Missing Mathematics for Extreme-scale CFD

David Keyes

Dean, Division of Mathematical and Computer Sciences and Engineering,

King Abdullah University of Science and Technology

&

Department of Applied Physics and Applied Mathematics

Columbia University

Context: decade of promise from tera to peta

These are downloadable; e-mail me if needed

Audience: CFD’ers, not CS’ers

- In the context of the NASA mission, we in computer science and applied mathematics are by federal parlance “enabling technologists”
  - modeling, numerical algorithms, discrete algorithms, visualization, programming models, etc.
- We are often “first marines on the beach” with respect to “extreme” computer architectures
  - vector, distributed memory, shared memory, heterogeneous
- Relevance to “non-extremists”: the extreme architectures of today are lab-group machines in a decade

Simulation driven by price and capability

By the Gordon Bell Prize, simulation *cost per performance* has improved by nearly a million times in two decades. Performance on *real applications* (e.g., mechanics, materials, petroleum reservoirs, gravitation) has improved *more* than a million times.

Gordon Bell Prize: Price Performance **Year**	**Cost per delivered Gigaflop/s**
1989	$2,500,000
1999	$6,900
2009	$8

Gordon Bell Prize: Peak Performance **Year**	**Gigaflop/s delivered to applications**
1988	1
1998	1,020
2008	1,350,000

Thought experiment:
How to use peanuts as price per ton falls?*

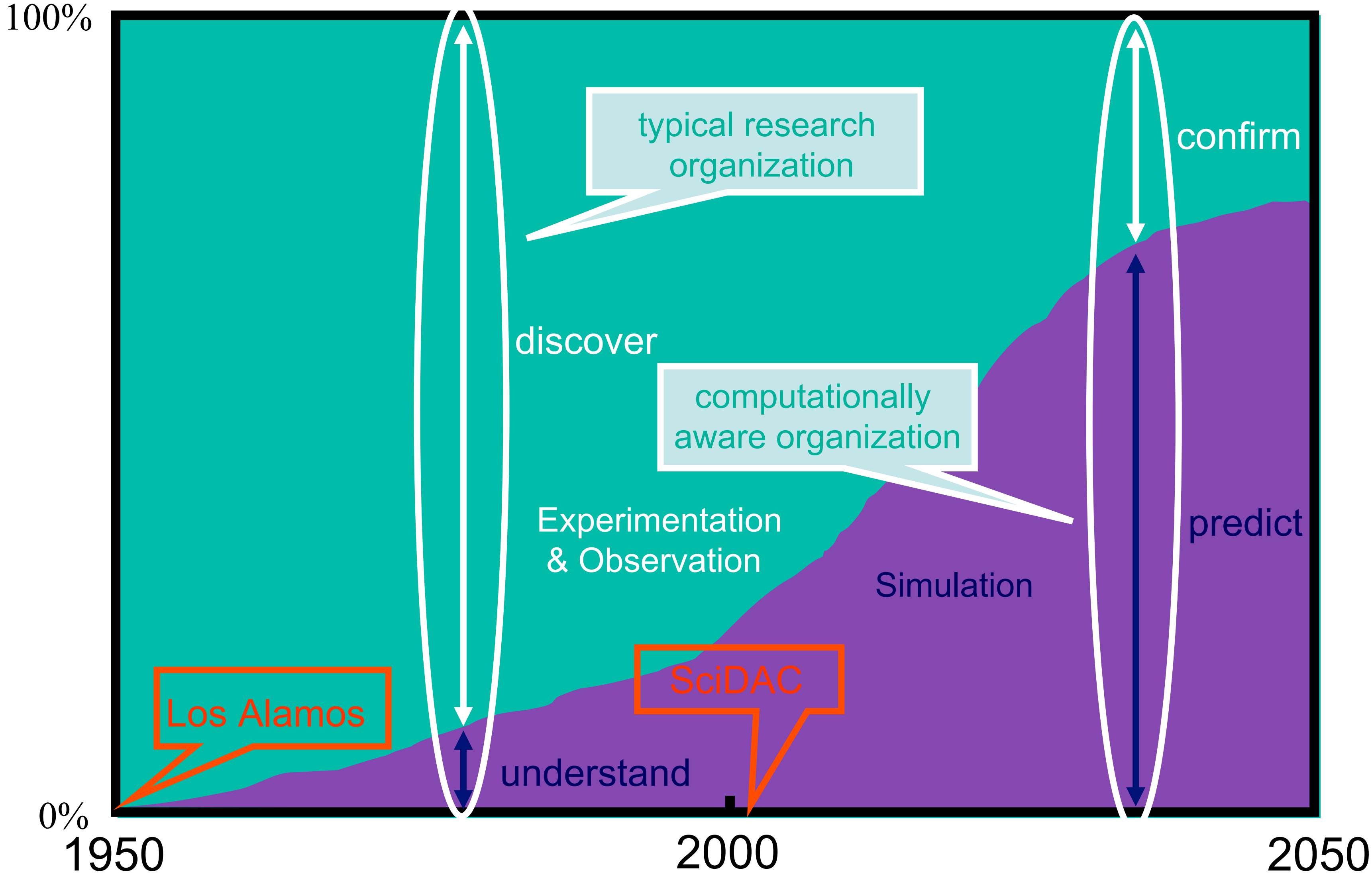
- In 2012, at $1,150./ton:
  - make sandwiches
- By 2015, at $115./ton:
  - make recipe substitutions
- By 2018, at $11.50/ton:
  - use as feedstock for plastics, etc.
- By 2021, at $1.15/ton:
  - heat homes
- By 2024, at $0.115/ton:
  - pave roads ☺

NISTSRM 2387
1028
STANDARD REFERENCE MATERIAL®
2387
Peanut Butter
www.nist.gov/srm
NIST
National Institute of Standards and Technology
Technology Administration, U.S. Department of Commerce

The cost of computing has been on a curve like this for two decades and promises to continue. Like everyone else, scientists and engineers plan increasing uses for it…

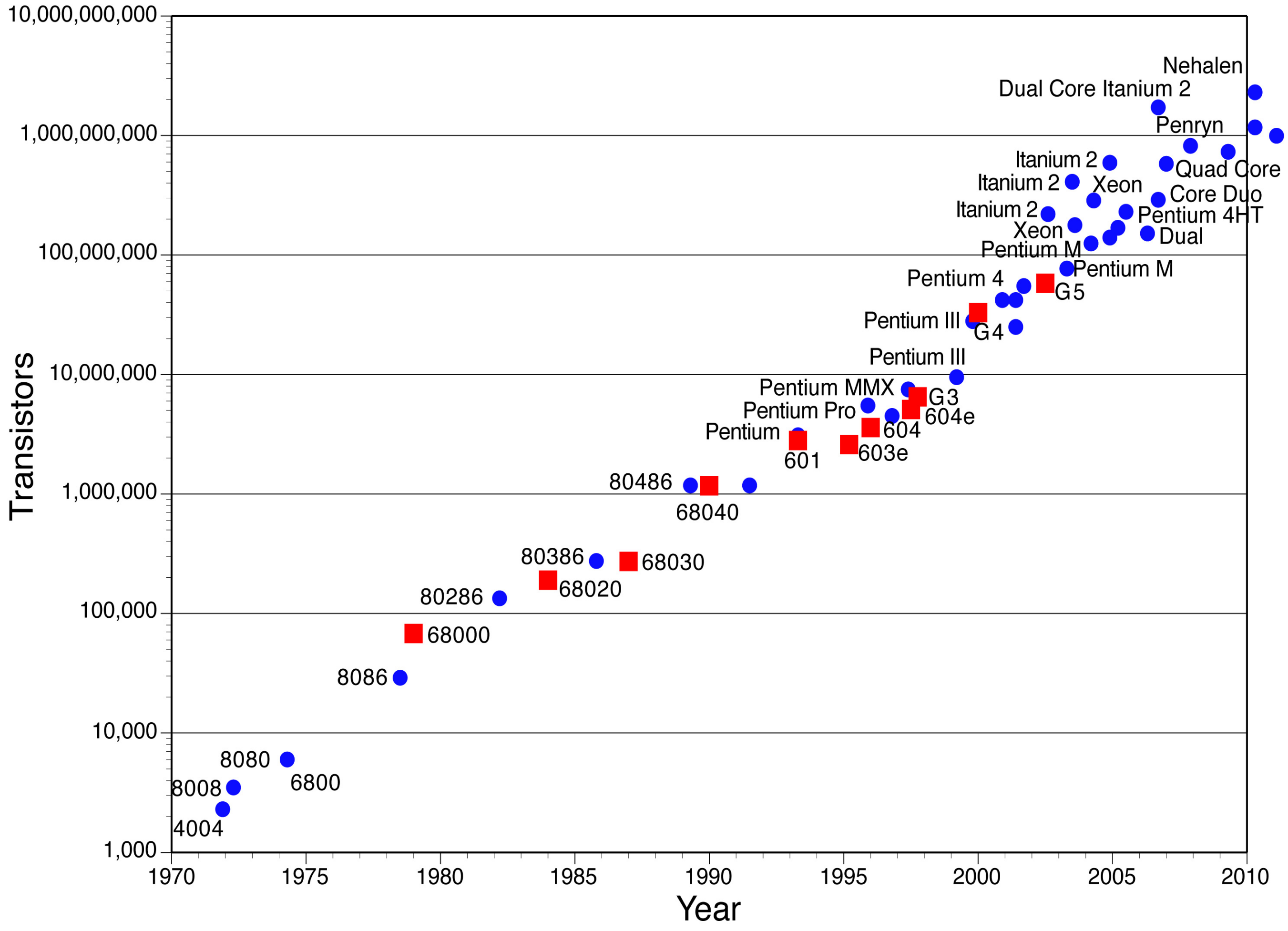
* inspired by Dean Chapman's 1979 Dryden Lecture

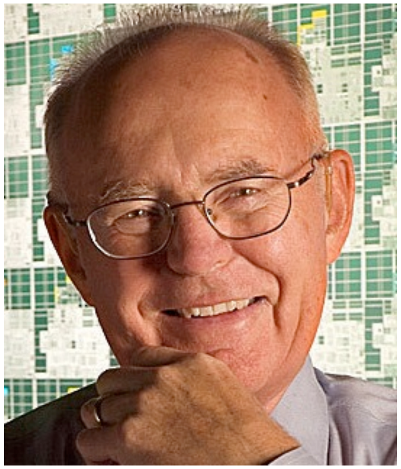
Balance shift in modality of scientific discovery



NIA 6 Aug 2012

Moore’s Law: exponential growth in time





Attributed to Gordon Moore of Intel from a paper in 1965 projecting CMOS transistor density, the term is applied today throughout science and technology

