Numerical Software: Foundational Tools for HPC Simulations

Presented to
ATPESC 2017 Participants

Lori Diachin
Department Head, Information Technology
Computation Directorate, LLNL

Q Center, St. Charles, IL (USA)
Date 08/07/2017

ATPESC Numerical Software Track

Track 4: Numerical Algorithms and Software: Tutorial Goals

1. Provide a basic understanding of a variety of applied mathematics algorithms for scalable linear, nonlinear, and ODE solvers as well as discretization technologies (e.g., adaptive mesh refinement for structured and unstructured grids)

2. Provide an overview of software tools available to perform these tasks on HPC architectures … including where to go for more info

3. Practice using one or more of these software tools on basic demonstration problems

This presentation gives a high-level introduction to HPC numerical software

- How HPC numerical software addresses challenges in computational science and engineering (CSE)
- Toward extreme-scale scientific software ecosystems
- Using and contributing: Where to go for more info

Why is this important for you?

– Libraries enable users to focus on their primary interests
  - Reuse algorithms and data structures developed by experts
  - Customize and extend to exploit application-specific knowledge
  - Cope with complexity and changes over time

– More efficient, robust, reliable, scalable, sustainable scientific software

– Better science, broader impact of your work

This work is founded on decades of experience and concerted team efforts to improve numerical software

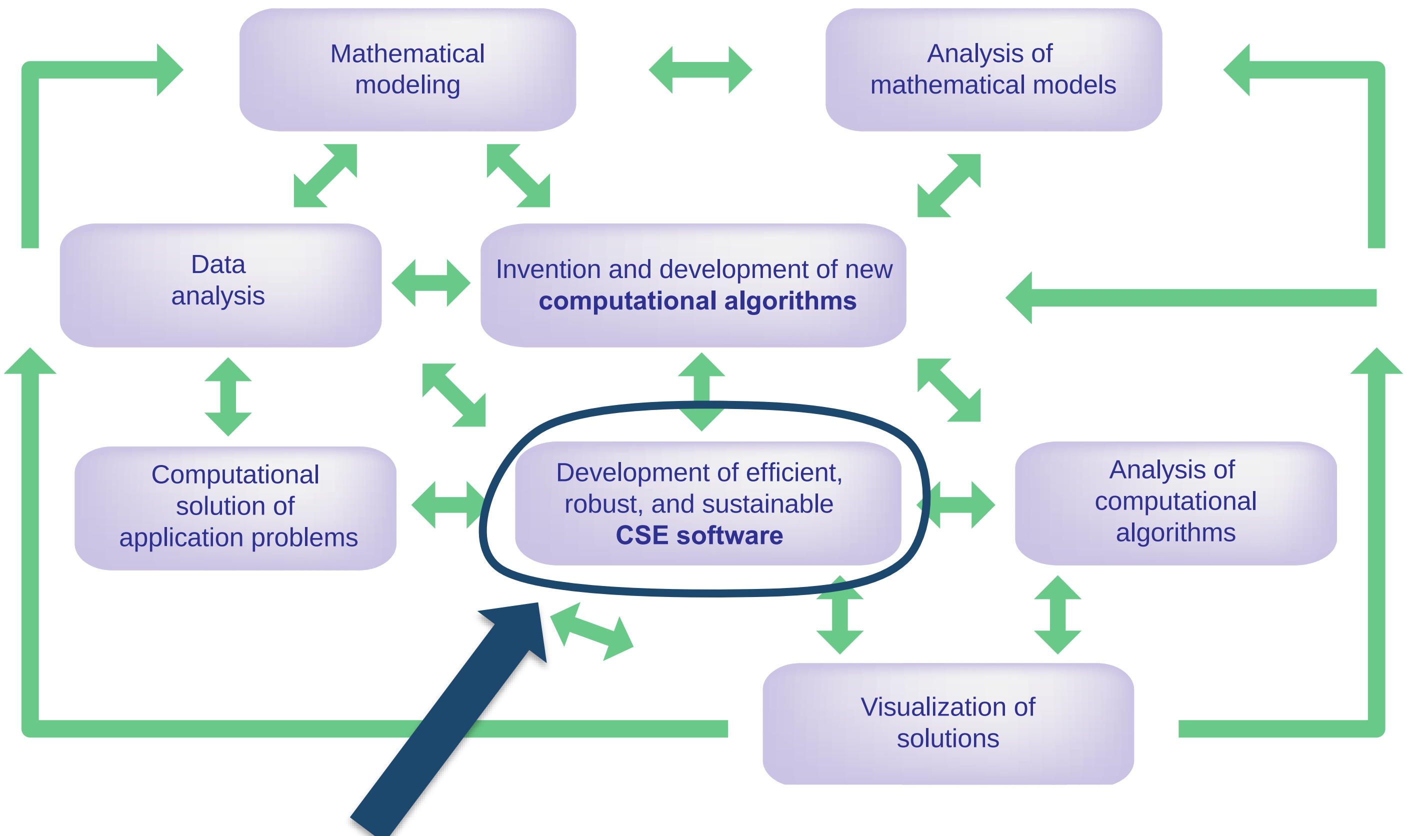
- FASTMath SciDAC Institute
- IDEAS Scientific Software Productivity
- Exascale Computing Project

fastmath-scidac.llnl.gov

ideas-productivity.org

Funded by the U.S. Department of Energy's Office of Science

Software is at the core of computational science and engineering



Software: foundation of sustained CSE collaboration and scientific progress

Multiphysics is a primary motivator for extreme-scale computing

Feb 2013
doi:10.1177/1094342012468181

THE INTERNATIONAL JOURNAL of HIGH PERFORMANCE COMPUTING APPLICATIONS

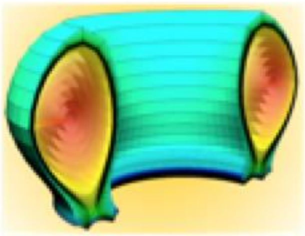
Multiphysics simulations: Challenges and opportunities

The International Journal of High Performance Computing Applications
27(1) 4–83
© The Author(s) 2012
Reprints and permissions:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/1094342012468181
hpc.sagepub.com
SAGE

David E Keyes[1,2], Lois C McInnes[3], Carol Woodward[4], William Gropp[5], Eric Myra[6], Michael Pernice[7], John Bell[8], Jed Brown[3], Alain Clo[1], Jeffrey Connors[4], Emil Constantinescu[3], Don Estep[9], Kate Evans[10], Charbel Farhat[11], Ammar Hakim[12], Glenn Hammond[13], Glen Hansen[14], Judith Hill[10], Tobin Isaac[15], Xiangmin Jiao[16], Kirk Jordan[17], Dinesh Kaushik[3], Efthimios Kaxiras[18], Alice Koniges[8], Kihwan Lee[19], Aaron Lott[4], Qiming Lu[20], John Magerlein[17], Reed Maxwell[21], Michael McCourt[22], Miriam Mehl[23], Roger Pawlowski[14], Amanda P Randles[18], Daniel Reynolds[24], Beatrice Rivière[25], Ulrich Rüde[26], Tim Scheibe[13], John Shadid[14], Brendan Sheehan[9], Mark Shephard[27], Andrew Siegel[3], Barry Smith[3], Xianzhu Tang[28], Cian Wilson[2] and Barbara Wohlmuth[23]

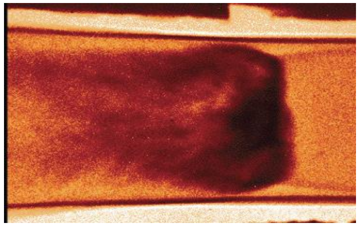
Multiphysics: greater than one component governed by its own principle(s) for evolution or equilibrium

Also: broad class of coarsely partitioned problems possess similarities to multi-physics

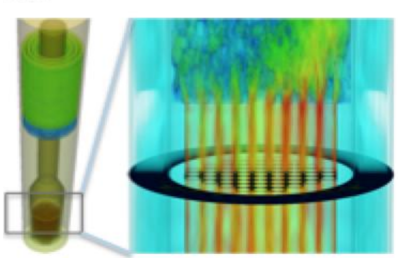


fusion
(A. Hakim, PPPL)

linear accelerators
(K. Lee, SLAC)



