Exceptional service in the national interest

Trilinos Progress, Challenges and Future Plans

Michael A. Heroux

Sandia National Laboratories

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000. SAND NO. 2011-XXXXP

What is Trilinos?

- Object-oriented software framework for…
- Solving big complex science & engineering problems
- A Project of Projects.
- More like LEGO™ bricks than Matlab™

BACKGROUND/MOTIVATION

Trilinos

R&D 100

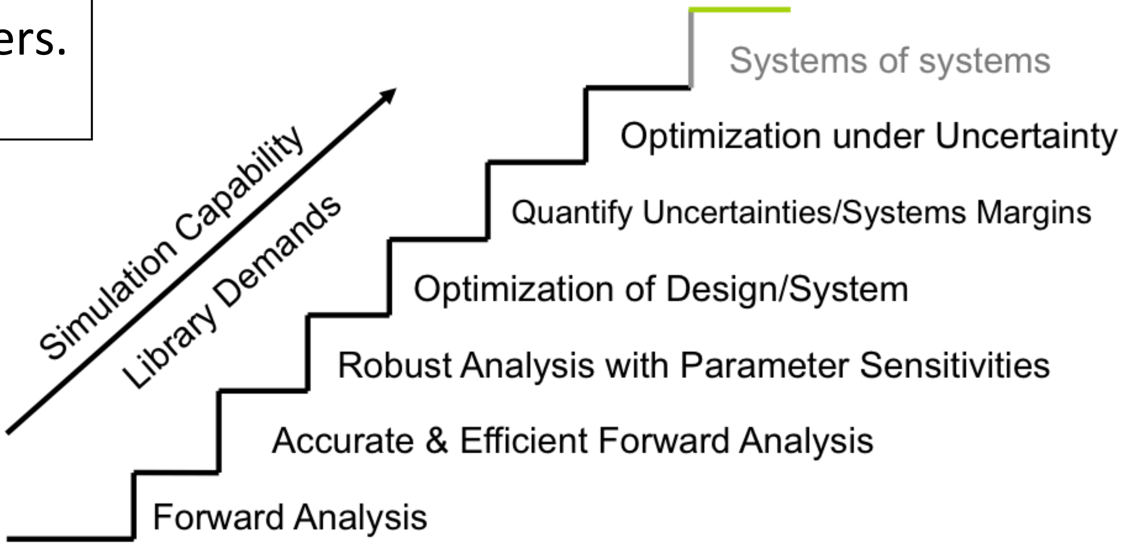
Laptops to

Leadership systems

Optimal Kernels to Optimal Solutions:

- Geometry, Meshing
- Discretizations, Load Balancing.
- Scalable Linear, Nonlinear, Eigen, Transient, Optimization, UQ solvers.
- Scalable I/O, GPU, Manycore

Transforming Computational Analysis To Support High Consequence Decisions



Simulation Capability
Library Demands
Systems of systems
Optimization under Uncertainty
Quantify Uncertainties/Systems Margins
Optimization of Design/System
Robust Analysis with Parameter Sensitivities
Accurate & Efficient Forward Analysis
Forward Analysis

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

- 60 Packages.
- Other distributions:
  - Cray LIBSCI
  - Public repo.

Applications

- All kinds of physical simulations:
  - Structural mechanics (statics and dynamics)
  - Circuit simulations (physical models)
  - Electromagnetics, plasmas, and superconductors
  - Combustion and fluid flow (at macro- and nanoscales)
- Coupled / multiphysics models
- Data and graph analysis (2D distributions).

Trilinos Strategic Goals

- **Scalable Computations**: As problem size and processor counts increase, the cost of the computation will remain nearly fixed.
- **Hardened Computations**: Never fail unless problem essentially intractable, in which case we diagnose and inform the user why the problem fails and provide a reliable measure of error.
- **Full Vertical Coverage**: Provide leading edge enabling technologies through the entire technical application software stack: from problem construction, solution, analysis and optimization.

Algorithmic Goals

- **Universal Interoperability:** All Trilinos packages, and important external packages, will be interoperable, so that any combination of packages and external software (e.g., PETSc, Hypre) that makes sense algorithmically will be possible within Trilinos.
- **Universal Accessibility**: All Trilinos capabilities will be available to users of major computing environments: C++, Fortran, Python and the Web, and from the desktop to the latest scalable systems.
- **TriBITS Lifecycle**: Trilinos will be:
  - Reliable: Leading edge hardened, scalable solutions for each of these applications
  - Available: Integrated into every major application at Sandia
  - Serviceable: “Self-sustaining”.

Software Goals

Capability Leaders:
Layer of Proactive Leadership

- Areas:
    - User Experience (W. Spotz).
    - Scalable I/O: (J. Lofstead).
    - Framework, Tools & Interfaces (J. Willenbring).
    - Software Engineering Technologies and Integration (R. Bartlett).
    - Discretizations (P. Bochev).
    - Geometry, Meshing & Load Balancing (K. Devine).
    - Scalable Linear Algebra (M. Heroux).
    - Linear & Eigen Solvers (J. Hu).
    - Nonlinear, Transient & Optimization Solvers (A. Salinger).
- Each leader provides strategic direction across all Trilinos packages within area.

Unique features of Trilinos

- Huge library of algorithms
  - Linear and nonlinear solvers, preconditioners, …
  - Optimization, transients, sensitivities, uncertainty, …
- Growing support for multicore & hybrid CPU/GPU
  - Built into the new Tpetra linear algebra objects
  - Unified intranode programming model
  - Spreading into the whole stack:
    - Multigrid, sparse factorizations, element assembly…
- Growing support for mixed and arbitrary precisions
  - Don’t have to rebuild Trilinos to use it.
- Growing support for huge (> 2B unknowns) problems.

Trilinos’ software organization

Trilinos is made of packages

- Not a monolithic piece of software
  - Like LEGO™ bricks, not Matlab™
- Each package:
  - Has its own development team and management
  - Makes its own decisions about algorithms, coding style, etc.
  - May or may not depend on other Trilinos packages
- Trilinos is not “indivisible”
  - You don’t need all of Trilinos to get things done
  - Any subset of packages can be combined and distributed
  - Current public release (11.2) contains 54 of the 60+ Trilinos packages
- Trilinos top layer framework
  - Not a large amount of source code: ~1.5%
  - Manages package dependencies
    - Like a GNU/Linux package manager
  - Runs packages’ tests nightly, and on every check-in
- Package model supports multifrontal development

Trilinos Package Summary

	Objective	Package(s)
Discretizations	Meshing & Discretizations	STK, Intrepid, Pamgen, Sundance, ITAPS, Mesquite
	Time Integration	Rythmos
Methods	Automatic Differentiation	Sacado
	Mortar Methods	Moertel
Services	Linear algebra objects	Epetra, Jpetra, Tpetra, Kokkos
	Interfaces	Thyra, Stratimikos, RTOp, FEI, Shards
	Load Balancing	Zoltan, Isorropia, Zoltan2
	“Skins”	PyTrilinos, WebTrilinos, ForTrilinos, Ctrilinos, Optika
	C++ utilities, I/O, thread API	Teuchos, EpetraExt, Kokkos, Triutils, ThreadPool, Phalanx, Trios
Solvers	Iterative linear solvers	AztecOO, Belos, Komplex
	Direct sparse linear solvers	Amesos, Amesos2
	Direct dense linear solvers	Epetra, Teuchos, Pliris
	Iterative eigenvalue solvers	Anasazi, Rbgen
	ILU-type preconditioners	AztecOO, IFPACK, Ifpack2
	Multilevel preconditioners	ML, CLAPS, Muelu
	Block preconditioners	Meros, Teko
	Nonlinear system solvers	NOX, LOCA, Piro
	Optimization (SAND)	MOOCHO, Aristos, TriKota, Globipack, Optipack, ROL
	Stochastic PDEs	Stokhos

Interoperability vs. Dependence

("Can Use") ("Depends On")

- Although most Trilinos packages have no explicit dependence, often packages must interact with *some* other packages:
  - NOX needs operator, vector and linear solver objects.
  - AztecOO needs preconditioner, matrix, operator and vector objects.
  - Interoperability is enabled at configure time.
  - Trilinos **`cmake`** system is vehicle for:
    - Establishing interoperability of Trilinos components…
    - Without compromising individual package autonomy.
- Architecture supports simultaneous development on many fronts.

Trilinos Presentation Forums

- Supercomputing 2014 Tutorial
  - Sunday, Nov 16, 2014
  - Linear Algebra Libraries for High-Performance Computing: Scientific Computing with Multicore and Accelerators
  - With Jack Dongarra, Jim Demmel, Jacub Kurzak
- EuroTUG:
  - March 2 – 4, 2015, CEA, Paris.
- Next Trilinos User Group Meeting:
  - Oct 26-29, 2015.

Themes for FY13/14

- trilinos.org becomes real.
  - We have a machine and sound legal foundation.
  - Community clone (out for some time, preferred model for many).
  - Developer agreements (based on OpenMPI/Apache) in the works.
- Execution of plans for "Extreme Scale" computing:
  - Multicore/Manycore for desktop/deskside.
  - Scalable multicore/manycore for high end.
  - Delivering the Tpetra/Kokkos stack to mainstream users.
- Expanding the Trilinos Developer Community.
- Toward completion and adoption of a lifecycle model.
  - TriBITS Lifecycle Model 1.0 completed.
  - Discussions on Thursday.
- Resilience: Start of a decade-long effort.
- Usability: Making Trilinos more accessible.