"Moore's Law" for fusion energy simulations

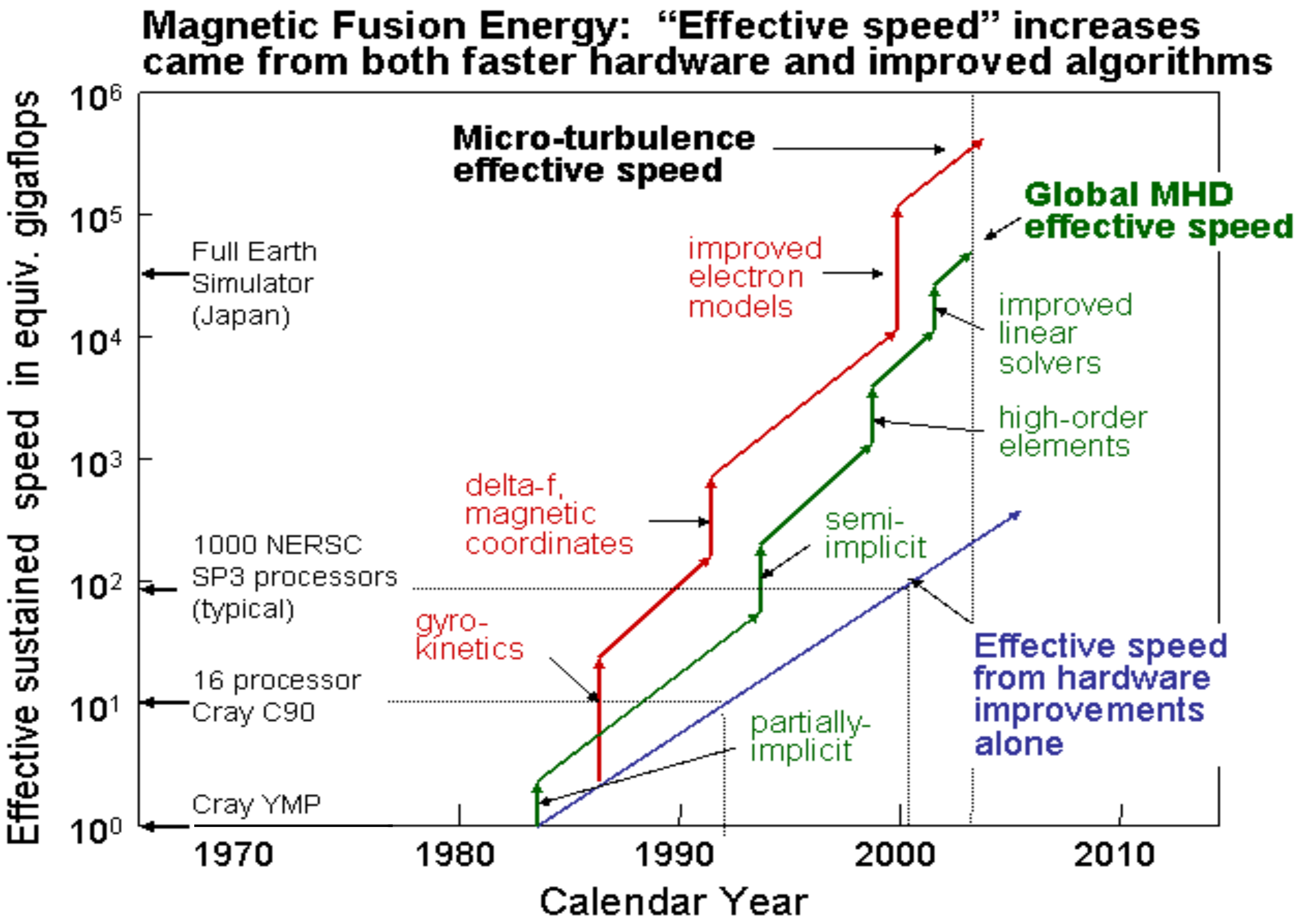


Figure from DOE "SCaLeS report" Volume 2 (Keyes et al., 2004)

"Moore's Law" for clean combustion simulations

Combustion: "Effective speed" increases came from both faster hardware and improved algorithms.

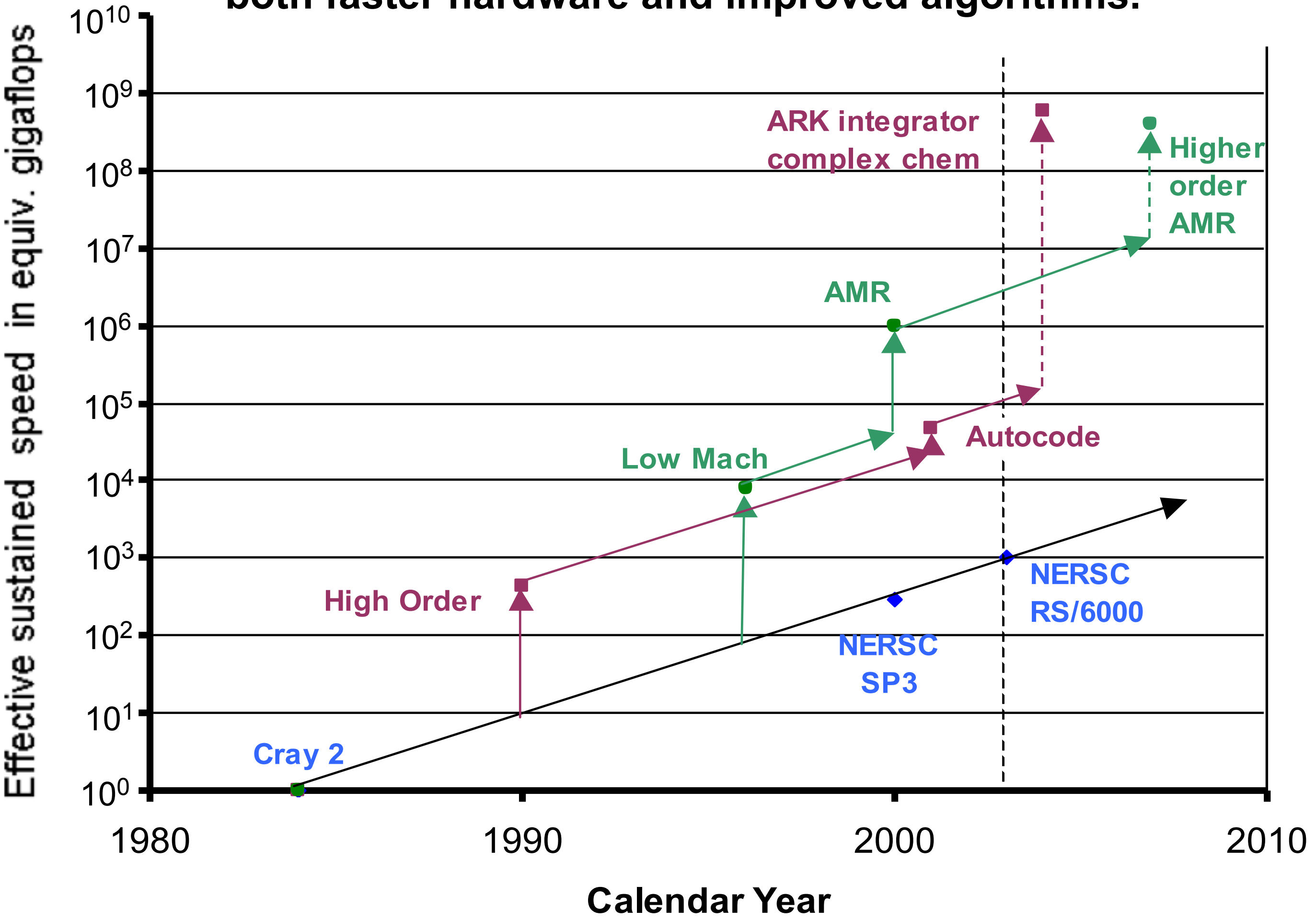
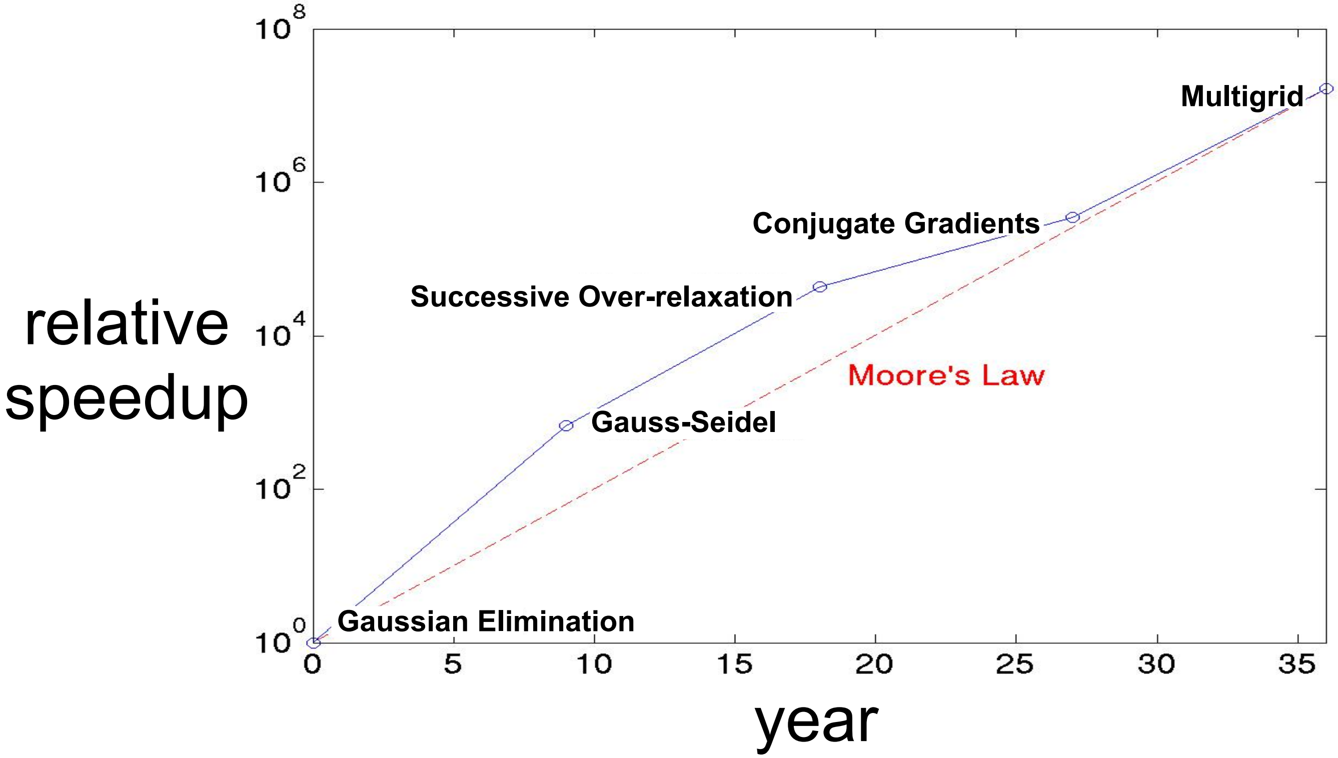
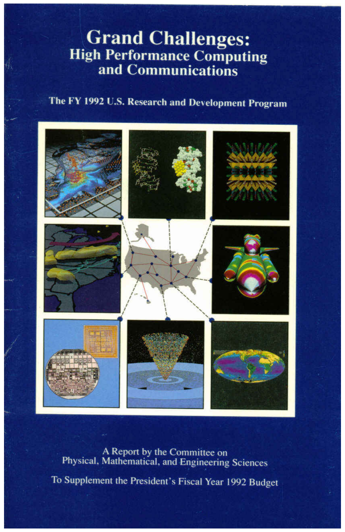