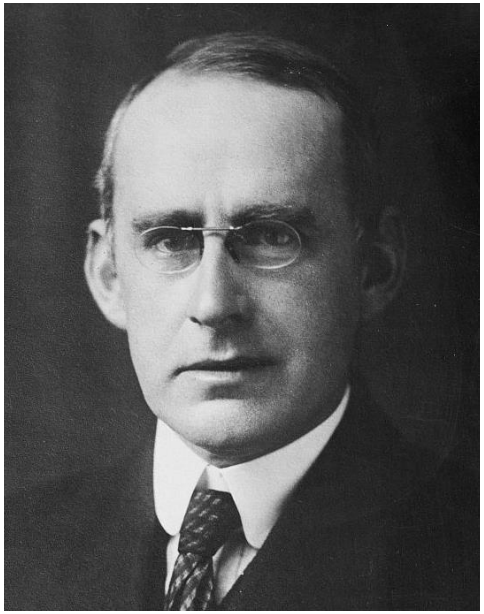
radiation hydrodynamics
(E. Myra, U Michigan)



nuclear reactors
(A. Siegel, ANL)

Multiphysics challenges … the study of ‘and’

“We often think that when we have completed our study of one we know all about two, because ‘two’ is ‘one and one.’ We forget that we still have to make a study of ‘and.’ ”



− Sir Arthur Stanley Eddington (1892−1944), British astrophysicist

Software is the practical means for sustained extreme-scale CSE collaboration

Enough computational power to enable:

- Multirate, multiscale, multicomponent, multiphysics
- Uncertainty quantification and sensitivities
- Simulations involving stochastic quantities
- Optimization and design over full-featured simulations
- Coupling of simulations and data analytics

Beyond interpretive simulations
…
working toward predictive science

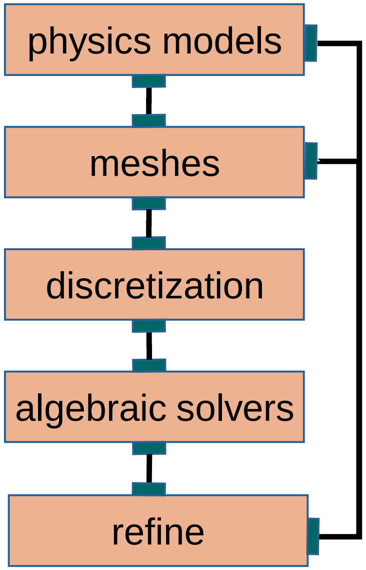
“The way you get programmer productivity is by eliminating lines of code you have to write.”

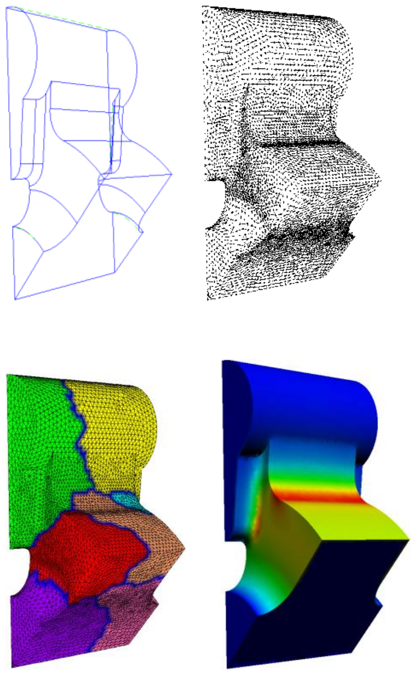
– Steve Jobs, Apple World Wide Developers Conference, Closing Keynote Q&A, 1997

CSE simulation relies on high-performance numerical algorithms and software

- Develop a mathematical model of the phenomenon of interest
- Approximate the model using a discrete representation
- Solve the discrete representation
- Adapt and refine the mesh or model
- Incorporate different physics, scales

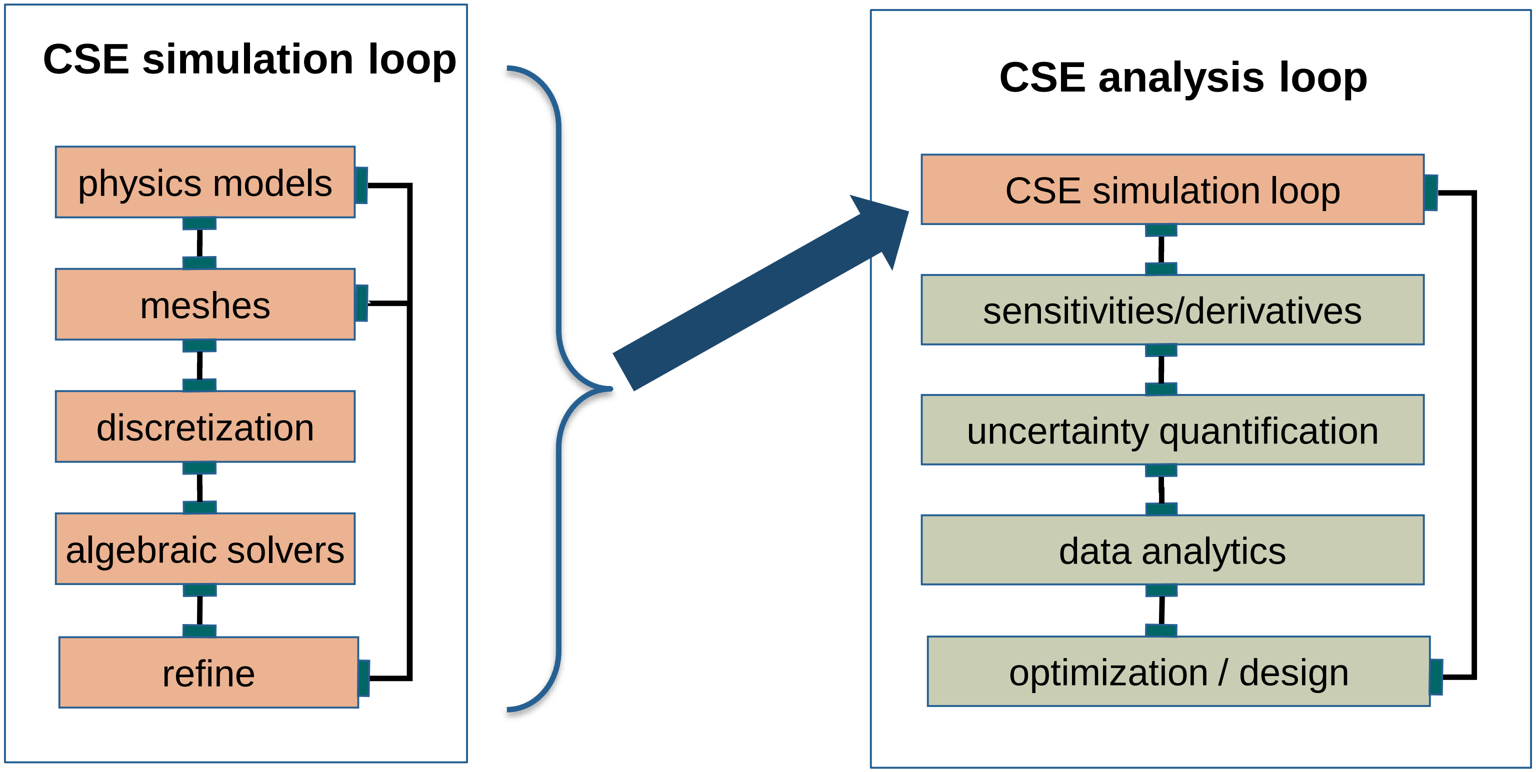
CSE simulation loop





These steps require: mesh generation, partitioning, load balancing, high-order discretization, time integration, linear and nonlinear solvers, eigensolvers, mesh refinement, multiscale/multiphysics coupling methods, etc.

