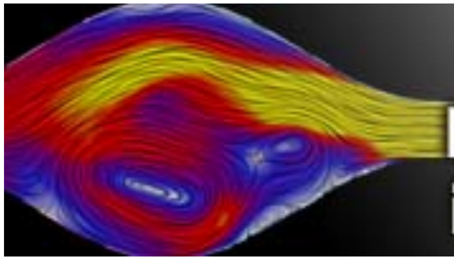


Sir James Lighthill
Distinguished Lectureship
in Mathematical Sciences

Tallahassee,
Florida

Exaflop/s,

Seriously!

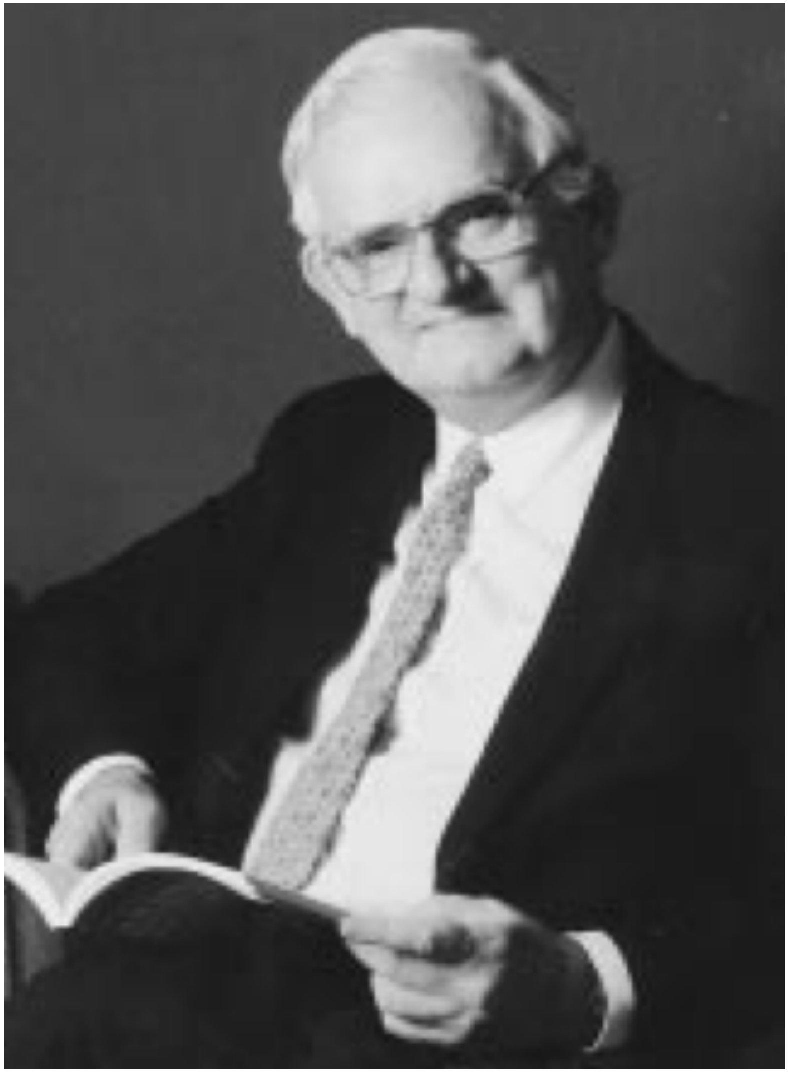
David Keyes

Division of Mathematical and Computer Sciences and Engineering, KAUST

Fu Foundation Professor of Applied Mathematics, Columbia University

Based on an article “The Exascale: Why and How” in
***Comptes Rendus de l’Academie des Sciences* (2011)**

In Memoriam



23 Jan 1924 – 17 July 1998

“James Lighthill was acknowledged throughout the world as one of the great mathematical scientists of this century. He was the prototypical applied mathematician, immersing himself thoroughly in the essence and even the detail of every engineering, physical, or biological problem he was seeking to illuminate with mathematical description, formulating a sequence of clear mathematical problems and attacking them with a formidable range of techniques completely mastered, or adapted to the particular need, or newly created for the purpose, and then finally returning to the original problem with understanding, predictions, and advice for action.”

(from the David Crighton memorial in *AMS Notices*)

Metric System prefixes

Prefix	**Multiplier**	**Power**
Kilo	1,000	3
Mega	1,000,000	6
Giga	1,000,000,000 *(1988)*	9
Tera	1,000,000,000,000 *(1998)*	12
Peta	1,000,000,000,000,000 *(2008)*	15
Exa	1,000,000,000,000,000,000 *(20??)*	18
Zetta	1,000,000,000,000,000,000,000	21
Yotta	1,000,000,000,000,000,000,000,000	24

Credits for this talk include the IESP team

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

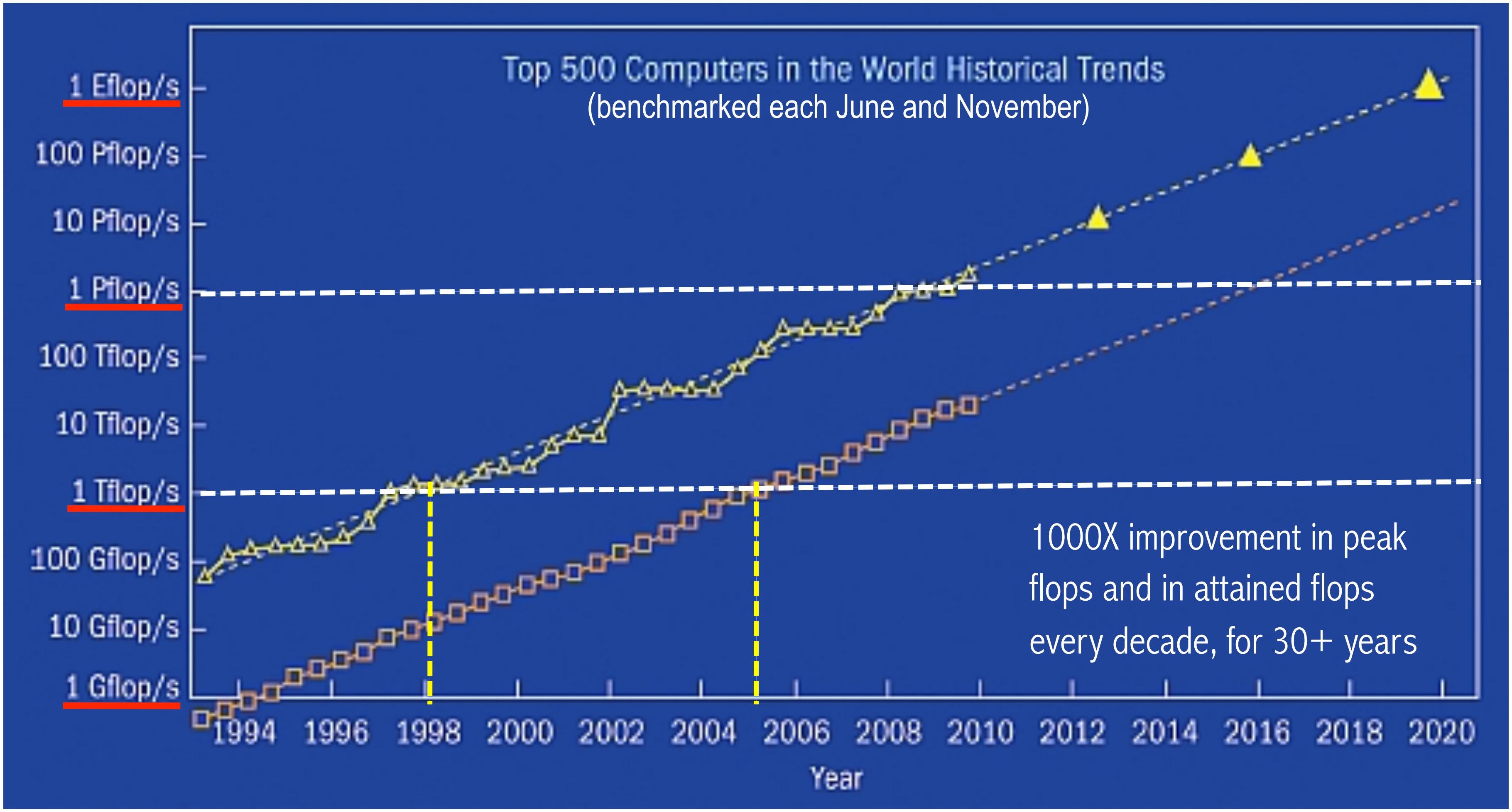
SPONSORS

Office of Science U.S. Department of Energy
NSF
ANR
cea
CERFACS
CRAY THE SUPERCOMPUTER COMPANY
eDF
EPSRC Engineering and Physical Sciences Research Council
FUJITSU
INRIA
KYOTO UNIVERSITY
GENCI
NVIDIA.
RIKEN
東京大学 THE UNIVERSITY OF TOKYO
筑波大学

The International Exascale Software Roadmap,

J. Dongarra, P. Beckman, et al., *International Journal of High Performance Computer Applications* **25**(1), 2011, ISSN 1094-3420.

"Top 500" list positions #1 and #500 over the decades – exaflop/s by 2020?



1000X improvement in peak flops and in attained flops every decade, for 30+ years

c/o *SciDAC Review* 16, February 2010

The G8 Research Councils Initiative on Multilateral Research Funding

The first Call for Proposals: Interdisciplinary Program on Application Software towards Exascale Computing for Global Scale Issues

The G8 Research Councils Initiative on Multilateral Research Funding

The Heads of Research Councils from the G8 countries (G8-HORCs) Canada, France, Germany, Japan, Russia, the UK, and the USA (Italy is not participating) have established a new joint funding initiative.

Global challenges need global solutions and this pilot initiative provides a new framework for co-operation across a broad range of disciplines and on a multilateral and multinational basis.

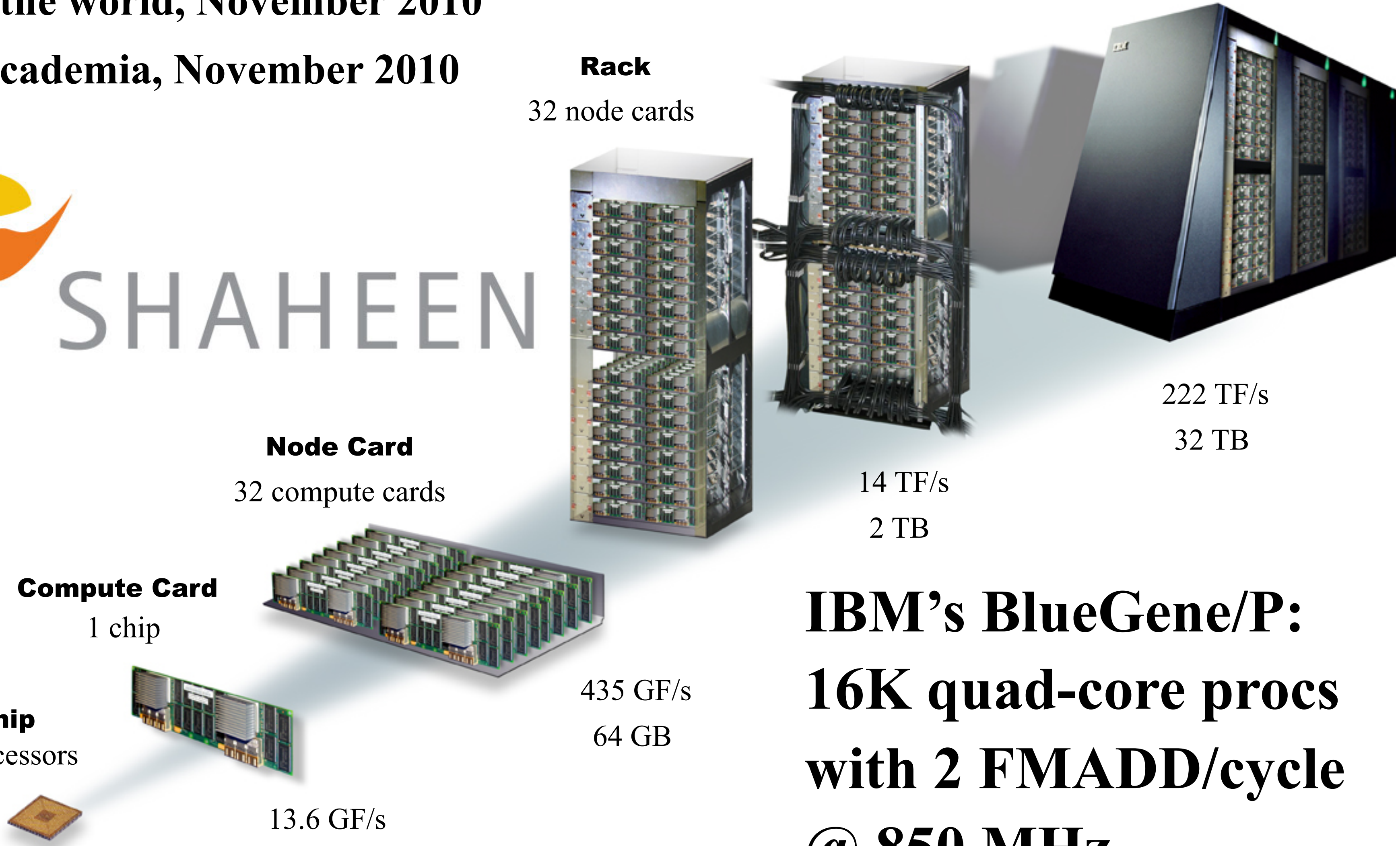
The initiative begins with a first call on Exascale Computing, further calls on other topics are expected in 2011 and 2012.

