

Working group name: Solvers, Algorithms and Libraries (SAL)

Members: D. Knoll (LANL), J. Wohlbier (LANL), M. Heroux (SNL), R. Hoekstra (SNL), R. Hornung (LLNL), U. Yang (LLNL), P. Hovland (ANL), R. Mills (ORNL), S. Li (LBNL)

Scope: The Solvers, Algorithms, and Libraries working group (SAL) addresses technical challenges and near-term R&D topics necessary for successful execution of PEM and IC application codes on future architectures:

- SAL and application team efforts are complementary and collaborative.
- Solvers include the full range of linear and nonlinear solvers and preconditioners.
- Algorithms include multiphysics time integration schemes, scale-bridging, load balancing, mesh generation, embedded UQ, and parallelization strategies.
- Libraries provide high-quality reusable software components that can be leveraged across multiple applications.

Assessment of current effort within ASC program:

SNL, LLNL, and LANL have several existing efforts addressing future architectures. These efforts represent only a small fraction of the SAL activities that will be required in the near future.

- New data structures, APIs, and programming models to support a broad range of linear and nonlinear solver methods for evolving modern architectures (SNL).
- Advanced linear solver capabilities for modern architectures including exploration of hybrid threading under MPI, new smoothers and other algorithm components for multigrid, manycore (CPU and GPU) capable preconditioners and block recycling and communication-avoiding iterative methods (LLNL, SNL).
- Miniapps representing key IC and PEM needs to enable performance studies and parallel algorithm exploration for new architecture/programming models (LANL, SNL).
- Exploration of coupled physics algorithms in current codes, such as ALE hydro and multispecies thermal transport, and assessment of their suitability to perform well on future platforms (LLNL).
- New algorithms to improve the mapping of IC time integration approaches to modern architectures (potentially less sequential access of unit physics). This same algorithmic development direction can lead to more self-consistent scale-bridging to many single physics PEM simulation tools on modern architectures (LANL).
- Fault tolerance and remediation in basic linear solver operations (LLNL, SNL).

Specific technical challenges to get to exascale:

Common technical challenges across all SAL activities: (i) migration to scalable manycore (where *scalable* refers to the system design including network, distributed OS, file system, etc.), (ii) increased need for data locality and data placement, (iii) use of hybrid programming models, such as MPI+threading, instead of MPI-only approach, (iv)

communication-reducing or avoiding approaches, (v) redistribution of data for better load balance, (vi) development of appropriate data structures for better data locality and easier access to parallelism, (vii) fault tolerance built into algorithms, and (viii) development of performance models to predict potential bottlenecks.

Specific issues related to Krylov solvers, preconditioners, and multigrid methods:

Most (if not all) of the successful applications on scalable manycore systems have a global SIMT (single instruction multiple thread) parallel execution pattern. This pattern is sufficient for many data parallel computations found in explicit methods and simple implicit problems. However, many NNSA applications have more complicated computational patterns and rely on solvers with sophisticated preconditioners. Right now we do not have algorithms that efficiently map to manycore nodes for these applications. Some specific issues for multigrid libraries include the effect of massive parallelism (millions or billions of cores) on multigrid convergence (e.g. some of the best smoothers are highly sequential, while parallel variants are often less effective), communication issues on coarser levels in algebraic multigrid (increasing stencil sizes require additional neighbor processors but less data per core on coarser levels), and the optimal use of shared memory nodes in the multilevel hierarchy, e.g., larger coarse grids resolved using a shared memory sparse direct solver.

Specific algorithmic challenges for current multiphysics IC code time integration

algorithms: Mapping unit physics packages / libraries onto an entire modern architecture and using standard sequential operator splitting will most likely result in inefficient use of exascale resources. We must re-think current, highly sequential, operator-splitting approaches for efficient implementation of IC codes in the Exascale Initiative. A related algorithmic challenge results from the goal to utilize modern architectures to increase IC and PEM code predictive capability. This goal can be achieved by using more first principles physics, and less phenomenological “modeling” on the system scale. We need to invest in the computational co-design of consistent scale-bridging algorithms.

Other technical challenges: We need to understand when computing data “on the fly” is more beneficial than using pre-computed data tables. We must motivate and consider the development and use of embedded UQ approaches. Validating existing miniapps against real codes and generating additional miniapps for use in development of new SAL methods will be challenging, and will require close interaction with the Apps and PM working groups. Additionally, C++ template meta-programming is currently the most effective way to write portable manycore software. However, this may not be sufficient for all computations in complex multiphysics codes. Regardless of the programming model, the software refactoring effort may be huge and the tools to do it are limited. This is a big risk factor, especially for Fortran based development. Emerging open standards for writing portable manycore software, such as OpenCL, must also be considered.

R&D Opportunities for the near term (defined as the next 3 years):

It should be clear that an increased R&D effort and subsequent progress in SAL activities is required for successful transition of IC and PEM activities to exascale resources. Given

the broad possibilities in the Exascale Initiative architecture landscape, it is imperative to embark on an equally broad SAL R&D program.

Manycore preconditioners: We must develop non-SIMT preconditioners (esp. smoothers for multigrid) that are more robust than polynomial or Jacobi scaling. This is an important area where we must have success in the next few years. Specifically for multigrid libraries, we must investigate approaches that reduce communication such as combining more local smoothing steps, aggregating messages to reduce latency overhead, using idle processors to accelerate convergence, and non-Galerkin approaches for algebraic multigrid to control stencil growth. Multigrid in time is also of interest.

Physics-based approaches: Solvers and preconditioners tailored to particular sets of equations and discretizations: rather than focus on making general solvers work at exascale, it might be better to have a suite of specialized solvers. By exploiting knowledge of physics and discretizations solvers may scale better and be more robust. Discretization-specific solvers have been shown to be necessary for E&M and MHD problems.

Multiphysics time integration: Co-design new algorithmic approaches for multiphysics time integration in IC codes. Current operator splitting approaches are highly sequential and require mapping each unit physics library / package across the entire architectures. Research is required to develop algorithms with less sequential operator splitting, co-designed with application code developers for modern architectures.

Scale-bridging: Co-design two-way, consistent scale-bridging algorithms. This area has been highlighted more by PEM, but has the potential to be very synergistic with multiphysics time integration research. The goal of this work is the collaborative development with app code teams of algorithmic capabilities that allow system-scale simulation with fine-scale physics fidelity. This is a natural candidate for effective use of exascale resources and reduced reliance on approximate closure models.

Embedded UQ: There is a significant R&D opportunity in the development, acceptance, and implementation of embedded UQ methods.

Fault tolerance: Develop techniques to validate the results of numerical algorithms without the use of replication and to correct errors, including random and fail-stop errors. Methods must maintain high performance, using load balancing and related techniques.

Miniapps: All efforts discussed in the section will benefit from development and use of miniapps that represent key aspects of PEM and IC efforts. We (SNL) have existing miniapps for unstructured implicit finite element (MiniFE), explicit dynamics (PhdMesh), molecular dynamics (MiniMD) and circuit modeling (MiniXyce). We (LANL) also have the MAMA project (miniapps on modern architectures) which uses C++ and OpenCL to create an environment for flexible task and data parallelism. MAMA will enable rapid miniapp development for modern architectures. We expect to develop more in order to provide coverage of important large-scale application needs.