Figure from DOE "SCaLeS report" Volume 2 (Keyes et al., 2004)

Moore’s Law and numerical algorithms

- **First popularized in the 1992 NITRD bluebook: apply successive generations of algorithms to a fixed problem (“Poisson equation”)**
- **In 24 “doubling times” (1.5 years) for Moore’s Law for transistor density, better algorithms (software) contributed as much as better hardware**
- **2^{24}≈16 million ⇒ 6 months of computing now takes 1 second on fixed hardware***
- ***Two* factors of 16 million each if the best *algorithm* runs on the best *hardware*!**





Grand Challenges:
High Performance Computing
and Communications

The FY 1992 U.S. Research and Development Program

A Report by the Committee on
Physical, Mathematical, and Engineering Sciences
To Supplement the President’s Fiscal Year 1992 Budget

*algorithmic factor of improvement increases with problem size

Why push to extreme scale?
(DOE CSGF application essay question #3)

- Better resolve model’s full, natural range of length or time scales
- Accommodate physical effects with greater fidelity
- Allow the model degrees of freedom in all relevant dimensions
- Better isolate artificial boundary conditions (e.g., in PDEs) or better approach realistic levels of dilution (e.g., in MD)
- Combine multiple complex models
- Solve an inverse problem, or perform data assimilation
- Perform optimization or control
- Quantify uncertainty
- Improve statistical estimates

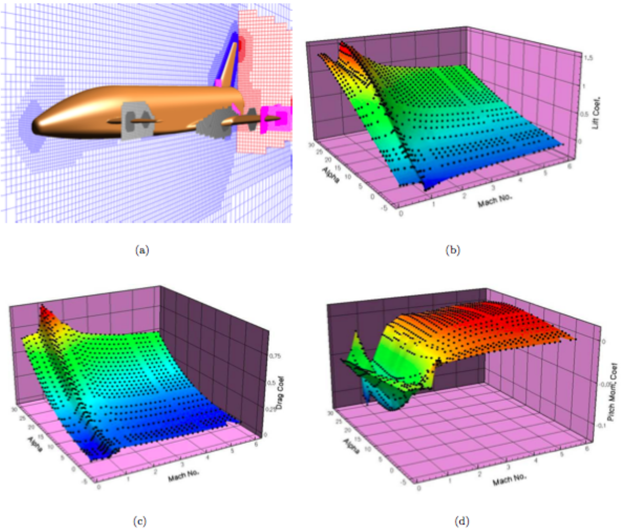
“Third paradigm”

- Operate without models (machine learning)

“Fourth paradigm”

Why push to extreme scale?
(AIAA paper, Mavriplis *et al.*, June 2007)

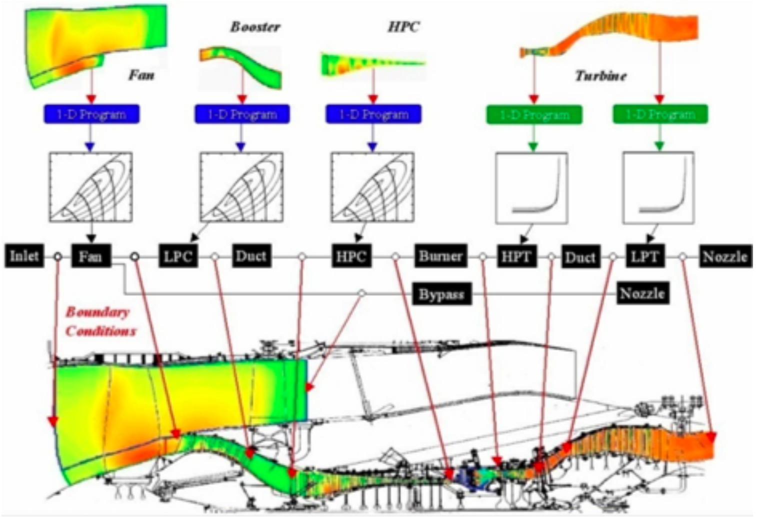
Digital Flight



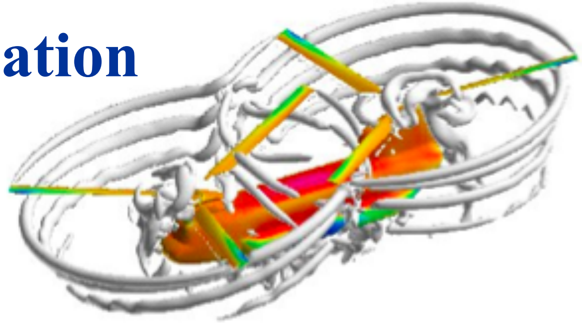
(a) (b)

(c) (d)

Propulsion



Design/Optimization



* Figures used by permission; see Mavriplis *et al.* 2007

For exascale background, see *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.

Extrapolating exponentials is unwise

- **Scientific computing world at a crossroads w.r.t. extreme scale**
- **Proceeded steadily for three decades from mega- (1970s) to giga- (1988) to tera- (1998) to peta- (2008) with *same* programming model and *same* algorithms**
  - **exa- is qualitatively different and will be much harder**
- **Core numerical analysis and scientific computing will ultimately confront exascale to maintain sponsor relevance**
  - **though obviously, there remain many mathematically fruitful directions are architecture-neutral**

NASA CFD relevance

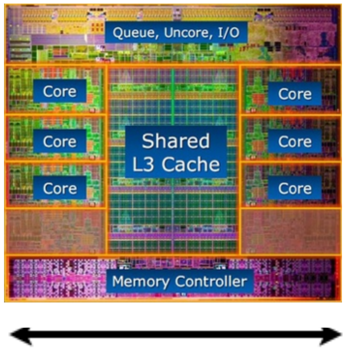
- Exascale's extremes change the game
  - mathematicians are on the front line
    - without contributions in the form of new mathematics (including statistics), the passage to the exascale will yield little fruit
  - mathematical scientists will find the computational power to do things many have wanted
    - room for creativity in "post-forward" problems (inverse problems and data assimilation)
    - mathematical scientists will participate in cross-disciplinary integration – "third paradigm" *and* "fourth paradigm"
    - remember that *exascale at the lab* means *petascale on the desk*
- Let's mention some mathematical opportunities, after quickly reviewing the hardware challenges

Why exa- is different

Which steps of FMADD take more energy?

64-bit floating-point fused multiply add **or** moving four 64-bit operands 20 mm across the die

```
     934,569.299814557                 input
x         52.827419489135904           input
----------------------------
= 49,370,884.442971624253823
+          4.20349729193958            input
----------------------------
= 49,370,888.64646892                  output
```



20 mm

(Intel Sandy Bridge, 2.27B transistors)

Going across the die requires up to an order of magnitude more!

DARPA study predicts that by 2019:

- **Double precision FMADD flop: 11pJ**
- **cross-die per word access (1.2pJ/mm): 24pJ (= 96pJ overall)**

c/o T. Schulthess (ETHZ); c/o P. Kogge (ND) *et al.*

Why exa- is different, cont.

Moore's Law (1965) does not end but Dennard's MOSFET scaling (1972) does

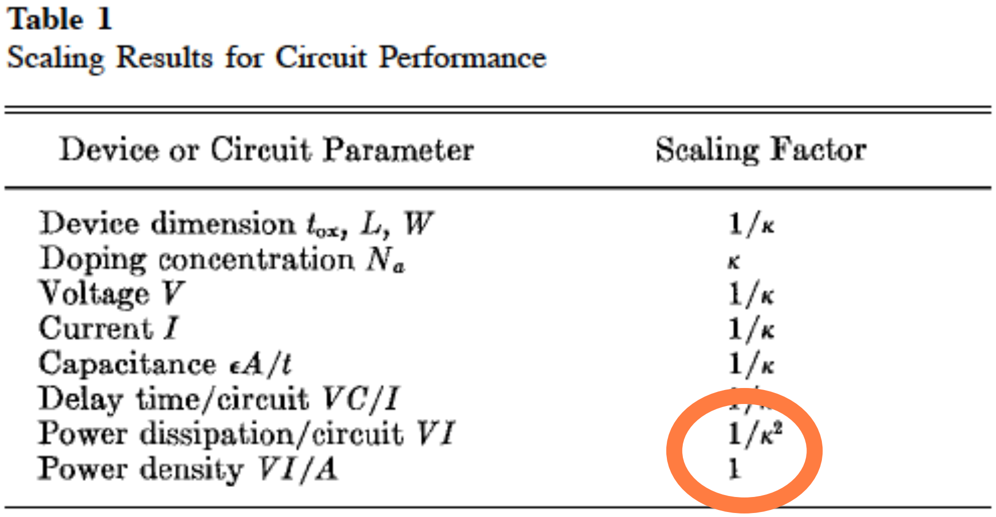


Table 1
Scaling Results for Circuit Performance

Device or Circuit Parameter	Scaling Factor
Device dimension t_{ox}, L, W	$1/\kappa$
Doping concentration N_a	κ
Voltage V	$1/\kappa$
Current I	$1/\kappa$
Capacitance $\epsilon A/t$	$1/\kappa$
Delay time/circuit VC/I	$1/\kappa$
Power dissipation/circuit VI	$1/\kappa^2$
Power density VI/A	1

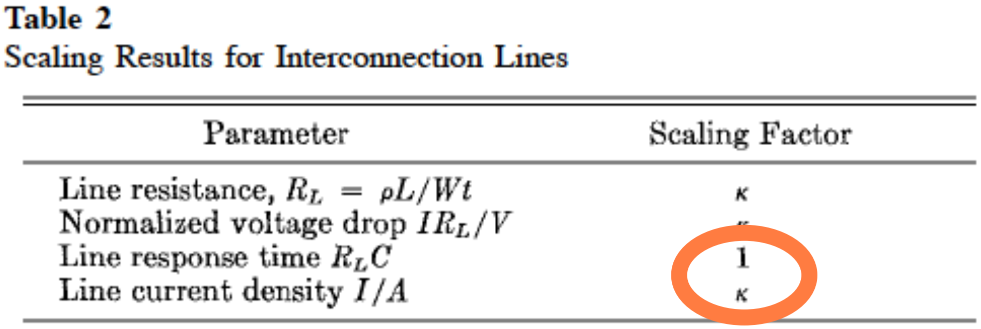


Table 2
Scaling Results for Interconnection Lines

Parameter	Scaling Factor
Line resistance, $R_L = \rho L/Wt$	κ
Normalized voltage drop IR_L/V	κ
Line response time R_LC	1
Line current density I/A	κ

