Petaflop/s, seriously

David Keyes
Department of Applied Physics & Applied Mathematics
Columbia University

Congratulations, Indian supercomputing colleagues!

- “Eka” (Sanskrit for “number one”)
- #1 in Asia for now
- Also, the first in a series soon to scale to peta
- May it further focus the dreams of India’s treasure of technical talent!

Presentation plan

- Reflect briefly on progress in high-end scientific computing
  - as captured in Gordon Bell prize trends
  - as forecast in petascale architecture projects (from DOE labs in USA)
  - as illustrated on physical applications based on partial differential equations (PDEs)
- Peek briefly at some motivating applications
  - Bell Prizes: mechanics, seismology, aerodynamics
  - SciDAC program: global climate, combustion
- Look generically at PDE-based simulation and the basis of continued optimism for its growth – capability-wise
- Look at some specific hurdles to PDE-based simulation posed by high-end architecture

Technical aspects of presentation

- Introduce a parameterized highly tunable class of algorithms for parallel implicit solution of PDEs: "Newton-Krylov-Schwarz" (ca. 1993)*
  - understand the source of their "weak scalability"
  - understand their lack of "strong scalability"
  - understand why explicit algorithms generally do not scale, even weakly, in the high spatial resolution limit
- Note some algorithmic "adaptations" to architectural stresses

* worked on by Dinesh Kaushik, Satish Balay, and Srikumar Kareti and now a core solver in PETSc from Argonne National Laboratory

Philosophy of presentation

- Applications are ***given*** (as function of time)
- Architectures (hardware and software) are ***given*** (as function of time)
- Algorithms ***must be adapted or created*** to bridge to "hostile" architectures for the sake of the applications
- 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

Gedanken experiment:
How to use a jar of peanut butter as its price slides downward?

- In 2007, at $3.20: make sandwiches
- By 2010, at $0.80: make recipe substitutions for other oils
- By 2013, at $0.20: use as feedstock for biopolymers, plastics, etc.
- By 2016, at $0.05: heat homes
- By 2019, at $0.0125: pave roads ☺

The cost of computing has been on a curve *much better than this* for two decades and promises to continue for at least one more. Like everyone else, scientists should plan increasing uses for it…

Gordon Bell Prize: "price performance"

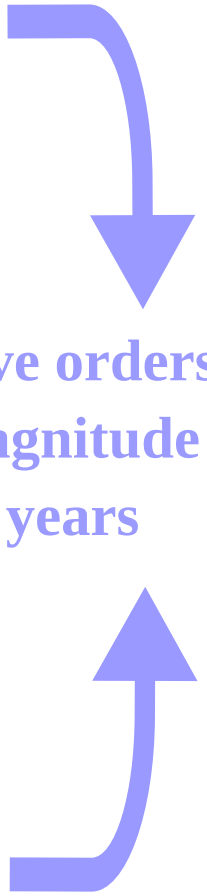
Year	*Application*	*System*	*$ per Mflops*
1989	Reservoir modeling	CM-2	2,500
1990	Electronic structure	IPSC	1,250
1992	Polymer dynamics	cluster	1,000
1993	Image analysis	custom	154
1994	Quant molecular dyn	cluster	333
1995	Comp fluid dynamics	cluster	278
1996	Electronic structure	SGI	159
1997	Gravitation	cluster	56
1998	Quant chromodyn	custom	12.5
1999	Gravitation	custom	6.9
2000	Comp fluid dynamics	cluster	1.9
2001	Structural analysis	cluster	0.24

Four orders of magnitude in 12 years

recent: submissions received for as little as $.03 per Mflop/s using GPUs

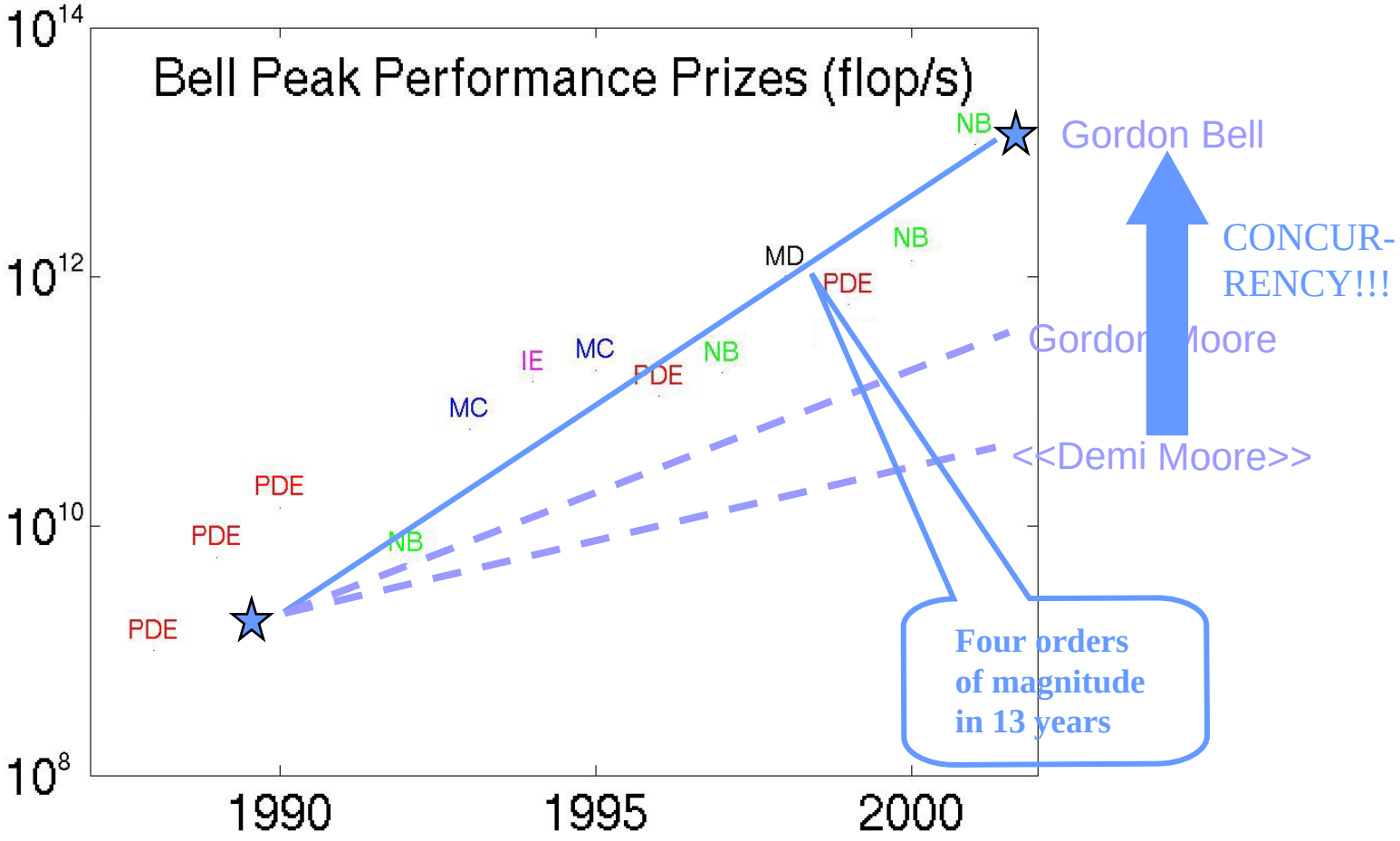
Gordon Bell Prize "peak performance"

Year	*Type*	*Application*	*No. Procs*	*System*	*Gflop/s*
1988	PDE	Structures	8	Cray Y-MP	1.0
1989	PDE	Seismic	2,048	CM-2	5.6
1990	PDE	Seismic	2,048	CM-2	14
1992	NB	Gravitation	512	Delta	5.4
1993	MC	Boltzmann	1,024	CM-5	60
1994	IE	Structures	1,904	Paragon	143
1995	MC	QCD	128	NWT	179
1996	PDE	CFD	160	NWT	111
1997	NB	Gravitation	4,096	ASCI Red	170
1998	MD	Magnetism	1,536	T3E-1200	1,020
1999	PDE	CFD	5,832	ASCI BluePac	627
2000	NB	Gravitation	96	GRAPE-6	1,349
2001	NB	Gravitation	1,024	GRAPE-6	11,550
2002	PDE	Climate	5,120	Earth Sim	26,500
2003	PDE	Seismic	1,944	Earth Sim	5,000
2004	PDE	CFD	4,096	Earth Sim	15,200
2005	MD	Solidification	131,072	BG/L	101,700
2006	MD	Elec. Struct.	131,072	BG/L	207,000



Five orders of magnitude in 17 years

Gordon Bell Prize outpaces Moore's Law



Whimsical remarks on simulation progress measured by Bell, since 1988

- If similar improvements in speed (10^5) had been realized in the airline industry, a 15-hour flight (e.g., JFK-BOM) would require one-half of a second today
- If similar improvements in storage (10^4) had been realized in the publishing industry, our office bookcases could hold the book portion of the collection of the U.S. Library of Congress (~20M volumes)
- If similar reductions in cost (10^4) had been realized in the higher education industry, tuition room and board (at a college in the USA) would cost about $2 per year

Some platforms capable of peak petaflop/s by 2009

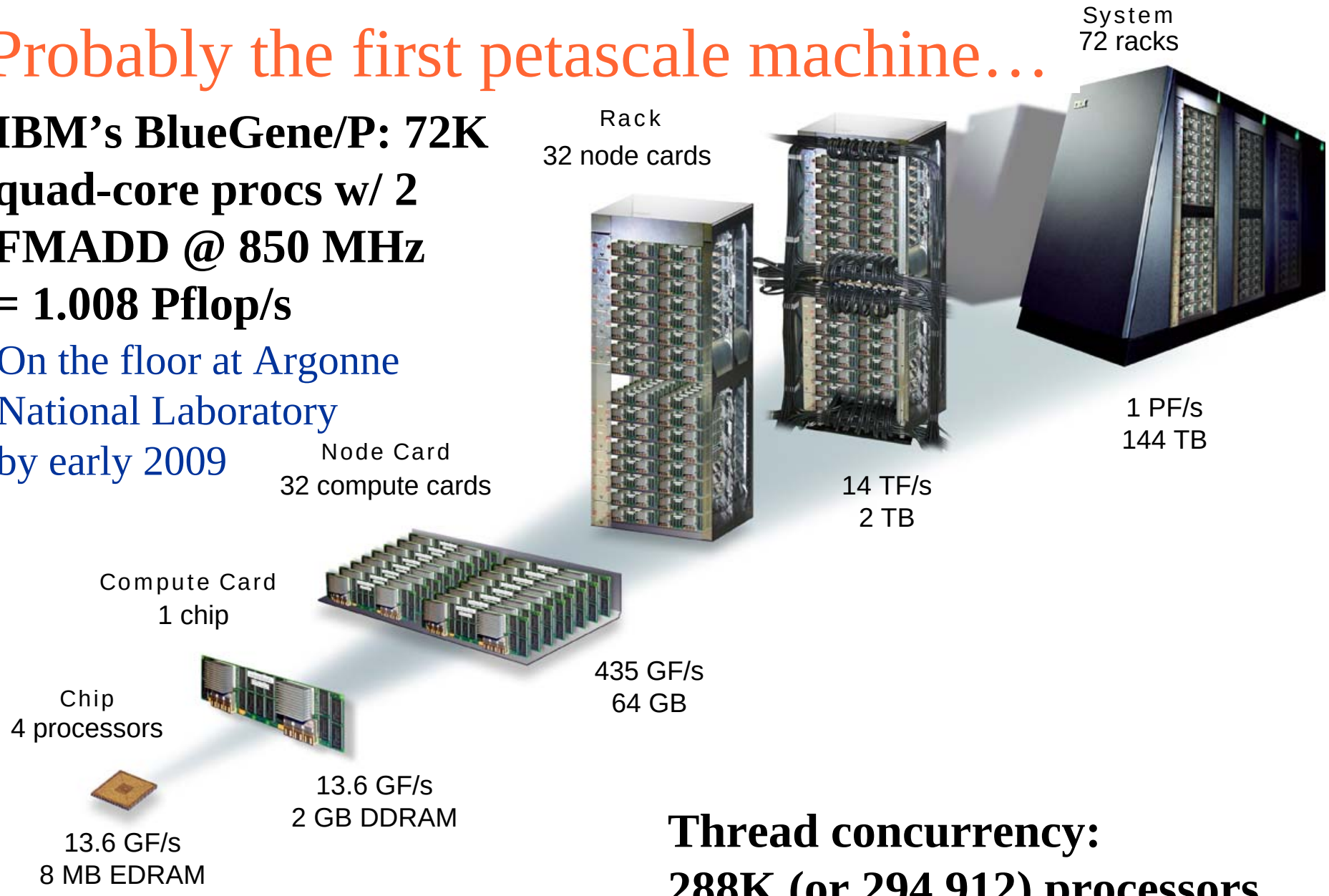
	Scale Demonstrated Factor to PF	Failures per Month Per TF	Power Consumption @ PF	Estimated System Cost
Cray XT3/XT4	23,016 cpus 4x to PF ~100,000 cpus	~.1 - ~1	~8MW^{XT4}	>$150M^{XT4} +memory
IBM Power5/6	12,208 cpus 6x to PF ~72,000 cpus	1.3	~9.4MW^{P6}	>$170M^{P6} +memory
Clusters x86-64/AMD64	14,240 cpus 6x to PF ~84,000 cpus	2.6-8.0	~6MW^{x86QC}	> $150M^{x86} +memory
Blue Gene L/P	212,992 cpus 1.4x to PF 294,912 cpus	.01-.03	~2.3MW^{P}	< $100M^{BG} Including 288TB

c/o Rick Stevens, Argonne National Lab (modified by K, Dec'07)

Probably the first petascale machine…

IBM's BlueGene/P: 72K quad-core procs w/ 2 FMADD @ 850 MHz = 1.008 Pflop/s

On the floor at Argonne National Laboratory by early 2009



Thread concurrency: 288K (or 294,912) processors

What will petaflop/s machines look like?