CSE analysis builds on the CSE simulation loop … and relies on even more numerical algorithms and software



These steps require: adjoints, sensitivities, algorithmic differentiation, sampling, ensemble simulations, uncertainty quantification, data analytics, optimization (derivative free and derivative based), inverse problems, etc.

First consider a very simple example

- 1D rod with one end in a hot water bath, the other in a cold water bath
- Mathematical model

$$\nabla^2 T = 0 \in \Omega$$

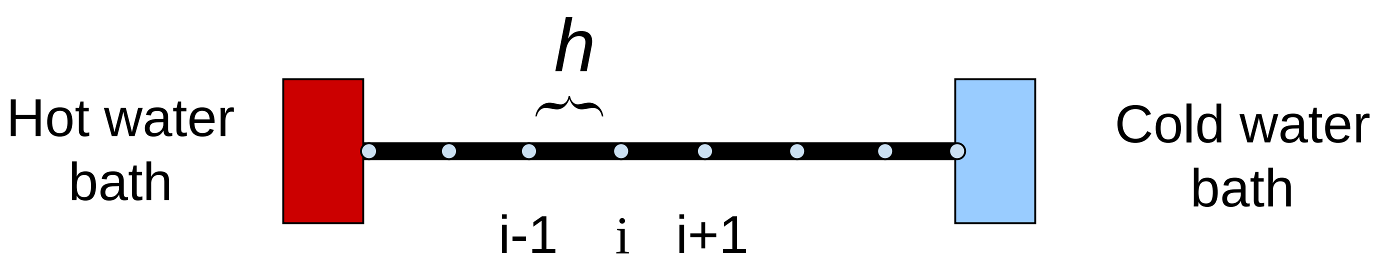
$$T(0) = 180^o \quad T(1) = 0^o$$

The first step is to discretize the equations

- Approximate the derivatives in the continuous equations with a discrete representation that is easier to solve
- One approach: Finite differences

$$\nabla^2 T \approx (T_{i+1} - 2T_i + T_{i-1})/h^2 = 0$$

$$T_0 = 180^o \qquad T_n = 0^o$$



Then you can solve for the unknowns T_i

- Set up a matrix of the unknown coefficients
  - include the known boundary conditions
- Solve the linear system for T_i

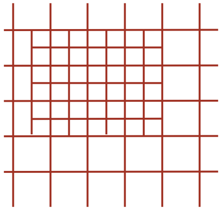
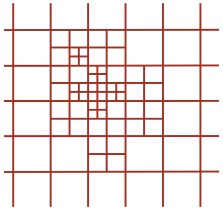
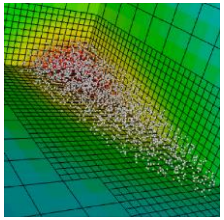
$$
\begin{pmatrix}
2 & -1 & 0 & \cdots & & 0 \\
-1 & 2 & -1 & 0 & \cdots & 0 \\
0 & -1 & 2 & -1 & 0 \cdots & 0 \\
 & & \cdots & & & \\
0 & \cdots & & 0 & -1 & 2
\end{pmatrix}
\begin{pmatrix}
T_1 \\ T_2 \\ T_3 \\ . \\ T_{n-1}
\end{pmatrix}
=
\begin{pmatrix}
180\, h^2 \\ 0 \\ 0 \\ . \\ 0
\end{pmatrix}
$$

- Visualize and analyze the results

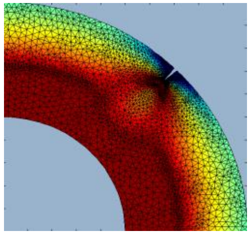
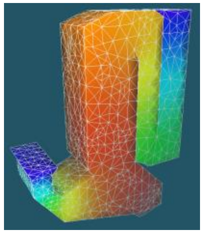
As problems get more complicated, so do the steps in the process

- Different discretization strategies exist for differing needs

Efficiency

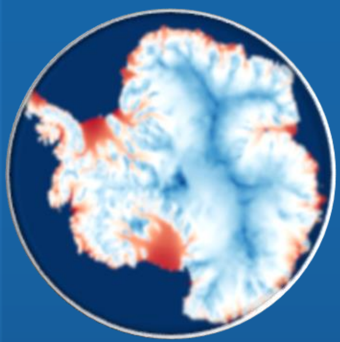




Flexibility

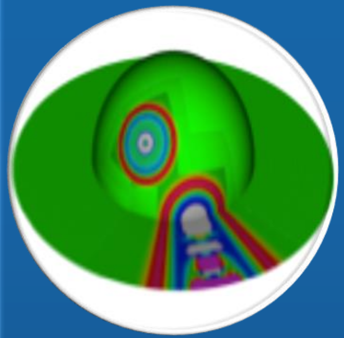



- Most problems are time dependent and nonlinear
  - Need higher algorithmic levels than linear solvers
- Increasingly combining multiple physical processes
  - Interactions require careful handling
- Goal-oriented problem solving requires optimization, uncertainty quantification

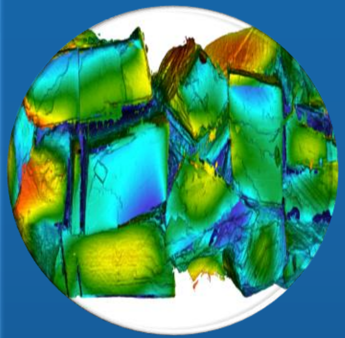
Structured grid efforts focus on high-order, mapped grids, embedded boundaries, AMR, and particles



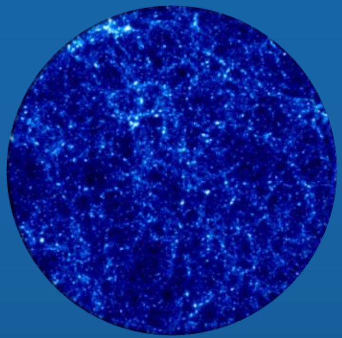
Structured AMR



Mapped-multiblock grids



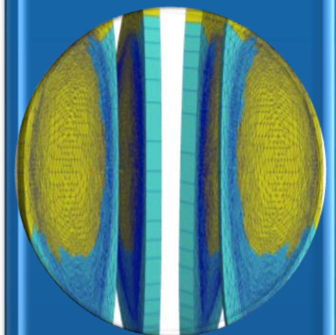
Embedded boundary methods



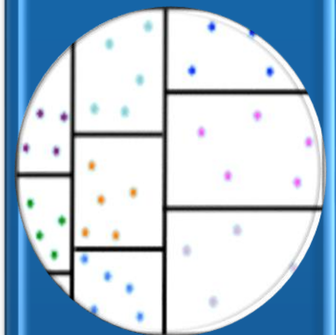
Particle-based methods

Application to cosmology, astrophysics, accelerator modeling, fusion, climate, subsurface reacting flows, low mach number combustion, etc.

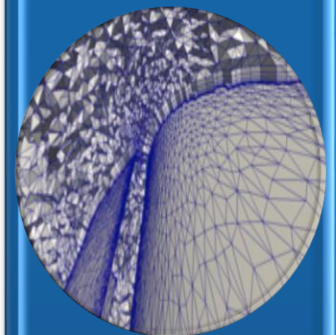
Unstructured grid capabilities focus on adaptivity, high-order, and the tools needed for extreme scaling



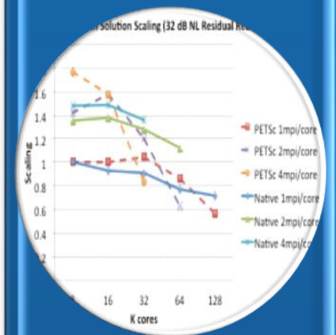
Parallel mesh infrastruct-ures



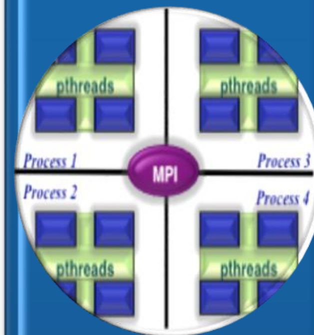
Dynamic load balancing



Mesh adaptation and quality control



Parallel performance on unstructured meshes

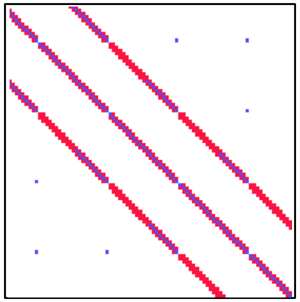


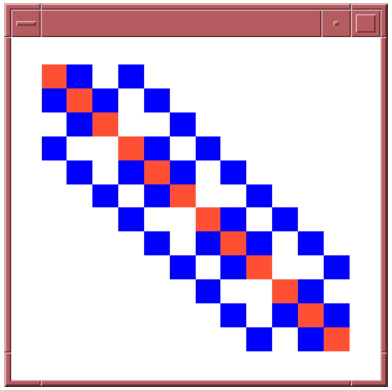
Architecture aware implementa-tions

Application to fusion, climate, accelerator modeling, NNSA applications, nuclear energy, manufacturing processes, etc.

