Petascale Applications Preparations

February 10, 2010

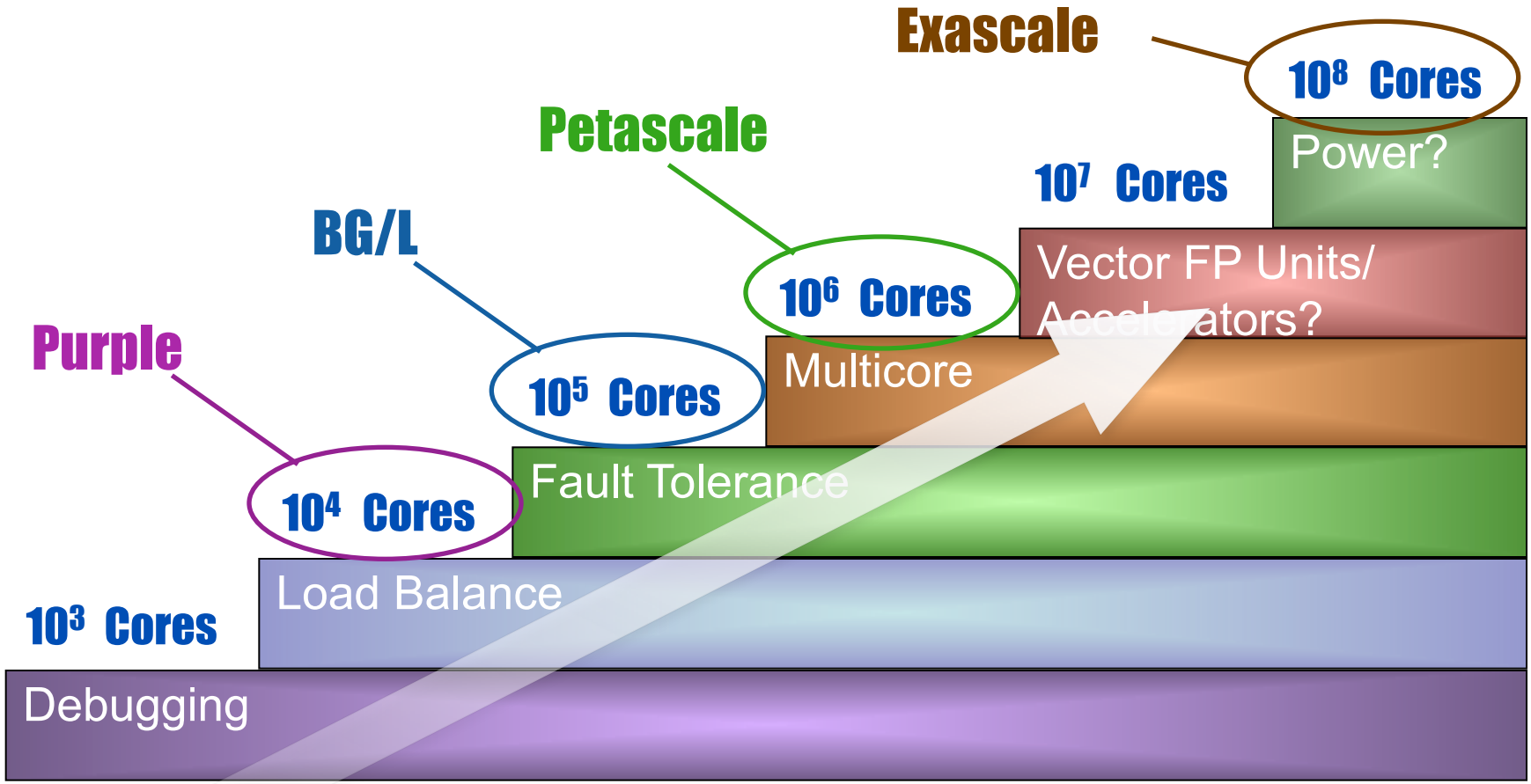
Scott Futral and Bronis R. de Supinski
Co-Leads, ASC Application Development Environment and Performance Team (ADEPT)
Lawrence Livermore National Laboratory

Lawrence Livermore National Laboratory, P. O. Box 808, Livermore, CA 94551

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

LLNL-PRES-423302

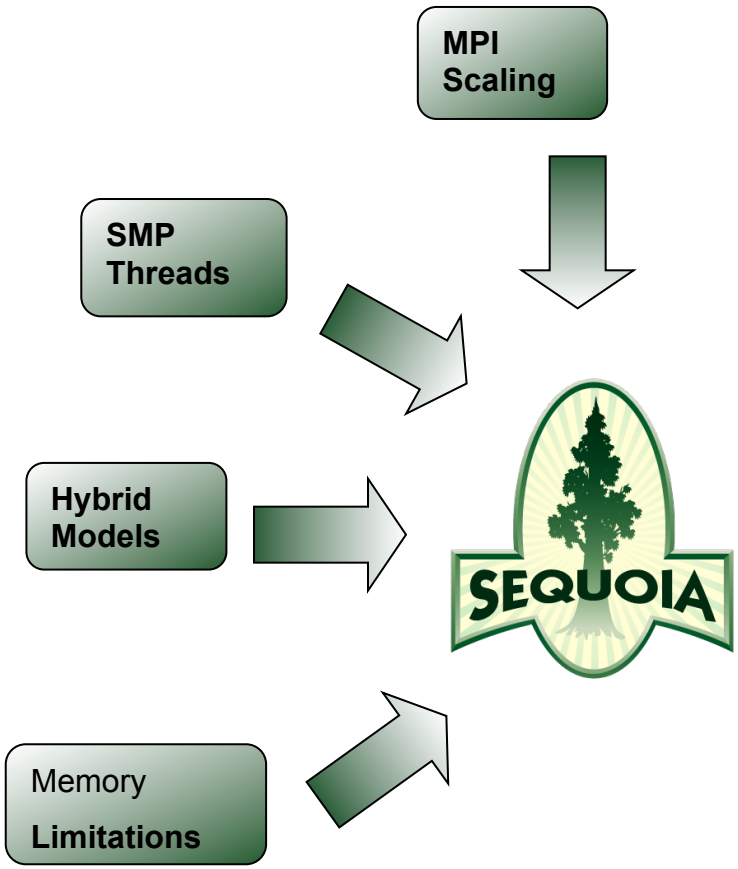
Our research targets key petascale barriers to position ourselves to attack exascale barriers