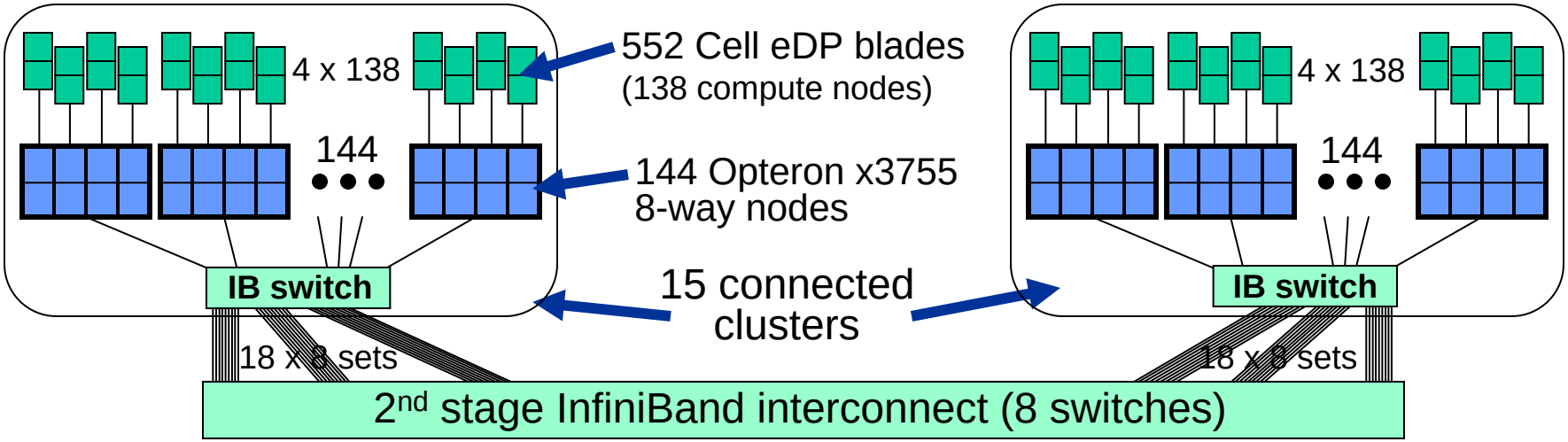
- Many paths beyond silicon, but not in 5 years, at petascale
- BG/P will likely be the first “general purpose” Pflop/s machine; other specialized machines may reach earlier
- Beyond BG/P, at least for PDE-based scientific codes:
  - programming model will still be message-passing (due to large legacy code base), adapted to multicore processors beneath the MPI interface
- Earliest and most significant device improvement will be nanotech memory – but not for earliest Pflop/s machines
  - up to tens of GB on a 1cm-square die
  - will deal directly with the “memory wall” problem

How much parallelism will be required to ***routinely sustain*** 1 petaflop/s

- Realistically, applications max out at about 25% (PDE) to 50% (MD) of peak (after great effort at tuning by experts)
- Hypothetical low power machines will feature 1.6M to 6.6M way parallelism
  - 32-64 cores per processor and up to 2-4 threads per core
  - Assume 25.6K nodes, each with 1 processor socket
- Hypothetical Intel terascale chip petascale system yields 1.5M way parallelism
  - 80 cores per processor
  - Assume 4,608 nodes each with 4 processor sockets
- This is about 8 to 32 times the concurrency of today's largest BlueGene/L!

Roadrunner architecture

"plan of record" for 2008 Cell-accelerated system



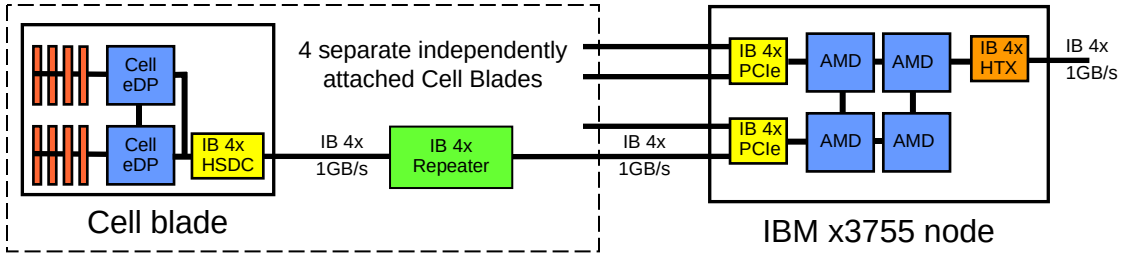
8,640 dual-core Opterons

⇨ **76 Teraflop/s**

16,560 Cell eDP chips

⇨ **1.7 Petaflop/s**

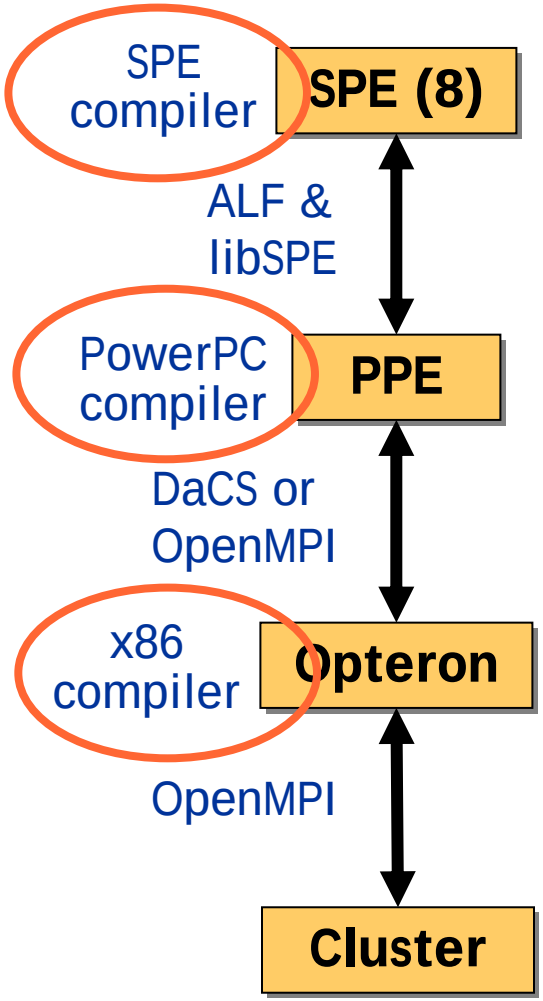
One Accelerated Node



1 Cell processor per Opteron core

Programming Roadrunner

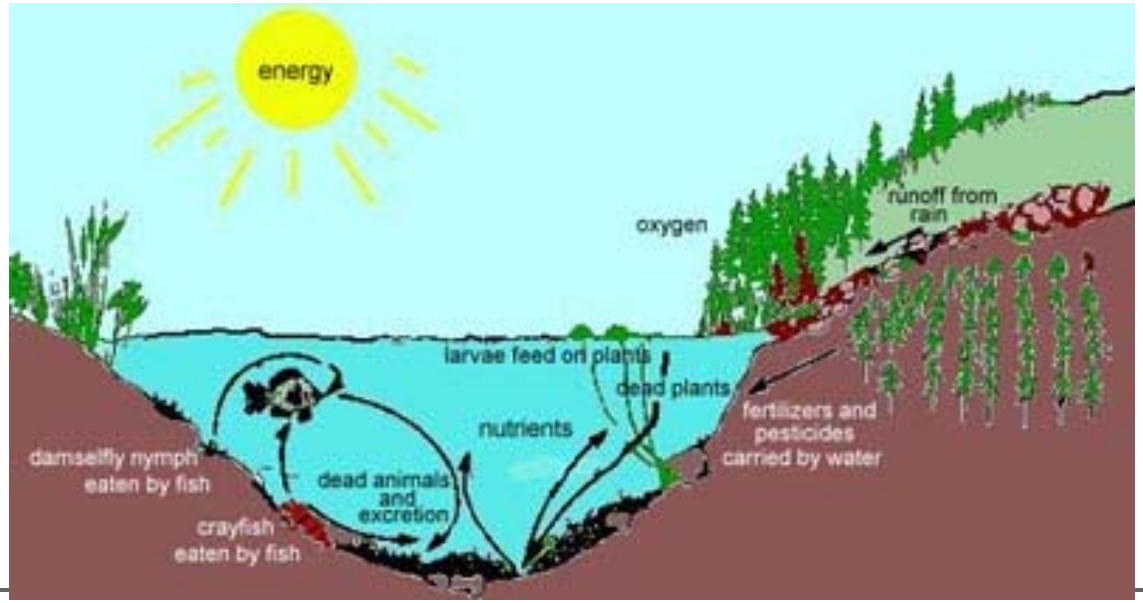
- **Computational Library (ALF w/ IBM)**
  - Task & work block queuing & management
  - Streaming & user-defined data partitioning
  - Process management
  - Error handling
- **Communication Library (DaCS w/ IBM)**
  - Data movement & synchronization
  - Process management & synchronization
  - Topology description
  - Error handling
  - First implementation may leverage OpenMPI
- **Longer term**
  - ALF & DaCS support in tools
  - ALF from Opteron ⇒ Cell directly
  - Compilers supporting some of this



“Ecosystem” for High Performance Computing

From the 2005 National Research Council Report on “The Future of Supercomputing”:

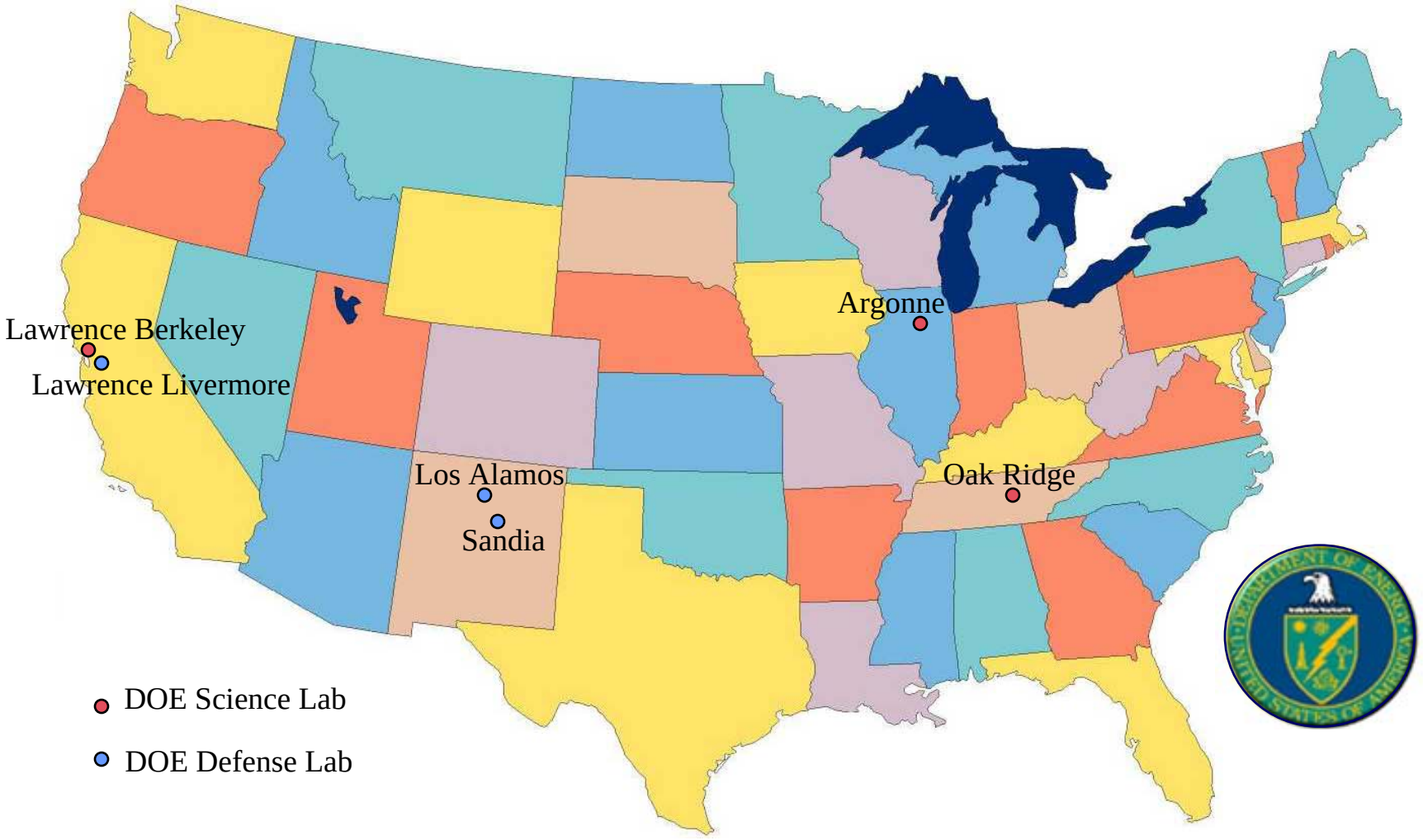
- **Platforms, software, institutions, applications, and people who solve supercomputing applications can be thought of collectively as an ecosystem**
- **Research investment in HPC should be informed by the ecosystem point of view - progress must come on a broad front of interrelated technologies, rather than in the form of individual breakthroughs.**