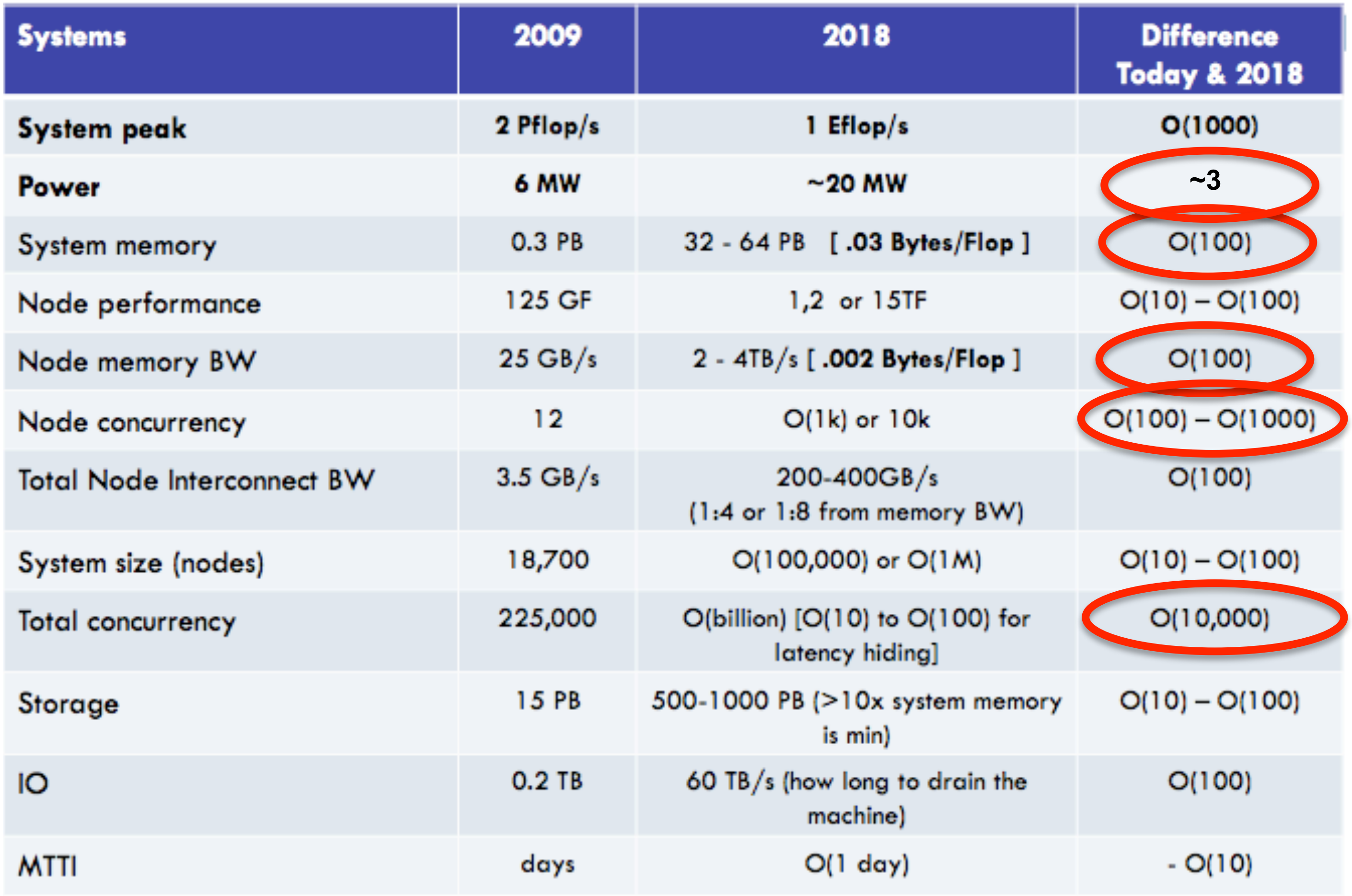
Robert Dennard, IBM
(inventor of DRAM, 1966)

Eventually processing will be limited by transmission

What will first "general purpose" exaflop/s machines look like?

- *Hardware*: many potentially exciting paths beyond today's CMOS silicon-etched logic, but not commercially at scale within the decade
- *Software*: many ideas for general-purpose and domain-specific programming models beyond "MPI + X", but not penetrating the main CS&E workforce within the decade

Prototype exascale hardware: a heterogeneous, distributed memory *GigaHz KiloCore MegaNode* system



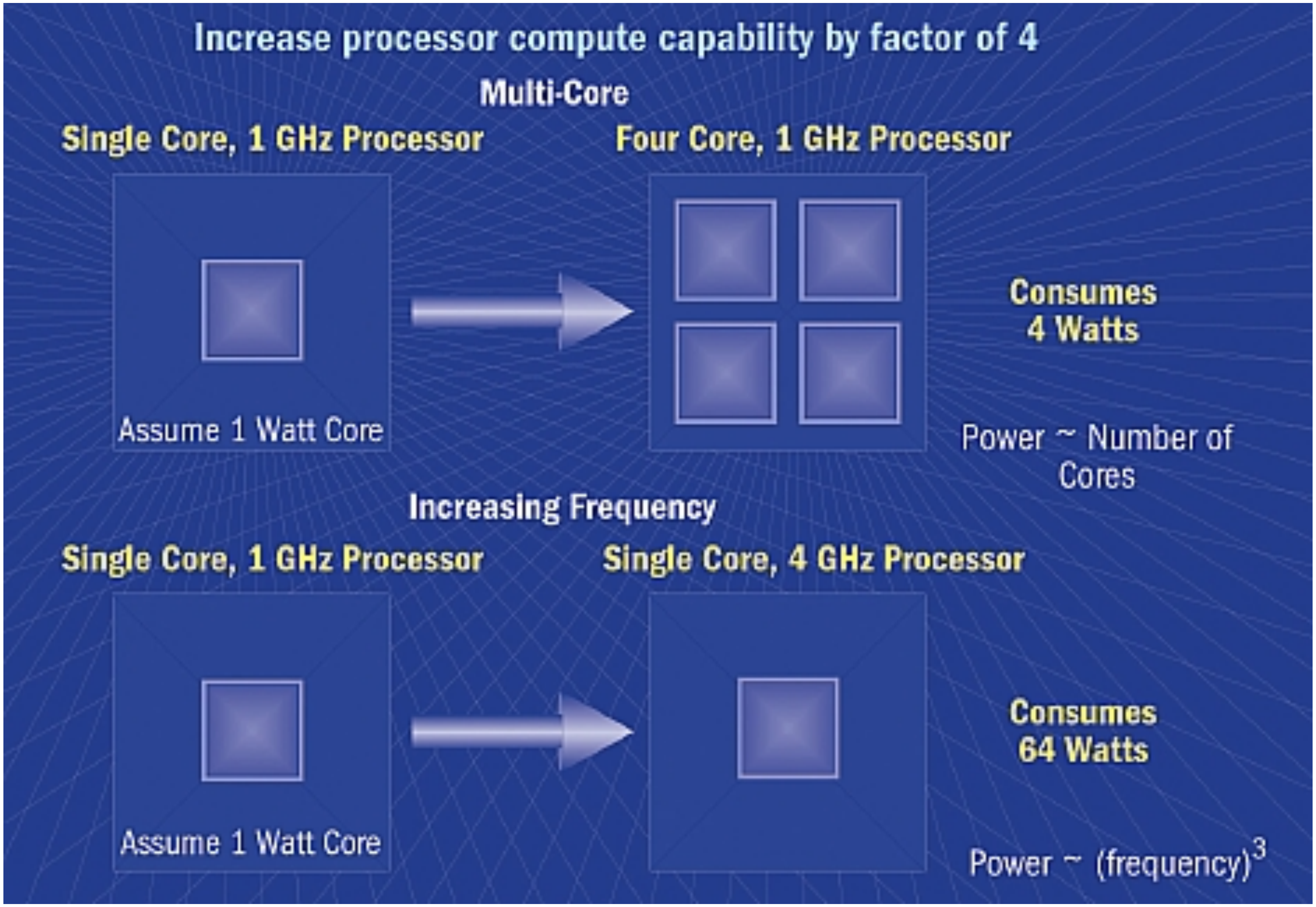
Systems	2009	2018	Difference Today & 2018
System peak	**2 Pflop/s**	**1 Eflop/s**	**O(1000)**
Power	**6 MW**	**~20 MW**	**~3**
System memory	0.3 PB	32 - 64 PB **[.03 Bytes/Flop]**	O(100)
Node performance	125 GF	1,2 or 15TF	O(10) – O(100)
Node memory BW	25 GB/s	2 - 4TB/s **[.002 Bytes/Flop]**	O(100)
Node concurrency	12	O(1k) or 10k	O(100) – O(1000)
Total Node Interconnect BW	3.5 GB/s	200-400GB/s (1:4 or 1:8 from memory BW)	O(100)
System size (nodes)	18,700	O(100,000) or O(1M)	O(10) – O(100)
Total concurrency	225,000	O(billion) [O(10) to O(100) for latency hiding]	O(10,000)
Storage	15 PB	500-1000 PB (>10x system memory is min)	O(10) – O(100)
IO	0.2 TB	60 TB/s (how long to drain the machine)	O(100)
MTTI	days	O(1 day)	- O(10)

c/o P. Beckman (ANL)

Some exascale themes

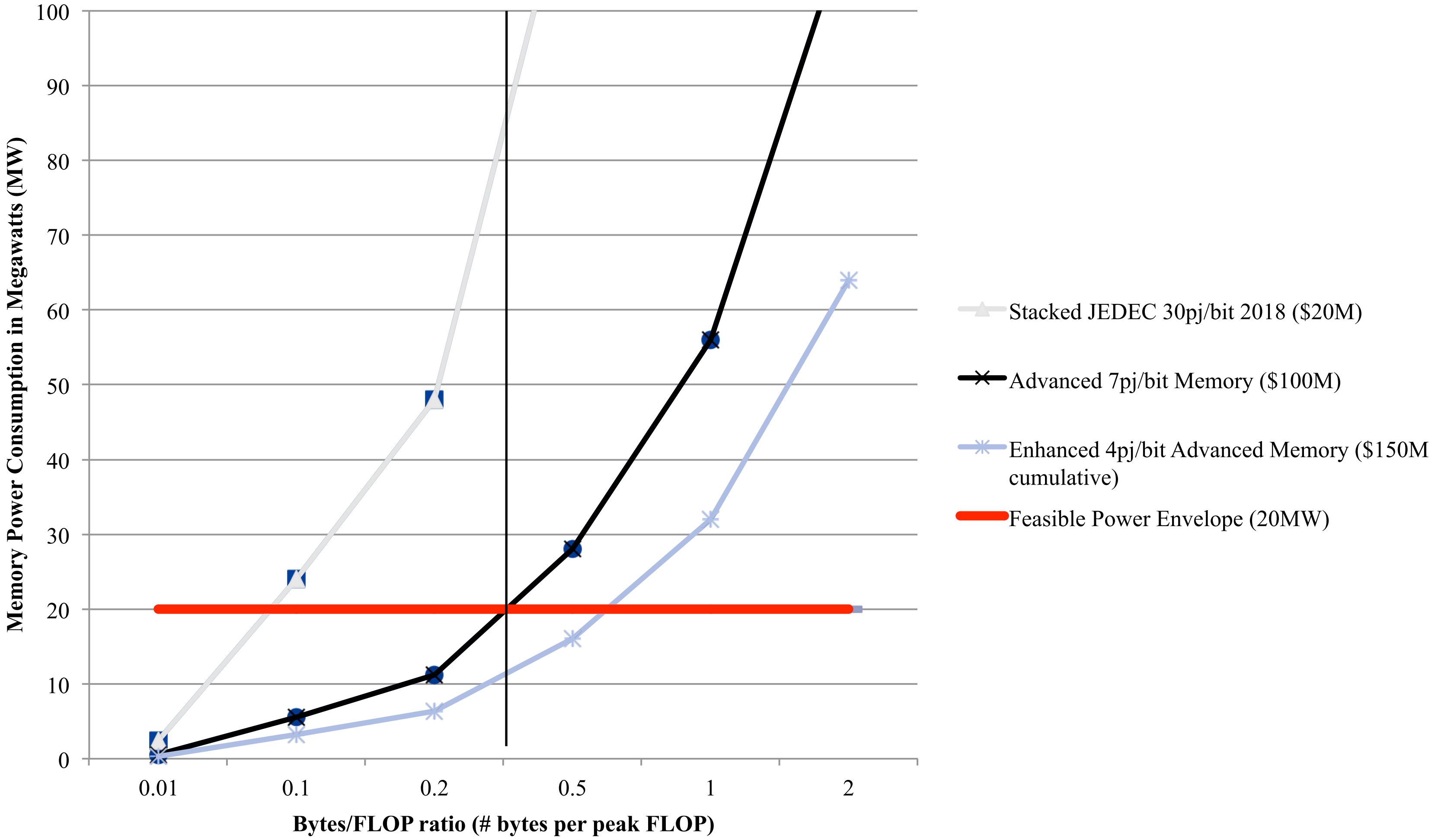
- Clock rates cease to increase while arithmetic capacity continues to increase dramatically w/concurrency consistent with Moore's Law
- Storage capacity diverges exponentially below arithmetic capacity
- Transmission capacity diverges exponentially below arithmetic capacity
- Mean time between hardware interrupts shortens
- Billions of dollars of scientific software hang in the balance until better algorithms arrive to span the architectural gap

