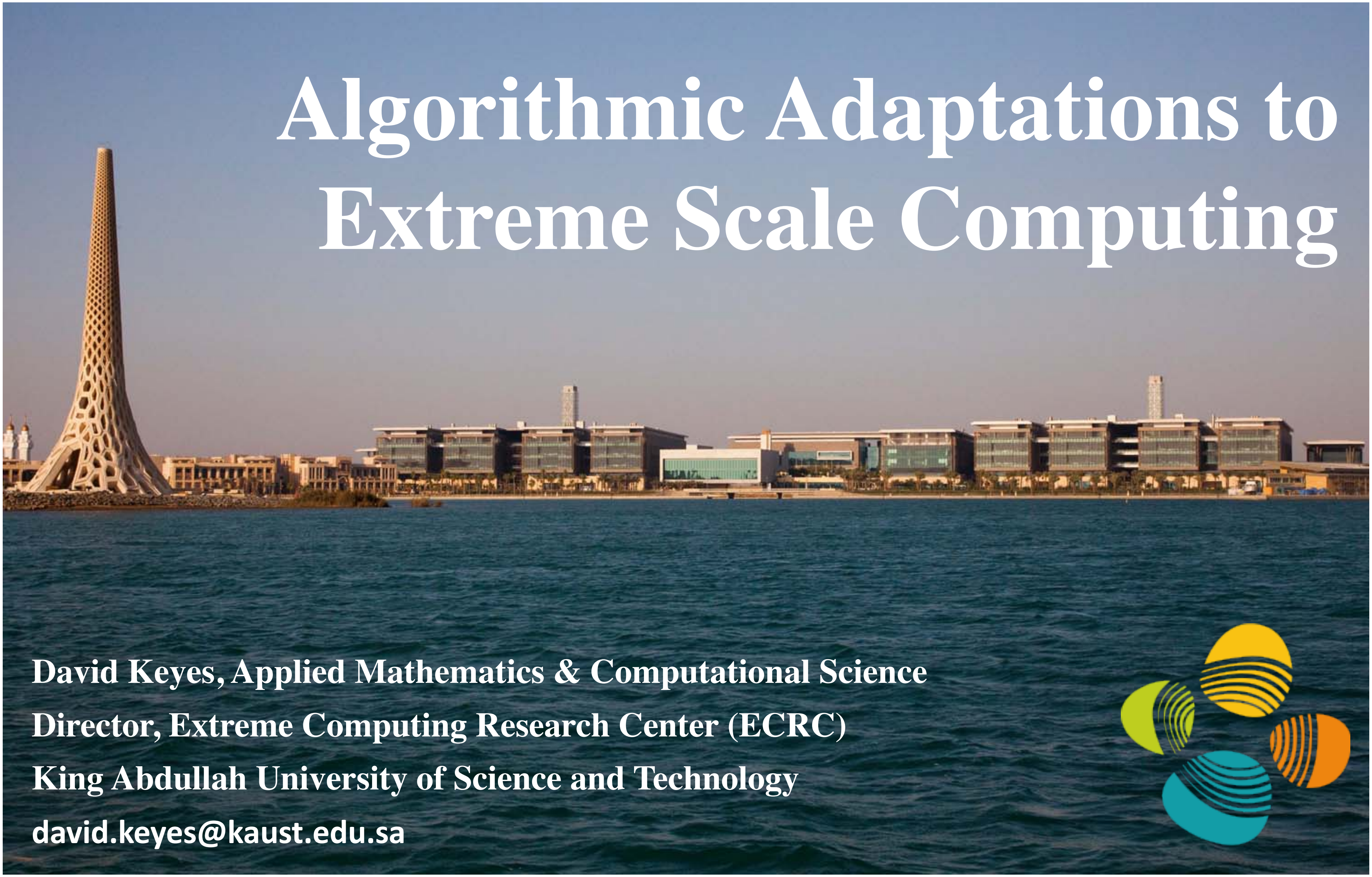
Future CFD Technologies Workshop

Bridging Mathematics and Computer Science for Advanced Simulation Tools



Algorithmic Adaptations to Extreme Scale Computing

David Keyes, Applied Mathematics & Computational Science

Director, Extreme Computing Research Center (ECRC)

King Abdullah University of Science and Technology

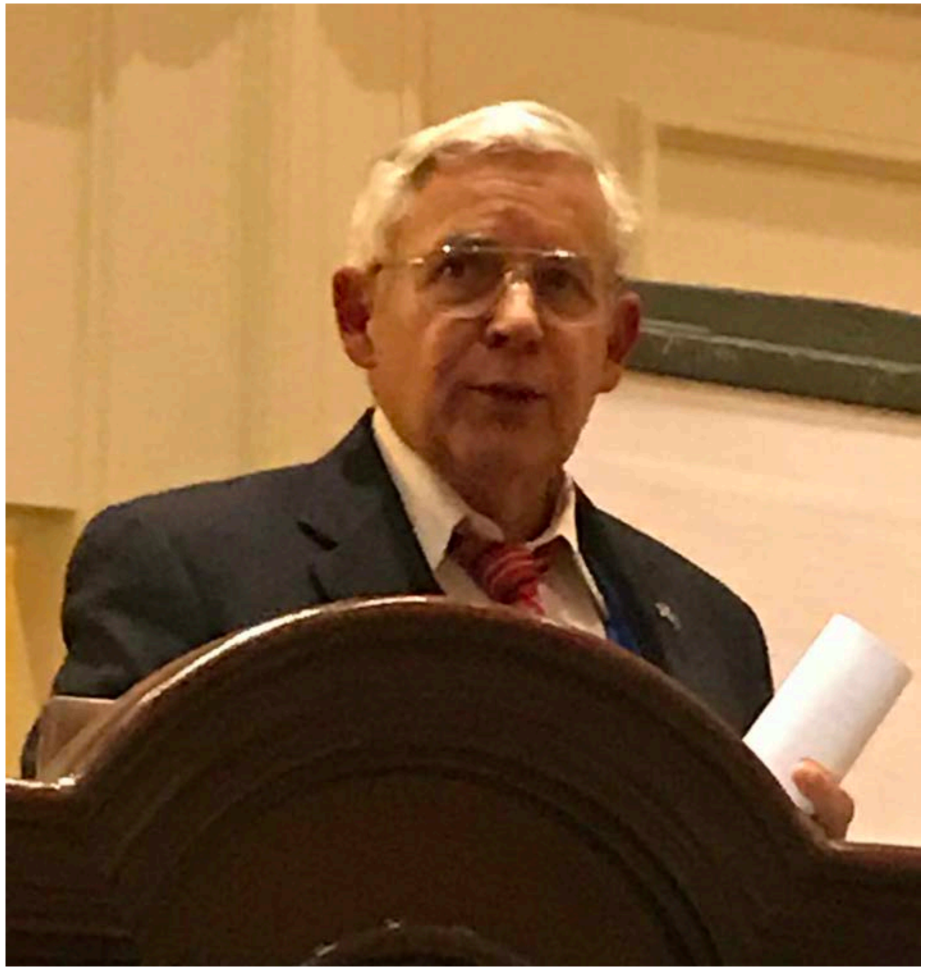
david.keyes@kaust.edu.sa

Offered in the spirit of

ICASE

1972-2002

“Salas” in Spanish means “rooms”



As a NASA Branch Head and then ICASE Director, Manny provided “room” for innumerable young scientists to grow in NASA mission-minded ways