KAUST's petascale system from IBM

#34 in the world, November 2010

#6 in academia, November 2010

SHAHEEN



System
16 racks

222 TF/s
32 TB

Rack
32 node cards

14 TF/s
2 TB

Node Card
32 compute cards

435 GF/s
64 GB

Compute Card
1 chip

13.6 GF/s
2 GB DDRAM

Chip
4 processors

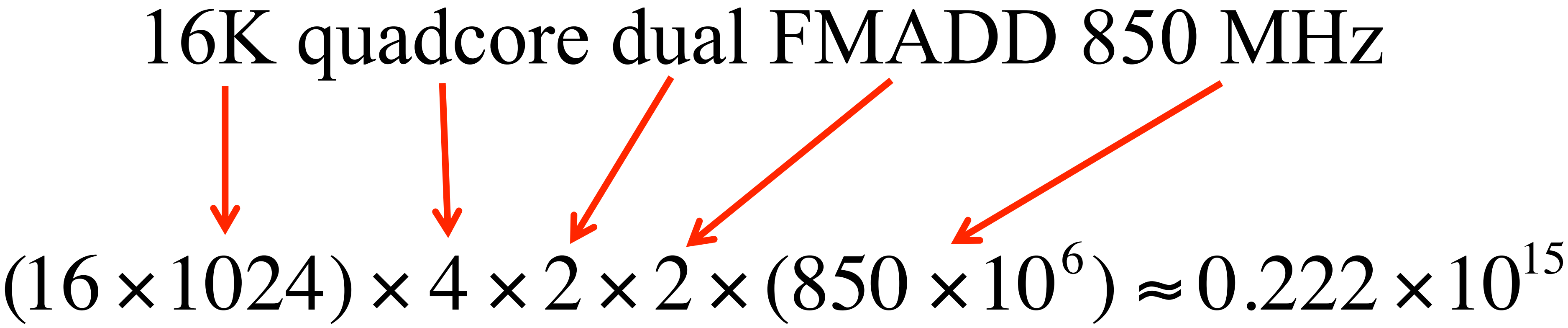
13.6 GF/s
8 MB EDRAM

IBM's BlueGene/P: 16K quad-core procs with 2 FMADD/cycle @ 850 MHz

= *0.222 Pflop/s*

How to calculate peak flop/s

Multiply together the number of processors, the number of cores per processor, the number of floating point operations that can be done in each core on each clock cycle , and the clock rate in cycles per second:



16K quadcore dual FMADD 850 MHz

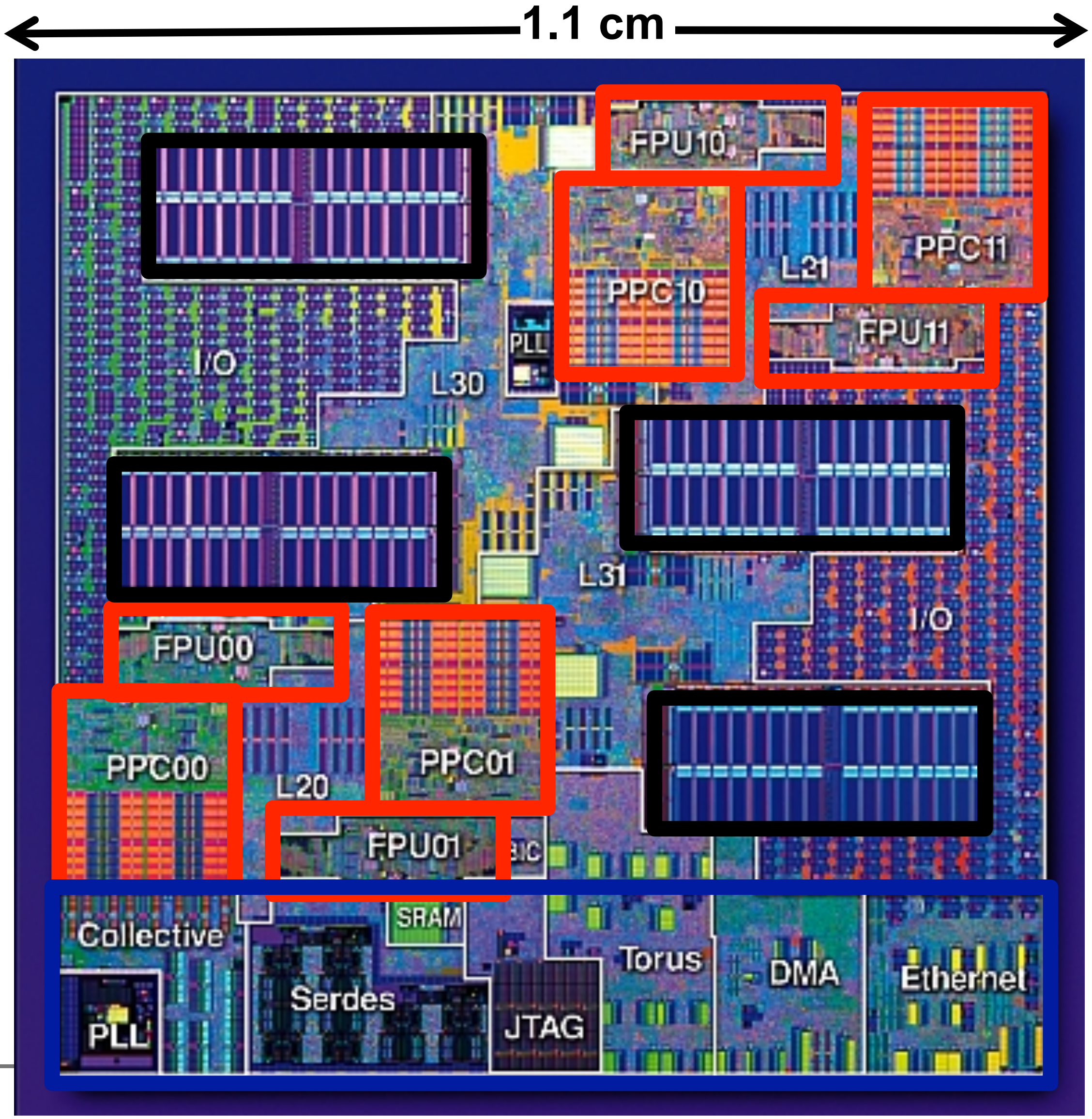
$$(16\times 1024)\times 4\times 2\times 2\times(850\times 10^{6})\approx 0.222\times 10^{15}$$

Layout of BG/P chip

BG/P System on a chip:

4 PowerPC 450 cores w/dual FPUs, memory controllers, ethernet controller, adaptive router with DMA, a global reduction network, 8MB of embedded DRAM – all in 208 million transistors on a 0.13μm line width

(a third of a trillion transistors in Shaheen’s processor chips, overall)



Where to find the “big iron” today

312 of the “Top 500” computer systems in the world are operated by industry (http://www.top500.org/)

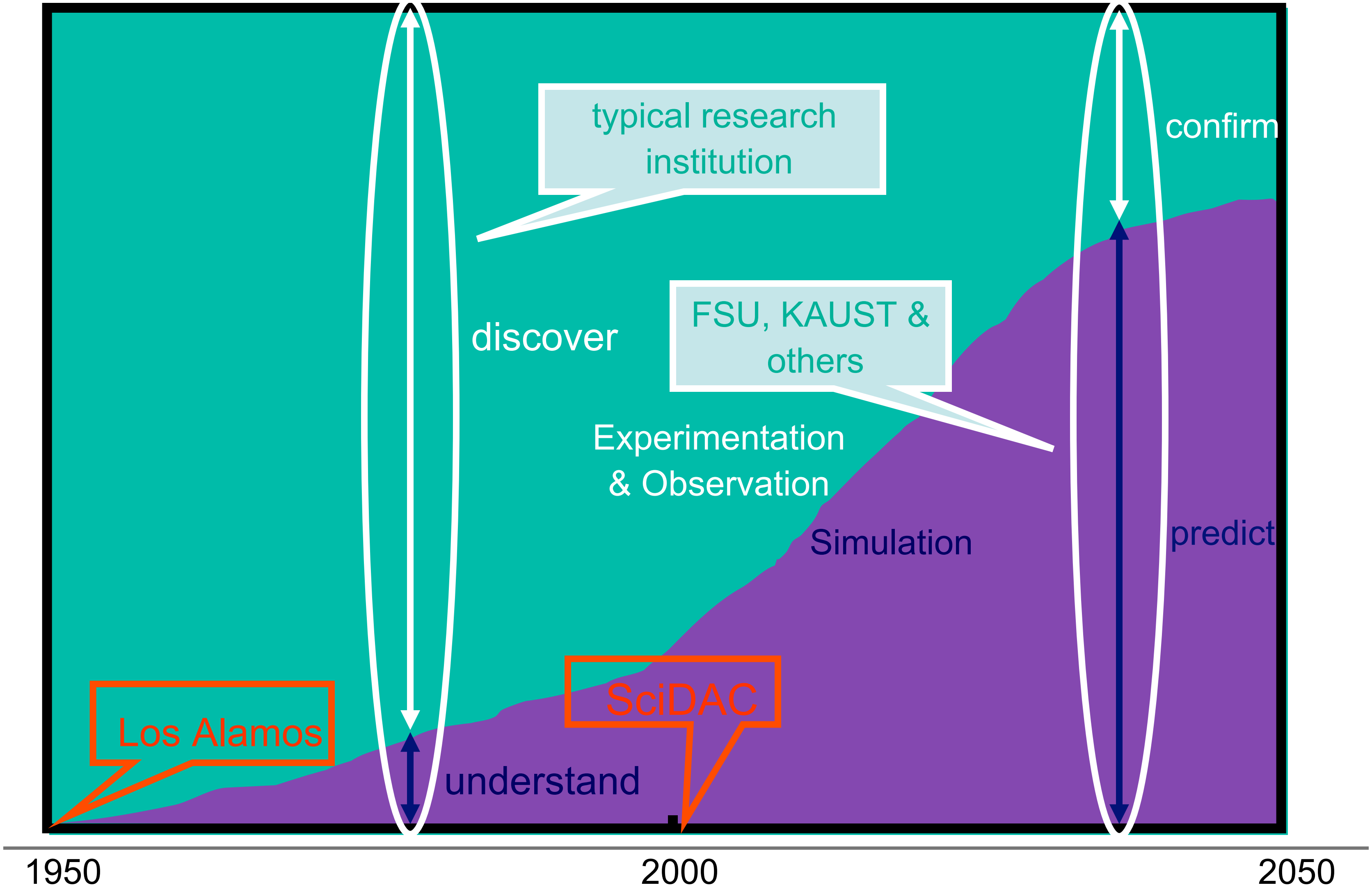
Sector	# sys	% share	R_{max} sum	R_{peak} sum	Proc sum
Industry	312	62.4	8.99 Pf	16.33 Pf	1,547,276
Research	91	18.2	12.48 Pf	16.27 Pf	2,155,447
Academic	79	15.8	5.57 Pf	7.13 Pf	826,584
Government	9	1.8	0.58 Pf	0.76 Pf	74,744
Classified	5	1.0	0.14 Pf	0.20 Pf	22,844
Vendor	4	0.8	0.22 Pf	0.27 Pf	37,732
TOTALS	500	100.0	27.98 Pf	40.95 Pf	4,664,627

Petaflop/s

Questions to consider

- Why the push to the exascale?
- What will systems a thousand times faster than today's petascale systems look like architecturally?
- What will be the implications of their budgets for power and acquisition?
- What should we do about it to prepare, algorithmically?