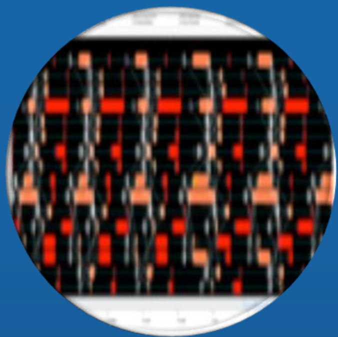
As problems grow in size, so do the corresponding discrete systems

- Targeting applications with billions grid points and unknowns
- Most linear systems resulting from these techniques are LARGE and sparse
- Often most expensive solution step
- Solvers:
  - Direct methods (e.g. Gaussian Elimination)
  - Iterative methods (e.g. Krylov Methods)
    - Preconditioning is typically critical
    - Mesh quality affects convergence rate
- Many software tools deliver this functionality as numerical libraries
  - hypre, PETSc, SuperLU, Trilinos, etc.

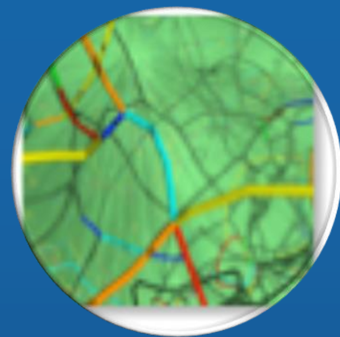




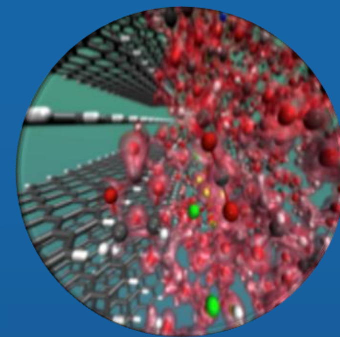
Research on algebraic systems provides key solution technologies to applications



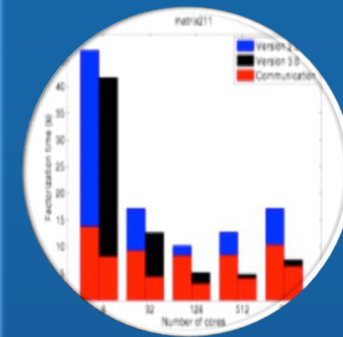
Linear system solution using direct and iterative solvers



Nonlinear system solution using acceleration techniques and globalized Newton methods



Eigensolvers using iterative techniques and optimization



Architecture aware implementations

Application to fusion, nuclear structure calculation, quantum chemistry, accelerator modeling, climate, dislocation dynamics etc,

Trends and challenges in today’s computing environment

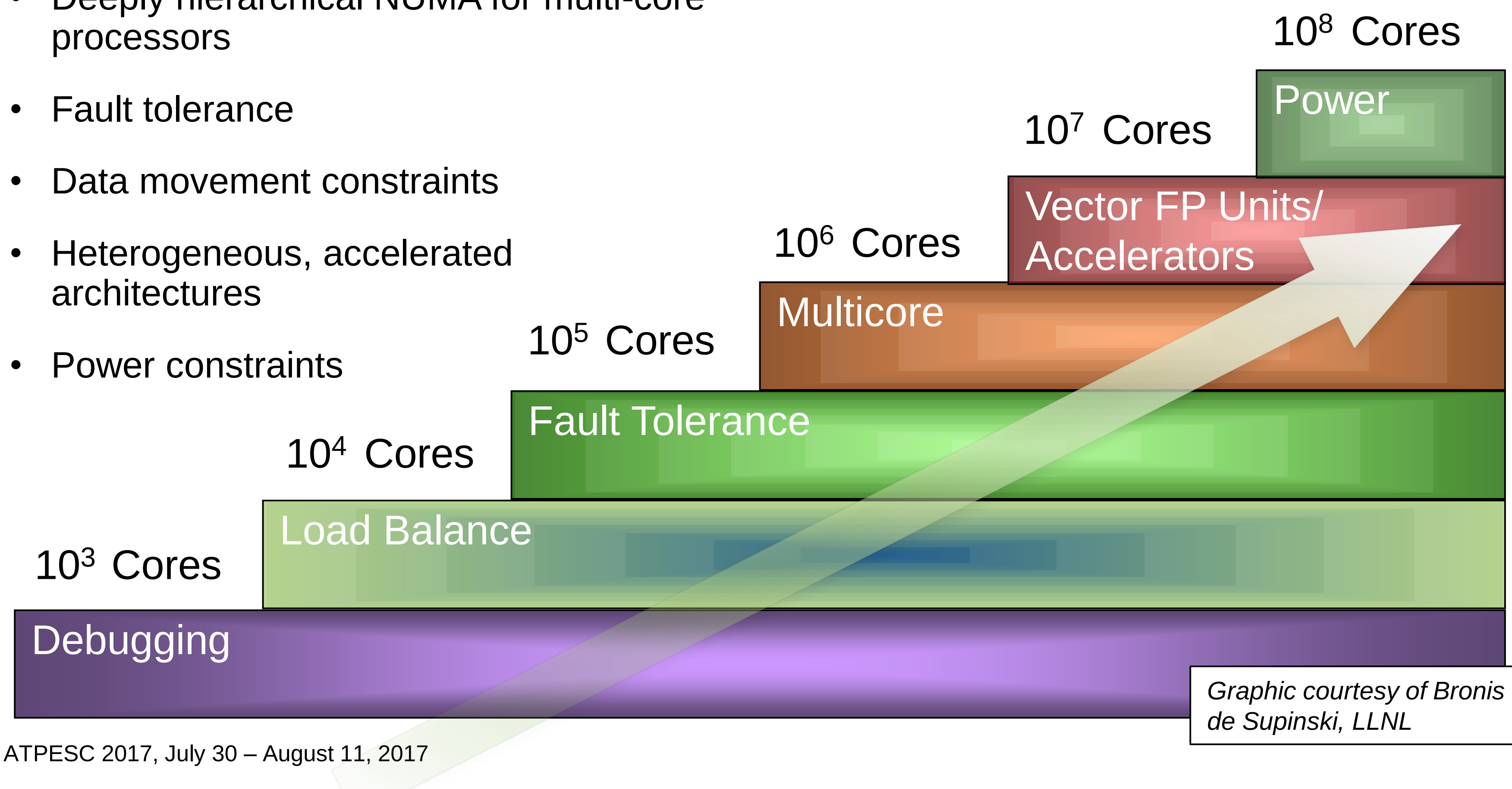
Theta: ALCF early production system

- **Fundamental trends:**
  - Disruptive hardware changes
    - Require algorithm/code refactoring
  - Need coupling, optimization, sensitivities
    - Multiphysics, multiscale, data analytics
- **Challenges:**
  - Need refactoring: Really, continuous change
  - Modest funding for app development: No monolithic apps
  - Requirements are unfolding, evolving, not fully known *a priori*
- **Opportunities:**
  - Better design, software practices, and tools are available
  - Better software architectures: toolkits, libraries, frameworks
  - Open-source software, community collaboration

Simulation is significantly complicated by the change in computing architectures

Scientific computing software must address ever increasing challenges:

- Million to billion way parallelism
- Deeply hierarchical NUMA for multi-core processors
- Fault tolerance
- Data movement constraints
- Heterogeneous, accelerated architectures
- Power constraints