Pond ecosystem image from http://www.tpwd.state.tx.us/expltx/eft/txwild/pond.htm

c/o Horst Simon, Lawrence Berkeley National Lab

US DOE labs with petascale roadmaps



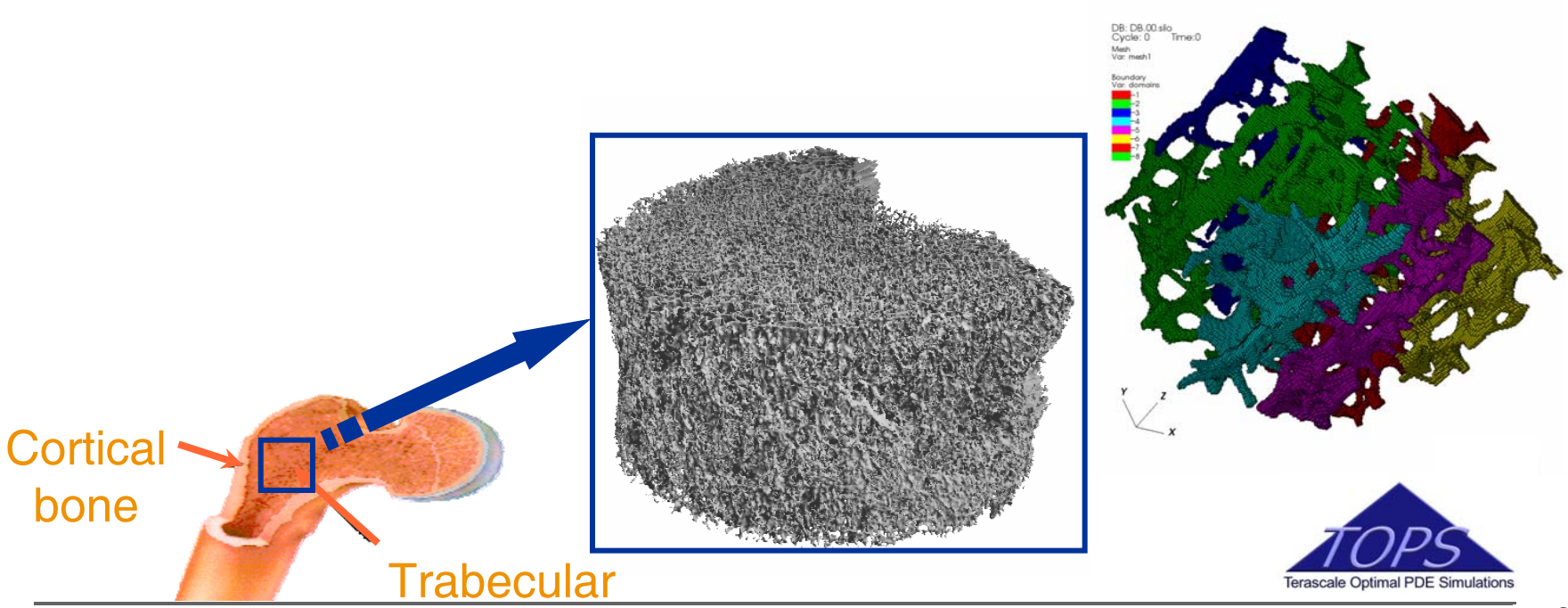
Progress in scaling PDE applications

- Both structured and unstructured grids
- Both explicit and implicit methods
- Fine spatial resolution (through mesh adaptivity)
- Many-thousand-fold concurrency
- Strong scaling within modest ranges
- Weak scaling without obvious limits

See, e.g., Gordon Bell "special" prizes in recent years …

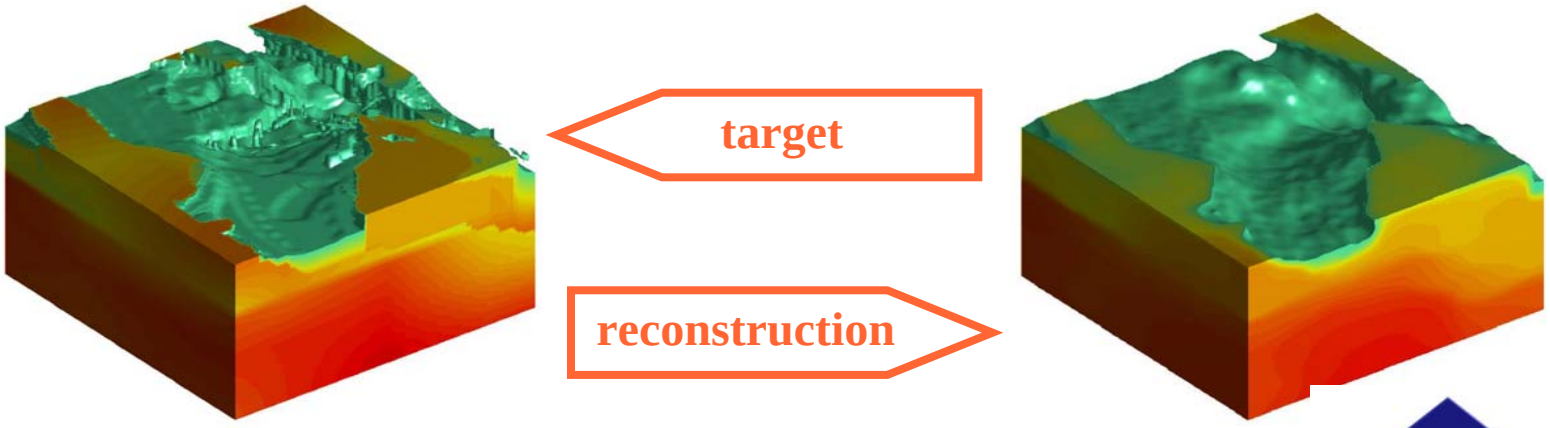
2004 Gordon Bell "special" prize

- 2004 Bell Prize in "special category" went to an implicit, unstructured grid bone mechanics simulation
  - 0.5 Tflop/s sustained on 4 thousand procs of IBM's ASCI White
  - 0.5 billion degrees of freedom
  - large-deformation analysis
  - employed in NIH bone research at Berkeley



2003 Gordon Bell "special" prize

- 2003 Bell Prize in "special category" went to unstructured grid geological parameter estimation problem
  - 1 Tflop/s sustained on 2 thousand processors of HP's "Lemieux
  - each explicit forward PDE solve: 17 million degrees of freedom
  - seismic inverse problem: 70 billion degrees of freedom
  - employed in NSF seismic research at CMU

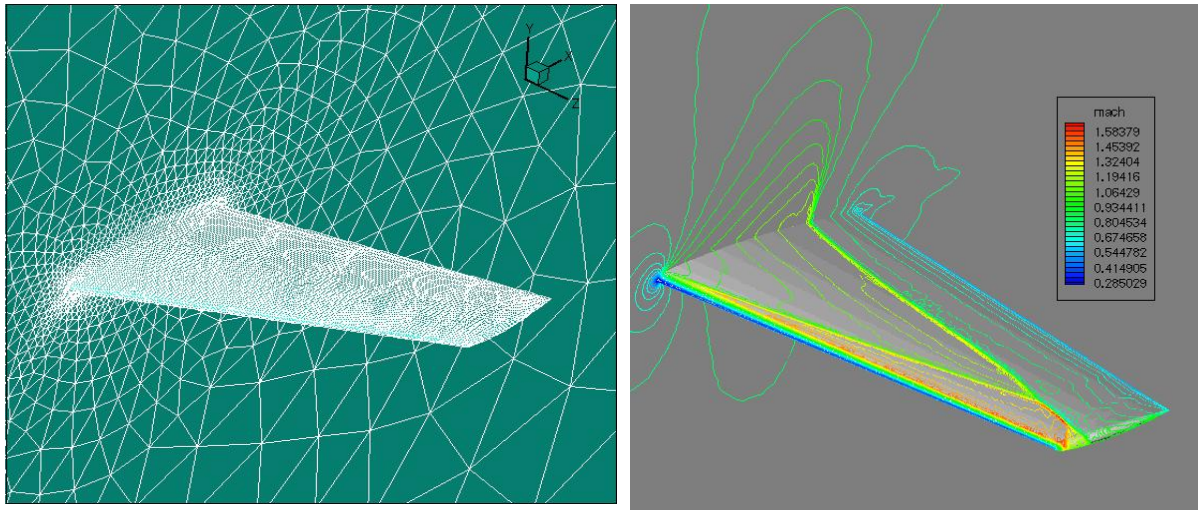


TOPS
Terascale Optimal PDE Simulations

1999 Gordon Bell "special" prize

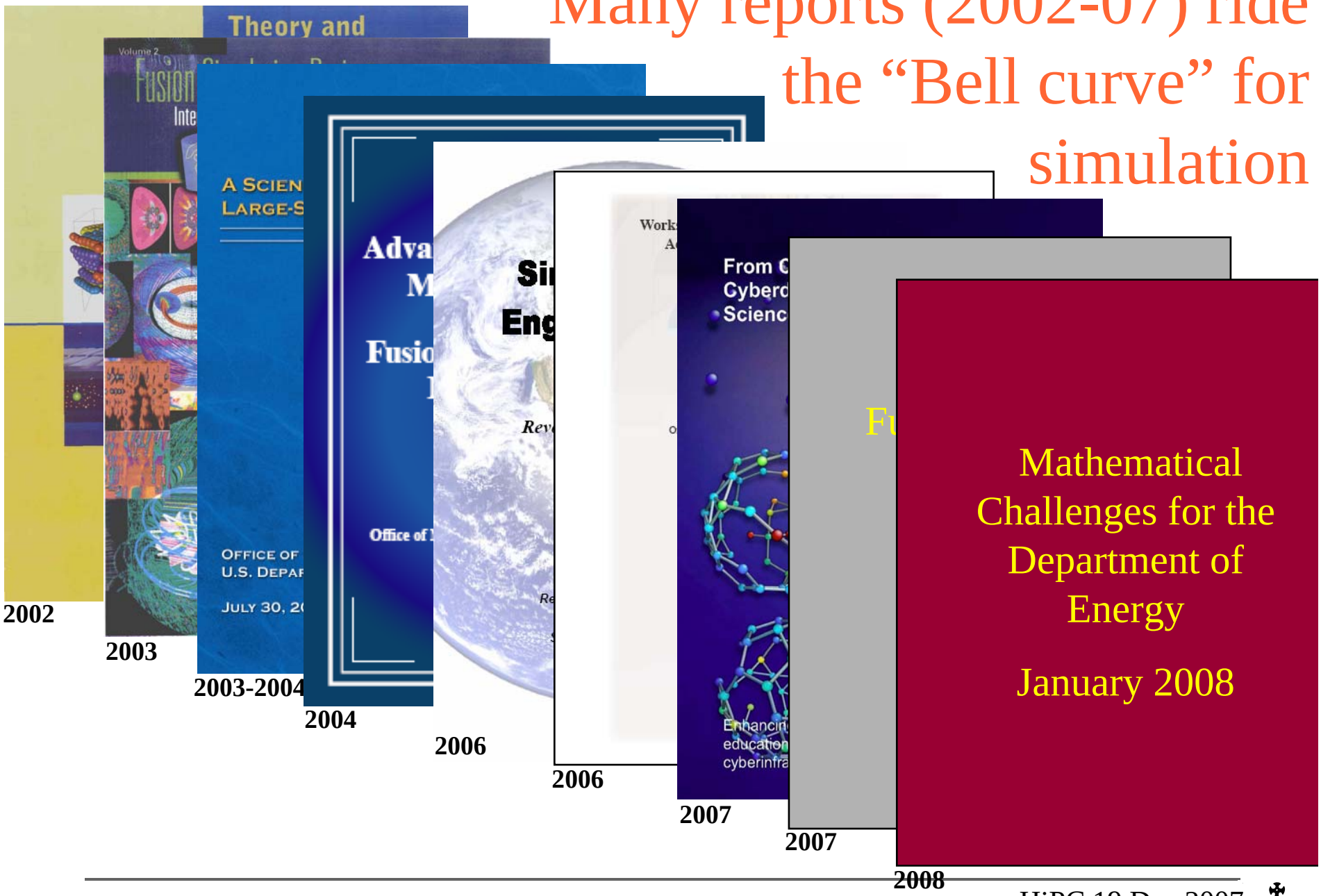
- 1999 Bell Prize in "special category" went to implicit, unstructured grid aerodynamics problems
  - 0.23 Tflop/s sustained on 3 thousand processors of Intel's ASCI Red
  - 11 million degrees of freedom
  - incompressible and compressible Euler flow
  - employed in NASA analysis/design missions

