Synchronization-reducing and Communication-reducing Algorithms and Programming Models for Large-scale Simulations:

Workshop Goals and Structure

David Keyes
Mathematical and Computer Sciences & Engineering, KAUST

www.exascale.org

INTERNATIONAL EXASCALE SOFTWARE PROJECT

ROADMAP 1.0

Jack Dongarra, Pete Beckman, Terry Moore, Patrick Aerts, Giovanni Aloisio, Jean-Claude Andre, David Barkai, Jean-Yves Berthou, Taisuke Boku, Bertrand Braunschweig, Franck Cappello, Barbara Chapman, Xuebin Chi, Alok Choudhary, Sudip Dosanjh, Thom Dunning, Sandro Fiore, Al Geist, Bill Gropp, Robert Harrison, Mark Hereld, Michael Heroux, Adolfy Hoisie, Koh Hotta, Yutaka Ishikawa, Fred Johnson, Sanjay Kale, Richard Kenway, David Keyes, Bill Kramer, Jesus Labarta, Alain Lichnewsky, Thomas Lippert, Bob Lucas, Barney Maccabe, Satoshi Matsuoka, Paul Messina, Peter Michielse, Bernd Mohr, Matthias Mueller, Wolfgang Nagel, Hiroshi Nakashima, Michael E. Papka, Dan Reed, Mitsuhisa Sato, Ed Seidel, John Shalf, David Skinner, Marc Snir, Thomas Sterling, Rick Stevens, Fred Streitz, Bob Sugar, Shinji Sumimoto, William Tang, John Taylor, Rajeev Thakur, Anne Trefethen, Mateo Valero, Aad van der Steen, Jeffrey Vetter, Peg Williams, Robert Wisniewski, Kathy Yelick