Themes for FY14/15

- Scalable Algorithms & Implementations:
  - Partitioning & Load Balancing.
  - Thread scalability.
- Productivity Focus:
  - IDEAS Project. More later.
- xSDK:
  - IDEAS extreme-scale scientific software ecosystem.
- Trilinos Community 2.0:
  - Revisiting the Trilinos value proposition.
- Resilience: Continuing a decade-long effort:
  - Algorithms, software.
- Preparations for task-centric/dataflow applications:
  - Application-library APIs for a task-centric app architecture?
  - Execution models.

Trilinos Availability/Information

- Trilinos and related packages:
  - Available via LGPL or BSD.
- Current release is 11.12.
- Preparing for 12.0 – April.
  - (backward compatibility break)
- Unlimited availability.
- More information:
  - http://trilinos.org

Trilinos Community 2.0

- GitHub, Atlassian:
  - Open source SW development, tools platforms.
  - Workflows for high-quality community SW product development.
- Trilinos value proposition:
  - Included these same things, but must re-evaluate.
  - Must address packages that want GitHub presence.
  - Must (IMO) move Trilinos itself to GitHub.
- New types of Trilinos packages (evolving):
  - Internal: Available only with Trilinos (traditional definition).
  - Exported: Developed in Trilinos repository, available externally.
  - Imported: Developed outside of Trilinos, available internally.

Trilinos Community 2.0

- Case studies:
  - TriBITS: Was an internal package, now external.
  - DTK: Has always been external.
  - KokkosCore: Is internal. Needs to be available externally.
- Issues to Resolve:
  - Package inclusion policies: Define for each package type.
  - Quality criteria: Contract between Trilinos and packages.
  - Workflows: Development, testing, documentation, etc.
  - Trilinos on GitHub: Evaluate.
  - Trilinos Value Proposition: Re-articulate Trilinos Strategic Goals implications.

THE HPC ECOSYSTEM & TRILINOS ACTIVITIES

Three Parallel Computing Design Points

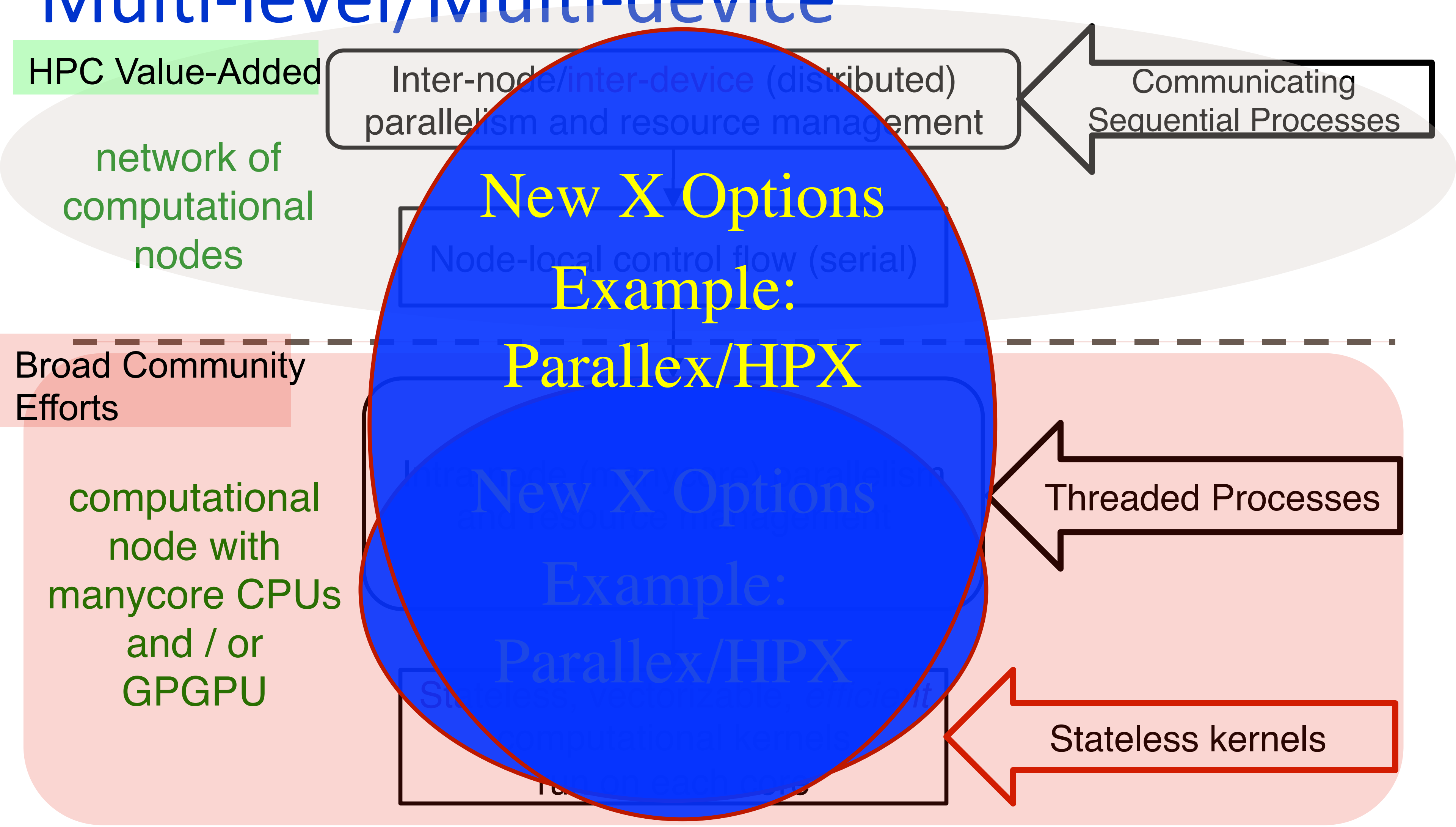
- Terascale Laptop: Uninode-Manycore
- Petascale Deskside: Multinode-Manycore
- Exascale Center: Manynode-Manycore

Common Element

Goal: Make
Petascale = Terascale + more
Exascale = Petascale + more

Most applications will not adopt an exascale programming strategy that is incompatible with tera and peta scale.

SPMD+X Parallel Programming Model: Multi-level/Multi-device


HPC Value-Added
Inter-node/inter-device (distributed) parallelism and resource management
Communicating Sequential Processes
network of computational nodes
Node-local control flow (serial)
Broad Community Efforts
computational node with manycore CPUs and / or GPGPU
Threaded Processes
Stateless kernels