ExaCT creates essential capabilities for exascale core counts

Our research barriers reflect increasing application programming requirements and challenges

- Availability of 1.6M cores pushes all-MPI codes to extreme concurrency
- Availability of many threads on many SMP cores encourages low-level parallelism for higher performance
- Mixed MPI/SMP programming environment and possibility of heterogeneous compute distribution brings load imbalance to the fore
- Memory costs dominate, encouraging innovative strategies to minimize memory and bandwidth bottlenecks



The SAP Project directly engages a broad set of application teams

The Scalable Application Preparation (SAP) Project supports code teams' preparations for Sequoia

Mission:

Pro-actively assist applications code developers in preparing for Sequoia

Goal:

Code teams have the information and support they need to succeed in using Sequoia

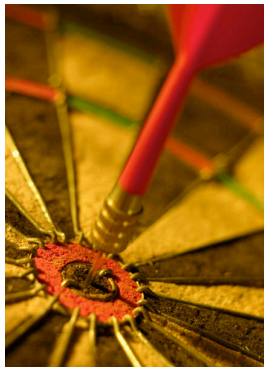
Methods:

Explore, Communicate (with teams and vendor), Evaluate, Document, Train, Support

The SAP Project activities will ensure Sequoia success

- Ongoing ANL/IBM/LLNL BlueGene collaboration
- Team provides expertise in compilers, debuggers, performance tools
- Providing access to IBM experts, including an on-site IBM applications analyst
- Leverage ExACT, ISCR Applications Testbed

- Collaborating closely with application teams
- Engaging third-party vendors, university research partners, and the open source community
- Addressing support issues via LC Hotline, User Training, and Documentation



Simulators are a key component in our plans to prepare for new Sequoia features

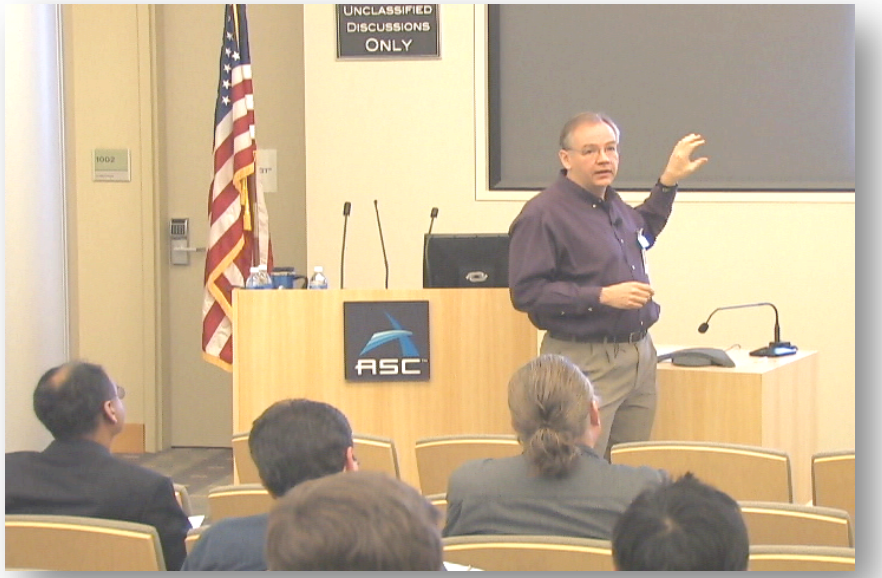
- Transactional memory and speculative execution offer potential performance benefits
  - Early access is useful for exploration and education
- IBM's Software Transactional Memory (STM) Compiler (stmxlc) is now available on LC uP and Purple systems
- Allows execution of user binaries and early access to runtime software (MPI & OpenMP)
- Remote access to a hardware device simulator for the Sequoia processor is now available
- IBM on-site analyst John Romo is the simulator POC and will assist code teams with access and usage

Early investigations of STM applied to unstructured mesh algorithms were presented at ScicomP'09

- For select algorithms, Transactional Memory promises thread-safety, easier programming, and performance all at the same time
- IBM has STM available now, ready to be tried out
- STMXLC has a useful statistics/diagnostics tool
- Demonstrated two algorithms that may benefit
- Pay-off is likely algorithm- and problem-dependent
- In random mode, STM over-predicts conflicts

  Conjecture: multiple STM retries in same transaction
- Run-time is already competitive with OpenMP "critical"

SAP training activities are under way…



Alan Gara, IBM Fellow and Chief Architect for Blue Gene, presents Sequoia overview

- 'From Here to Sequoia, a Code Developer Perspective' presented on 3/23/09
- 90 participants from 6 Labs
- Introduction to Sequoia and motivator for code developers
- 8 talks: system architecture to Dawn account requests

SAP user events include training on tools, Dawn

- TAU Workshop
  - Focused on performance experiments for multi-core architectures
- Continuing IBM presentations
  - HPCToolkit presentation by I-Hsin Chung
  - C++0X/XL Compiler presentation bby Michael Wong
- Participated in Sandia Applications Workshop
- Tri-Lab tools workshops are in the planning stages
- Dawn User Meetings
  - Initial meeting addressed I/O issues
  - User-driven agenda, 3rd Thursdays at 3PM
  - Goal: make it available through Access Grid
- Dawn BG/P training by Blaise Barney at Sandia/LANL

SAP meetings with application teams continue

- Discussions focus on:
  - Understanding the code teams current architecture and status for scalability
  - Longer-term issues for scalability for code teams and LC
  - Plans for addressing multi-core systems in particular
- Meetings have been held with representatives from ALE3D, ARES, KULL, ISCR, Charon, Sierra, Trilinos, Alegra.
- Conclusions: teams are exploring OpenMP and threading models, but are not confident of success. I/O performance also concerns many teams.