1999 Gordon Bell Prize

Achieving High Sustained Performance in an Unstructured Mesh CFD Application

http://www.mcs.anl.gov/petsc-fun3d

Kyle Anderson, NASA Langley Research Center

William Gropp, Argonne National Laboratory

Dinesh Kaushik, Old Dominion University & Argonne

David Keyes, Old Dominion University, LLNL & ICASE

Barry Smith, Argonne National Laboratory

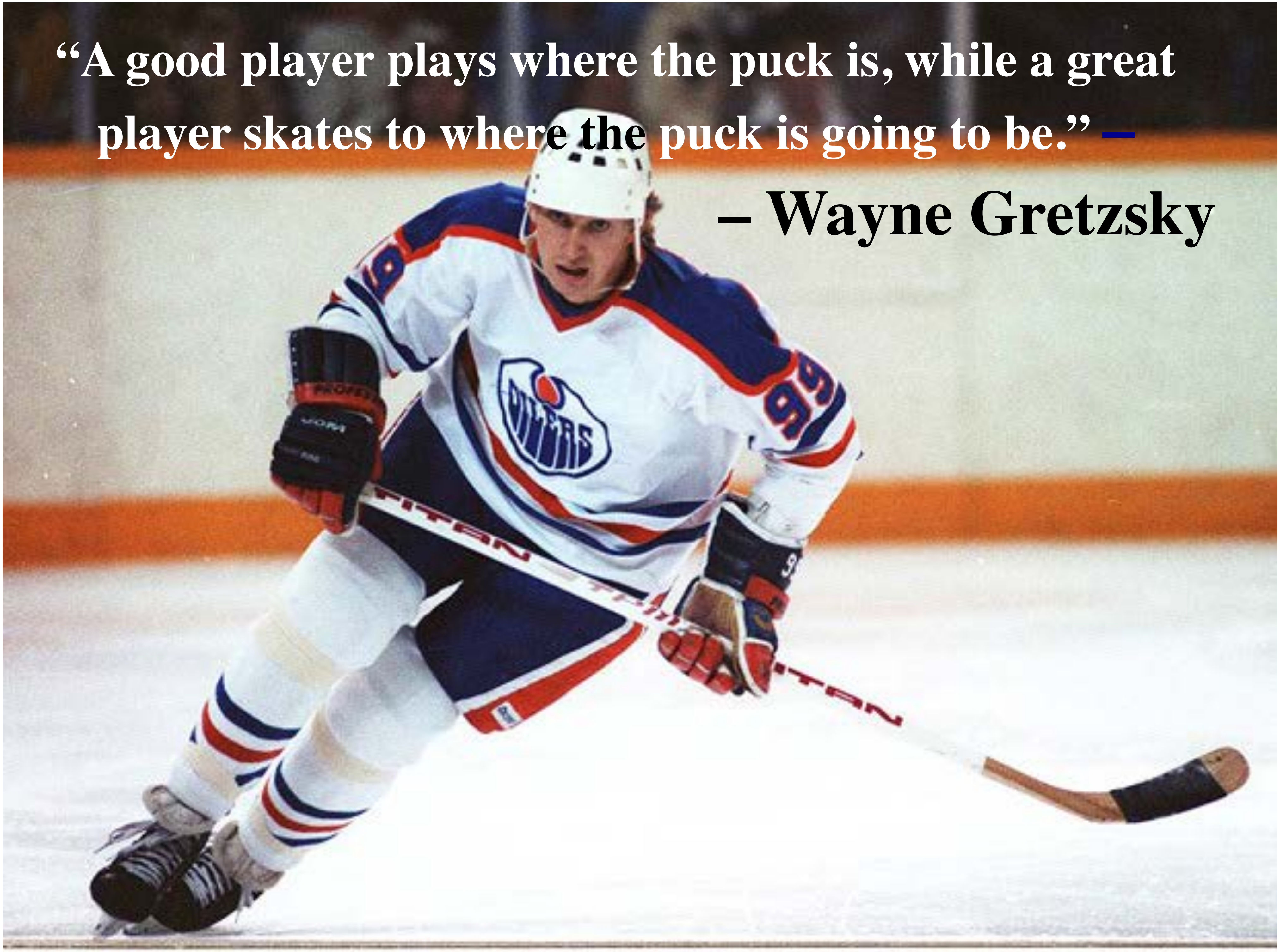
→ Abbreviated and updated version of the web-archived "Argonne Training Program in Extreme Scale Computing" (ATPESC) plenary of 1 August 2017:

"Algorithmic Adaptations to Extreme Scale Computing"

at

https://extremecomputingtraining.anl.gov/sessions/presentation-algorithmic-adaptations-to-extreme-scale-computing/

(See longer slide deck here for examples)


“A good player plays where the puck is, while a great player skates to where the puck is going to be.” –
– Wayne Gretzsky

Aspiration for this talk

To paraphrase Gretzsky:

"Algorithms for where architectures are going to be"

Outline

- **Four architectural trends**
  - **limitations of our current software infrastructure for numerical simulation at exascale**
- **Four algorithmic imperatives**
  - **for extreme scale, tomorrow *and today***
- **Four sets of "bad news, good news"**
- **Four widely applicable strategies**

Four architectural trends

- **Clock rates cease to increase while arithmetic capability continues to increase through concurrency (flooding of cores)**
- **Memory storage capacity increases, but fails to keep up with arithmetic capability *per core***
- **Transmission capability – memory BW and network BW – increases, but fails to keep up with arithmetic capability *per core***
- **Mean time between hardware errors shortens**

➔ **Billions of**

$ ¥ € £

of scientific software worldwide hangs in the balance until our algorithmic infrastructure evolves to span the architecture-applications gap

Architectural background

www.exascale.org/iesp

INTERNATIONAL EXASCALE SOFTWARE PROJECT ROADMAP 1.0

10^{18}

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**:3-60, 2011.

Uptake from IESP meetings

- **While obtaining the next order of magnitude of performance, we need another order of performance *efficiency***
  - **target: 50 Gigaflop/s/W, today typically ~ 5 Gigaflop/s/W**
- **Required reduction in power per flop and per byte may make computing and moving data less reliable**
  - **smaller circuit elements will be subject to more noise per signal, with less redundancy for hardware resilience**
  - **more errors may need to be caught and corrected in software**
- **Processor clock rates may vary during a run**
  - **makes per-node performance rate unreliable**

Today's power costs per operation

Operation	approximate energy cost
DP floating point multiply-add	100 pJ
DP DRAM read-to-register	4800 pJ
DP word transmit-to-neighbor	7500 pJ
DP word transmit-across-system	9000 pJ

2 orders of magnitude energy cost (worse ratio for latency)

A *pico* (10^{-12}) of something done *exa* (10^{18}) times per second is a *mega* (10^{6})-somethings per second

- 100 pJ at 1 Eflop/s is 100 MW (for the flop/s only!)
- 1 MW-year costs about \$1M (\$0.12/KW-hr × 8760 hr/yr)
    - We "use" 1.4 KW continuously, so 100MW is 71,000 people

c/o J. Shalf (LBNL)

Why exa- is different

Dennard's MOSFET scaling (1972) ends before Moore's Law (1965) ends

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

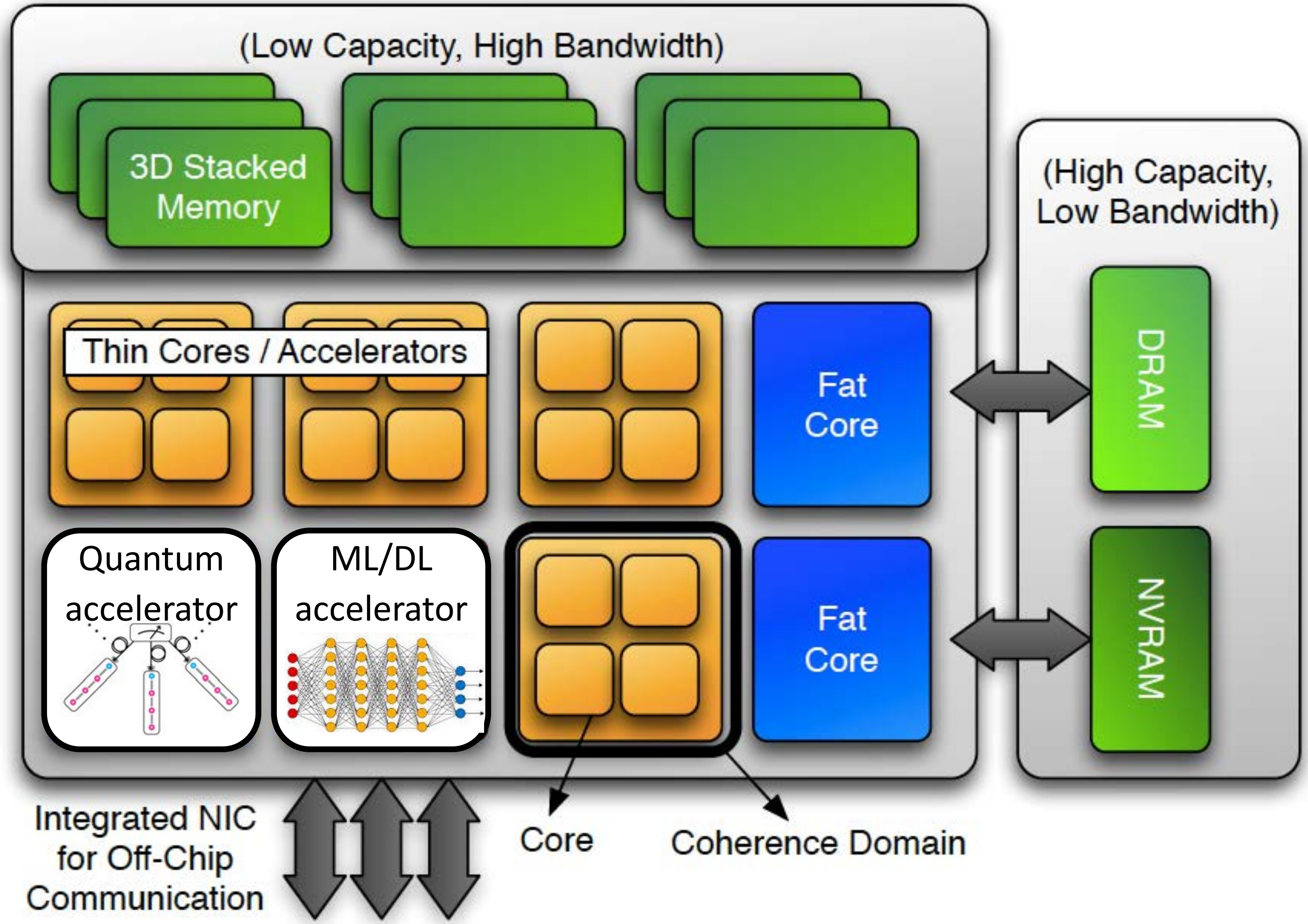
Table 2
Scaling Results for Interconnection Lines