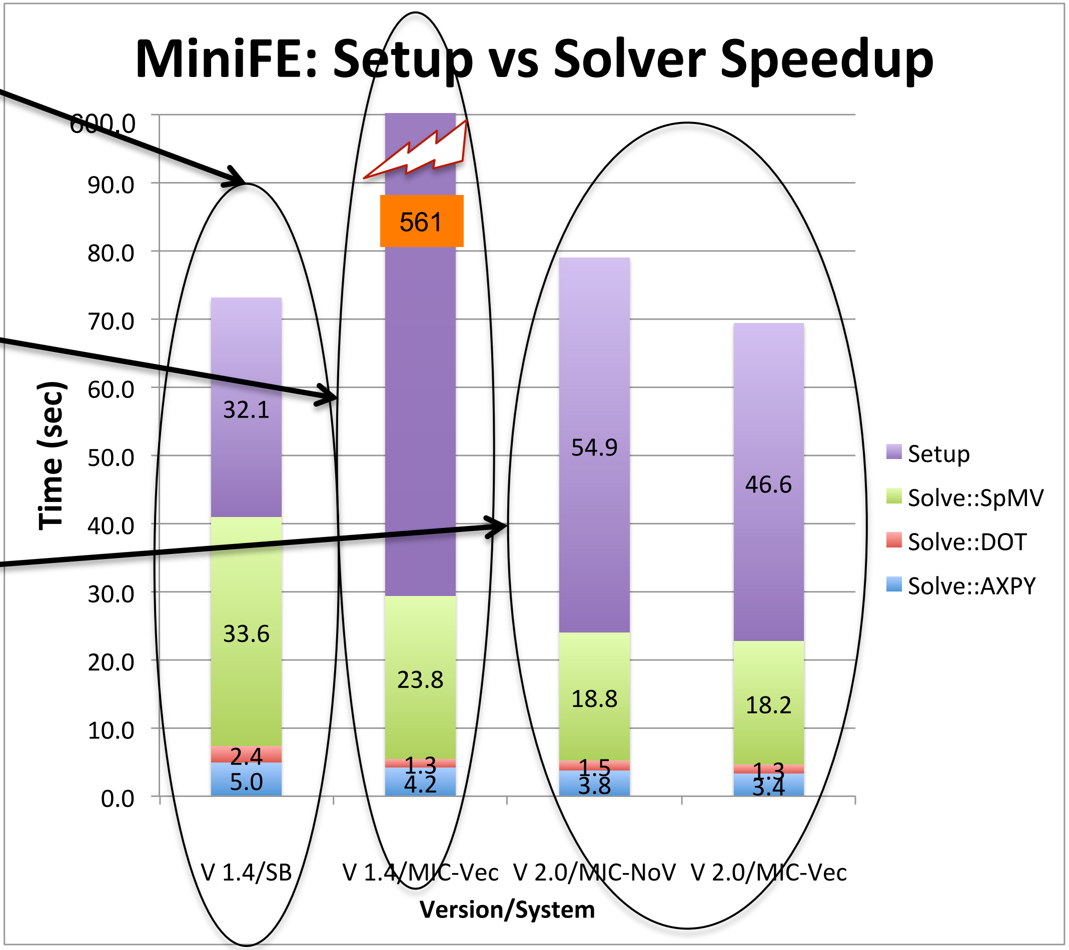
New X Options Example: Parallex/HPX

New X Options Example: Parallex/HPX

The work ahead of us: Threads and vectors

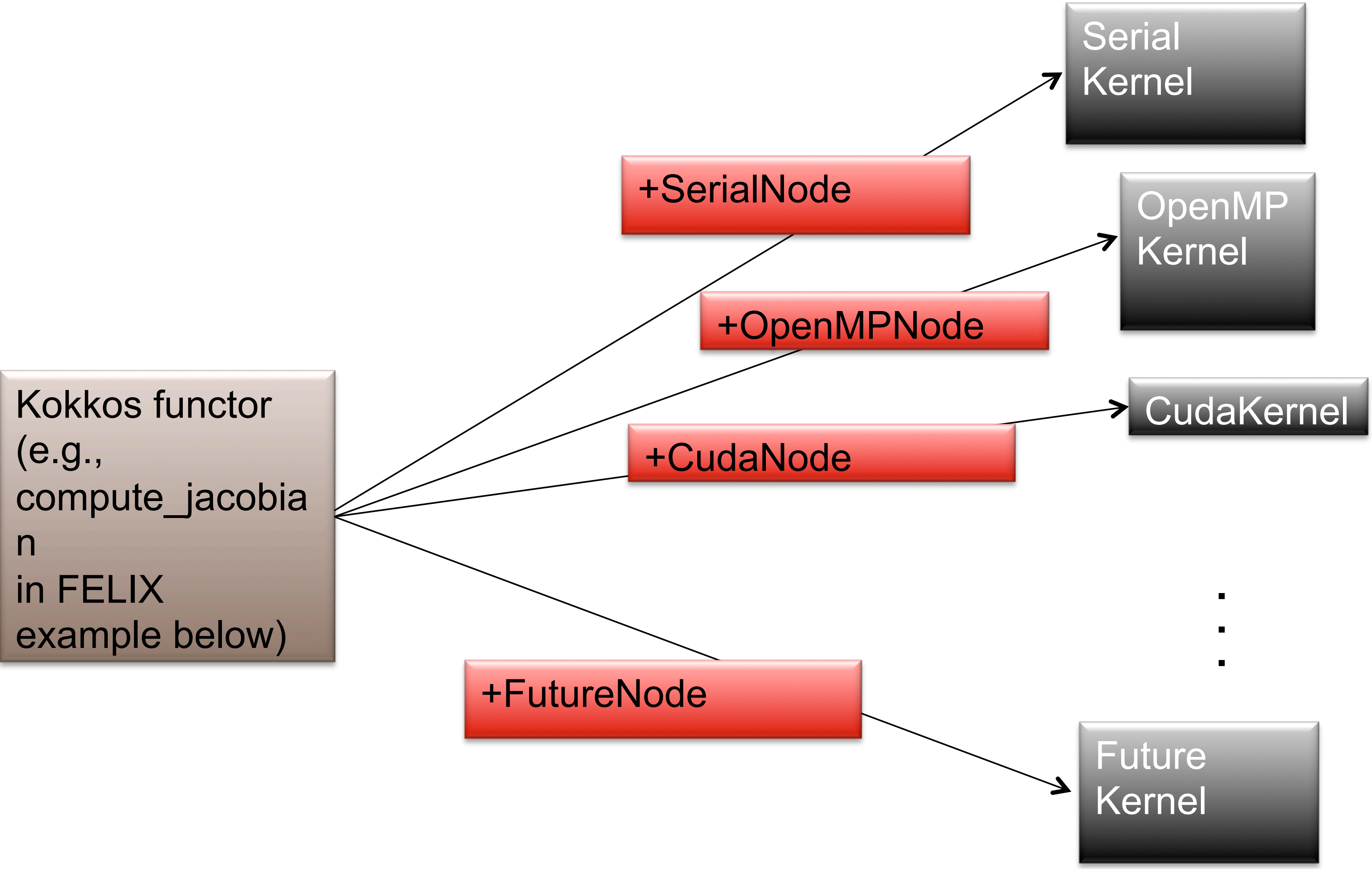
MiniFE 1.4 vs 2.0 as Harbingers

- Typical MPI-only run:
  - Balanced setup vs solve
- First MIC run:
  - Thread/vector solver
  - No-thread setup
- V 2.0: Thread/vector
  - Lots of work:
    - Data placement, const /restrict declarations, avoid shared writes, find race conditions, …
  - Unique to each app



WITH C++ AS YOUR HAMMER, EVERYTHING LOOKS LIKE YOUR THUMB.

Compile-time Polymorphism



Kokkos functor (e.g., compute_jacobia n in FELIX example below)

+SerialNode

Serial Kernel

+OpenMPNode

OpenMP Kernel

+CudaNode

CudaKernel

⋮

+FutureNode

Future Kernel

Struct-of-Arrays vs. Array-of-Structs

A False Dilemma

Multi-dimensional Dense Arrays

- Many computations work on data stored in multi-dimensional arrays:
  - Finite differences, volumes, elements.
  - Sparse iterative solvers.
- Dimension are (k,l,m,…) where one dimension is long:
  - A(3,1000000)
  - 3 degrees of freedom (DOFs) on 1 million mesh nodes.
- A classic data structure issue is:
  - Order by DOF: A(1,1), A(2,1), A(3,1); A(1,2) … or
  - By node: A(1,1), A(1,2), …
- Adherence to raw language arrays forces a choice.

Kokkos_functor example: compute_jacobian

```
for(int cell = 0; cell < worksetNumCells; cell++) {
    for(int qp = 0; qp < numQPs; qp++) {
      for(int row = 0; row < numDims; row++){
        for(int col = 0; col < numDims; col++){
          for(int node = 0; node < numNodes; node++){
             jacobian(cell, qp, row, col) +=
                         coordVec(cell, node, row)
                        *basisGrads(node, qp, col);
           } // node
         } // col
      } // row
    } // qp
  } // cell
```

```
Kokkos::parallel_for ( worksetNumCells,
 compute_jacobian<ScalarT, Device, numQPs, numDims,
numNodes> (basisGrads, jacobian, coordVec));
```

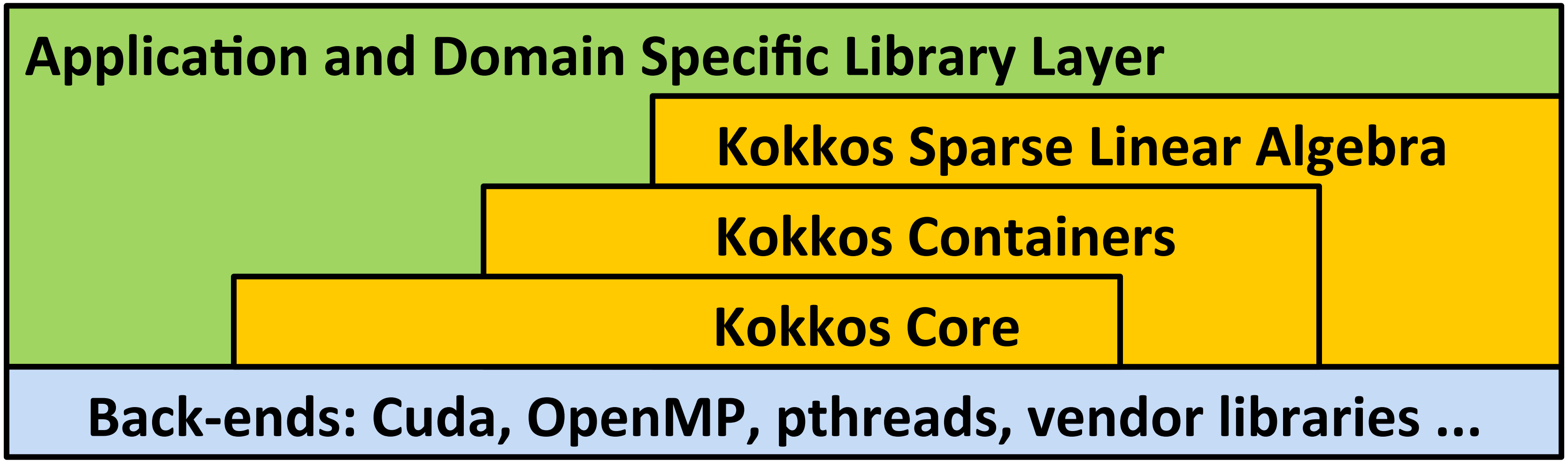
```
template < typename ScalarType, clas DeviceType, int numQPs_,
                              int numDims_, int numNodes_ >
class compute_jacobian  {
  Array3 basisGrads_;
  Array4 jacobian_;
  Array3_const coordVec_;
 public:
  typedef DeviceType device_type;
  compute_jacobian(Array3 &basisGrads, Array4 &jacobian,
        Array3 &coordVec)
   : basisGrads_(basisGrads)
   , jacobian_(jacobian)
   , coordVec_(coordVec){}

KOKKOS_INLINE_FUNCTION
 void operator () (const std::size_t i) const
 {
  for(int qp = 0; qp < numQPs_; qp++) {
      for(int row = 0; row < numDims_; row++){
        for(int col = 0; col < numDims_; col++){
          for(int node = 0; node < numNodes_; node++){
             jacobian_(i, qp, row, col) += coordVec_(i, node, row)
                                             *basisGrads_(node, qp, col);
           } // node
         } // col
      } // row
    } // qp
 }
 };
```

Kokkos Key Features

- Multi-dimensional array containers:
  - Index space abstraction:
    - Physics (i,j,k) does not dictate storage (i,j,k).
  - Physical layout determined by:
    - Memory placement policy.
    - At compile time.
- Parallel patterns execution framework:
  - parallel_for/reduce/scan – Similar to TBB.
  - Task-DAG – under development.
- Goals:
  - Provide portability layer for apps and libraries.
  - Provide prototypes for eventual C++ standards.