Hurdle #1: power requires slower clocks and greater concurrency


Increase processor compute capability by factor of 4
Multi-Core
Single Core, 1 GHz Processor
Four Core, 1 GHz Processor
Assume 1 Watt Core
Consumes
4 Watts
Power ~ Number of Cores
Increasing Frequency
Single Core, 1 GHz Processor
Single Core, 4 GHz Processor
Assume 1 Watt Core
Consumes
64 Watts
Power ~ (frequency)³

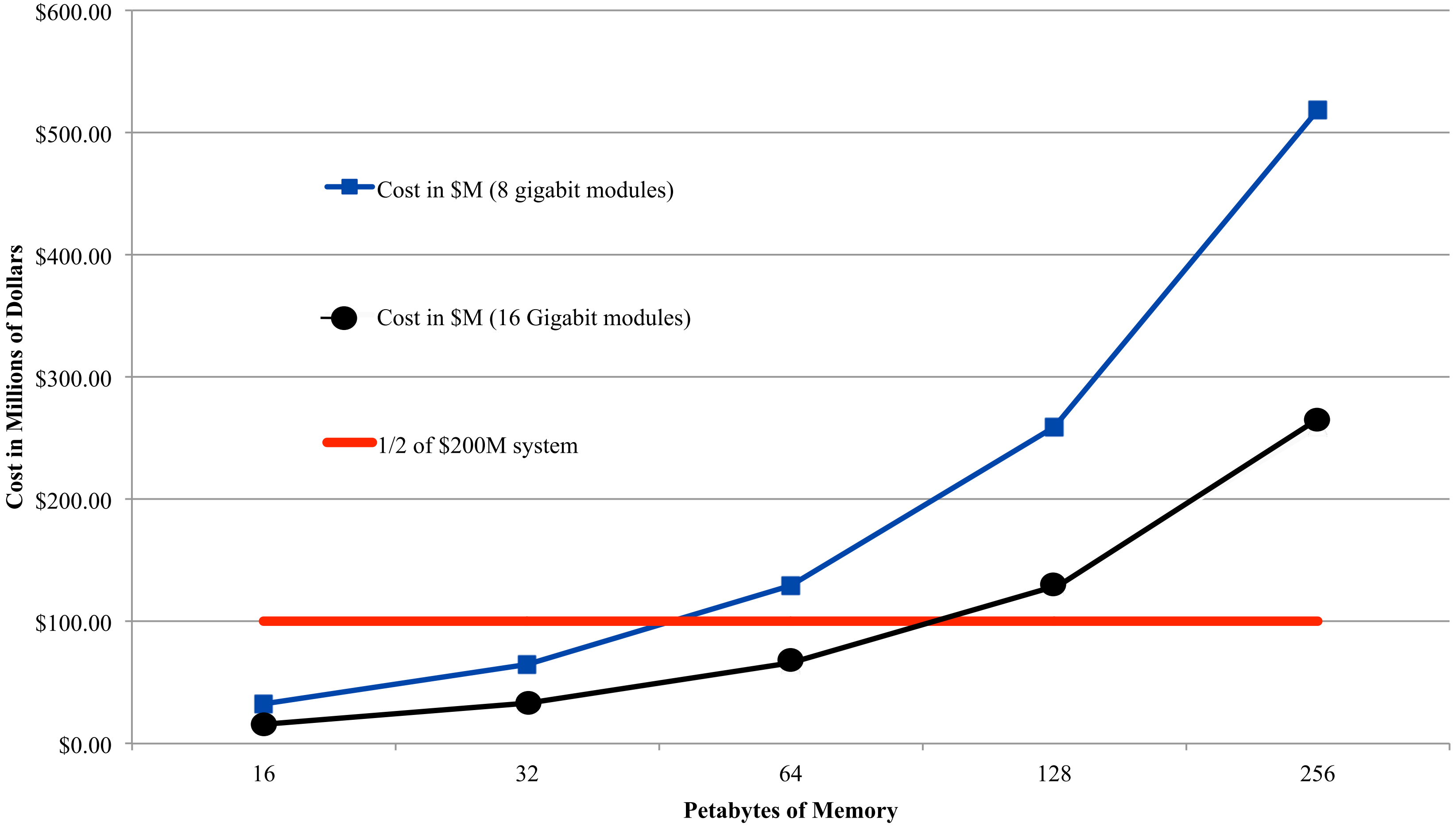
c/o *SciDAC Review* 16, February 2010

Hurdle #2: memory bandwidth could eat up the entire power budget



c/o John Shalf (LBNL)

Hurdle #3: memory capacity could eat up the entire fiscal budget



c/o John Shalf (LBNL)

Implications of operating on the edge

- Draconian reduction required in power per flop and per byte will make computing and copying data less reliable
  - voltage difference between “0” and “1” will be reduced
  - circuit elements will be smaller and subject to greater physical noise per signal
  - there will be more errors that must be caught and corrected
- Power will have to be cycled off and on or clocks slowed and speeded based on compute schedules and based on cooling capacity
  - makes per node performance rate unreliable

Implications of operating on the edge

- **Expanding the number of nodes (processor-memory units) beyond 10^6 would *not* a serious threat to algorithms that lend themselves to well-amortized precise load balancing**
  - **provided that the nodes are performance reliable**
- **A real challenge is expanding the number of cores on a node to 10^3**
  - **must be done while memory and memory bandwidth per node expand by (at best) ten-fold less (basically "strong" scaling)**
- **It is already about 10^3 slower to to retrieve an operand from main DRAM memory than to perform an arithmetic operation – will get worse by a factor of ten**
  - **almost all operands must come from registers or upper cache**

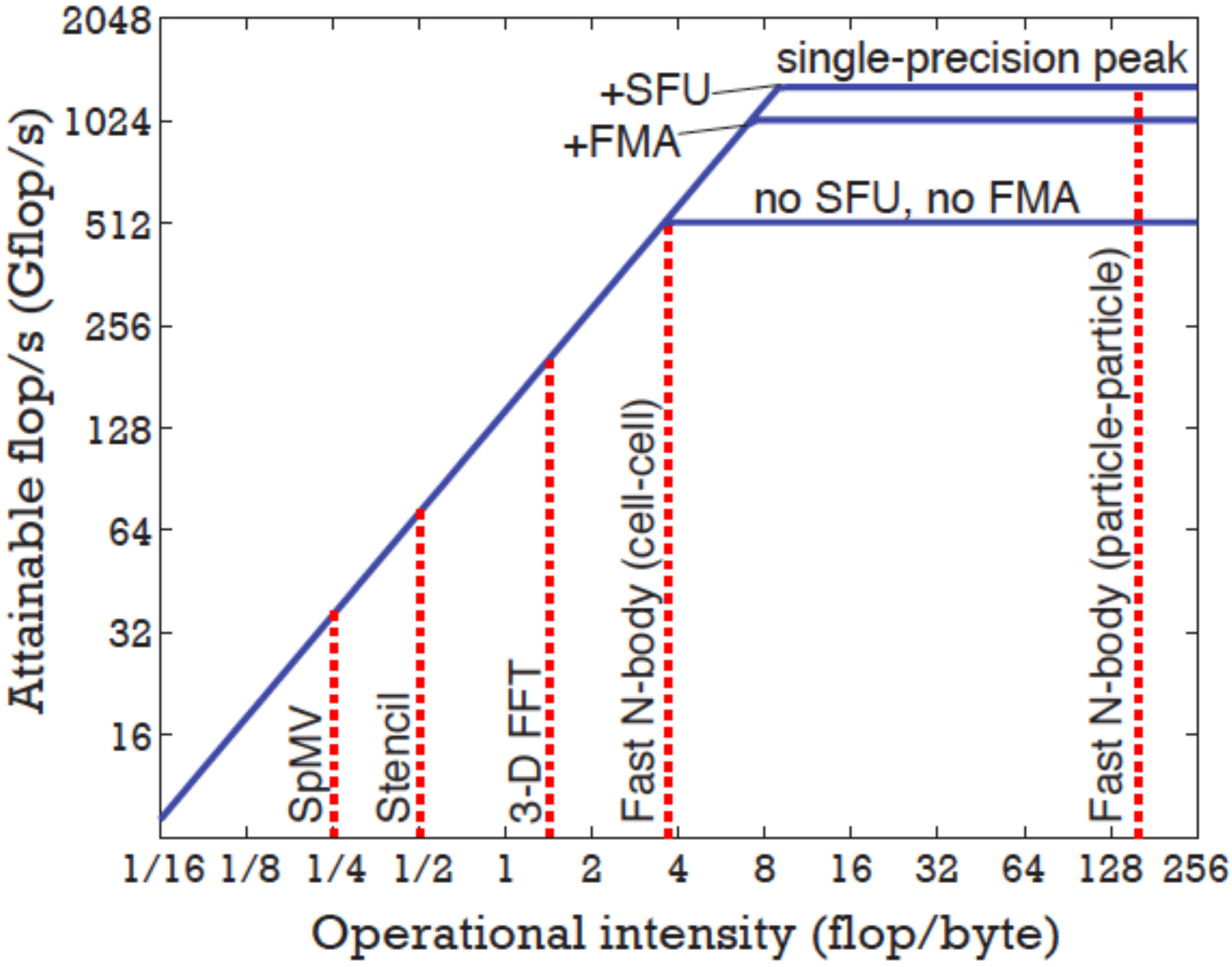
“Missing” mathematics

- New formulations with
  - greater arithmetic intensity (flops per bytes moved into and out of registers and upper cache)
  - reduced communication
  - reduced synchronization
  - assured accuracy with (adaptively) less floating-point precision
- Quantification of trades between limiting resources
- *Plus* all of the exciting analytical agendas that exascale is meant to exploit

Arithmetic intensity illustration

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)

Classical ideas in communication reduction and synchronization reduction for *Ax*=*b*

- **Amortize communication over many computational steps**
  - **s-step Krylov methods: power kernels with wide halos**
  - **"tall skinny QR": recursively double the row-scope of independent QRs**
  - **Block Krylov methods: solve *b* several independent systems at once with improved convergence (based on λ_{max}/λ_b rather than $\lambda_{max}/\lambda_{min}$)**
- **Enable less synchrony between inner loop steps**
  - **new synchronization-reducing sparse matrix-vector multiply on IBM's SPI environment in BG/Q**
  - **perform local multiplies while pushing data to neighbors and finish up as off-processor data becomes available**

Miracles "need not apply"

- **We should not expect to escape causal dependencies**
  - **if the input-to-output map of a problem description has all-to-all data dependencies, like an elliptic PDE Green's function, and if we need the solution accurately everywhere, we will have all-to-all communication**
- **But we should ask fundamental questions:**
  - **for the science of interest, do we need to evaluate the output everywhere?**
  - **is there another formulation that can produce the same required scientific observables in less time and energy?**

How are most workhorse simulations implemented at the infra-petascale today?