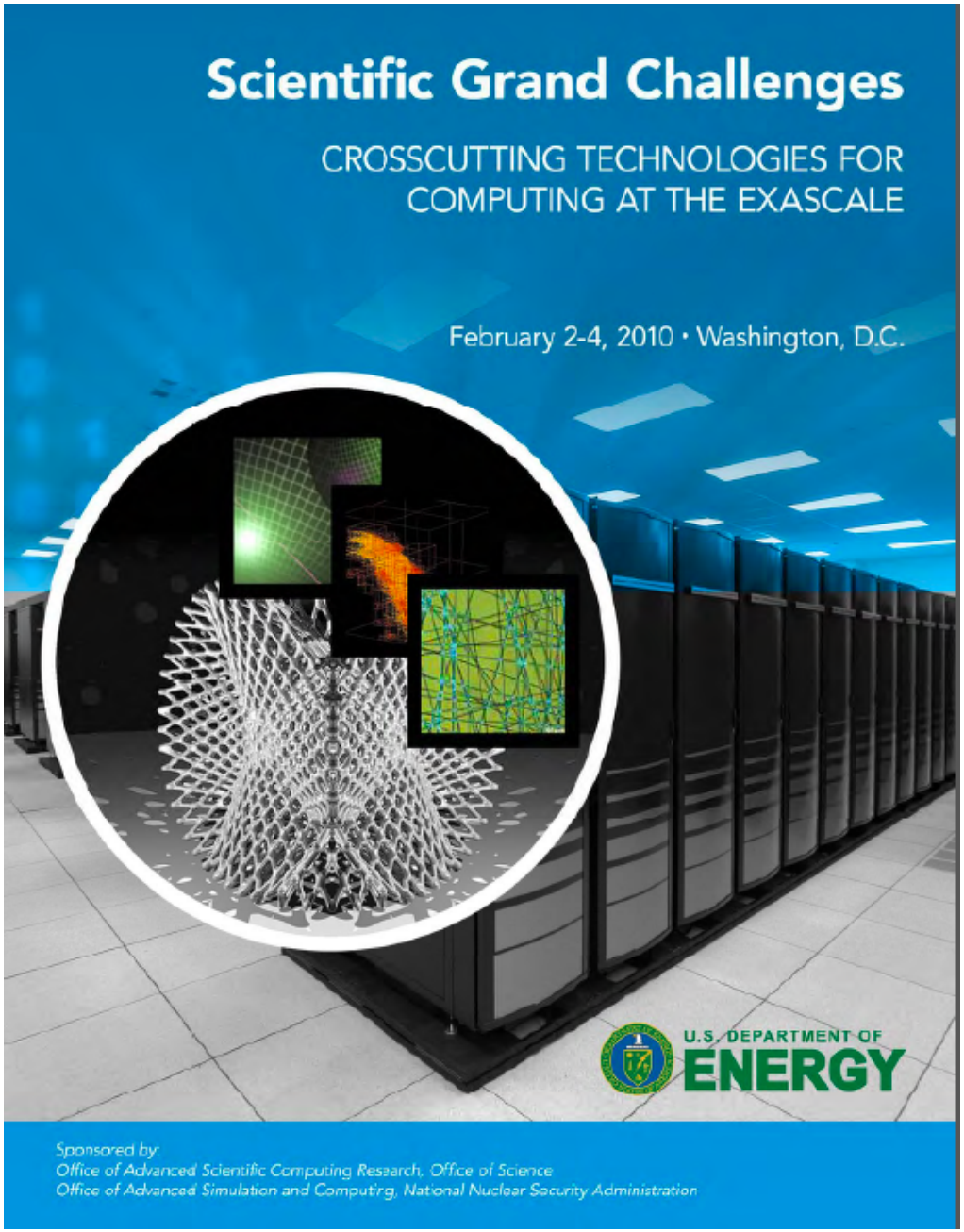
SPONSORS

The International Exascale Software Roadmap

J. Dongarra, P. Beckman, et al.

International Journal of High Performance Computer Applications **25**(1), 2011, ISSN 1094-3420.

extremecomputing.labworks.org



"From an operational viewpoint, these sources of non-uniformity are interchangeable with those that will arise from the hardware and systems software that are too dynamic and unpredictable or difficult to measure to be consistent with bulk synchronization."

"To take full advantage of such synchronization-reducing algorithms, greater expressiveness in scientific programming must be developed. It must become possible to create separate sub-threads for logically separate tasks whose priority is a function of algorithmic state not unlike the way a time-sharing operating system works."

www.exascale.org

INTERNATIONAL EXASCALE SOFTWARE PROJECT ROADMAP 1.0

Jack Dongarra, Pete Beckman, Terry Moore, Patrick Aerts, Giovanni Aloisio, Jean-Claude Andre, David Barkai, Jean-Yves Berthou, Taisuke Boku, Bertrand Braunschweig, Franck Cappello, Barbara Chapman, Xuebin Chi

Alok Choudhary, Sudip Dosanjh, Thom Dunning, Sandro Fiore, Al Geist, Bill Gropp, Robert Harrison, Mark Hereld, Michael Heroux, Adolfy Hoisie, Koh Hotta, Yutaka Ishikawa, Fred Johnson

Sanjay Kale, Richard Kenway, David Keyes, Bill Kramer, Jesus Labarta, Alain Lichnewsky, Thomas Lippert, Bob Lucas, Barney Maccabe, Satoshi Matsuoka, Paul Messina, Peter Michielse, Bernd Mohr

Matthias Mueller, Wolfgang Nagel, Hiroshi Nakashima, Michael E. Papka, Dan Reed, Mitsuhisa Sato, Ed Seidel, John Shalf, David Skinner, Marc Snir, Thomas Sterling, Rick Stevens, Fred Streitz

Bob Sugar, Shinji Sumimoto, William Tang, John Taylor, Rajeev Thakur, Anne Trefethen, Mateo Valero, Aad van der Steen, Jeffrey Vetter, Peg Williams, Robert Wisniewski, Kathy Yelick

SPONSORS

"Even current systems have a 10^3-10^4 cycle hardware latency in accessing remote memory. Hiding this latency requires algorithms that achieve a computation/communication overlap of at least 10^4 cycles."

"Many current algorithms have synchronization points (such as dot products/allreduce) that limit opportunities for latency hiding (this includes Krylov methods for solving sparse linear systems). These synchronization points must be eliminated. Finally, static load balancing rarely provides an exact load balance; experience with current terascale and near petascale systems suggests that this is already a major scalability problem for many algorithms."