Evolving roles of simulation and data



Why push to extreme scale?

- 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”

The third paradigm

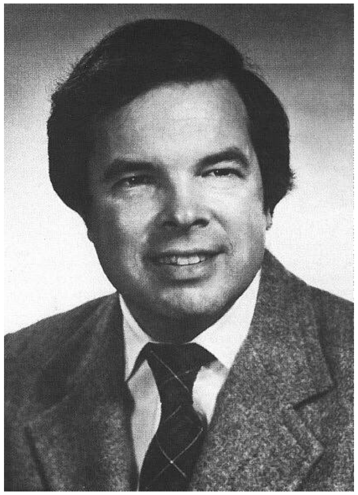
The "third paradigm" paper (1986)

"During its spectacular rise, the computational has joined the theoretical and the experimental branches of science…"

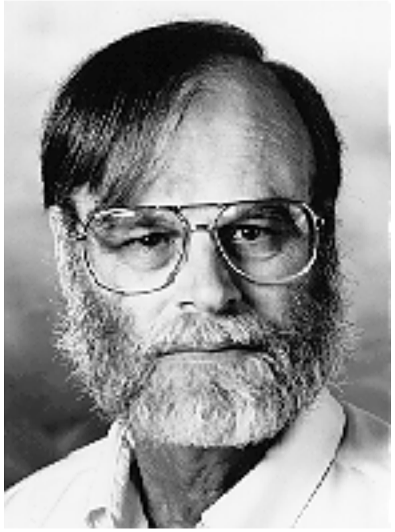
– Peter D. Lax, in *J. Stat. Phys*., 43:749-756

The "Grand Challenge" paper (1989)

"Grand Challenges of Computational Science" … define opportunities to open up vast new domains of scientific research, domains that are inaccessible to traditional experimental or theoretical models of investigation." – Kenneth G. Wilson, in *Future Generation Computer Systems*, 5:171-189



The fourth paradigm



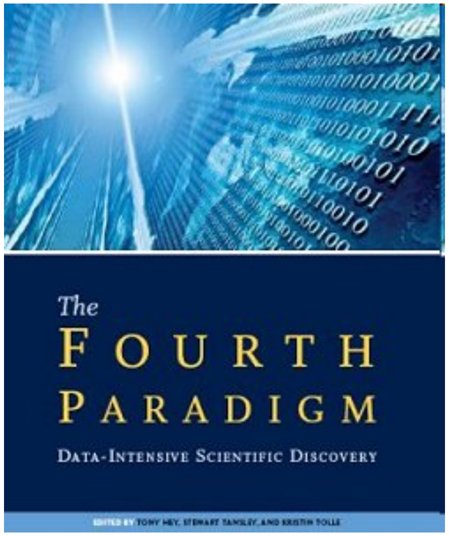
The Data-centric world paper (2006)

"The data need to be curated with metadata, stored under a schema with a controlled vocabulary, and indexed and organized for quick and efficient temporal, spatial, and associative search."

– James N. Gray *et al.*, in *IEEE Computer*, 39:110-112

"….Authors in this volume … refine an understanding of this new paradigm from a variety of disciplinary perspectives."

— Gordon Bell, *Microsoft Research*



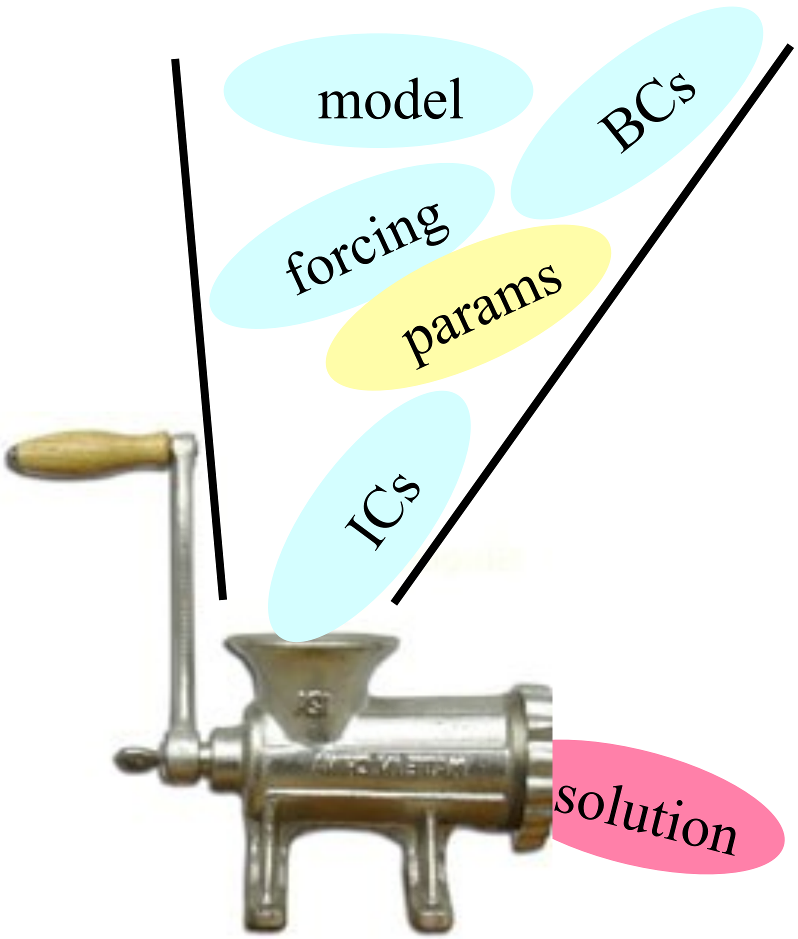
The FOURTH PARADIGM

DATA-INTENSIVE SCIENTIFIC DISCOVERY

EDITED BY TONY HEY, STEWART TANSLEY, AND KRISTIN TOLLE

Combining the paradigms: forward vs. inverse problems

forward problem



model

BCs

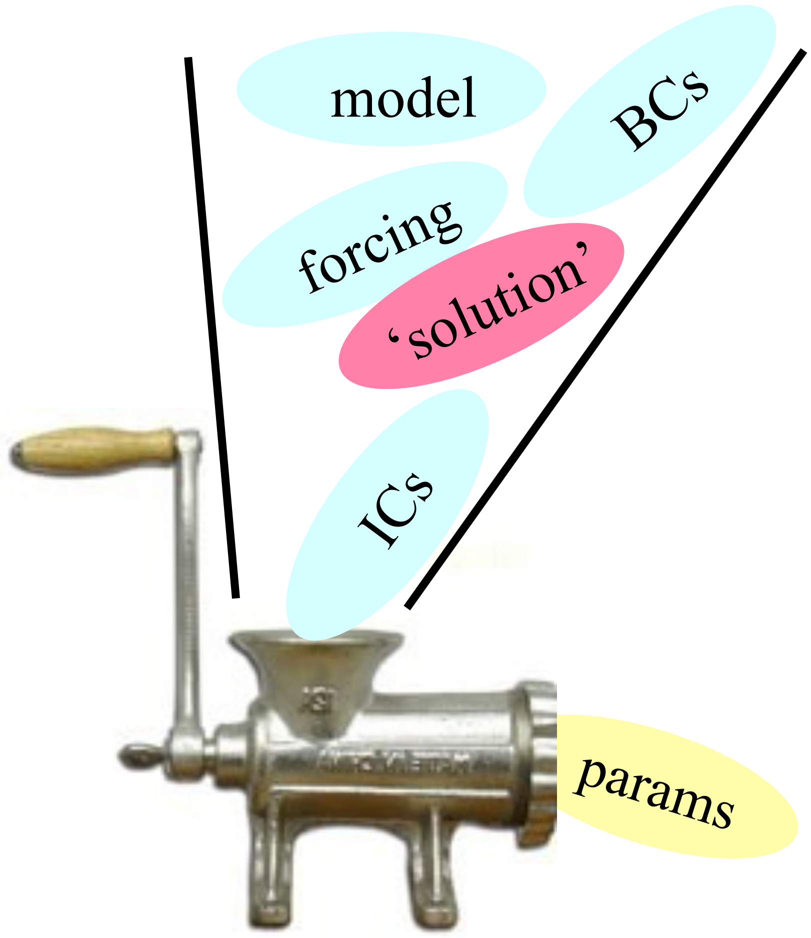
forcing

params

ICs

solution

inverse problem



model

BCs

forcing

‘solution’

ICs

params

+ regularization

How would these needs be felt at, *e.g.*, an oil company?

- Better resolve model's full, natural range of length or time scales
  - Discretize a reservoir into more layers and horizontal cells
- Accommodate physical effects with greater fidelity
  - Replace black oil assumption with fuller compositional effects
- Allow the model degrees of freedom in all relevant dimensions
  - Here, all three space dimensions, plus time
- Better isolate artificial boundary conditions or better approach realistic levels of dilution
  - Use additional cells outside of the production and injection zones to buffer the zones where data is needed from unknown geology and fluxes

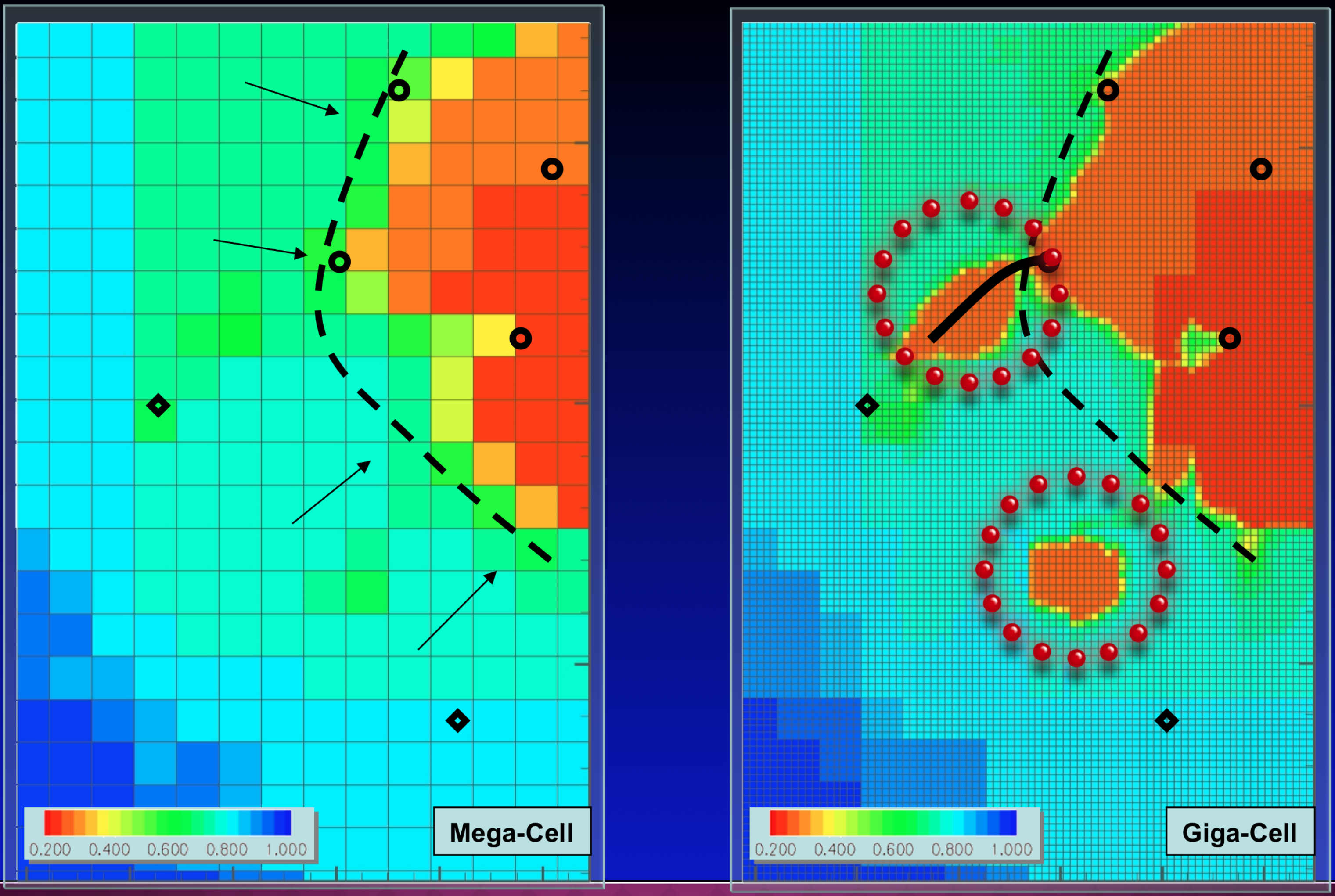
How would these needs be felt at, *e.g.*, an oil company?

