in the serial regions between parallel regions. MPI calls are not permitted in the OpenMP parallel loops and regions.

SE/TM: The code is written to be sequential, although perhaps with "hints" to the compiler and kernel as to how threads or transactions might be recognized and synchronized at run time. The kernel, the compiler, and the hardware cooperate to speculatively execute threads or transactions without locking, using instead the ability to abort thread activity or transactions and possibly to re-execute them if a synchronization conflict arises. No MPI calls are permitted in the SE/TM regions, but are permitted in the serial regions.

Kernel threads within an MPI task may be able to synchronize without the use of kernel calls.

3.2.2 LWK Supported System Calls (TR-1)

Offeror may propose a LWK that may be compatible with the BOS on the ION. The LWK may support at least the following system calls, either as traps or through library wrappers:

exit, read, write, open, close, link, unlink, chdir, time, chmod, lchown, lseek, getpid, getuid, alarm, utime, access, kill, rename, mkdir, rmdir, dup, dup2, times, brk, getgid, getuid, geteuid, getegid, fcntl, umask, getppid, sigaction, setrlimit, getrlimit, getrusage, gettimeofday, symlink, readlink, mmap, munmap, truncate, ftruncate, fchmod, fchown, statfs, fstatfs, socketcall, setitimer, getitimer, stat, lstat, fstat, fsync, sigreturn, clone, uname, sigprocmask, llseek, getdents, readv, writev, sysctl, sched_yield, nanosleep, chown, getcwd,

truncate64, ftruncate64, stat64, lstat64, fstat64, getdents64, fcntl64, futex, set_tid_address, exit_group, execve

The LWK should support all arguments and behavior of these calls as in Linux, except for arguments and behaviors that are exclusively used to support functionality that should be omitted from the LWK, as described below (e.g. no preemptive thread scheduling, no `fork()`, `exec()` prior to `MPI_Init()`, etc.)

In addition, Offeror may propose a LWK that extends Linux with specific syscalls to fetch the MPI node rank mappings as well as node specific personality data (coordinates, etc).

All I/O and file system calls may be implemented through a function-shipping mechanism to the associated ION BOS, rather than directly implemented in the LWK. All file IO will have user configurable buffer lengths. LWK will automatically flush all user buffers associated with a job upon normal completion or explicit call to "abort()" termination of the job. LWK will also have an API for application invoked flushing of all user buffers.

3.2.3 LWK Job Launch (TR-1)

The proposed LWK running on CN may support the launching and running of ASC applications based on multiple languages, including Python as well as the Linux/Unix OS proposed for the LN, SN or ION does. Python applications use dynamically linked libraries and SWIG (www.swig.org) and f2py generated wrappers for the Python defined API for the ability to call C, C++ and Fortran03 library routines.

Comment [CU4]: This sentence isn't exactly correct. Python itself defines an API to interface C/C++ code (compiled into DLLs) into Python code. SWIG just provides an automated way to generate wrappers for existing C and C++ code using this API. Also, SWIG doesn't currently wrap Fortran code (although it may plan to). A wrapper for Fortran that does exist is f2py which is part of numpy (http://www.scipy.org/F2py)

3.2.4 Diminutive Noise LWK (TR-1)

In order to support petascale ASC applications running effectively on the aggregate of CNs, the proposed LWK may provide applications with a diminutive noise environment. The LWK has a diminutive noise environment if the threaded FWQ benchmark, described in section 9.1.2.3, run on the LWK of representative CN produces produces time samples required to accomplish a fixed work quanta with scaled noise mean of less than 10^{-6}, and standard deviation of less than 10^{-3} and Kertosis of less than 10^{2}.

3.2.5 LWK Application Remote Debugging Support (TR-1)

The proposed LWK may allow the remote debugging interface to function on user applications as described in Section 3.7.1.4. In addition, the overhead with implementing these functions may allow the remote debugging interface latency for basic operations to be below that specified in Section 3.7.1.3.

3.2.6 LD_PRELOAD Mechanism (TR-2)

Offeror may propose LWK functionality equivalent to Linux LSB 3.2 (or later) LD_PRELOAD mechanism. This mechanism may allow the LWK dynamic loader to load an LLNS supplied interposition agent into the address space of a target process prior to loading any other libraries. This mechanism may provide the LLNS provided memory tools with the functionality to interpose tracking functionality in the `malloc()` and `free()` libc functions in order to detect memory leaks and memory access errors, including ones that occur in the system-provided software such as libc.

3.2.7 LWK Limitations (TR-1)

The features excluded from a LWK are as important as those implemented features: with OS sometimes *less is better*! These exclusions may allow the general performance of the kernel to be

improved, the performance noise level to be reduced, and the reliability to increase. The following list of features may *not be supported* in the LWK.

preemptive thread scheduling or time slicing: Consistent with the "Livermore Model," threading may be supported, but with the constraint that there need never be more threads on a node than there are hardware threads to execute them, so no preemptive scheduling mechanism is needed. This, of course, refers to kernel created threads used in the context of Pthreads, OpenMP and speculative multithreading or transactional memory. It does not apply to application level threads that are invisible to the kernel.

demand paging to and from disk: Dynamic address translation from virtual to real addresses may be supported. However, demand paging to and from secondary storage may not be supported.

TLB misses: Although dynamic address translation may be supported, the TLB mapping registers may be managed statically, so that ASC Application do not experience any TLB misses while executing on the CN.

dynamic task/process creation: All processes on a node will be created at job launch time. There may be no support for dynamic process creation (`fork()` and `vfork()`). There may be no support for the dynamic task creation parts of MPI 2.0 that would require them.

interprocess communication: Most interprocess communication mechanisms between MPI tasks on the CN may not be supported. Only MPI, low-level native packet transport libraries, and shared memory regions within a node, may be permitted. Classical Unix pipes may be excluded, as may both interprocess signals and IP communication directly between processes on the compute nodes. However, IP communication between an MPI task on a compute node and a process on an I/O node or another host, and also signals between the I/O and compute nodes, may be supported via function shipping to the I/O nodes.

The exec() family: Unrestricted `execve()` calls may not be supported (since they would disturb MPI). But one exec-type call should be permitted by each MPI task, as long as it is executed prior to the `MPI_INIT()` call.

3.2.8 RAS Management (TR-1)

The proposed LWK (or node BIOS/microcode) may report all RAS events that the hardware encounters to the RAS database server in the SN. Along with the type of event that occurred, the LWK may also gather relevant information as appropriate to help isolate or understand the error condition. The reporting RAS events may be accomplished by CN sending messages to the SN via the management network. This approach requires active polling by the SN to extract the message from the CN.

3.2.9 LWK 64b HPM Support (TR-1)

The proposed LWK may provide individual threads within MPI tasks access to the 64b hardware performance monitors (section 2.4.10) on the CN hardware that handles overflow and saturation of those 64b counters. This functionality may allow applications to accurately sample the counters via PAPI Version 4 or later (Section 2.4.10) with no need for additional processing to prevent counter overflow.

3.2.10 Application Checkpoint and Restart (TR-2)

The proposed CN operating environment may support reliable application-initiated checkpoint and restart of parallel MPI applications. Offeror proposed API for checkpointing and restarting an application may read, reset and save to the application checkpoint, the checksums calculated

for all data injected into the CN interconnect. Offeror may propose a command line utility usable by standard BOS user accounts on the LN that will read a series of checkpoint files from an application with multiple restarts and overlapping computations to verify that the link checksums from the multiple overlapping computations are the same. Upon detecting a difference between checksums the utility may indicate the CN(s), link(s) and interconnect components(s) with the errors. In addition, Offeror may propose a checksum API that allows user applications to read the network checksums in between checkpoint operations and automatically save these checksums to reserved memory that is later saved to a the next checkpoint file by the Offeror proposed checkpointing software in the next checkpoint operation. The checksum API may also provide the user application with the ability to specify the number ≥0 of bytes at the beginning of the packet to ignore in the network checksumming calculation. Offeror proposed command line utility that reads a series of checkpoint files may be capable of utilizing the multiple checksums saved by an application calling the checksum API multiple times in between successive checkpoints.

3.2.11 LWK "RAM Disk" Support (TR-2)

The proposed LWK may provide a file system interface to a portion of the CN or ION memory (i.e., a "RAM disk" in CN or ION memory or a region of CN memory accessible via MMAP interface). The "RAM disk" may be read and written from user applications with standard POSIX file I/O or MMAP functions using the "RAM disk" mount point. The "RAM disk" file system, files and data may survive application abnormal termination, restarts and thereby permit the restarted application to read previously written application restart files and data from the "RAM disk." The "RAM disk" file system may allocate memory to be used when data is written to a "RAM disk" file and return the memory when any "RAM disk" resident file is deleted. This "RAM disk" would be used to provide very fast application checkpointing and to aggregate I/O before it is written to the Lustre global file system.

3.3 Distributed Computing Middleware

3.3.1 Kerberos (TR-1)

Offeror may provide the Massachusetts Institute of Technology (MIT) Kerberos V5 reference implementation, Release1.6 or later, client software on the proposed system. This may include a fully supported integrated login mechanism, including a Kerberos V5 PAM module, that may support the use of password authentication and Kerberos V5 ticket authentication against a MIT Kerberos authentication server. Support for Public Key Cryptography for the initial authentication in Kerberos (PKINIT) may also be provided as a PAM module. This mechanism may comply with the authorization policies established for MIT Kerberos principals (e.g., password lifetime, account lockout) and may be capable of acquiring and storing the user's Kerberos credentials for a login session. This mechanism, supplied by the Offeror, along with the following login utilities and daemons- rsh, rcp, rlogin, ssh, scp, sftp and ftp, may be fully interoperable with the login utilities and daemons in Kerberos V5 Release 1.6, or later, as distributed by MIT or in OpenSSH V2 distributions for ssh, scp, sftp.

3.3.2 LDAP Client (TR-1)

Offeror may provide LDAP version 3 (or then current) client software on each ION, SN and LN in the proposed system. This may include the use of SASL/GSSAPI for authentication and SSL for integrity and privacy with support for, but may not be limited to, Kerberos V5 as a security mechanism. The supplied command-line utilties- ldapsearch, ldapmodify, ldapdelete, ldapadd

and client libraries may be fully interoperable with an OpenLDAP Release 2.4 or later LDAP server. The Offeror may provide directory service integration software to enable UNIX C library calls that perform user and group queries (e.g., getpwnam(), getpwuid(), getpwent(), getgrnam(), getgruid(), getgrent()) to obtain this information from an LDAP directory via a caching daemon and client service library. This may include support for IETF RFC 2307 (http://www.ietf.org/rfc/rfc2307.txt) and support a mechanism to map object classes and attributes as well as perform rules-based data transformations and filtering on attribute values or sets of attribute values.

3.3.3 NFSv4.1 Client (TR-1)

Offeror may provide NFS version 4.1 (http://www.ietf.org/internet-drafts/draft-ietf-nfsv4-minorversion1-21.txt) client software with the BOS. This may include the use of RPCSEC_GSS for authentication, integrity and privacy with support for, but should not be limited to, Kerberos V5 as a security mechanism. This may include a fully supported NFSv4.1 ACL mechanism with ACL editing utilities. The ACL mechanism may provide support for users and groups in a multi-domain environment (i.e., recognize identity@domain identifiers). Offeror may support mapping NFSv4 name identifiers to UNIX UID and GID values. This may include providing directory service integration software for obtaining this information from an LDAP directory via a caching daemon and a client service library.

3.3.4 Cluster Wide Service Security (TR-1)

All system services including debugging, performance monitoring, event tracing, resource management and control may be performed using a secure authentication and authorization protocol that interfaces to the PAM (Section 3.1.6). This protocol may be efficiently and scalably implemented so that authentication and authorization step for any size job launch is less than 50% of the total job launch time.

3.4 System Resource Management (SRM) (TR-1)

The overall System Resource Management (SRM) requirement is to integrate Sequoia into the existing LLNS SCF enterprise-wide job management system based upon Moab[6] and SLURM[7]. Moab is a highly configurable policy-based intelligence engine that integrates scheduling, managing, monitoring and reporting of cluster workloads across multiple computers and sites. Moab relies upon resource managers on the individual clusters to manage the cluster's resources and jobs. SLURM is a highly scalable open source resource manager in use on hundreds of the largest computers in the world. It provides three key functions within an individual clusters. First, it allocates resources to users for some duration of time so they can perform work. Second, it provides a framework for initiating and managing work, typically a parallel job, on the set of allocated resources. Finally, it arbitrates conflicting resources by managing a queue of pending work.

In order for SLURM to provide resource management on Sequoia interface requirements between SLURM and Offeror's proposed software are detailed below.

[6] www.clusterresources.com

[7] www.llnl.gov/linux/slurm

3.4.1 SRM Security (TR-1)

SRM components and communications between components must be secure: users can only see and manipulate their applications and data and SRM components may not run as the "root" user account. User identities may be maintained throughout the chain of SRM components without giving users login capability directly on ION or SN BOS.

3.4.2 SRM API Requirements (TR-1)

Offeror's proposed APIs may not write to STDOUT/STDERR, but may provide documented status codes. Offeror's APIs may be reliable so that they complete successfully or return correct error codes with no more than one in ten thousand (1 in $1x10^4$) attempts failing. The APIs may be thread-safe. The APIs speed is important and all APIs may return within 10 milliseconds (polling or event triggers can be used if more time is required to complete any API function). All APIs may be usable from an I/O or service node. No SLURM damon may execute on the compute nodes. Documentation may be provided for the APIs.

3.4.3 Node Reboot API (TR-1)

Offeror may provide APIs to reboot CN and ION. The API may provide the ability to reboot individual or groups of ION and the corresponding CN. The API may provide the ability to reboot individual or groups of ION without requiring the reboot of the corresponding CN. The APIs may control the LWK image to be loaded on the CN. Use of this API may be restricted so that normal users cannot write programs that use the services provided by the API.

3.4.3.1 Node "RAM Disk" API (TR-2)

Offeror may provide API that allows configuration of LWK "RAM Disk" as specified in Section 3.2.11, if bid. This API may allow turning the "RAM Disk" feature on and off and if it is turned on the specification of "RAM Disk" size and mount point. This API may also allow clearing of the "RAM Disk."

3.4.4 Network Topology API (TR-1)

Offeror may provide APIs to determine the network topology connecting CN. This information will be used by SLURM in order to determine optimal resource allocations for pending jobs.

3.4.5 Job Manipulation Commands and API (TR-1)

Linux command line utilities and APIs may be available to reliably manipulate a job as a single entity: including kill, modify, query characteristics, and query state. Offeror's commands and API may be reliable so that they complete successfully with all tasks of the job being having been correctly manipulated by the command or API or return correct error codes with no more than one in ten thousand (1 in $1x10^4$) calls to the API failing or failing to correctly manipulate all tasks in the job.

3.4.6 Job Signaling API (TR-1)

Offeror may provide APIs to send an arbitrary signal to SLURM specified individual or groups of user tasks. Signal delivery may be reliable so that every task in the SLURM specified group of user tasks receives the signal and executes the correct signal handler with the nominal results with failure less than 1 in $1.0x10^4$ calls to the API. In particular, SIGKILL may reliably terminate any task. Use of the API may be restricted to prevent a user from signaling another user's job.

3.4.7 User Task Launch API (TR-1)

Offeror may provide APIs to launch user tasks on CN. The APIs may provide the capability of executing different applications with different arguments for each task. The APIs may provide the capability of launching specific tasks on SLURM specified core(s) within a node and binding tasks to SLURM specified core(s). Job launch time may vary by no more than the log of the task count. Job sizes up to one task per CN core will be supported. Launching tasks into a stopped state may be supported for debugging. Use of the API must be restricted so that users may not have the ability to launch tasks on resources that have not been allocated to them. Task launch time for jobs with small binaries may not exceed 3 seconds for 8,192 tasks.

3.4.8 User Task Connectivity API (TR-1)

Offeror may provide APIs to establish connectivity for application programs to make use of the CN interconnect for their communications. Use of the API may be restricted so that users must not have the ability gain access to other jobs' communications.

3.4.9 SRM STDIO (TR-1)

Offeror may provide SRM APIs that allow SLURM to distinguish during job launch between and identify STDOUT and STDERR for each user MPI task in a job. This API may allow SLURM to send data to STDIN of each user MPI task in a job.

3.4.10 System Initiated Checkpoint API (TR-3)

Offeror may provide APIs to checkpoint a parallel job. The API may provide support for creating a checkpoint and either continuing execution or terminating. Use of the API may be restricted to prevent a user from checkpointing another user's job. Offeror may provide APIs another mechanism of restarting a previously checkpointed job. Use of the API and/or checkpoint file permissions may be restricted to prevent a user from restarting another user's job.

3.4.11 Predicting Failed Nodes (TR-2)

Offer may provide an API to provide a list of CN and ION that are predicted to fail within the SLURM specified period of time. This facility will be used by SLURM to drain CN from the available pool and prevent queued jobs from running on them until CN or ION was repaired.

3.5 Integrated System Administration Tools

3.5.1 Single Point for System Administration (TR-1)

Offeror may provide a set of facilities to administer the Sequoia CN, ION, SN and LN as a single entity. In particular, Offeror may provide fully supported implementation of a single-point system administration tool to effect configuration actions on: file system mounts; node booting; node status; node self-consistency checks of system configuration parameters; software installation; resource administration; node shutdown/restart; system patch installation; login control (provide capability to restrict login access to certain processors, and cluster-wide monitoring of failed login attempts by an individual); and system back-ups, including ability to dump multiple volumes of tapes without operator intervention. This single-point of control may provide a command-line interface that effects one or more actions from each command issued with error return code allowing the system administrator the ability to script (automate) redundant configuration tasks for multiple or all nodes in the system. This command-line interface may be capable of performing all of the above system administration configuration actions. The Offeror may provide a fully supported implementation of mechanisms for detecting

and reporting failures of critical resources, including processors, network paths, and disks. The diagnostic routines may be capable of isolating hardware problems down to the FRU level in both the system and its peripheral equipment.

3.5.2 System Admin (TR-1)

CN, ION diskless environments may be installed and maintained on the SN. Multiple CN and associated ION environments may be selectable on a per boot basis. Installing CN, ION diskless environments may not require patching source code nor compiling from source code. Offeror provided system administration utilities may allow the boot/reboot of individual or groups of ION and associated CN together or just the ION separately (without rebooting CN). Reboot of BOS and LWK may not be required for normal day-to-day operations (e.g., changing configuration files).

3.5.2.1 Fast, Reliable System Reboot (TR-1)

Rebooting the entire system will take less than fifteen (15) minutes and once initiated may not require human intervention. This time will include the time to reboot the nodes, switches, mount any local file systems (if applicable) and return all system daemons to operating condition. This system reboot time specifically does include the time to unmount idle (no pending IOs, no open files or file locks active) remote file systems (including the NFS and Lustre file systems), but does not include the time to mount remote file systems.

3.5.2.2 Multi Configuration Boot, Install and Patch (TR-1)

The system may have the ability to boot ten (10) alternate system software release and/or configurations. This may be used to test new system releases in “debug shots” or provide multiple kernels for CN. Switching between these any alternative system software release may be accomplished with a single system reboot and take less than ninety (90) minutes including reboot time (Section 3.5.2.1). It may be possible to patch any system software release and/or configuration. It may be possible to back out any patches applied to any system software release and/or configuration. Installing, upgrading, and patching (applying or backing out) any configuration that is not active may be accomplished with the system on-line and under user workload and may take less than eight (8) hours for installs and upgrades, and two (2) hours for patches. This includes any system reboots.

3.5.3 System Debugging and Performance Analysis (TR-2)

Offeror may provide a set of facilities with a single-point of control to analyze the entire system performance and make tuning modifications. In particular, the Offeror may provide fully supported implementation of a single-point of control system tuning tool to dynamically monitor and modify the following system attributes: processor status; key resources: system CPU usage, memory usage, page faults; run queues per node; scheduling priority of each process and each thread within a process; and current system configuration. The tuning parameter changes may take affect without requiring an operating system reboot. This single-point of control may require root access to make modifications, but only normal user privileges to monitor the system. Due to the large number of system attributes and components, this single-point of control may be constructed to be fast and efficient when monitoring and modifying the entire system. All system information and control functions may be presented in a hierarchical fashion.

3.5.4 Scalable Centralized Resource Data Base (TR-2)

Offeror may provide an Open Source SQL compliant scalable centralized resource data base (CRDB), keeping track of the state of all system resources, their current usage policies, and a

system error log. The schema used by the CRDB for storing the data will also be available as Open Source license. This facility and the system utilities/functions that depend on it, may be constructed so that the CRDB does not become a single point of failure or contention (bottleneck) within the system. In particular, SQL updates to the CRDB and SQL queries from the CRDB during system changes impacting at least 50% of the nodes (e.g., rebooting, major system disruptions) may be done in parallel so as to not impede rapid system transitions. The degree of parallelism supported in the CRDB may be a system tunable parameter.

3.5.5 User Maintenance (TR-2)

Offeror may provide a secure (only root access) tool for managing user administration, including some means of integrating the namespace manager and the authentication server in order to facilitate adding, removing, and modifying users. In addition, the Offeror may provide a tool for managing groups, including initial creation of groups, modification of groups, and user membership in groups. Offeror provided user administration tools may allow/disallow user accounts on CN, LN, SN and ION separately. These tools may provide a scriptable interface and may not require human interaction with a GUI to perform any functions.

3.5.6 Login Load Balancing Service(TR-2)

In order to balance the user logins across the LN, Offeror may propose hardware and software to route individual user logins to different LN for each successive login attempt. Proposed solution should integrate with LLNS 1/10 GbE infrastructure and allow site-specific policies for choosing load balancing algorithms.

3.6 Parallelizing Compilers/Translators

3.6.1 Baseline Languages (TR-1)

Offeror may provide fully supported implementations of Fortran 2003 (ISO/IEC 1539-1:2004, ISO/IEC TR 15580:2001(E), SO/IEC TR 15581:2001(E), ISO/IEC TR 19767:2005(E)) see URL: http://www.nag.co.uk/sc22wg5/IS1539-1_2003.html, C (ANSI/ISO/IEC 9899:1999; ISO/IEC 9899:1999 Cor. 1:2001(E), ISO/IEC 9899:1999 Cor. 2:2004(E)) see URL http://www.open-std.org/jtc1/sc22/wg14/www/standards, and C++ (ANSI/ISO/IEC 14882:1998, ISO/IEC 9945-1:1990/IEEE POSIX 1003.1-1990; ANSI/ISO-IEC 9899-1990 C standard, with support for Amendment 1:1994) see URL: http://www.open-std.org/jtc1/sc22/wg21/docs/standards, and Python Version 3.0 or later as released by http://www.python.org . Fortran03, C, C++ and Python are referred to as the baseline languages. In addition, an assembler may be provided. Offeror may provide the fully supported capability to build programs from a mixture of the baseline languages (i.e., inter-language subprocedure invocation may be supported).

3.6.2 Baseline Language Optimizations (TR-1)

Offeror may provide baseline language compilers that perform high levels of optimization that allow the application programmer to utilize of all CN supported hardware features such as SIMD, vectorization, programmable memory prefetch, transactional memory, software managed memory and speculative execution directly in the baseline languages.

3.6.3 Baseline Language 64b Pointer Default (TR-1)

Offeror may provide compilers for the baseline languages that are configured with the default mode of producing 64b executables. A 64b executable is one with all virtual memory pointers having 64b. All operating system calls may be available for use by 64b executables. All Offeror

supplied libraries may provide 64b objects (versions of the API). Offeror's supplied software may be fully tested with 64b executables.

3.6.4 Baseline Language Standardization Tracking (TR-1)

Offeror may provide a version of the baseline languages that is standard compliant within eighteen months after ANSI or ISO/IEC standardization, whichever occurs earlier. Offeror is encouraged to adhere to the current proposed standard.

3.6.5 Common Preprocessor for Baseline Languages (TR-2)

Offeror may provide the capability of preprocessing ANSI C preprocessor directives in programs written in any of the baseline languages.