Approximate power costs (in picoJoules)

	2010	2018
DP FMADD flop	**100 pJ**	**10 pJ**
DP DRAM read	**2000 pJ**	**1000 pJ**
DP copper link traverse (short)	**1000 pJ**	**100pJ**
DP optical link traverse (long)	**3000 pJ**	**500 pJ**

Mission Statement

> *The mission of the Institute for Computational and Experimental Research in Mathematics (ICERM) is to support and broaden the relationship between mathematics and computation: specifically, to expand the use of computational and experimental methods in mathematics, to support theoretical advances related to computation, and address problems posed by the existence and use of the computer through mathematical tools, research and innovation.*

Purpose of this presentation

- **Establish wide topical playing field**
- **Propose workshop goals**
- **Describe workshop structure**
- **Provide some motivation and context**
- **Give concrete example of a workhorse that may need to be sent to the glue factory – or be completely re-shoed**
- **Establish a dynamic of interruptability and informality for the entire meeting**

Workshop goals (web)

As concurrency in scientific computing pushes beyond a million threads and performance of individual threads becomes less reliable for hardware-related reasons, attention of mathematicians, computer scientists, and supercomputer users and suppliers inevitably focuses on reducing communication and synchronization bottlenecks.

Though convenient for succinctness, reproducibility, and stability, instruction ordering in contemporary codes is commonly overspecified.

This workshop attempts to outline evolution of simulation codes from today's infra-petascale to the ultra-exascale and to encourage importation of ideas from other areas of mathematics and computer science into numerical algorithms, new invention, and programming model generalization.

“other areas …”

… besides traditional HPC, that is

This could include, among your examples:

- **formulations beyond PDEs and sparse matrices**
- **combinatorial optimization for schedules and layouts**
- **tensor contraction abstractions**
- **machine learning about the machine or the execution**

“other areas …”

… and revivals of classical parallel numerical ideas:

- **dataflow-based (dynamic) scheduling**
- **mixed (minimum) precision arithmetic**
- **wide halos for multi-stage sparse recurrences**
- **multistage unrolling of Krylov space generation with aggregated inner products and reorthogonalization**
- **dynamic rebalancing/work-stealing**

“other areas”

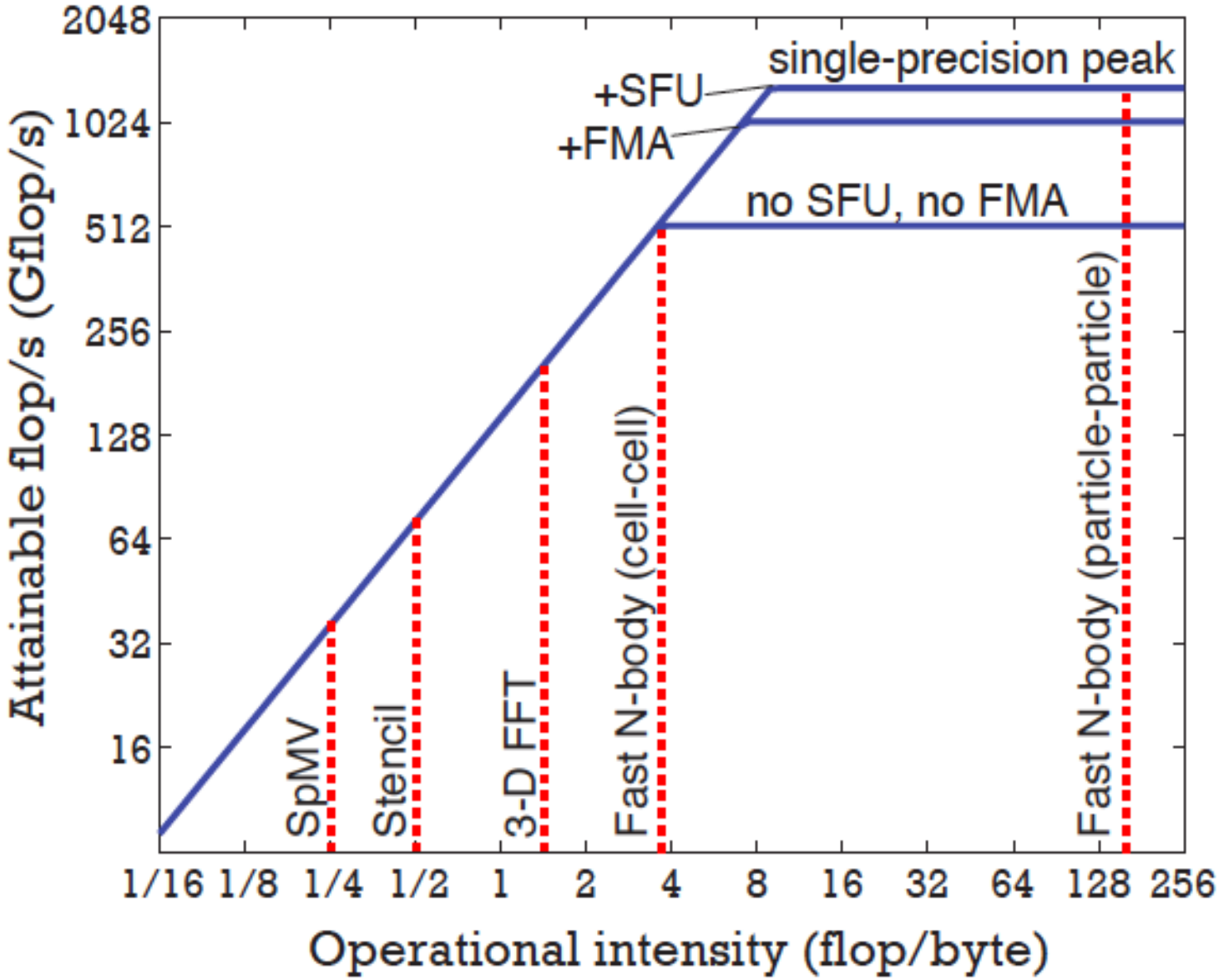
This could also include more radical ideas:

- **on-the-fly data compression/decompression**
- **statistical substitution of missing/delayed data**
- **user-controlled data placement**
- **user-controlled error handling**

Formulations w/better arithmetic intensity

Roofline model of numerical kernels on an NVIDIA C2050 GPU (Fermi). The 'SFU' label is used to indicate the use of special function units and 'FMA' indicates the use of fused multiply-add instructions.

(The order of fast multipole method expansions was set to p = 15.)



c/o L. Barba (BU); cf. "Roofline Model" of S. Williams (Berkeley)

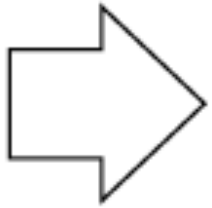
FMM should be applicable as a preconditioner

Differential operator

$$\mathcal{L}(u)(\mathbf{x}) = f(\mathbf{x})$$

Linear system

$$A\mathbf{x} = \mathbf{b}$$



Integral operator

$$u(\mathbf{x}) = \int_{\Omega} G(\mathbf{x}, \mathbf{y}) f(\mathbf{y})$$

Linear system

$$\mathbf{x} = A^{-1}\mathbf{b} = G\mathbf{b}$$

known

$O(N)$

scalable

For complex geometries with boundary conditions

combine with boundary element methods

For equations that don't have a Green's function

still can be used as a preconditioner

Revival of lower/mixed-precision

- **Algorithms in provably well-conditioned contexts**
  - **Fourier transforms of relative smooth signals**
- **Algorithms that require only approximate quantities**
  - **matrix elements of preconditioners**
  - **used in full precision with padding, but transported and computed in low**
- **Algorithms that mix precisions**
  - **classical iterative correction in linear algebra, and other delta-oriented corrections**

Statistical completion of missing (meta-)data

- Once a sufficient number of threads hit a synchronization point, missing threads can be assessed
- Some missing data may be of low or no consequence
  - contributions to a norm allreduce, where the accounted for terms already exceed the convergence threshold
  - contributions to a timestep stability estimate where proximate points in space or time were not extrema
- Other missing data, such as actual state data, may be reconstructed statistically
  - effects of uncertainties may be bounded (e.g., diffusive problems)
  - synchronization may be released speculatively, with ability to rewind

Bad news/good news (1)

- **One may have to control data motion**
  - **carries the highest energy cost in the exascale computational environment**
- **One finally will get the privilege of controlling the vertical data motion**
  - **horizontal data motion under control of users under *Pax MPI*, already**
  - **but vertical replication into caches and registers was (until now with GPUs) scheduled and laid out by hardware and runtime systems, mostly invisibly to users**