- Solve an inverse problem, or perform data assimilation
  - Match simulation with drilling and historical production records to better estimate unknown parameters in the model and nudge the simulation towards reality, where and when reality is known
- Combine multiple complex models
  - Unify the simulations of adjacent fields to capture the effects of producing one of them on the other, monitor production through changes in seismic profiles, or model surface subsidence
- Perform optimization or control
  - Select a strategy for injection and pumping in the thousands of wells per reservoir

How would these needs be felt at, *e.g.*, an oil company?

- Quantify uncertainty
  - Given the many unresolvable uncertainties in program inputs, bound the error in the outputs in terms of errors in the inputs
- Improve statistical estimates
  - Treating uncertain inputs as random, improve output estimates

GigaPOWERS™ Impact
High Resolution - Higher Recovery



c/o A. Dogru (used by permission)

Need for extreme scale goes far beyond these, however!

- Oil companies are vast, with dozens of reservoirs “upstream” and many refineries and transportation systems “downstream”
- Oil companies function under many constraints for product supply and must seek to *maximize profit* while *satisfying output demands* in many different product streams producible from the same crude
- Tens of thousands of “valves” (literal and figurative, like workforce and other controls) need to be scheduled continuously
- Mathematically, this is a massive, nonlinear and possibly non-robust constrained optimization problem – insatiably power hungry

How should such resources be managed?

- They are too complex for a self-contained, self-consistent theory
- They are unsuitable for experiment, because you can only do the experiment (*e.g.*, exploiting a reservoir) once
- Engineering heuristics are useful (and used!), but
  - gone are the days when employees spent decades understanding a single reservoir
  - the external forcings (*e.g.*, world economy) change daily, making history less useful
- *Simulation* is an incredibly useful tool for exploring scenarios experimentally in a virtual world
- *Data mining and machine learning* may be even more useful tools in the future

Dominant findings on models and scaling on today's petascale simulations (ref: IESP)

- Predominantly bulk synchronous, MPI and SPMD, either domain-, particle-, or other object-decomposed
- Electronic Structure codes, however, are dominantly *not* MPI, but global shared memory (e.g., GlobalArrays)
- Other models are Charm++ (NAMD) and distributed objects built on top of active messages and Pthreads
- Some codes have multiple phases with not necessarily compatible load-balanced decompositions
- Typically already running hybrid MPI/OpenMP with "small" numbers of cores, up to 48
- We can typically weak-scale without serious loss of scaling out to the edge of today's machines (300K cores), and theoretically beyond

Dominant findings on resources

- Memory bandwidth
  - ◆ Majority of the apps are already bandwidth limited in most phases even with relatively few cores
- Flop/s per byte of storage
  - ◆ Generally linear or log-linear, in weak scaling
- I/O versus computation
  - ◆ With notable exceptions that are I/O intensive, the majority of our peta-apps need intensive I/O only at start-up and in dumping the output
  - ◆ We expect check-pointing to become much more intensive at the exascale, and perhaps limiting, even with users taking charge of check-pointing minimal state

Dominant findings on resources, cont.

- Communication versus computation
  - In weak scaling there is generally a constant ratio of near-neighbor communication to computation, represented by a minimum collections of vertices, cells, particles, etc., per processor
- Synchronization versus computation
  - We require frequent global collectives
    - At least once per timestep in evolutionary codes
    - Even more frequently when implicit linear algebra needs to be performed within each timestep

Remaining presentation plan

- Reflect briefly on progress in high-end scientific computing
  - as captured in ACM Gordon Bell prize trends
  - as analyzed in some of the eight U.S. DOE “extreme scale” workshops of 2009-2010 (extremescale.labworks.org)
- Peek briefly at structure of a core motivating application
- Take a look at a few hurdles and possible solutions in algorithmic and programming model development arenas

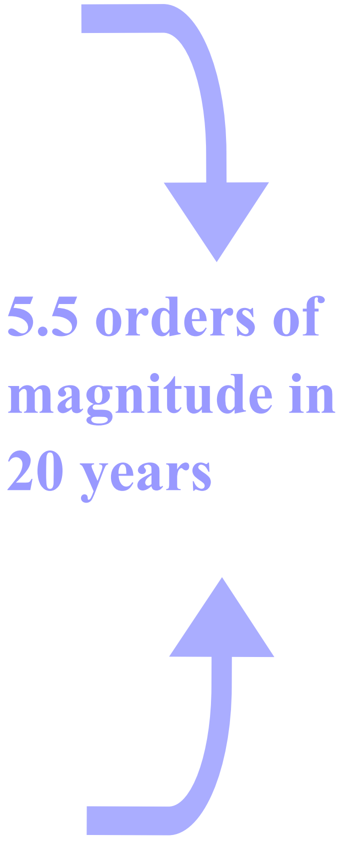
Tracking third paradigm progress: Gordon Bell Prize "peak performance"

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	DFT	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	DFT	Elec. Struct.	131,072	BG/L	207,000
2007	MD	Kelvin-Helm.	131,072	BG/L	115,000
2008	DFT	Crystal Struct.	150,000	XT-5	1,352,000
2009	DFT	Nanoparticles	223,000	XT-5	2,330,000
2010	FMM	Physiology	196,608	XT-5	780,000

Six orders of magnitude in 20 years

Tracking third paradigm progress: 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
2009	Gravitation & turbulence	cluster (GPU)	0.0081



5.5 orders of magnitude in 20 years

Gedanken experiment: How to use the palm date, as its price slides downward?

In 2011, at $16/kg:
eat, as a delicacy
By 2014, at $4/kg:
substitute for other oils and sugars
By 2017, at $1/kg:
use as a feedstock for biopolymers, plastics, etc.
By 2020, at $0.25/kg:
heat homes
By 2023, at $0.06/kg:
pave roads ☺

The cost of computing has been on a curve *much better than this* for two decades. Finally, after everyone else, *scientists and engineers* (very conservative people, on the whole) are planning increasing uses for it…

Whimsical remarks on simulation progress measured by Bell, since 1988

- If similar improvements in speed (10^6) had been realized in the airline industry, a 19-hour flight (e.g., SIN-EWR) would require one-fifteenth 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 (10^5) had been realized in the higher education industry, tuition room and board (at a college in the USA) would cost about $0.20 per year

Exascale considerations

- Applications
  - What do we want to simulate at the exascale, why, and in what best formulation or sets of formulations?
- Architectures
  - What are the hurdles of granularity, cost, power, programmability, systems software, reliability (resilience) in delivering exascale systems, and how can they be surmounted?
- Algorithms
  - How can we get the applications to run on architectures that are physically and fiscally achievable ("co-design" of architecture and application)

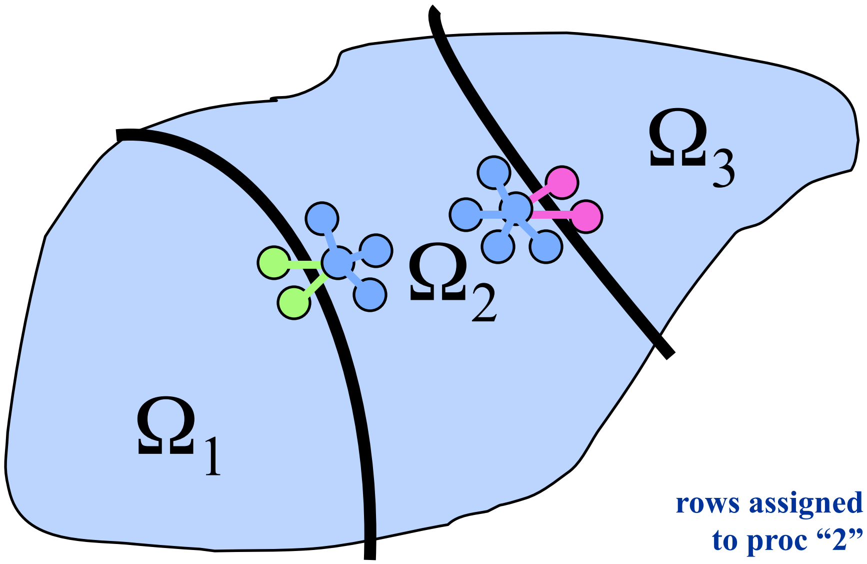
Philosophy of an algorithmicist

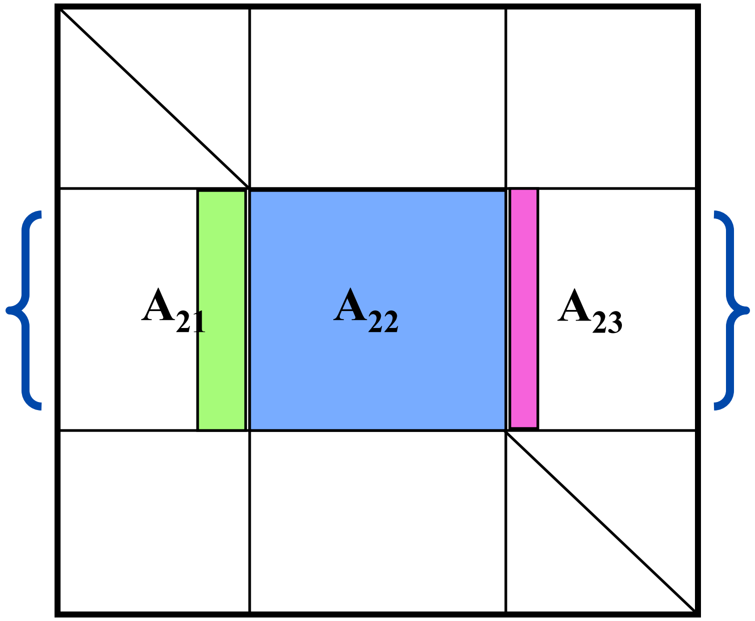
- Applications are ***given*** (as function of time)
- Architectures are ***given*** (as function of time)
- Algorithms and software ***must be adapted or created*** to bridge to hostile architectures for the sake of the complex applications
    - as important as ever today, with transformation of Moore's 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

How are problems like these solved at the petascale today?

- *Iterative methods* based on *domain decomposition* and *message-passing*
  - Each individual processor works on a subdomain of the original problem and exchanges information at its boundaries with other processors that own subdomains with which it interacts causally, to evolve in time or to establish equilibrium (steady state)
- The programming model is SPMD/BSP/CSP
  - Single Program, Multiple Data
  - Bulk Synchronous Programming refers to alternating nearly uniformly sized chunks of work on each processor with bursts of exchanges of data
  - a.k.a. Communicating Sequential Processes
- Nearly all successful large-scale simulations are built this way today – and this may change radically at the exascale

SPMD parallelism with domain decomposition





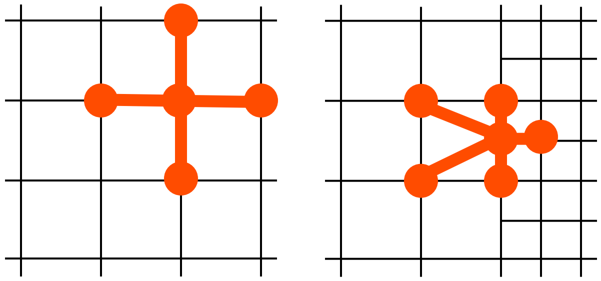
Partitioning of the grid induces block structure on the system matrix (Jacobian)

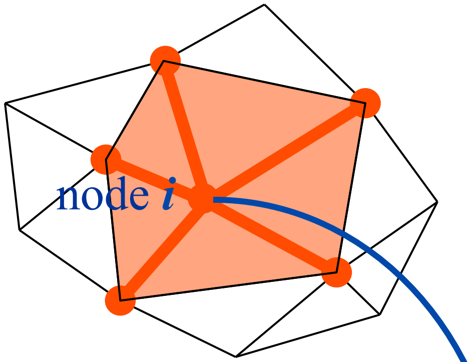
Domain decomposition is relevant to any local stencil formulation