Transonic "Lambda" Shock, Mach contours on surfaces

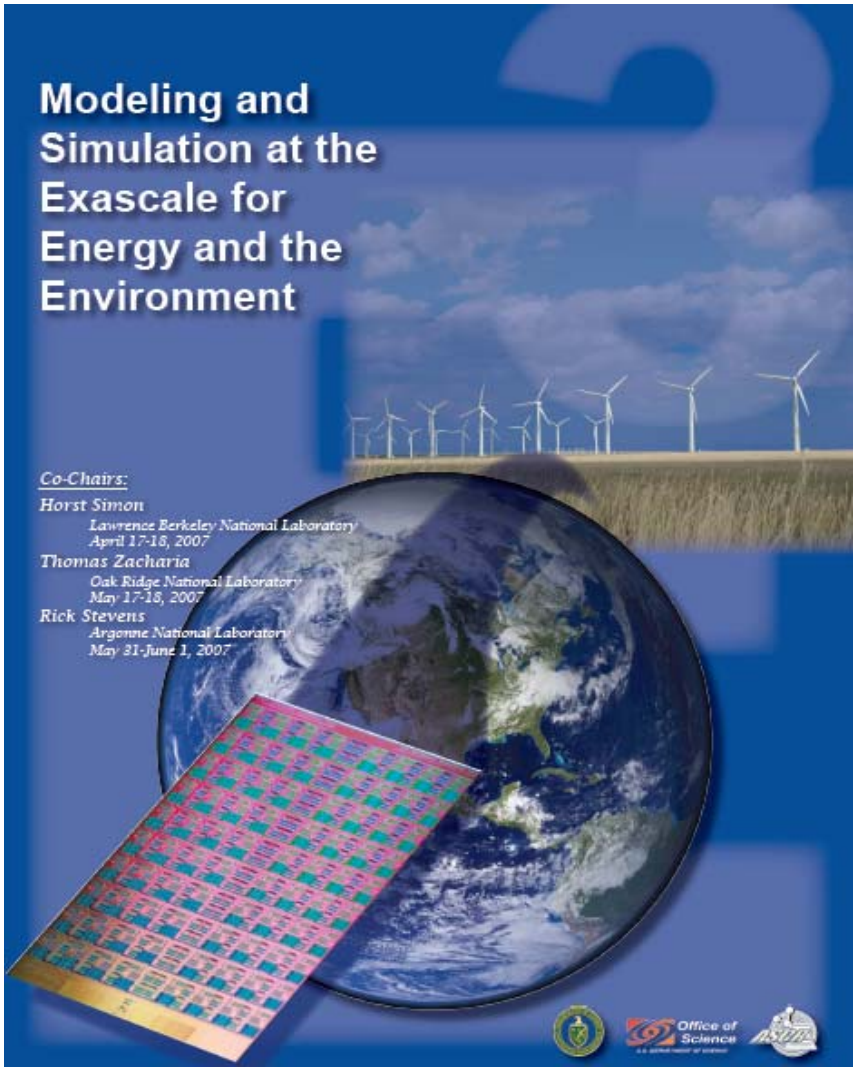


TOPS
Terascale Optimal PDE Simulations

Many reports (2002-07) ride the "Bell curve" for simulation



Recent "E^3" report

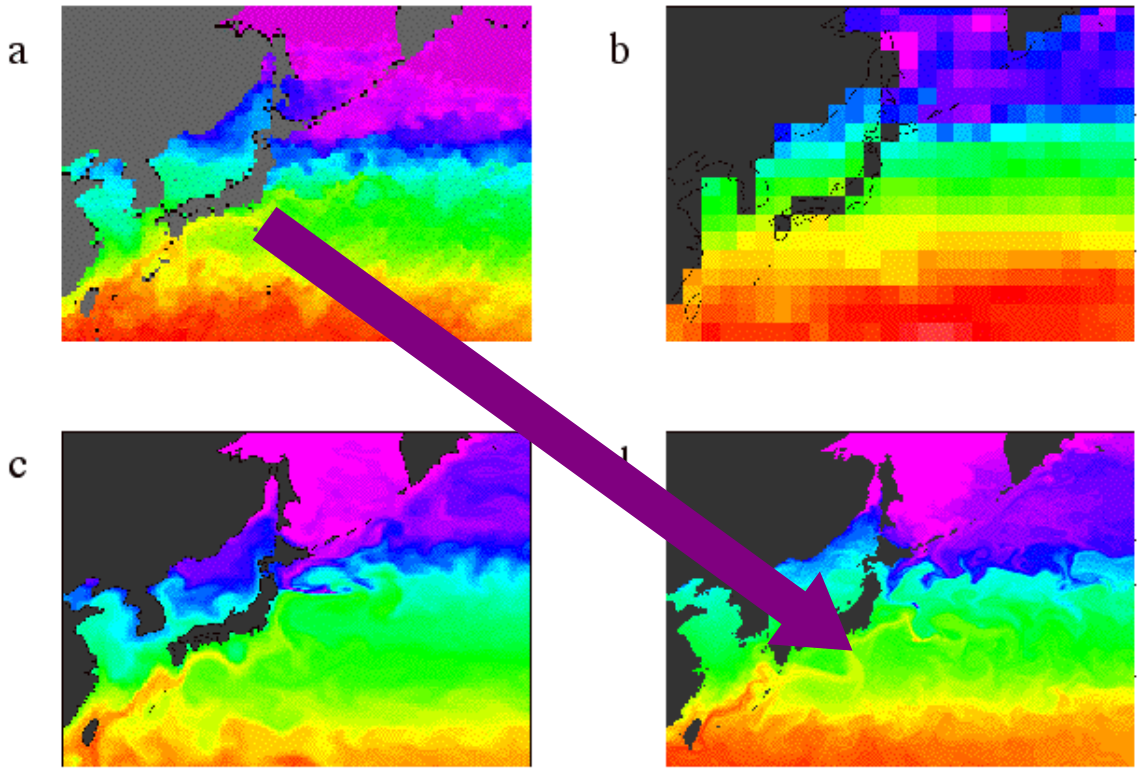


- **Chapter 1. *Climate***
- **Chapter 2. *Combustion, fusion and fission energy technologies***
- **Chapter 3. *Biology***
- **Chapter 4. *Socio-economic modeling***
- **Chapter 5. *Astrophysics***
- **Chapter 6. *Mathematics***
- **Chapter 7. *Software***
- **Chapter 8. *Hardware***
- **Chapter 9. *Cyberinfrastructure****

*** Support for distributed virtual organizations, workflow management, data management, cyber security**

What would we do with 100-1000x more?
Example: predict future climates

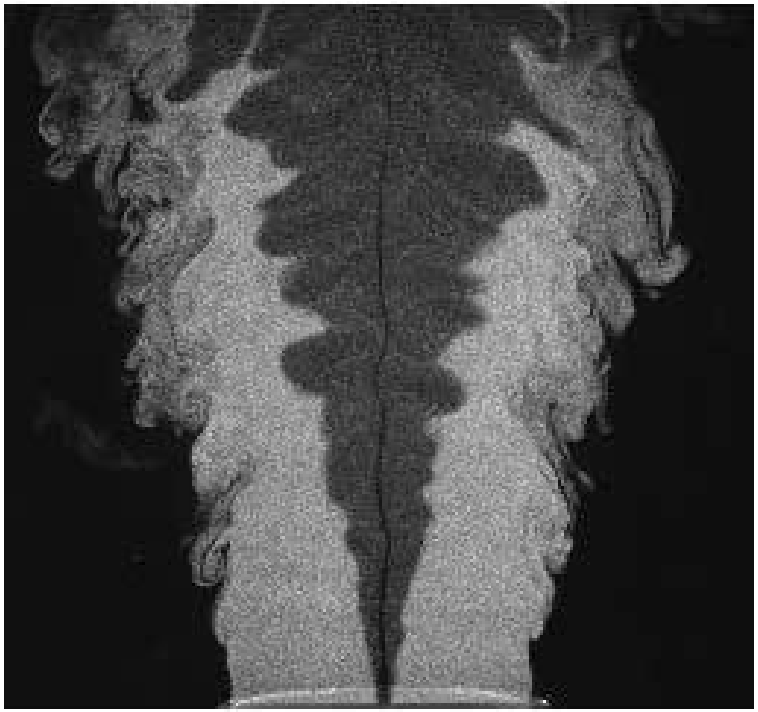
Resolution of Kuroshio Current: Simulations at various resolutions have demonstrated that, because equatorial meso-scale eddies have diameters ~10-200 km, the grid spacing must be < 10 km to adequately resolve the eddy spectrum. This is illustrated in four images of the sea-surface temperature. Figure (a) shows a snapshot from satellite observations, while the three other figures are snapshots from simulations at resolutions of (b) 2°, (c) 0.28°, and (d) 0.1°.



What would we do with 100-1000x more? *Example:* predict future climates

- Resolution
  - refine horizontal atmospheric scale from 160 to 40 km
  - refine horizontal ocean scale from 105 to 7.5km
- New physics
  - atmospheric chemistry
  - carbon cycle (currently, carbon release is external driver)
  - dynamic terrestrial vegetation (nitrogen and sulfur cycles and land-use and land-cover changes)
- Improved representation of subgrid processes
  - clouds
  - atmospheric radiative transfer

What would we do with 100-1000x?
Example: control combustion



Experimental PIV measurement

Instantaneous flame front imaged by density of inert marker

Simulation

Instantaneous flame front imaged by fuel concentration

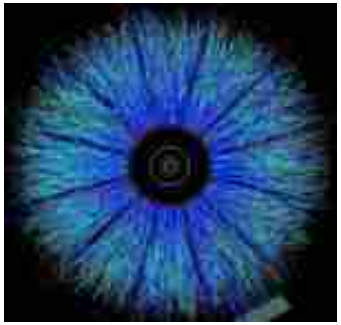
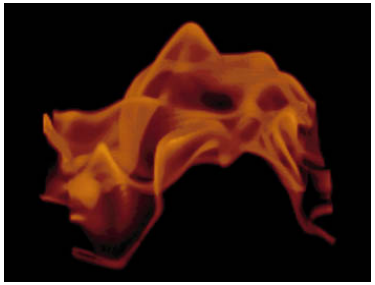
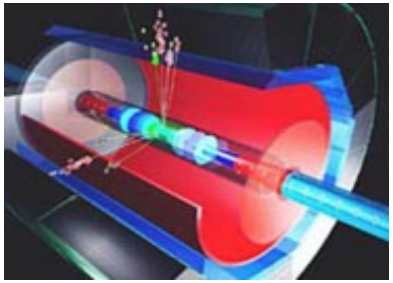
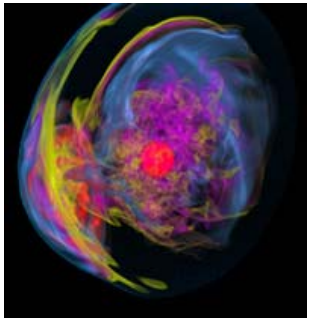
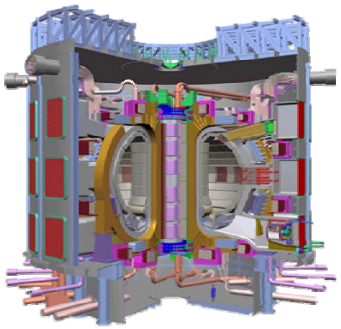
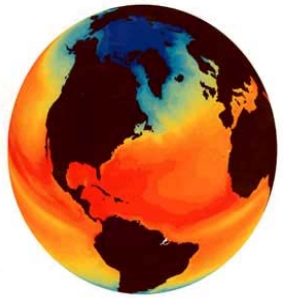
Images c/o R. Cheng (left), J. Bell (right), LBNL, and NERSC

2003 SIAM/ACM Prize in CS&E (J. Bell & P. Colella)

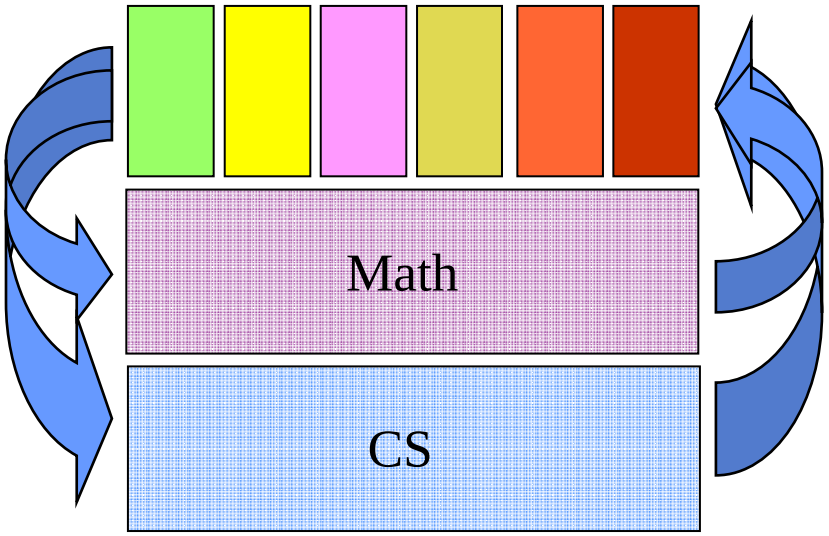
What would we do with 100-1000x? *Example*: control combustion