"Dawn speeds a man on his journey, and speeds him too in his work" ...Hesiod (~700 B.C.E)

Dawn Specifications

- IBM BG/P architecture
- 36,864 compute nodes (500TF)
- 147,456 PPC 450 cores
- 4GB memory per node (147.5TB)
- 128-to-1 compute to I/O node ratio
- 288 10GE links to file system

Dawn Installation

- January - first18 rack delivery Phase I
- Feb 27th - final 18 rack delivery Phase II
- March 5th - 36 rack integration complete
- End of March - ***Machine Acceptance***
- Beginning of April - ***Science Runs***
- *Dec 1st 2009 – SCF G. A. for TRI-LABS*

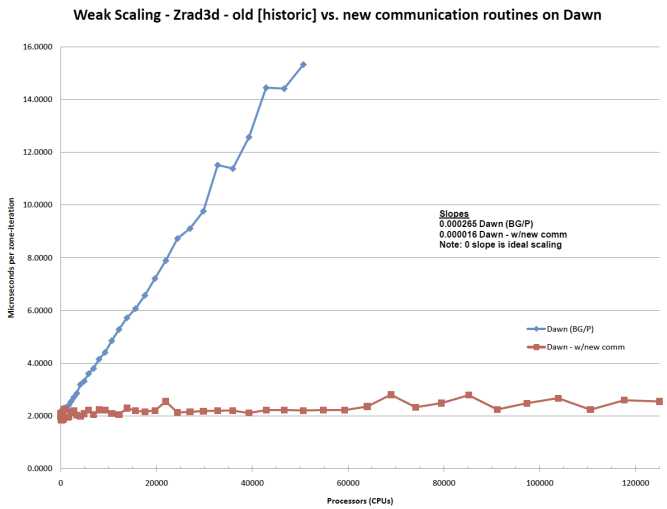
ARES scaling tests for communication and threads demonstrate progress on Dawn

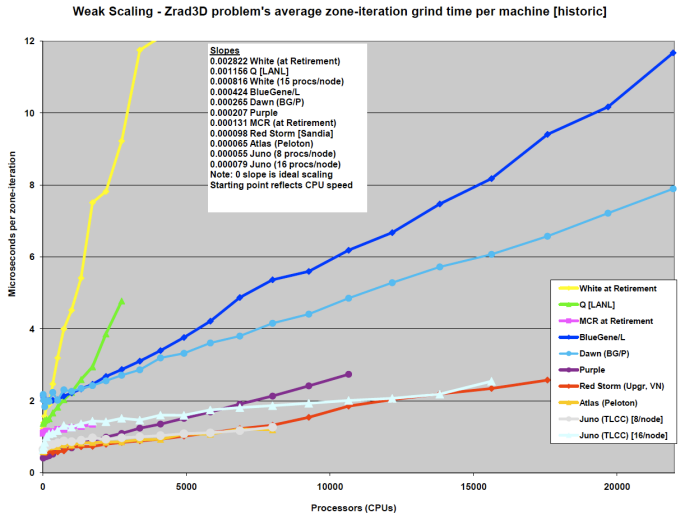
Scaling studies show Zrad3D (ARES code test problem) scaling similar to Purple, and scales well even at large processor counts. ARES team re-wrote the communications infrastructure to reduce memory utilization at large scale, and dramatically improve scaling performance.

- Scales to the full machine (147,456 processors)
- Demonstrates that threading is viable on Dawn.

Threads tests demonstrates progress toward running on Sequoia, where threads may be required to get the desired performance out of the architecture.

-Michael Collette, LLNL





LC and Computations stand ready to address key challenges Sequoia poses for applications teams

- BG/L, and now Dawn, inform Sequoia scalability
- OpenMP & Posix threads experience on Linux/AIX
- Integrated codes currently run at Purple capability scale
- Dawn adds early experience for code development:
  - SMP parallelism
  - Python and Dynamic Libraries
  - Larger memory per core than BG/L
  - ISCR Applications Testbed
  - Production use by ASC codes
  - **Dawn is a Tri-Lab ASC resource**

ExaCT will further address petascale barriers on 3 key LLNL applications

Application Drivers

pF3D: NIF Laser Plasma Interaction

ARES: B Division Integrated Code

KULL: A/X Dvision Integrated Code

Solutions here will ensure success across full range of LLNL applications

Research & Development Targets

Scalable Multicore Algorithms

Application Level Fault Tolerance

Asynchronous Load Analyzer and Adaptor

Debugging Methodologies

Experimental Platforms

Dawn

SEQUOIA

Our strategy mitigates risk to assure immediate impact on application drivers and longer term success

Current capabilities

MPI large grain parallelism
Basic checkpoint/restart
Ill-defined load imbalances
Debugging ≤ 4096 cores

Shorter Term Payoff

Cache oblivious data layouts

Checkpoint compression

Load balance analysis

Behavioral and performance equivalence classes

Terascale capabilities

Multicore-adapted algorithms
Faster checkpoint/restart
Understood load imbalances
Targeted debugging