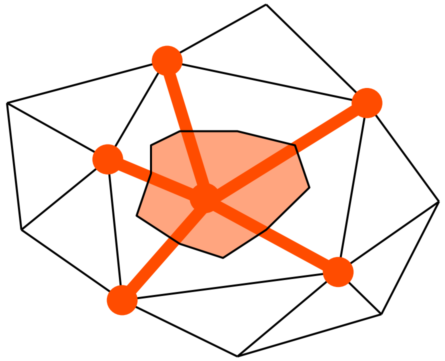
finite differences

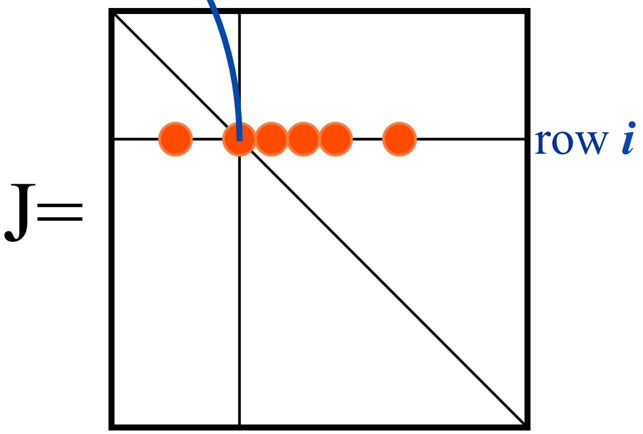
finite elements

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**

Krylov-Schwarz: a linear solver "workhorse"

$$Ax = b$$

$$x = \underset{v\in V\equiv\{b,Ab,A^2b,\cdots\}}{\arg\min} \{Av - b\}$$

- Krylov
  - accelerator

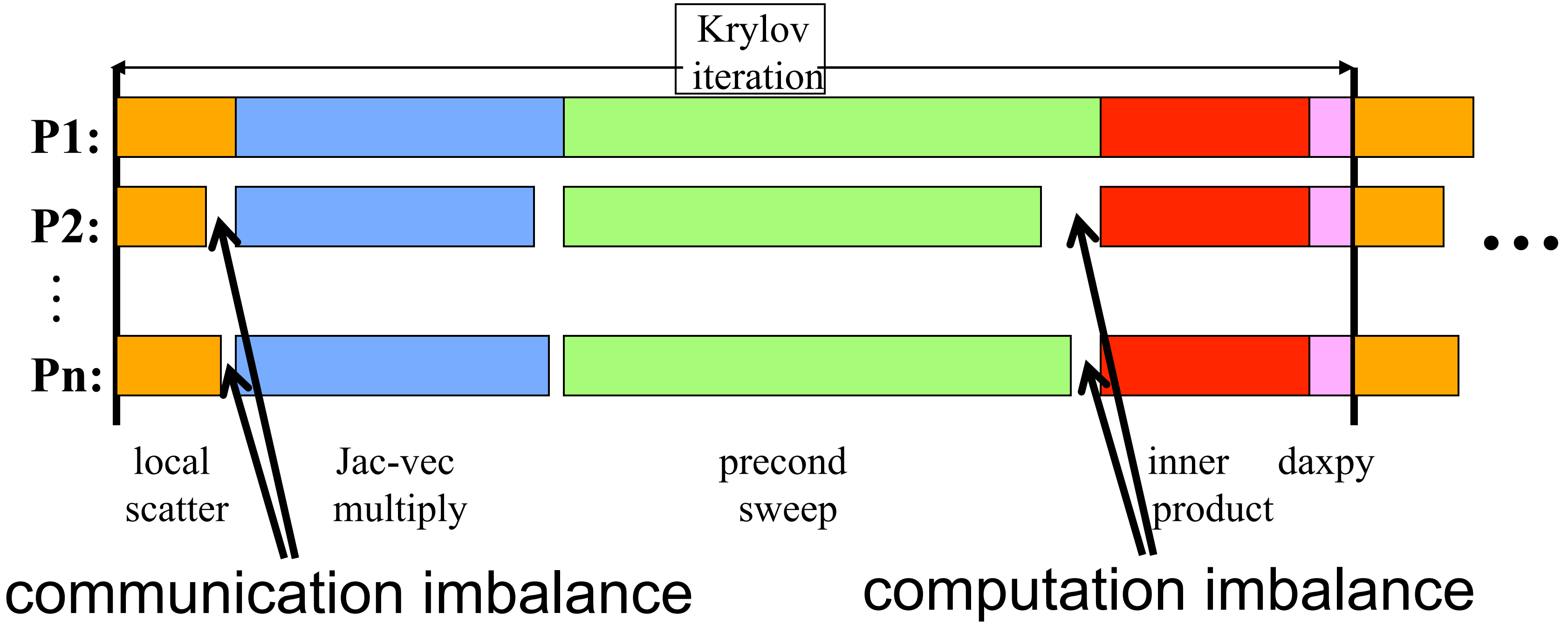
- *spectrally adaptive*

$$B^{-1}Ax = B^{-1}b$$

$$B^{-1} = \sum_i R_i^T (R_i A R_i^T)^{-1} R_i$$

- Schwarz
  - preconditioner
- *parallelizable*

Workhorse innards: Krylov-Schwarz, a Bulk Synchronous Process ("BSP")

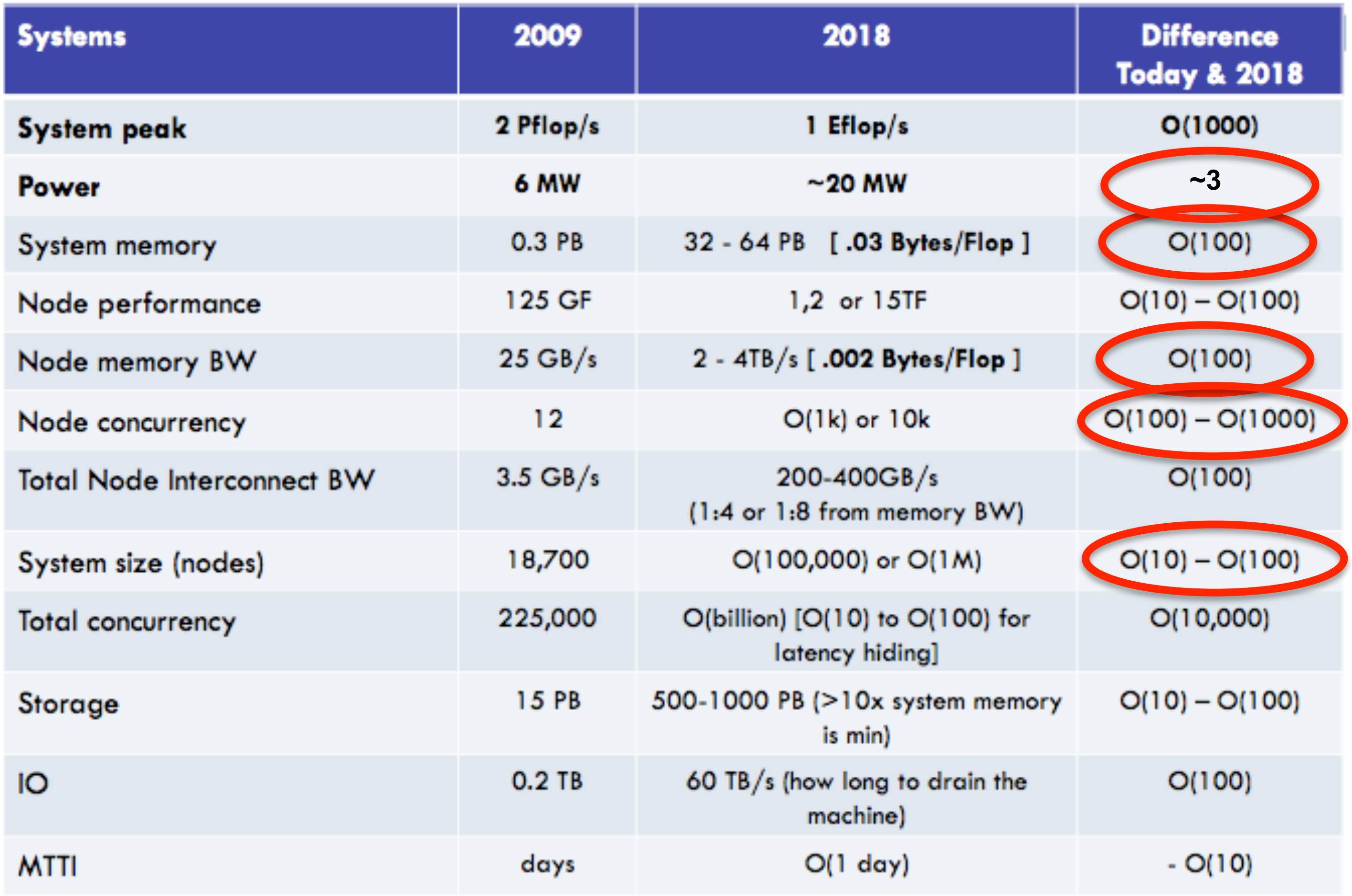


Idle time due to load imbalance becomes a challenge at, say, one million cores, when *one* processor can hold up *all* of the rest at a synchronization point

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

- Many paths beyond today's CMOS silicon-based logic
- Earliest and most significant post-CMOS device improvement *may* be carbon nanotube memory, but not in 10 years
  - up to tens of GB on a 1 cm-square die
  - will deal directly with the "memory wall" problem
- Two paths from peta- to exa-
  - IBM: BlueGene's successor, maybe at 22nm linewidth technology – some architectural merger of BlueGene, Power, and Cell
  - All others: GPGPU-based spinoff
- 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, and made less synchronous

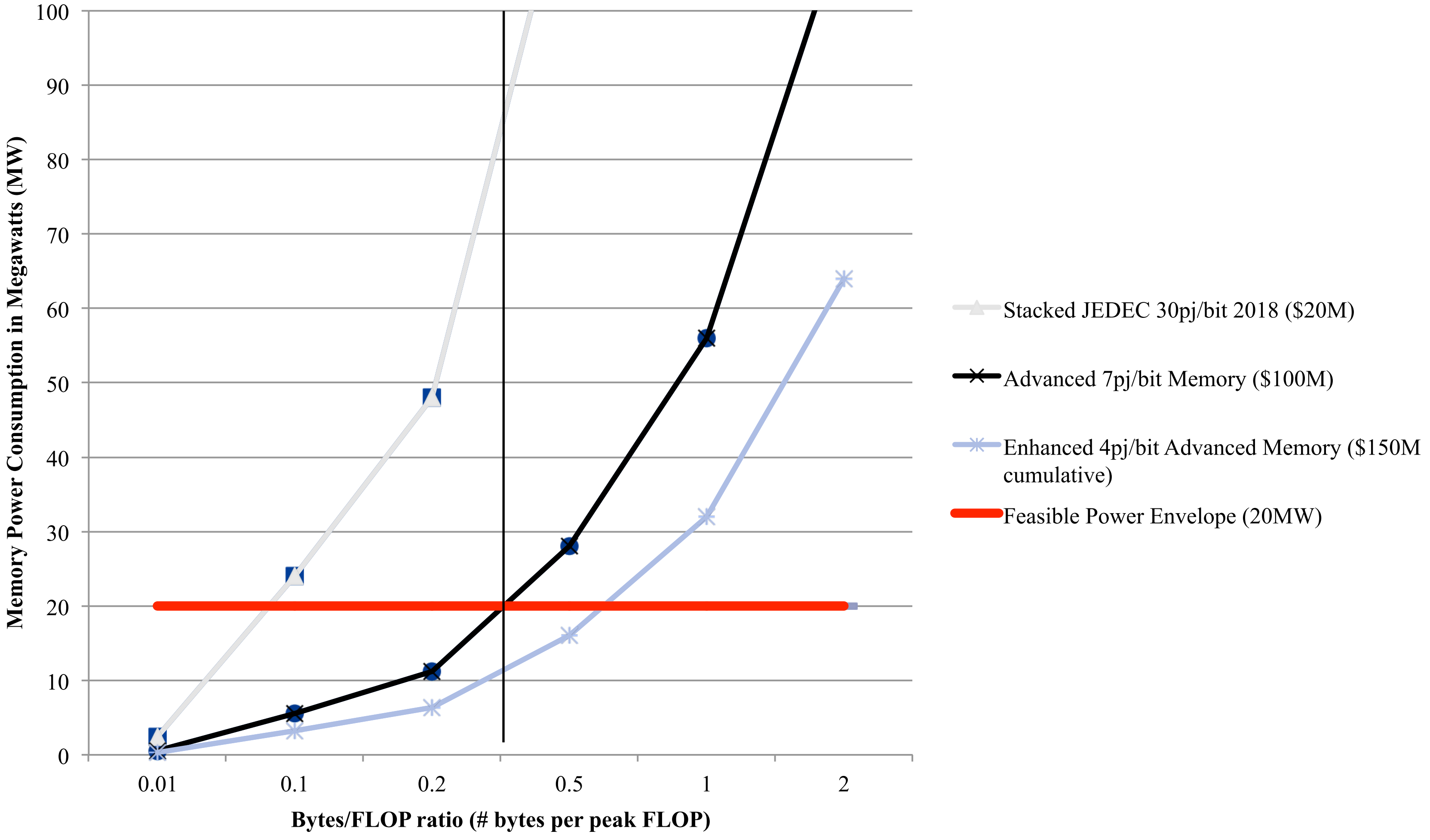
Prototype exascale hardware: a heterogeneous, distributed memory *GigaHz KiloCore MegaNode* system