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

Robert Dennard, IBM
(inventor of DRAM, 1966)

Eventually processing is limited by transmission, as known for 4.5 decades

Heterogeneity: fifth architectural trend



c/o J. Ang et al. (2014), *Abstract Machine Models and Proxy Architectures for Exascale Computing*

Seek balance of architectural resources

- **Processing cores**
  - **heterogeneous (CPUs, MICs, GPUs, FPGAs,…)**
- **Memory**
  - **hierarchical (registers, caches, DRAM, flash, stacked, …)**
  - **partially reconfigurable**
- **Intra-node network**
  - **nonuniform bandwidth and latency**
- **Inter-node network**
  - **nonuniform bandwidth and latency**

For performance tuning:

Which resource is limiting, as a function of time?

Well established resource trade-offs

- **Communication-avoiding algorithms**
  - **exploit extra memory to achieve theoretical lower bound on communication volume**
- **Synchronization-avoiding algorithms**
  - **perform extra flops between global reductions or exchanges to require fewer global operations**
- **High-order discretizations**
  - **perform more flops per degree of freedom (DOF) to store and manipulate fewer DOFs**

Node-based "weak scaling" is routine; thread-based "strong scaling" is the game

- **An exascale configuration: 1 million 1000-way 1GHz nodes**
- **Expanding the number of nodes (processor-memory units) beyond 10^6 would *not* be a serious threat to algorithms that lend themselves to well-amortized precise load balancing**
  - **provided that the nodes are performance reliable for load balancing**

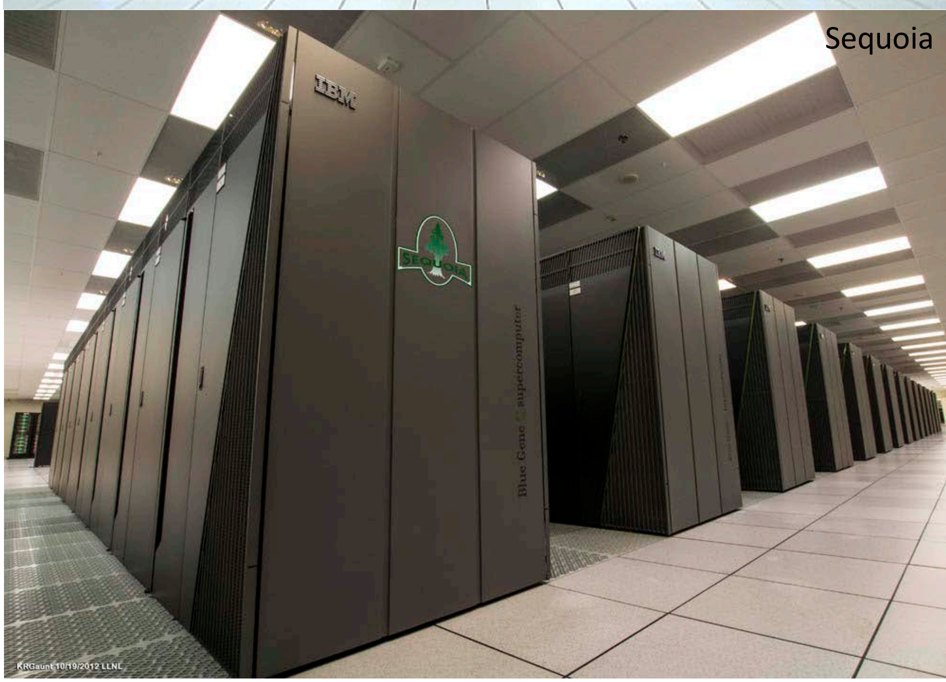
Node-based "weak scaling" is routine; thread-based "strong scaling" is the game

- **Real challenge is usefully expanding the number of cores sharing memory 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)**
  - **don't need to wait for full exascale systems to experiment in this regime – the contest is being waged on individual shared-memory nodes today**