High Long Term Payoff

Algorithms with fine grain parallelism

Recomputation from local, partial checkpoints

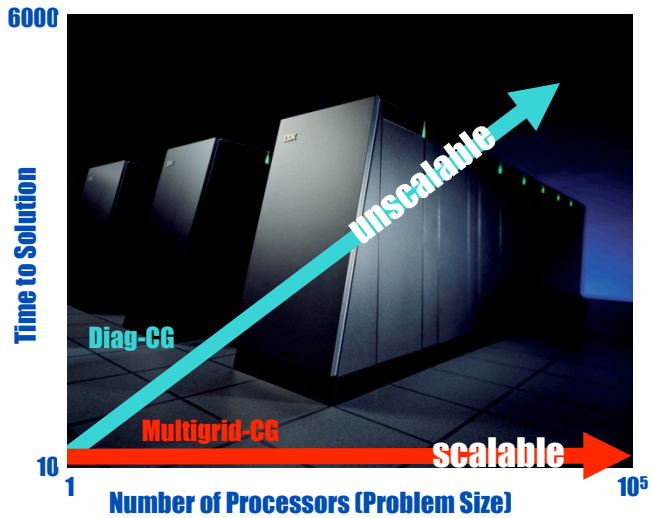
Automated asynchronous load rebalancing

Root cause analysis & algorithmic debugging

Petascale capable & Exascale prepared

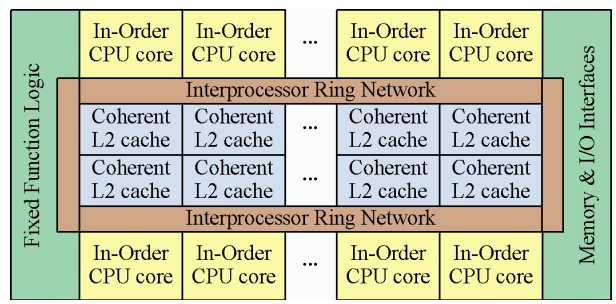
Multicore-aware algorithms
Application-level fault tolerance
Well-balanced application load
Automated error analysis

Exascale computing will require the development of novel mathematically-scalable multicore algorithms

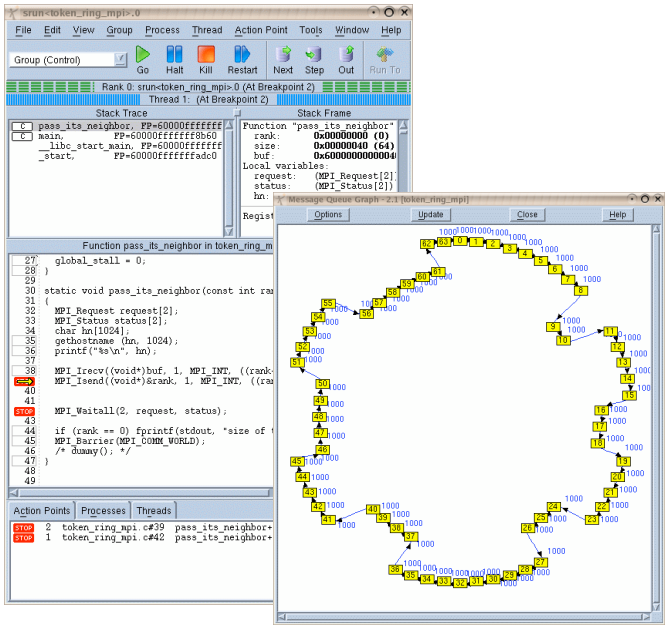


- Multigrid solvers are essential components of LLNL simulation
  - State-of-the-art is *hypre*, with proven scalabilty to BG/L class machines
- Current solvers will break down on tomorrow's exascale machines
  - Enormous core counts and fine-grain parallelism will degrade convergence

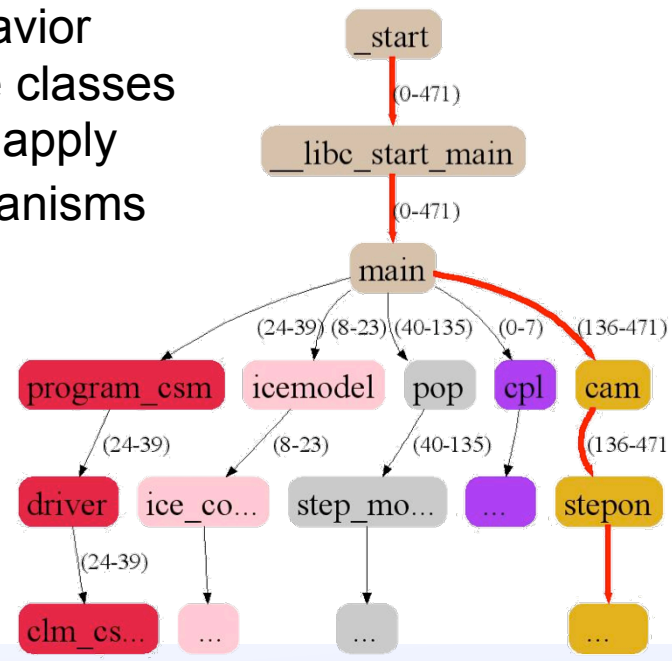
- New mathematics R&D (convergence)
  - Effective smoothers that are also highly parallel (e.g., CG, block methods)
  - Effective interpolation for highly parallel coarsening schemes
- Techniques for multicore (performance/FT)
  - Design and automate new data layouts
  - Design and develop fault tolerant solvers



We are designing and developing methodologies that will transform petascale (and exascale) debugging