Kokkos: A Layered Collection of Libraries

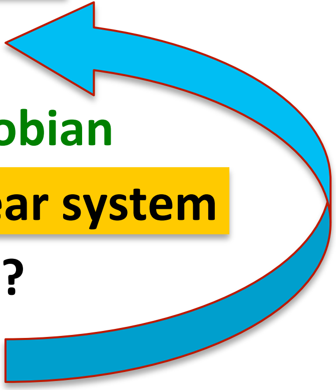
- Applications and Domain Libraries written in Standard C++
  - *Not* a language extension like OpenMP, OpenACC, OpenCL, CUDA, ...
  - Require C++1998 standard (supported everywhere except IBM's xlC)
  - Prefer C++2011 for its concise lambda syntax (LLNL's RAJA requires this)
    - As soon as vendors catch up to C++2011 language compliance


Application and Domain Specific Library Layer
Kokkos Sparse Linear Algebra
Kokkos Containers
Kokkos Core
Back-ends: Cuda, OpenMP, pthreads, vendor libraries ...

- Kokkos implemented with C++ template meta-programming
  - In *spirit* of TBB, Thrust & CUSP, C++AMP, LLNL's RAJA, ...

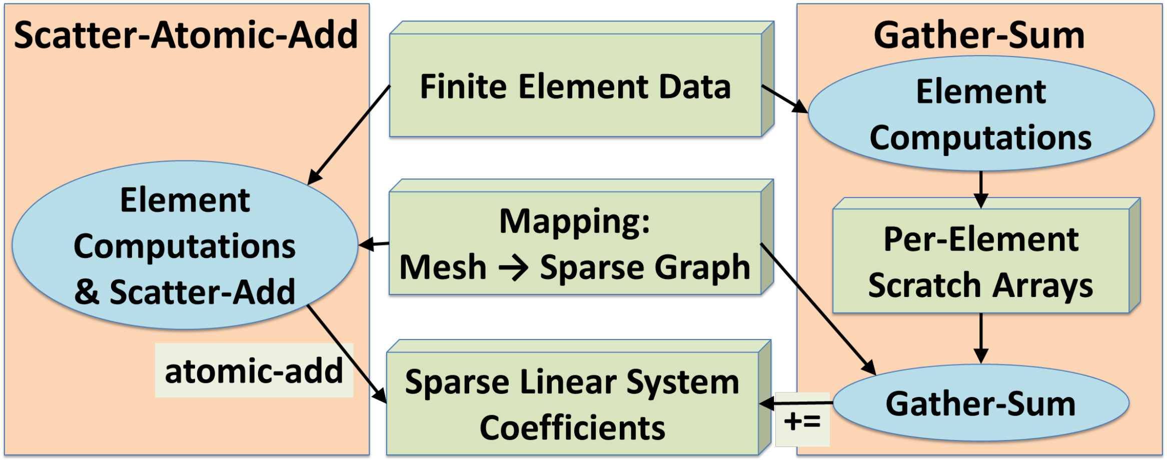
MiniFENL Proxy Application

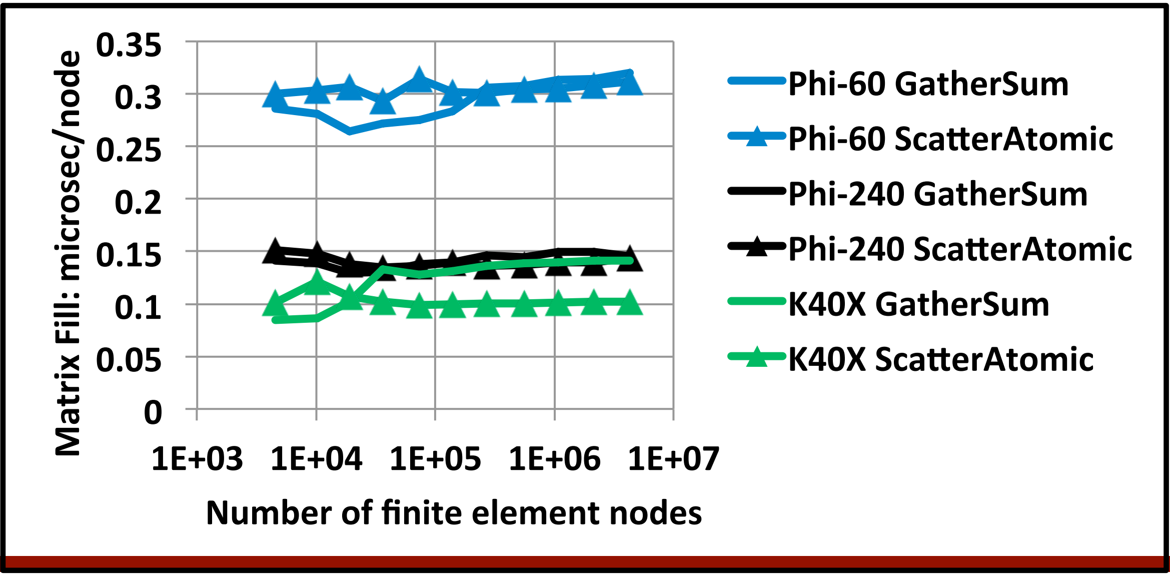
- Solve nonlinear finite element problem via Newton iteration
  - Focus on construction and fill of sparse linear system
  - Thread safe, thread scalable, and performant algorithms
  - Evaluate thread-parallel capabilities and programming models
- Construct sparse linear system graph and coefficient arrays
  - Map finite element mesh connectivity to degree of freedom graph
  - Thread-scalable algorithm for graph construction
- Compute nonlinear residual and Jacobian
  - Thread-parallel finite element residual and Jacobian
  - Atomic-add to fill element coefficients into linear system
    - Atomic-add for thread safety, performance?
- Solve linear system for Newton iteration



Thread-Scalable Fill of Sparse Linear System

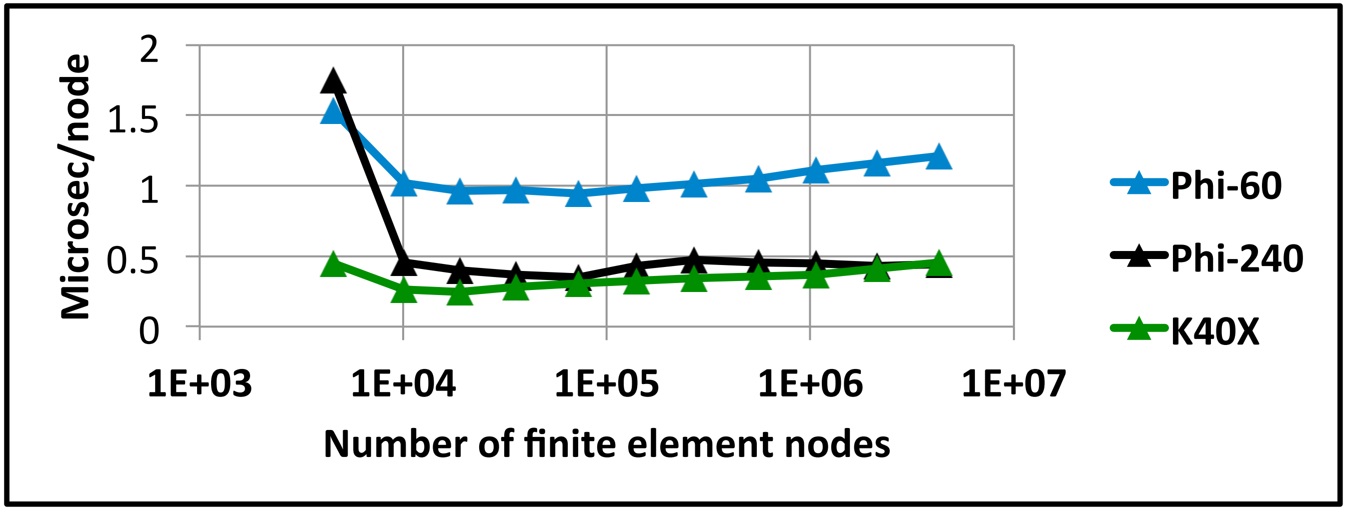
- MiniFENL: Newton iteration of FEM:
- Fill sparse matrix via Scatter-Atomic-Add or Gather-Sum ?
- Scatter-Atomic-Add
  - + Simpler
  - + Less memory
  - – Slower HW atomic
- Gather-Sum
  - + Bit-wise reproducibility
- Performance win?
  - Scatter-atomic-add
  - ~equal Xeon PHI
  - 40% faster Kepler GPU
- ✓ Pattern chosen
  - Feedback to HW vendors: performant atomics


Scatter-Atomic-Add
Finite Element Data
Gather-Sum
Element Computations
Element Computations & Scatter-Add
Mapping: Mesh → Sparse Graph
Per-Element Scratch Arrays
atomic-add
Sparse Linear System Coefficients
+=
Gather-Sum



Thread-Scalable Sparse Matrix Construction

- MiniFENL: Construct sparse matrix graph from FEM connectivity
- Thread scalable algorithm for constructing a data structure
  1. Parallel-for : fill Kokkos lock-free unordered map with FEM node-node pairs
  2. Parallel-scan : sparse matrix rows' column counts into row offsets
  3. Parallel-for : query unordered map to fill sparse matrix column-index array
  4. Parallel-for : sort rows' column-index subarray