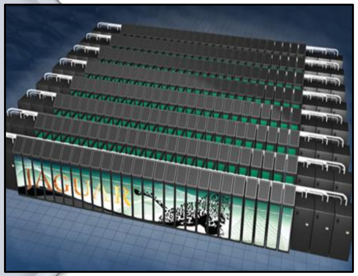


Graphic courtesy of Bronis de Supinski, LLNL

Research to improve performance on HPC platforms focuses on inter- and intra-node issues

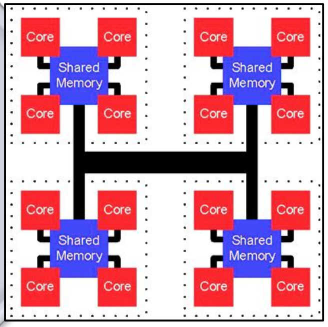
Inter-node: Massive Concurrency

- Reduce communication
- Increase concurrency
- Reduce synchronization
- Address memory footprint
- Enable large communication/computation overlap



Intra-node: Deep NUMA

- MPI + threads for many packages
- Compare task and data parallelism
- Thread communicator to allow passing of thread information among libraries
- Low-level kernels for vector operations that support hybrid programming models



New algorithms are being developed that address key bottlenecks on modern day computers

Reduce communication

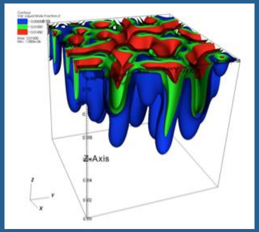
- AMG: develop non-Galerkin approaches, use redundancy or agglomeration on coarse grids, develop additive AMG variants (hypre) (2X improvement)
- Hierarchical partitioning optimizes communication at each level (Zoltan) (27% improvement in matrix-vector multiply)
- Relaxation and bottom solve in AMR multigrid (Chombo) (2.5X improvement in solver, 40% overall)

Increase concurrency

- New spectrum slicing eigensolver in PARPACK (Computes 10s of thousands of eigenvalues in small amounts of time)
- New pole expansion and selected inversion schemes (PEXSI) (now scales to over 100K cores)
- Utilize BG/Q architecture for extreme scaling demonstrations (PHASTA) (3.1M processes on 768K cores unstructured mesh calculation)

Reduce synchronization points

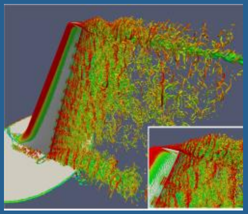
- Implemented pipelined versions of CG and conjugate residual methods; 4X improvement in speed (PETSc) (30% speed up on 32K cores)



Used in PFLOTRAN applications

Address memory footprint issues

- Predictive load balancing schemes for AMR (Zoltan) (Allows AMR runs to complete by maintaining memory footprint)
- Hybrid programming models



Used in PHASTA extreme scale applications

Increase communication and computation overlap

- Improved and stabilized look-ahead algorithms (SuperLU) (3X run time improvement)

Used in Omega3P accelerator simulations

Software libraries facilitate CSE progress

- **Software library:** a high-quality, encapsulated, documented, tested, and multiuse software collection that provides functionality commonly needed by application developers
  - Organized for the purpose of being reused by independent (sub)programs
  - User needs to know only
    - Library interface (not internal details)
    - When and how to use library functionality appropriately

- **Key advantages** of software libraries
  - Contain complexity
  - Leverage library developer expertise
  - Reduce application coding effort
  - Encourage sharing of code, ease distribution of code

- References:
  - https://en.wikipedia.org/wiki/Library_(computing)
  - What are Interoperable Software Libraries? Introducing the xSDK

A broad range of HPC numerical software addresses these challenges

Some packages with general-purpose, reusable algorithmic infrastructure in support of high-performance CSE:

- **AMReX** – https://github.com/AMReX-codes/amrex
- **Chombo** - https://commons.lbl.gov/display/chombo
- **Clawpack** - http://www.clawpack.org
- **Deal.II** - https://www.dealii.org
- **FEniCS** - https://fenicsproject.org
- **hypre** - http://www.llnl.gov/CASC/hypre
- **libMesh** - https://libmesh.github.io
- **MAGMA** - http://icl.cs.utk.edu/magma
- **MFEM** - http://mfem.org
- **PETSc/TAO** – http://www.mcs.anl.gov/petsc
- **PUMI** - http://github.com/SCOREC/core
- **SUNDIALS** - http://computation.llnl.gov/casc/sundials
- **SuperLU** - http://crd-legacy.lbl.gov/~xiaoye/SuperLU
- **Trilinos** - https://trilinos.org
- **Uintah** - http://www.uintah.utah.edu
- **waLBerla** - http://www.walberla.net

See info about scope, performance, usage, and design, including:

- tutorials
- demos
- examples
- how to contribute

★ Discussed today: Gallery of highlights

… and many, many more...

Explore, use, contribute!

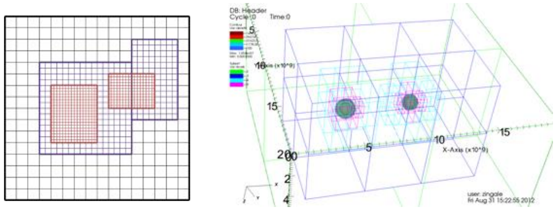
Gallery of highlights

- Overview of HPC numerical software packages
- 1 slide per package, emphasizing key capabilities, highlights, and where to go for more info

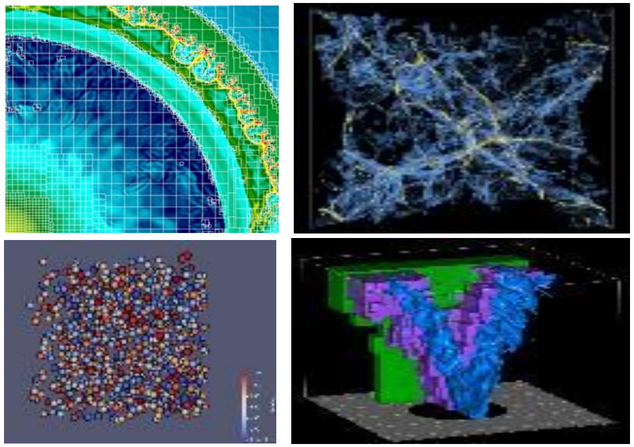
AMReX

Block-structured adaptive mesh refinement framework. Support for hierarchical mesh and particle data with embedded boundary capability.

- **Capabilities**
  - Support for solution of PDEs on hierarchical adaptive mesh with particles and embedded boundary representation of complex geometry
    - Core functionality in C++ with frequent use of Fortran90 kernels
  - Support for multiple modes of time integration
  - Support for explicit and implicit single-level and multilevel mesh operations, multilevel synchronization, particle, particle-mesh and particle-particle operations
  - Hierarchical parallelism -- hybrid MPI + OpenMP with logical tiling to work efficiently on new multicore architectures
  - Native multilevel geometric multigrid solvers for cell-centered and nodal data
  - Highly efficient parallel I/O for checkpoint/restart and for visualization – native format supported by Visit, Paraview, yt
  - Tutorial examples, Users Guide available in download
- **Open source software**
  - Used for a wide range of applications including accelerator modeling, astrophysics, combustion, cosmology, multiphase flow…
  - Freely available on github



Examples of 2D and 3D grids



Examples of AMReX applications

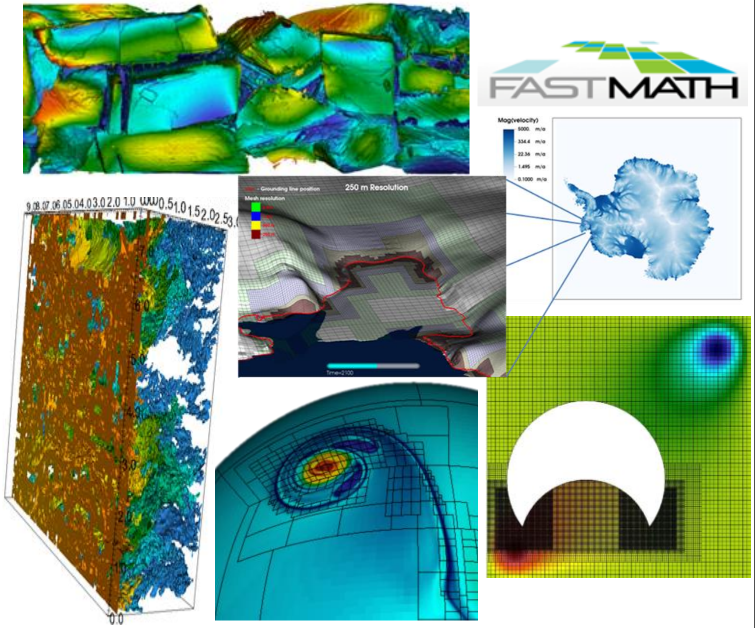
https://www.github.com/AMReX-Codes/amrex

Aug 2017

Chombo

Scalable adaptive mesh refinement (AMR) framework. Enables implementing scalable AMR applications with support for complex geometries.