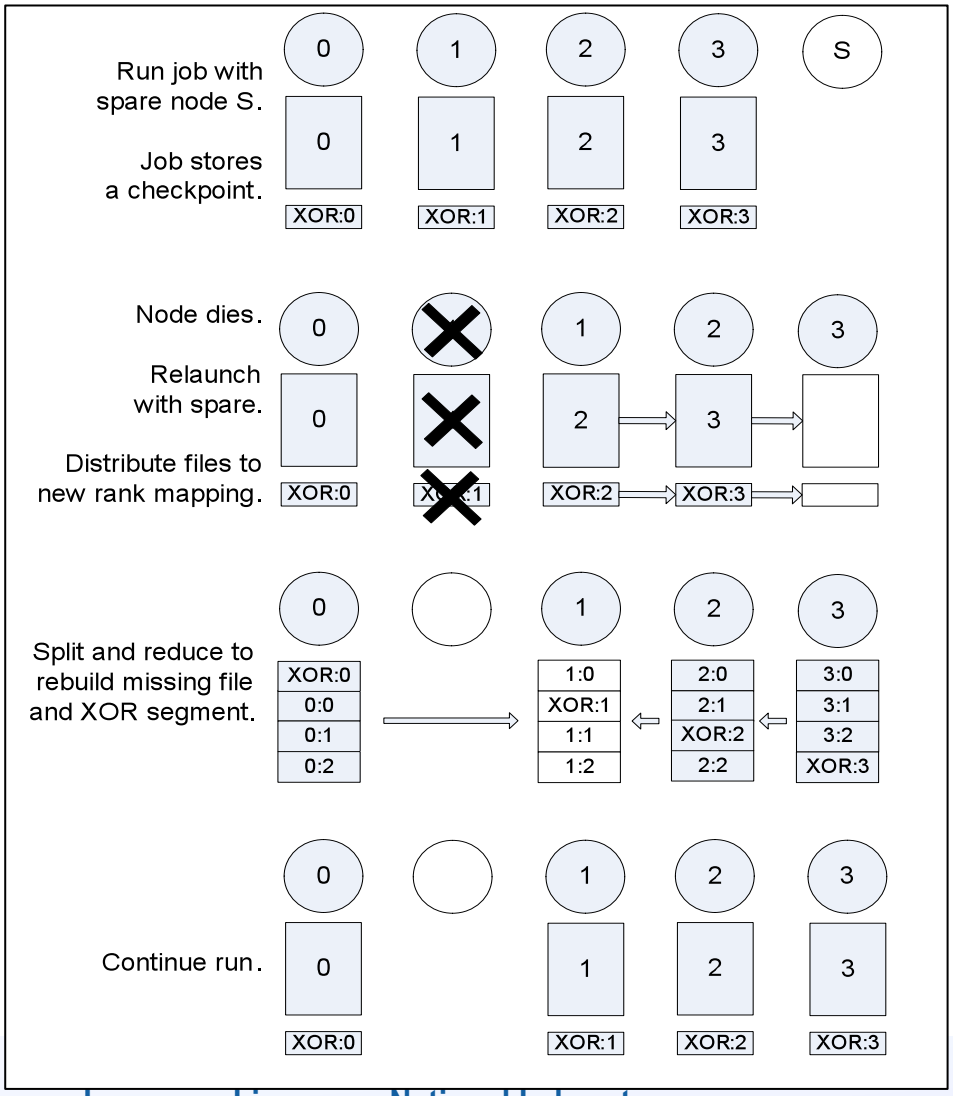


- Existing tools too slow even at thousands of cores
- We propose two new methodologies
- Lightweight tools
  - Complement existing techniques
  - Identify behavior equivalence classes on which to apply those mechanisms



- Transformational automated techniques
  - Statistical methods to locate errors quickly
  - Apply machine learning techniques to data from correct and erroneous executions
  - Analyze program semantics to locate source of errors automatically
- Working with broader community

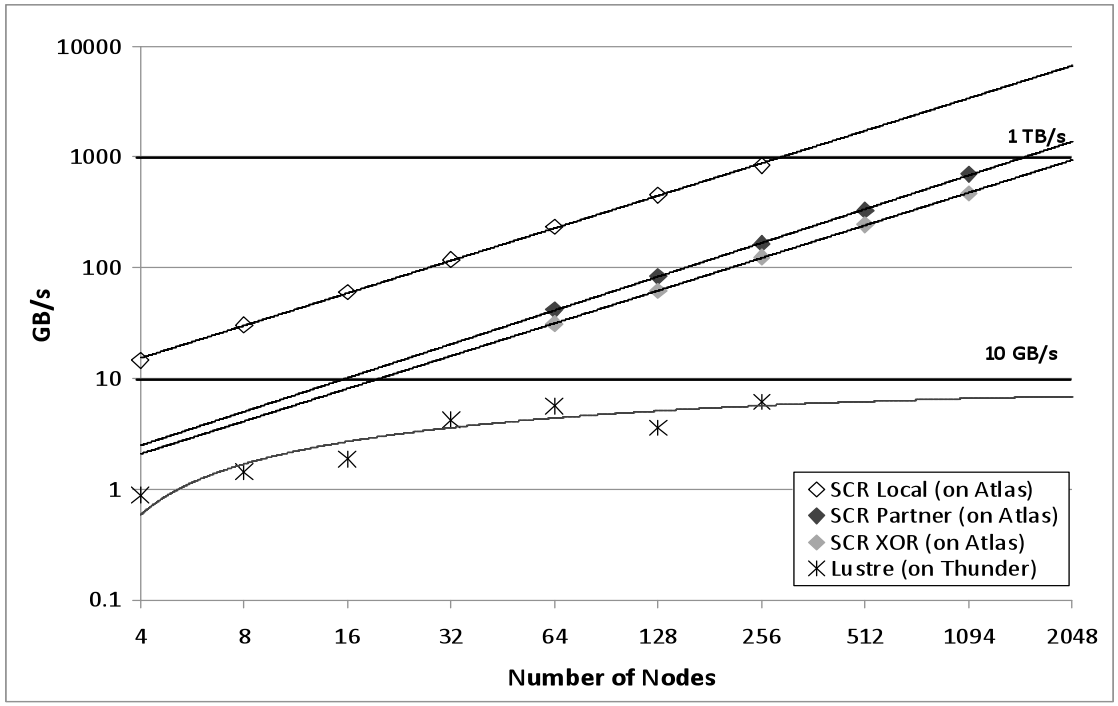
The Scalable Checkpoint / Restart (SCR) Library exploits checkpoint requirements to reduce cost



- Applications only need their most recent checkpoint
- Typical failures only disable a small piece of the system
- Two-level checkpointing technique
  - Cache checkpoint files on compute nodes
  - Write to the parallel file system as needed
  - Use RAID-like strategy to withstand node failures

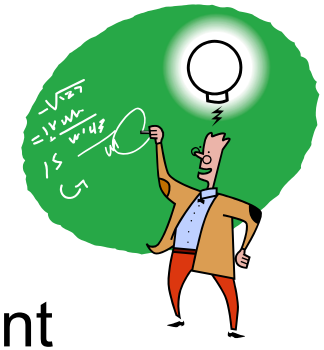
SCR increases productivity of large-scale calculations

- Large-scale pf3d runs on Atlas for NIF target simulation
  - Reduced time to checkpoint from 20 minutes to 10 seconds
  - 2X checkpoint frequency
  - Reduce checkpoints written to Lustre by a factor of 7
  - Checkpoint cost reduced from 25% to 5.3% of runtime
  - Increased mean time before failure from a few hours to tens of hours