- Resolution
  - evolve 3D time-dependent large-eddy simulation (LES) codes to direct Navier-Stokes (DNS; work ~ $Re^{11/4}$)
  - multi-billions of mesh zones required
- New "physics"
  - explore coupling between chemistry and acoustics (currently filtered out)
  - explore sooting mechanisms to capture radiation effects
  - capture autoignition with realistic fuels
- Integrate with experiments
  - pioneer simulation-controlled experiments to look for predicted effects in the laboratory

Common support of PDE-based applications: philosophy of the US DOE SciDAC program

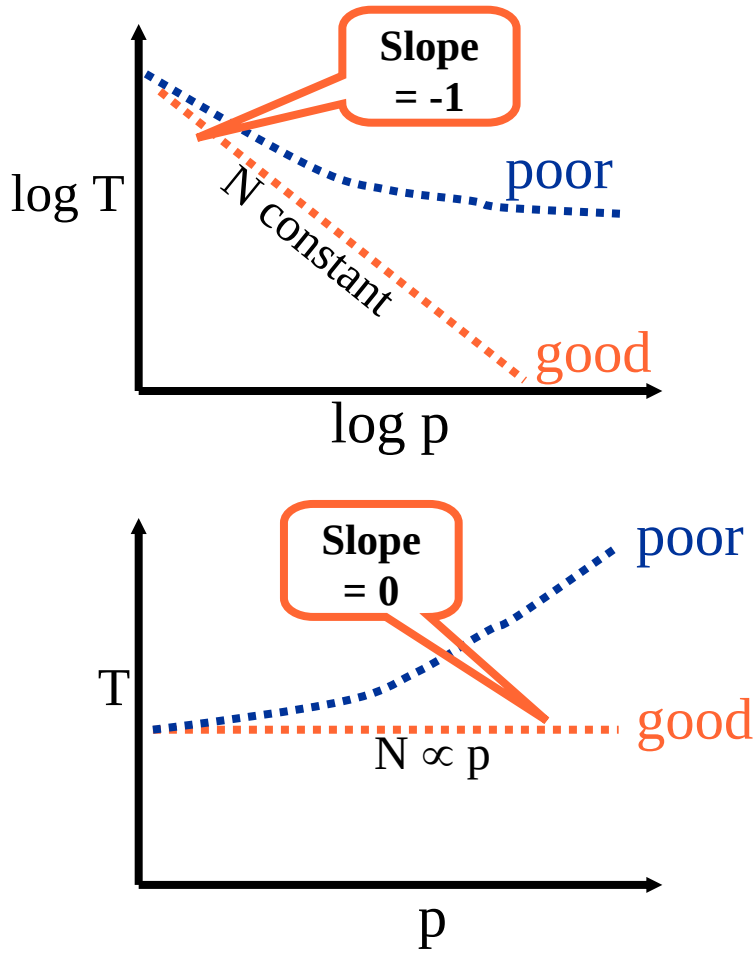
Applications drive



Enabling technologies respond

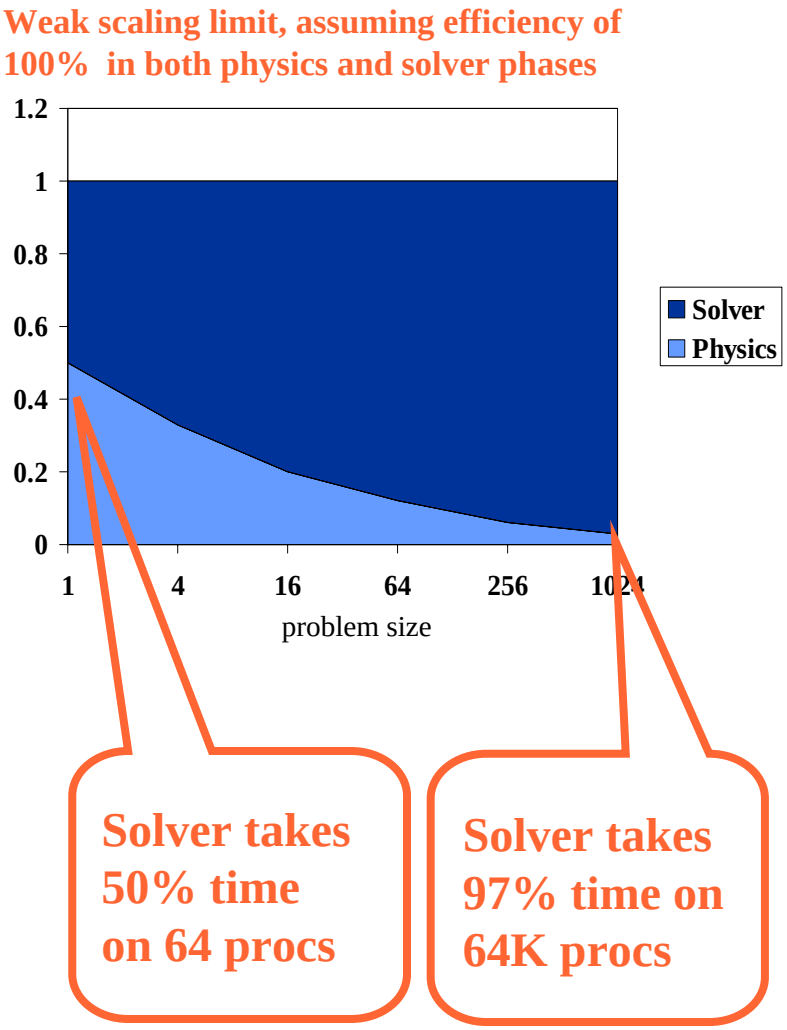
Review: two definitions of scalability

- "Strong scaling"
  - execution time decreases in inverse proportion to the number of processors
  - *fixed size problem overall*
  - often instead graphed as reciprocal, "speedup"
- "Weak scaling"
  - execution time remains constant, as problem size and processor number are increased in proportion
  - *fixed size problem per processor*
  - Various sub-types of weak-scaling "memory bound", etc. (see Kumar et al.)



It's *all* about the solver (at the tera-/peta-scale)

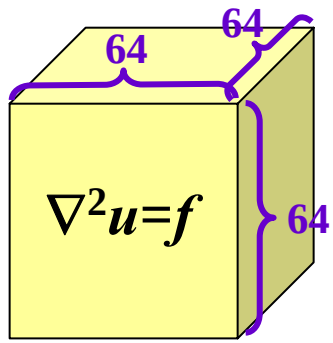
- Given, for example:
  - a "physics" phase that scales as $O(N)$
  - a "solver" phase that scales as $O(N^{3/2})$
  - computation is almost all solver after several doublings
- Most applications groups have not yet "felt" this curve in their gut
  - BG/L will change this
  - 64K-processor machine delivered in 2005



The power of optimal algorithms

- Advances in algorithmic efficiency can rival advances in hardware architecture
- Consider Poisson's equation on a cube of size $N=n^3$

Year	*Method*	*Reference*	*Storage*	*Flops*
1947	GE (banded)	Von Neumann & Goldstine	n^5	n^7
1950	Optimal SOR	Young	n^3	$n^4 \log n$
1971	CG	Reid	n^3	$n^{3.5} \log n$
1984	Full MG	Brandt	n^3	n^3

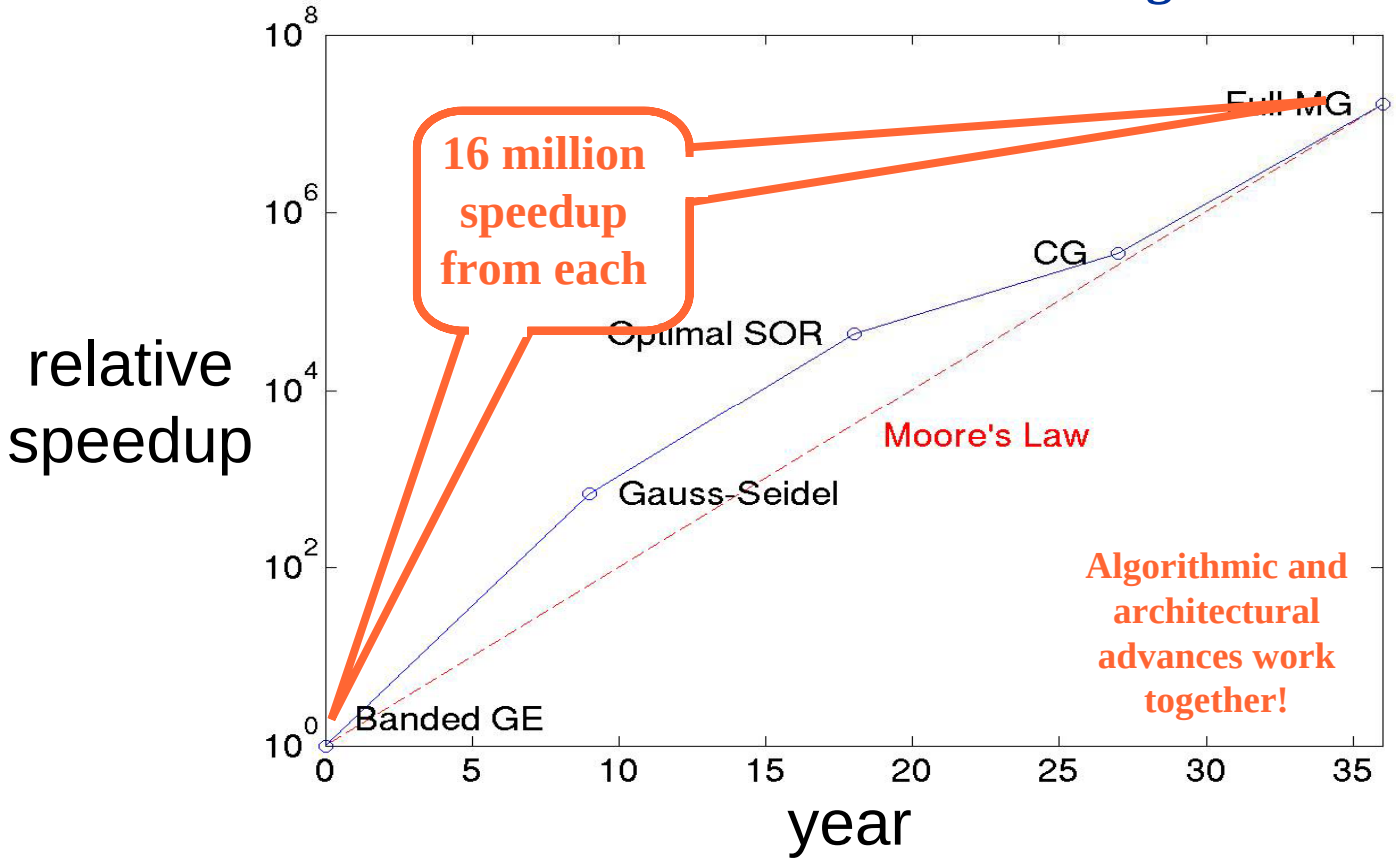


- If $n=64$, this implies an overall reduction in flops of ~16 million *

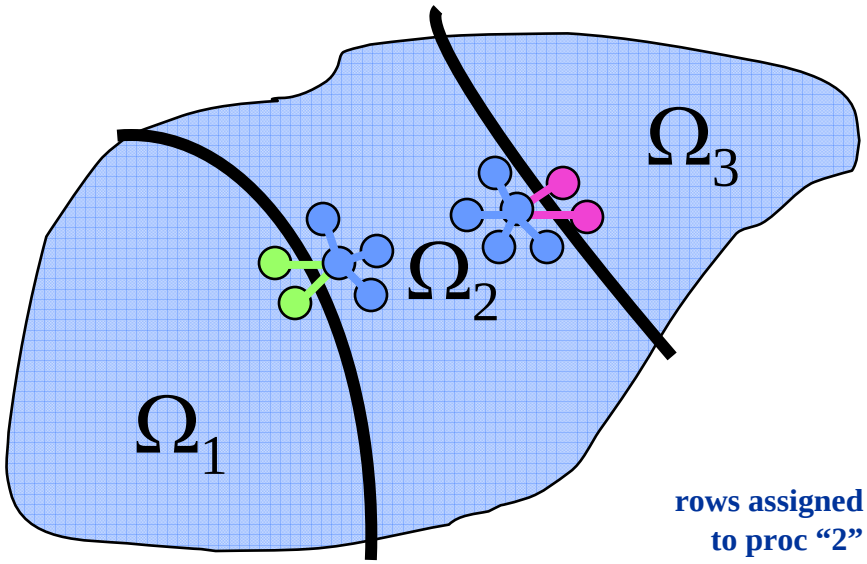
*Six-months is reduced to 1 s

Algorithms and Moore's Law

- **This advance took place over a span of about 36 years, or 24 doubling times for Moore's Law**
- **$2^{24} \approx 16$ million $\Rightarrow$ the same as the factor from algorithms alone!**