- **Adaptive Mesh Refinement (AMR)**
  - Block structured AMR dynamically focuses computational effort where needed to improve solution accuracy
  - Designed as a developers' toolbox for implementing scalable AMR applications
  - Implemented in C++/Fortran
  - Solvers for hyperbolic, parabolic, and elliptic systems of PDEs
- **Complex geometries**
  - Embedded-boundary (EB) methods use cut-cell approach to embed complex geometries in a regular Cartesian mesh
  - EB mesh generation is extremely efficient
  - Structured EB meshes make high performance easier to attain
- **Higher-order finite-volume**
  - Higher (4^{th})-order schemes reduce memory footprint and improve arithmetic intensity
  - Good fit for emerging architectures
  - Both EB and mapped-multiblock approaches to complex geometry



Clockwise from top – Crushed calcite in capillary tube (EB), ice sheet modeling (AMR), 4^{th}-order EB/AMR, Shallow-water vortices on a sphere (AMR/mapped-multiblock), flooding in fractured shale (EB).

http://Chombo.lbl.gov

Aug 2017

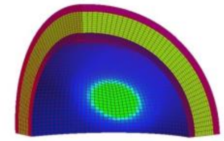
hypre

Lawrence Livermore National Laboratory

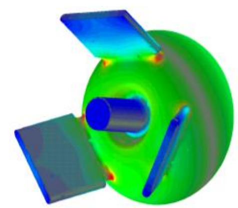
Highly scalable multilevel solvers and preconditioners.
Unique user-friendly interfaces. Flexible software design.
Used in a variety of applications. Freely available.

- **Conceptual interfaces**
  - Structured, semi-structured, finite elements, linear algebraic interfaces
  - Provide natural "views" of the linear system
  - Provide for more efficient (scalable) linear solvers through more effective data storage schemes and more efficient computational kernels
- **Scalable preconditioners and solvers**
  - Structured and unstructured algebraic multigrid (including constant-coefficient solvers)
  - Maxwell solvers, H-div solvers, and more
  - Matrix-free Krylov solvers
- **Open source software**
  - Used worldwide in a vast range of applications
  - Can be used through PETSc and Trilinos
  - Available on github

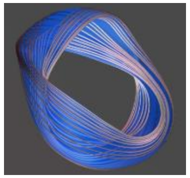
hypre high performance preconditioners



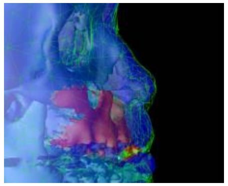
Elasticity / Plasticity



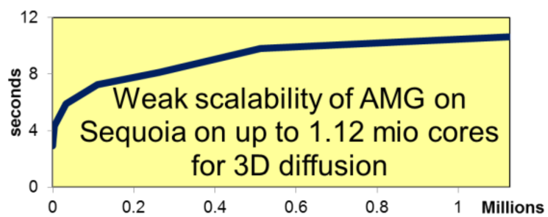
Electro-magnetics



Magneto-hydrodynamics



Facial surgery



http://www.llnl.gov/CASC/hypre

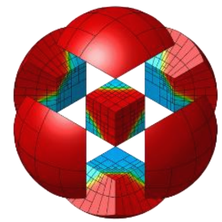
Aug 2017

MFEM

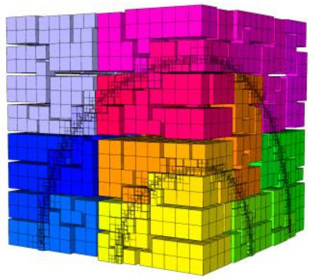
Lawrence Livermore National Laboratory

Free, lightweight, scalable C++ library for finite element methods. Supports arbitrary high order discretizations and meshes for wide variety of applications.

- **Flexible discretizations on unstructured grids**
  - Triangular, quadrilateral, tetrahedral and hexahedral meshes.
  - Local conforming and non-conforming refinement.
  - Bilinear/linear forms for variety of methods: Galerkin, DG, DPG, …
- **High-order and scalable**
  - Arbitrary-order H1, H(curl), H(div)- and L2 elements. Arbitrary order curvilinear meshes.
  - MPI scalable to millions of cores. Enables application development on wide variety of platforms: from laptops to exascale machines.
- **Built-in solvers and visualization**
  - Integrated with: HYPRE, SUNDIALS, PETSc, SUPERLU, …
  - Accurate and flexible visualization with VisIt and GLVis
- **Open-source software**
  - LGPL-2.1 with thousands of downloads/year worldwide.
  - Available on GitHub. Part of ECP's CEED co-design center.

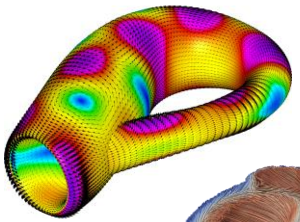


High order curved elements



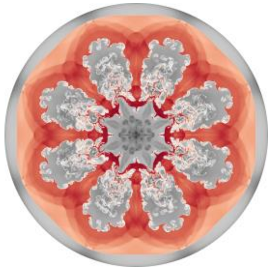
Parallel non-conforming AMR

ECP ASC CEED EXASCALE DISCRETIZATIONS FASTMATH



Surface meshes

Heart modelling



Compressible flow ALE simulations

http://mfem.org

Aug 2017

PETSc/TAO:

Portable, Extensible Toolkit for Scientific Computation / Toolkit for Advanced Optimization

Scalable algebraic solvers for PDEs. Encapsulate parallelism in high-level objects. Active & supported user community. Full API from Fortran, C/C++, Python.

Optimization

Time Integrators

Nonlinear Algebraic Solvers

Krylov Subspace Solvers

Preconditioners

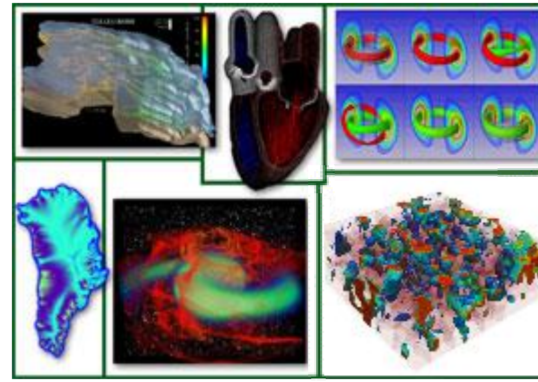
Domain-Specific Interfaces: Networks; Quadtree / Octree; Unstructured Mesh; Structured Mesh

Vectors | Index Sets | Matrices

Computation & Communication Kernels

- **Easy customization and composability of solvers <u>at runtime</u>**
  - Enables optimality via flexible combinations of physics, algorithmics, architectures
  - Try new algorithms by composing new/existing algorithms (multilevel, domain decomposition, splitting, etc.)
- **Portability & performance**
  - Largest DOE machines, also clusters, laptops
  - Thousands of users worldwide

Argonne NATIONAL LABORATORY



PETSc provides the backbone of diverse scientific applications.

clockwise from upper left: hydrology, cardiology, fusion, multiphase steel, relativistic matter, ice sheet modeling