3.6.6 Base Language Interprocedural Analysis (TR-2)

Offeror may provide mechanisms to perform basic interprocedural analysis (e.g., variable cross-reference listing, COMMON block analysis, use/def analysis) for programs written in the baseline languages.

3.6.7 Baseline Language Compiler Generated Listings (TR-2)

Offeror may provide baseline language compiler option(s) to produce source code listings that include information such as pseudo-assembly-language listings, optimizations performed and/or inhibitors to those optimizations on a line-by-line, code block-by-code block or loop-by-loop basis as appropriate, and variable types and memory layout.

3.6.8 C++ Functionality (TR-2)

Offeror may provide an implementation of the ISO/IEC 14882 C++ standard compiler including: member function templates, partial specialization of classes, partial ordering of functions, name spaces including std::namespace for standard C++ libraries, and default template parameters. Standard C++ library including Standard Template Library and header files without ".h" extensions.

3.6.9 Cray Pointer Functionality (TR-2)

Offeror may provide Cray style pointers implemented in an ANSI X3.9-1977 Fortran compliant compiler.

3.6.10 Baseline Language Support for the "Livermore Model" (TR-1)

All the proposed baseline languages may support the "Livermore Model" by providing programmers the ability to produce MPI parallel programs that can exploit multiple cores and hardware threads with at least the multiple styles of single node parallelism within the MPI tasks described in the subsections below. These multiple styles of single node parallelism may nest. To efficiently support this nesting of parallel styles, the Offeror's runtime support may repurpose a fixed number of software threads between the different styles of parallelism with the restriction that only one master thread executes the subroutine call/return between packages written with different styles and the other helper threads call special routines indicating that they can be repurposed. Special hardware and runtime software mechanism are required for efficient implementation of thread repurposing. The overhead associated with repurposing may be less than a subroutine call/return.

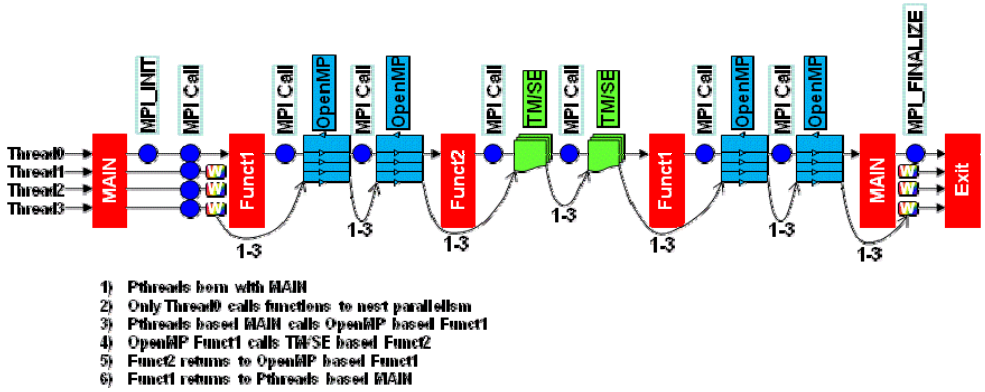


Figure 3-1: Unified Nested Node Concurrency.

3.6.10.1 Baseline Language Support for OpenMP Parallelism (TR-1)

All the baseline languages (i.e., Fortran03, C, C++ and Python) compilers or interpreters may support node parallelism through OpenMP Version 3.0 or then current directives or language constructs (http://www.openmp.org). As an optimization feature, all the baseline language compilers may perform automatic parallelization. The baseline language compilers may produce symbol tables and any other information required by the debugger to enable debugging of OpenMP parallelized ASC applications.

3.6.10.1.1 OpenMP Performance Optimizations (TR-2)

The baseline languages and runtime library support for the CN may include optimizations that minimize the overhead of locks, critical regions and self-scheduling "do-loops" by utilizing special hardware features of the CN hardware.

3.6.10.1.2 OpenMP Performance Interface (TR-3)

The baseline languages may implement the portable OpenMP performance interface as specified in the white paper adopted by the OpenMP Forum (see http://www.openmp.org/blog/resources/#White%20Papers; alternatively direct link to the white paper is http://www.compunity.org/futures/omp-api.html). The baseline languages may provide proper decoding and demangling of instructions and identifiers to support the mapping of results back to source code with respect to Fortran03 modules and C++ namespaces.

3.6.10.1.3 OpenMP Runtime Efficiency (TR-2)

The proposed OpenMP runtime may be efficiently implemented using special hardware features that accelerate frequent OpenMP operations. The time to execute an OpenMP barrier with NCORE OpenMP threads may be less than 200 clock cycles. The overhead for OpenMP Parallel FOR with NCORE OpenMP threads may be less than 500 cycles in the case of static scheduling.

3.6.10.2 Baseline Language Support for POSIX Threads (TR-1)

All the baseline languages may support programming node parallelism through POSIX threads Version 2.0 or then current standard (http://www.opengroup.org/onlinepubs/007908799/xsh/threads.html). The baseline language

compilers and/or interpreters may produce symbol tables and any other information required by the debugger to enable debugging of POSIX thread parallelized ASC applications.

3.6.10.3 Baseline Language Support for SE/TM (TR-2)

Offeror may propose baseline language support for efficiently and automatically (with the aid of language constructs or compiler directives) exploit any innovative node hardware support for parallel thread execution defined in Sections 2.4.6 and 2.4.7

3.6.11 Baseline Language and GNU Interoperability (TR-1)

The baseline language compilers may produce binaries that are compatible with the GNU compilers and loaders. In particular, the delivered baseline compiler OpenMP runtime libraries may be compatible with the GNU OpenMP libraries. That is, a single OpenMP based application can be built, run and debugged using modules generated from both Offeror supplied baseline language compilers and GNU compilers.

3.6.12 Runtime GNU Libc Backtrace (TR-2)

The baseline language compilers runtime support may provide the same backtrace functionality that the GNU libc does. Refer to:
http://www.gnu.org/software/libc/manual/html_node/Backtraces.html

3.6.13 Debugging Optimized Applications (TR-2)

The baseline languages will produce symbol tables and any other information required by the debugger to enable the debugging, in the presence of "-O -g" code optimization, of ASC applications. In particular, the baseline languages will provide a set of command line options that generate sufficient OpenMP optimized code and symbol table information so that the debugger can debug OpenMP threaded applications without loss of information about variables or source code context. Refer to Section 3.7.2.8.

3.6.14 Floating Point Exception Handling (TR-2)

The baseline languages will provide compiler flags that allow an application to detect Floating Point Exception (FPE) conditions occurring at runtime within a module compiled with those flags. This support will provide the compiled modules with an option to select any combinations of the floating point exceptions defined for IEEE-754 that include, but are not limited to, overflow, underflow, divided-by-zero, inexact, imprecise, quiet NaN and signaling NaN. With this support enabled, the application will receive a SIGFPE signal whenever a selected floating point exception condition occurs in any of the floating point hardware units (i.e. the main floating point unit and the SIMD unit).

The baseline languages will provide compiler flags that allow for imprecise exception flag setting so that exceptions may be raised by software checking exception flags on subroutine boundaries, block boundaries, loop boundaries and after each floating-point instruction is completed depending on how the compiler flag is set. Each decrease in software exception flag checking resolution will allow the resulting binary to run faster.

Further, the baseline languages will provide a compiler flag to inject 64b signaling NaNs into the heap and stack memory such that an application can easily detect the use of uninitialized memory.

3.7 Debugging and Tuning Tools

All debugging and tuning tools will be 64b executables and operate on 64b user applications by default.

3.7.1 Petascale Code Development Tools Infrastructure (TR-1)

Offeror will propose a hierarchal mechanism for code development tools to interact with petascale applications on the system in an efficient, secure, reliable and *scalable* manner. Hierarchal Code Development Tools Infrastructure (CDTI) components are distributed throughout the system. See Figure 3-2. Individual code development tool "front-end" components that interact with the user execute on the LN (although the tool X-Window may be displayed remotely on the users workstation). Code development tool communications mechanisms interface the tool "front-ends" running on the LN with the "back-end" manipulating the user application running on the CN through a single level fan-out hierarchy running on the ION. Since the IONs run a full BOS and the CNs run an LWK, actual manipulations of user job processes and threads running on the CN may be accomplished by function shipping these interfaces from the LWK to the BOS running on the ION.

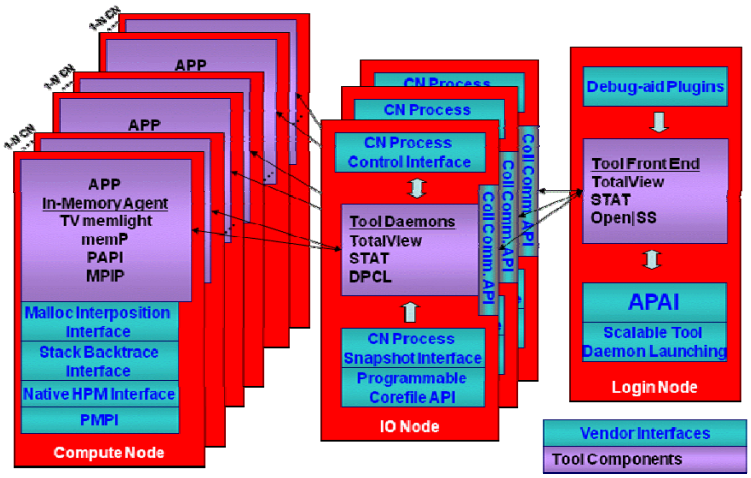


Figure 3-2: Code development tools hierarchal infrastructure components are distributed throughout the system.

3.7.1.1 CDTI Security (TR-1)

CDTI components and communications between components may be secure: users can only see and manipulate their applications and data and CDTI components may not run as the "root" user account. User identities may be maintained throughout the chain of CDTI components without giving users login capability directly on ION or SN BOS.

3.7.1.2 CDTI Reliability (TR-2)

Operations initiated on the "Front-End" components by users may successfully complete with no more than one failure per 10^6 user operations. Conditions set on CN such as Watchpoints, must correctly detect those conditions and successfully report back to the user

with no more than one failure in 10^6 events. Data communications between components may not be lost or corrupted more than one lost or corrupted message in 10^{12} messages.

3.7.1.3 CDTI Efficiency (TR-2)

The latency for basic operations including a memory/register read/write may not exceed 200 µs.

3.7.1.4 Remote Process Control Tools Interface (TR-1)

The basic functionality that the proposed Remote Process Control Tools Interface (RCPTI) may include, but are not limited to, an ability to control a CN process and threads (attaching, detaching, continuing, stopping, and single-stepping), an ability to read/write to/from the process-address and thread-address space and the register sets of a CN process and threads, and an ability to send a signal to a CN process.

Additionally, Offeror may provide necessary support via this interface if the CN hardware and OS feature more advanced process control capabilities. Additional capabilities may support setting a hardware watchpoint, fast trap, dynamic library debugging, and thread debugging, as described in Section 3.7.2.

3.7.1.5 Scalable CDT Daemon Launching Mechanism (TR-2)

When a job is launched under the control of a CDT (e.g., TotalView) then Offeror provided job launch mechanism should also launch the associated CDT daemon on the ION associated with the CN on which the job is launched. In addition, Offeror provided job launch mechanism should also launch these daemons for the situation in which the user wants to perform dynamic CDT interaction with the job (e.g., TotalView attach to a running job) after a job is started not under the control of a CDT. Both job and daemon launch time via this mechanism must be efficient and scalable. For example, daemon launch time may vary by no more than the log of the daemon count. Similarly job launch time under the control of a CDT may vary by no more than the log of the MPI task count. This daemon launching mechanism will provide each daemons associated with a job a single, randomly chosen socket port number via Offeror supplied API, which may be used for CDT daemon network bootstrapping.

3.7.1.6 Scalable CDT Daemon Bootstrapping Mechanism (TR-2)

Offeror may propose a scalable mechanism and API that allows LLNS provided CDT daemons associated with a specific user job to determine on what other ION their counterparts are running on and how to connect to them.

3.7.1.7 Scalable CDT Communications Infrastructure (TR-1)

Offeror may propose a scalable communication infrastructure that allows tools to control their respective daemons running on the IONs, to communicate with instrumentation inserted into the target application, and to aggregate and communicate gathered performance and debugging data back to the tool running on the LN (see Figure 3-2). This infrastructure may be hierarchical using a tree topology, if necessary to achieve scalability deploying additional layers of tool daemons between the daemons on the IONs and the tool on the LN. Further, this infrastructure may be capable of aggregating any stream of data using dynamically loaded and activated aggregation “filters”. An example and strongly preferred prototype and API for this functionality is MRNet (http://www.paradyn.org/mrnet/).

3.7.1.8 Programmable Core File Generation (TR-2)

Offeror may propose CDTI components in LWK and BOS on ION that will allow LLNS to develop and provide a programmable core file generation daemon (*pcfgd*) on the BOS on ION. These components will catch and forward signals generated by user process or thread(s) on any CN that would result in the application dumping core to the BOS running on the corresponding ION. The ION BOS component will notify the *pcfgd* associated with the job of the abnormal termination condition of the job. These components must allow the invoked *pcfgd* to perform operations on the job through an Offeror provided API on the ION BOS to make use of job debug information including stack traces, global MPI context such as global ranks. In addition, the Offeror provided API may provide a mechanism that allows the invoked *pcfgd* determine which other *pcfgd*'s are associated with the jobs on all other ION BOS associated with the job and to connect to them.

LLNS provided *pcfgd'*s associated with the job will then perform a set of operations on the job. For example, a simple operation would be to translate the raw address of each function frame of a stack trace into a symbolic name, enhancing readability of a core file. A more advanced technique would be to generate a globally merged call graph prefix tree by communicating trace data with other tool daemons.

3.7.1.9 Process Snapshot Interface for CN Processes (TR-2)

Offeror may propose CDTI components in LWK and BOS on ION that will allow LLNS to develop and provide a snapshot daemon (*snapd*) on the BOS on ION. Offeror provided components may provide an interface or a service that generates process snapshot information about the associated CN processes and passes this information to the LLNS provided *snapd*. The information includes, but is not limited to, a process's running state (i.e. running, stopped, or uninterruptible sleep), personality (i.e. pid), architectural state (i.e. PC value), various memory statistics and performance data including cumulative user time and system time.

3.7.1.10 Node Level Dynamic Instrumentation (TR-2)

Offeror supplied APIs in Sections 3.7.7, 3.7.7.1 and 3.7.7.3 will provide a means for dynamically inserting and removing, activating and deactivating, reading and resetting data for profiling, trace, and performance statistic instrumentation in the form of a port of Dyninst v5.2 or then current (see http://www.dyninst.org) to the target platform. Daemons running on the ION utilizing this API will be able to dynamically control, activate and deactivate the instrumentation on individual tasks or threads as well as groups of tasks and threads of a job running on associated CN through the remote process control interface described in Section 3.7.1.4.

3.7.1.11 Scalable Dynamic Instrumentation (TR-2)

Offeror will supply a mechanism to coordinate the node level instrumentation described in Section 3.7.1.10 across the whole machine in a scalable manner and to collect and dynamically aggregate results gathered through instrumentation. This mechanism may provide the Open Source DPCL API and functionality and may be built on top of Dyninst v5.2 or then current as described in Section 3.7.1.10. A reference implementation is available through the Open|SpeedShop project (http://www.openspeedshop.org).

3.7.2 Debugger for Petascale Applications (TR-1)

Offeror will provide an interactive debugger with an X11-based graphical user interface enabling single-point of control (multiple debugger invocations to control individual processes are not acceptable) that can debug petascale applications with multiple parallel programming paradigms (e.g., message passing, OpenMP thread and process parallelism). In particular the debugger will be able to handle debugging jobs with the Unified Nested Node Concurrency model (section 3.6.10) including jobs with one MPI task per core on the CN or the other extreme of one MPI task per CN and one mutable thread per core on that node for every CN. The petascale debugger will provide all functionality, including the ability to set breakpoints, to execute next and step commands and to examine the contents of language level variables, at the initial source level (before any preprocessing) for programs developed with inter-mixed baseline languages. Transitions between languages within a single program, must occur at the source level. A command line interface will be available for sequential and parallel programs. The capability of attaching/detaching the debugger to/from an executing (serial or parallel) program and modifying program state and continuing execution will be provided. If the code was not compiled for debugging, it is understood that access to source-level information will be limited. For MPI codes the debugger will display the status of message queues, such as number of pending messages and associated length, source, and sink at a breakpoint. For MPI codes the debugger will be able to breakpoint individual or groups or all tasks in a single GUI operation, step or continue an individual task, groups of tasks, or all tasks in a single GUI operation. For OpenMP threaded code the debugger will display the status of all threads, thread local and global variables, breakpoint individual or all OpenMP threads, step or continue individual or all OpenMP threads. Debugger functionality will include, but is not limited to: control of processes and threads (start/stop, breakpoints, and single-step into/over subprocedure invocations); examination of program state (stack tracebacks, contents of variables, array sections, aggregates, and blocks of memory, current states, registers, and source locations of processes); and modification of program state (changes to contents of variables, aggregates, and blocks of memory). The TotalView Technologies TotalView debugger (http://www.totalviewtech.com/productsTV.htm) is highly preferred.

3.7.2.1 Distributed Debugger Command and Control Architecture (TR-1)

Offeror's provided debugger will be based on the CDTI (3.7.1), perform data aggregation and reduction, and distribute command and control in a hierarchical manner (see Figure 3-2). In particular, interactions between processes and/or threads running on CN that don't require a direct user response, may be controlled (in parallel) by debugger daemons running on the ION without resorting to the debugger front-end running on the LN.

3.7.2.2 Scalable Dynamic Debugging of Running Jobs (TR-1)

Offeror provided debugger may be able to dynamically attach and debug running petascale jobs. This facility may also allow users the ability to detach from and later reattach to running petascale jobs.

3.7.2.3 Visual Representation of Data (TR-2)

Offeror will provide a parallel debugger capable of displaying multiple visual representations of values in a matrix or 2-D array section (e.g., bitmap showing elements exceeding a threshold value, colormap, surface map, contour map) with zoom and pan capability for the visual displays to facilitate scaling and display of large arrays. This functionality will be provided for all baseline languages. Offeror will provide a conditional data filtering

capability for large data sets integrated with data display functions, both textual and graphic. All visual capability will be invoked directly from the debugger.

3.7.2.4 User-Programmable Visual Data Display (TR-2)

Offeror may provide the parallel debugger with a user-programmable data display GUI feature. For setup, the user will register a callback function for each type to be specially displayed. When the user later asks the debugger to display a variable of one of these types, the callback function is passed a pointer to the variable and passes back to the debugger an array (for rows) of 3-tuples (data name pointer, type name of data display pointer, and data pointer). These become the columns displayed for each row, where of course, the Field name, Type name, and data Value displayed is expected to vary from row to row. A data value in one display could be used to request a second display, and so on. Because each displayed value is backed by memory, the debugger will allow the user to edit the displayed value according to its declared type. The debugger will then update the memory with the edited value. The debugger will automatically refresh each display whenever its focus thread stops.

3.7.2.5 Fast Conditional Breakpoints (TR-2)

Offeror proposed debugger may support fast conditional breakpoints in all the baseline languages. That is, an implementation for source code conditional breakpoints may add an overhead of less than 10 microseconds (1.0×10^{-5} seconds) per execution of the non-satisfied condition when the condition is a simple compare of up to two variables local to the process or thread.

3.7.2.6 Fast Data Watchpoints (TR-2)

Offeror proposed debugger may utilize the hardware data watchpoint facility (Section 2.4.11) when a user sets a data watchpoint in all of the baseline languages. The debugger may notify the user if it is unable to utilize the hardware data watchpoint facility for the watchpoint as requested by the user. Offeror proposed debugger may be architected and implemented to minimize the time required to evaluate a simple conditional watchpoint. This facility may be architected and implemented to scale to 8,192 MPI tasks. That is, LLNS desires that the conditional watchpoint facility may have an overhead of less than one microsecond (1×10^{-6} seconds) per execution of the non-satisfied condition when the condition is a simple compare of up to two variables local to the process or thread.

3.7.2.7 Memory Leak Debugging (TR-2)

Offeror will provide the capability of reporting memory access errors and pointing to the offending source line in the baseline languages. Memory access errors reported will include: accessing/freeing beyond allocated block; accessing/freeing unallocated blocks; memory leaks (accumulated memory chunks from malloc calls that can no longer be accessed or freed); and uninitialized memory read/write. This capability may utilize the LD_PRELOAD facility (Section 3.2.6).

3.7.2.8 Debugging Optimized Applications (TR-2)

The parallel debugger will, in the presence of "-O -g" code optimization, provide a fully supported mechanism for reporting information on program state (stack traceback, access to variables that have not been eliminated), breakpoints at basic block boundaries, single-stepping at the basic block level, and stepping over subroutines. In particular, the debugger

will be able to debug OpenMP threaded applications without loss of information about variables or source code context.

3.7.2.9 Debugger Expression Evaluator (TR-2)

The parallel debugger may have an evaluator capable of calculating the results of simple expressions (in "free floating" C and/or Fortran03) such as values of conditionals, indirect array references, etc. It is also desired that the evaluator handle the supported languages. This might be a language interpreter, but for the purposes of user code to be executed at breakpoints, or watchpoints, some form of compiled code is more desirable to make impact on execution smaller.

3.7.2.10 Parallel Debugger Barrier-Points (TR-2)

The parallel debugger will have an expanded breakpoint functionality for control of parallel processes by setting a "barrier-point." With a barrier point, the process will be held until all processes reach the same point, not responding to "start" commands until the barrier point is satisfied, or released.

3.7.2.11 Post-Mortem Debugging (TR-2)

The debugger will have a fully supported implementation of some mechanism for invoking the debugger for examining the final state of a program that failed ("postmortem debugging"). Facilities for modifying program state and/or continuing execution need not be available in this mode. If the code was not compiled for debugging, it is understood that access to source-level information will be limited.

3.7.2.12 Symbol Table (TR-2)

The time to initialize the debugger on an application with a 50 MB symbol table will be less than a minute longer than the time to initialize the debugger on the same number of processors, but with no symbol table.

3.7.2.13 Data Aggregation (TR-2)

The parallel debugger will have a capability of accumulating the local values of variables that are replicated across multiple threads/processes, and presenting a condensed summary within a single window. In addition, where distributed arrays are supported by the programming model, the debugger will have the capability of gathering the elements of a distributed 2-D array and presenting them in a single table/visualization.

3.7.2.14 Fast DLL Debugging Interface (TR-2)

Offeror may propose dynamically linked library (DLL) debugging support. Offeror can provide the functionality directly via the remote process control interface (section 3.1.3). Alternatively, Offeror can provide a Linux style interface where the interface is provided through well-known symbols within the CN's dynamic linker/loader (i.e. ld.so). In either case, DLL debugging mechanism may carefully be designed and reviewed because it has been the major source of performance bottlenecks.

3.7.2.15 Scalable Subset Debugging (TR-2)

Offeror provided debugger may implement a subset (from one to the number of MPI tasks in the job) debugging capability that allows a user to *scalably* debug a subset of processes/threads of a petascale job either at job launch under the control of the debugger or via dynamic debugging of running jobs (Section 3.7.2.2). When a subset is attached and being debugged, the performance of the debugger will scale as a function of the

process/thread count of the subset instead of the process/thread count of the job. For example, the performance of debugger operations in debugging an 1,024-MPI-task subset of a larger job will be equivalent to that of debugging a job with 1,024 total MPI tasks.

3.7.2.16 Scalability and Performance Enhancement (TR-2)

Offeror may propose a development plan to improve the usable performance of the debugger up to 32,768 MPI tasks. The plan may include, but is not limited to, enhancing parallelism among debugger daemons and using a tree-based debugger daemon hierarchy for data aggregation and reduction.

3.7.2.17 SE/TM Debugging (TR-2)

Offeror may propose mechanisms for aiding in debugging SE/TM programming model.