The familiar

Taihu Light

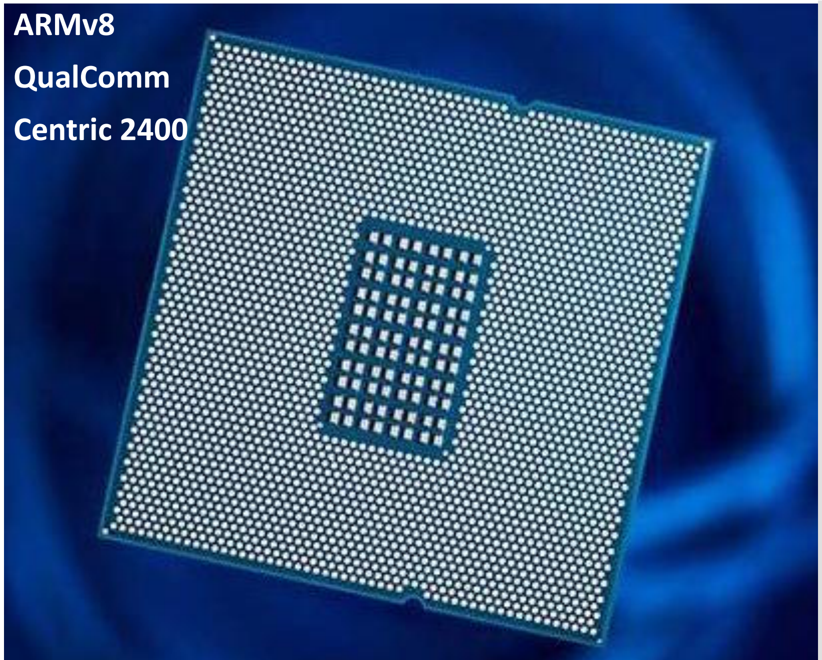
Shaheen

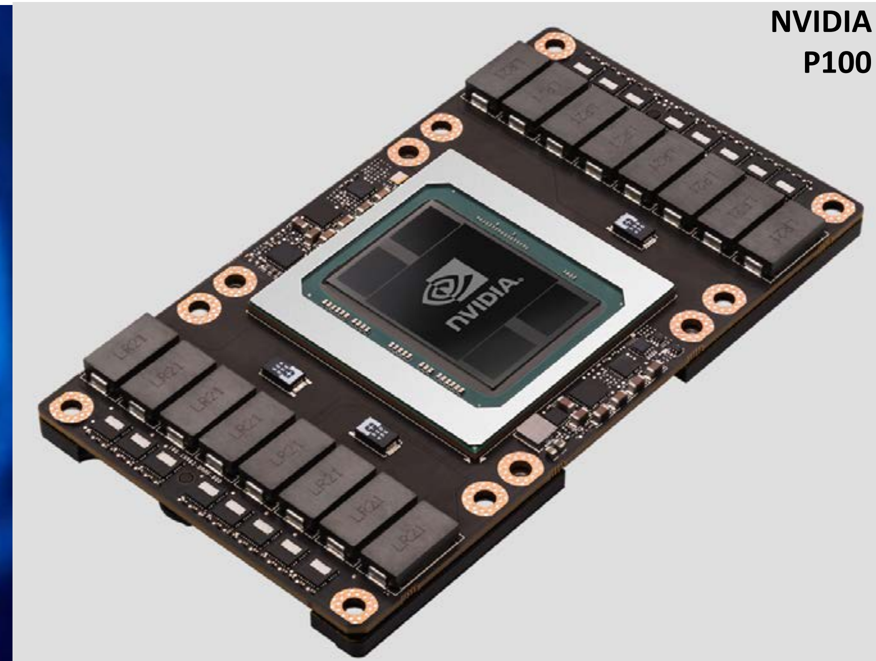


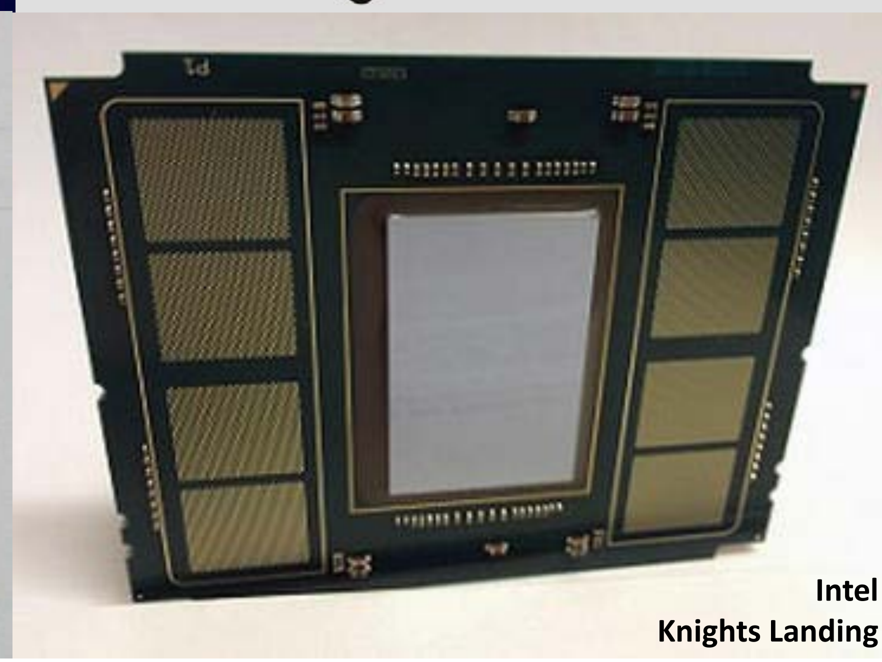
Sequoia

K

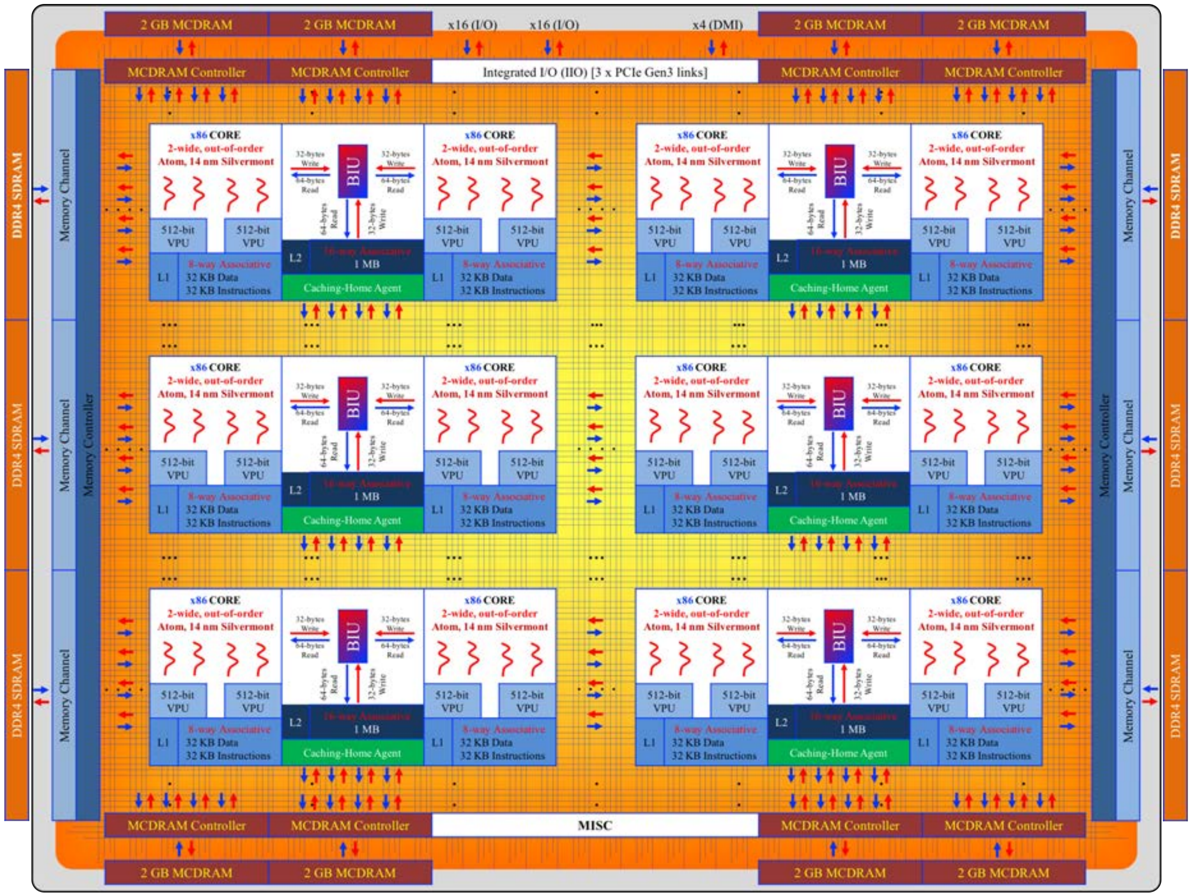
The challenge







Don’t need to wait for full exascale systems to experiment in this regime…



Schematic of Intel Xeon Phi KNL by M. Farhan, KAUST

The main contest is already being waged on individual shared-memory nodes

Just two decades of evolution

1997 | 2017

ASCI Red at Sandia

1.3 TF/s, 850 KW

Cavium ThunderX2

~ 1.1 TF/s, ~ 0.2 KW

3.5 orders of magnitude

Supercomputer in a node

System	Peak DP TFlop/s	Peak Power KW	Power Efficiency GFlop/s/Watt
ASCI Red	1.3	850	0.0015
ThunderX2 Cavium	1.1	0.20	5.5

Supercomputer in a node

System	Peak DP TFlop/s	Peak Power KW	Power Efficiency GFlop/s/Watt
ASCI Red	1.3	850	0.0015
ThunderX2 Cavium	1.1	0.20	5.5*
Knights Landing Intel	3.5	0.26	14
P100 Pascal NVIDIA	5.3	0.30	18
Taihu Light CAS	125,000	15,000	8.3
Exascale System (~2021)	1,000,000	20,000	50

* 8 memory channels in Cavium ARM vs. 6 for Intel KNL

How are most scientific simulations implemented at the petascale today?