Bad news/good news (2)

- **"Optimal" formulations and algorithms may lead to poorly proportioned computations for exascale hardware resource balances**
  - **today's "optimal" methods presume flops are expensive and memory and memory bandwidth are cheap**
- **Architecture may lure users into more arithmetically intensive formulations (e.g., fast multipole, lattice Boltzmann, rather than mainly PDEs)**
  - **tomorrow's optimal methods will (by definition) evolve to conserve what is expensive**

Bad news/good news (3)

- **Default use of high precision may come to an end, as wasteful of storage and bandwidth**
  - **we will have to compute and communicate "deltas" between states rather than the full state quantities, as we did when double precision was expensive (e.g., iterative correction in linear algebra)**
  - **a combining network node will have to remember not just the last address, but also the last values, and send just the deltas**
- **Equidistributing errors properly while minimizing resource use will lead to innovative error analyses in numerical analysis**

Engineering design principles

- Optimize the right metric
- Measure what you optimize, along with its sensitivities to the things you can control
- Oversupply what is cheap to utilize well what is costly
- Overlap in time tasks with complementary resource constraints if other resources (e.g., power, functional units) remain available
- Eliminate artifactual synchronization and artifactual ordering

User-controlled reliability

- Hidden energy cost of reliability is large, in terms of chip real estate and operating power
- Currently we describe data type (including precision)
- We could in addition describe:
  - scope of cacheability, prefetchability
  - reliability requirements for robustness
    - Krylov coefficient must be reliable
    - Pixel color code need not be reliable
    - State vector component in a diffusive system may or may not

Aggressive de-synchronization

- Isolate deferrable from critical path tasks
- Estimate the costs of deferring tasks
  - unreleased memory
  - degraded convergence
- Use the decomposition into deferrable and critical tasks and cost estimates to determine dynamically the priority for execution of tasks

Newton-Krylov-Schwarz: a fully implicit "workhorse" based on global linearization

$$F(u) \approx F(u_c) + F'(u_c)\delta u = 0 \qquad J\delta u = -F \qquad M^{-1}J\delta u = -M^{-1}F$$

$$u = u_c + \lambda\,\delta u \qquad \delta u = \underset{x \in V \equiv \{F, JF, J^2F, \cdots\}}{\arg\min} \{Jx + F\} \qquad M^{-1} = \sum_i R_i^T (R_i J R_i^T)^{-1} R_i$$