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)

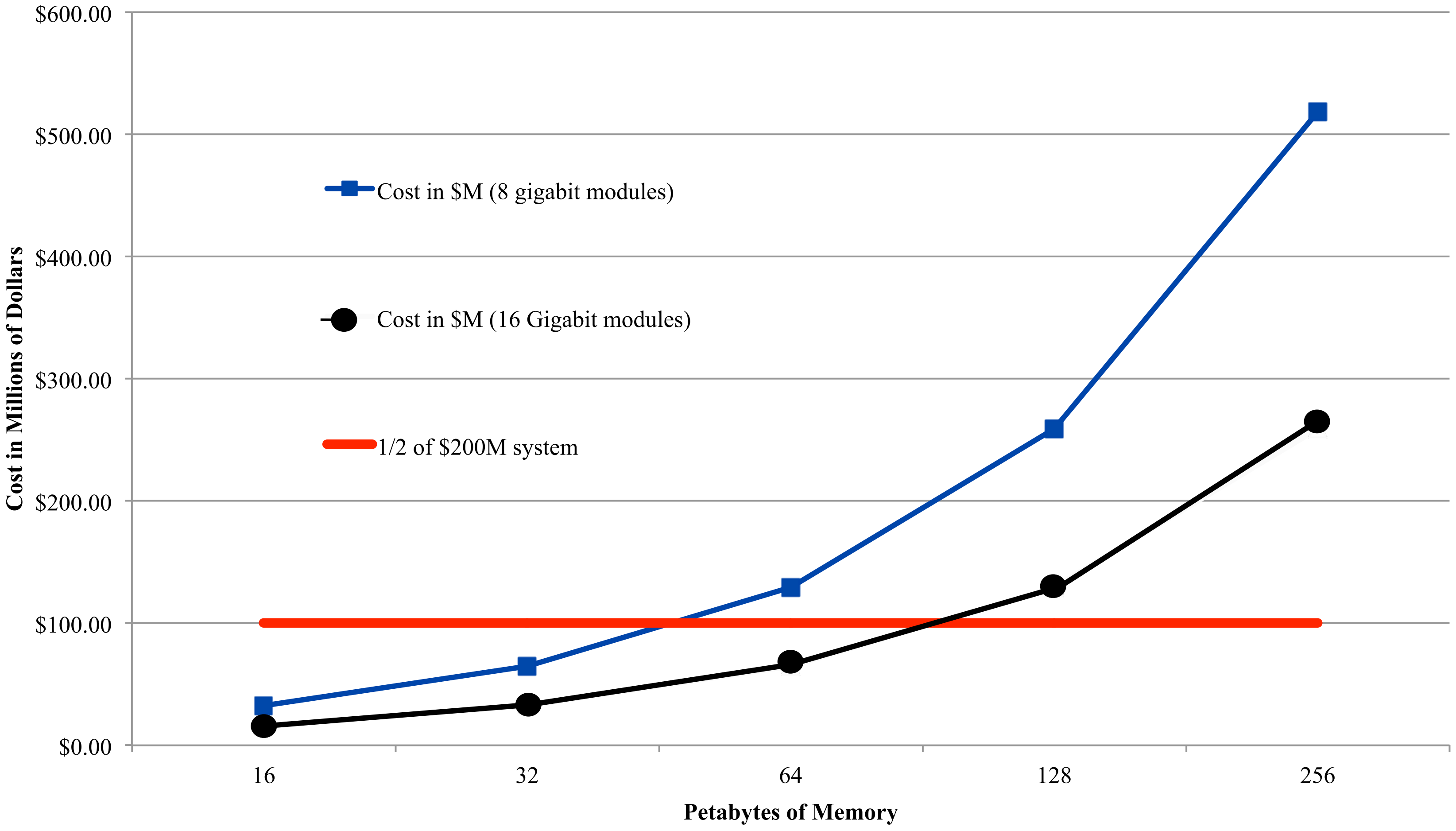
c/o P. Beckman

Hurdle #1: memory bandwidth eats up the entire power budget



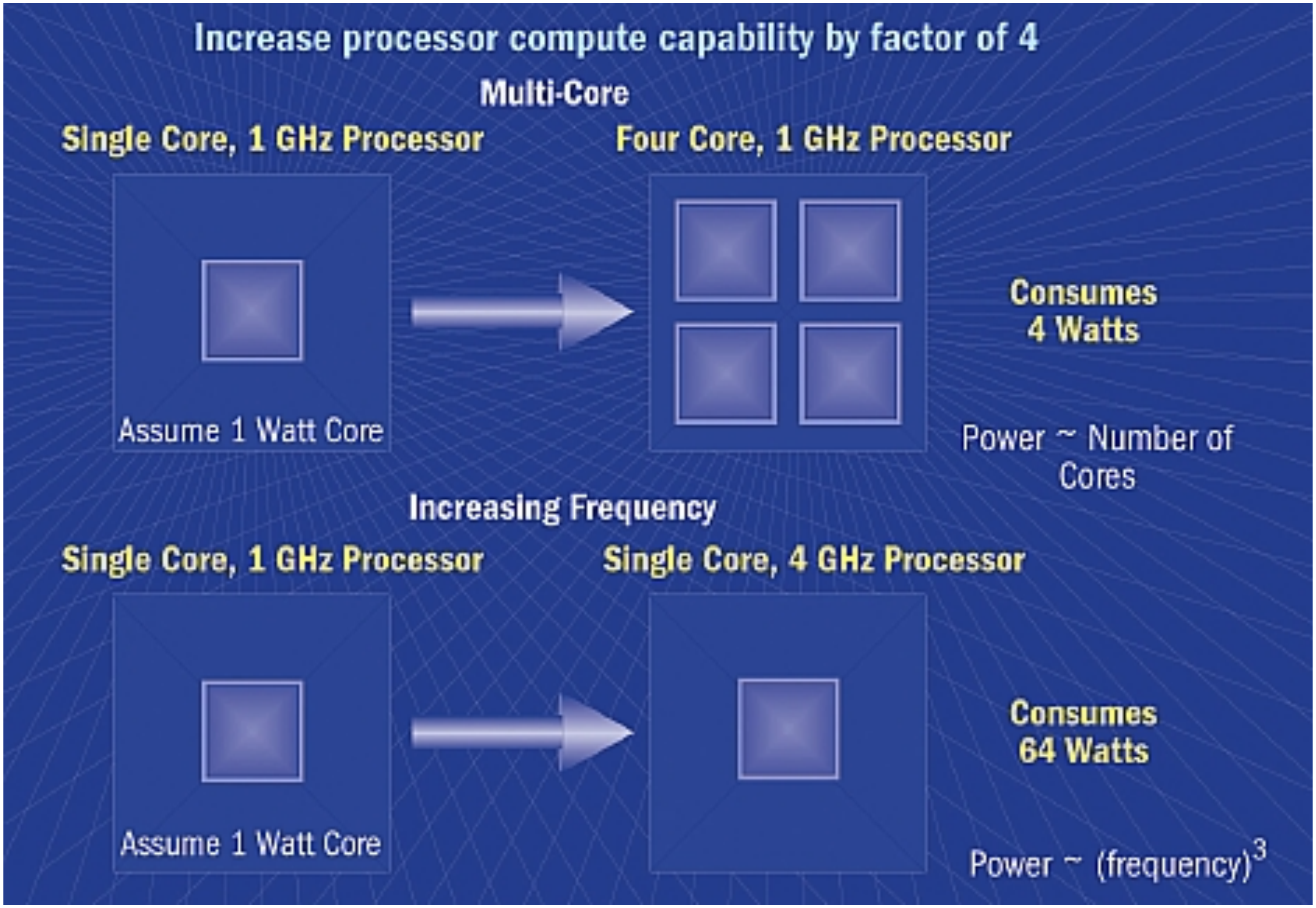
c/o John Shalf, LBNL

Hurdle #2: memory capacity eats up the entire fiscal budget



c/o John Shalf, LBNL

Hurdle #3: 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

Implications

- Expanding the number of nodes (processor-memory units) to 10^6 is *not* a serious threat to algorithms that lend themselves to well-amortized precise load balancing (like PDEs)
  - Provided that the nodes are performance reliable
- The 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
- 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

Implications, cont.

- 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

Sources of nonuniformity

- System
  - Manufacturing, dynamic power management, soft errors, hard component failures, OS jitter, software-mediated resiliency, TLB/cache performance variations, network contention, etc.
- Algorithmic
  - Physics at gridcell 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

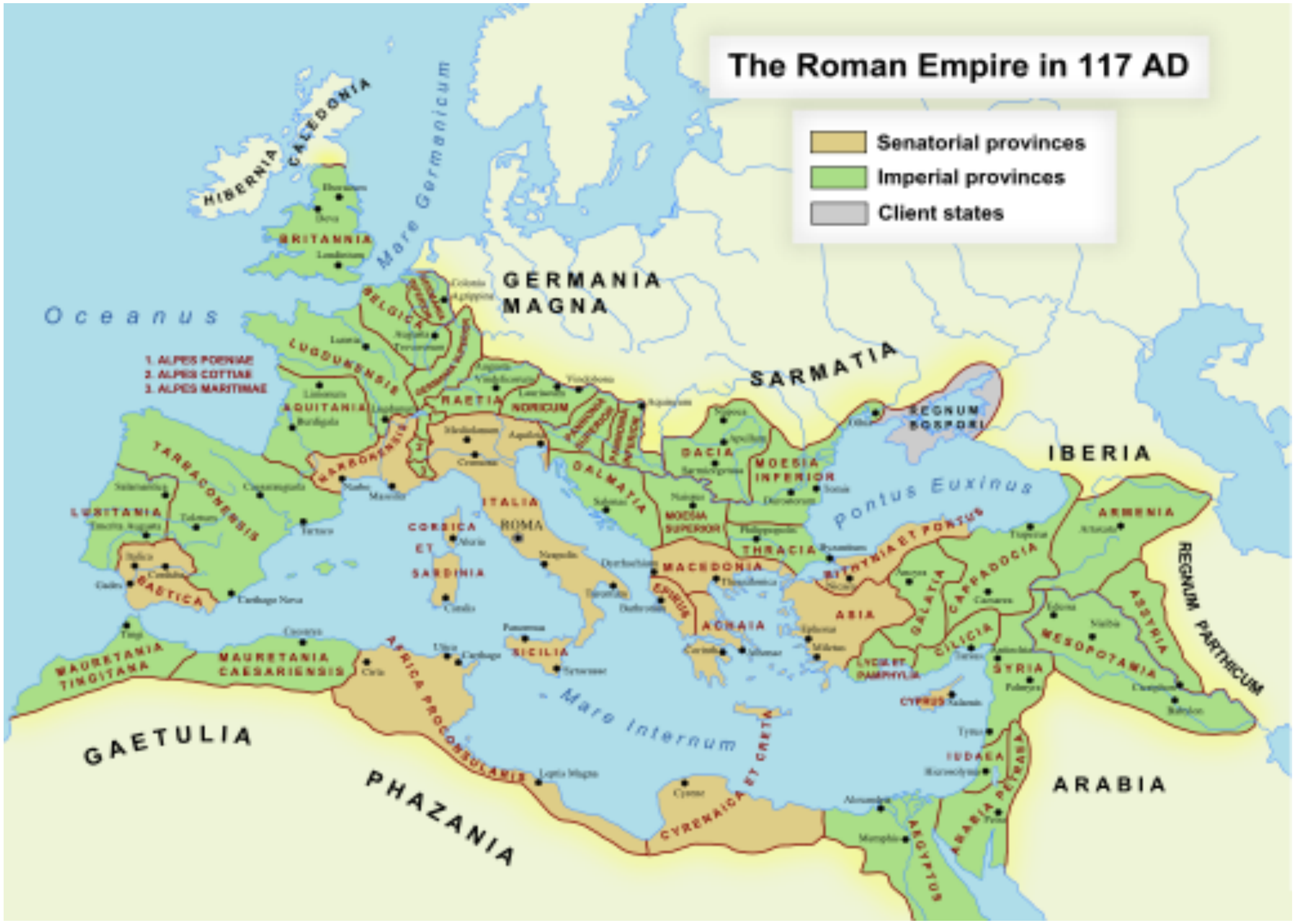
Pax MPI is over

Pax MPI

- 1994-2010*
- Universal language
  - enabled free trade in applications and libraries
- Safe porting
  - codes linked and ran "first time" across desktops, clusters, high-end systems
- Was protested, but conferred advantages

* MPI continues for a *long time*. However, the empire of MPI-only begins to disintegrate into non-universal programming model groups.