- **Iterative methods based on data decomposition and message-passing**
  - each individual processor works on a portion of the original problem and exchanges information at its boundaries with other processors that own portions with which it interacts causally, to evolve in time or to establish equilibrium
  - computation and neighbor communication are both fully parallelized and their ratio remains constant in weak scaling
- **The programming model is SPMD/BSP/CSP**
  - Single Program, Multiple Data
  - Bulk Synchronous Programming
  - Communicating Sequential Processes

Estimating scalability

- **Given complexity estimates of the leading terms of:**
  - **the concurrent computation (per iteration phase)**
  - **the concurrent communication**
  - **the synchronization frequency**
- **And a 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**
  - **simply differentiate time estimate in terms of problem size *N* and processor number *P* with respect to *P***

3D stencil computation weak scaling

(assume fast local network, tree-based global reductions)

- **Total wall-clock time per iteration (ignoring local comm.)**

$$T(N,P) = A\frac{N}{P} + C\log P$$

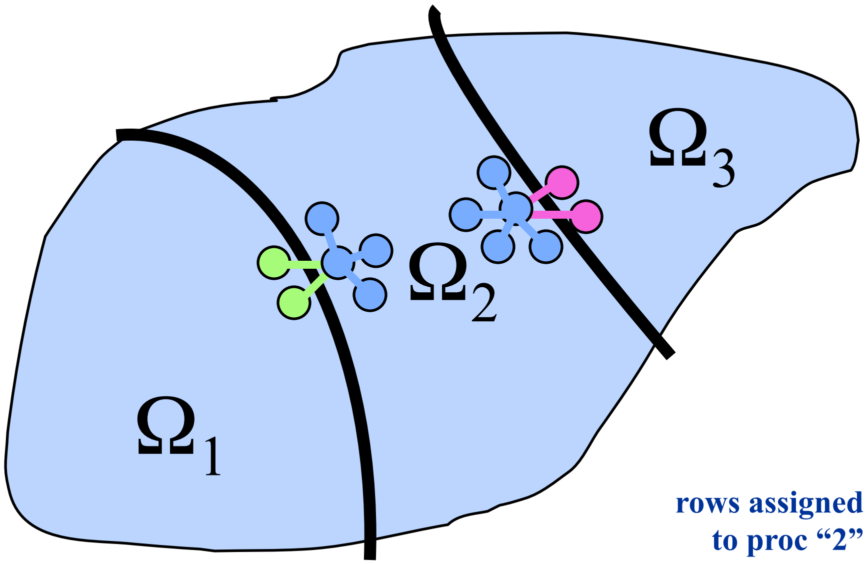
- **For optimal P,** $\frac{\partial T}{\partial P} = 0$ **, or** $-A\frac{N}{P^2} + \frac{C}{P} = 0$

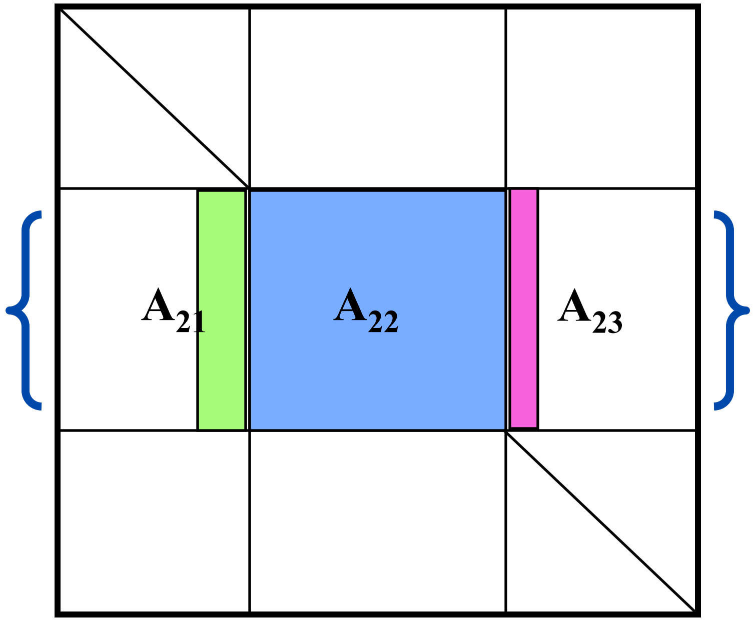
or $P_{opt} = \frac{A}{C}N$

- **P can grow linearly with N, and running time increases "only" logarithmically – *as good as weak scaling can be!***
- **Problems: (1) assumes perfect synchronization,**

(2) log of a billion may be "large"

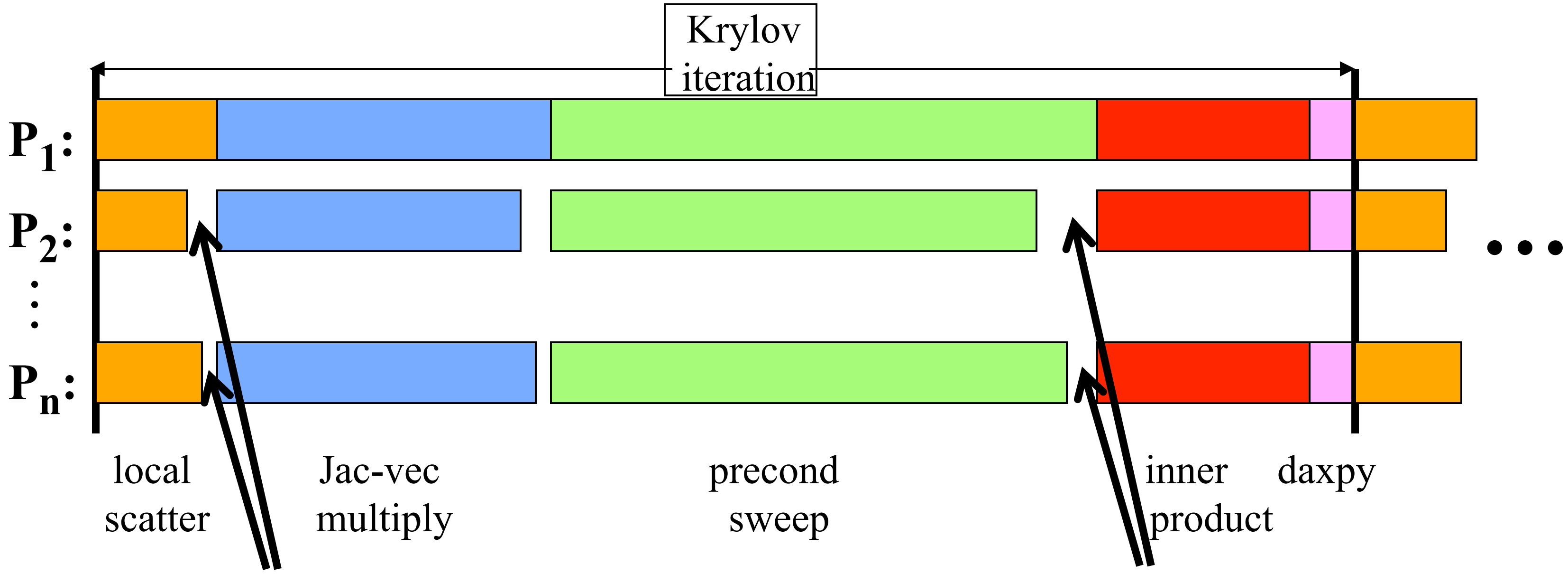
SPMD parallelism w/ domain decomposition: *an endangered species?*