Newton

nonlinear solver

asymptotically quadratic

Krylov

accelerator

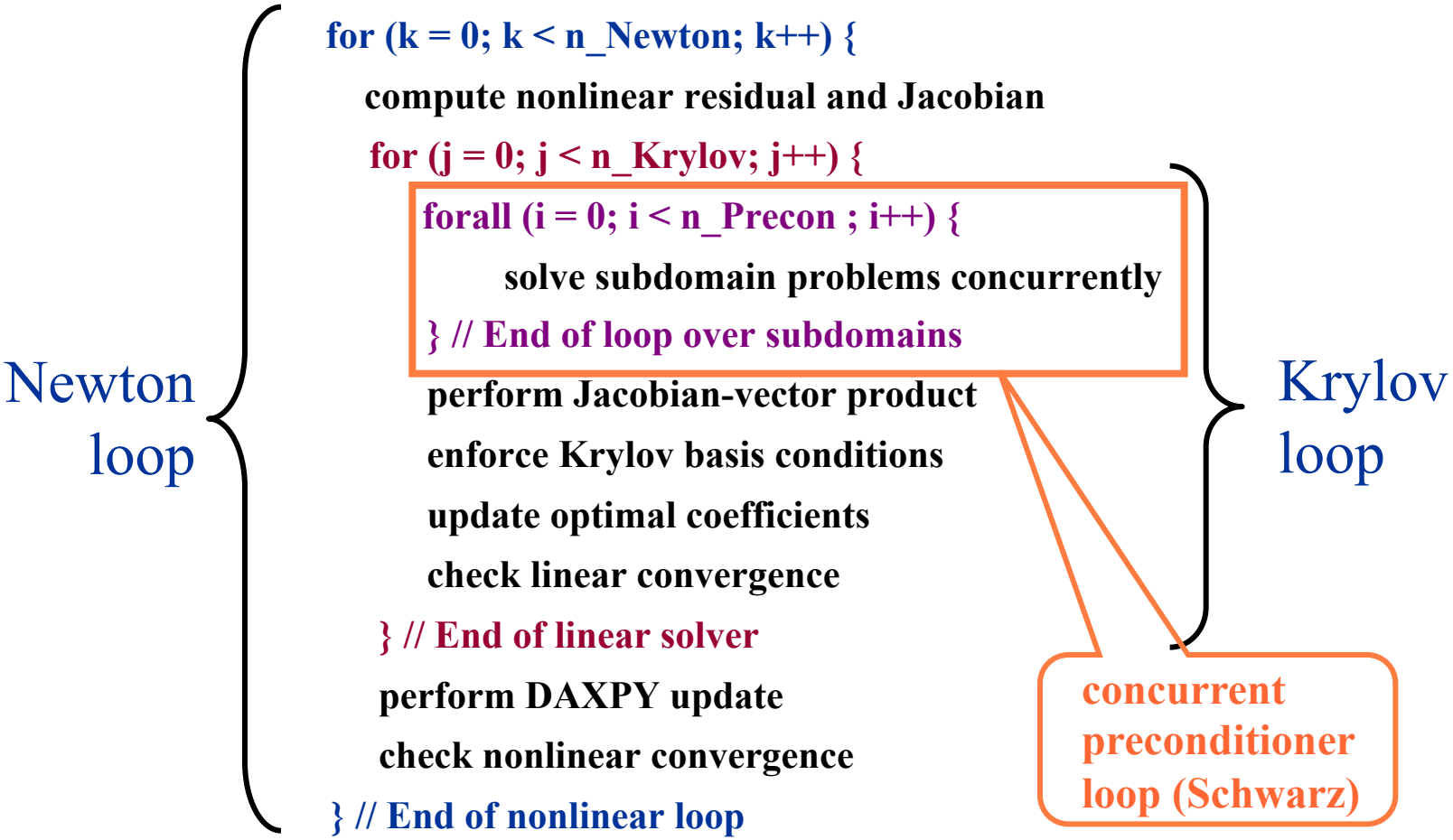
spectrally adaptive

Schwarz

preconditioner

parallelizable

Newton-Krylov-Schwarz loop



Yet outer loops: continuation, implicit timestepping, optimization

How will PDE computations adapt?

- Programming model will still be message-passing (due to large legacy code base), adapted to multicore processors beneath a relaxed synchronization MPI-like interface
- Load-balanced blocks, scheduled today with nested loop structures will be separated into critical and non-critical parts
- Critical parts will be scheduled with directed acyclic graphs (DAGs)
- Noncritical parts will be made available for work-stealing in economically sized chunks

Adaptation to asynchronous programming styles

- To take full advantage of such asynchronous algorithms, we need to develop greater expressiveness in scientific programming
  - create separate threads for logically separate tasks, whose priority is a function of algorithmic state, not unlike the way a time-sharing OS works
  - join priority threads in a directed acyclic graph (DAG), a task graph showing the flow of input dependencies; fill idleness with noncritical work or steal work
- Steps in this direction
  - Asynchronous Dynamic Load Balancing (ADLB) [Lusk (Argonne), 2009]
  - Asynchronous Execution System [Steinmacher-Burrow (IBM), 2008]

Evolution of Newton-Krylov-Schwarz: breaking the synchrony stronghold

- Can write code in styles that do not require artifactual synchronization
- Critical path of a nonlinear implicit PDE solve is essentially

  … lin_solve, bound_step, update; lin_solve, bound_step, update …
- However, we often insert into this path things that could be done less synchronously, because we have limited language expressiveness
  - Jacobian and preconditioner refresh
  - convergence testing
  - algorithmic parameter adaptation
  - I/O, compression
  - visualization, data mining

Philosophy of an algorithmicist

- Applications are *given* (as function of time)
- Architectures are *given* (as function of time)
- Algorithms and software *must be adapted or created* to bridge to hostile architectures for the sake of the complex applications
  - as important as ever today, with transformation of Moore's Law from speed-based to concurrency-based, due to power considerations
  - scalability still important, but new memory-bandwidth stresses arise when on-chip memories are shared
  - greatest challenge is lack of performance robustness of individual cores, which can spoil load balance
- Knowledge of algorithmic capabilities can usefully influence
  - the way applications are formulated
  - the way architectures are constructed
- Knowledge of application and architectural opportunities can usefully influence algorithmic development