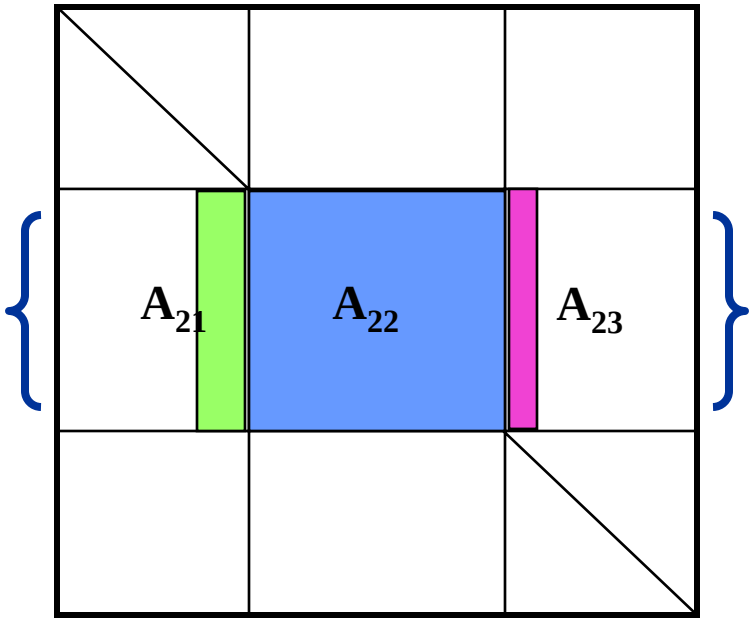


SPMD parallelism w/domain decomposition



rows assigned
to proc "2"

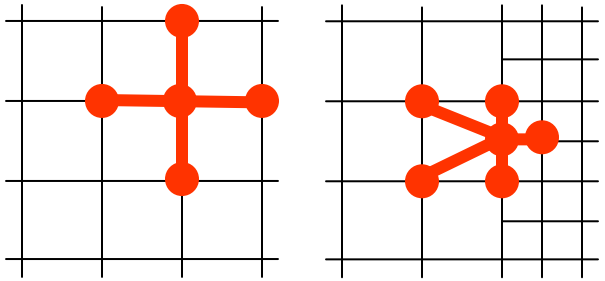
(volume) work to (surface) communication is preserved under weak scaling



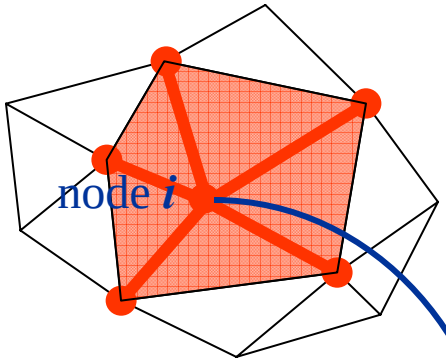
Partitioning of the grid induces block structure on the system matrix (Jacobian)

DD relevant to any local stencil formulation

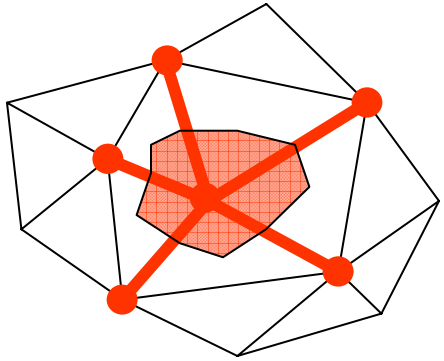
finite differences

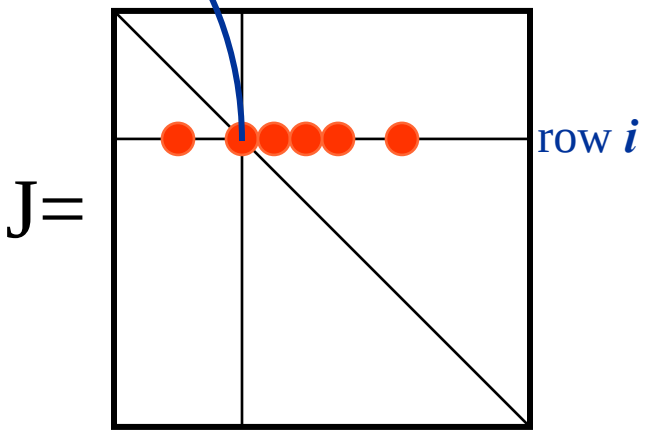


finite elements



finite volumes





- **All lead to sparse Jacobian matrices**
- **However, the inverses are generally dense; even the factors suffer unacceptable fill-in in 3D**
- **Want to solve in subdomains only, and use to precondition full sparse problem**

An algorithm for PDE simulation: Newton-Krylov-Schwarz

Newton

nonlinear solver

asymptotically quadratic

Krylov

accelerator

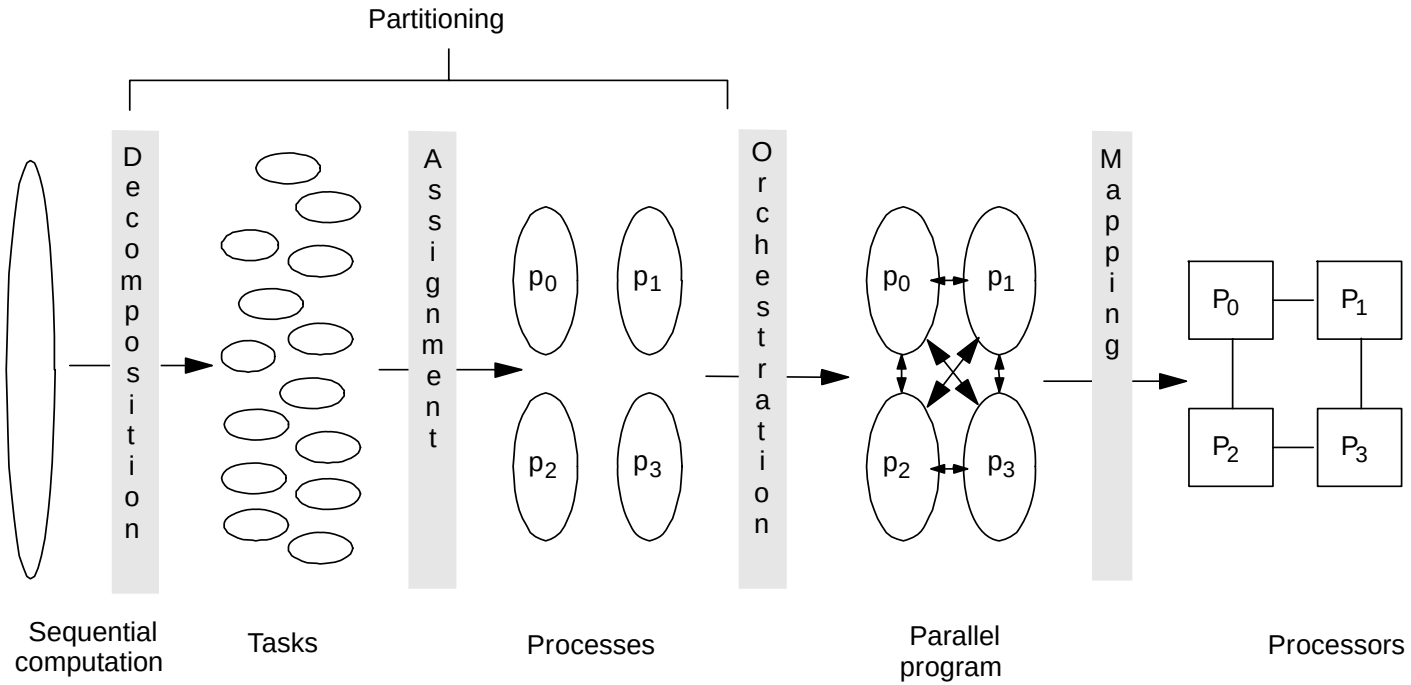
spectrally adaptive

Schwarz

preconditioner

parallelizable

Four steps in creating a parallel program

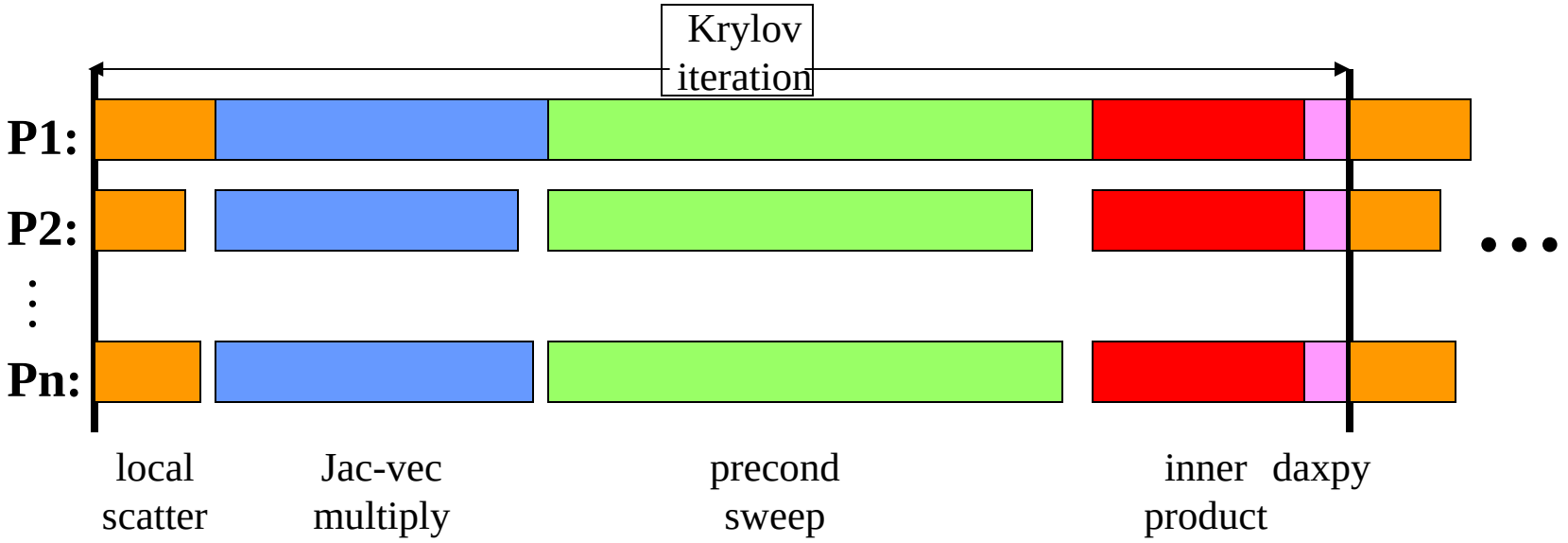


- Decomposition of computation in tasks
- Assignment of tasks to processes
- Orchestration of data access, communication, synchronization
- Mapping processes to processors

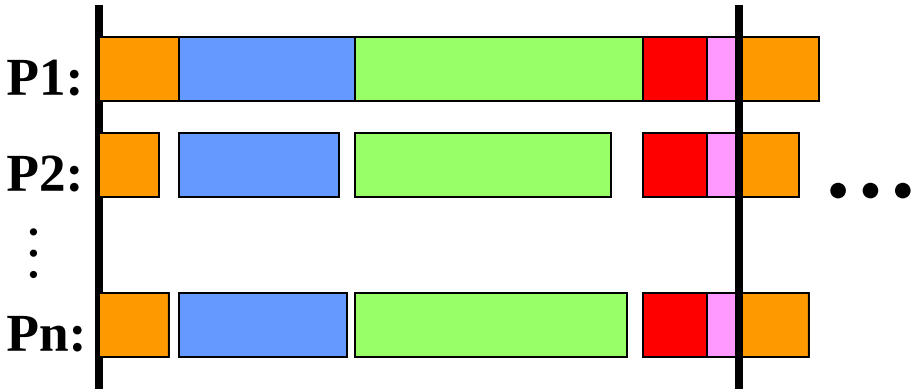
Krylov-Schwarz parallelization is simple!