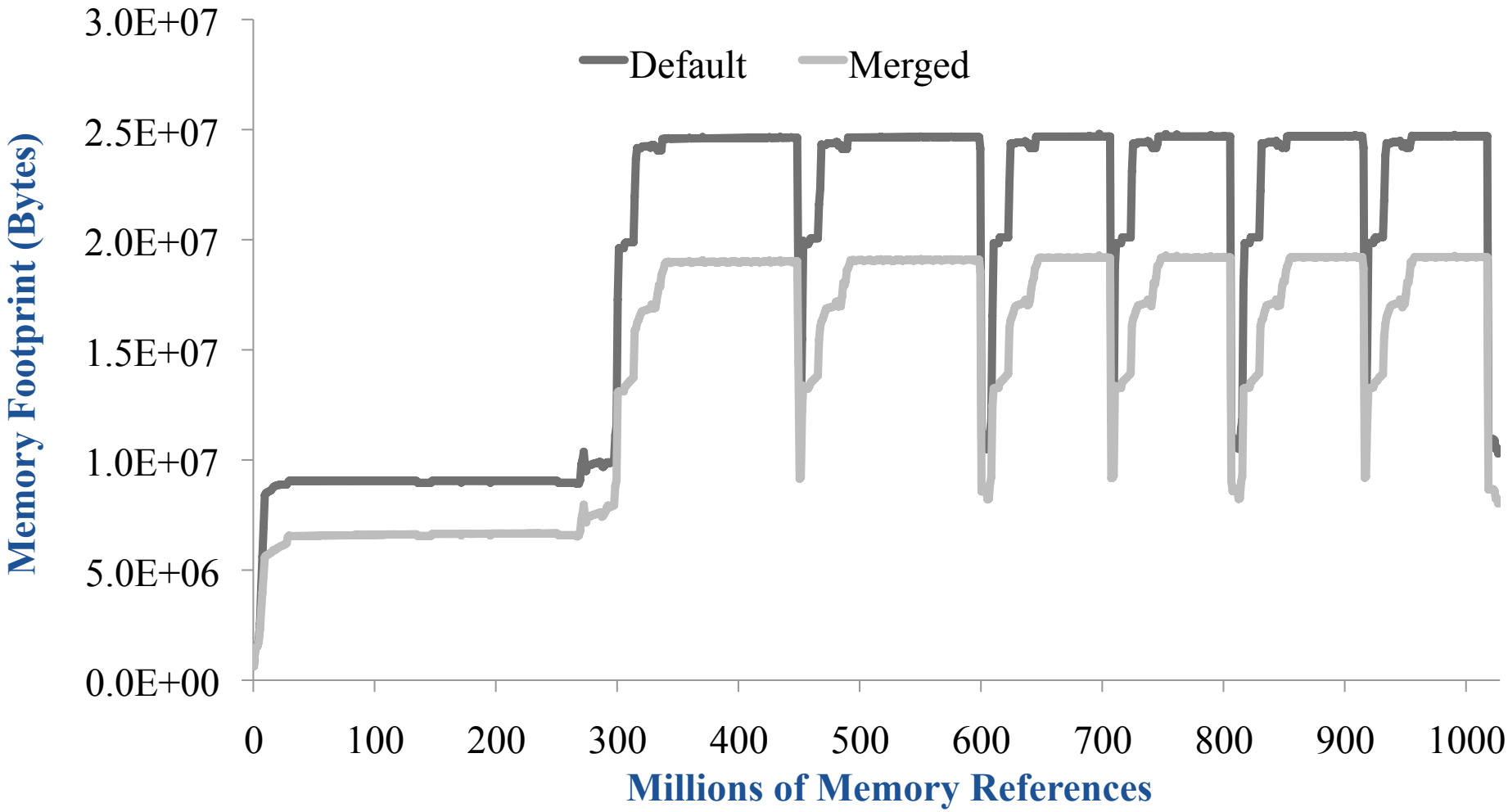


Context-aware memory management can increase problem sizes attainable by MPI applications