- Pattern and tools generally applicable to construction and dynamic modification of data structures

Porting in Progress: Trilinos

- Trilinos : SNL's suite of equation solver libraries (and others)
  - Currently MPI-only parallel
  - Incremental refactoring to MPI+Kokkos parallel
- Tpetra : Trilinos' core parallel sparse linear algebra library
  - Vectors, multi-vectors, sparse matrices, parallel data distribution maps
  - Fundamental operations: axpy, dot, matrix-vector multiply, ...
  - Templated on "scalar" type: float, double, automatic differentiation (AD), embedded uncertainty quantification (UQ), ...
- Port Tpetra to MPI+Kokkos, other libraries follow
  - On schedule to complete in Spring 2015
  - Use of NVIDIA's unified virtual memory (UVM) expedited porting effort
- Embedded UQ already Kokkos-enabled through LDRD
  - Greater computational intensity leads to significant speed-ups compared to non-embedded UQ sampling algorithms

Node API Futures: MPI Analogy

- ls Trilinos/packages/aztecoo/src/md_wrap_*

```
md_wrap_intel_c.c md_wrap_ncube_c.c md_wrap_scalar_c.c
md_wrap_mpi_c.c   md_wrap_puma_c.c  md_wrap_sp2_c.c
```

- Excerpt from `md_wrap_sp2_c.c` (next slide).

```
int md_write(char *buf, int bytes, int dest, int type, int *flag)
{
  int err, buffer;

  if (bytes == 0) {
      err = mpc_bsend(&buffer, 1, dest, type);
  } else {
      err = mpc_bsend(buf, bytes, dest, type);
  }
 if (err!=0) (void) fprintf(stderr, "mpc_bsend error = %d\n", mperrno);
return 0;
}
```

Original md_write function (prior to MPI)

```
int md_write(char *buf, int bytes, int dest, int type, int *flag)
{
#if defined (MPL)
  int err, buffer;

  if (bytes == 0) {
      err = mpc_bsend(&buffer, 1, dest, type);
  } else {
      err = mpc_bsend(buf, bytes, dest, type);
  }
 if (err!=0) (void) fprintf(stderr, "mpc_bsend error = %d\n", mperrno);
#elif defined (MPI)
  int err, buffer;
  if (bytes == 0) {
      err = MPI_Send(&buffer, 1, MPI_BYTE, dest, type, MPI_COMM_WORLD);
  } else {
      err = MPI_Send(buf, bytes, MPI_BYTE, dest, type, MPI_COMM_WORLD);
  }
  if (err != 0) (void) fprintf(stderr, "MPI_Send error = %d\n", err);
#endif
  return 0;
}
```

All node APIs are transitional…

- Except the one(s) that becomes the standard.
- Recommended activities:
  - Write a light-weight API:
    - Compile-time polymorphism.
    - Match your needs and nothing more.
    - Wrap other functionality: CUDA, OpenMP, ~~OpenCL~~, pthreads, …
    - Don't fall in love with your implementation!
    - Examples: OCCA, Mint, Kokkos, RAJA, and more.
  - Participate in standards committees.
  - Prepare for eventual replacement by a standard.

Non-incremental Efforts

- Use of Futures:
  - Exploit previously inaccessible, fine-grain dynamic parallelism.
  - Natural framework for expressing data-driven parallelism.
- Better than MPI:
  - Beyond functional mimic of MPI.
  - AGAS: Truly adaptive mesh refinement.
- Overarching goal: Show that non-incremental approaches
  - Work.
  - Superior to MPI+X in one or more metric:
    - Performance: Extracting latent parallelism.
    - Portability: Performance obtained from system’s underlying runtime.
    - Productivity: Easier to write, understand, maintain.
- More in a minute.

RESILIENT COMPUTING & TRILINOS

Four Resilient Programming Models

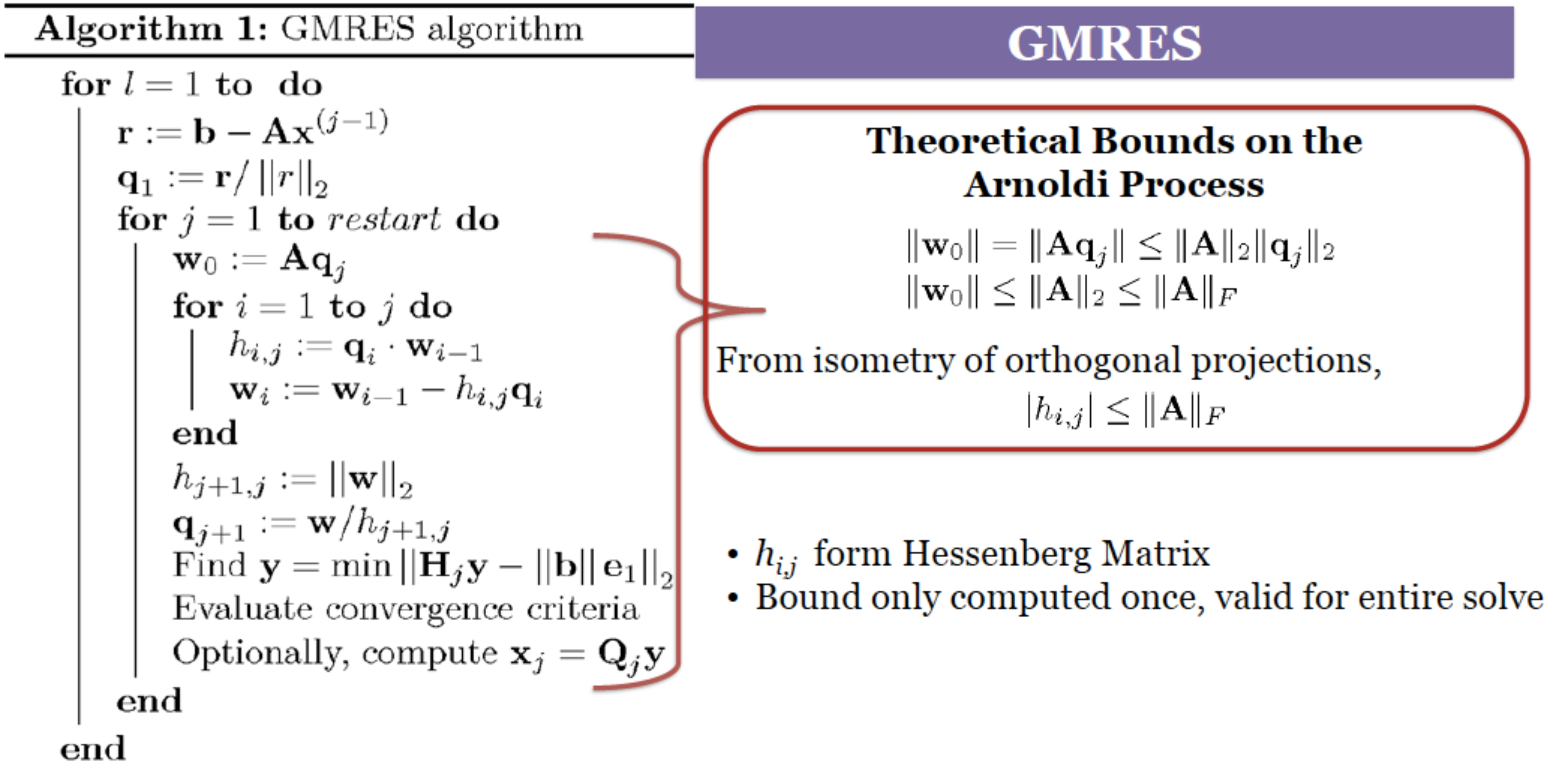
- Skeptical Programming. (SP)
- Relaxed Bulk Synchronous (rBSP)
- Local-Failure, Local-Recovery (LFLR)
- Selective (Un)reliability (SU/R)

Toward Resilient Algorithms and Applications
Michael A. Heroux arXiv:1402.3809v2 [cs.MS]

Skeptical Programming

I might not have a reliable digital machine

- Expect rare faulty computations
- Use analysis to derive cheap "detectors" to filter large errors
- Use numerical methods that can absorb *bounded error*



Algorithm 1: GMRES algorithm

```
for $l = 1$ to  do
    $\mathbf{r} := \mathbf{b} - \mathbf{A}\mathbf{x}^{(j-1)}$
    $\mathbf{q}_1 := \mathbf{r}/\,||r||_2$
    for $j = 1$ to $restart$ do
        $\mathbf{w}_0 := \mathbf{A}\mathbf{q}_j$
        for $i = 1$ to $j$ do
            $h_{i,j} := \mathbf{q}_i \cdot \mathbf{w}_{i-1}$
            $\mathbf{w}_i := \mathbf{w}_{i-1} - h_{i,j}\mathbf{q}_i$
        end
        $h_{j+1,j} := ||\mathbf{w}||_2$
        $\mathbf{q}_{j+1} := \mathbf{w}/h_{j+1,j}$
        Find $\mathbf{y} = \min||\mathbf{H}_j\mathbf{y} - ||\mathbf{b}||\,\mathbf{e}_1||_2$
        Evaluate convergence criteria
        Optionally, compute $\mathbf{x}_j = \mathbf{Q}_j\mathbf{y}$
    end
end
```