- **Iterative methods based on data decomposition and message-passing**
  - data structures are distributed
  - each individual processor works on a subdomain of the original
  - exchanges information with other processors that own data 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

Three decades of stability in programming model

Bulk Synchronous Parallelism

Leslie Valiant, F.R.S., N.A.S.
2010 Turing Award Winner

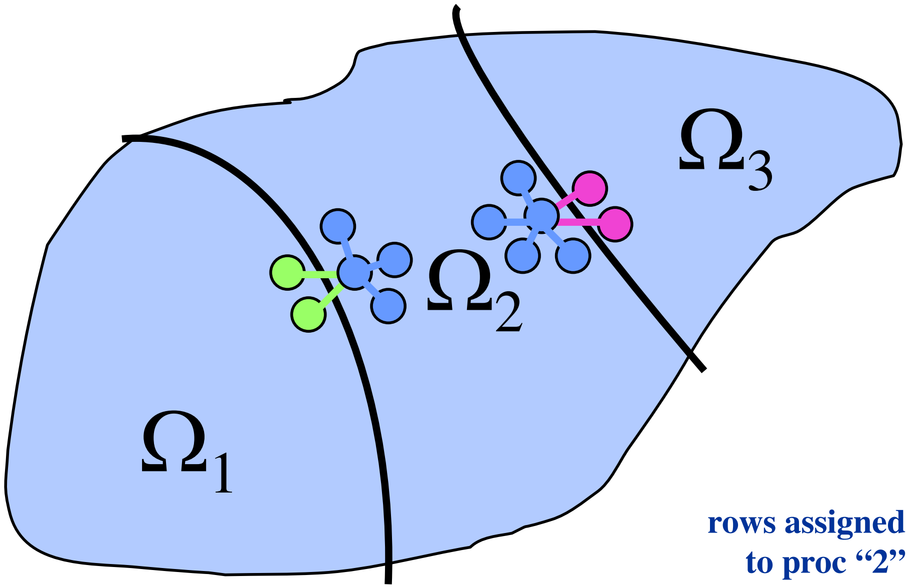
A Bridging Model for Parallel Computation

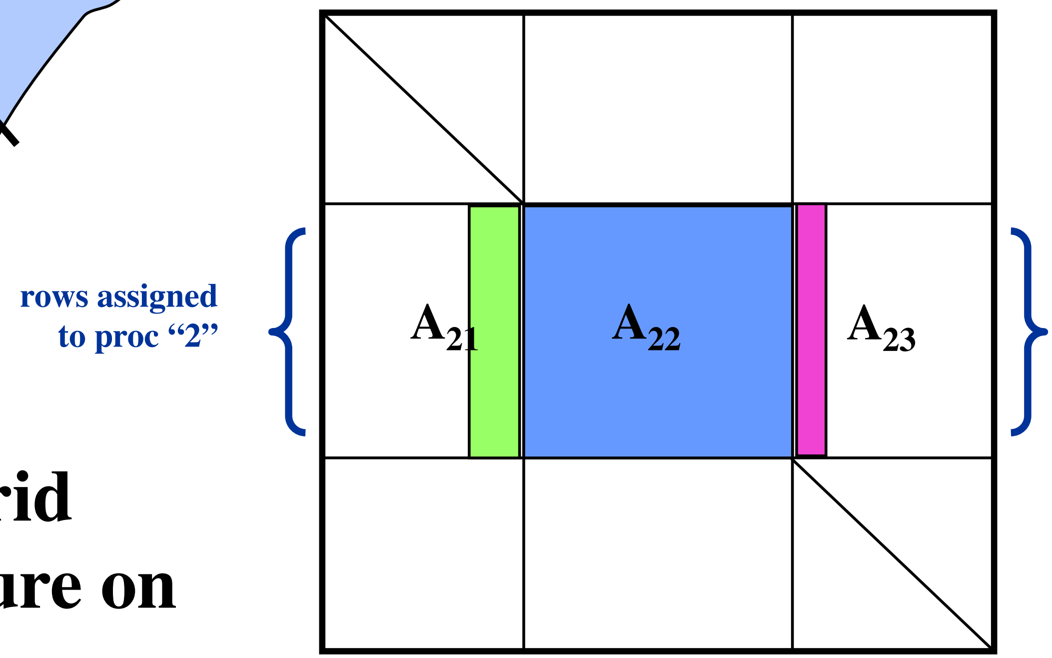
The success of the von Neumann model of sequential computation is attributable to the fact that it is an efficient bridge between software and hardware: high-level languages can be efficiently compiled on to this model; yet it can be efficiently implemented in hardware. The author argues that an analogous bridge between software and hardware is required for parallel computation if that is to become as widely used. This article introduces the bulk-synchronous parallel (BSP) model as a candidate for this role, and gives results quantifying its efficiency both in implementing high-level language features and algorithms, as well as in being implemented in hardware.

Leslie G. Valiant

Comm. of the ACM, 1990

BSP parallelism w/ domain decomposition





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

BSP has an impressive legacy

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

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

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

Riding exponentials

- **Proceeded steadily for decades from giga- (1988) to tera- (1998) to peta- (2008) with**
  - ***same*** **BSP programming model**
  - ***same*** **assumptions about who (hardware, systems software, applications software, etc.) is responsible for what (resilience, performance, processor mapping, etc.)**
  - ***same*** **classes of algorithms (*cf*. 25 yrs. of Gordon Bell Prizes)**
- **Scientific computing now at a crossroads with respect to extreme scale**

Extrapolating exponentials eventually fails

- **Exa- is qualitatively different and looks more difficult**
  - **but we once said that about message passing**
- **Core numerical analysis and scientific computing will confront exascale to maintain relevance**
  - **potentially big gains in colonizing exascale for science and engineering**
  - **not a "distraction," but an intellectual stimulus**
  - **the journey will be as fun as the destination ☺**

Main challenge going forward for BSP

- **Almost all "good" algorithms in linear algebra, differential equations, integral equations, signal analysis, etc., like to globally synchronize – and frequently!**
  - **inner products, norms, pivots, fresh residuals are "addictive" idioms**
  - **tends to hurt efficiency beyond 100,000 processors**
  - **can be fragile for smaller concurrency, as well, due to algorithmic load imbalance, hardware performance variation, etc.**
- **Concurrency is heading into the billions of cores**
  - **already 10 million on the most powerful system today**


Generation
Energy
Aware
Generation
Bulk
Synchronous

Four algorithmic imperatives

- **Reduce synchrony (in frequency and/or span)**
- **Reside "high" on the memory hierarchy**
  - **as close as possible to the processing elements**
- **Increase SIMT/SIMD-style shared-memory concurrency**
- **Build in resilience ("algorithm-based fault tolerance" or ABFT) to arithmetic/memory faults or lost/delayed messages**

Bad news/good news

- **Must explicitly control more of the data motion**
  - **carries the highest energy and time cost in the exascale computational environment**
- **More opportunities to control the *vertical* data motion**
  - ***horizontal* data motion under control of users already**
  - **but vertical replication into caches and registers was (until recently) mainly scheduled and laid out by hardware and runtime systems, mostly invisibly to users**

Bad news/good news

- **Use of uniform high precision in nodal bases on dense grids may decrease, to save storage and bandwidth**
  - **representation of a smooth function in a hierarchical basis or on sparse grids requires fewer bits than storing its nodal values, for equivalent accuracy**
- **We may compute and communicate "deltas" between states rather than the full state quantities**
  - **as when double precision was once expensive (e.g., iterative correction in linear algebra)**
  - **a generalized "combining network" node or a smart memory controller may remember the last address and the last value, and forward just the delta**
- **Equidistributing errors properly to minimize resource use will lead to innovative error analyses in numerical analysis**

Bad news/good news

- **Fully deterministic algorithms may be regarded as too synchronization-vulnerable**
  - **rather than wait for missing data, we may predict it using various means and continue**
  - **we do this with increasing success in problems without models (“big data”)**
  - **should be fruitful in problems coming from continuous models**
  - **“apply machine learning to the simulation machine”**
- **A rich numerical analysis of algorithms that make use of statistically inferred “missing” quantities may emerge**
  - **future sensitivity to poor predictions can often be estimated**
  - **numerical analysts will use statistics, signal processing, ML, etc.**

Bad news/good news

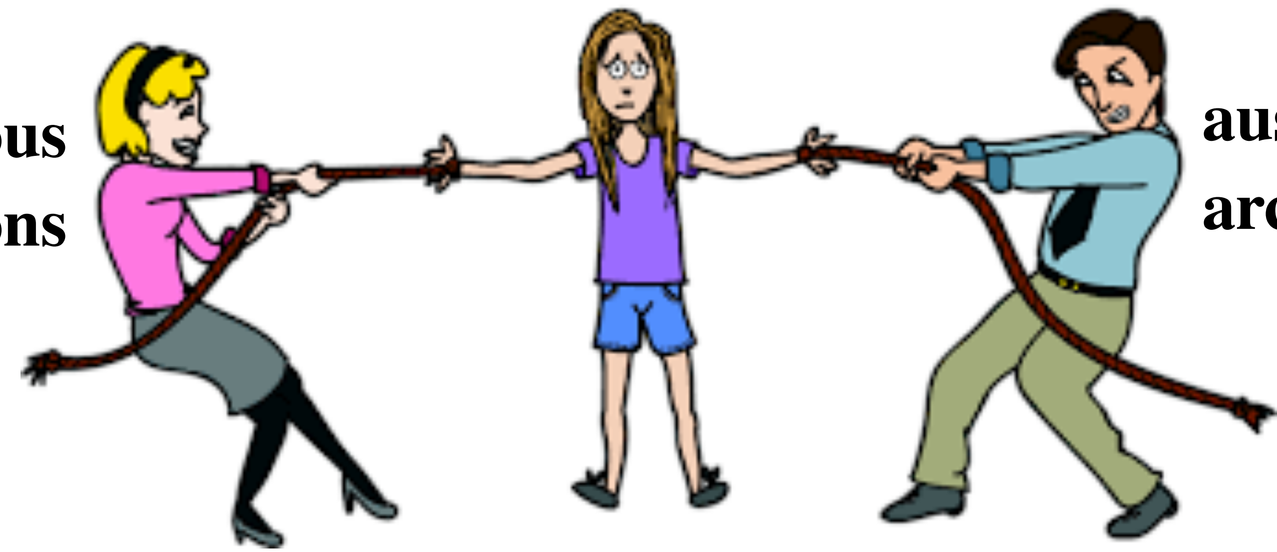
- **Fully hardware-reliable executions may be regarded as too costly**
- **Algorithmic-based fault tolerance (ABFT) will be cheaper than hardware and OS-mediated reliability**
  - **developers will partition their data and their program units into two sets**
    - **a small set that must be done reliably (with today's standards for memory checking and IEEE ECC)**
    - **a large set that can be done fast and unreliably, knowing the errors can be either detected, or their effects rigorously bounded**
- **Many examples in direct and iterative linear algebra**
- **Anticipated by Von Neumann, 1956 ("Synthesis of reliable organisms from unreliable components")**