ECP xSDK FASTMATH R&D 100

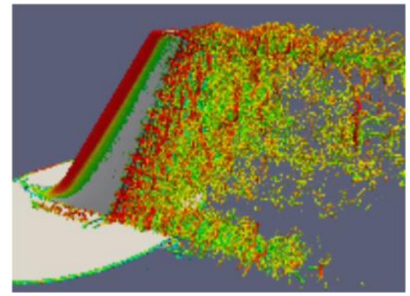
https://www.mcs.anl.gov/petsc

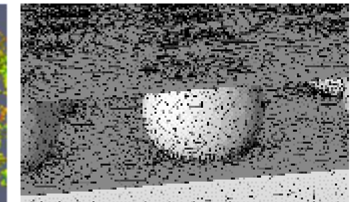
Aug 2017

Parallel Unstructured Mesh Infrastructure

Parallel management & adaptation of unstructured meshes. Interoperable components to support the development of unstructured mesh simulation workflows

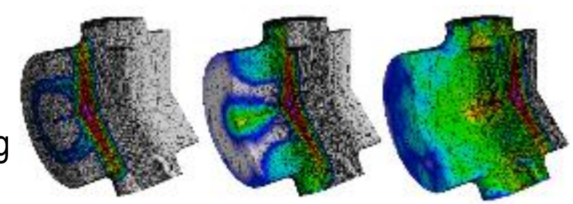
- **Key PUMI Components:**
  - Distributed, conformant mesh with entity migration, remote read-only copies, fields and their operations
  - Link to geometry and physical attributes
  - Mesh adaptation (straight and curved), mesh motion
  - Multi-criteria partition improvement
  - Distributed mesh support for Particle-In-Cell methods
- **Designed for integration into existing codes**
- **In-memory integrations:**
  - PHASTA (FE or turbulent flows)
  - FUN3D (FV CFD)
  - Proteus (multiphase FE)
  - ACE3P (High-order FE electromagnetics)
  - M3D-C[1] (FE-based MHD)
  - Nektar++ (High-order FE flow)
  - Albany/Trilinos (Solid mechanics FE)

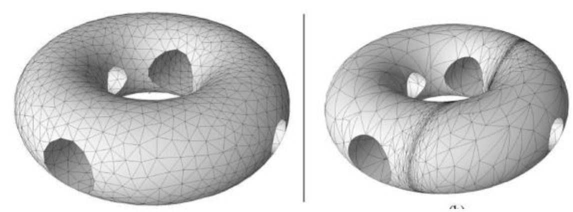




Application on multi-billion element meshes

Mesh adaptation for evolving features





Anisotropic adaptation for curved meshes

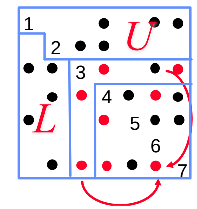
Source Code: http://github.com/SCOREC/core
Paper: www.scorec.rpi.edu/REPORTS/2014-9.pdf

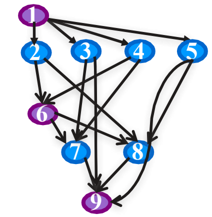
Aug 2017

SuperLU

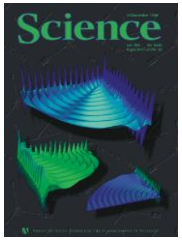
Supernodal sparse LU direct solver. Unique user-friendly interfaces. Flexible software design. Used in a variety of applications. Freely available.

- **Capabilities**
  - Serial (thread-safe), shared-memory (SuperLU_MT, OpenMP or Pthreads), distributed-memory (SuperLU_DIST, hybrid MPI+ OpenM + CUDA).
    - Implemented in C, with Fortran interface
  - Sparse LU decomposition, triangular solution with multiple right-hand sides
  - Incomplete LU (ILU) preconditioner in serial SuperLU
  - Sparsity-preserving ordering:
    - Minimum degree ordering applied to A^TA or A^T+A
    - Nested dissection ordering applied to A^TA or A^T+A [(Par)METIS, (PT)-Scotch]
  - User-controllable pivoting: partial pivoting, threshold pivoting, static pivoting
  - Condition number estimation, iterative refinement.
  - Componentwise error bounds
- **Open source software**
  - Used worldwide in a vast range of applications
  - Can be used through PETSc and Trilinos
  - Available on github





ITER tokamak



quantum mechanics

Widely used in commercial software, including AMD (circuit simulation), Boeing (aircraft design), Chevron, ExxonMobile (geology), Cray's LibSci, FEMLAB, HP's MathLib, IMSL, NAG, SciPy, OptimaNumerics, Walt Disney Animation.

http://crd-legacy.lbl.gov/~xiaoye/SuperLU

Aug 2017

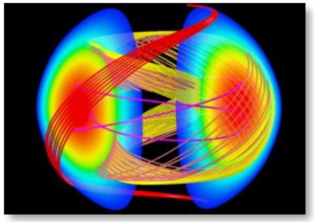
SUNDIALS

SUite of Nonlinear DIfferential /ALgebraic equation Solvers

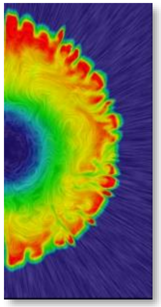
Adaptive time integrators for ODEs and DAEs and efficient nonlinear solvers. Used in a variety of applications. Freely available. Encapsulated parallelism.

- **ODE integrators:**
  - CVODE(S): variable order and step stiff BDF and non-stiff Adams with forward and adjoint sensitivity analysis
  - ARKode: variable step implicit, explicit, and additive IMEX Runge-Kutta
- **DAE integrators**: IDA/IDA(S) - variable order and step stiff BDF integrators with forward and adjoint sensitivity analysis
- **Nonlinear Solver:** KINSOL Newton-Krylov, Picard, and accelerated fixed point
- **Design**
  - Written in C with interfaces to Fortran
  - Modular design: users can supply own data structures
- **Open Source Software**
  - CMake-based portable build system
  - Freely available (BSD license); > 11K downloads/year
  - Active user community & sundials-users email list

Lawrence Livermore National Laboratory



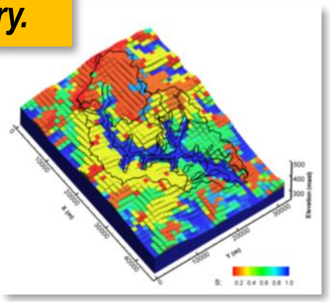
Magnetic reconnection



Core collapse supernova

SUNDIALS has been used worldwide in applications from research and industry.

Dislocation dynamics



Subsurface flow

FASTMATH ECP xSDK

http://www.llnl.gov/CASC/sundials

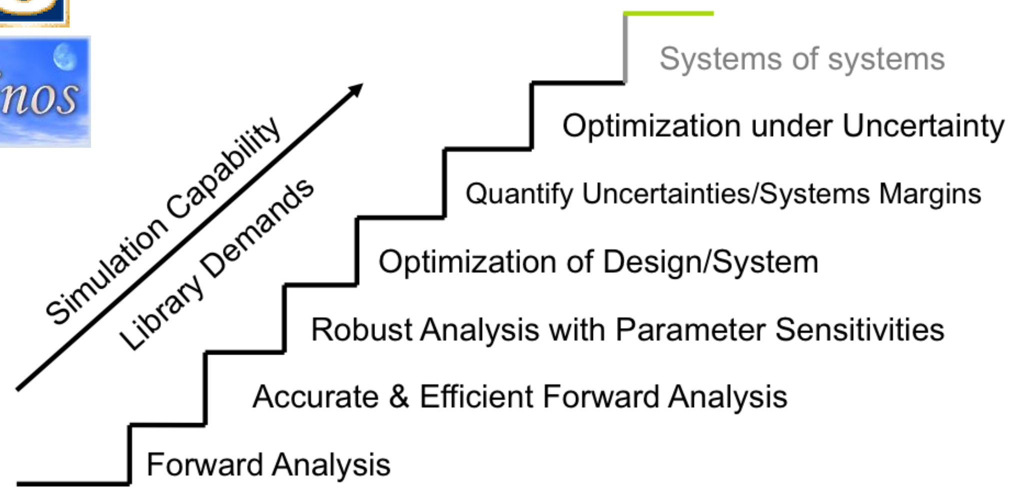
Aug 2017

Trilinos

Optimal kernels to optimal solutions. Over 60 packages. Laptops to leadership systems. Next-gen systems, multiscale/multiphysics, large-scale graph analysis.