3.7.3 Stack Traceback (TR-2)

Offeror may propose runtime support for stack traceback error reporting. Critical information will be generated to STDERR upon interruption of a process or thread involving any trap for which the user program has not defined a handler. The information may include a source-level stack traceback (indicating the approximate location of the process or thread in terms of source routine and line number) and an indication of the interrupt type.

Default behavior when an application encounters an exception for which the user has not defined a handler is that the application dumps a core file. By linking in an Offeror provided system library this behavior may be modified to dump a stack traceback instead of a core file. The stack traceback indicates the stack contents and call chain as well as the type of interrupt which occurred.

Further, Offeror may provide APIs that allow a running process or thread to query its current stack traceback as well. The information will include a source-level stack traceback (indicating the approximate location of the process or thread in terms of source routine and line number) and an indication of the interrupt type, if any. The GNU backtrace runtime support as described in Section 3.6.12 and DynInst V5.2 (or then current) StackWalkerAPI described in Section 3.7.1.10 (see http://www.cs.wisc.edu/~legendre/stackwalker.ps) are highly preferred.

3.7.4 User Access to A Scalable Stack Trace Analysis Tool (TR-2)

Offeror may supply a scalable stack trace analysis and display X-11 GUI based tool that will allow normal users from the LN to securely and interactively obtain a merged stack traceback from a running petascale job or set of lightweight corefiles (Section 3.7.5).

3.7.5 Lightweight Corefile API (TR-2)

Offeror may provide the standard lightweight corefile API, defined by the Parallel Tools Consortium, to trigger generation of aggregate traceback data like that described in 3.7.3. The specific format for the lightweight corefile facility are defined by the Parallel Tools Consortium. See (http://web.engr.oregonstate.edu/~pancake/ptools/lcb/)

Offeror may provide an environment variable (or an associated command-line flag), with which users can specify that the provided runtime may generate lightweight corefiles instead of standard Linux/Unix corefiles. In addition, a provided library function, which generates the LCF may be available to the user. The core file may be written in the format specified by the Parallel Tools Consortium Lightweight Corefile Format. LLNS strongly prefers two modern extensions to the aging PTools LCF definition. First, STACK-ENTRY entries may be expanded to include

all available source information, such as the full path to the source file. Second, LCF files should include all thread traceback data and may be generated on a per-MPI Task basis.

3.7.6 Profiling Tools for Applications (TR-1)

Offeror may provide tools for profiling compute time distribution from all processes or threads in a parallel program, at the levels of subprocedures and coarse blocks (e.g., large loops). The tools may include a capability for restricting the amount of profiling data collected to certain portions of the source code (e.g., a specific subset of procedures), through the use of compiler directives, API or command-line switches. The tools may display the profiling data in a GUI showing the CPU time distribution on a source code level. The granularity of this display will be down to the source code block level. The statistics gathering and GUI functions may be usable when profiling an MPI/OpenMP threaded application running over an entire Sequoia system. This functionality may be made available both through the gprof toolset as well as through Open|SpeedShop (http://www,openspeedshop.org). Additionally, Offerer may provide a mechanism to export profile data from at least one of these tools to the PERIXML format (www.peri-scidac.org/wiki/images/5/5c/**PERIXML**-paper-2008.doc), which functions as a common interchange format between visualizers and profiling tools. TAU (http://www.cs.uoregon.edu/research/tau/home.php) and Open|SpeedShop are the preferred tools for visualizing profiling data.

3.7.7 Event Tracing Tools for Applications (TR-1)

Offeror may provide event tracing tools for petascale applications. Distributed mechanisms for generating event records from all process and threads in the parallel program will include timestamp and event type designators and will be formatted in a well-documented data format. This functionality may be provided for all baseline languages. The event tracing tool API will provide functions to activate and deactivate event monitoring during execution from within a process. By default, event tracing tools may not require dynamic activation to enable tracing. The OTF trace file format (http://www.tu-dresden.de/zih/otf) is highly preferred and the preferred tracing tools are the VampirTrace library for MPI and OpenMP events as well as performance counters (http://www.tu-dresden.de/zih/vampirtrace) and the Open|SpeedShop I/O tracer, both provided through the Open|SpeedShop toolset (http://www.openspeedshop.org).

3.7.7.1 Binary Event Trace Output Translation (TR-1)

If the provided trace file format is not in ASCII, Offeror may provide a supported and documented utility that converters binary event trace files to human readable ASCII text files. ASCII output format may allow for easy “grep-ing” or “gawk-ing” out of individual or groups of events. Offeror provided documentation may include an explanation of every event type and all their encoded fields.

3.7.7.2 Message-Passing Event Tracing (TR-1)

Offeror may provide a fully supported implementation of some mechanism for tracing message sends, receives, and synchronizations, including non-blocking messages, for the MPI libraries.

3.7.7.3 I/O Event Tracing (TR-1)

Offeror may provide a fully supported implementation of some mechanism for tracing I/O calls in user codes.

3.7.7.4 FPE Event Tracing (TR-2)

Offeror may provide a fully supported implementation of some mechanism for tracing all FPE events (as specified in Section 3.6.14) occurring during the execution of an application.

3.7.7.5 Lightweight Message-Passing Profiling (TR-1)

Offeror may provide a lightweight, scalable profiling library for MPI that captures only timing statistics about each MPI task. Instead of capturing entire traces, this tool captures limited data that includes min/max/cumulative time and a call count for each MPI callsite on a per task basis. The mpiP library (http://mpip.sourceforge.net/) is strongly preferred for this functionality.

3.7.8 Performance Statistics Tools for Applications (TR-1)

Offeror may provide performance statistics tools, whereby performance measures obtained for individual threads or processes are reported and summarized for LLNS application. Offeror may deliver the PAPI Version 4 API that gives user applications access to the 64b hardware performance monitors (Section 2.4.10). The PAPI based HPM API may include functions that allow user applications to initialize the 64b HPM, initiate and reset 64b HPM counters, read the 64b HPM counters and generate interrupts on HPM counter overflow and register interrupt handlers. This PAPI based HPM API may expose all 64b HPM functionality to user applications.

3.7.9 Scalable Visualization of Trace Data (TR-1)

Offeror may provide a scalable GUI tool or set of GUI tools that display trace data (as defined in 3.7.7) generated from MPI/OpenMP threaded applications. Both timeline and aggregate views are required. The statistics gathering and GUI functions for tracing may be usable when applied to an MPI/threaded application running over an entire Sequoia system. The preferred solution is to provide both Open|SpeedShop and VampirServer (a product in the Vampir tool suite) (http://www.vampir.eu).

3.7.10 Timer API (TR-2)

Offeror may provide an implementation of the Parallel Tools Consortium API for interval wall clock and for interval CPU timers local to a thread/process. The interval wall clock timer mean overhead may be less than 250 nanoseconds to invoke and may have a resolution of 1 processor clock period. The system and user timers mean overhead may be less than 1.5 microsecond to invoke and may have a global resolution of 10 milliseconds (i.e., this wall clock is a system wide clock and is accurate across the system to 10 milliseconds).

Comment [MS5]: Should we require user initiated timer callbacks or signals to enable sampling techniques? That would be very useful for tools, but might have a negative impact on the LWK performance?

3.7.11 Valgrind Infrastructure and Tools (TR-1)

Offeror may provide the open source Valgrind infrastructure and tools (http://valgrind.org) for the CN as well as for the LN and ION environments. For the CN, it is acceptable to provide a solution that requires the application to link with Valgrind prior to execution. This model has been successfully demonstrated on at least two LWK systems in current existence. It is the strong preference of LLNS that the provided Valgrind tool ports be made publicly available through the Valgrind.org maintained repository. At a minimum, LLNS may be provided the source code and the ability to build the Valgrind tools. At a minimum, the Valgrind release 3.3.0 (or then current) tools Memcheck and Helgrind may be provided.

3.8 Applications Building

3.8.1 LN Cross-Compilation Environment for CN and ION (TR-1)

Offeror may provide a complete cross-compilation environment that allows LLNS to compile and load applications on the LN for execution on the CN and daemons for the ION. This environment on the LN may allow LLNS to build automatically configured libraries and applications to detect the correct CN and ION ISA (Instruction Set Architecture), OS (Operating System), runtime libraries for the CN and ION rather than the LN using standard GNU AUTOCONF tools Version 2.61 (or then current). For correct operation, GNU Autoconf requires corresponding versions of GNU M4 and GNU Perl.

3.8.2 Linker and Library Building Utility (TR-1)

Offeror will provide an application linker with the capability to link object and library modules into a dynamic and static executable binary. By static execution binary we mean a binary that has all user object modules and libraries statically linked when the binary is created. By dynamic executable binary we mean that all the user object modules and static libraries are linked at binary creation, but that the user and system dynamic libraries are loaded at runtime on a demand basis. The linker and library building utility will produce executable binaries and static and dynamic load libraries that are 64b by default. In addition the linker will be capable of re-linking selected portions of an application (i.e., replace specific objects within the binary) rather rebuilding the executable binary from scratch. Offeror will include a facility to build and incrementally update static and dynamic libraries of object modules. The loader will be able to generate a full link listing of the load indicating at a minimum: which object file and original source file every function was taken from; which system functions were loaded from what library; complete memory map including function start points and the layout of all static and dynamic variables. If the microprocessor architecture possesses a memory reference model that includes segments, then the memory layout may be delineated by segment. The linker will provide the user with the capability of managing the memory layout by specifying the order in which libraries are loaded, the order variables and functions are loaded, etc. The compiler/linker combination will provide users the ability to control the placement of underscores (_), or other Offeror provided name mangling mechanisms, in front of or behind of externally visible variable and function names.

3.8.3 GNU Make Utility (TR-1)

Offeror may provide the GNU make utility with the ability to utilize parallelism in performing the tasks in a makefile.

3.8.4 Source Code Management (TR-2)

Offeror may provide a set of tools for the management of source code in a multiple programmer project environment (e.g., SCCS, USM, RCS, CVS,SVN).

3.8.5 Dynamic Processor Allocation (TR-2)

By setting various Linux shell environment variables and/or interactive or batch command line options, users may be able to run threaded applications compiled from any combination of the baseline languages exploiting automatic parallelization, compiler options, and/or MPI parallel application on varying numbers of processors and/or nodes without recompilation or relinking.

3.9 Application Programming Interfaces (TR-1)

All Offeror supplied APIs may support 64b executables and be fully tested in 64b mode. In particular, Marquee benchmarks may be 64b executables that utilize MPI with multiple styles of

SMP parallelism in a single 64b executable and run successfully with at least 2 GiB of user memory per user process over the entire machine.

3.9.1 Optimized Message-Passing Interface (MPI) Library (TR-1)

Offeror may provide a fully supported implementation of the MPI-2 standard, as defined by the most recent MPI-2 specification of the MPI forum. The system may be delivered with an optimized MPI-2 version compliant with current MPI-2 standard (without the MPI2 dynamic tasking) as defined by:

http://www.mpi-forum.org/docs/mpi-20-html/mpi2-report.html

The MPI library will be highly optimized in the sense that it will effectively and efficiently utilize all available hardware on the Sequoia system. In particular, the MPI library will operate transparently and directly on the Sequoia cluster interconnect network (i.e., not over TCP/IP or some other intermediate software layer). If the cluster interconnect network has multiple planes, then the MPI library will utilize the multiple planes to increase effective single task and node aggregate MPI off-node performance. The MPI library will be architected and implemented to minimize latency for small messages and maximize bandwidth for large messages under normal operating conditions. The negative performance impact of software layers implementing the MPI functionality between the user application and the hardware may be minimized. The delivered MPI library may be thread safe and allow applications to utilize MPI from individual threads. Two threaded application modes may be supported: thread multiple and thread funnel. The MPI library may be architected and implemented to utilize shared memory for communications between MPI tasks on a single node. The MPI global operations such as MPI_Barrier, MPI_Allreduce, MPI_Reduce, MPI_Broadcast may be architected and implemented to utilize hardware reduce and broadcast features of the system interconnect and take advantage of shared memory on a node to do Barriers, reductions and broadcasts first between task on a node and then between nodes as separate steps. It is insufficient to utilize shared memory solely for fast task to task communications in these operations. The MPI library will support up to one MPI task per core in the entire Sequoia system. The MPI buffers will be managed so that an application can set the amount of buffer space required for point-to-point and all-to-all communications. In particular, if an application guarantees that receives are posted before sends, then it will be possible to avoid MPI buffers completely. The Offeror may provide (electronic) written documentation that describes the performance features of the MPI implementation for each software release on the proposed Sequoia hardware. All environmental settings that impact MPI operation, buffering and performance and their impact to 64b user applications performance may be tested and their effectiveness and reliability documented.

3.9.1.1 PMPI Profiling Interface (TR-1)

Offeror may provide the PMPI profiling interface. Offeror may provide all appropriate Fortran and C conversion functions such as MPI_Request_f2c and MPI_Request_c2f. Offeror may deliver an instrumented version of the Sequoia MPI library. Instrumentation may collect mutually agreeable data during an application run on CNs and save that data to files on the Lustre file system. The format of the resulting data files may be documented and published. Data collected through instrumentation may be available for analysis by third-party tools running on LN.

3.9.1.2 Support for MPI Message Queue Debugging (TR-2)

Offeror provided MPI library and ADI interface may enable MPI message queue debugging to work with TotalView on LLNS applications. In addition, Offeror may provide a library that allows the TotalView debugger to access message queue information in MPI. The library may export a set of entry points as documented in the MPI message queue debug support API (Application Programming Interface) specification. Offeror may demonstrate its compatibility with dlopen call made from 64b debugger process running on the LN. The Offeror may also demonstrate its compatibility with the default MPI implementation on Sequoia. Such dynamic library will be loaded into the process-address space of a debugger process running LN and help the debugger process to accurately extract message queue information from debug requirements on CN.

3.9.2 Low Level Communication API (TR-1)

Documentation for Low level communications layer that MPI is built on may be provided. Interface may be published and non-proprietary.

3.9.3 User Level Thread Library (TR-1)

Offeror may provide a mechanism so that user applications can utilize all cores on the Sequoia CN with one MPI task per node. Some LLNS applications require a thread library that is IEEE POSIX 1003.1c-1995 standard Pthreads (www.llnl.gov/computing/tutorials/pthreads/) compliant. Other LLNS applications require OpenMP style parallelism with minimal overhead. The user level thread library may also support efficient compiler generated OpenMP parallelism. The overhead for self scheduling "do-loops" may be minimized by using unique hardware features of the Sequoia CN. User level scheduling of these threads is sufficient. Delivered thread libraries may allow the debugger to debug threaded applications.

3.9.4 Link Error Verification Facilities

Offeror may provide an API for user applications to call to periodically to verify that there have been no undetected transmission errors over the Sequoia interconnect. This interface may check the link 32b CRC calculated on each end of the link for every link on the Sequoia interconnect utilized by the application and return an error code if any pair of link 32b CRCs are different. Upon an error return, this interface may supply a list of links that have link 32b CRC errors. When called, this function may reset the CRC counters.

Offeror may provide an API that reads and returns the checksums calculated for all data injected into the Sequoia interconnect. These checksums can then be saved to disk by the application in order to verify correct network functioning in reproducible calculations after restarting from a previous checkpoint and rereading the new checksums at the appropriate point in the computation and comparing against the saved copies.

3.9.5 Graphical User Interface API (TR-1)

Offeror will provide the standard X11R7.3 (http://www.x.org/wiki/), Motif 2.1 (http://www.opengroup.org/motif/) and Qt 4.3 (http://en.wikipedia.org/wiki/Qt_(toolkit)), or current versions, applications, servers and API libraries. Secure viewing and usage of X-Windows to users remote workstations will be accomplished by LLNS provided SSH encrypted tunneling. All provided GUI API may be compatible with this approach.

3.9.6 Visualization API (TR-2)

Offeror will provide OpenGL 2.1, or current version, (http://www.opengl.org).

3.9.7 Math Libraries (TR-2)

Offeror may provide SMP and floating point (e.g., SIMD, Vectorization) optimized single-node mathematics libraries including: standard Offeror math libraries, Level 1 BLAS, Level 2 BLAS, Cholesky and LU factorization for dense double precision real matrices. LLNS may assist Offeror in optimizing selected routines out of FFTW as required by LLNS applications.

3.9.8 Hardware Debugging API (TR-2)

Offeror may propose a fully supported, published and documented API that allows users to access the hardware debugging support proposed under Section 2.4.11.

3.10 Compliance with DOE Security Mandates (TR-1)

DOE Security Orders have changed over time. From time to time, these mandates cause LLNS and/or it's Subcontractors, to fix bugs or implement security features in vendor operating systems and utilities.

Offeror may handle bug fixes as follows: A "bug" is interpreted to mean that a product does not perform as documented. If a "bug" is discovered, there are standard reporting procedures which must be followed for tracking purposes. Additionally, the report from Offeror would be escalated by the Offeror or the ASC Program office to achieve priority resolution.

For implementation of security features in Offeror's operating system and utilities, Offeror requires written notification of the changes to DOE Security Orders or their interpretation that would force changes in system functionality. If the request for change would result in a modification consistent with standard commercial offerings and product plans, the Offeror may perform the change. If the change is outside the range of standard offerings, the Offeror may make the operating system source code available to the LLNS (at no additional cost, assuming the LLNS holds the proper USL and other prerequisite licenses) under the terms and conditions of the Offeror's standard source code offering.

3.11 On-Line Document (TR-2)

Offeror may supply hardcopy and on-line documentation by http based mechanism for all major hardware and software subsystems viewable from standard Mozilla Firefox or Windows Explorer or Apple Safari browsers.

3.12 Early Access to Sequoia Software Technology (TR-1)

The Offeror may propose mechanisms to provide LLNS early access to Sequoia software technology and to test software releases and patches before installation on Sequoia that includes other steps before installing the software on Dawn and the dual boot environment on Sequoia.

End of Section 3.0

4.0 Dawn High-Level Hardware Requirements

The ASC Program requires early access to stable code development platforms for the rapid development of Stockpile Stewardship applications for the Sequoia system. As such, the Dawn system size, measured in number of cores, should be more than half way between Purple at LLNL (8,192 core hero runs) and Sequoia O(2.5M) cores+threads. This will allow ASC code developers to achieve a significant advance in ASC IDC and Science applications scalability in the Dawn timeframe and lessen the jump from Purple and BGL at LLNL to Sequoia. Thus, it is imperative that the Dawn code development environment present the same hardware and software model to code developers that will be available on Sequoia. In particular, hardware issues such as memory hierarchy, CN interconnect topology and message passing support should present the same issues to application performance that will be experienced in the Sequoia. As an example, if high cache utilization is required for application performance on Sequoia, then Dawn must present the same performance challenges and opportunities for optimization.

In addition, the ASC Program requires additional production petascale capacity. Thus, the Dawn system should deliver stable, cost-effective cycles to a wide variety of the ASC Program workload.

It is envisioned that Dawn will come from Offeror's current (or very near term) product offering and provide, to the maximum extent possible, a code development environment congruent with that of Sequoia.

In specifying the Dawn system LLNS was motivated by four factors:

- provide a delivery vehicle that focuses the efforts of the ASC Program / Offeror partnership before the delivery of Sequoia;
- provide an early hardware and software code development environment that closely resembles the Sequoia system;
- provide additional capacity beyond what is currently available at LLNL;
- provide a vehicle for early testing of Sequoia hardware and/or software.

This Dawn system will be delivered to LLNL.

The specific hardware requirements of the Dawn system are delineated in Section 4.0, "Dawn Hardware Requirements". The specific software requirements for the Dawn system are delineated in Section 5.0, "Dawn Software Requirements". There is only one mandatory requirement for Dawn, Section 4.1 "Dawn 0.5 petaFLOP/s System." Technical options for Dawn are identified in Section 4.3.

In addition, the integrated system features of Section 6.0 apply to Dawn as well as Sequoia. Rather than replicate all of the Sequoia hardware requirements in this section, LLNS includes all of the Sequoia hardware requirements and their associated priorities (TR-1, TR-2 and TR-3) and then notes differences for Dawn. However, during the course of executing the Sequoia subcontract(s), LLNS anticipates that more functionality will be provided over time. The RFP Proposal Evaluation and Proposal Preparation Instructions indicate where in the RFP response Offerors may delineate in detail the proposed schedule of hardware and software deliverables and. in particular, the Dawn and technology refresh deliverables, if applicable.

In addition to the hardware and software requirements, the Offeror will deliver any additional features consistent with the objectives of this project and Offeror's Full-Term Plan of Record, which the Offeror believes will be of benefit to LLNS.

All of the Sequoia Hardware requirements (section 2.0) apply to the Dawn system(s), except that the following requirements supercede Section 2.0.

4.1 Dawn 0.5 petaFLOP/s System (MR)

This requirement supercedes Section 2.1. The Offeror shall propose a fully configured, complete and functional Dawn System with at least 0.5 petaFLOP/s peak performance.

4.2 (4.3) Dawn Component Scaling (TR-1)

This requirement supercedes Section 2.3. In order to provide the Offeror with maximum flexibility to meet the goals of the ASC Program, the exact configuration of the Dawn scalable system is not specified. Rather, the Dawn configuration is given in terms of lower bounds on component attributes relative to the peak performance of the proposed configuration. The Dawn scalable system configuration may meet or exceed the following parameters:

Memory Size (Byte:FLOP/s) ≥ 0.3

Memory Bandwidth (Byte/s/FLOP/s) ≥ 1.0

Intra-Cluster Network Aggregate Link Bandwidth (Bytes/s/FLOP/s) ≥ 0.1

Intra-Cluster Network Bi-Section Bandwidth (Bytes/s/FLOP/s) ≥ 0.001

System Sustained SAN Bandwidth (GB/s:petaFLOP/s) ≥ 384

High Speed External Network Interfaces (GB/s:petaFLOP/s) ≥ 96

4.3 (4.12) Dawn Hardware Options

This section superceeds Section 2.12.

Offeror may propose each of the following TOs, as separately priced options. Offeror may technically describe, in the following sections of its technical proposal(s), how the options will be effected, if exercised by LLNS.

4.3.1 (4.12.1) Dawn Enhanced IO Subsystem (TO-1)

Offeror may propose an enhanced IO subsystem for Dawn that provides for double the baseline IO performance for jobs spanning 50% of the machine and 25% of the compute nodes. That is, the enhanced IO subsystem proposed may deliver at least 100% of the full system IO delivered bandwidth to jobs using 100% of the CN and may achieve 100% of the full system IO delivered bandwidth for jobs using 50% of the CN and may achieve 50% of the full system IO delivered bandwidth for jobs using 25% of the CN.

4.3.2 (4.12.2) Dawn Double Memory (TO-1)

Offeror may propose Dawn CN with double the memory of the baseline Dawn system. In this option, the ION/LN memory may be remain consistent with Section 4.3. That is, the memory size component scaling B:F ratio for this CN (only) memory option may meet or exceed:

Memory Size (Byte:FLOP/s) ≥ 0.6

4.3.3 (4.12.2) Dawn Double ION/LN Memory (TO-2)

Offeror may propose Dawn ION/LN with double the memory of the baseline Dawn system. That is, the memory size component scaling B:F ratio for this ION/LN (only) memory option may meet or exceed:

Memory Size (Byte:FLOP/s) ≥ 0.6

End of Section 4.0

5.0 Dawn High Level Software Requirements

All of the Sequoia Software requirements (Section 3.0) apply to the Dawn system(s). The following requirements supercede the corresponding requirements in Section 3.0.

5.6.10.1 Baseline Language Support for OpenMP Parallelism (TR-1)

All the baseline languages (i.e., Fortran03, C, C++ and Python) compilers or interpreters may support node parallelism through OpenMP Version 2.5 directives or language constructs (http://www.openmp.org/drupal/mp-documents/spec25.pdf). As an optimization feature, all the baseline language compilers may perform automatic parallelization. The baseline language compilers may produce symbol tables and any other information required by the debugger to enable debugging of OpenMP parallelized ASC applications.