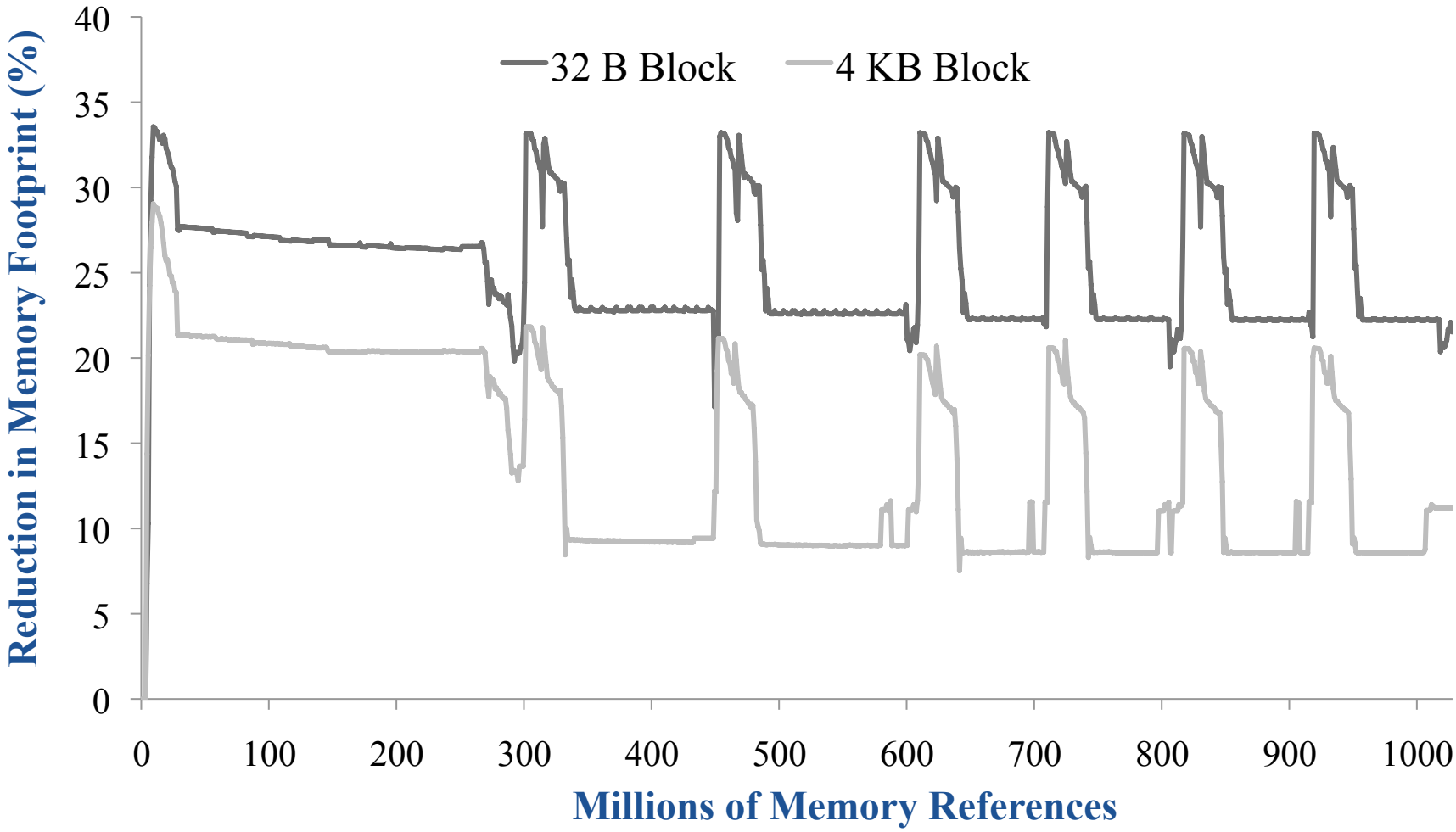
- Substantial data redundancy across MPI tasks
  - Common tables used in database-like fashion
  - Similarities in dynamically evolving data
  - Limits problem sizes even now, worse in future
- Possible ways to reduce redundancy
  - Thread MPI tasks
  - Shared memory with software caching
- SBLLMalloc provides context-aware management
  - Use shared memory with automatic merging
  - Low programmer burden
  - Reduce redundancy in data that is NOT just write once



MPI applications, such as IRS, exhibit high degree of memory similarity across tasks



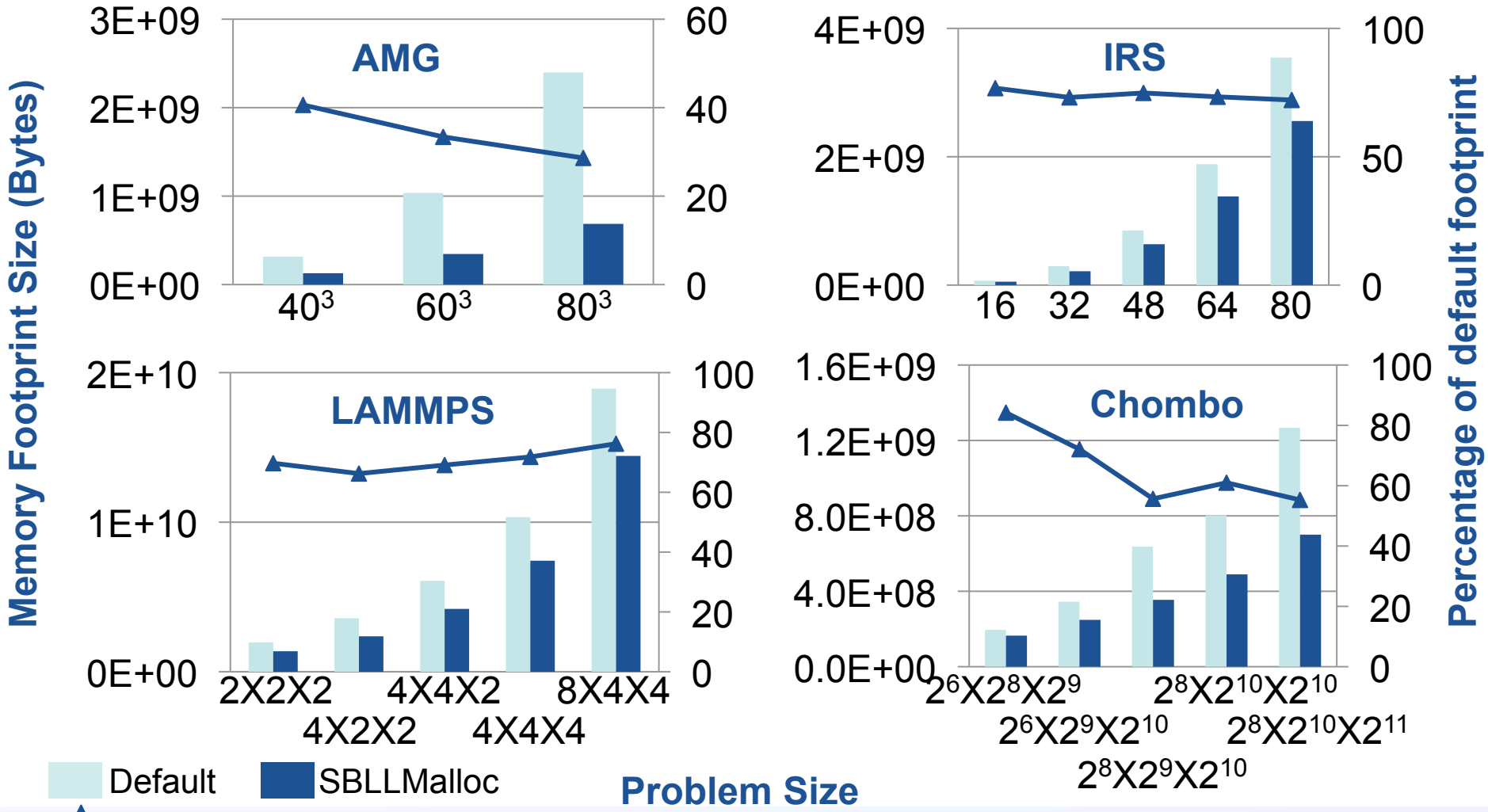
Memory similarity dependence on granularity of comparison and evolves dynamically