- **Optimal kernels to optimal solutions**
  - Geometry, meshing
  - Discretization, load balancing
  - Scalable linear, nonlinear, eigen, transient, optimization, UQ solvers
  - Scalable I/O, GPU, manycore
- **60+ packages**
  - Other distributions: Cray LIBSCI, Github repo
  - Thousands of users, worldwide distribution
  - Laptops to leadership systems

Transforming Computational Analysis To Support High Consequence Decisions



Each stage requires *greater performance* and *error control* of prior stages:
Always will need: more accurate and scalable methods.
more sophisticated tools.

https://trilinos.org

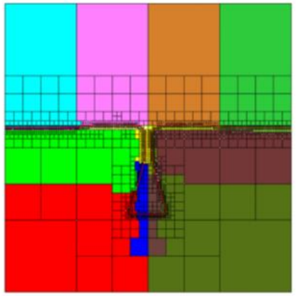
Aug 2017

Zoltan/Zoltan2

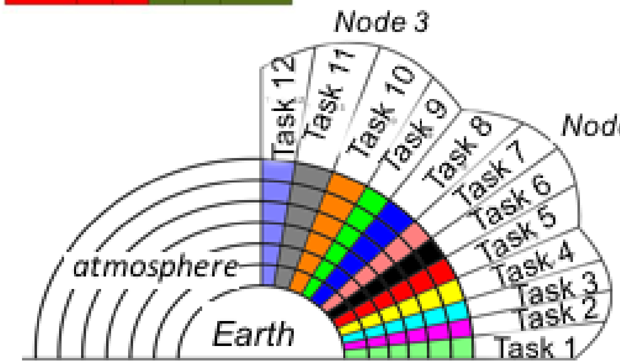
Parallel partitioning, load balancing, task placement, graph coloring, matrix ordering, unstructured communication utilities, distributed directories.

- **Suite of partitioning/load-balancing methods to support many applications**
  - Fast geometric methods maintain spatial locality of data (e.g., for adaptive finite element methods, particle methods, crash/contact simulations)
  - Graph and hypergraph methods explicitly account for communication costs (e.g., for electrical circuits, finite element meshes, social networks).
  - Includes single interface to popular partitioning TPLs: XtraPuLP (SNL, RPI); ParMA (RPI);
PT-Scotch (U Bordeaux); ParMETIS (U Minnesota)
- **Architecture-aware MPI task placement**
  - Places interdependent MPI tasks on "nearby" nodes in computing architecture
  - Reduces communication time and network congestion
- **Use as a stand-alone library or as a Trilinos component**

Sandia National Laboratories FASTMATH ASC



Zoltan's fast, geometric partitioner redistributes work to maintain load balance in a surface deposition simulation with adaptive meshing



Zoltan2's task placement reduced communication time in ACME HOMME (atmospheric modeling) by up to 31% on 16K cores of IBM BG/Q

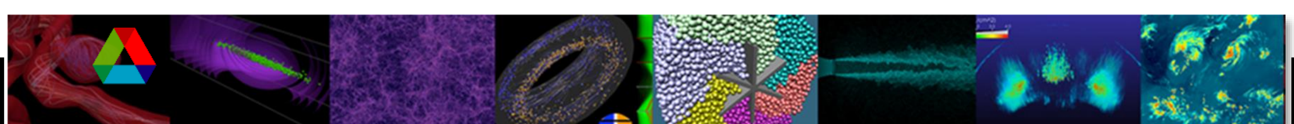
http://www.cs.sandia.gov/Zoltan

Aug 2017

Track 4: Numerical Algorithms and Software for Extreme-Scale Science

MONDAY, August 7

Time	Title of presentation	Lecturer
9:30 am	Numerical Software: Foundational Tools for HPC Simulations	Lori Diachin, LLNL
	… with hands-on sessions throughout the day for various topics	
11:00 am	Structured Mesh Technologies	Ann Almgren, LBNL
11:45 am	Unstructured Mesh Technologies	Tzanio Kolev, LLNL and Mark Shephard, RPI
12:30 pm	Lunch	
1:30 pm	Panel: Heterogeneity and Performance Portability	Mark Miller, LLNL (Moderator)
2:15 pm	Time Integration	Carol Woodward, LLNL
3:00 pm	Nonlinear Solvers and Krylov Methods	Barry Smith, ANL
3:35 pm	Break	
4:05 pm	Sparse Direct Solvers	Sherry Li, LBNL
4:35 pm	Algebraic Multigrid	Ulrike Yang, LLNL
5:05 pm	Introducing the xSDK and Spack	Lois Curfman McInnes and Barry Smith, ANL



+ Hands-on

Track 4: Numerical Algorithms and Software for Extreme-Scale Science

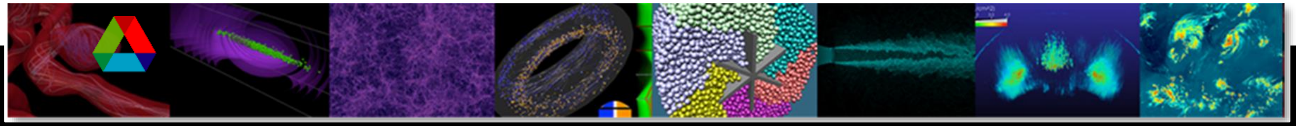
MONDAY, August 7

Time	Title of presentation	Lecturer
5:30 pm	Dinner + Panel: Extreme-Scale Algorithms and Software	Mark Miller, LLNL (Moderator)
6:30 pm	Conforming and Nonconforming Adaptivity for Unstructured Meshes	Tzanio Kolev, LLNL and Mark Shephard, RPI
7:00 pm	Open hands-on time	All
7:30 pm	Enabling Optimization Using Adjoint Software	Hong Zhang, ANL
8:00 pm	Open hands-on time	All
8:30 pm	One-on-one discussions with ATPESC participants	
9:30 pm	Adjourn	

Hands-on Lead: Mark Miller (LLNL)

Additional contributors to lectures and hands-on lessons:
Satish Balay (ANL), Aaron Fisher (LLNL), David Gardner (LLNL), Lois Curfman McInnes (ANL)

Additional contributors to Gallery of Highlights:
Karen Devine (SNL), Mike Heroux (SNL), Dan Martin (LBNL)



+ Hands-on

Sign up for 1-on-1 discussions with numerical software developers

Via Google docs folder: See link in email:

- **Your name, institution, email address**
  - **Topical interests**
  - **Pointers to other relevant info**

Meeting opportunities include:

- Today, 8:30-9:30 pm
- Other days/times, opportunities for communication with developers who are not attending today

HandsOnLessons

HandsOnLessons

A github pages site hosting hands-on lessons in the use, design and development of scientific computing software packages

View on GitHub

Welcome to HandsOnLessons

Hosted here are a series of increasingly sophisticated *hands-on* lessons aimed at helping users of all experience levels learn to use a variety of scientific software packages for solving complex numerical problems. We begin with custom, hand-coded solutions to the homogeneous, one-dimensional heat equation to demonstrate basic numerical and performance issues such as accuracy, stability, time to solution, memory, and flops required, along with motivation for the use of numerical software packages to help achieve more robust, efficient, scalable, extensible, and portable software.

Go to the Lessons

- Basic, One-Dimensional Heat Equation
- Structured Meshes
- Time Integrators
- Iterative Solvers
- Sparse Direct Solvers
- Algebraic Multigrid
- Finite Elements Convergence
- Adjoint Solvers

And more lessons to come

Github pages site:

https://xsdk-project.github.io/HandsOnLessons

Auspices and Disclaimer

Support for this work was provided through Scientific Discovery through Advanced Computing (SciDAC) program and the Exascale Computing Project funded by U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research

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