GMRES

Theoretical Bounds on the Arnoldi Process

$$\|\mathbf{w}_0\| = \|\mathbf{A}\mathbf{q}_j\| \leq \|\mathbf{A}\|_2\|\mathbf{q}_j\|_2$$
$$\|\mathbf{w}_0\| \leq \|\mathbf{A}\|_2 \leq \|\mathbf{A}\|_F$$

From isometry of orthogonal projections,

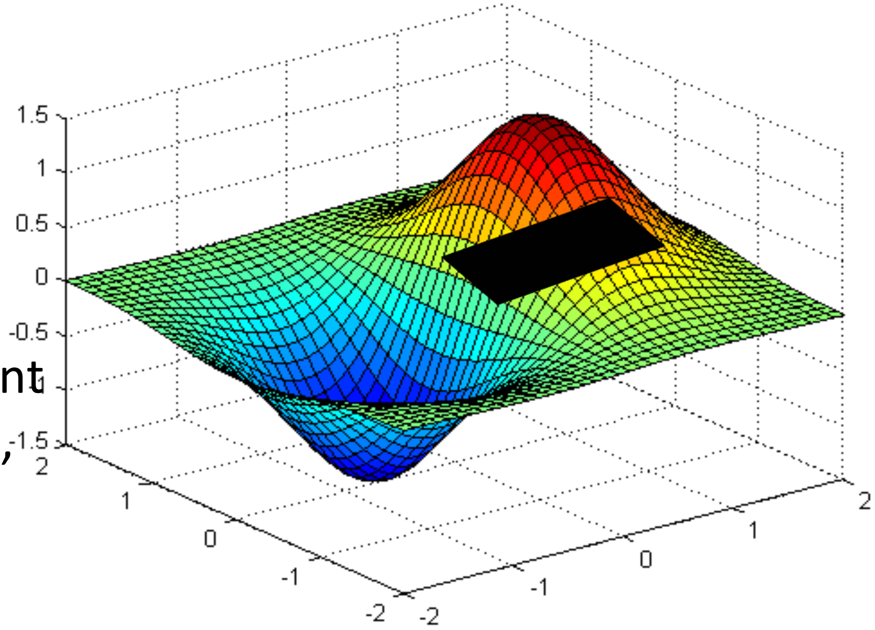
$$|h_{i,j}| \leq \|\mathbf{A}\|_F$$

- $h_{i,j}$ form Hessenberg Matrix
- Bound only computed once, valid for entire solve

Evaluating the Impact of SDC in Numerical Methods
J. Elliott, M. Hoemmen, F. Mueller, SC'13

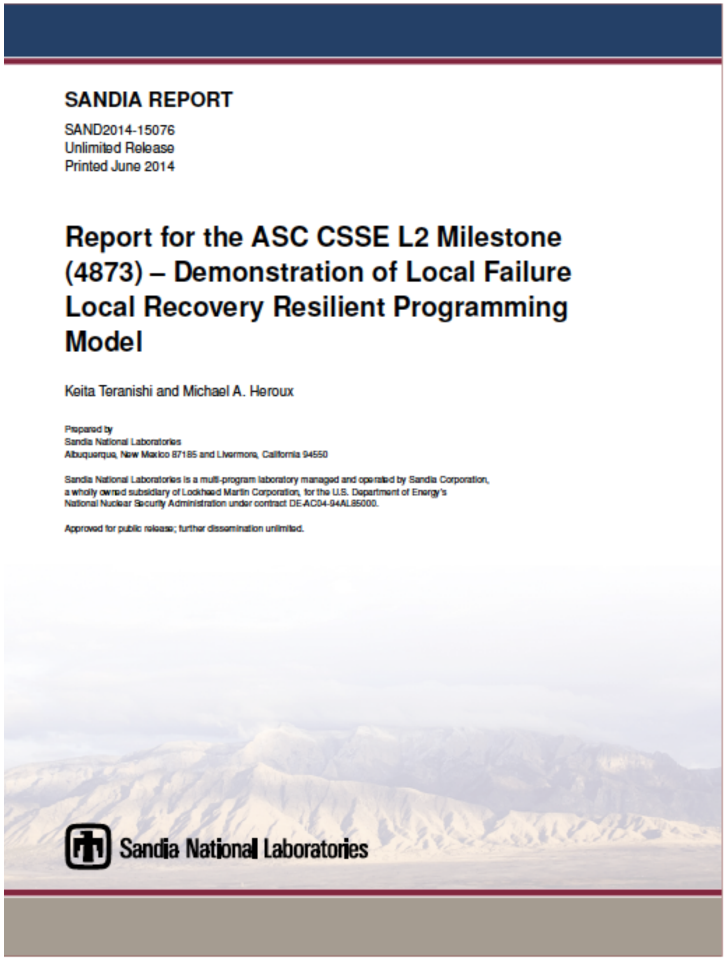
Enabling Local Recovery from Local Faults

- Current recovery model:
  Local node failure,
  global kill/restart.
- Different approach:
  - App stores key recovery data in persistent local (per MPI rank) storage (e.g., buddy, NVRAM),
    and registers recovery function.
  - Upon rank failure:
    - MPI brings in reserve HW, assigns to failed rank, calls recovery fn.
    - App restores failed process state via its persistent data (& neighbors'?).
    - All processes continue.



Motivation for LFLR:

- Current practice of Checkpoint/ Restart is global response to single node (local) failure
  - Kill all processes (global terminate), then restart
  - Dependent on Global File system
  - SCR (LLNL) is fast, but adheres global recovery
- Single node failures are predominant
  - 85% on LLNL clusters (Moody et al. 2010)
  - 60-90% on Jaguar/Titan (ORNL)
- Need for scalable, portable and application agnostic solution
  - Local Failure Local Recovery Model (LFLR)



SANDIA REPORT

SAND2014-15076
Unlimited Release
Printed June 2014

Report for the ASC CSSE L2 Milestone (4873) – Demonstration of Local Failure Local Recovery Resilient Programming Model

Keita Teranishi and Michael A. Heroux

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Approved for public release; further dissemination unlimited.

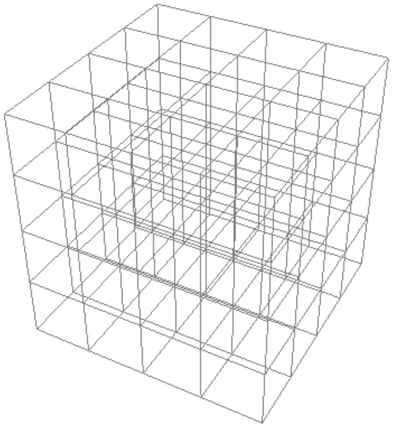
Sandia National Laboratories

TOWARD A NEW APPLICATION ARCHITECTURE

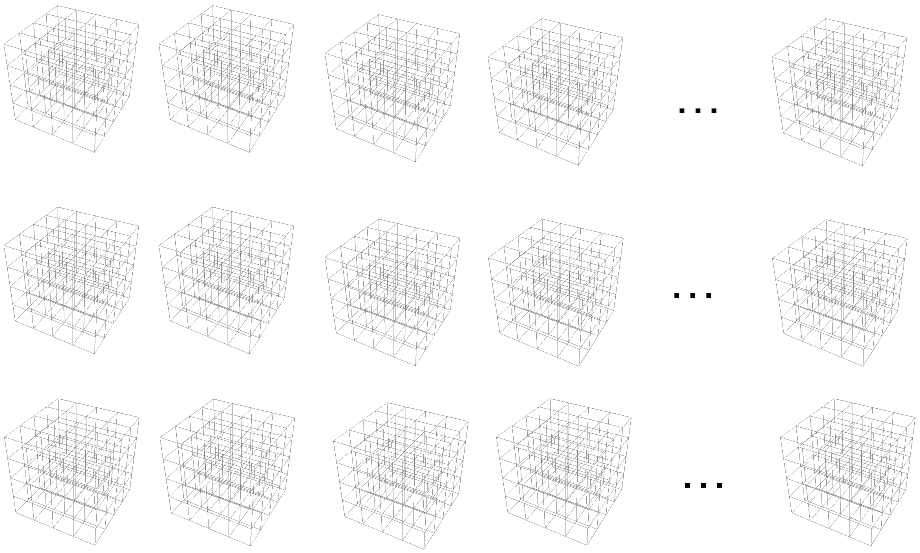
Manytasking: A Productive Application Architecture

- Atomic Unit: Task
  - Domain scientist writes code for a task.
  - Task execution requirements:
    - Tunable work size: Enough to efficiently use a core once scheduled.
    - Vector/SIMT capabilities.
- Utility of Task-based Approach:
  - Oversubscription: Latency hiding, load balancing.
  - Dataflow: Task-DAG or futures.
  - Resilience: Re-dispatch task from parent.
  - Déjà vu for apps developers: Feels a lot like MPI programming.
  - Universal portability: Works within node, across nodes.

Task-centric Benefits



- □ MPI:
  - Halo exchange.
  - Local compute.
  - Global collective.
  - Halo exchange.



- Task-centric: Many tasks
  - Async dispatch: Many in flight.
  - Natural latency hiding.
  - Higher message injection rates.
  - Better load balancing.
  - Compatible with “classics”:
    - Fortran, vectorization, small-scale OMP.
    - Used within a task.
  - Natural resilience model:
    - Every task has a parent (can regenerate).
  - Demonstrated concept:
    - Co-Design centers, PSAAP2, others.