End of Section 5.0

6.0 Integrated System Features (TR-1)

The following requirements deal with the functional aspects of the integrated Dawn and Sequoia systems. Both Dawn and Sequoia are intended for classified production usage at LLNL in the Secure Computing Facility (SCF) by the ASC and Stockpile Stewardship Tri-Laboratory Communities. LLNS therefore requires that the Dawn and Sequoia systems have highly effective, scalable RAS features and prompt hardware and software maintenance.

For hardware maintenance, the strategy is that LLNS personnel will provide on-site, on-call 24x7 hardware failure response. LLNS envisions that these hardware technicians and system administrators will be trained by the selected Offeror to perform on-site service on the delivered hardware. For easily diagnosable node problems, LLNS personnel will perform repair actions in-situ by replacing Field Replaceable Units (FRUs). For harder to diagnose problems, LLNS personnel will swap out the failing node(s) with on-site hot spare node(s) and perform diagnosis and repair actions in the separate Hot-Spare Cluster (HSC). Failing FRUs or nodes (except for writable nonvolatile media) will be returned to the Offeror for replacement. Hard Disks FRUs and writeable nonvolatile media (e.g., EEPROM) from other FRUs will be destroyed by LLNS according to DOE/NNSA computer security orders. Thus, LLNS requires an on-site parts cache of all FRUs and a small system of fully functional hot-spare nodes of each node type. The Offeror will work with LLNS to diagnose hardware problems (either remotely or on-site, as appropriate). On occasions, when systematic problems with the cluster are found, the selected Offeror's personnel will augment LLNS personnel in diagnosing the problem and performing repair actions.

In order for the Dawn and Sequoia systems to fulfill the mission of providing "capability" computing resources for LLNS, they must be highly stable and reliable from both a hardware and software perspective. The number of failing components per unit time (weekly) should be kept to a minimum. System components should be fully tested and burned in before delivery (initially and as FRU or hot-spare node replacement). In addition, in order to minimize the impact of failing parts, LLNS community must have the ability to quickly diagnose problems and perform repair actions. A comprehensive set of diagnostics that are actually capable of exposing and diagnosing problems are required. It has been LLNS' experience that this is a difficult but achievable goal, and the selected Offeror will need to specifically apply sufficient resources to accomplish it.

For software, the strategy is similar to the hardware strategy in that LLNS personnel will perform the Level 1 (initial call, routine questions and answers, routine software documentation) and Level 2 (routine bug fix, detailed questions and answers, detailed software documentation) software support functions. Specifically, LLNS personnel will diagnose software bugs to determine the failing component. The problem will be handed off to the appropriate LLNS organization for resolution. For LLNS supplied system tools, LLNS personnel will fix the bugs. For Offeror-supplied system tools, the selected Offeror will need to supply problem resolution. For the Linux kernel and associated utilities, LLNS intends to separately subcontract with Red Hat for Enterprise level support. For file system related SW problems, the LLNS intends to separately subcontract with Sun Microsystems for Lustre support. For compilers, debuggers and application performance analysis tools, LLNS intends to separately subcontract with the appropriate vendors for support.

This software support strategy depends on all software components being Open Source and source code available to LLNS for viewing, modification, compilation and execution on the provided systems. It is absolutely necessary that the selected Offeror provide LLNS any unique development environment

components required to reproduce from source code any portion of the Dawn or Sequoia software environment, except for compilers and runtime support. Any bug fixes developed by LLNS personnel will be provided back to the selected Offeror. If Offeror proposed system components are not Open Source, then full source code software licenses that allow LLNS to perform these support functions is required.

6.1 System RAS (TR-1)

Offeror proposed systems may include an integrated Reliability, Availability and Serviceability (RAS) maintenance strategy integrated into the overall architecture, design and implementation that results in a highly usable and robust production system for ASC programmatic usage. To optimize the proposed systems for maximum uptime, Offeror's strategy may include redundancy of individual components that fail most frequently, and the ability to repartition the system to isolate known faulty sectors. LLNS will have regular scheduled maintenance to replace known failed components. Most of these components may be in N+1 redundant systems as discussed below. Thus Dawn and Sequoia should have the feature that its reliability continues to improve over time, as the weaker components are replaced.

6.1.1 Hardware Failure Rate Impact on Applications (TR-1)

The proposed systems may have Mean Time Between Application Failure (MTBAF) due to a hardware failure or hardware transient error of greater than 168.0 hours (7.0 days). A hardware induced application error is any hardware failure or transient error that causes an application running on the system to abnormally terminate. Hardware failures or transient errors that do not cause an application to abnormally terminate, such as failure of an N+1 redundant power supply, do not count against this MTBAF statistic. Offeror will provide a system MTBAF estimate with the proposal response. Offeror may propose methods and means to mitigate the impact of hardware failures or transient errors on applications such as checkpoint/restart if these are reliable and transparent to the application and its users.

6.1.2 Mean Time Between Failure Calculation (TR-1)

Offeror will provide the Mean Time Between Failure (MTBF) calculation for each FRU and node type. Offeror will use these statistics to calculate the MTBF for the proposed Dawn and Sequoia systems. This calculation will be performed using a recognized standard. Examples of such standards are Military Standard (Mil Std) 756, Reliability Modeling and Prediction, which can be found in Military Handbook 217F, and the Sum of Parts Method outlined in Bellcore Technical Reference Manual 332. In the absence of relevant technical information in an Offeror's proposal, LLNS will be forced to make pessimistic reliability, availability, and serviceability assumptions in evaluating the Offeror's proposal.

6.1.3 Failure Protection Methods (TR-1)

Because of the large number of individual components constituting a petascale system, great care may be taken to limit the effects of failures. The system ASIC such as processors, network interface chips (NIC), memory ASIC may incorporates error detection and correction circuitry on the components with high failure rates due to soft and hard errors. These components may include the node memory, the processor or NIC memory hierarchy (L3, L2 and L1 cache, SRAM arrays for inter-core synchronization and communication). The internal register arrays and critical dataflow busses man have at a minimum parity for error detection. Power supplies on the proposed system may have power distribution that provides active-active or N+1 redundancy and are individually high reliability. All air moving devices may be N+1 redundant and may be

operated at lower speed when all fans are active in order to improve reliability. In the event of a failure, the system may be reconfigured or repartitioned to remove the fail. After system reconfiguration or repartitioning, the application that terminated due to the failure may be restarted from the last checkpoint and continue computations.

6.1.4 Data Integrity Checks (TR-1)

Another important source of errors is the links connecting the nodes. These links may incorporate an error detection check (CRC) on packets that may cover multiple bit errors. After a packet error is detected, the link controller may retry the failed packet. The system interconnect may have 24 bits in the CRC for user data.

6.1.5 Interconnect Reliability (TR-1)

The system interconnect may reliably deliver a single copy of every packet injected into it, or it may indicate an unrecoverable error condition. Therefore, send-side software need not retain copies of injected messages, and receive-side software need not maintain sequence numbers. This level of hardware reliability is required because software techniques such as sliding window protocols do not scale well to petascale systems. Interconnect reliability may be provided by a combination of end-to-end, and link-level, error detection. In most cases, the link-level error detection features may discover, and often recover from an error. The end-to-end error detection may be used primarily to discover errors caused by the routers themselves and missed by the link-level protocol.

6.1.6 Link-Level Errors (TR-1)

The link-level error detection scheme may use CRC bits appended to every packet. Because most modern interconnects use cut-through routing techniques, it is highly likely that a packet detected as corrupt has already been forwarded through multiple downstream routers, so it cannot simply be dropped and re-transmitted. Instead, the router detecting the error may modify the packet to indicate the error condition, causing the packet to be dropped by whichever router eventually receives it. In the case where a corrupt packet is entirely stored in a cut-through FIFO, it is possible to drop it immediately. In addition to marking the corrupt packet, the router detecting the error may also cause a link-level re-transmission. This recovery mechanism may insure that only one “good” copy of every packet arrives at the intended receiver. Packets that are marked as corrupt may be discarded automatically by a router, and not inserted into a reception FIFO. Another possible source of link-level errors is “lost” bits, which would lead to a misrouted, malformed packet. Worse yet, this could lead to a lack of synchronization between adjacent routers. Although it is possible to recover from this situation, the hardware investment would be significant, and the occurrence is expected to be quite rare. Therefore, the network and Offeror proposed software may simply report this condition to the RAS database and allow system software to recover. In addition, every interconnect link may have an additional 32b CRC that is calculated on each end of the link. These 32b CRC can be used to verify that the data was correctly transmitted across the links and check for packet 24b CRC error escapes. After a job fails, every link in the job can be checked. The interconnect logic may checksum (not CRC) all data that is injected in the interconnect. This may be read out by libraries supplied by the Offeror from user applications on a regular basis, say on every time step of the simulation, and saved away. Then when the application checkpoints with Offeror supplied checkpoint library, these checksums may also be written out. This may be used to roll-back an application to a previous checkpoint and verify that recomputed time steps generate the same checksums. If the

checksums don't match, then the first processor that has a different checksum indicates where the error is located.

6.1.7 Capability Application Reliability (TR-1)

A user application job spanning 80% of the nodes in the system may complete a run with correct results that utilizes 200 hours (8.33 days) of system plus user core time per core in at most 240 wall clock hours (10.0 days) without human intervention. A user application job spanning 30% of the nodes in the system may complete a run with correct results that utilizes 200 hours of system plus user core time per core in at most 220 wall clock hours without human intervention. These runs may be accomplished utilizing application checkpointing on a frequency recommended by Offeror and multiple dependent SLURM/Moab jobs for restarting.

6.1.8 Power Cycling (TR-3)

The system will be able to tolerate power cycling at least once per week over its life cycle.

6.1.9 Hot Swap Capability (TR-2)

Hot swapping of failed Field Replaceable Units (FRUs) may be possible without power cycling the cabinet in which the FRU is located The service strategy may ensure that a granular FRU structure is implemented. A granular FRU structure means that the maximum number of components (such as processors, memory, disks and power supplies) contained in or on one FRU may be less than 0.1% of the components of that type in the system for system components with at least 1,000 replications.

6.1.10 Production Level System Stability (TR-2)

The system (both hardware and software) may execute 100 hour capability jobs (jobs exercising at least 90% of the computational capability of the system) to successful completion 95% of the time. If application termination due to system errors can be masked by automatic system initiated parallel checkpoint/restart, then such failures may not count against successful application completion. That is, if the system can automatically take periodic application checkpoints and upon application failure due to system errors automatically restart the application without human intervention, then these interruptions to application progress do not constitute failure of an application to successfully complete.

6.1.11 System Down Time (TR-2)

Over any four week period, the system will have an effectiveness level of at least 95%. The effectiveness level is computed as the weighted average of period effectiveness levels. The weights are the period wall clock divided by the total period of measurement (four weeks). A new period of effectiveness starts whenever the operational configuration changes (e.g., a component fails or a component is returned to service). Period effectiveness level is computed as LLNS operational use time multiplied by max[0, (N-2D)/N] divided by the period wall clock time. Where N is the number of compute nodes in the system and "D" is the number compute nodes unable to run user jobs. Scheduled Preventive Maintenance (PM) is not included in LLNS operational use time.

Example: A system with 50,000 compute nodes would have an effectiveness level of 96.43% with one day of full system downtime or an effectiveness level of 98.83% if 8,192 CN were down for one day or 95.32% if the 8,192 CN were down for 4 days or an effectiveness level of 97.95% if 512 CN were down for 28 days.

6.1.12 Scalable RAS Infrastructure (TR-1)

The Offeror will provide a scalable RAS infrastructure that monitors and logs the system health from a centralized set of SN. All system maintenance functions will be executable from the SN by the system administration staff.

6.1.12.1 Highly Reliable Management Network (TR-1)

The system management Ethernet will be a highly reliable network that does not drop a single managed element from the network more than once a year. This is both a hardware and a software (Linux Ethernet device driver) requirement. In addition, the management network will be implemented with connectors on the node mating to the management Ethernet cabling and connectors (Section 2.10) so that manually tugging or touching the cable at a node or switch does not drop the Ethernet link. The management Ethernet switches (Section 2.10) will be configured such that they behave as standard multi-port bridges.

6.1.12.2 Sequoia System Monitoring

Offeror may propose a RAS database and infrastructure for system monitoring and control called RASD. The RASD may be available from all SN. All communication for the RASD may be over IP on the system management Ethernet. All control and monitoring actions may be initiated from the RAS facility. All control/monitoring may be event driven or gathered by periodic polling by the RAS facility. The RAS facility is organized as a set of management processes running on the SN. The RAS facility may be comprised of the following management components: 1) Open Source relational database (RASD) to maintain all system state; 2) System initialization (Init/Discovery) to identify hardware as it is powered on; 3) Job Control and Launch (JCL) to process requests to allocate hardware, run and monitor jobs; and 4) RAS and monitoring (Monitor) support for both hardware and software events.

The RAS facility may be able to control power (power up and down and status power) where power can be controlled by software; diagnose, detect and report system hardware failures and potential failures.

6.1.12.2.1 System Hardware Status Database

The RASD may include a persistent database available from any SN for controlling the system. The RASD may contain at least the following information:

machine topology (compute nodes and I/O nodes);

IP address of each hardware component (e.g., node, chassis, PDU, rack) management interface;

state (assumed and/or measured) of each device;

RASD may have the capability to both query and update the database from any SN. Causing RAS facility to perform actions on system hardware components may be accomplished through manipulating the RASD (e.g.: resetting a node may be accomplished by setting the appropriate field in the database). Only the root user may have the ability to modify the RASD and perform system hardware manipulation actions. Access to the RASD may be controlled through database privilege mechanism.

6.1.12.3 Scalable System Monitoring (TR-1)

All bit errors in the system (e.g., memory errors, data transmission errors, local disk read/write errors, SAN interface data corruption), over temperature conditions, voltage irregularities, fan speed fluctuations, and disk speed variations may be logged in the RASD. All bit errors may be logged for recoverable and non-recoverable errors. The RAS facility may automatically monitor this database constantly, determine irregularities in subsystem function and promptly notify the system administrators. RAS subsystem configuration will include calling out what items are monitored, at what frequency monitoring is done for each item, what constitutes a problem with at least three severity levels (low, medium and high) and notification mechanisms for each item at each severity level.

6.1.12.4 Highly Reliable RAS Facility (TR-1)

The provided scalable RAS facility may be highly reliable in the sense that there are no single points of failures in the RAS facility and any single component failure may not impact the ability to continue to process the workload on compute, login, I/O and visualization nodes.

6.1.12.5 Failure Isolation Mode (TR-2)

FRU failures and FRU with intermittent failures will be quickly and reliably identified by Offeror supplied diagnostic utilities (not divine intervention), isolated, and routed around without system shutdown. These diagnostic utilities will utilize FRU error detection and fault isolation hardware. The diagnostic utilities will utilize built in error detection and fault isolation circuitry to accurately detect and report failures in all core components and in particular the floating-point units, memory and interconnect. These diagnostics will stress test FRUs and reliably cause failures in marginally functional or intermittently failing parts and accurately detect these failures. Accuracy and reliably here means less than 1 false positive (miss identification of a fully functional FRU as a failed FRU) or false negative (miss identification of a failed FRU as a fully functional FRU) out of 1,000,000 individual FRU runs of the diagnostic stress test. Quickly here means the diagnostics may be able to achieve these results with less than two hour runtimes (i.e., the diagnostics get in, get it right and get out quickly). The operators will be able to reconfigure the system to allow for continued operation without use of the failed node or FRU. The capability will be provided to perform this function from a remote network workstation.

6.1.12.6 Scalable System Diagnostics (TR-2)

There will be a scalable diagnostic code suite that checks processor, cache and RAM memory, network functionality, and I/O interfaces for the full system in less than 30 minutes. The supplied diagnostic utilities will quickly, reliably and accurately determine the processor or Node or FRU failures.

6.1.13 System Graceful Degradation Failure Mode (TR-2)

The failure of a single component such as a single core, processor, a single memory component, a single node, or a single communications channel may not cause the full system to become unavailable. It is acceptable for the application executing on a failed processor or node to fail but not for applications executing on other parts of the system to fail.

6.1.14 Node Processor Failure Tolerance (TR-2)

Any multi-socket ION, LN and SN may be able to run with a processor and/or socket disabled, and to do so with minimal performance degradation. That is, ION, LN and SN nodes will be able to tolerate processor failures through graceful degradation of performance.

6.1.15 Node Memory Failure Tolerance (TR-2)

The Offeror may propose nodes that are able to run with one or more memory components disabled, and to do so with minimal performance degradation. That is, the nodes may be able to tolerate failures through graceful degradation of performance where the degradation is proportional to the number of FRUs actually failing.

6.2 Hardware Maintenance (TR-1)

Offeror may supply hardware maintenance for both the Dawn and Sequoia systems for a five-year period starting with system acceptance. LLNS personnel will attempt on-site first-level hardware fault diagnosis and repair actions. Offeror will provide second-level hardware fault diagnosis and fault determination during normal business hours. That is, if LLNS personnel cannot repair failing components from the on-site parts cache, then Offeror personnel will be required to make on-site repairs. Offeror supplied hardware maintenance response time will be before the end of the next business day from incident report until Offeror personnel perform diagnosis and/or repair work. The proposed system will be installed in a limited access area vault type rooms (VTR) at LLNL and maintenance personnel must obtain DOE P clearances for repair actions at LLNL and be escorted during repair actions. USA Citizenship for maintenance personnel is highly preferred because it takes at least 30 days to obtain VTR access for foreign nationals.

During the period from the start of system installation through acceptance, Offeror support for hardware will be 12 hour a day, seven days a week (0800-2000 Pacific Time Zone), with one hour response time.

6.2.1 On-site Parts Cache (TR-1)

A scalable parts cache (of FRUs and hot spare nodes of each type proposed) at LLNL is desired that will be sufficient to sustain necessary repair actions on all proposed hardware and keep them in fully operational status for at least one month without parts cache refresh. That is, the parts cache, based on Offeror's MTBF estimates for each FRU and each rack, will be sufficient to perform all required repair actions for one month without the need for parts replacement and should scale up as system racks are delivered. Offeror will resupply/refresh the parts cache as it is depleted for the five year hardware maintenance period. Offeror will propose sufficient quantities of FRUs and hot-spare nodes for the parts cache. The parts cache will be enlarged, at the selected Offeror's expense, should the on-site parts cache prove, in actual experience, to be insufficient to sustain the actually observed FRU or node failure rates. However, at a minimum, the on-site parts cache will include the following fully configured CN, ION, and LN FRUs. LLNS will store and inventory the on-site parts cache components. Parts in the parts cache are LLNS property. Failed parts become Offeror's property when RMAed back to Offeror.

Offeror may propose a Return Merchandise Authorization (RMA) reporting and tracking database that allows LLNS to report failing or failed parts and keeps track of failed FRU by serial number. Offeror will ship replacement FRU upon authorization of RMA request and not wait until LLNS returns failed FRU under the RMA. LLNS may have access to this RMA database to survey the data. RMAs may be issued to LLNS by interacting with the database.

Offeror may ship replacement FRUs immediately upon RMA, while LLNS is shipping filed FRUs back to Offeror. Offeror may not wait for LLNS shipment of failed FRU to arrive at Offeror's facility and/or waiting for supplier replacement.

6.2.2 Secure FRU Components (TR-1)

The selected Offeror may identify any FRU in the proposed system that persistently holds data in non-volatile memory or storage prior to system and on-site spare parts cache delivery. Selected Offeror may deliver prior to system and on-site parts cache delivery a Statement of Volatility for every and all unique FRU that contain only volatile memory or storage and thus cannot hold user data after being powered off. FRU with non-volatile memory or storage that potentially contains user data will not be returned to the Offeror. Instead, FRU with non-volatile memory or storage that could potentially contained user data will be certified by LLNS to Selected Offeror as destroyed as part of Offeror's RMA replacement procedure under the hardware maintenance plan.

6.3 Software Support (TR-1)

Offeror will supply software maintenance for each Offeror supplied software component, specifically including the supplied LWK, starting with the Dawn or Sequoia system acceptance and ending five years after the Dawn or Sequoia system acceptance. Offeror provided software maintenance may include an electronic trouble reporting and tracking mechanism and periodic software updates. In addition, the Offeror will provide software fixes to reported bugs. The electronic trouble reporting and tracking mechanism may allow the LLNS to report bugs and status bug reports 24 hours a day, seven days a week. The LLNS will prioritize software defects so that Offeror can apply the software maintenance resources to the most important problems.

During the period from the start of system installation through acceptance, Offeror support for supplied software will be 12 hour a day, seven days a week (0800-2000 Pacific Time Zone), with one hour response time.

6.4 On-site Analyst Support (TR-1)

Offeror may supply two on-site analysts to LLNS. One on-site systems programmer will be highly skilled in Linux systems programming and may support LLNS personnel in providing solutions to the current top ten issues. The systems programmer may have proficiency in C programming and familiarity with interpreted languages (e.g., Perl, Python, Expect, and Bash Shell) and have experience maintaining open source software. This includes merging with upstream releases, submission of local patches upstream, code repository maintenance, build and test process implementation. In addition the systems programmer will be highly skilled in operational activities of the proposed Dawn and Sequoia systems and may support LLNS personnel in day-to-day operations and software systems upgrades. The systems programmer will use LLNS and Offeror trouble ticket mechanisms and hardware and software problem tracking data bases to maintain an accurate systems availability, effectiveness level, calculate actual MTBAF and support response time statistics. The systems programmer may interact with the Offeror development and support organizations as an advocate for LLNS top10 issues.

One on-site applications analyst will be highly skilled in applications development and porting to the Dawn and Sequoia systems. This applications analyst will provide expertise to the Tri-Laboratory ASC code development teams in the areas of software development tools, parallel applications libraries and applications performance. The proposed system will be installed in a classified area at

LLNL and so analyst personnel may obtain DOE Q clearances. LLNS may request additional on-site analysts, which will be priced separately.

End of Section 6.0

7.0 Facilities Requirements

An existing facility, portions of the West and East computer floors in the LLNL B453, will be used for siting the Dawn (west end of East floor) and Sequoia (east end of West floor). See Figure 7-1. Today, the B453 building has approximately 2x125' x 195' = 47,500ft^2 and 15 MW (7.5 MW for the West floor and 7.5 MW for the East floor) of power for computing systems and peripherals and associated cooling available for this purpose. Prior to the deployment of Sequoia the B453 building will be upgraded to 30 MW total (15.0 MW for the West floor and 15.0 MW for the East Floor). The Purple system will be retired after Dawn is deployed, but before Sequoia is deployed. This will leave approximately 15.0 MW available for Sequoia from the West computer floor. There is approximately 5.0 MW available for Dawn. Facilities modifications to provide the necessary power and cooling for Dawn and Sequoia will need to be accomplished prior to rack delivery. It is therefore essential that Offeror make available to LLNS detailed and accurate (not grossly conservative overestimates) site requirements for the Dawn system at proposal submission time. Less accurate power and cooling estimates for Sequoia at proposal submission, but not grossly conservative overestimates) will be of substantial value as well. LLNS will be responsible for supplying the external elements of the power, cooling, and cable management systems.

System	208V Power	480V Power	Cooling (Tons)	Floor Space
Dawn	3.5 MW	1.5 MW	2,000	9,000 ft^2
Sequoia		15.0 MW	6,000	15,000 ft^2

B453 cooling can be used as air cooling, or if required as chilled water with additional facilities modification. All computer power is projected to be fairly reliable, clean, but not conditioned, and there is no UPS. The computer floor is 48" raised floor with 250 lbs/ft^2 loading. However, racks with up to 500 lbs/ft^2 floor loading can be accommodated with additional floor bracing. The overall system average floor loading (including isles between rows) can not exceed 250 lbs/ft^2. In addition, rolling weight of racks during installation can not exceed 250 lbs/ft^2. Power will be provided to racks by under floor electrical outlets supplied by LLNS to Offeror's specifications. Circuit breakers are available in wall panels that can be modified to Offeror's specifications. All other cables must be contained in cable trays supplied by LLNS to Offeror's specifications. Straight point-to-point cable runs can **NOT** be assumed. LLNS will provide floor tile cut to Offeror's specifications. In addition, it is anticipated that the selected Offeror's equipment will be placed in adjacent rows so that air intakes in racks from adjacent rows are abutting with Offeror's specified separations and hot air exhausts in racks from adjacent rack rows are abutting with Offeror's specified separations. That is, the racks will be placed so that there are HOT and COLD aisle ways between racks with chilled air entering in the COLD isles and warmed air exiting in the HOT aisles. LLNS would prefer system layouts with less than 3' for HOT isles and less than 4' for HOT isles. Offeror will describe any unique cooling solutions that allow for more efficient utilization of computer floor space and provide information on facilities impacts. LLNS will provide 2'x2' grated floor tiles with 80% void in "cold aisles" to product up to 2,500 CFM airflow per tile.