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

Workhorse innards: e.g., Krylov-Schwarz, a bulk synchronous implicit solver

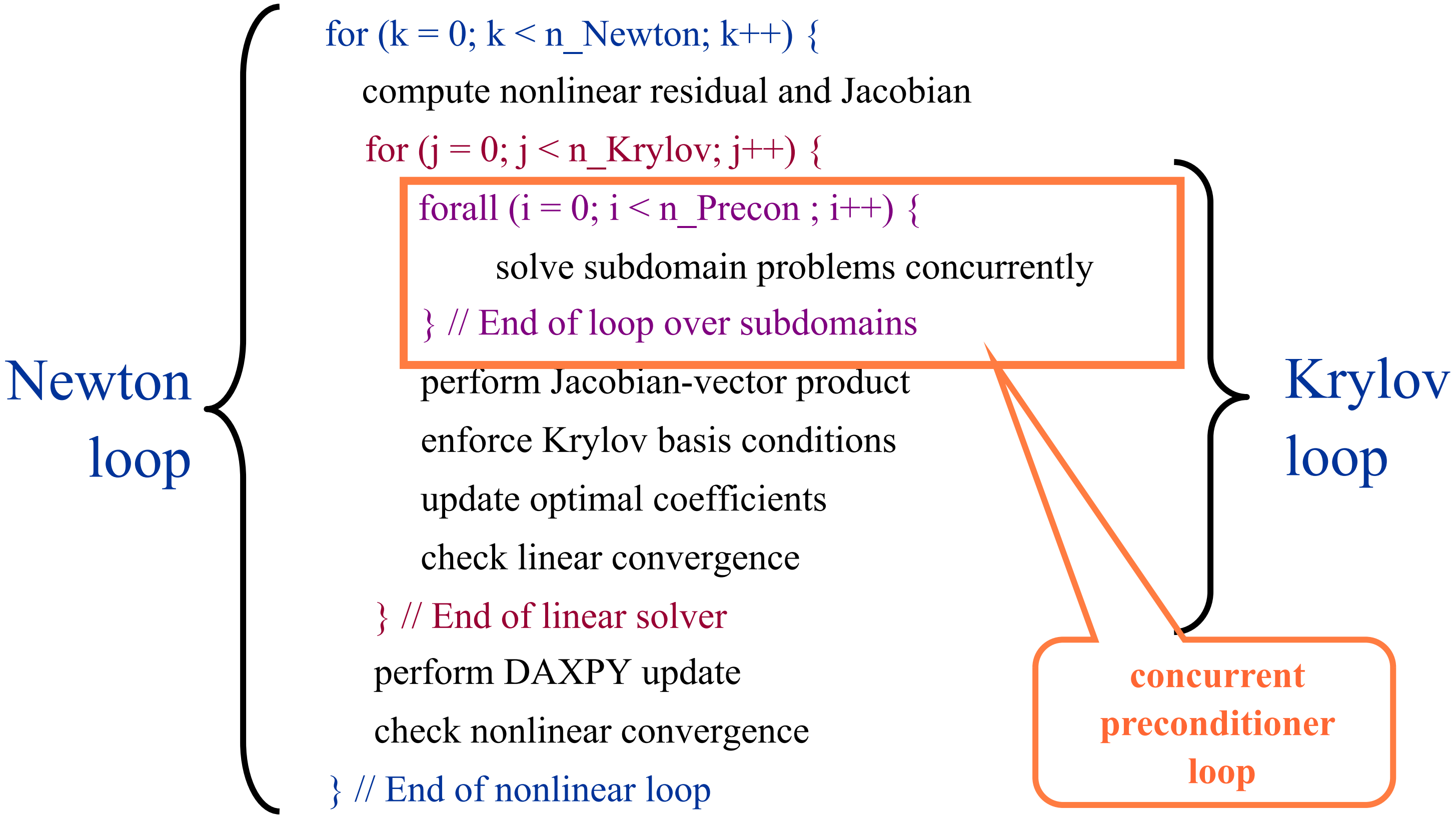


communication imbalance

computation imbalance

Idle time due to load imbalance becomes a challenge at, say, one billion cores, when *one* processor can hold up *all* of the rest at a synchronization point

Our programming idiom is nested loops, e.g., Newton-Krylov-Schwarz



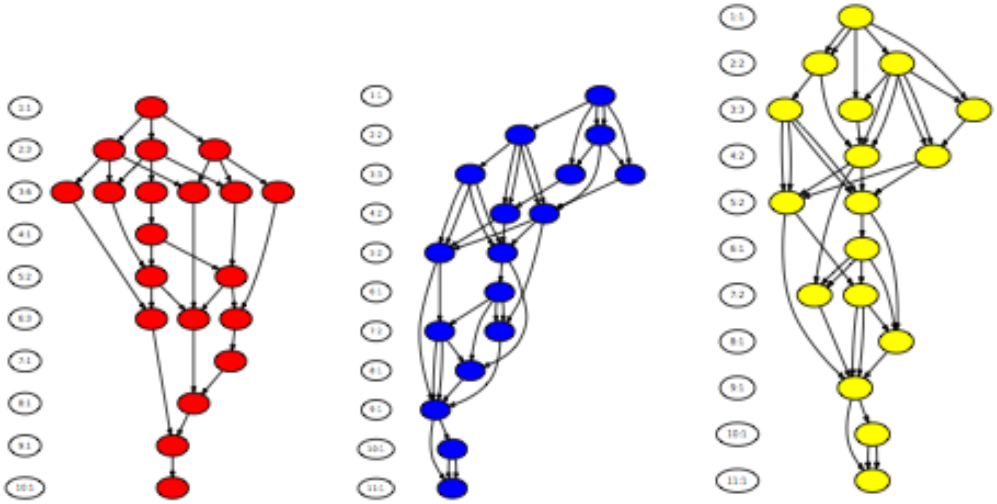
```
for (k = 0; k < n_Newton; k++) {
    compute nonlinear residual and Jacobian
    for (j = 0; j < n_Krylov; j++) {
        forall (i = 0; i < n_Precon ; i++) {
            solve subdomain problems concurrently
        } // End of loop over subdomains
        perform Jacobian-vector product
        enforce Krylov basis conditions
        update optimal coefficients
        check linear convergence
    } // End of linear solver
    perform DAXPY update
    check nonlinear convergence
} // End of nonlinear loop
```

Outer loops (not shown): continuation, implicit timestepping, optimization

Dataflow Illustration: Generalized Eigensolver

$$Ax = \lambda Bx$$

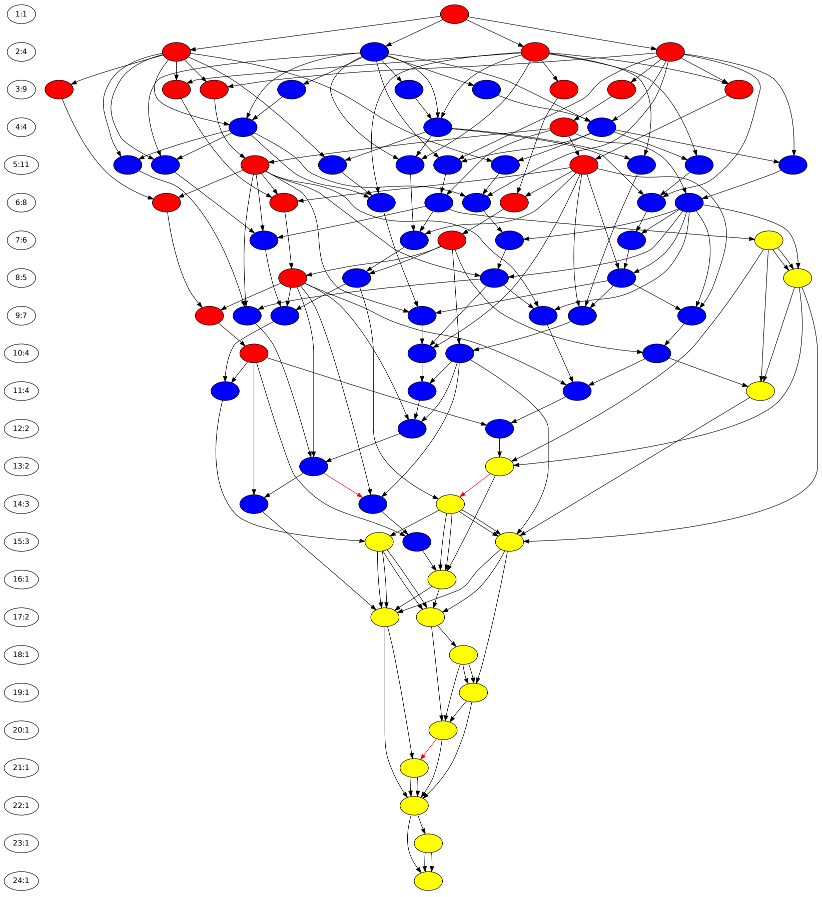
Operation	Explanation	LAPACK routine name
1 $B = L \times L^T$	Cholesky factorization	POTRF
2 $C = L^{-1} \times A \times L^{-T}$	application of triangular factors	SYGST or HEGST
3 $T = Q^T \times C \times Q$	tridiagonal reduction	SYEVD or HEEVD
4 $Tx = \lambda x$	QR iteration	STERF



c/o H. Ltaief (KAUST)

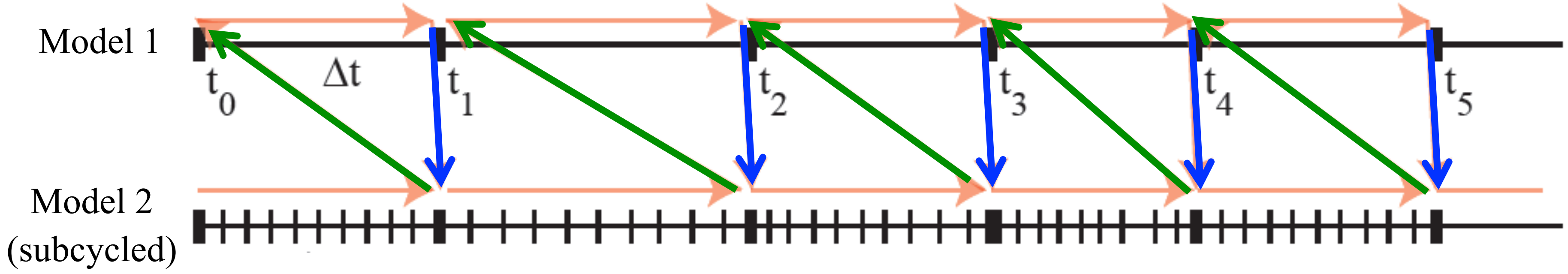
These loops, with their artifactual orderings, need to be replaced with DAGs

- Diagram shows a dataflow ordering of the steps of a 4×4 symmetric generalized eigensolver
- Nodes are tasks, color-coded by type, and edges are data dependencies
- Time is vertically downward



c/o H. Ltaief (KAUST)

Multiphysics w/ legacy codes: *an endangered species?*



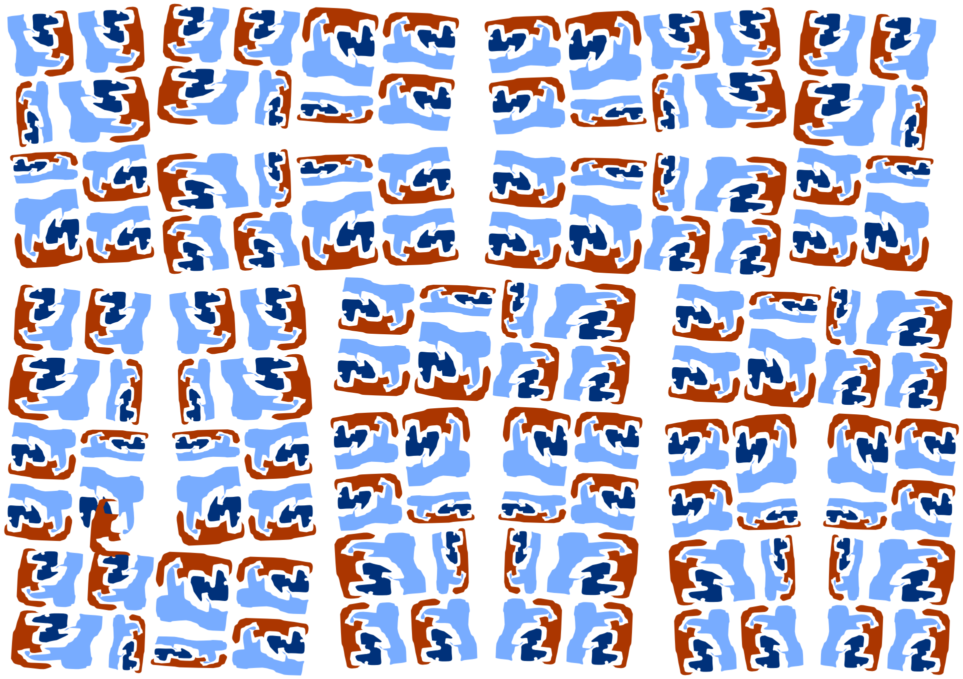
- **Many multiphysics codes operate like this, where the models may occupy the same domain in the bulk (e.g., reactive transport) or communicate at interfaces (e.g., ocean-atmosphere)***
- **The data transfer cost represented by the blue and green arrows may be much higher than the computation cost of the models, even apart from first-order operator splitting error and possible instability**

*see Keyes, et al., 2011 paper from DOE ICiS workshop for IJHPCA

Many codes have the algebraic and software structure of multiphysics

- Exascale is motivated by these:
  - uncertainty quantification, inverse problems, optimization, immersive visualization and steering
- These may carry auxiliary data structures to/from which blackbox model data is passed and they act like just another "physics" to the hardware
  - pdfs, Lagrange multipliers, etc.
- Today's separately designed blackbox algorithms for these may not live well on exascale hardware: co-design may be required due to data motion

Multiphysics layouts must invade blackboxes



- Each application must first be ported to extreme scale (distributed, hierarchical memory)
- Then applications may need to be interlaced at the data structure level to minimize copying and allow work stealing at synchronization points

c/o W. D. Gropp (UIUC)

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)

- Hardware nonuniformity may force abandonment of the Bulk Synchronous Programming (BSP) paradigm
  - it will be impossible for the user to control load balance sufficiently to make it work
- Hardware and algorithmic nonuniformity will be indistinguishable at the performance level
  - good solutions for the dynamically load balancing in systems space will apply to user space, freeing users

Bad news/good news (4)

- **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**

Bad news/good news (5)

- Fully deterministic algorithms may simply come to be regarded as too synchronization-vulnerable
  - Rather than wait for data, we may infer it, taking into account sensitivity to poor guesses, and move on
- A rich numerical analysis of algorithms that make use of statistically inferred "missing" quantities may emerge

How will PDE computations adapt?