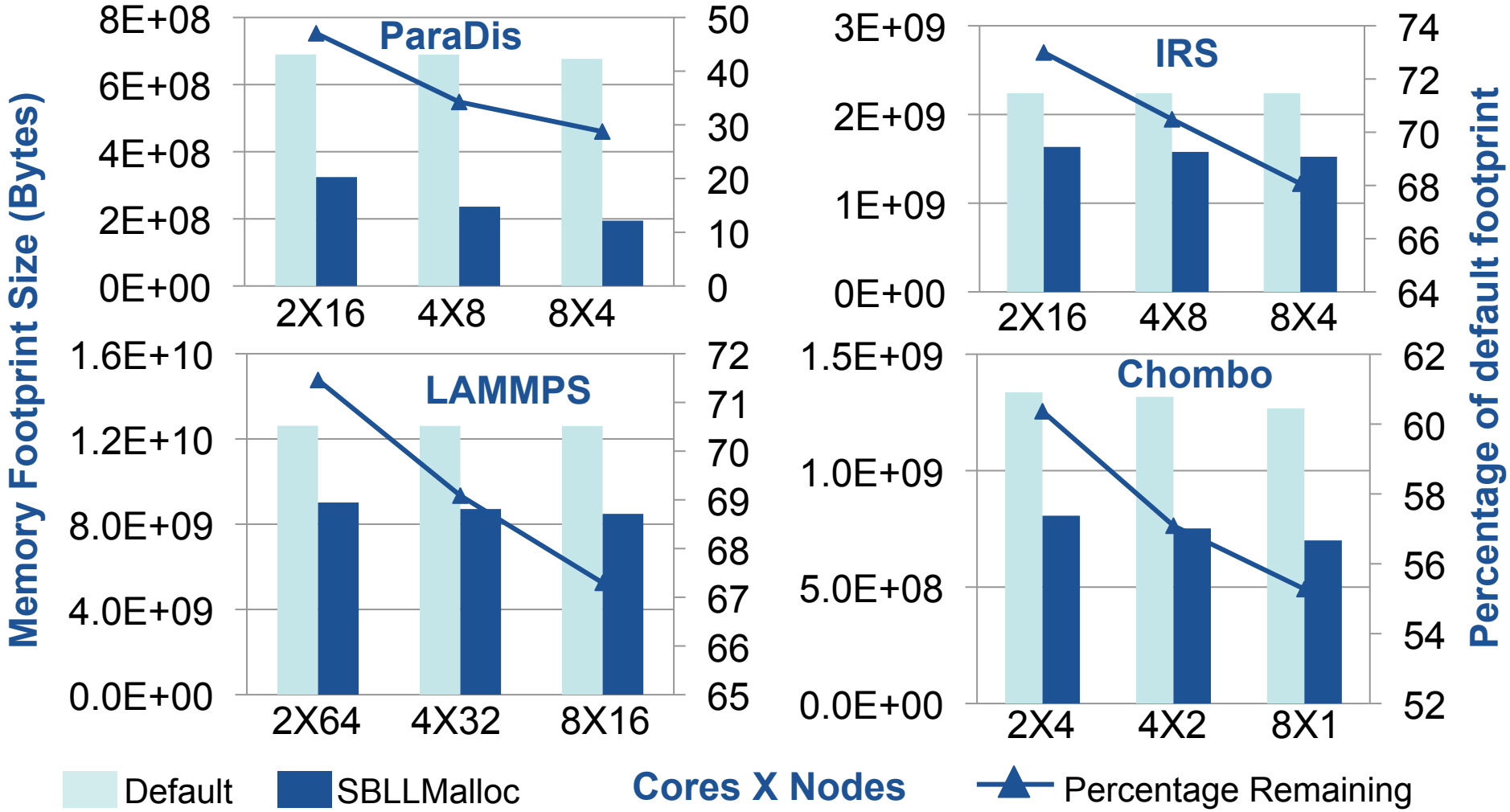
SBLLMalloc implements portable context-aware memory management

- Portable implementation uses POSIX shared memory
  - Similar to distributed shared memory *mmap* techniques
  - Copy-on-write (use *mprotect* to change permissions)
  - Locks (semaphore) for consistency
- Minimal system requirements
  - No kernel modifications required
  - MREMAP_FIXED (> 2.4 kernel)
  - Impacted by limit on mmap'ed pages
    - Higher limit supports more merging
    - Can increase with single command
  - Disable address space randomization
    setarch `uname -m` -R <program> (common HPC default)
- Library-based solution, mostly transparent to users

Substantial memory footprint reductions that are sustained or increase with problem size



Footprint reductions increase with MPI tasks/node



SBLLMalloc achieves is biggest goal: larger problems within memory constraints

AMG Run in 4 nodes (32 cores, 32GB memory)

Run Result 1 AMG run fails without SBLLmalloc from out-of-memory error

```
21893 Killed ./amg2006 -P 2 4 4 -n 120 120 120 -r 20 20 20 -27pt
18160 Killed ./amg2006 -P 2 4 4 -n 120 120 120 -r 20 20 20 -27pt
30576 Killed ./amg2006 -P 2 4 4 -n 120 120 120 -r 20 20 20 -27pt
21561 Killed ./amg2006 -P 2 4 4 -n 120 120 120 -r 20 20 20 -27pt
```

Similar result for IRS: increase problem size 33%

Node	Memory Footprint		SBLLmalloc
	Default	SBLLmalloc	Footprint(%)
1	8105811968	5493301248	67.7699
2	8543088640	5699960832	66.7201
3	8273264640	5708562432	69.0001
4	8526184448	6699823104	78.5794

The SAP and ExaCT Projects equip Livermore to ‘scale’ Sequoia