Execution Policy for Task Parallelism

- TaskManager< ExecSpace > execution policy
  - Policy object shared by potentially concurrent tasks

    ```
    TaskManager<...> tm( exec_space , ... );
    Future<> fa = spawn( tm , task_functor_a ); // single-thread task
    Future<> fb = spawn( tm , task_functor_b );
    ```

  - Tasks may be data parallel

    ```
    Future<> fc = spawn_for( tm.range(0..N) , functor_c );
    Future<value_type> fd = spawn_reduce( tm.team(N,M) , functor_d );
    wait( tm ); // wait for all tasks to complete
    ```

  - Destruction of task manager object waits for concurrent tasks to complete
- Task Managers
  - Define a scope for a collection of potentially concurrent tasks
  - Have configuration options for task management and scheduling
  - Manage resources for scheduling queue

Summary: Task-centric app design

- Scalable application design will move to a task-centric architecture:
  - Provides a sequential view for domain scientists.
    - Looks a lot like MPI programming.
    - Only added requirements: Consumer/producer dependencies.
  - Support vectorization/SIMT within a task.
  - Supports many (all, really) threading environments.
  - Permits continued use of Fortran.
  - Provides a resilience-capability architecture.
- Challenges to developing task-centric apps:
  - Much more complicated MPI node-level interactions:
  - OS/RT support for task-DAGS:
    - What are the Apps responsibility? How can OS/RT assist?
    - Concurrent execution is essential for scalability.
      - Must be reading/writing from memory, computing simultaneously.

TRILINOS FRAMEWORK AND TOOLS

Trilinos Framework and Tools

- Changes since last year
  - User-focused web content moved from trilinos.sandia.gov to trilinos.org
    - Some links still need to be cleaned up
  - Moved to time-based, rather than feature-based releases (4 per year)
  - Updated tutorial delivery - Trilinos_tutorial
    - Student accounts accessible via ssh
    - Virtual machine image
    - Trilinos_tutorial is on Github
  - First collaborator agreements in place
    - A long way to go yet

Trilinos Framework and Tools

- Plans for Upcoming Year
  - Make public repo identical to developer repo
    - Put on Github, support pull request workflow
  - Move trilinos-user and trilinos-announce mail lists to trilinos.org
  - 12.0 release scheduled for April
    - Chance for packages to break backward compatibility
  - More collaboration agreements
  - Improve support for and integration of externally developed packages

TRILINOS SW ENGINEERING TECHOLOGIES AND INTEGRATION

Trilinos Software Engineering Technologies and Integration

Progress in last year:

- TriBITS Hosted on Github
  - URL: https://github.com/TriBITSPub/TriBITS
  - Github issues, pull requests, etc.
  - Snapshotted into Trilinos (keep integrated).
    - http://trac.trilinos.org/wiki/TriBITSTrilinosDev
- TriBITS Documentation: Developers guide 170+ pages, build/test reference 30+ pages, overview document in progress …

Plans for Next Year:

- TriBITS System (IDEAS Project)
  - Partition TriBITS into lighter-weight framework(s), better support wrapping external software as a TriBITS package, etc.
  - Merge TriBITS concepts of Packages and TPLs => Construct larger meta-projects, build/install/test meta-projects in pieces, extract and build/install individual packages, (optionally) build Trilinos with TPLs, use export XXXConfig.cmake files as glue.
  - Standard installations of TriBITS
  - Overview and tutorials
  - Implementation of TriBITS Lifecycle Model in TriBITS system => Targeted metrics and testing of backward compatibility, valgrind, coverage, etc.
- TriBITS Lifecycle Model Adoption and Refinement: IDEAS, Trilinos, CASL

TRILINOS EMBEDDED NONLINEAR ANALYSIS TOOLS

Embedded Nonlinear Analysis Tools Capability Area

We are: top level algorithms (outermost loops)

- solution of nonlinear equations
- time integration
- bifurcation tracking / stability analysis / parameter continuation
- optimization (black-box, PDE-constrained, full-space)
- uncertainty quantification
- multi-physics coupling
- model order reduction

Governing Philosophy: "Analysis beyond Simulation,"

Goal: to automate many computational analysis and design tasks, using applied math and algorithms to replace trial-and-error or repeated simulation.

- parameter studies
- sensitivity analysis
- calibration
- optimization
- locating instabilities
- performing UQ

Trilinos Strategic Goals that we align with:

1. **Full Vertical Coverage**
2. Hardened Solvers
3. Scalability

Trilinos Packages in the Embedded Nonlinear Analysis Capability Area

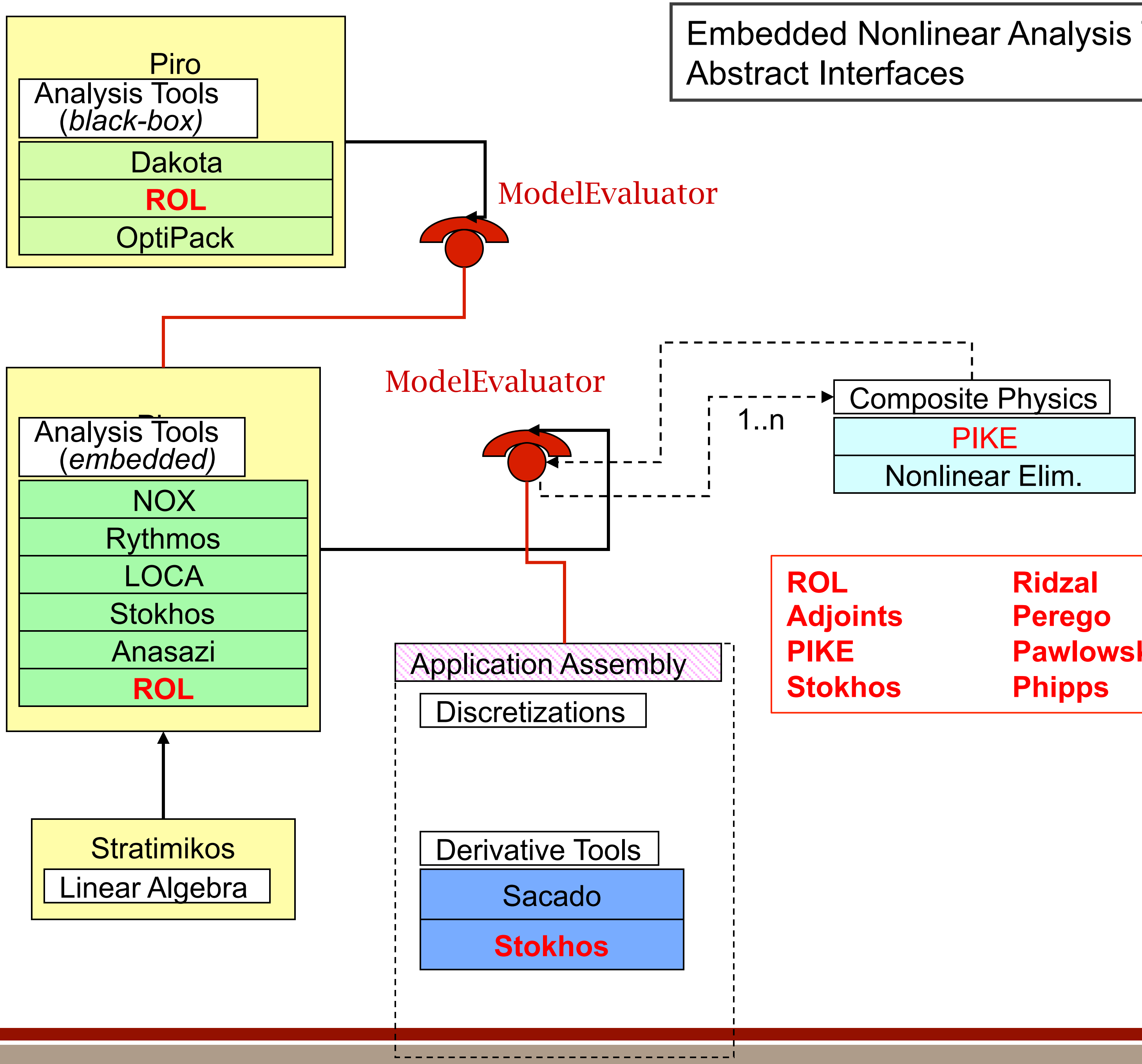
Package Name	Quick Description	Point of Contact
Piro	Uniform Wrapper for most ENAT Capabilities	Andy Salinger
NOX	Nonlinear Solver with Globalized Newton’s methods	Roger Pawlowski
LOCA	Parameter Continuation, Bifurcation Tracking, 4D	Eric Phipps
Rythmos	Time integration algorithms	
Sacado	Automatic Differentiation using Expression Templates	Eric Phipps
Stokhos	Stochastic-Galerkin Uncertainty Quantification Tools	Eric Phipps
TriKota	Interface to Dakota for a Trilinos app	Andy Salinger
ROL	Embedded Optimization	Ridzal, Kouri
PIKE	Multi-Physics coupling	Pawlowski
Moocho	Embedded (PDE-constrained) Optimization, rSQP	Roscoe Bartlett
OptiPack	Nonlinear CG	Roscoe Bartlett
GlobiPack	Library of Line search methods for globalizations	Roscoe Bartlett
Aristos	Full-Speace Optimization	Denis Ridzal
Razor	Model Order Reduction	Cortial & Carlberg, Kalashnikova

Related Efforts Outside of Trilinos

Dakota — Dakota is a mature and widely-used software toolkit that delivers many analysis capabilities using a non-intrusive (a.k.a. blackbox) interface...

Albany, Panzer — Codes built in Albany or on Panzer are born with transformational analysis capabilities.