Required software enabling technologies

Model-related

- Geometric modelers
- Meshers
- Discretizers
- Partitioners
- Solvers / integrators
- Adaptivity systems
- Random no. generators
- Subgridscale physics
- Uncertainty quantification
- Dynamic load balancing
- Graphs and combinatorial algs.
- Compression

Development-related

- Configuration systems
- Source-to-source translators
- Compilers
- Simulators
- Messaging systems
- Debuggers
- Profilers

Production-related

- Dynamic resource management
- Dynamic performance optimization
- Authenticators
- I/O systems
- Visualization systems
- Workflow controllers
- Frameworks
- Data miners
- Fault monitoring, reporting, and recovery

High-end computers come with little of this stuff. Most has to be contributed by the user community

Other workshop goals

- **Connect communities and develop a web archive**
- **Produce (downstream) a collection of whitepapers or a review paper**

Workshop structure

Hour	MON 9 Jan *algorithms*	TUE 10 Jan *solvers*	WED 11 Jan *architecture*	THU 12 Jan *programming*	FRI 13 Jan *compilers*
9	Goals & Logistics Lightning Talks	Adams	Gropp	Gunnels	Pingali
10	Coffee	Coffee	Hammond	Cavazos	Coffee
11	Yelick	Org. Breakouts Breakouts	Coffee Breakouts	Coffee Breakouts	Presentation 4 Presentation 5 Presentation 6
12	Working	Working	Working		
1	Lunch	Lunch	Lunch		
2	Miller	Cohen	Owens	Barba	Paths Forward
3	Coffee	Coffee	Ltaief	Coffee	Coffee
4	Ballard Poulson Eijkhout	Vuduc Kaushik Brown	Coffee Preliminary Reports	Presentation 1 Presentation 2 Presentation 3	Wrap-up
5	Reception				
6		Dinner			

Workshop flow (spiral structure)

Workshop flow (linear structure)

- Algorithms (Monday)
- Solvers (Tuesday)
- Architectures (Wednesday)
- Programming models (Thursday)
- Compilers (Friday)

Workshop atmosphere

- At a conference, present mainly accomplishments
  - dissemination
- At a workshop, present mainly work in progress
  - feedback

Now, Matt will explain:

- **Lightning talks**
- **Breakout groups**

Lightning talks

- **Short presentations so that all participants, with a reserved speaking slot or not, have a chance to put items onto the breakout group agendas and into the conversation**

Breakout Groups

- Six working groups that will meet in parallel for one hour each before lunch on Tue, Wed, and Thu
- Intended to assess and make recommendations about a particular challenge in the migration of scientific codes to the exascale, in the light of synchronization and communication bottlenecks
- Preliminary reports after first two hours, Wed PM
- Reports and full group discussion Thu PM and Fri AM
- Topics are suggested; groups may diverge or merge

Breakout Group #1

- Impact of new architectures on software libraries
- Leader: Jack Dongarra

Breakout Group #2

- Impact of new algorithms (that is, the ones that have better arithmetic intensity and memory access predictability) on today's software libraries
- Leader: Rob Schreiber

Breakout Group #3

- Case study of the impact of new architectures and algorithms on a particular application and a path forward
- Leader: Esmond Ng

Breakout Group #4

- What kinds of tools do we need to develop new libraries with this technology and assess application needs. (This could include compilers, code generators, integrated debuggers and profilers, and transformation/optimization tools and other things, e.g., that the NSF Blue Waters project and its successors will need.)
- Leader: Bill Gropp

Breakout Group #5

- What capabilities (hardware and software) can vendors provide to allow better control of memory management by programmers?
- Leader: Jim Sexton

Breakout Group #6

- **What kinds of mathematics will we need to support these developments in architecture, algorithms, and software? What connections can we draw among different mathematical disciplines (algorithm analysis, complexity, algebraic geometry, analysis) to understand them?**
- **Leader: Jan Hesthaven**

EOF