Algorithmic philosophy

Algorithms must span a widening gulf …

adaptive
algorithms

ambitious
applications



austere
architectures

A full employment program for algorithm developers ☺

What will exascale algorithms look like?

- **For weak scaling, must *start* with algorithms with optimal asymptotic order, $O(N \log^p N)$**
- **Some optimal hierarchical algorithms**
  - **Fast Fourier Transform (1960's)**
  - **Multigrid (1970's)**
  - **Fast Multipole (1980's)**
  - **Sparse Grids (1990's)**
  - **$\mathcal{H}$ matrices (2000's)**
  - **Randomized algorithms (2010's)**

"With great computational power comes great algorithmic responsibility." – Longfei Gao, KAUST

Required software

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. Most is 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

Recap of algorithmic agenda

- **New formulations with**
  - **reduced synchronization and communication**
    - **less frequent *and/or* less global**
  - **reside high on the memory hierarchy**
    - **greater arithmetic intensity (flops per byte moved into and out of registers and upper cache)**
  - **greater SIMT/SIMD-style thread concurrency for accelerators**
  - **algorithmic resilience to various types of faults**
- **Quantification of trades between limited resources**
- ***Plus* all of the exciting "outer-loop" analytical agendas that exascale is meant to exploit**
  - **"post-forward" problems: optimization, data assimilation, parameter inversion, uncertainty quantification, etc.**

Four widely applicable strategies

- **Employ dynamic runtime systems based on directed acyclic task graphs (DAGs)**
  - e.g., ADLB, Argo, Charm++, HPX, Legion, OmpSs, Quark, STAPL, StarPU
- **Exploit data sparsity of hierarchical low-rank type**
  - meet the "curse of dimensionality" with the "blessing of low rank"
- **Employ high-order discretizations**
- **Code to the architecture, but present an abstract API**

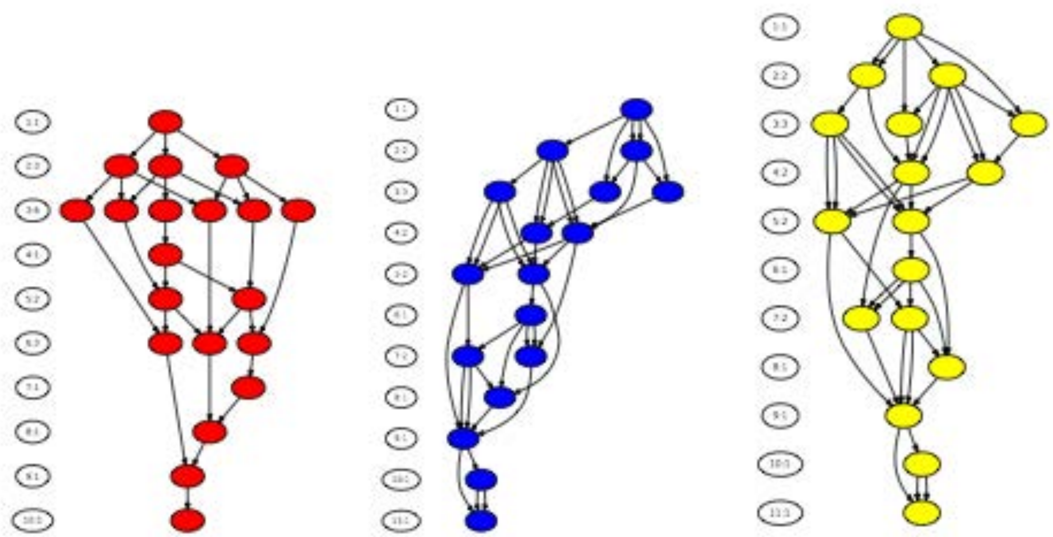
Taskification based on DAGs

- Advantages
  - remove artifactual synchronizations in the form of subroutine boundaries
  - remove artifactual orderings in the form of pre-scheduled loops
  - expose more concurrency
- Disadvantages
  - pay overhead of managing task graph
  - potentially lose some memory locality

Reducing over-ordering and synchronization through dataflow, ex.: generalized eigensolver

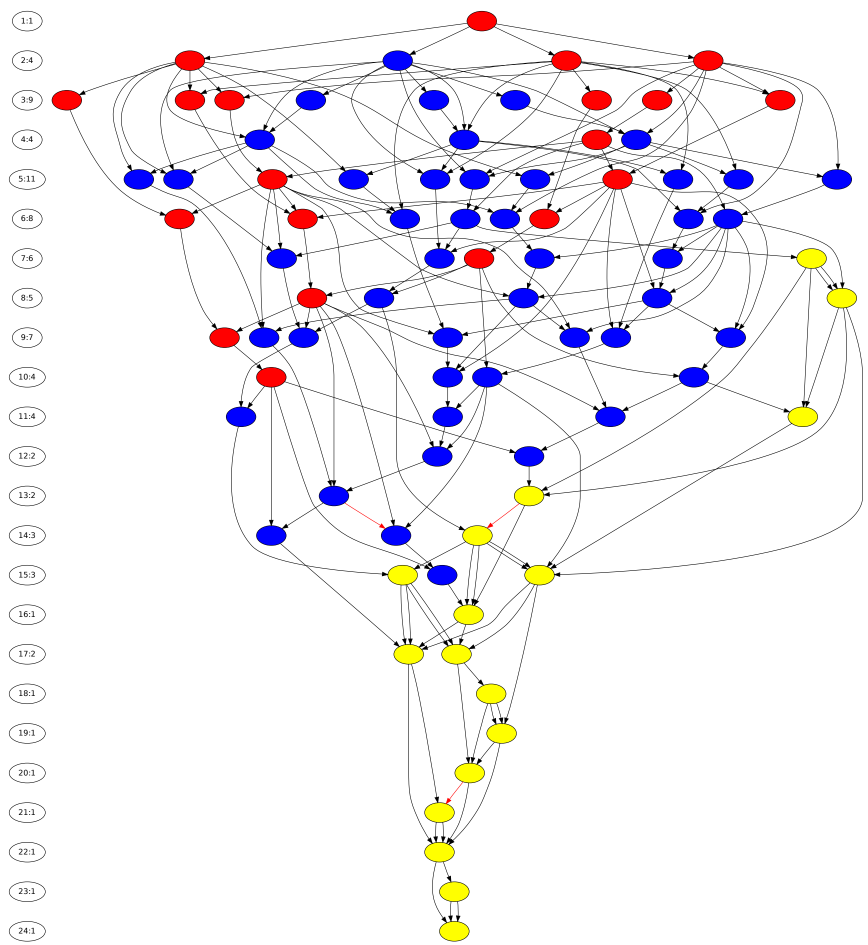
$$Ax = \lambda Bx$$

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



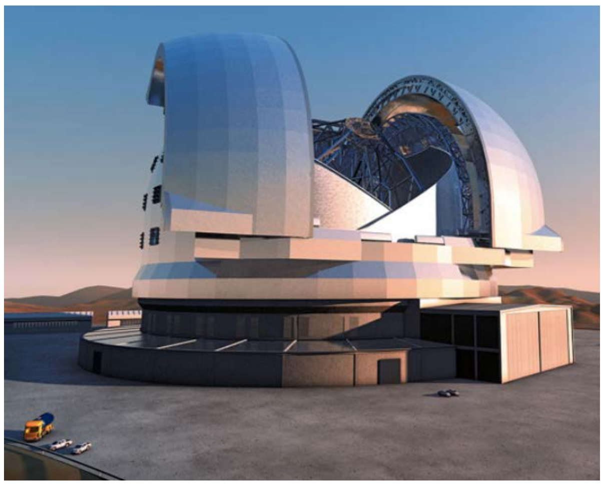
Loop nests and subroutine calls, with their over-orderings, can be replaced with DAGs