Embedded Nonlinear Analysis Tools and Key Abstract Interfaces



ROL	**Ridzal**	**Wed 10:00**
Adjoints	**Perego**	**Wed 10:50**
PIKE	**Pawlowski**	**Wed 11:10**
Stokhos	**Phipps**	**Wed 2:40**

TRILINOS UX

Trilinos User Experience Capability Area

Bill Spotz, lead

Web Site Design (Dena Vigil, lead)

- Trilinos.org is on-line
  - Largely just a placeholder right now
  - Missing content
- Plans are converging for Content Management System
  - Thursday brown bag roundtable for data base design **(please participate!)**
  - DB relationship to canonical information in Trilinos repository
    - Policies and tools to keep information synchronized
- Collapsible documentation for ParameterLists
  - Automated system for dynamic content in doxygen **(demo)**
  - Infrastructure is ready for developers to start implementing
  - Instructions in `doc/DocumentingParametertLists/memo`

Trilinos User Experience Capability Area

Documentation, Examples and Tutorials (Mark Hoemmen, lead)

- We have 3 tutorials this year!
  - VECPAR (summer, full day!)
  - EuroTUG (summer)
  - TUG (this week)
- New resources and tutorial materials
  - Trilinos virtual machine / install (Jim Willenbring)
  - Paratools servers (Sameer Shende, U Oregon)
  - MueLu tutorial with its own virtual machine (Tobias Wiesner)
  - Kokkos tutorial examples (Christian Trott)
  - Anasazi examples (Alicia Klinvex) and manual (Rich Lehoucq)
- Progress in making tutorials scalable (meaning O(1) time to prepare or hand off)
  - **(+) More examples live in Trilinos repository**
  - **(+) Tested nightly; appear in Doxygen automatically**
  - **(-) Virtual machine / WebTrilinos prep still manual**
- Big challenge: MPI+X
  - Programming models and interfaces are in flux
  - Refactor (necessarily) takes time from UX
- Think of ways to make your time scale
  - Leverage nightly tests and other automation
  - Make examples also serve as tests
  - Build a tutorial out of examples
  - **Add searchable metadata to examples**
- Cross-package higher-level interfaces?
  - If I'm a new user and I want to build a preconditioner for my linear system, where do I look? Do I go straight to the packages or look in Stratimikos?

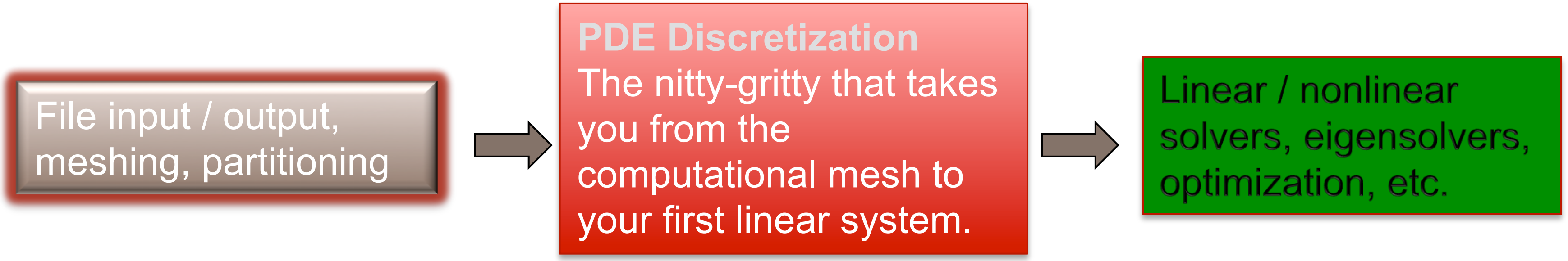
Trilinos User Experience Capability Area

Simplified Layers & Skins (Bill Spotz, lead)

- Enthought SBIR
  - Develop a distributed array capability for Python (DistArray), with a NumPy-like interface
  - Develop a protocol for copy-less conversions to data structures to and from Trilinos, PETSc, Global Arrays, etc.
- Sandia Role: Enhance PyTrilinos to work with DistArrays
  - New *Domi* package for distributed multi-dimensional data
  - New wrappers for Tpetra, Domi, *others to come …*
  - Employ converters to use DistArrays
- Not much work on ForTrilinos or CTrilinos fronts
  - Much discussion about using Stratimikos as a high-level interface
  - Registration, templates, circular dependencies…

What is the Discretization Capability Area?

The Discretization Capability Area is a collection of low-level software tools that enable rapid development of application codes based on the numerical solution of partial differential equations (PDEs).



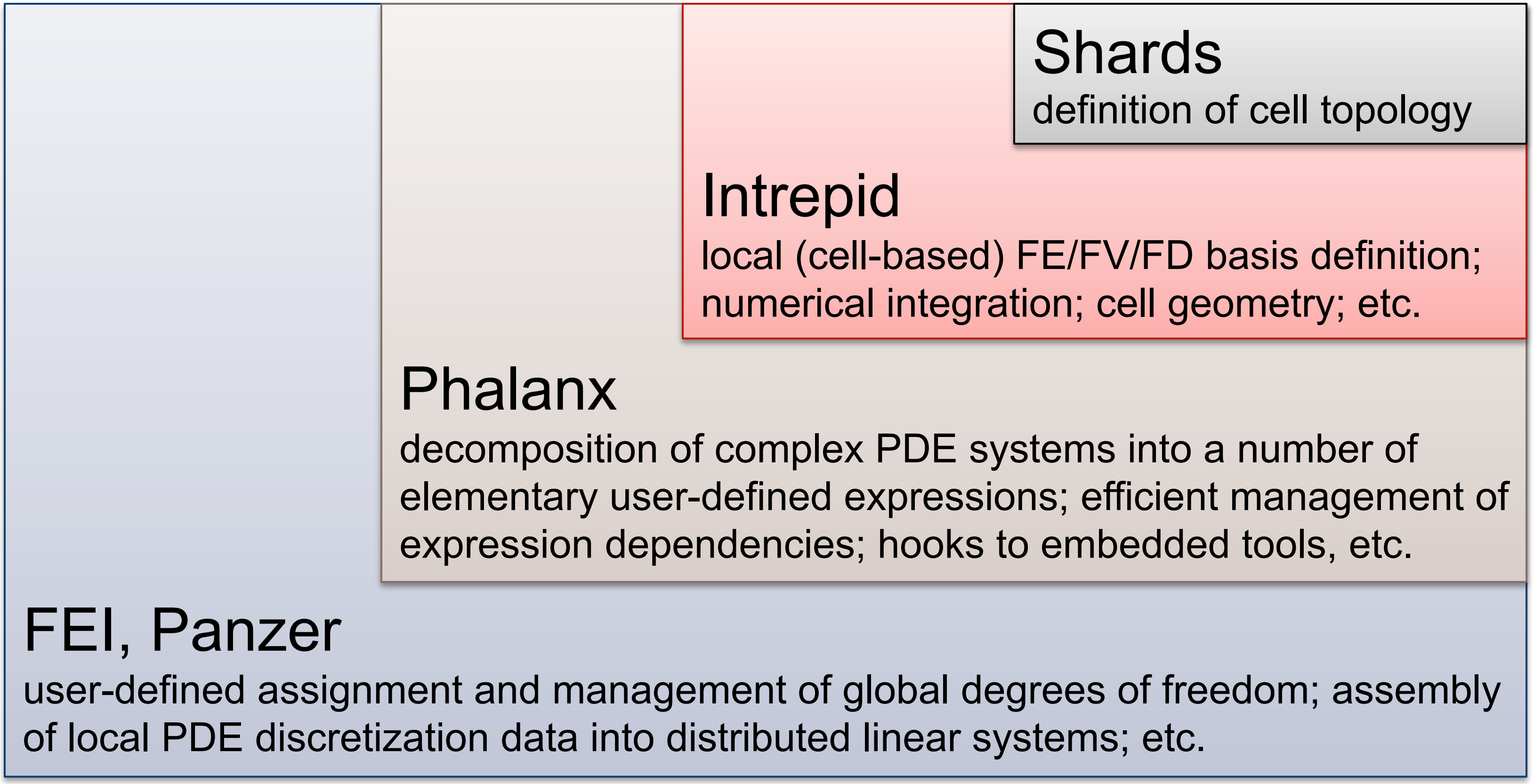
File input / output, meshing, partitioning

PDE Discretization
The nitty-gritty that takes you from the computational mesh to your first linear system.

Linear / nonlinear solvers, eigensolvers, optimization, etc.

Discretization Capability Area

Which packages?



Shards

definition of cell topology

Intrepid

local (cell-based) FE/FV/FD basis definition; numerical integration; cell geometry; etc.

Phalanx

decomposition of complex PDE systems into a number of elementary user-defined expressions; efficient management of expression dependencies; hooks to embedded tools, etc.

FEI, Panzer

user-defined assignment and management of global degrees of freedom; assembly of local PDE discretization data into distributed linear systems; etc.

Developments and directions

Intrelab: (developers: D. Ridzal & J. Young)

- Enables access to Trilinos from Matlab
- Resides in Intrepid
- Access to Intrepid and ML.
- Can be extended to include other packages as needed.
- Great tool for rapid prototyping

Wednesday

- 8:30-8:50 – Introduction (D. Ridzal)
- 8:50-9:10 – Applications to elasticity (P. Kuberry)

Developments and directions

Flexibility of Intrepid enables non-standard use cases

Wednesday talks:

- A spectral Semi-Lagrangian transport scheme
  9:10-9:30 S. Moe
- A second order CVFEM method
  9:30-10:00 K. Peterson

Transitioning to Kokkos

- Thursday, 8:30-9:00 (Demeshko/Edwards)