Pax Romana


The Roman Empire in 117 AD
🟨 Senatorial provinces
🟩 Imperial provinces
⬜ Client states
Oceanus
Mare Germanicum
HIBERNIA
CALEDONIA
BRITANNIA
Eboracum
Deva
Londinium
GERMANIA MAGNA
SARMATIA
BELGICA
LUGDUNENSIS
AQUITANIA
Lutetia
Burdigala
Lugdunum
Colonia Agrippina
Augusta Treverorum
1. ALPES POENIAE
2. ALPES COTTIAE
3. ALPES MARITIMAE
RAETIA
NORICUM
Vindobona
Aquincum
Mediolanum
Aquileia
Cremona
NARBONENSIS
Narbo
Massilia
TARRACONENSIS
LUSITANIA
BAETICA
Caesaraugusta
Tarraco
Toletum
Emerita Augusta
Italica
Corduba
Gades
Carthago Nova
ITALIA
ROMA
Neapolis
Brundisium
CORSICA ET SARDINIA
Aleria
Caralis
SICILIA
Panormus
Syracusae
DALMATIA
Salonae
DACIA
Sarmizegetusa
Apulum
Napoca
MOESIA INFERIOR
MOESIA SUPERIOR
Naissus
Tomis
Durostorum
THRACIA
Philippopolis
Byzantium
MACEDONIA
Thessalonica
Dyrrhachium
EPIRUS
ACHAIA
Corinthus
Athenae
Olbia
REGNUM BOSPORI
Pontus Euxinus
IBERIA
ARMENIA
Artaxata
BITHYNIA ET PONTUS
Nicaea
ASIA
Ephesus
Miletus
GALATIA
Ancyra
CAPPADOCIA
Caesarea
Trapezus
CILICIA
Tarsus
LYCIA ET PAMPHYLIA
CYPRUS
Salamis
SYRIA
Antiochia
Palmyra
Tyrus
MESOPOTAMIA
Edessa
Nisibis
ASSYRIA
Ctesiphon
Babylon
REGNUM PARTHICUM
IUDAEA
Hierosolyma
ARABIA PETRAEA
Petra
ARABIA
AEGYPTUS
Alexandria
Memphis
CRETA ET CYRENAICA
Cyrene
Mare Internum
AFRICA PROCONSULARIS
Carthago
Utica
Leptis Magna
Cirta
MAURETANIA CAESARIENSIS
Caesarea
MAURETANIA TINGITANA
Tingi
GAETULIA
PHAZANIA

Pax Romana

- 27 BC – 180 AD*
- Universal language
- Safe ports and roads
- Occupancy was protested, but conferred advantages

* Rome continues *for centuries*. However, the empire begins to disintegrate into non-universal governments and cultures.

Some exascale themes

- Clock rates cease to increase while arithmetic capacity increases dramatically w/concurrency*
- Storage capacity diverges exponentially below arithmetic capacity
- Transmission capacity diverges exponentially below arithmetic capacity
- Mean time between hardware interrupts shortens
- Billions of dollars/euros/etc. of scientific software hang in the balance until better algorithms arrive to span the architectural gap

*Moore's Law (1965) does not end, but Dennard scaling (1974) does

Moans from application developers

- The present consensus path to exascale is thousand-fold manycore
  - However, memory bandwidth is already limiting today's low core count nodes to less than 10% of peak on most apps, whose kernels offer little cache reuse (e.g., stencil ops or sparse matvecs)
  - Processors are cheap and (relative to memory) small in chip area and relatively low in power, so there is no harm in having them in excess most of the time, but the opportunities for exploiting the main new source for performance are undemonstrated for most applications
- While there is opportunity for combining today's individually high capability simulations into complex simulations, there is no silver bullet for merging the data structures of the separate applications
  - The data copying inherent in the code coupling will likely prevent exploitation of the apparent concurrency opportunities

Chief issues identified by apps groups of the International Exascale Software Project

- I/O
- Fault tolerance
- Reproducibility of computations
- Programming models and algorithms

I/O

- For some important apps, I/O is a likely bottleneck
  - For input, for output (including visualization), or for checkpointing, or any combination
- I/O must be acknowledged as primary for many apps (though certainly not all), but is beyond the scope of this talk

Fault tolerance

- IESP users reluctantly recognize that fault tolerance is a shared responsibility
  - It is too wasteful of I/O and processing cycles to handle faults purely automatically
- Different types of faults may be handled different ways, depending upon consequences evaluated by scientific impact
- Strategic, minimal workingset checkpoints can be orchestrated by application developers and users

Reproducibility

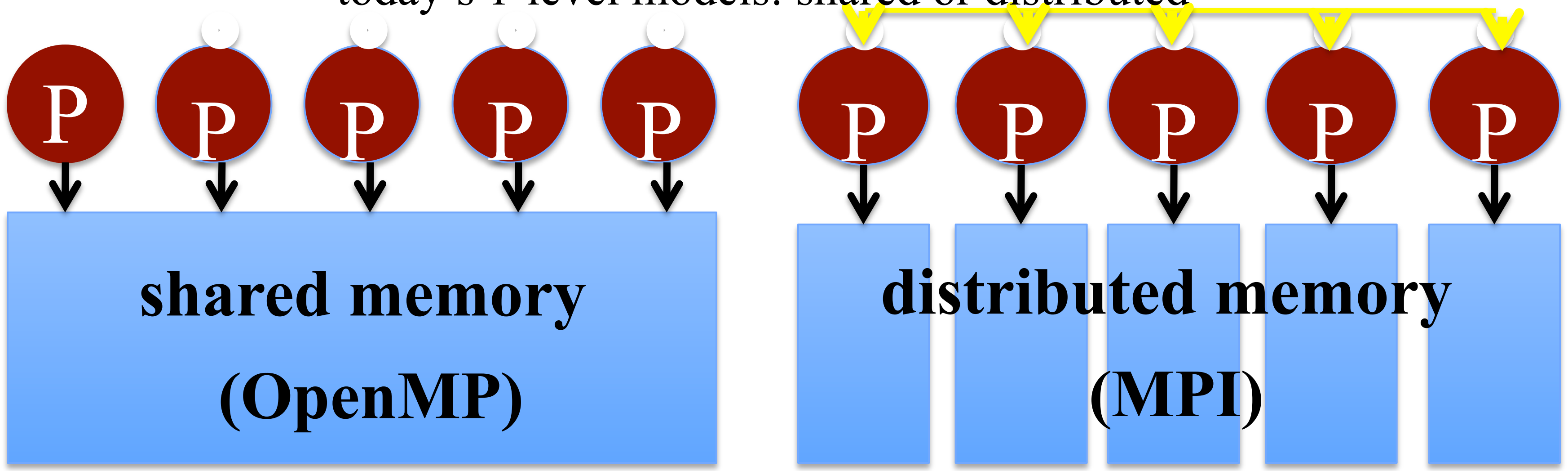
- IESP users realize that bit-level reproducibility is unnecessarily expensive most of the time
- Though scientific outcomes must be run- and machine-independent, we have no illusions about bit-level reproducibility for individual pairs of executions with the same inputs
  - Since operands may be accessed in different orders, even floating point addition is not commutative in parallel and on inhomogeneous hardware platforms; this has been true for a long time
  - A new feature, with an emphasis on low power (low voltage switching), is that lack of reproducibility may emerge for many *other* (hardware-based) reasons
  - If applications developers are tolerant of irreproducibility for their own reasons, e.g., for validation and verification through ensembles, then this has implications for considering less expensive, less reliable hardware

Programming model

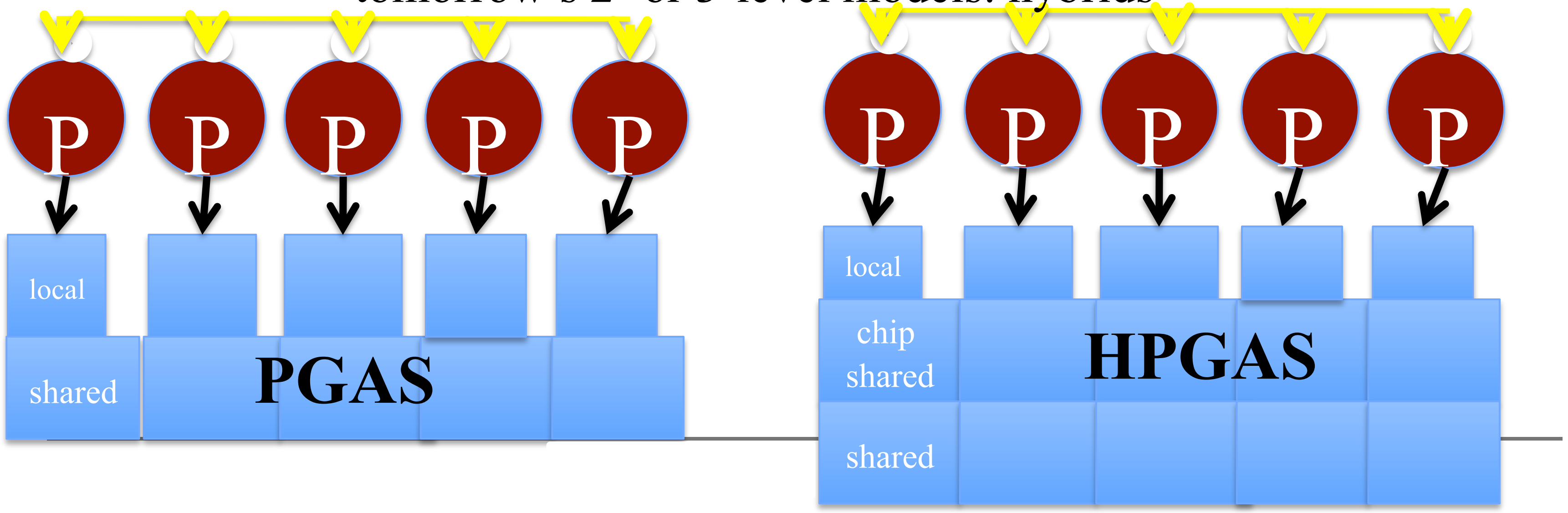
- Prior to possessing exascale hardware, users can prepare themselves by exploring new programming models
  - on manycore and heterogeneous nodes
- Attention to locality and reuse is valuable at all scales
  - will produce performance paybacks today *and* in the future
- New algorithms and data structures can be explored under the assumption that flop/s are cheap and moving data is expensive
- Bandwidth pressure can be reduced by considering mixed-precision algorithms, using lower precision wherever possible
- Relaxation of synchrony could relieve pressure on load balance