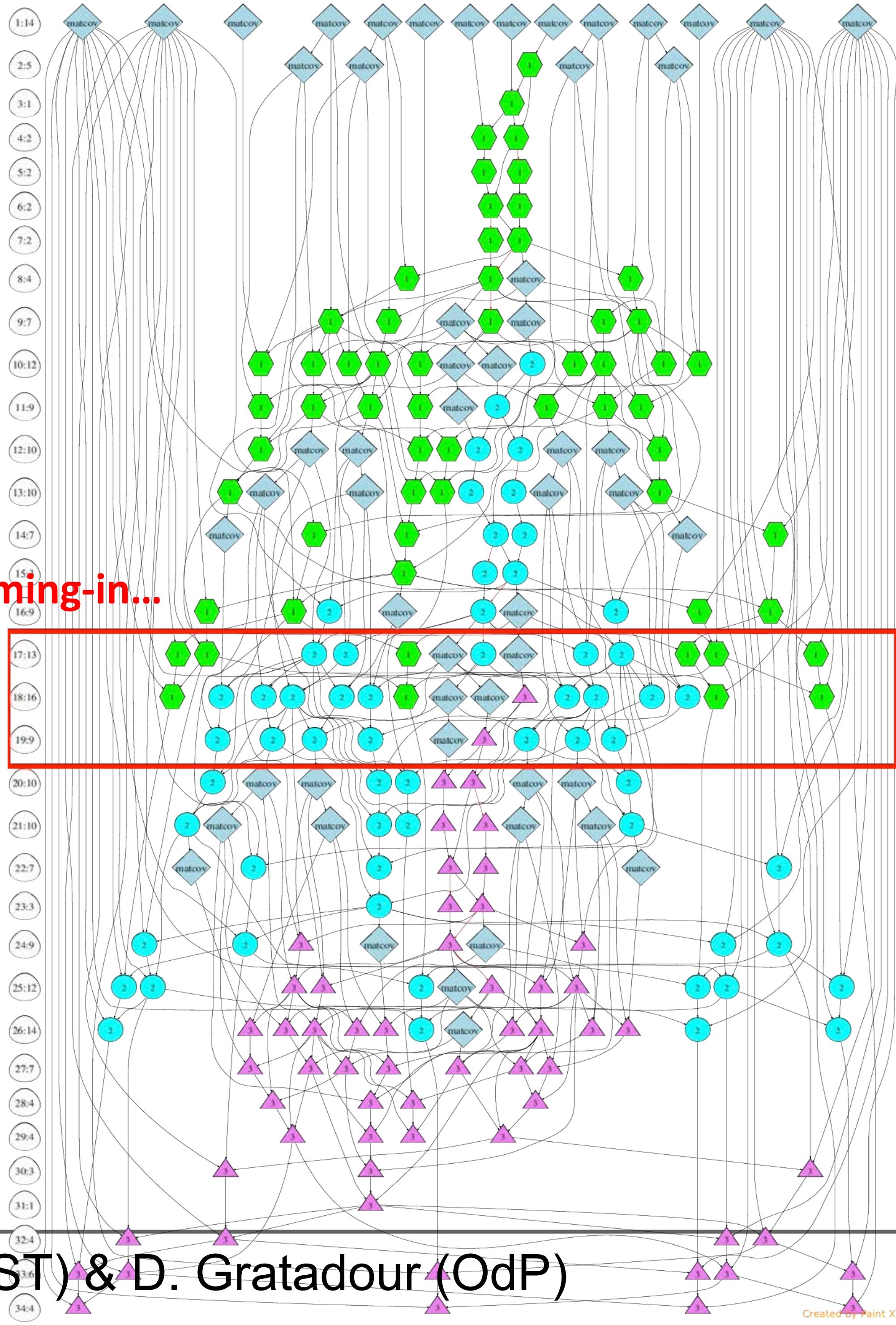
- **Diagram shows a dataflow ordering of the steps of a 4×4 symmetric generalized eigensolver**
- **Nodes are tasks, color-coded by type, and edges are data dependencies**
- **Time is vertically downward**
- **Wide is good; short is good**

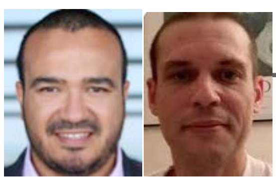


Loops can be overlapped in time

Green, blue and magenta symbols represent tasks in separate loop bodies with dependences from an adaptive optics computation



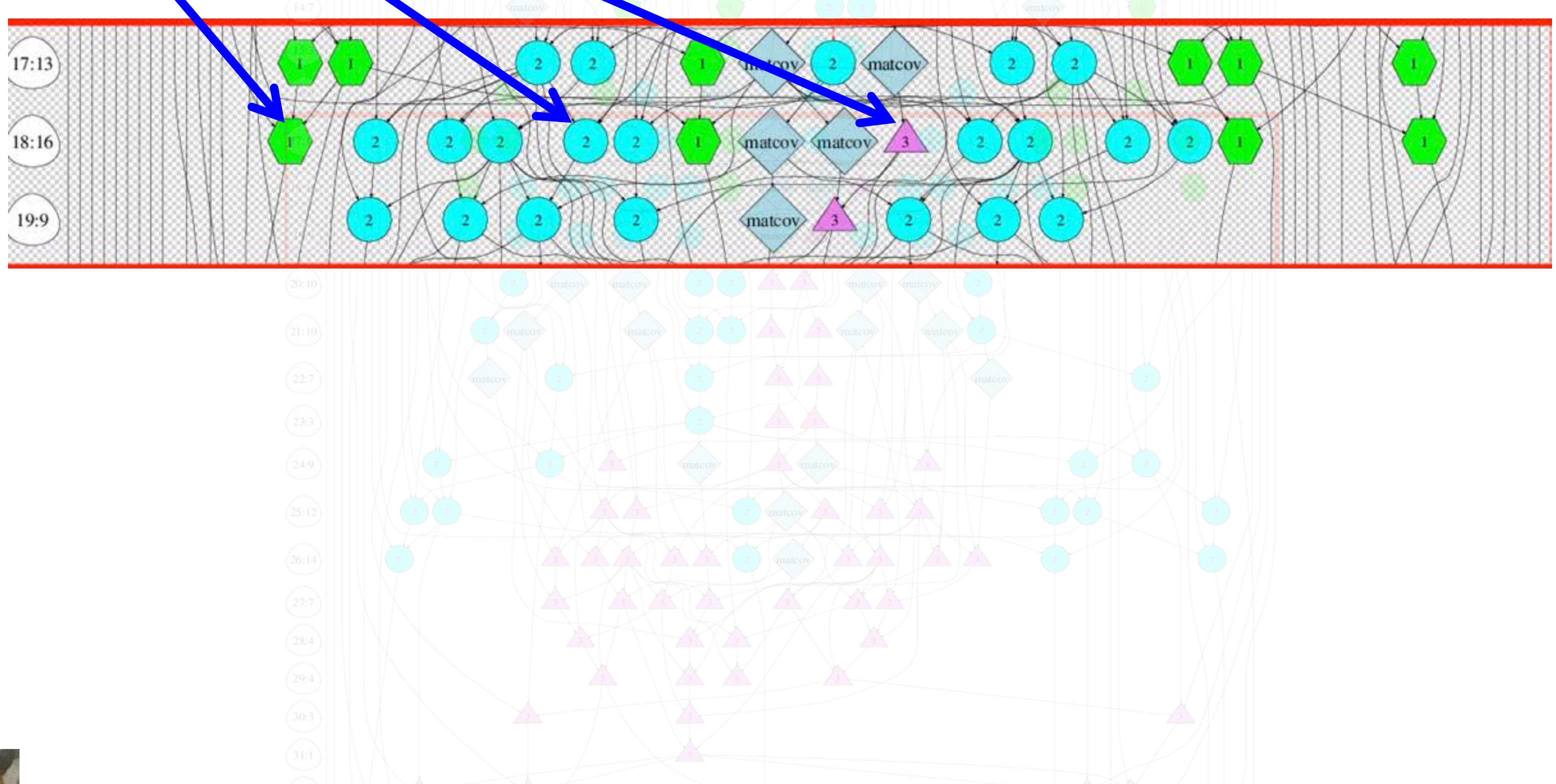


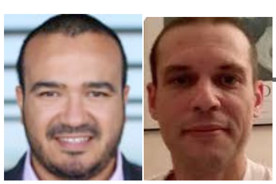


c/o H. Ltaief (KAUST) & D. Gratadour (OdP)