During installation, racks will transit from Offeror's delivery trucks through 12' (W) x 12' (H) roller doors to an interior delivery dock. Note that the delivery dock height is 45" and can only accommodate one tractor-semitrailer rig at a time. The racks will transit down a 300' long 9' 8" (W) x 8' 6" (H)

hallway. The racks must transit several doors of size 7' 10" (W) x 7' 10" (H) and ride a freight elevator up one floor. The freight elevator doors are 8' 4" (W) x 8'(H), the elevator area is 8' (W) x 12' (D) and the maximum loading of the freight elevator is 10,000 lbs. Racks may be staged on the B453 computer floor for unloading from packaging or unpackaged on the interior delivery dock.

After acceptance, the Dawn and Sequoia systems will be migrated to classified operation with access to the SCF networking faculties. In addition, Dawn and Sequoia will be physically located inside a Limited Access Area in a Vault Type Room (VTR). LLNS will only provide access to the room to authorized personnel under Authorized Escort. All on-site personnel will be required to submit applications for access and be approved by standard LLNS procedures prior to entry into this facility. All on-site personnel will require being DOE P-cleared or P-clearable. Offeror proposals should indicate if the on-site team has members that are other than U.S. citizens. Physical access to this computer facility by foreign nationals from sensitive countries (www.llnl.gov/expcon/sensitive.html) will not be allowed. Dialup capability and internet access to the system will be allowed up through acceptance, but not after systems are migrated to classified operation. Authorized individuals may be allowed remote access for running diagnostics and problem resolution. Interaction of the on-site engineering staff with factory support personnel may be limited in some ways (e.g., dissemination of memory dumps from the system may be restricted). These limitations emphasize the importance of local access to source code, particularly for operating system daemons.

On-site space will be provided for personnel and equipment storage.

Personnel must practice safe work habits, especially in the areas of electrical and mechanical work.

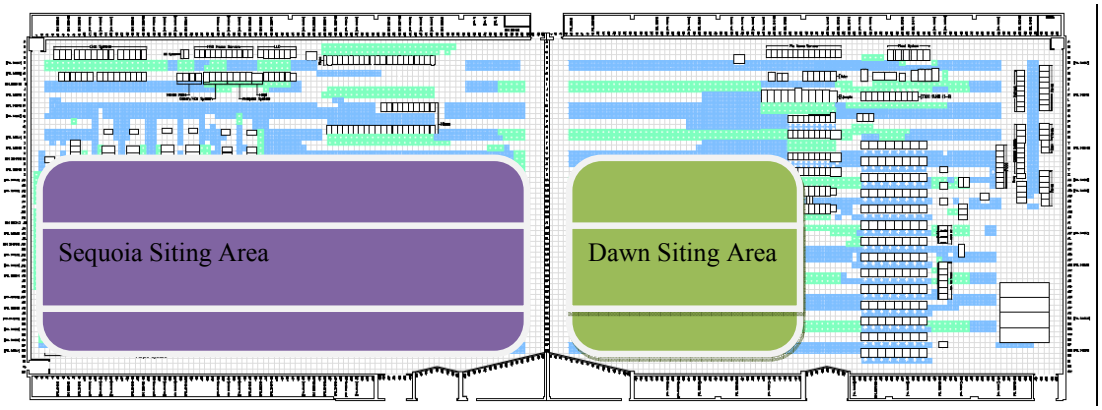


Figure 7-1: Dawn and Sequoia Siting locations within the LLNL B453 computer floors.

The Dawn and Sequoia systems will be installed and physically located inside an exclusion area within a security area. Due to the fact that the TSF will be an unclassified work area during construction, the TSF first computer floor will be unclassified work area until after Sequoia acceptance and stabilization. The TSF first computer floor will become a classified vault type room (VTR) as part of the migration of Sequoia to classified operations. There will be no dialup capability to classified systems. No remote diagnostics will be allowed. Interaction of the on-site engineering staff with factory support personnel may be limited in some ways (e.g., dissemination of memory dumps from a classified system may not be

allowed). This limitation emphasizes the importance of local access to source code, particularly for operating system daemons. Head disk assemblies (HDAs) from disks used for classified processing cannot be taken off-site or returned to the factory. All on-site personnel will require to be DOE Q-cleared or Q-clearable. It will be extremely difficult to provide LLNL site access to foreign nationals.

On-site space will be provided for personnel and equipment storage. They will be expected to practice safe work habits, especially in the areas of electrical, mechanical, and laser activities.

7.1 Power & Cooling Requirements (TR-1)

Offeror will minimize the power and cooling required by the proposed systems. Offeror may provide documentation for the estimated total amount of power in kW (kilowatts) required by the complete Dawn and Sequoia systems including any subsystems (e.g., SN, LN, RAID disks, external networking, etc.) and the estimated total amount of cooling in BTU (British Thermal Units) required by the complete Dawn and Sequoia systems. These estimates may be based on the actual proposed configuration and not gross over estimates based on "worst case assumptions and maximal configurations." In Offeror proposal, Offeror may indicate the basis for these estimates (e.g., engineering estimates of individual components and power supply efficiencies or simulation results or actual measurements, etc.) and the degree of uncertainty in them (e.g., ±5%, +10%/-5%). Offeror may list separately room air and any liquid cooling required for each system, in relation to the heat load created by operation of each system. Offeror may provide monthly updates to such estimates and review the basis for such updated estimates with Offeror until system installation. In addition, full system estimates for a fully configured system (with maximum amount of SDRAM, I/O subsystems, etc) may be provided. This documentation may break down the power and cooling loads to individual racks for each component part of the system.

7.1.1 Rack Power and Cooling (TR-1)

For racks with air cooling solutions that require all the cooling from air provided by the facility, each rack will not require more than 50 kW of power, and corresponding cooling, assuming front to back or bottom to top air cooling. If the rack requires more than the above power envelopes, then Offeror will propose less dense solutions and/or alternative cooling apparatus that reduces the intake air cooling load. Offeror will fully describe the liquid cooling apparatus and the implications for siting and facilities modifications (e.g., chilled water feeds, flow rates).

7.1.2 Rack PDU (TR-1)

Rack PDU will minimize the number of 480V circuit breakers required in wall panels at LLNL siting location. Specifically, redundant power feeds may not be proposed. One (1) power feed per rack would be ideal. In addition, the amperage of the required circuit breakers should be calibrated so that the utilization is maximized, but below 80% of the rated load during normal operation with user applications running. If the equipment in the rack requires more power during power-up (so called surge power), the rack PDU may not be calibrated to this surge power, but rather to the normal operating power with user applications running.

The Rack PDU will have on-off switches or switch rated circuit breakers to allow system administrator to power down all components in a rack with switches or circuit breakers in the PDU.

7.2 Floor Space Requirements (TR-1)

Offeror will minimize the floor space required by the proposed systems. Offeror may provide a proposed floor plan of the proposed Dawn and Sequoia systems that fits into the space requirements

specified below and showing the placement of all system components including, but not limited to: node racks, input/output device racks, interconnect racks, external networking frames, and RAID disk device frames.

7.2.1 Dawn Floor Space Requirement (TR-1)

The Dawn system may be installed in B453 east end of the west main computer room and may be less than 9,000 ft^2 as accommodated by this computer floor. This includes, all system components including, but not limited to: node racks, input/output device racks, interconnect racks, external networking racks, and RAID disk device racks.

7.2.2 Sequoia Floor Space Requirement (TR-1)

The Sequoia system may be installed in B453 west end of the east main computer room and may be less than 15,000 ft^2. This includes, all system components including, but not limited to: computer racks, input/output device racks, interconnect racks, external networking frames, and RAID disk device frames.

In the event Offeror has to provide additional hardware to achieve performance requirements specified in the subcontract, this equipment may be installed on this floor as well.

7.3 Rack Height and Weight (TR-1)

System racks will not be taller than 84" high (48U) and not place an average weight load of more than 250 lbs/ft^2 over the entire footprint of the system, including hot and cold isles. If Offeror proposes a rack configuration that weighs more 250 lbs/ft^2 over the footprint of the rack, then Offeror will indicate how this weight can be redistributed over more area to achieve a load less than 250 lbs/ft^2.

7.4 Rack Seismic Protection (TR-2)

Proposed system racks may be seismically qualified (in accordance with IEEE 344, ICC AC 156, or similar) and may be appropriately anchored to resist seismic load in accordance with the following criteria and requirements. The seismic anchorage design may conform to 2006 International Building Code using the following variables and may address both sliding and overturning of system racks:

- Occupancy Category IV ($I_E = 1.5$)
- Site Class C
- Geographic Coordinates: 37°41'13.59"N, 121°42'13.89"W
- Computer Floor Height = 0.5 Roof Height

The following two approaches meet the above rack seismic protection requirements. However the conventional rack tie down approach should only be proposed if the IsoBase approach cannot be made to work.

1. Use of WorkSafe Technologies' IsoBase isolation platforms. Note that use of IsoBase isolation platforms requires coordination with the manufacturer of the platforms for items including, but not limited to, installation coordination, platform openings for ventilation, added cable and wire lengths to accommodate seismic displacement, and attachment of nylon strap restraints at the base of the system racks.

2. Use of conventional tension rod seismic tie down anchorage. With conventional anchorage, the tension rods may be secured to the existing strip system of floor-anchored channel struts. If selected, LLNL can provide locations, details and strength limits of the channel system. Use of ductile steel components is required, and shear transfer for horizontal loads may be achieved through use of floor clips attached to the existing 24" square Tate FF 3000 cast aluminum access floor panels.

Adjacent racks in a row may be interconnected with a minimum of four corner bolts. Bolts may be ¼" diameter minimum and may be ASTM A 307 Grade A bolts, or equal. Bolts may be provided with suitable nuts and washers. Racks will be delivered pre-drilled. Field-drilling of holes for interconnection may not be made on the computer room floor prior to installation of ANY equipment.

System rack bases may have removable wheels and leveling feet, as well as at least corner anchorage points with sufficient strength and stiffness to transfer the seismic demands determined by the Offeror's Structural Engineer. If the racks have wheels, they may be adjusted or removed from racks prior to permanent installation so that they do not touch the floor or IsoBase™.

7.5 Installation Plan (TR-2)

Offeror will provide site installation instructions to LLNS delineating all site preparation work necessary to install and operate the systems, as configured in the subcontract. These instructions may delineate the type of electrical equipment required for installation (power couplings and placement, floor loading, etc.). This information will be delivered to LLNS within 30 days of receipt of subcontract for Dawn and within six months of receipt of contract for the technology refresh, if applicable, and Sequoia systems.

End of Section 7.0

8.0 Project Management

Achieving petascale performance for Dawn and Sequoia is an extremely daunting task. In order to be successful, the selected Offeror / LLNS partnership will need to focus efforts in three major areas: 1) scalability; 2) scalability; and 3) scalability. Hardware scalability is key to minimizing power consumption, RAS and application performance. Software scalability is key to system management and RAS and the ability of applications to be able to efficiently utilize O(1.5-3.0M) cores/threads. Integrated system scalability is the key to system usability, power consumption, system physical size and RAS. The challenges the Tri-Laboratory community faces in providing a platform on the scale of Dawn and Sequoia to meet the DOE Stockpile Stewardship programmatic requirements are no less so. Moreover, these challenges are not only technical, but also manifest themselves in the management and administration of the project. All have substantial impact regarding risk and therefore the probability of project success. LLNS recognizes that, ultimately, the selected Offeror is responsible for the successful integration of all the elements, including those acquired from third-parties, academia, and other ASC-related efforts, to provide the petascale computing environment needed to meet the national goals of the Stockpile Stewardship Program. LLNS, NNSA and the selected Offeror must recognize this acquisition as a primary institutional commitment. LLNS expects its partners to successfully meet this commitment.

The experience gained by Lawrence Livermore, Los Alamos, and Sandia in the installation of the first six generations of ASC Platforms: Red, Blue, White, Q, Purple (and BlueGene/L) and RedStorm, systems has demonstrated that such an activity taxes the resources and management capabilities of even the largest and best-managed organizations.

Some of the lessons collectively learned in fielding and integrating those systems into a useful scientific simulation environment include the following:

The most important lesson learned is that this effort, if it is to succeed, must truly be a “partnership” among all involved. While careful mutual planning on the part of LLNS and the selected Offeror is essential to meeting requirements, unforeseen events and changes are likely. These events can only be successfully dealt with by a partnership that goes beyond an ordinary vendor-customer relationship. It must be one in which teaming, mutual respect, and an honest desire to achieve success is present on the part of everyone involved.

Changes in a company’s technology roadmap can have significant consequences on the success of the project. Whether from development delay or fundamental changes in a company’s technology decisions, change is almost inevitable. It is therefore important that such changes be quickly evaluated for their impact on the project. In addition, strategies must be developed and discussed to mitigate technical and scheduling problems.

Component availability for system manufacture can affect delivery schedules. This is particularly true for new equipment, for which only a limited quantity of components is available. LLNS has also found this to be the case for some older components owing to the large volumes needed for a system of this size, as well as to the commitments lower-tier suppliers may have to other customers.

Manufacturing, assembly, and QA for a system of this size can tax even the largest companies. It is therefore important to ensure that sufficient capacity, without compromising quality assurance, is available at the times necessary to meet delivery schedules. In addition, the development and systems stress testing of software releases and patches is an on-going problem for ASC sized systems. This is

due to the fact that these systems are usually the largest systems fielded by a vendor by a wide margin. Careful planning of software testing and releases must be done in order to cost effectively test software.

Significant resources are needed at the factory for pre-delivery staging and testing. LLNS has found it best to perform pre-delivery staging and testing of portions of the system prior to shipment to LLNL to minimize installation problems later. It reduces the number of "DOA" and infant–mortality component failures, helps to ensure correct hardware, software and firmware operation, and allows for execution of company and LLNS test programs.

LLNS has found that such activity requires the selected Offeror to provide resources at the factory in the form of floor space, ancillary equipment (i.e., disks, interconnects), and personnel.

Installation needs to proceed in a logical, coordinated manner. Systems that are shipped without disks, for example, are generally not useable and take up valuable installation resources and floor space. This problem also speaks to the need for outstanding coordination among all elements within a company to ensure that hardware and software availability be coordinated (i.e., when new hardware is available the software to drive it is also available).

Shipment logistics have an impact. LLNS expects to have only approximately 1,500 square feet adjacent to the computer building loading dock to stage deliveries prior to installation in the computer room. This limitation should be taken into account when formulating delivery plans.

It is therefore important to quickly install each shipment as it arrives. Arrangements may be necessary to ensure that sufficient personnel with the appropriate training are available for installation, as well as factory-based resources to assist as needed if problem escalation is warranted. Again, the sheer size and complexity of this system may require that extraordinary measures be taken by the selected Offeror. Successful installation test completion will be required prior to the initiation of any acceptance test.

Acceptance testing is an extension of earlier testing. Although the pre-delivery and installation tests will identify many problems prior to acceptance testing, it has been LLNS' experience that new problems may surface. The availability of on-site and factory-based resources to correct such problems is important.

Stabilizing the system as quickly as possible is programmatically important. It is imperative that the selected Offeror and those supplying third-party products work closely to that end. This arrangement will require that applications engineers as well as hardware and system software engineers be on-site to resolve problems. LLNS access to the system and third-party source code, although important in earlier and later stages, is critical at this point, and cannot be overemphasized. Again, the unprecedented size and complexity of this system dictate that resources over and above the norm will undoubtedly need to be brought to bear.

Post-stabilization resource requirements will also be significant. It is easy to underestimate the number of hardware engineers and software and applications analysts with the appropriate experience and training required to maintain high reliability and availability, to make the best use of the system, and to resolve problems quickly.

It is also easy to underestimate the extent of the necessary spare parts inventory. Because of the classified environment, use of remote diagnostic procedures will not be allowed.

Because of the complexity of this activity, a very strong project plan is of great importance. The Offeror's understanding of LLNS' requirements, approach to meeting those requirements, commitment of resources, and attention to cost are critical to the success of the project. In the same vein, the

approach to managing this activity is critical. The need to have the support of corporate senior management and a major commitment to a quality assurance plan are also examples of areas critical to the success of the project.

The specific detailed planning and effort tracking and documentation requirements for the development and manufacturing efforts that will be delivered as part of the subcontract(s) are delineated in sections 8.2, Detailed Sequoia Plan Of Record

The specific target delivery milestones for the project are delineated in Section 8.3, Project Milestones.

8.1 Performance Reviews (TR-1)

Quarterly performance reviews will be conducted between the selected Offeror's corporate executives, the selected Offeror's Sequoia Project Team (also known as "technical team"), and LLNS. The selected Offeror will submit a Quarterly Project Status Report to LLNS at least five working days before each quarterly review. The report will provide the status of all work breakdown structure tasks and milestones in the critical path. It will also contain narrative descriptions of anticipated and actual problems, solutions, and the impact on the project schedule. Numbered action items will be taken, assigned, logged, and tracked by the Offeror. The minutes of all project reviews will be recorded in detail by the selected Offeror and provided to LLNS for approval within 5 working days after the review.

8.2 Detailed Sequoia Plan Of Record (TR-1)

This project envisions a quantum advance in delivered performance capability for ASC scientists and engineers. To successfully reach this level of delivered performance the selected Offeror may submit, within thirty (30) days of subcontract award, a full term, highly focused plan of record, per Sections 8.2.1, 8.2.2 and 8.2.3, delineating the management, research, development, acquisition, manufacturing, testing, demonstration, delivery, integration and acceptance testing activities to achieve the project goals. LLNS and the selected Offeror may jointly develop a detailed full term project plan of record for LLNS' approval. The plan at the time of submission must be accurate and up to date. At a minimum, the full term plan may contain the following components: management; hardware; software; risk assessment, mitigation and fallback strategies; collaborations. In addition, each year (by the end of the calendar year) the selected Offeror may develop a detailed year plan (Section 8.2.4) for the next calendar year and track the project during the year with this plan. The full term plan may be revised on an on-going basis to reflect the changes in the management team, actual development schedule, risk mitigation strategy and may be submitted for formal review semi-annually at the first and third quarterly meetings. The plan at the time of submission must be accurate and up to date (within ten days of submission). LLNS may review the submitted plan and provide the selected Offeror written comments within two weeks. The selected Offeror may revise the plan based on LLNS feedback and resubmit the plan within two weeks of receiving written comments.

8.2.1 Full-Term Project Management Plan (TR-1)

The selected Offeror will develop a detailed full term project management plan of record for LLNS review and approval. It is essential that the management plan be kept up to date with respect to changing personnel and company reorganizations and changes in the Offeror's Sequoia management structure. The plan will contain at least the following components:

Management teams and structure: The selected Offeror's Sequoia project will be managed with two teams: executive team and technical team. The executive team will meet quarterly and have

direct input from, and feedback to, the technical team and LLNS. The selected Offeror's designated NNSA Partnership Executive will meet quarterly with the Lawrence Livermore National Laboratory Director, NNSA Assistant Secretary for Defense Programs, NNSA Deputy Assistant Secretary for Strategic Computing and Simulation to track hardware and system software milestones as well as other strategic partnership issues. The technical team and LLNS will have quarterly face-to-face meetings, monthly video teleconferences and weekly teleconference calls. The selected Offeror will develop and revise quarterly a top-ten issues list. The monthly meeting will have a technical focus and go over project status and the action items stemming from the top-ten list. The quarterly meeting will be higher level and go over project status and recent technical issues and accomplishments. The management plan will list the members of the management and technical teams, provide their resumes and list their roles and responsibilities. The management plan will have an organizational chart of the management and technical teams and lines of reporting to various parts of the company.

Organization for core team: List the contributing organizations within the company and how they will be coordinated. For Open Source software, describe how the community contributions will be managed. Provide an organizational chart of the company that depicts these groups and their lines of responsibility. Include hardware R&D, software R&D, productization, field team and applications support, manufacturing, purchasing and quality assurance. Indicate how these areas will be coordinated by the management team.

Full term project plan and schedule. Provide a Work Breakdown Structure (including milestones) for the project giving at least five levels of detail, as appropriate, with projected start and finish dates and interdependencies of deliverables. This project plan will elaborate on the tasks and milestones committed to in the scaleable systems development section and clearly delineate the project critical path tasks (see below). Provide a Project Schedule that starts at contract award and ends with successful contract termination. The schedule will be developed using the Critical Path Method (CPM) scheduling technique and will utilize the same numbering scheme as the Work Breakdown Structure. The Project Schedule will be placed under configuration control to ensure that all project schedule updates are accomplished in a manner that preserves an audit trail from the original Project Schedule to the current schedule status. The Schedule will contain sufficient detail to ensure that LLNS and the selected Offeror can measure progress on an appropriate number of milestones and tasks on any path or on parallel paths to measure progress and to determine the true critical path to project completion.

Risk reduction plan. In order to meet the project goals and objectives in a timely manner, indicate fall-back strategies that will become operative should delivery schedules not proceed as rapidly as predicted. Indicate additional resources that will be available, if applicable, to the effort in the event that problems develop. Indicate the potential impact to the program and the mitigation plan, should the potential occur. Delineate the problem escalation and resolution path. Risks will be categorized as to their impact (low, medium and high) and to their probability of occurrence (low, medium and high). The risk mitigation strategies will have decision dates specified for executive partnership decisions on the main plan vs. various fall-back strategies.

Open Source Collaboration Plan. As Open Source components are critical components to the success of the Sequoia objectives, Offeror will describe the overall strategy for interfacing with and managing the work flow of these groups. Offeror may describe how improvements produced in order to meet Sequoia requirements will be fed back to the community. Offeror may describe the role for Tri-Laboratory Open Source contributions will be managed. Of particular

interest is how Offeror proposes to interface with LLNS for the deployment of Lustre parallel file system, MOAB/SLURM resource management and other Open Source code development tools.

8.2.2 Full-Term Hardware Development Plan (TR-1)

The hardware (as defined in Sections 2.0, 4.0, 6.0 and 7.0) full-term development plan may contain at least the following components:

Processor Technology. Identify the planned milestones for processor development that lead to those to be deployed in the Dawn and Sequoia systems. In particular, provide milestones for silicon process development, sampling, engineering quantities, and production quantities for each processor generation between the Dawn and Sequoia systems.

Node Development. Provide the planned tasks and milestones for CN, ION, SN and LN product development for system generations covered by this contract. Include tasks and milestones for at least the following development areas: memory architecture; cache coherency protocols; ASIC development; performance modeling efforts; applications analysis; functional verification test; system test. Indicate how and when this technology will be inserted at LLNL to meet subcontract milestones.

CN Interconnect Development. Provide the planned tasks and milestones for CN interconnect research and development between the Dawn and Sequoia system generations. Include tasks and milestones for at least the following development areas: switch ASIC development; interface components; cabling components; NIC and/or router design; overall BER reduction; microcode, driver and MPI software development including support for multiple network adapters per node; functional verification test; system test. Indicate how and when this technology will be inserted at LLNL to meet subcontract milestones.

SAN Access Development. By SAN access, LLNS means the standards-based networking (e.g., InfiniBand™ 4x QDR and 10 Gigabit Ethernet) to connect the Dawn and Sequoia clusters to system area networks at LLNL. It also includes the IO path (hardware) and supporting software for accessing the LLNS provided Lustre file system. Provide the planned tasks and milestones for development of SAN access to the parallel I/O subsystem including functional verification and system test. The SAN access test plan must delineate component and end-to-end testing. End-to-end testing is defined as starting (or ending) at an LLNL parallel application running on the Dawn and Sequoia clusters through the parallel I/O libraries down through the transport layers, through the device drivers and RAID hardware to the disks. Include tasks and milestones for at least the following development areas: RAID adapters; SAN networking; disk development; remote I/O devices and links; architecture planning and modeling; development and architecture. Indicate how and when this technology will be inserted at LLNL to meet subcontract milestones.