Evolution of parallel programming models: strong scaling within a node

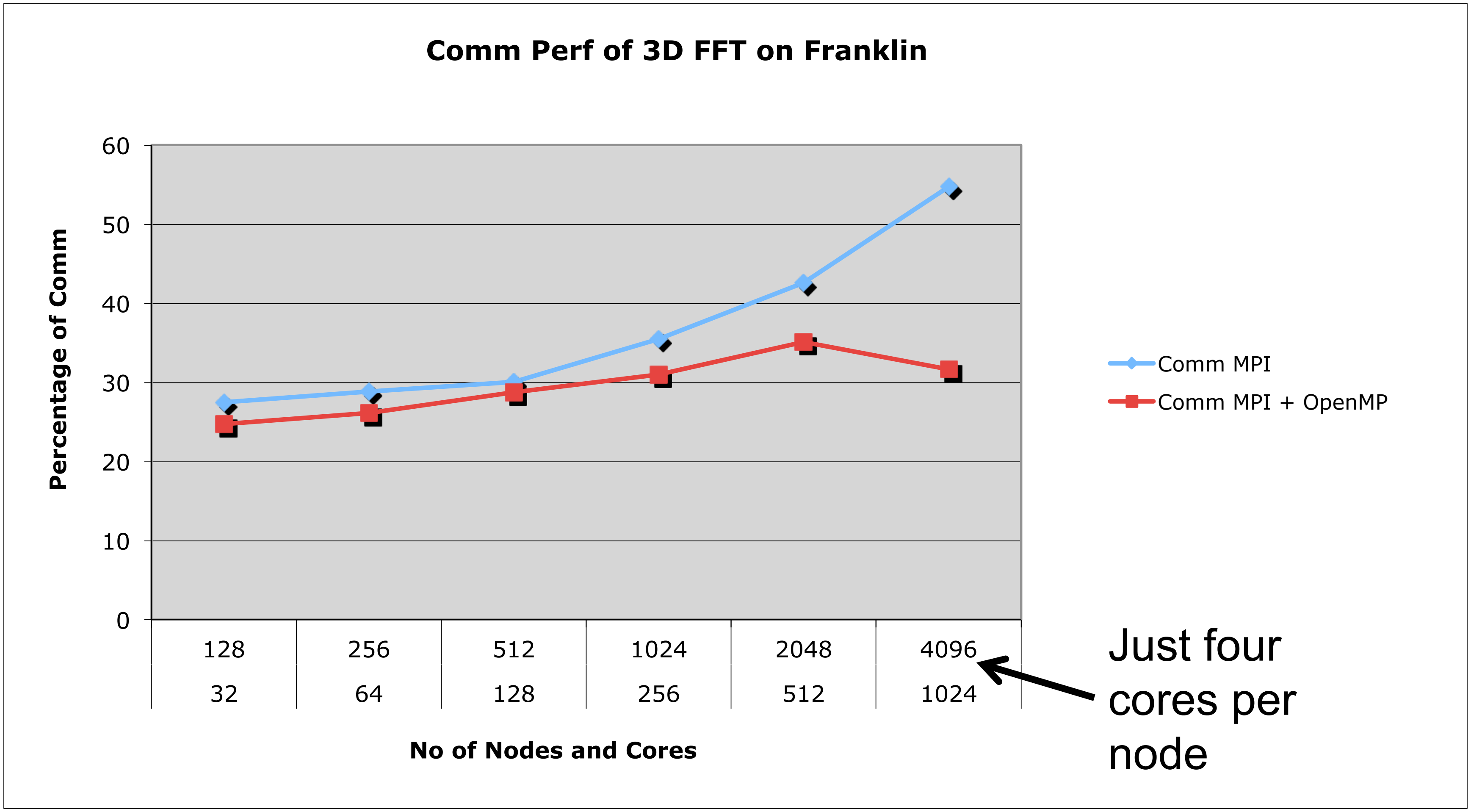
today’s 1-level models: shared or distributed



tomorrow’s 2- or 3-level models: hybrids



Benefits of 2-level parallelism for 3D FFT



Just four cores per node

c/o John Shalf, LBNL

Hybrid programming model for a full application: petascale bio-electro-magnetics

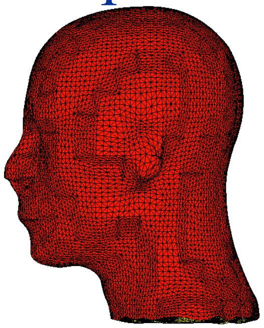
High-Fidelity BIOEM Simulation

- Minimize health risks & increase efficiency of wireless devices
- Existing results contradictory, e.g., do children's smaller heads absorb more radiation or allow deeper penetration? Need reproducible, reliable, high-accuracy, high-resolution simulations
- Solve coupled Maxwell's EM and Penne's Bio-Heat Transfer equations : Human & device models must resolve geometry, material, EM wavelength, and thermal mechanisms (*petascale problem*)
- Advance the state-of-the-art: Develop novel petascale BIOEM simulators, investigate and quantify modeling & analysis errors

Two-pronged Approach (FEM/BEM)

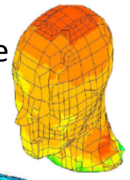
- Cross-validate, verify, & build confidence in unprecedented petascale results
- Evolve sub-mm resolution human body models
- Next generation EM solvers: FFT-accelerated integral-equation solvers (BEM prong); HP-adaptive differential-equation solvers (FEM prong); novel preconditioned iterative solvers
- Target multi-core clusters: Nested distributed-/shared-memory parallelism and hybrid programming (MPI/ OpenMP)
- High-fidelity petascale BIOEM simulations
- Shed light to controversial scientific and engineering questions

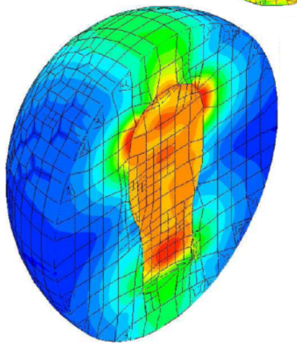
Samples

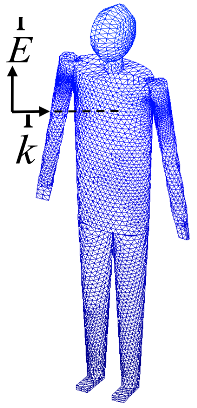


Mesh-based Geometry

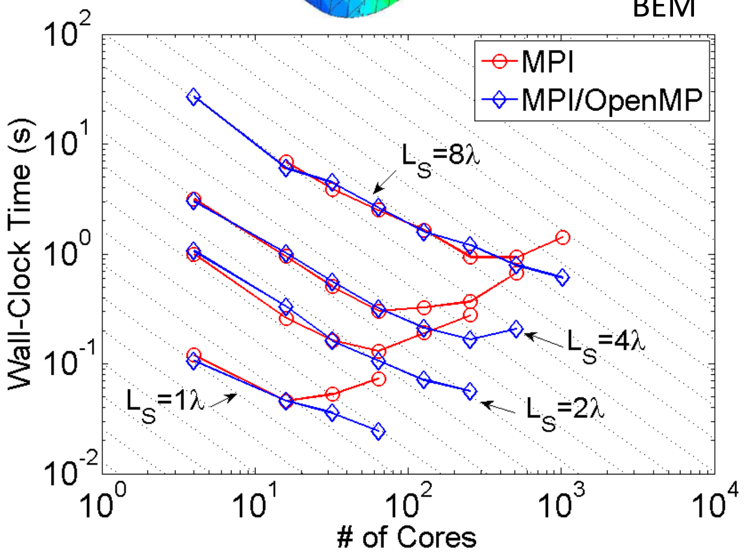
hp-adaptive FEM







FFT-accelerated BEM



c/o Leszek Demkowicz, UTexas

Hybrid programming models not enough

- Tools for monitoring the availability and predicted performance of resources within an architecture-adaptive and application-adaptive are improving
- However, even perfect knowledge of resource capabilities at every moment and perfect load balancers will not rescue billion-thread SPMD implementations of PDE simulations, etc.
  - cost of rebalancing frequently is too large
  - Amdahl penalty of failing to rebalance is fatal

Evolution of parallel programming models: 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

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]

Algorithmic adaptation to asynchronous programming styles

- Additive versions of algorithms are often available that significantly relax synchronicity
- Such algorithms have received a bad rap historically
  - “chaotic relaxation,” Chazan & Miranker, 1969, for instance
- However, they can sometimes be made virtually as good as their multiplicative cousins
  - “AFACx” versus “AFAC,” Lee, McCormick, Philip & Quinlan, 2003, for instance

Peta to exa for algorithms

- Things we need to do for exascale will help us at petascale and terascale
  - Reducing memory requirements and memory traffic
  - Exploiting hybrid and less synchronous parallel programming models
  - Co-design of hardware and software (for, e.g., power management)
- Though it inveighs against the CS aesthetic of “separation of concerns”, and involves more issues, co-design requires similar attitude and aptitude as in, say, MPI programming today
  - Applications programmers have “bit the bullet” and designed excellent MPI-based codes, by using quality libraries designed and ported by specialists
  - Hopefully, we will be able to isolate applications programmers from many of the hardware and software architectural details, just as we do today from message-passing details

Peta to exa

- Billion-way parallelism of GigaHertz cores will not significantly expand today's million-way flat parallelism at the node level
  - MPI legacy code will still be usable on the "outside" on a million nodes
  - Changes will be mainly *within* a node, where we will need to evolve thousand-way parallelism: "MPI+*X*"
- Principal challenges from peta to exa are within the node, and the burden is *shared by the marketplace* at all scales of node aggregation

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

Algorithmic Priority Research Directions (1)

- Advanced mathematical methods for scientific understanding in exascale simulations, including *in situ*
  - Uncertainty quantification, intrusive and nonintrusive
  - Optimization, inverse problems, sensitivity
  - Analysis and Visualization
  - Validation and Verification

Algorithmic Priority Research Directions (2)

- Exascale algorithms that expose and exploit multiple levels of parallelism
  - Communication-reducing algorithms
  - Synchronization-reducing algorithms
  - Fault resilient algorithms
- Support for multiphysics, multiscale methods
  - Break the SPMD and BSP paradigms when joining multiple different codes
  - Stability of coupling

Algorithmic Priority Research Directions (3)

- Exascale algorithms for constructing and adapting discrete objects
  - Algorithms that deal with unpredictable, dynamic workloads and have few flops to hide
- Mixed precision arithmetic, to use the lowest precision required to achieve a given accuracy outcome
  - Improves runtime, power consumption
  - Reformulate algorithms to find corrections, rather than solutions

Progressive by-product

- The consolation for the architecture-induced hard work of reducing synchrony is that algorithms have been waiting for this freedom for a long time
  - freedom to adapt the mesh or vary the timestep locally
  - freedom to vary the physical models at the cells or particle level
  - Freedom to vary the precision
- Once the synchronization is thrown on the programming model and runtime system, developers are less constrained

Summary:
High reward R&D themes for algorithms

- Mixed precision arithmetic, to use the lowest precision required to achieve a given accuracy outcome
  - Improves runtime, power consumption
  - Reformulate algs to find corrections, rather than solutions
- Prioritization of critical path and noncritical tasks
  - DAG scheduling of critical path tasks
  - Allows taking advantage of asynchronicity between major steps and adaptive load balancing for noncritical tasks
- Communication-reducing algorithms
- And, of course, better mathematical formulations

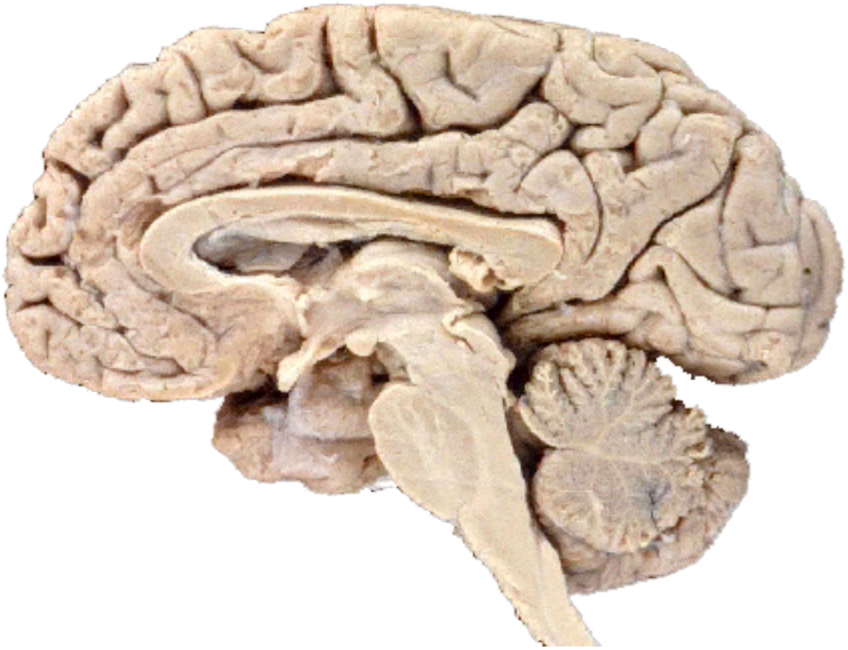
Recapitulation

- Reflected briefly on progress in high-end scientific computing
- Peeked briefly at some motivating applications
- Looked generically at PDE-based simulation and the basis of continued optimism for its growth – capability-wise
- Looked at some specific hurdles to and opportunities for PDE-based simulation posed by high-end architecture

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

(volume 1.5 liters, weight 3 lbs, consumes 20 W)

Zetta-scale, anyone?