- Decomposition into concurrent tasks
  - by domain
- Assignment of tasks to processes
  - typically one subdomain per process
- Orchestration of communication between processes
  - to perform sparse matvec – *near neighbor communication*
  - to perform subdomain solve – *nothing*
  - to build Krylov basis – *global inner products*
  - to construct best fit solution – *global sparse solve (redundantly?)*
- Mapping of processes to processors
  - typically one process per processor

Inner Krylov-Schwarz kernel in parallel: a Bulk Synchronous Process ("BSP")

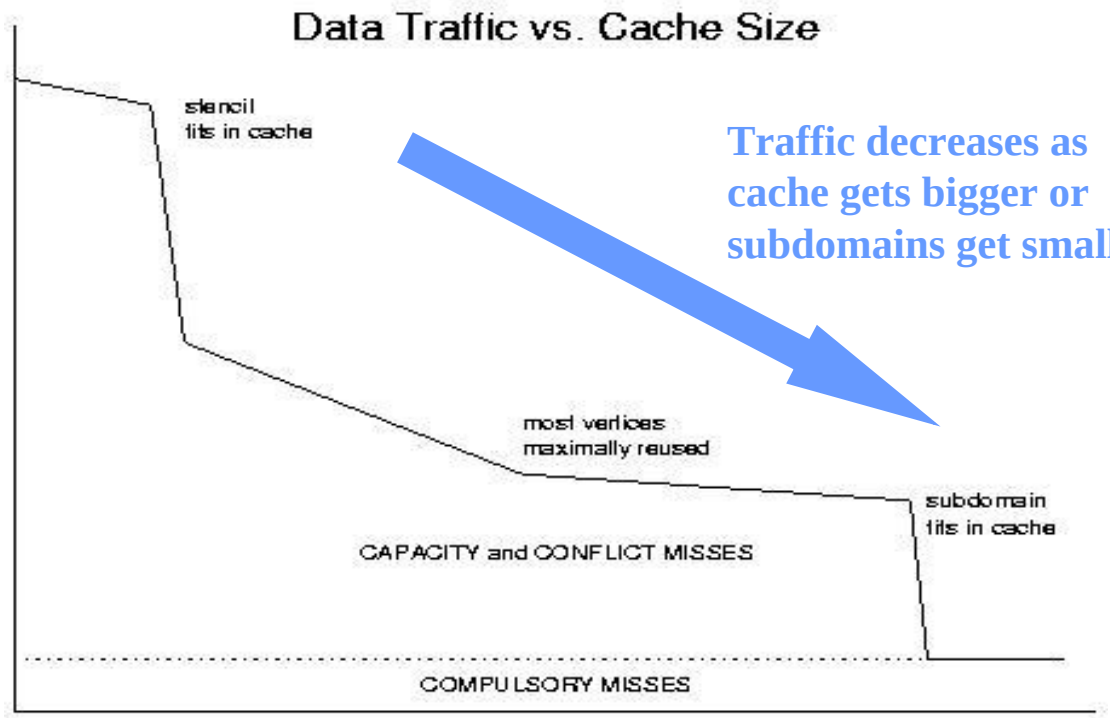


What happens if, for instance, in this (schematicized) iteration, arithmetic speed is ***doubled***, scalar all-gather is ***quartered***, and local scatter is ***cut by one-third***? Each phase is considered separately. Answer is to the right.



Krylov-Schwarz compelling in serial, too

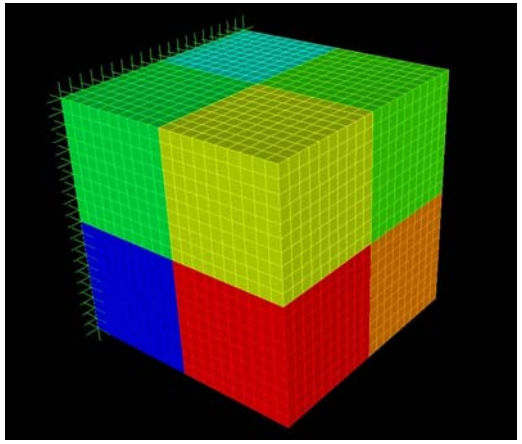
- As successive workingsets “drop” into a level of memory, capacity (and with effort conflict) misses disappear, leaving only compulsory misses, reducing demand on main memory bandwidth
- Cache size is not easily manipulated, but domain size *is*



Estimating scalability of stencil computations

- Given complexity estimates of the leading terms of:
  - the concurrent computation (per iteration phase)
  - the concurrent communication
  - the synchronization frequency
- And a bulk synchronous model of the architecture including:
  - internode communication (network topology and protocol reflecting horizontal memory structure)
  - on-node computation (effective performance parameters including vertical memory structure)
- One can estimate optimal concurrency and optimal execution time
  - on per-iteration basis, or overall (by taking into account any granularity-dependent convergence rate)
  - simply differentiate time estimate in terms of *(N,P)* with respect to *P*, equate to zero and solve for *P* in terms of *N*

Estimating 3D stencil costs (per iteration)



- grid points in each direction n, total work $N=O(n^3)$
- processors in each direction p, total procs $P=O(p^3)$
- memory per node requirements $O(N/P)$

- concurrent execution time per iteration $A\, n^3/p^3$
- grid points on side of each processor subdomain n/p
- Concurrent neighbor commun. time per iteration $B\, n^2/p^2$
- cost of global reductions in each iteration $C \log p$ or $C\, p^{(1/d)}$
  - C includes synchronization frequency
- same dimensionless units for measuring A, B, C
  - e.g., cost of scalar floating point multiply-add

3D stencil computation illustration

Rich local network, tree-based global reductions

- total wall-clock time per iteration

$$T(n,p) = A\frac{n^3}{p^3} + B\frac{n^2}{p^2} + C\log p$$

- for optimal $\boldsymbol{p}$, $\frac{\partial T}{\partial p} = 0$, or $-3A\frac{n^3}{p^4} - 2B\frac{n^2}{p^3} + \frac{C}{p} = 0,$

or (with $\theta \equiv \frac{32B^3}{243A^2C}$),

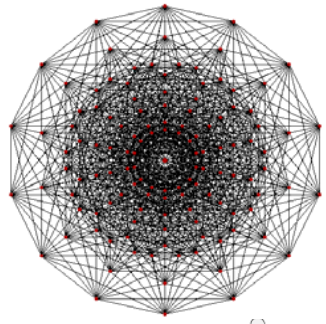
$$p_{opt} = \left(\frac{3A}{2C}\right)^{1/3}\left(\left[1+(1-\sqrt{\theta})\right]^{1/3} + \left[1-(1-\sqrt{\theta})\right]^{1/3}\right)\cdot n$$

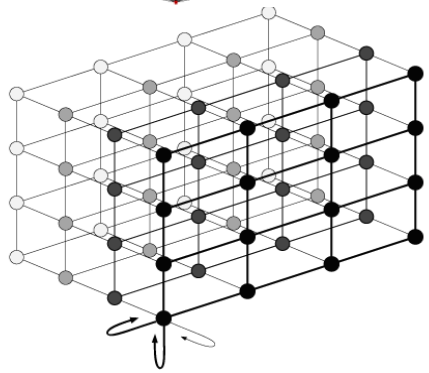
- without "speeddown," $\boldsymbol{p}$ can grow with $\boldsymbol{n}$
- in the limit as $B/C \to 0$

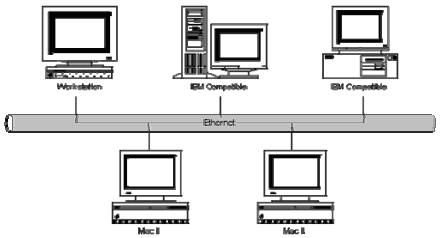
$$p_{opt} = \left(\frac{3A}{C}\right)^{1/3}\cdot n$$

Scalability results for DD stencil computations

- With tree-based (logarithmic) global reductions and scalable nearest neighbor hardware:
  - optimal number of processors scales *linearly* with problem size
- With 3D torus-based global reductions and scalable nearest neighbor hardware:
  - optimal number of processors scales as *three-fourths* power of problem size (almost “scalable”)
- With common network bus (heavy contention):
  - optimal number of processors scales as *one-fourth* power of problem size (not “scalable”)







What's under the rug?

- This generic weak scaling type of argument has been made for ten years
  - in Petaflops Workshop series (1997 onward)
  - in "all-hands" group meetings of SciDAC users (2001 onward)
- Why isn't everyone "humming" on BG/L already?

Contraindications of scalability

- Fixed problem size
  - Amdahl-type constraints
    - "fully resolved" discrete problems (e.g., protein folding, network problems)
    - "sufficiently resolved" problems from the continuum
- Scalable problem size
  - Resolution-limited progress in "long time" integration
    - explicit schemes for time-dependent PDEs
    - suboptimal iterative relaxations schemes for equilibrium PDEs
  - Nonuniformity of threads
    - adaptive schemes
    - multiphase computations (e.g, particle and field)

Amdahl's Law (1967)

- Fundamental limit to strong scaling due to small overheads
- Independent of number of processors available
- Analyze by binning code segments by degree of exploitable concurrency and dividing by available processors, up to limit
- Illustration for just two bins:
  - fraction f_1 of work that is purely sequential
  - fraction $(1\text{-}f_1)$ of work that is arbitrarily concurrent
- Wall clock time for p processors $\propto f_1 + (1 - f_1)/p$
- Speedup $= 1/[f_1 + (1 - f_1)/p]$
  - for f_1=0.01

p	1	10	100	1000	10000
S	1.0	9.2	50.3	91.0	99.0

- Applies to any performance enhancement, not just parallelism

Resolution-limited progress (weak scaling)

- Illustrate for CFL-limited time stepping
- Parallel wall clock time

$$\propto T\, S^{1+\alpha/d}\, P^{\alpha/d}$$

- Example: explicit wave problem in 3D ($\alpha=1$, $d=3$)

Domain	$10^3\times 10^3\times 10^3$	$10^4\times 10^4\times 10^4$	$10^5\times 10^5\times 10^5$
Exe. time	1 day	10 days	3 months

- Example: explicit diffusion problem in 2D ($\alpha=2$, $d=2$)

Domain	$10^3\times 10^3$	$10^4\times 10^4$	$10^5\times 10^5$
Exe. time	1 day	3 months	27 years

d*-dimensional domain, length scale *L

d+1*-dimensional space-time, time scale *T

h mesh cell size

τ time step size

τ=O(h^α) bound on time step

n=L/h number of mesh cells in each dim

N=n^d number of mesh cells overall

M=T/τ number of time steps overall

O(N) total work to perform one time step

O(MN) total work to solve problem

P number of processors

S storage per processor

PS total storage on all processors ***(=N)***

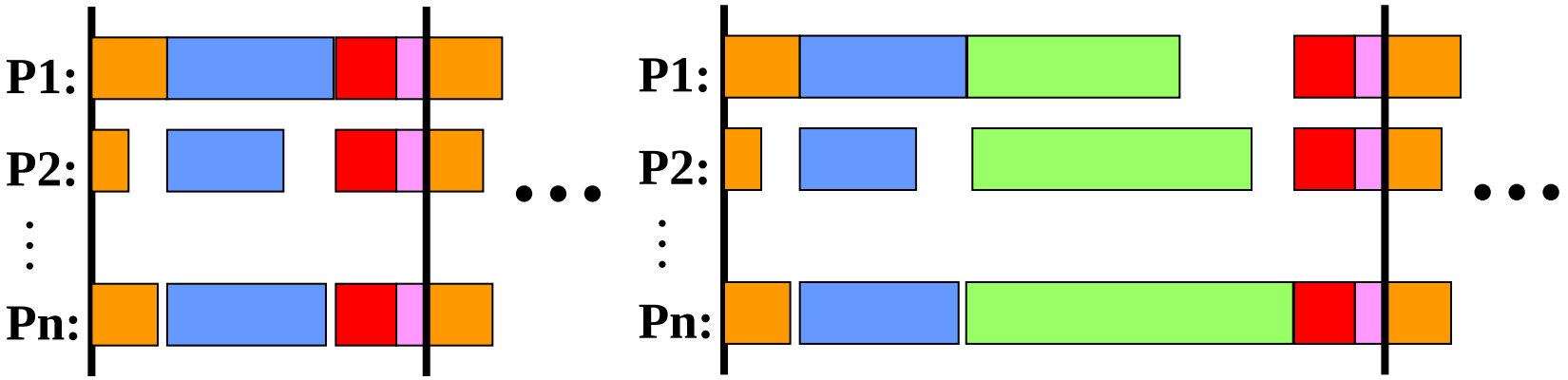
O(MN/P) parallel wall clock time

$\propto (T/\tau)(PS)/P \propto T\, S^{1+\alpha/d}\, P^{\alpha/d}$

(since $\tau \propto h^\alpha \propto 1/n^\alpha = 1/N^{\alpha/d} = 1/(PS)^{\alpha/d}$)

Thread nonuniformity

- Evolving state of the simulation can spoil load balance
  - adaptive scheme
    - local mesh refinement
    - local time adaptivity
  - state-dependent work complexity
    - complex constitutive or reaction terms
    - nonlinear inner loops with variable convergence rates
  - multiphase simulation
    - bulk synchronous alternation between different phases with different work distributions



Algorithmic adaptation

- No computer system is well balanced for *all* computational tasks, or even for all phases of a *single* well-defined task, like solving nonlinear systems arising from discretized differential equations
- Given the need for high performance in the solution of these and related systems, one should be aware of which computational phases are limited by which aspect of hardware or software.
- With this knowledge, one can design algorithms to “play to” the strengths of a machine of given architecture, or one can intelligently select or evolve architectures for preferred algorithms.