System Scalability and Performance Testing. Provide the planned tasks and milestones for the scalability testing of system components. Include development of hardware for reliability, availability and serviceability (RAS).

8.2.3 Full-Term Software Development Plan (TR-1)

The software (as defined in Sections 3.0, 5.0 and 6.0) full term project plan may contain at least the following components. In each of these areas, the specific Open Source community model and development, testing and support plans should be discussed.

LWK Development. Provide the planned tasks and milestones for Light-Weight Kernel operating system development. Include tasks and milestones for at least the following development areas: diminutive noise environment for petascale applications scalability; support for dynamically linked libraries and Python based applications; exploitation of novel SMP parallelism techniques (TM/SE); shared memory regions; boot and RAS; user access to hardware performance monitoring hardware; low latency user thread mechanisms for Pthreads, OpenMP and TM/SE; memory management; full 64-bit support, CN interconnect access; ION function shipping.

BOS Development. Provide the planned tasks and milestones for Linux operating system development. Include tasks and milestones for at least the following development areas: IO function shipping from CN support; OS Virtualization (if applicable); shared memory locality of reference (if applicable); support for hardware and system performance monitoring; low latency user callable thread mechanism; memory management; full 64-bit support, journaled file systems; reboot time minimization; high-performance access to SAN and External networking.

Integrated System Management Development. Provide the planned tasks and milestones for development of infrastructure and tools to manage the CN, ION, SN and LN as a single system via integrated system management. Include tasks and milestones for at least the following development areas: system administration tools for installing and managing the cluster as a single system; user management and system scalable authentication mechanisms; load balancing between LN;

Reliability Availability and Serviceability. Provide the planned tasks and milestones for the development of scalable end-to-end RAS infrastructure and tools across CN, ION, SN and LN. Include tasks and milestones for at least the following development areas: Open Source RAS database on SN; RASD tools and infrastructure; system component discovery and monitoring; scalable FRU failure diagnostics and predictive failure approaches; error detection vs. retry; scalable system and CN interconnect diagnostics.

Resource Management Support. Provide the planned tasks and milestones for resource management development. Include tasks and milestones for at least the following development areas: Moab/SLURM required interfaces; system monitoring tools; system initiated checkpoint/restart; scalable and reliable job launch, termination and control;

Parallel I/O Development. Provide the planned tasks and milestones for supporting high-performance IO for petascale parallel applications. Include tasks and milestones for at least the following development areas: CN to ION IO function shipping; SAN network drivers; IO path performance tuning; and MPI I/O parallel I/O development.

Compiler and Runtime Development. Provide the planned tasks and milestones for baseline language (C, C++, Fortran03 and Python) development. Include tasks and milestones for at least the following development areas: mixed language support; compatibility with GNU compiler runtime; exploitation of novel hardware features for automatic and directed parallelization (SE/TM, OpenMP) of applications; latency reduction techniques; compiler optimization for specialized hardware (e.g., vectorization or SIMD); migration support (from Dawn to Sequoia). Indicate any points where compatibility with Fortran 77 applications decreases. Specific attention to ASC applications performance and interaction with LLNS in this area is required.

Message Passing Environment. Provide the planned tasks and milestones for message passing development. Include tasks and milestones for at least the following development areas: bandwidth and latency targets for MPI; MPI standard tracking; integration with debuggers, profilers and performance analysis tools; interoperability to cluster external resources.

Code Development Tools. Provide the planned tasks and milestones for code development tools development. Include tasks and milestones for at least the following development areas: petascale code development tools infrastructure; remote process control tools interface; scalable CDT daemon launch and bootstrapping; parallel make, profilers, debuggers, application performance monitoring tools, GUI development for code development tools.

8.2.4 Detailed Year Plan (TR-1)

Each year (by the end of the calendar year) the selected Offeror may develop and submit to LLNS, for review and approval, a detailed year plan for the next calendar year. The selected Offeror may track the project during the year with this plan. This plan may be revised on an on-going basis to reflect the changes in the actual development schedule and may be submitted for formal review quarterly at the quarterly meetings. The plan at the time of submission will be accurate and up to date. At a minimum, the detailed year plan will contain the following components: Work Breakdown Structure (WBS), Gantt chart, Offeror product Plan of Record line items; I/O test plan, software test plan, system manufacturing and testing plans (in the years with system deliveries) and descriptive narrative. This plan will cover the hardware (Section 8.2.2) and software (Section 8.2.3) areas above with more detail and precision.

LLNS will review the submitted plan and provide the selected Offeror written comments within two weeks. The selected Offeror will revise the plan based on LLNS feedback and resubmit the plan within two weeks of receiving written comments.

8.3 Project Milestones (TR-1)

Because of the need to meet Advanced Scientific Computing and Stockpile Stewardship Program goals as quickly as possible, the project schedule and milestones are of critical importance. Meeting the following milestones is critical to the success of the project; earlier is much better. In addition, rapid insertion of technology is important. To this end, LLNS envisions a process whereby the systems are delivered, stabilized, accepted, brought under load of science runs (small number of large core count and memory footprint, long running jobs with few users active at a time), brought under "limited availability" load (programmatic workload with a limited number of users, typically 5-10 from each of LLNL, LANL and SNL, targeted specifically at achieving programmatic milestones) and finally "general availability" status (general ASC workload with no limits on the number of or type of work done by user accounts.

The implementation will entail the installation of Dawn and Sequoia systems at LLNL. Each system will be assembled from individual nodes that are interconnected with a high-speed, low-latency interconnect supplied by the selected Offeror. These systems will be connected to the site's local campus network and to the wide area network that interconnects the Tri-Laboratory community. Access to the resources will be provided locally via the site's existing campus networks and remotely through the ASC-supplied WAN.

The following milestones are provided as a general framework. These milestones include target dates based on ASC programmatic requirements and anticipated fiscal year funding. These target dates are TR-1 requirements (i.e., not mandatory) and can be modified to more closely match an

Offeror's product roadmap. However, there is a significant value to LLNS and the ASC Program for early delivery of technology and capability. In particular, Sequoia acceptance in 3QCY11 is highly desirable. Offeror will provide LLNS, in its proposal response, a set of milestones for this section and an associated payment schedule that is applicable to Offeror's proposed development and deployment timeline and methodology. This general framework assumes the build-demo-deliver scenario for fielding Dawn, technology refresh and Sequoia clusters.

8.3.1 Full-Term Sequoia Plan of Record (TR-1)

The selected Offeror will provide a detailed full-term project management plan, and a full-term hardware development and software development plan thirty (30) days after subcontract award.

8.3.2 FY09 On-Site Support Personnel (TR-1)

Within thirty (30) days after subcontract award, the selected Offeror will supply at least two full-time equivalent on-site personnel as set forth in Section 6.4 with the following job functions:

One on-site systems programmer will provide solutions to the current top ten issues, as directed by LLNS, and will provide system administration and day-to-day operations support and be responsible for maintaining an accurate systems availability, MTBF and support response time statistics, as directed by LLNS;

One on-site applications analyst will provide expertise to LLNS code development teams in the areas of software development tools, parallel applications libraries and applications performance.

8.3.3 CY09 Plan and Review – Jan 2009

The selected Offeror will deliver a detailed plan of activities and deliverables for calendar year 2009 for LLNS review and approval in the first quarter of calendar year 2009 (1QCY09).

As part of the CY09 plan, LLNS will provide to the selected Offeror the Dawn Synthetic WorkLoad (SWL) test plan. The selected Offeror may participate in the definition of the Dawn SWL content. LLNS and the selected Offeror will mutually agree on the Dawn SWL test plan and the criteria for its successful completion. The test plan will include the following requirements: 1) it will be capable of execution in no more than seven (7) days; and 2) it will not impose technical requirements beyond those set out in this statement of work; and 3) execution of the SWL may utilize LLNS provided resource management (Moab/SLURM) and representative applications. In evaluating SWL progress, LLNS will not hold the selected Offeror responsible for Hardware and Software provided by LLNS. If after one week of running the SWL without successful completion due to LLNS supplied hardware or software, LLNS and the selected Offeror will mutually agree on a methodology for measuring the selected Offeror's deliverables against the requirements of this SOW without the impedance of LLNS supplied hardware or software. Neither party will unreasonably withhold agreement on such a methodology. The selected Offeror will use reasonable effort to find a workaround to LLNS supplied hardware or software. If no workaround can be found within a week, then LLNS will deem the selected Offeror compliant with all requirements that were impeded by LLNS supplied hardware or software. LLNS anticipates delivery of the Dawn SWL content (source code to SWL applications and tests) to the selected Offeror by January 15 2009.

This milestone is complete when LLNS reviews the CY09 plan with LLNS and LLNS' Technical Representative approves the CY09 plan.

8.3.4 Dawn Demonstration – Feb 2009 (TR-1)

Prior to shipment, the selected Offeror will demonstrate the Dawn system consistent with requirements in Sections 4.0, 5.0, 6.0 and 7.0. Offeror may successfully execute on the Dawn system, the Dawn SWL Dawn Pre-Ship test defined in Milestone 8.3.3.

This milestone is complete when the Dawn system successfully completes the exit criteria for the Dawn SWL Dawn Pre-Ship test and LLNS Technical Representative accepts the selected Offeror's plan to remedy any deficiencies and the equipment leaves the selected Offeror's facility. This plan may contain at a minimum: TPP LINPACK; HPC challenge benchmarks and the five marquee benchmarks run for four hours with correct answers and SAN performance and stability testing (8 hours of IOR stress test) on a mutually agreeable file system to representative RAID devices. Shipment of Dawn may be accomplished by the end of the first quarter calendar year 2009 (1QCY09). Earlier is better.

8.3.5 Dawn Acceptance – March 2009 (TR-1)

The selected Offeror will deliver and install and support the Dawn system consistent with requirements in Sections 4.0, 5.0, 6.0 and 7.0. Delivery will be to LLNL.

This milestone is complete when the Dawn system: 1) is fully installed and configured and successfully completes the Dawn system SWL acceptance test plan exit criteria, as verified by LLNS Technical Representative; 2) Offeror delivers a "statement of volatility" for all FRU that do not contain non-volatile memory or storage; and 3) LLNS Technical Representative accepts the selected Offeror's plan to remedy any deficiencies. This plan may contain at a minimum: TPP LINPACK; HPC challenge benchmarks and the five marquee benchmarks run for four hours with correct answers and Lustre performance and stability testing (8 hours of IOR stress test). This milestone may be completed by the end of first quarter of calendar year 2009 (1QCY09). Earlier is better. Completion of this milestone starts the five year Dawn maintenance period.

8.3.6 GFY10 On-Site Support Personnel – Oct 2009 (TR-1)

The selected Offeror will provide three on-site personnel as set forth in Section 6.4. The on-site personnel will provide systems programming support, systems technical support and applications analysis as directed by LLNS, during GFY10.

8.3.7 GFY10 Dawn Support – Oct 2009 (TR-1)

The selected Offeror will supply self hardware maintenance and software support for the Dawn system during GFY10.

8.3.8 CY10 Plan and Review – Dec 2009 (TR-1)

The selected Offeror will provide a detailed plan of activities and deliverables for calendar year 2010 for LLNS review and approval in the fourth quarter of calendar year 2009 (4QCY09).

8.3.9 Sequoia Prototype Review – June 2010

The selected Offeror will deliver a final report on the Sequoia prototype results for LLNS review and approval. As part of this review, LLNS and the selected Offeror will review the progress of Sequoia research and development in meeting the requirements of this Statement of Work. At a minimum the following Sequoia prototype results will be addressed at this review:

Full Sequoia TPP LINPACK and five marquee benchmarks performance estimate provided

Measurements of DGEMM, LINPACK, five marquee benchmarks

Measurements of CN Interconnect MPI collective, point-to-point and all-to-all performance

Measurements of CN Interconnect delivered minimum bi-section bandwidth

Measurements of CN Interconnect link delivered bandwidth and latency, if possible

Measurements of end-to-end parallel IO through the SAN interfaces to a mutually agreeable global parallel file system on representative RAID devices.

The Sequoia job-scheduling, RASD infrastructure, petascale code development tools strategy will be finalized at this review. All LLNS required APIs and infrastructure requirements will be finalized.

The SAN interconnect for Sequoia will be finalized.

This milestone is complete when the project is reviewed at a face-to-face meeting, an updated plan is approved by LLNS Technical Representative in writing.

8.3.10 GFY11 On-Site Support Personnel – Oct 2010 (TR-1)

The selected Offeror will provide three on-site personnel as set forth in Section 6.4. The on-site personnel will provide systems programming support, systems technical support and applications analysis as directed by LLNS, during GFY11.

8.3.11 GFY11 Dawn Support – Oct 2010 (TR-1)

The selected Offeror will supply self hardware maintenance and software support for the Dawn system during GFY11.

8.3.12 CY11 Plan and Review – Dec 2010 (TR-1)

The selected Offeror will provide a detailed plan of activities and deliverables for calendar year 2011 for LLNS review and approval in the fourth quarter of calendar year 2010 (4QCY10).

8.3.13 Sequoia Build – March 2011 (TR-1)

The selected Offeror will build the Sequoia system at the selected Offeror's facility in accordance with Sections 2.0, 3.0, 6.0, and 7.0. This milestone is complete when all hardware components have been installed, and a majority of the required software is installed and a prototype SAN has been built, as verified by LLNS Technical Representative. The target build date for Sequoia is first quarter of calendar 2011 (1QCY11).

8.3.14 Sequoia Demonstration – June 2011 (TR-1)

Prior to shipment, the selected Offeror will demonstrate the Sequoia system consistent with requirements in Sections 2.0, 3.0, 6.0, and 7.0, which is sized according to LLNS' exercised option(s), if any. The selected Offeror will successfully execute on the Sequoia system, the Sequoia SWL Sequoia pre-ship test defined in Milestone 8.3.3. This plan will contain at a minimum: Sequoia meets the system performance metric M in Section 2.1.1 with correct answers; TPP LINPACK with at least 75% efficiency; HPC challenge benchmarks and SAN performance and stability testing (8 hours of IOR stress test) on a mutually agreeable file system to representative RAID devices.

This milestone is complete when: 1) the Sequoia system successfully completes the exit criteria for the Sequoia SWL Sequoia pre-ship test, as verified by LLNS Technical Representative; 2) LLNS Technical Representative accepts the Offeror's plan to remedy any deficiencies; and 3)

the equipment leaves the Offeror's facility. Shipment of Sequoia may be accomplished by the end of the second quarter calendar year 2011 (2QCY11). Earlier is better.

8.3.15 Sequoia Acceptance and LA – Sept 2011 (TR-1)

The selected Offeror will deliver and install and support the Sequoia system consistent with requirements in Sections 2.0, 3.0, 6.0, and 7.0, which is sized according to LLNS exercised option(s), if any . Delivery may be to LLNL.

This milestone is complete when the Sequoia system: 1) is fully installed and configured; successfully completes the exit criteria for the Sequoia SWL Sequoia acceptance test, as verified by LLNS Technical Representative; 2) Offeror delivers a "statement of volatility" for all FRU that do not contain non-volatile memory or storage; and 3) LLNS Technical Representative accepts the Offeror's plan to remedy any deficiencies. This plan will contain at a minimum: Sequoia meets the system performance metric M in section 2.1 with correct answers; TPP LINPACK with at least 80% efficiency; HPC challenge benchmarks and Lustre performance and stability testing (8 hours of IOR stress test). Acceptance of Sequoia will be accomplished by the end of the third quarter calendar year 2011 (3QCY11). Earlier is better. Completion of this milestone starts the five (5.0) year Sequoia maintenance period.

8.3.16 GFY12 On-Site Support Personnel – Oct 2011 (TR-1)

The selected Offeror will provide two on-site personnel as set forth in Section 6.4. The on-site personnel will provide systems programming support, systems technical support and applications analysis as directed by LLNS, during GFY12.

8.3.17 GFY12 Dawn Support – Oct 2011 (TR-1)

The selected Offeror will supply self hardware maintenance and software support for the Dawn system during GFY12.

8.3.18 Sequoia Production General Availability – Dec 2011 (TR-1)

The selected Offeror will improve the delivered performance of mutually agreeable ASC applications by improving the scalability of the system with these applications and improve the delivered SMP parallelization within an MPI task and individual MPI task and/or thread delivered performance through improvements in the provided compilers and runtime system. The selected Offeror will improve stability of the Sequoia system and keep Sequoia availability and utilization high enough to support a mixed capability and capacity productive usage by the ASC and SSP program elements. Additionally, the selected Offeror will deliver improvements to the scalability of code development tools and the use of those tools on ASC petascale applications on the Sequoia system.

This milestone is complete, as verified by the LLNS Technical Representative, when Sequoia satisfies the reliability requirements in Section 6.0; and Sequoia system achieves the level of productive usage by the ASC Program as the Tri-Laboratory capability platform for a limited set of ASC milestone users with capability jobs and thereby achieves General Availability usage status. This milestone may complete by the end of the fourth quarter of calendar year 2011 (4QCY1). Earlier is better.

8.3.19 GFY13 On-Site Support Personnel – Oct 2012 (TR-1)

The selected Offeror will provide two on-site personnel as set forth in Section 6.4. The on-site personnel will provide systems programming support, systems technical support and applications analysis as directed by LLNS, during GFY13.

8.3.20 GFY13 Dawn Support – Oct 2012 (TR-1)

The selected Offeror will supply self hardware maintenance and software support for the Dawn system during GFY13.

8.3.21 GFY13 Sequoia Support – Oct 2012 (TR-1)

The selected Offeror will supply self hardware maintenance and software support for the Sequoia system during GFY13.

8.3.22 GFY14 On-Site Support Personnel – Oct 2013 (TR-1)

The selected Offeror will provide two on-site personnel as set forth in Section 6.4. The on-site personnel will provide systems programming support, systems technical support and applications analysis as directed by LLNS, during GFY15.

8.3.23 FY14 Dawn Support – Oct 2013 (TR-1)

The selected Offeror will supply self hardware maintenance and software support for the Dawn system commencing October 1, 2014 and ending December 31, 2014.

8.3.24 GFY14 Sequoia Support – Oct 2013 (TR-1)

The selected Offeror will supply self hardware maintenance and software support for the Sequoia system during GFY14.

8.3.25 GFY15 On-Site Support Personnel – Oct 2014 (TR-1)

The selected Offeror will provide two on-site personnel as set forth in Section 6.4. The on-site personnel will provide systems programming support, systems technical support and applications analysis as directed by LLNS, during GFY15.

8.3.26 GFY15 Sequoia Support – Oct 2014 (TR-1)

The selected Offeror will supply self hardware maintenance and software support for the Sequoia system during GFY15.

8.3.27 GFY16 On-Site Support Personnel – Oct 2015 (TR-1)

The selected Offeror will provide two on-site personnel as set forth in Section 6.4. The on-site personnel will provide systems programming support, systems technical support and applications analysis as directed by LLNS, during GFY16.

8.3.28 GFY16 Sequoia Support – Oct 2015 (TR-1)

The selected Offeror will supply self hardware maintenance and software support for the Sequoia system during GFY16.

End of Section 8.0

9.0 Performance of the System

Each of the Sequoia benchmarks has been carefully chosen and developed to represent a particular subset and/or specific characteristic of the expected ASC workload on the Dawn and Sequoia systems, which consists of solving complex scientific and engineering problems using a variety of computational techniques.

Sequoia benchmarks serve three purposes in the Sequoia procurement. First, the benchmarks provide the Offeror with the opportunity to provide LLNS concrete data (in its RFP response) associated with the performance, reliability and scalability of the proposed systems on programmatically important applications. Offeror's proposal response should include information of many kinds, including a report on the results of running Sequoia benchmarks on existing Offeror hardware, simulators and extrapolations to future proposed systems. In this role, the benchmarks play an essential role in the proposal evaluation process. Second, the benchmarks will be used as an integral part of the systems SWL tests (see Section 8.3.3) in order to assess the proposed systems ability to meet or exceed Sequoia performance, scalability and stability technical requirements. Third, these benchmarks will be used to assess continuous performance improvement of compiler and other critical software technologies during development and after acceptance over the lifetime of the Dawn and Sequoia systems.

While performance is an important consideration in any computer acquisition, for the ASC Program it matters greatly how improved performance is achieved. See Section 9.4.1 for a discussion of allowed modifications to the Sequoia benchmarks.

The Sequoia benchmarks described below may be executed by the Offeror and results presented in its proposal response for the purpose of measuring the execution performance and compiler capabilities of existing systems that may be proposed. In addition, these benchmark results from existing systems may be used by Offeror to extrapolate and/or estimate the benchmark performance on future proposed systems. The general requirements and constraints outlined below apply to all of the benchmark codes. Additional requirements and/or constraints found in individual benchmark readme files apply to that individual benchmark.

Although all of the benchmark results are considered important and will be carefully analyzed by LLNS during proposal evaluation, LLNS understands that Offerors are working with limited resources. The benchmarks are divided into three tiers to give Offerors the relative priority of the benchmarks. The Sequoia Marquee Benchmarks, also known as the Tier 1 codes, are designated as TR-1 requirements and Offeror may report results in the response evaluation as described in the proposal evaluation attachment. Tier 2 codes are designated as TR-2 requirements, while the Tier 3 codes are designated as TR-3 requirements. All of the Tier 3 codes are "micro kernels" that are provided as a courtesy to Offerors who wish to report results from prototype node or simulators of processor or node that are not yet available as hardware that are proposed as offerings for the Sequoia procurement. "Micro kernels" also serve as compiler challenges, especially for threading and the exploitation of Vector or SIMD hardware.

ASC Sequoia Benchmarks									
		Language				Parallelism			Description
Tier	Code	F	Py	C	C++	MPI	OpenMP	Pthreads	

1	UMT	X	X	X	X	X			Marquee performance code. Single physics package code. Unstructured-Mesh deterministic radiation Transport
1	AMG			X		X	X		Marquee performance code. Algebraic Multi-Grid linear system solver for unstructured mesh physics packages
1	IRS			X		X	X		Marquee performance code. Single physics package code. Implicit Radiation Solver for diffusion equation on a block structured mesh
1	SPhot	X				X	X		Marquee performance code. Single physics package code. Monte Carlo Scalar PHOTon transport code
1	LAMMPS				X	X			Marquee performance code. Full-system science code. Classical molecular dynamics simulation code (as used)
2	Pynamic		X			X			Subsystem functionality and performance test. Dummy application that closely models the footprint of an important Python-based multi-physics ASC code
2	CLOMP			X			X		Subsystem functionality and performance test. Measure OpenMP overheads and other performance impacts due to threading
2	FTQ			X				X	Fixed Time Quantum test. Measures operating system noise
2	IOR			X			X		Interleaved or Random I/O Benchamrk. IOR is used for testing the performance of parallel filesystems using various interfaces and access patterns.
2	Phloem MPI Benchmarks			X		X			Subsystem functionality and performance tests. Collection of independent MPI Benchmarks to measure the health and stability of various aspects of MPI performance including interconnect messaging rate, latency, aggregate bandwidth, and collective latencies under heavy network loads.
2	Memory Benchmarks								**Memory Subsystem functionality and performance tests.** Collection of STREAMS and STRIDE memory benchmarks to measure the memory subsystem under a variety of memory access patterns
3	UMTMk	X							Threading compiler test and single core performance
3	AMGMk			X			X		Sparse matrix-vector operations single core performance and OpenMP performance
3	IRSMk			X					Single core optimization and SIMD compiler challenge
3	SPhotMK	X							Single core integer arithmetic and branching performance
3	CrystalMK			X					Single core optimization and SIMD compiler challenge.

Table 9-1: ASC Sequoia Benchmarks are categorized into three tiers of importance.