- Programming model will still be message-passing (due to large legacy code base), adapted to multicore or hybrid 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

Sources of nonuniformity

- System
  - manufacturing, OS jitter, TLB/cache performance variations, network contention, dynamic power management, soft errors, hard component failures, software-mediated resiliency, etc.
- Algorithmic
  - physics at gridcell/particle scale (e.g., table lookup, equation of state, external forcing), discretization adaptivity, solver adaptivity, precision adaptivity, etc.
- Effects are similar when it comes to waiting at synchronization points
- Possible solutions for system nonuniformity will improve programmability, too

Programming practice

- **Prior to possessing exascale hardware, users can prepare themselves by exploring new programming models**
  - **on manycore and heterogeneous nodes**
- **Attention to locality and reuse is valuable at all scales**
  - **will produce performance paybacks today *and* in the future**
  - **domains of coherence will be variable and hierarchical**
- **New algorithms and data structures can be explored under the assumption that flop/s are cheap and moving data is expensive**
- ***Independent tasks that have complementary resource requirements can be interleaved in time in independently allocated spaces***

Path for scaling up applications

- **Weak scale applications up to distributed memory limits**
  - **proportional to number of nodes**
- **Strong scale applications beyond this**
  - **proportional to cores per node/memory unit**
- **Scale the workflow, itself**
  - **proportional to the number of instances (ensembles)**
  - **integrated end-to-end simulation**
- **Co-design process is staged, with any of these types of scaling valuable by themselves**
- **Big question: does the software for co-design factor? Or is all the inefficiency at the data copies at interfaces between the components after a while?**

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

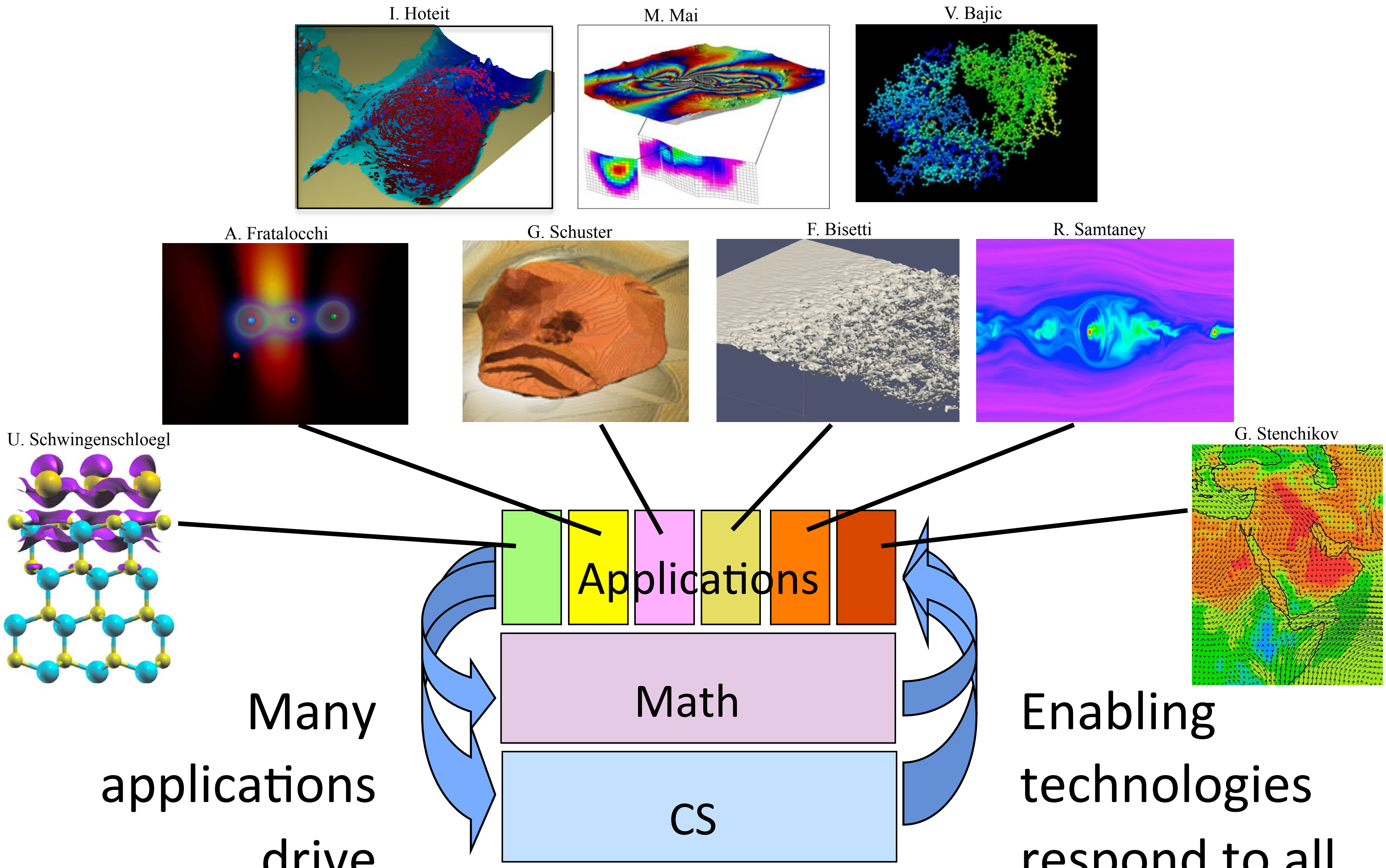
- Configuration systems
- Source-to-source translators
- Compilers
- Simulators
- Messaging systems
- Debuggers
- Profilers

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

Production-related

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

“SciDAC philosophy” of software investment



I. Hoteit

M. Mai

V. Bajic

A. Fratalocchi

G. Schuster

F. Bisetti

R. Samtaney

U. Schwingenschloegl

G. Stenchikov

Applications

Math

CS

Many applications drive

Enabling technologies respond to all

Kennedy’s Challenge, 1962

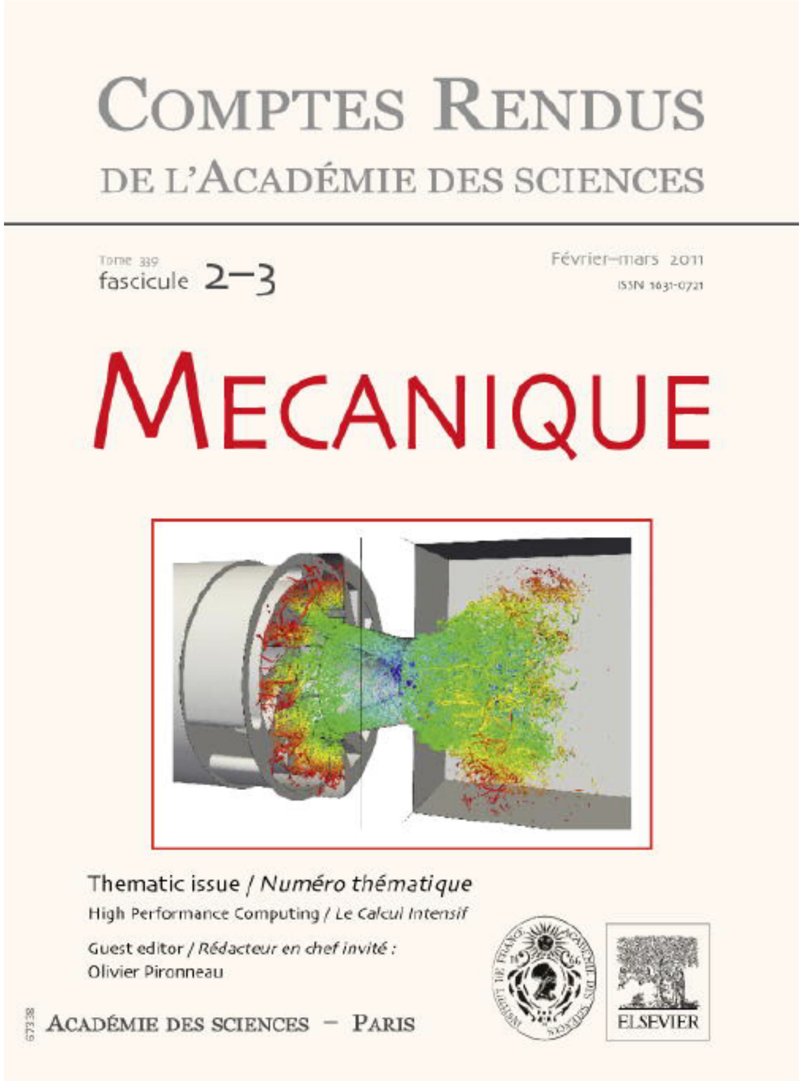
“We choose to do [these] things, not because they are easy, but because they are hard, *because that goal will serve to organize and measure the best of our energies and skills*, because that challenge is one that we are willing to accept, one we are unwilling to postpone, and one which we intend to win...”

Acknowledgment:
today's Peta-op/s machines

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

1.4 kilograms, 20 Watts

See 2011 special issue of *Comptes Rendus*



COMPTES RENDUS
DE L'ACADÉMIE DES SCIENCES

Tome 339
fascicule 2–3

Février–mars 2011
ISSN 1631-0721

MECANIQUE

Thematic issue / *Numéro thématique*
High Performance Computing / *Le Calcul Intensif*

Guest editor / *Rédacteur en chef invité* :
Olivier Pironneau

ACADÉMIE DES SCIENCES – PARIS

ELSEVIER

Exaflop/s: The why and the how, D. E. Keyes, *Comptes Rendus de l'Academie des Sciences* **339**, 2011, 70—77.