Four potential limiters on scalability in large-scale parallel scientific codes

- Insufficient localized concurrency
- Load imbalance at synchronization points
- Interprocessor message latency
- Interprocessor message bandwidth

"horizontal aspects"

Four potential limiters on arithmetic performance

- Memory latency
  - Failure to predict which data items are needed
- Memory bandwidth
  - Failure to deliver data at consumption rate of processor
- Load/store instruction issue rate
  - Failure of processor to issue enough loads/stores per cycle
- Floating point instruction issue rate
  - Low percentage of floating point operations among all operations

"vertical aspects"

Candidate stresspoints of PDE kernels

- Vertex-based loops
  - memory bandwidth
- Edge-based “stencil op” loops
  - load/store (register-cache) bandwidth
  - internode bandwidth
- Sparse, narrow-band recurrences
  - memory bandwidth
  - internode bandwidth, internode latency, network diameter
- Inner products and norms
  - memory bandwidth
  - internode latency, network diameter

Summary of observations for CFD case study (aerodynamics simulation – 1999 Bell Prize)

- Processor scalability is *no problem*, in principle
  - if network is richly connected
- For fixed-size problems, global synchronization and near neighbor communication are eventually bottlenecks (strong scaling)
- Coarse grids in hierarchical solvers can become bottlenecks
  - coarse grid concurrency may need to be coarser than fine grid concurrency (recur: multigrid)
- Memory latency is not a serious problem, in principle
  - due to predictability of memory transfers in PDEs
- Memory bandwidth is a *major* bottleneck
- Processor Load-Store functionality *may* be a bottleneck
- Infrequency of floating point instructions in unstructured problems *may* be a bottleneck

Some noteworthy algorithmic adaptations to distributed memory architecture

- Restricted Schwarz (Cai & Sarkis)
  - omit every other local communication (actually leads to *better* convergence, now proved)
- Extrapolated Schwarz (Garbey & Tromeur-Dervout)
  - hide interprocessor latency by extrapolating messages received in time integration, with rollback if actual messages have discrepancies in lower Fourier modes (higher mode discrepancies decay anyway)
- Nonlinear Schwarz (Cai & Keyes)
  - reduce global Krylov-Schwarz synchronizations by applying NKS within well-connected subdomains and performing *few* global outer Newton iterations (interchange of loops, move synchronization outside)
- Aggressive coarsening in AMG (Falgout, Yang, et al.)
  - reduce size of coarse problems to trade-off cost per iteration with number of iterations (and many other such preconditioner quality ideas)

Algebraic multigrid on BG/L

- Algebraic multigrid a key algorithmic technology
  - Discrete operator defined for finest grid by the application, itself, *and* for many recursively derived levels with successively fewer degrees of freedom, for solver purposes
  - Unlike geometric multigrid, AMG not restricted to problems with "natural" coarsenings derived from grid alone
- Optimality (cost per cycle) intimately tied to the ability to coarsen aggressively
- Convergence scalability (number of cycles) and parallel efficiency also sensitive to rate of coarsening
- While much research and development remains, multigrid will clearly be practical at BG/L-scale concurrency

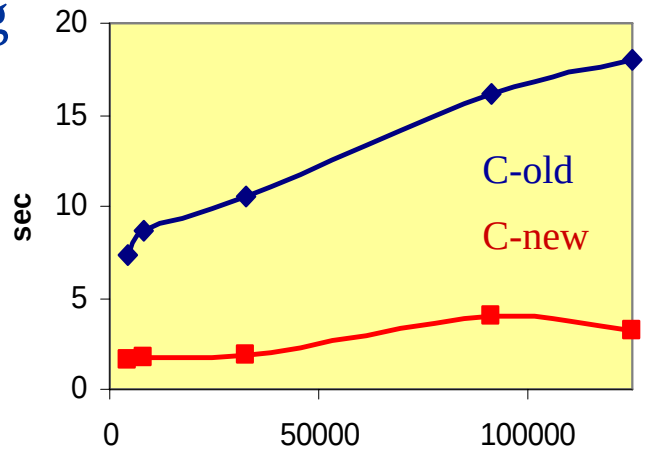


Figure shows weak scaling result for AMG out to 131,072 processors, with one 25× 25×25 block per processor (from 15.6K dofs up to 2.05B dofs)

Some noteworthy algorithmic adaptations to hierarchical memory architecture

- ATLAS/Sparsity (Whalley & Dongarra, Demmel & Yelick)
  - block (and and selectively fill and reorder for sparse) for optimal cache performance of linear kernels
- Block-vector Krylov methods (Baker *et al.*)
  - amortize the unavoidable streaming of large sparse Jacobian through cache over several matrix-vector multiplies
- Block relaxation methods (Douglas *et al.*)
  - similar to above, but for triangular backsolves
- Reduced precision preconditioning (Smith *et al.*)
  - double effective bandwidth by truncating precision of already approximate operators

Adaptation to asynchronous programming styles

- 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 more asynchronously, because we have limited language expressiveness
  - Jacobian and preconditioner refresh
  - Convergence testing
  - Algorithmic parameter adaptation
  - I/O, compression
  - Visualization, data mining
- See Browne, others, on “associative communication”

Often neglected algorithmic possibilities for more scalability

- Parallelization in the time (or generally causal) dimension, particularly in nonlinear problems after spatial concurrency is exhausted
- Creating independent ensembles for asynchronous evaluation (parameter exploration or stochastic model) after space-time concurrency is exhausted on the direct problem
- Trading finely resolved discretizations (very sparse) for higher-order discretizations (block dense), or other algorithmic innovations that alter the granularity of bulk synchronous work between data movements

From *SIAM News*, Volume 39, Number 7, September 2006

Taking on the ITER Challenge, Scientists Look to Innovative Algorithms, Petascale Computers

By Michelle Sipics

The promise of fusion as a clean, self-sustaining and essentially limitless energy source has become a mantra for the age, held out by many scientists as a possible solution to the world's energy crisis and a way to reduce the amounts of greenhouse gases released into the atmosphere by more conventional sources of energy. If self-sustaining fusion reactions can be realized and maintained long enough to produce electricity, the technology could potentially revolutionize energy generation and use.

ITER, initially short for International Thermonuclear Experimental Reactor, is now the official, non-acronymic name (meaning "the way" in Latin) of what is undoubtedly the largest undertaking of its kind. Started as a collaboration between four major parties in 1985, ITER has evolved into a seven-party project that finally found a physical home last year, when it was announced that the ITER fusion reactor would be built in Cadarache, in southern France. (The participants are the European Union, Russia, Japan, China, India, South Korea, and the United States.) In May, the seven initialed an agreement documenting the negotiated terms for the construction, operation, and decommissioning of the ITER tokamak, signifying another milestone for both the project itself and its eventual goal of using fusion to facilitate large-scale energy generation for the world.

Problems remain, however—notably the years, and perhaps decades, of progress needed to attain such a goal. In fact, even *simulating* the proposed ITER tokamak is currently out of reach. But according to David Keyes, a computational mathematician at Columbia University and acting director of the Institute for Scientific Computing Research (ISCR) at Lawrence Livermore National Laboratory, the ability to perform such simulations may be drawing closer.

Hardware 3, Software 9

"Fusion scientists have been making useful characterizations about plasma fusion devices, physics, operating regimes and the like for over 50 years," Keyes says. "However, to simulate the dynamics of ITER for a typical experimental 'shot' over scales of interest with today's most commonly used algorithmic technologies would require approximately 10^{24} floating-point operations." That sounds bleak, given the 280.6 Tflop/s (10^{12} flops/s) benchmark performance of the IBM BlueGene/L at Lawrence Livermore National Laboratory—as of June the fastest supercomputer in the world. But Keyes is optimistic: "We expect that with proper algorithmic ingenuity, we can reduce this to 10^{15} flops."

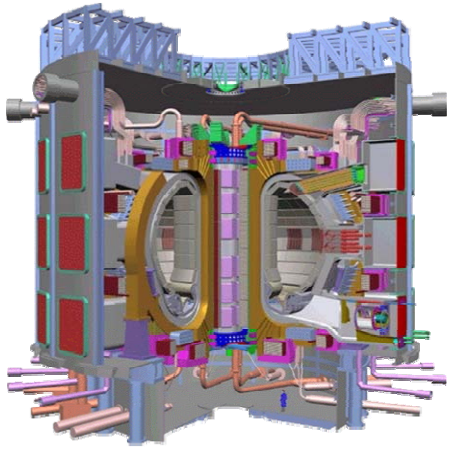
Optimizing the algorithms used, in other words, could lower the computing power required for some ITER simulations by an astounding nine orders of magnitude. Even more exciting, those newly feasible simulations would be at the petascale—ready to run on the petaflop/s supercomputers widely expected within a few years.

The ingenuity envisioned by Keyes even has a roadmap. Together with Stephen Jardin of the Princeton Plasma Physics Laboratory, Keyes developed a breakdown that explains where as many as 12 orders of magnitude of speedup will come from over the next decade: 1.5 from increased parallelism, 1.5 from greater processor speed and efficiency, four from adaptive gridding, one from higher-order elements, one from field-line following coordinates, and three from implicit algorithms.

Scaling fusion simulations up to ITER

name	symbol	units	CDX-U (Small tokamak)	DIII-D (Large tokamak)	ITER (Huge tokamak)
Field	B_0	Tesla	0.22	1	5.3
Minor radius	a	meters	.22	.67	2
Temp.	T_e	keV	0.1	2.0	8.
Lundquist no.	S		1×10^{4}	7×10^{6}	5×10^{8}
Mode growth time	$\tau_A S^{1/2}$	s	2×10^{-4}	9×10^{-3}	7×10^{-2}
Layer thickness	$aS^{-1/2}$	m	2×10^{-3}	2×10^{-4}	8×10^{-5}
zones	$N_R\times N_\theta\times N_\phi$		3×10^{6}	5×10^{10}	3×10^{13}
CFL timestep	$\Delta X/V_A$ (Explicit)	s	2×10^{-9}	8×10^{-11}	7×10^{-12}
Space-time pts			6×10^{12}	1×10^{20}	6×10^{24}

10^{12} needed (explicit uniform baseline)



International Thermonuclear Experimental Reactor

2017 – first experiments, in Cadaraches, France

Where to find 12 orders of magnitude in 10 years?

Hardware: 3

- 1.5 orders: in[illegible]eased p[illegible]ssor spee[illegible]l and efficiency
- 1.5 orders: in[illegible]

Software: 9

- 1 orde[illegible]he
  - Same[illegible] elements
- 1 ord[illegible]
  - Le[illegible]
- 4 order[illegible]
  - Zo[illegible] requir[illegible]lume and resolution requirements away [illegible] the [illegible]s severe
- 3 orders: implicit solvers
  - Mode growth time 9 orders longer than Alfven-limited CFL

Algorithmic improvements bring yottascale (10^{24}) calculation down to petascale (10^{15})!

Comments on JK roadmap

- increased processor speed
  - 10 years is 6.5 Moore doubling times
- increased concurrency
  - BG/L is already 2^{17} procs, MHD now at ca. 2^{12}
- higher-order discretizations
  - low-order FE preconditioning of high-order discretizations (Orszag, Fischer, Manteuffel, etc.)
- flux-surface following gridding
  - evolve mesh to approximately follow flux surfaces
- adaptive gridding
  - adapt mesh to concentrate points in high-gradient regions
- implicit solvers
  - we propose Newton-like fully implicit, with Krylov/MG innards

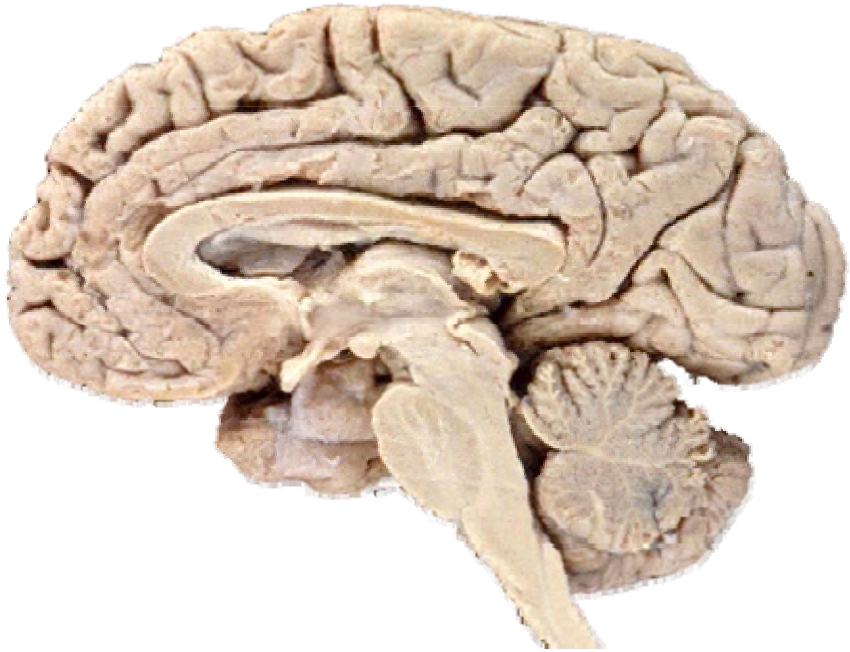
Reminder about the source of simulations

- Computational science and engineering is not about individual large-scale analyses, done fast and "thrown over the wall"
- Both "results" and their sensitivities are desired; often multiple operation points to be simulated are known *a priori*, rather than sequentially
- Sensitivities may be fed back into optimization process
- Full PDE analyses may also be inner iterations in a multidisciplinary computation
- In such contexts, "petaflop/s" may mean 1,000 analyses running somewhat asynchronously with respect to each other, each at 1 Tflop/s – clearly a less daunting challenge and one that has better synchronization properties for exploiting "The Grid" – than 1 analysis running at 1 Pflop/s

Acknowledgments

- DOE ASC and SciDAC programs
- NSF ITR program
- PETSc software team
- Hypre software team

Acknowledgment: today's Peta-op/s machines



10^{12} neurons @ 1 KHz = 1 PetaOp/s