9.1 Benchmark Suite

The benchmarks are listed in Table 9-1. Offeror may execute these benchmarks to measure the execution performance and compiler capabilities of the reference system to the extent defined in the benchmark readme file for each code. The general requirements and constraints outlined below apply to all of the benchmark codes. Additional requirements and/or constraints found in individual benchmark readme files apply to that individual benchmark.

The benchmark programs are available via the Web at the following URL:

https://asc.llnl.gov/sequoia/benchmarks/

For each benchmark code there is a brief summary, a tar file, and a change log. Tar files also contain test problems and instructions for determining that the code has been built correctly. For the

four IDC workload benchmarks there is also a file describing a Sequoia specific set of problems to be run for the RFP.

In addition to the marquee benchmarks discussed extensively in Section 9.1.1, the benchmark suite contains an additional eleven benchmarks that must be run on the reference system and will be involved in later acceptance of the machine–five groups of functionality tests (Section 9.1.2), and five micro-kernels (Section 9.1.3).

An Excel spreadsheet called "Sequoia_Benchmark_Results" is available on the benchmark website that should be used to report the results for all runs reported as part of the Offeror's proposal response. Each reported run must explicitly identify: 1) the hardware and system software configuration used; 2) the build and execution environment configuration used; and 3) the source change configuration used. The spreadsheet contains three worksheets that define the specific characteristics to be reported for each of these three configuration types.

9.1.1 Sequoia Marquee Benchmarks

The five Sequoia Marquee Benchmarks will be used to measure performance for two ASC workload types; the integrated design code (IDC) workload, and the science simulation workload. Each marquee benchmark has its own figure of merit (FOM), and weights are defined for aggregating individual FOMs into a single "benchmark FOM".

The IDC workload is emulated using the first four marquee codes (UMT, AMG, SPhot, and IRS). The target performance for this set is good scaling across all processors of a system with multiple, simultaneous "Purple-sized" jobs. This capacity for "Purple capability level" usage model is of direct relevance to ASC program plan milestones, and NNSA uncertainty quantification mission requirements. The specific definition of "multiple, simultaneous" jobs, and the sustained FOM requirements are found in Section 9.4.2.

The science simulation workload is emulated using the open source LAMMPS code (22 Jun 2007 C++ version). The EAM potential, weak scaling problem will be used, as defined on the LAMMPS web site under "benchmarks". As a classical molecular dynamics problem, the computing and inter-processor communication needs are well characterized and widely understood. The target performance demonstration for the LAMMPS benchmark is excellent scaling across all processors of a system with a single job. The details of the sustained FOM requirement are found in Section 9.4.2. These demonstrations are expected to bring recognition to the ASC Program, and to the Offeror, as well as to the larger scientific high performance computing community.

See Section 9.3 for more information on the marquee benchmark test procedures.

9.1.1.1 UMT Marquee Benchmark (TR-1)

The UMT Sequoia Marquee benchmark performs 3D, deterministic, multi-group, photon transport on an unstructured mesh. The transport algorithm solves the first-order form of the time-dependent Boltzmann transport equation. The energy dependence is modeled using multiple photon energy groups. The angular dependence is modeled using a collocation of discrete directions, or "ordinates." The spatial variable is modeled with an "upstream corner balance" finite volume differencing technique. The solution proceeds by tracking through the mesh in the direction of each ordinate for each energy group, accumulating the desired solution on each zone in the mesh. Hence, memory access patterns may vary substantially for each ordinate on a given mesh and the entire mesh is "swept" multiple times.

This code was chosen because the core computational techniques and software engineering methods are congruent with those that are anticipated to consume many computer cycles in support of ASC applications milestones over the lifetime of the Sequoia machine. Due to the complex data structures utilized, UMT performance is dominated by delivered memory bandwidth on a single node. Communication has a much smaller impact. This benchmark also demonstrates a critically important ASC programming methodology: an application implemented in multiple languages (Fortran95, C, and C++), controlled by Python scripts, and used with OpenMP thread parallelism within an MPI task and MPI messaging between MPI tasks. Multiple MPI tasks may be used on a single compute node and multiple cores/threads may be used within an MPI task. Its use as a Sequoia marquee demonstration benchmark is to validate the correct system hardware and software correct functionality and stability for a code that stresses the memory and communications subsystems with mixed language and parallelism implementation. UMT scales well to very large core/thread counts.

The natural "raw" Figure of Merit (FOM) for UMT is corner flux iterations per second. See Section 9.4 for a more detailed discussion on figures of merit.

The list of "Sequoia specific" runs found on the Sequoia benchmark web site (see Section 9.3) are to be run on the Offeror's benchmark system and submitted as part of its proposal response. For the selected Offeror, this same problem set will be repeated before shipment and during the acceptance testing for both the Dawn and Sequoia systems.

In addition to the above problem runs, Offeror will also run UMT as part of the sustained, aggregate, weighted figure of merit test described in Section 9.3. The exact problem size will necessarily be determined by the actual Dawn, and Sequoia system sizes, the available application memory, and the achieved sustained computation rate.

9.1.1.2 AMG Marquee Benchmark (TR-1)

The AMG marquee benchmark uses algebraic multi-grid algorithms to solve large, sparse linear systems of the sort that arise from implementing physics simulations on unstructured or block structured meshes. AMG is part of a larger solver library called hypre that is used extensively by ASC, and often controls the overall performance of these codes. AMG's use as a Sequoia marquee demonstration benchmark is to validate the correct system hardware and software functionality and stability for a code that stresses the memory and communications subsystems.

AMG is written in standard C. The performance of AMG is strongly influenced by the amount of main memory bandwidth and small message inter-node communication performance. To date, AMG has concentrated on using MPI for parallelism, and all current production uses only MPI parallelism.

An attempt to introduce OpenMP parallelism was made many years ago, but poor performance and the lack of a compelling need to use node parallelism within an MPI task at the time led to the discontinuation of threading efforts in AMG. The coding for this effort was left in place in AMG, but it has not been kept up to date as AMG have been further developed and expanded. Versions 0.9.2 and earlier of the tar file on the website include this original implementation.

Beginning with version 0.9.3 of the tar file, the OpenMP implementation has been changed to improve the OpenMP performance of AMG for the benchmark test problems only.

The natural "raw" FOM for AMG is solution vector updates per second and is computed as the linear system size* iterations required to achieve a specified accuracy in the solution/solve time See Section 9.4 for a more detailed discussion on figures of merit.

The list of "Sequoia specific" runs found on the Sequoia benchmark web site (see section 9.3) are to be run on the Offeror's benchmark system and submitted as part of its proposal response. For the selected Offeror, this same problem set will be repeated before shipment and during the acceptance testing for both the Dawn and Sequoia systems.

In addition to the above problem runs, the Offeror will also run AMG as part of the sustained, aggregate, weighted figure of merit test described in Section 9.3. For the sustained, aggregate FOM, AMG's contribution to the aggregate will be the average of the FOM obtained using both solver 3 and solver 4. The exact problem size will necessarily be determined by the actual Dawn, and Sequoia system sizes, the available application memory, and the achieved sustained computation rate.

9.1.1.3 IRS Marquee Benchmark (TR-1)

The IRS Sequoia Marquee Benchmark iteratively solves a 3D radiation diffusion equation set on a block-structured mesh. In addition to representing an important physics package, this benchmark is also representative of a style of writing C for loops and array indexing that is used in several production physics packages. IRS's use as a Sequoia marquee demonstration benchmark is to validate the correct system hardware and software functionality and stability for a code that stresses the memory and communications subsystems. On past ASC supercomputers, IRS was usually sensitive to OS "noise" as a cause for poor parallel scaling. This sensitivity arises from the use of many global reduction operations.

IRS is written in standard C. It uses MPI message passing between MPI tasks and OpenMP parallelism within each MPI task. Multiple MPI tasks may be used on a single compute node and multiple cores/threads may be used within an MPI task. The measured OpenMP efficiency within an MPI task on existing ASC systems is considered good.

The natural "raw" FOM for IRS is *zone temperature iterations per second* and is computed as the number of zones in the problem * iterations performed / runtime. See Section 9.4 for a more detailed discussion on figures of merit.

The list of "Sequoia specific" runs found on the Sequoia benchmark web site (see Section 9.3) are to be run on the Offeror's benchmark system and submitted as part of its proposal response. For the selected Offeror, this same problem set will be repeated before shipment and during the acceptance testing for both the Dawn and Sequoia systems.

In addition to the above problem runs, the Offeror will also run IRS as part of the sustained, aggregate, weighted figure of merit test described in Section 9.3. The exact problem size will necessarily be determined by the actual Dawn, and Sequoia system sizes, the available application memory, and the achieved sustained computation rate.

9.1.1.4 SPhot Marquee Benchmark (TR-1)

The SPhot Sequoia Marquee Benchmark implements Monte Carlo photon transport on a small, 2D structured mesh. Although current ASC codes focus on 3D and other mesh types, SPhot's computational kernel is still very representative of the single CPU performance controlling characteristics. Much production computer time is spent performing this fundamental computational kernel. As a benchmark, SPhot's computation phase (the only

part that influences the FOM) places no load on the inter-processor communication system of a parallel computer. The collection of edit information does use MPI collectives. The algorithm is "embarrassingly" parallel. In spite of this, running SPhot on large numbers of processors has found that perfect scaling is not always achieved.

SPhot is written in Fortran77, and uses MPI between nodes and OpenMP on a node.

The natural "raw" FOM for SPhot is *particle track updates per second*. See Section 9.4 for a more detailed discussion on figures of merit.

The list of "Sequoia specific" runs found on the Sequoia benchmark web site (see Section 9.4) are to be run on the Offeror's benchmark system and submitted as part its proposal response. For the selected Offeror, this same problem set will be repeated before shipment and during the acceptance testing for both the Dawn and Sequoia systems.

In addition to the above problem runs, the Offeror will also run SPhot as part of the sustained, aggregate, weighted figure of merit test described in Section 9.3. The exact problem size will necessarily be determined by the actual Dawn, and Sequoia cluster sizes, the available application memory, and the achieved sustained computation rate.

9.1.1.5 LAMMPS Marquee Benchmark (TR-1)

The open source LAMMPS code (22 Jun 2007 C++ version, http://lammps.sandia.gov) is used as a Sequoia Marquee Benchmark to test full-system performance. The EAM potential, weak scaling problem will be used, as defined on the LAMMPS web site under "benchmarks". Although the LAMMPS code can simulate a wide variety of different "particle" systems, only the classical molecular dynamics functionality will be used. As a classical molecular dynamics problem, the computing and inter-processor communication needs are well characterized and widely understood. Most communication is nearest neighbors with a small amount of data reduction done with MPI collectives. Parallelism is implemented by MPI only. The target performance demonstration for the LAMMPS benchmark is excellent scaling across all compute node cores/threads of a system with a single job. The sustained FOM requirement is found in Section 9.4.2.

The natural "raw" FOM for LAMMPS is *atoms updated per second*. See Section 9.4 for a more detailed discussion on figures of merit.

The list of "Sequoia specific" runs found on the Sequoia benchmark web site (see Section 9.3) are to be run on the Offeror's benchmark system and submitted as part its proposal response. For the selected Offeror, this same problem set will be repeated before shipment and during the acceptance testing for both the Dawn and Sequoia systems.

In addition to the above problem runs, the Offeror will also run LAMMPS as part of the sustained, aggregate, weighted figure of merit test described in Section 9.4. The exact problem size will necessarily be determined by the actual Dawn, and Sequoia cluster sizes, the available application memory, and the achieved sustained computation rate.

9.1.2 Sequoia Tier 2 Benchmarks

Tier 2 benchmarks are TR-2 requirements

9.1.2.1 Pynamic Benchmark (TR-2)

Pynamic is the Python Dynamic Benchmark and is designed to test a system's ability to handle the heavy use of dynamically linked libraries exhibited in large Python-based

applications. Pynamic is based on pyMPI, an MPI extension to the Python programming language. Pynamic adds a code generator that creates a user-specified number of Python modules and utility libraries to be linked into pyMPI. With the appropriate parameters, Pynamic can build a dummy application that closely models the footprint of an important Python-based multiphysics code at LLNL. This multiphysics code uses about five hundred dynamically link libraries (DLLs) and stresses a system's dynamic loading ability. For more information see the Sequoia Benchmarks website at https://asc.llnl.gov/sequoia/benchmarks/

A successful run of Pynamic (i.e., no errors) is sufficient verification of functionality. A time comparison between pynamic-pyMPI and pyMPI runs measures the runtime overhead of dynamic libraries.

9.1.2.2 CLOMP Benchmark (TR-2)

CLOMP is the C version of the Livermore OpenMP benchmark developed to measure OpenMP overheads and other performance impacts due to threading in order to influence future system designs. Current best-in-class implementations of OpenMP have overheads at least ten times larger than is required by many of our applications for effective use of OpenMP. For these applications to effectively use OpenMP, they require thread barrier latencies of less than 200 processor cycles and total OpenMP "parallel for" overheads of less than 500 processor cycles. The CLOMP benchmark can be used to demonstrate the need for new techniques for reducing thread overheads and to evaluate the effectiveness of these new techniques. The CLOMP benchmark is highly configurable and can also be used to evaluate the handling of other well-known threading issues such as NUMA memory layouts, cache effects, and memory contention that also can significantly affect performance. For more information see the Sequoia Benchmarks website at https://asc.llnl.gov/sequoia/benchmarks/

9.1.2.3 FTQ Benchmark (TR-2)

The FTQ benchmark measures Operating System overhead or 'noise'. This benchmark is used within the Sequoia RFP to a measure compute node light-weight kernel overhead or "noise." Sequoia SOW Section 3.2.4 gives the criteria, using the Kertosis and Skewness of the FTQ samples output from the benchmark measurements, to characterize an LWK as meeting the requirement for a "diminutive noise" LWK. For more information see the Sequoia Benchmarks website at https://asc.llnl.gov/sequoia/benchmarks/

9.1.2.4 IOR Benchmark (TR-2)

The Interleaved or Random (IOR) benchmark is used for testing the performance of parallel filesystems using various interfaces and access patterns typical in our HPC I/O environments. IOR measures the sequential read and write performance for different file sizes, I/O transaction sizes, and concurrency. IOR supports traditional POSIX I/O interfaces and parallel I/O interfaces, including MPI I/O, HDF5, and parallelNetCDF. IOR also supports different file strategies including a shared file or a single file per MPI task/processor.

9.1.2.5 Phloem MPI Benchmarks (TR-2)

The Sequoia MPI Benchmarks provide a collection of independent MPI benchmarks which are used to measure various aspects of MPI performance and scalability and usability including interconnect messaging rate, latency, aggregate link bandwidth, collectives performance, and system sensitivity to MPI task placement. The Sequoia Phloem MPI benchmarks include linktest, mpiBench, mpiGraph, Presta MPI latency, and aggregate bandwidth tests, SQMR and torustest.

9.1.2.6 Memory Subsystem Benchmarks (TR-2)

The Memory Subsystem Benchmarks provide two memory benchmarks which are used to measure the various aspects of memory performance and scalability including a variety of memory access patterns and memory performance in a multi-threaded environment. The two memory benchmarks are STRIDE and STREAMS.

The STRIDE benchmark consists of 8 separate benchmarks designed to severely test and stress the memory subsystem of a single node of a computational platform. The tests are STRID3, VECOP, CACHE, STRIDOT, and CACHEDOT. The first three benchmarks include C and Fortran language versions. All of the benchmarks utilize combinations of loops of scalar and vector operations and measure the measure the MFLOP computed as a function of the vector access patterns and length. The MFLOP rating of the various access patterns within an individual test can then be compared to provide an understanding of the performance implications.

The STREAM benchmark is a simple, synthetic benchmark designed to measure the sustainable memory bandwidth and computational rate for simple vector computational kernels written in C and Fortran. STREAM can be run on uniprocessor and multiprocessors machines. For multiprocessor machines, STREAM includes OpenMP directives and includes the necessary instructions for setting the relevant OpenMP environment variables.

9.1.3 Sequoia Tier 3 Benchmarks

The Sequoia microkernel benchmarks are all tier 3 benchmarks.

9.1.3.1 UMTMk (TR-3)

UMTMk is the microkernel for the UMT marquee benchmark code described in Section 9.1.1.1. This microkernel also serves as a threading compiler test and single CPU performance benchmark. More information can be found on the Sequoia Benchmark website at https://asc.llnl.gov/sequoia/benchmarks/

9.1.3.2 AMGMk (TR-3)

AMGMk is the microkernel for the AMG marquee benchmark code described in Section 9.1.1.2. This microkernel serves as a benchmark for sparse matrix-vector operations, single CPU performance, and OpenMP performance. More information can be found on the Sequoia Benchmark website at https://asc.llnl.gov/sequoia/benchmarks/

9.1.3.3 IRSMk (TR-3)

IRSMk is the microkernel for the IRS marquee benchmark code described in Section 9.1.1.3. This microkernel serves as a single CPU benchmark and is a SIMD compiler challenge. More information can be found on the Sequoia Benchmark website at https://asc.llnl.gov/sequoia/benchmarks/

9.1.3.4 SPhotMk (TR-3)

SPhotMk is the microkernel for the SPhot marquee benchmark code described in Section 9.1.1.4. This microkernel serves as a single CPU integer arithmetic and branching performance test. More information can be found on the Sequoia Benchmark website at https://asc.llnl.gov/sequoia/benchmarks/

9.1.3.5 CrystalMk (TR-3)

CrystalMk is a microkernel that serves as a single CPU optimization benchmark and SIMD compiler challenge. More information can be found on the Sequoia Benchmark website at https://asc.llnl.gov/sequoia/benchmarks/

9.2 Benchmark System Configuration (TR-1)

The reference benchmark system may be a scaled down version of the proposed Dawn system. The reference benchmark system may contain 1,024 cores at a minimum, with a desired core count of 8,192. The memory per node must be at least 2 GiB. The reference benchmark system used should be fully described as part of the benchmark proposal response. The reference system will require only a modest amount of I/O to run the benchmarks; NFS or equivalent. The marquee benchmarks use a single node to read input, followed by a broadcast to all other nodes. Alternate benchmarking configurations (such as mixtures of current products, future products, and simulators) may be utilized after discussion with LLNS on the benchmarking strategy and relevance of the results to the proposed systems.

The benchmark system should contain the same processors, cache, memory, nodes, interconnects, I/O interfaces, etc., that is proposed for the Dawn system. If this is not possible, benchmark results from an alternative system that meets the conditions specified in the previous paragraph may be reported. The Offeror may also provide estimated scaled performance for the Dawn configuration consistent with the benchmark system configurations as identified in the previous paragraph. All scaling arguments should be fully described by the Offeror in its proposal response and will be reviewed and evaluated by LLNS; supporting documentation may be provided. LLNS will be the sole judge of the validity of any scaled results.

9.3 Sequoia Marquee Benchmark Test Procedures (TR-1)

The following procedure has been chosen to directly demonstrate on Sequoia the successful execution of the ASC Program's highest-level objective for the Sequoia acquisition plan–a key element of ASC's multi-year platform acquisition plan as part of the Synthetic Workload (SWL) Sequoia system acceptance testing. The 24x Purple sustained, aggregate, weighted FOM demonstration consists of running six "identical problems" for each of the four IDC workload marquee benchmarks on the Sequoia system simultaneously. The 20x BG/L sustained, weighted FOM demonstration consists of running a single LAMMPS run with twenty times as many atoms as the LLNL BG/L LAMPS benchmark. This combined sustained workload run of 25 simultaneous problems from 24 IDC workload and 1 science workload benchmarks will last for four hours.

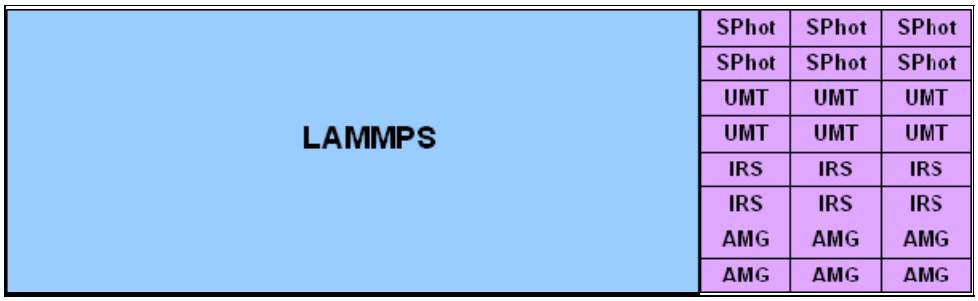


Figure 9-1: Sequoia target 25 simultaneous problems includes 24 IDC and 1 Science benchmarks.

Because the test problems defined for the benchmarks run in a different length of time, Offerors will run the above under the control of a batch scheduling system that resubmits a new problem to replace finishing problems. The official FOM of each *completed* run is saved to become part of the final report as described in Section 9.4. When the test has run for four hours, the 25 running problems are terminated without recording a figure of merit. It is planned that each "stream" of 25 problems will generate many FOM results during the total time of the test.

Because the "peak plus sustained" measurement (defined in Section 9.4.2) will be performed while running 25 simultaneous problems, the marquee benchmarks do not test the performance of individual marquee benchmarks running at full machine scale. The size of each problem has been chosen based on the largest problems that can currently be run on the ASC Purple system. In this scenario, the Sequoia system can be thought of as providing "capacity at the Purple capability level". Larger problems will also be run on Sequoia, and scaling IDC performance beyond the ASC Purple level is a key goal of the multiyear partnership envisioned between ASC/LLNL and the selected Offeror.

The benchmark runs will be made according to the following test procedures. The ASC systems will be primarily used in a high-level language environment. It is the intent of these benchmarks to measure performance of the reference system from this standpoint. Recoding of the benchmarks or portions of the benchmarks in assembly language is prohibited. The use of library routines that currently exist in an Offeror's supported set of general or scientific libraries, or will be in such a set when the Dawn and Sequoia systems are delivered, is allowed at Offeror's discretion when they do not specialize or limit the applicability of the benchmark nor violate the measurement goals of the particular benchmark. Source preprocessors, execution profile feedback optimizers, etc. are allowed as long as they are, or will be, available and supported as part of the compilation system for the Dawn and Sequoia systems. All benchmarks will be run in double precision (64b) floating point arithmetic and as 64b executables (64b virtual memory addressing). All benchmarks that use the message-passing programming paradigm will use a supported 64b virtual memory pointer, thread safe communication library that implements the MPI standard. All benchmarks that use the threads programming paradigm will use a supported communication library that implements the OpenMP standard. MPI and OpenMP functionality must be simultaneously usable by single application codes. The required run configurations for each benchmark will be described in the individual benchmark readme files. OpenMP based parallelism should be utilized to the extent possible on each node. Each node will be a set of cores sharing random access memory within the same memory address space.

Changes to accommodate unique hardware and software characteristics of a system that are consistent with the preceding paragraph will be allowed except where specifically prohibited in the constraints for each benchmark. Code modifications will be documented in the form of initial and final source files, with mandatory accompanying text describing the changes. An audit trail will be supplied to LLNS for any changes made to the benchmark codes. The audit trail will be sufficient for LLNS to determine that changes made violate neither the spirit of the benchmark nor the specific restrictions on the various benchmark codes. LLNS requires that all benchmark codes first be run as provided, without any code modifications, in each required configuration and that these baseline results be included along with any results obtained from modified code. Further discussion of the value to ASC/LLNS of specific types of modification can be found below in Section 9.4.1.

The specific problems to be run during the sustained performance test are defined as follows:

AMG – The mesh refinement factors rx, ry, and rz should all be set to 6. For the six simultaneous runs, three should be made using solver 3 and three should be used made using solver 4.

SPhot – Nruns should be set to 262,144.

UMT – Run C in the RFP benchmark problem set.

IRS – Standard problem with 25 zones per domain side. For runs using Open MP, the number of domains per MPI task may be increased to equal the number of OpenMP threads per MPI task.

LAMMPS – Standard LAMMPS EAM potential benchmark with at least 83,886,080,000 atoms and a target of 32,000 atoms per MPI task. The Offeror may not decrease the total number of atoms in the benchmark, but may adjust the number of atoms per MPI task in order to reduce the number of MPI tasks in the benchmark run in order to allow all 24 of the IDC codes to run.

9.4 Performance Measurements (TR-1)

All performance measurements for marquee benchmarks are stated as "figures of merit" (FOM). All other benchmarks use wall clock execution time as their performance measurement.

The first four marquee benchmarks print a raw figure of merit at the end of the benchmark run. For the LAMMPS benchmark, the raw figure of merit is calculated by multiplying the total number of atoms in the simulation times the total number of time steps (which is 100) and dividing by the total wall clock loop time (in seconds).

Each marquee code uses its own "raw" FOM, because each benchmark implements a different kind of physics/algorithm, and the "natural" figures of merit for each are different in type and magnitude. For example, in the SPhot and AMG benchmarks, very little floating point arithmetic is performed making FLOP/s (FLoating point OPerations per second) a poor performance metric. Performance is dominated by integer arithmetic, array indexing and branching. For SPhot, the correct balance of floating point arithmetic, integer arithmetic, array indexing and branching is captured in the sequence of instructions that update the location of a Monte Carlo particle to its next "event", be that a scattering collision, absorption, reaching a zone boundary, etc. Thus particle "track updates" per second is the natural figure of merit for meaningful comparison between computers.

For all benchmarks, the raw FOM has been chosen to also factor out the change in the difficulty of the problems as the size of the problems is increased. For example, the number of iterations to converge an answer by the IRS, UMT, and AMG benchmarks increases as system size increases. This is a characteristic of the algorithms, independent of the hardware used. So rather than use wall clock time to completion as the FOM, these three benchmarks use solution updates per second,

defined as system size (e.g., the number of fundamental state variables) times the number of iterations performed, divided by the wall clock time (in seconds) as their raw figures of merit. Each benchmark's raw FOM is then multiplied by a separate weight (see Table 9-2 below) that both balances the importance of the benchmarks with respect to each other, and with respect to the peak FLOP/s (floating point *operations* per second) of the system.

Benchmark	SPhot	UMT	AMG	IRS	LAMMPS
Weights	57,512	12,240	269,200	203,200	37,840

Table 9-2: Sequoia Marquee Benchmarks FOM Weights

The *weighted* FOM performance of the four IDC workload marquee benchmarks on the ASC Purple system for 1,024 (1 Ki) to 8,192 (8 Ki) processors is shown in Figure 9-2.

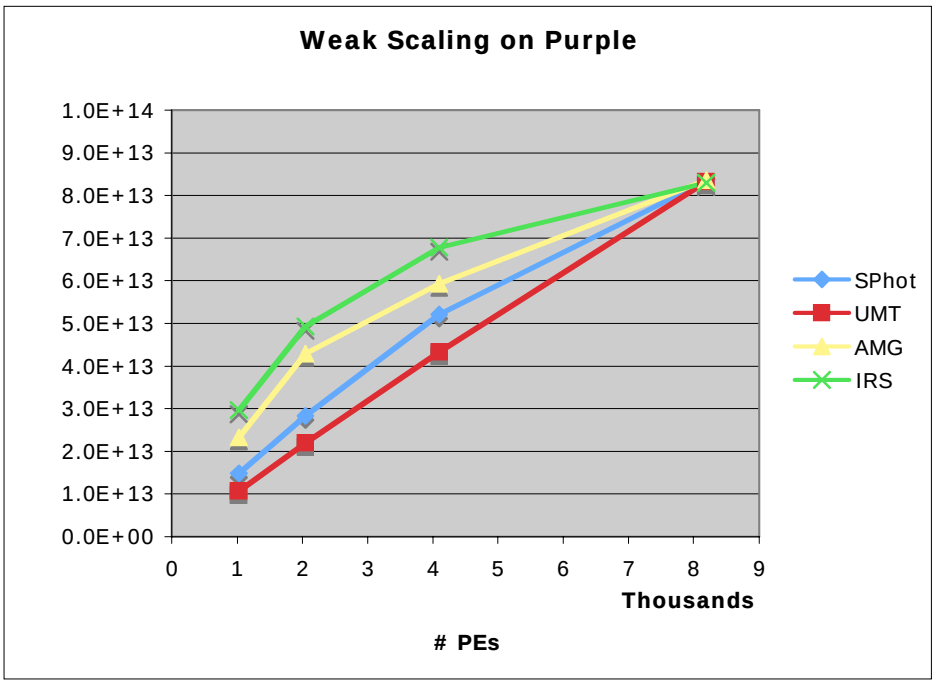


Figure 9-2: Sequoia Marquee benchmark scaling to 8,192 MPI tasks on Purple.

The Offeror must commit to a specific sustained, aggregate, weighted figure of merit while running under conditions described in Section 9.4.2.

The *raw* FOM performance of the LAMMPS science workload marquee benchmark on the ASC BG/L system up to 128 Ki processors is shown in Figure 9-3. (NOTE: For processor counts up to and including 64 Ki processors, LAMMPS was run using only one of the two processors on a node. For 128 Ki processors, both processors was used on the same 64 Ki nodes. The 64 Ki to 128 Ki speedup was 1.84.)

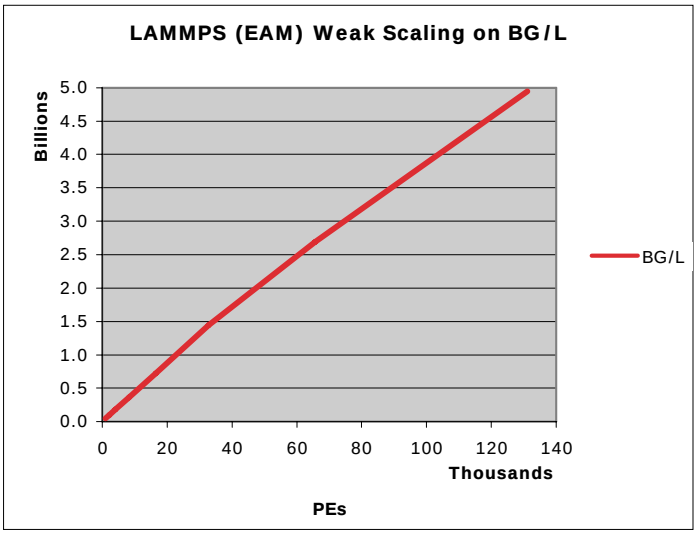


Figure 9-3: LAMMPS scaling up to 131,072 (128Ki) MPI tasks on BG/L.

9.4.1 Modifications

The source code and compile scripts downloaded from the Sequoia benchmark web site may be modified as necessary to get the benchmarks to compile and run on the Offeror's system. Once this is accomplished, a full set of benchmark runs must be reported with this "as is" source code.

Beyond this, the benchmarks can be optimized as desired by the Offeror. The highest value optimizations are those obtained from standard compiler flags, and other compiler flag hints. Next in value are performance improvements from pragma-style guidance in C, C++, and Fortran source files. Changes in the OpenMP and MPI implementation are allowed, as they are likely to benefit performance of LLNS's benchmarks on many platforms. Wholesale algorithm changes, or even manual rewriting of loops to become strongly architecture specific are of less value, because the ASC program's large installed code base makes extensive code rewrites prohibitively expensive, both in manpower and schedule.

Offeror will continue its efforts to improve the efficiency and scalability of the benchmarks. Offeror's goal in these improvement efforts is to emphasize higher level optimizations as well as compiler optimization technology improvements while maintaining *readable and maintainable* code.

If Python is not available on all systems that are used by Offeror to run the benchmarks for the RFP, LLNS will help the Offeror create a non-Python version to use when running the RFP benchmarks. However, in the event of subcontract award, the selected Offeror must demonstrate a Python-based version of the UMT benchmarks on the Dawn system prior to shipment.

9.4.2 Sequoia Execution Requirements

The conditions for running the sustained aggregate, weighted figure of merit for the final Sequoia delivery are as follows. For the IDC workload, six separate (but identical) problems will be run simultaneously using each of the four marquee benchmark codes, for a total of 24 simultaneously running IDC problems. (See Section 9.3 for the specific problems to be run.)

Each problem is sized to be the largest possible problem that can be run on the ASC Purple machine for that IDC benchmark (8,192 MPI tasks) and 20x BG/L run with 131,072 MPI tasks and 32,000 atoms/ MPI task for LAMMPS. Simultaneous with these runs, a single LAMMPS run with at least 83,886,080,000 atoms = 20 * 131,072 MPI tasks * 32,000 atoms/MPI task will be run. Thus, the aggregate throughput of the Sequoia system during the sustained performance test must be 24x Purple for IDC codes *plus* 20x BG/L for the science workload. The "raw" figures of merit printed out by each of the four marquee IDC benchmarks on 8,192 Purple processors (MPI-only), and LAMMPs on 131,072 CPUs on BG/L are shown in Table 9-3.

Benchmark	SPhot	UMT	AMG	IRS	LAMMPS
"Raw" FOM	11.59e+9	54.39e+9	2.484e+9	3.27e+9	5.28E+9

Table 9-3: Raw FOM for Sequoia marquee benchmarks on their reference system.

Using the weighting factors in Table 9-2 and the raw figures of merit for of each of the five marquee benchmark codes on Purple in Table 9-3 above, the final sustained figure of merit may be no less than 20.0e+15.

If 8,192 MPI tasks (MPI only) in the IDC benchmarks are insufficient to achieve the required aggregate, weighted FOM, more cores and threads may be used via thread parallelism.

9.4.2.1 Sequoia14 Execution Requirements

If the Sequoia14 System Performance Mandatory Option (Section 2.12.3) is exercised, then the procedure for measuring the aggregate sustained performance as described in Section 9.4.2 will be adjusted as following. The LAMMPS problem size will be decrease to a total of 58,720,256,000 atoms = 14 * 131,072 MPI tasks * 32,000 atoms/MPI task. The number of simultaneously running jobs for each IDC code will be decreased from six to four, making a total of 16 simultaneously running IDC jobs which are also simultaneous with the LAMMPS job. All multipliers for raw figures of merit will remain the same.

End of Section 9.0

10.0 Appendix A Glossary

10.1 Hardware

b	bit. A single, indivisible binary unit of electronic information.
B	Byte. A collection of eight (8) bits.
32b floating-point arithmetic	Executable binaries (user applications) with 32b (4B) floating-point number representation and arithmetic. Note that this is independent of the number of bytes (4 our 8) utilized for memory reference addressing.
32b virtual memory addressing	All virtual memory addresses in a user application are 32b (4B) integers. Note that this is independent of the type of floating-point number representation and arithmetic.
64b floating-point arithmetic	Executable binaries (user applications) with 64b (8B) floating-point number representation and arithmetic. Note that this is independent of the number of bytes (4 our 8) utilized for memory reference addressing.
64b virtual memory addressing	All virtual memory addresses in a user application are 64b (8B) integers. Note that this is independent of the type of floating-point number representation and arithmetic. Note that all user applications should be compiled, loaded with Offeror supplied libraries and executed with 64b virtual memory addressing by default.
CE	On-site hardware customer engineer performing hardware maintenance with DOE Q-clearance.
CN	System compute nodes. Compute Nodes (CN) are nodes in the system that user MPI jobs execute on.
Core	Portion of processor that contains execution units (e.g., instruction dispatch, integer, branch, load/store, floating-point, etc), registers and typically at least L1 data and instruction caches. Typical cores implement multiple hardware threads of execution and interface with other cores in a processor through the memory hierarchy and possibly other specialized synchronization and interrupt hardware.
FLIN	Computing thread Floating Point INstruction. Note that FLIN and FLOP are quite different as multiple FLOPS can be accomplished with a single FLIN (e.g., SSE2 or AltiVec)
FLINS	Plural of FLIN.
FLIN/s	Floating Point INstruction retired per second.
FLOP	Floating Point OPeration.
FLOPS	Plural of FLOP.

FLOP/s	Floating Point OPeration per second.
FMA	Fused Multiply Add (FMA) is a single 64b or 32b floating-point instruction that operates on three inputs by multiplying one pair of the inputs together and adding the third input to the multiply result and produces one 64b or 32b floating-point output. Typically FMA instructions can be pipelined and have a completion rate of one per core per clock.
FPE	Floating Point Exception.
GB	gigaByte. gigaByte is a billion base 10 bytes. This is typically used in every context except for Random Access Memory size and is 10^9 (or 1,000,000,000) bytes.
GiB	gibiByte. gibiByte is a billion base 2 bytes. This is typically used in terms of Random Access Memory and is 2^{30} (or 1,073,741,824) bytes. For a complete description of SI units for prefixing binary multiples see URL: http://physics.nist.gov/cuu/Units/binary.html
GFLOP/s or GOP/s	gigaFLOP/s. Billion (10^9 = 1,000,000,000) 64-bit floating point operations per second.
IBA	InfiniBand™ Architecture (IBA) http://www.infinibandta.org/specs
ION	System IO nodes. IO Nodes are nodes in the system that support IO functions for the CN.
ISA	Instruction Set Architecture. The architectural definition of the processor and the instruction set executed on the processor including cache coherency models and architectural registers and other processor resources.
LN	System Login Nodes. Login Nodes are nodes where users can login in and interact with the system.
MB	megaByte. megaByte is a million base 10 bytes. This is typically used in every context except for Random Access Memory size and is 10^6 (or 1,000,000) bytes.
MiB	mebiByte. mebiByte is a million base 2 bytes. This is typically used in terms of Random Access Memory and is 2^{20} (or 1,048,576) bytes. For a complete description of SI units for prefixing binary multiples see URL: http://physics.nist.gov/cuu/Units/binary.html
MFLOP/s or MOP/s	megaFLOP/s. Million (10^6 = 1,000,000) 64-bit floating point operations per second.

MTBAF	Mean Time Between (Hardware) Application Failure. A measurement of the expected hardware reliability of the system or component as seen from an application perspective. The MTBAF figure can be developed as the result of intensive testing, based on actual product experience, or predicted by analyzing known factors. Hardware failures of or transient errors in redundant components such as correctable single bit memory errors or the failure of an N+1 redundant power supply and do not cause an application to abnormally terminate do not count against this statistic. Thus, MTBAF ≥ MTBF.
MTBF	Mean Time Between (Hardware) Failure. A measurement of the expected hardware reliability of the system or component. The MTBF figure can be developed as the result of intensive testing, based on actual product experience, or predicted by analyzing known factors. See URL: http://www.t-cubed.com/faq_mtbf.htm
NCN	Number of CN in the proposed system.
NCORE	The number of cores in the CN allocatable to and directly programmable by user MPI tasks. If the peak petaFLOP/s system characteristic requires multiple threads per core to be issuing floating-point instructions, then NCORE is the number of allocatable cores times that number of threads.
Node	Shared memory Multi-Processor. A set of cores sharing random access memory within the same memory address space. The cores are connected via a high speed, low latency mechanism to the set of hierarchical memory components. The memory hierarchy consists of at least processor registers, cache and memory. The cache will also be hierarchical. If there are multiple caches, they will be kept coherent automatically by the hardware. The access mechanism to every memory element will be the same from every processor. More specifically, all memory operations are done with load/store instructions issued by the core to move data to/from registers from/to the memory. From the SRM perspective, is the indivisible resource that can be allocated to a job consisting of one or more cores and their associated memory.
Non-Volatile	Non-volatile memory, nonvolatile memory, NVM or non-volatile storage, is computer memory that can retain the stored information even when not powered. Examples of non-volatile memory include read-only memory, flash memory, most types of magnetic computer storage devices (e.g. hard disks, floppy disk drives, and magnetic tape), optical disc drives, and early computer storage methods such as paper tape and punch cards. See http://en.wikipedia.org/wiki/Non-volatile
NUMA	Non-Uniform Memory Access architecture. The distance in processor clocks between processor registers depends on where in main memory the address points to. That is, a load/store operation latency for some memory locations is larger than that for others.
OP	Computing thread operation or instruction.
OPS	Plural of OP.

OP/s	Computing thread operation or instruction retired per second.
PB	petaByte. petaByte is a quadrillion base 10 bytes. This is typically used in every context except for Random Access Memory size and is 10^{15} (or 1,000,000,000,000) bytes.
PiB	pebiByte. pebiByte is a quadrillion base 2 bytes. This is typically used in terms of Random Access Memory and is 2^{50} (or 1,125,899,906,842,620) bytes. For a complete description of SI units for prefixing binary multiples see URL: http://physics.nist.gov/cuu/Units/binary.html
Peak FLOP/s (FLIN/s) Rate	The maximum number of 64-bit floating point instructions (add, subtract, multiply or divide) or operations (instructions) per second that could conceivably be retired by the system. For muti-threaded, multi-core processors, the peak rate is typically calculated as the maximum number of floating point operations (instructions) that each thread in a core can retire per clock times the clock rate times the number of threads in a core times the number of cores in a processor.
Peta-Scale	The environment required to fully support production-level, realized petaFLOP/s performance. This environment includes a robust and balanced processor, memory, mass storage, I/O, and communications subsystems; robust code development environment, tools and operating systems; and an integrated cluster wide systems management and full system reliability and availability.
Processor	The computer ASIC die and package. A VLSI ASIC chip constituting with the computational cores (integer, floating point, and branch units) and threads (stack pointer, instruction pointer, copy of ISA defined hardware registers, but shares execution units on a core with other threads), registers and memory interface (virtual memory translation, TLB and bus controller).
Scalable	A system attribute that increases in performance or size as some function of the peak rating of the system. The scaling regime of interest is at least within the range of 1 petaFLOP/s to20.0 petaFLOP/s peak rate.
SECDED	Single Error Correction Double Error Detection. Storage and data transfer protection mechanism that can detect parity errors (single bit errors) and detect storage or data transfer errors with multiple bits in them.
SIMD	Single Instruction, Multiple Data (SIMD) instructions are processor instructions that operate on more than one set of input 64b or 32b floating-point values and produce more than one 64b or 32b floating-point value. Fused Multiply-Add (FMA) instructions are not SIMD. Examples of this are x86-64 SSE2 and Power VMX instructions.
SN	System Service Nodes. Service Nodes are nodes in the system that system administrators use to manage the system including RAS, OS install, etc.

Thread	Hardware thread of execution is a hardware context within a core of a processor that executes instructions. Multiple threads within a core share the core's computational units, but have separate instruction pointer, stack and heap pointers, and ISA defined hardware registers. Hardware threads are typically exposed to through the operating system as independently schedulable sequences of instructions. A hardware thread executes a software thread within a Linux (or other) OS process.
TB	TeraByte. TeraByte is a trillion base 10 bytes. This is typically used in every context except for Random Access Memory size and is 10^{12} (or 1,000,000,000,000) bytes.
TiB	TebiByte. TebiByte is a trillion bytes base 2 bytes. This is typically used in terms of Random Access Memory and is 2^{40} (or 1,099,511,627,776) bytes. For a complete description of SI units for prefixing binary multiples see URL: http://physics.nist.gov/cuu/Units/binary.html
TLB	Translation Look-aside Buffer (TLB) is a set of content addressable hardware registers on the processor that allows fast translation of virtual memory addresses into real memory addresses for virtual addresses that have an active TLB enetry.
TFLOP/s	teraFLOP/s. Trillion (10^{12} = 1,000,000,000,000) 64-bit floating point operations per second.
UMA	Uniform Memory Access architecture. The distance in core clocks between core registers and every element of node memory is the same. That is, load/store operations that are serviced by the node memory have the same latency to/from every core, no matter where the target physical location is in the node memory assuming no contention.

10.2 Software

32b executable	Executable binaries (user applications) with 32b (4B) virtual memory addressing. Note that this is independent of the number of bytes (4 our 8) utilized for floating-point number representation and arithmetic.
64b executable	Executable binaries (user applications) with 64b (8B) virtual memory addressing. Note that this is independent of the number of bytes (4 our 8) utilized for floating-point number representation and arithmetic. Note that all user applications should be compiled, loaded with Offeror supplied libraries and executed with 64b virtual memory addressing by default.
API (Application Programming Interface)	Syntax and semantics for invoking services from within an executing application. All APIs will be available to both Fortran and C programs, although implementation issues (such as whether the Fortran routines are simply wrappers for calling C routines) are up to the supplier.

Baseline Languages	The Baseline Languages are Fortran03, C, C++ and Python.
BIOS	Basic Input-Output System (BIOS) is low level (typically assembly language) code usually held in flash memory on the node that tests and functions the hardware upon power-up or reset or reboot and loads the operating system.
BOS	Base Operating System (BOS). Linux (LSB 3.1) compliant Operating System run on the ION, SN and LN.
CDTI	The hierarchal Code Development Tools Infrastructure (CDTI) components are distributed throughout the Sequoia system. Individual code development tool "front-end" components that interact with the user execute on the LN (although the display may be remoted via an X-Window). Code development tool communications mechanisms interface the tool "front-ends" running on the LN with the "back-end" manipulating the user application running on the CN through a single level fan-out hierarchy running on the ION.
Current standard	Term applied when an API is not "frozen" on a particular version of a standard, but will be upgraded automatically by Offeror as new specifications are released (e.g., "MPI version 2.0" refers to the standard in effect at the time of writing this document, while "current version of MPI" refers to further versions that take effect during the lifetime of this contract.
Fully supported (as applied to system software and tools)	A product-quality implementation, documented and maintained by the HPC machine supplier or an affiliated software supplier.
Job	An allocation of resources to a user for a specified period of time. The user should be given control over which resources can be allocated to a job.
Job step	A set of (possibly parallel) tasks launched concurrently and able to communicate using a high-speed interconnect. A job typically executes one or more job steps sequentially and/or concurrently. A job step may use any or all of the resources allocated to the job. The user should be given control over which resources are used by a job step.
LWK	Light-Weight Kernel providing operating system functions to user applications running on CN.
Moab	Cluster Resources batch scheduler (www.cluserresources.com)
OFED	Open Fabrics Enterprise Edition (OFED) (http://www.openfabrics.org/)
OS	Operating System

Process	A process is an executing (i.e., running) instance of a program. Processes are also frequently referred to as tasks or MPI tasks. Processes are dynamic entities scheduled and controlled by the Operating System that are constantly changing as their machine code instructions are executed by the CPU. Each process consists of (1) system resources that are allocated to it, (2) a section of memory, (3) security attributes (such as its owner and its set of permissions) and (4) the processor state. The processor state includes the contents of its registers and physical memory addresses. Registers are a very small amount of very fast memory that is built into a processor in order to speed up its operations by providing quick access to commonly used values. Processes may have one or more operating system threads associated with it. See: http://www.linfo.org/process.html
Published (as applied to APIs):	Where an API is not required to be consistent across platforms, the capability lists it as "published," referring to the fact that it will be documented and supported, although it will be Offeror- or even platform-specific.
RPCTI	Remote process control code development tools interface that allows code development tools to interface from the LN through the LUOS on the ION to the LWK on the CN and operate on user processes and threads on the CN.
Single-point control (as applied to tool interfaces)	Refers to the ability to control or acquire information on all processes/PEs using a single command or operation.
SLURM	Simple Linux Resource Management (SLURM) is an Open Source cluster resource management system and parallel application job launch facility. (www.llnl.gov/linux/slurm)
Standard (as applied to APIs)	Where an API is required to be consistent across platforms, the reference standard is named as part of the capability. The implementation will include all routines defined by that standard (even if some simply result in no-ops on a given platform).
Task	A process launched as a job step component, typically an MPI task.
Thread	A portion of a program that can run independently of and concurrently with other portions of the program. Operating system threads are schedule by the OS. User space threads are scheduled by the runtime system in user space and don't require intervention of the OS. Threads within a process share the same memory address space. However, each thread has its own set of architectural registers, stack, heap, and instruction counter or pointer and executes instructions independently of other threads.
XXX-compatible (as applied to system software and tool definitions)	Requires that a capability be compatible, at the interface level, with the referenced standard, although the lower-level implementation details will differ substantially (e.g., "NFSv4-compatible" means that the distributed file system will be capable of handling standard NFSv4 requests, but need not conform to NFSv4 implementation specifics).

End of Section 10.0