DAG-based safe out-of-order execution

Tasks from 3 loops of optical "reconstructor" pipeline are executed together





c/o H. Ltaief (KAUST) & D. Gratadour (OdP)

Hierarchically low-rank operators

- Advantages
  - shrink memory footprints to live higher on the memory hierarchy
    - higher means quick access
  - reduce operation counts
  - tune work to accuracy requirements
    - e.g., preconditioner versus solver
- Disadvantages
  - pay cost of compression
  - not all operators compress well

Key tool: hierarchical matrices

- [Hackbusch, 1999] **: off-diagonal blocks of typical differential and integral operators have low effective rank**
- **By exploiting low rank,** k **, memory requirements and operation counts approach optimal in matrix dimension** n**:**
  - **polynomial in** k
  - **lin-log in** n
  - **constants carry the day**
- **Such hierarchical representations navigate a compromise**
  - **fewer blocks of larger rank (“weak admissibility”) or**
  - **more blocks of smaller rank (“strong admissibility”)**

Example: 1D Laplacian

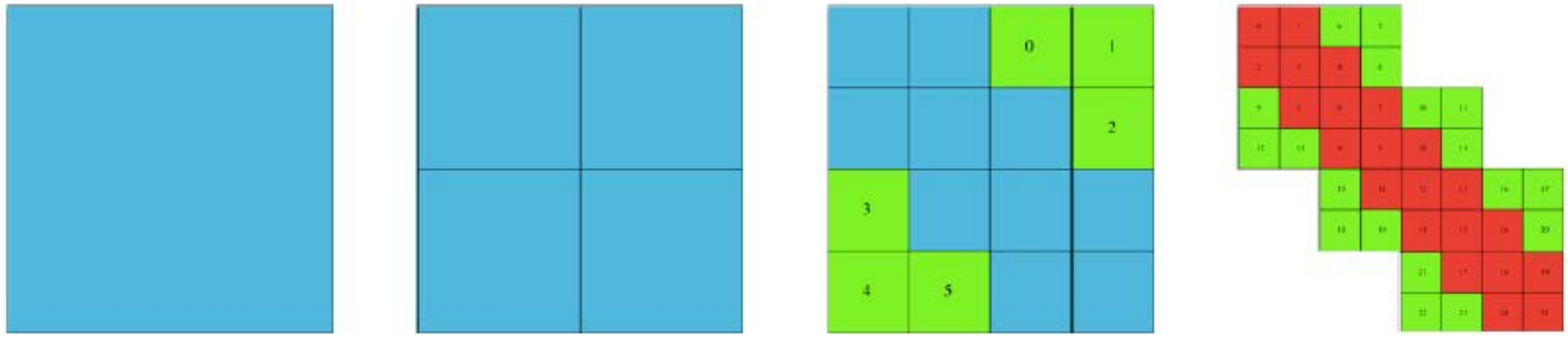
$$A = \left[\begin{array}{ccc|cccc} 2 & -1 & & & & & \\ -1 & 2 & -1 & & & & \\ & -1 & 2 & -1 & & & \\ \hline & & -1 & 2 & -1 & & \\ & & & -1 & 2 & -1 & \\ & & & & -1 & 2 & -1 \\ & & & & & -1 & 2 \end{array}\right]$$

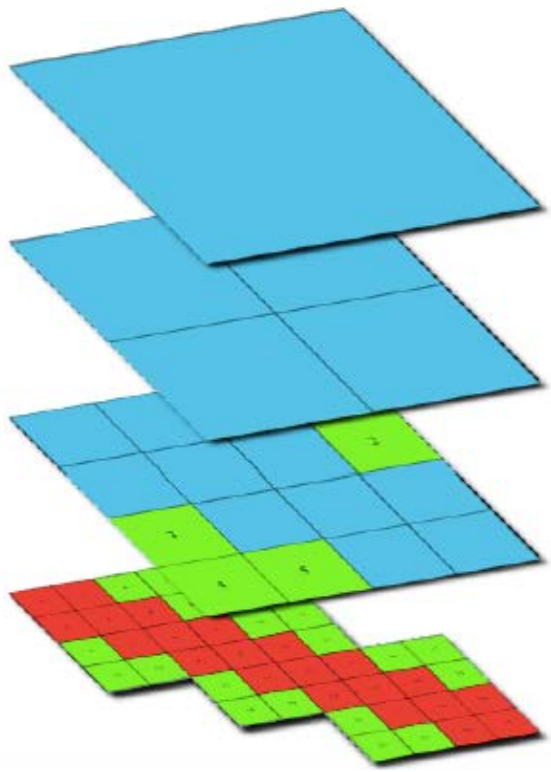
$$= \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 0 & 0 \end{bmatrix}$$

$$A^{-1} = \frac{1}{8} \times \left[\begin{array}{ccc|cccc} 7 & 6 & 5 & 4 & 3 & 2 & 1 \\ 6 & 12 & 10 & 8 & 6 & 4 & 2 \\ 5 & 10 & 15 & 12 & 9 & 6 & 3 \\ \hline 4 & 8 & 12 & 16 & 12 & 8 & 4 \\ 3 & 6 & 9 & 12 & 15 & 10 & 5 \\ 2 & 4 & 6 & 8 & 10 & 12 & 6 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array}\right]$$

$$= \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \begin{bmatrix} 4 & 3 & 2 & 1 \end{bmatrix}$$

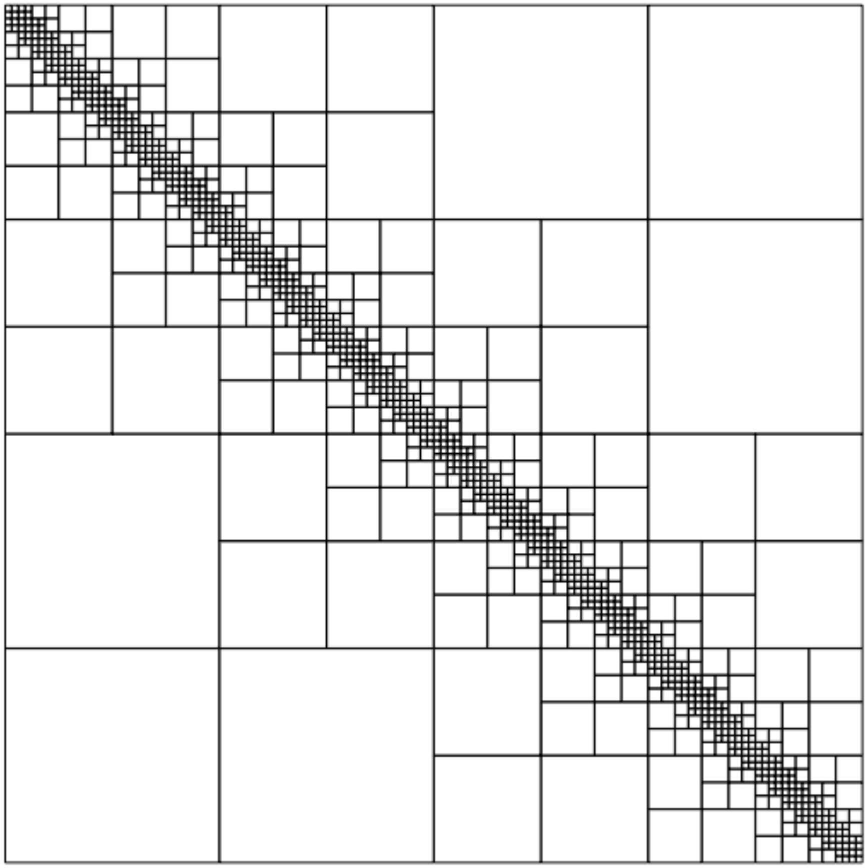
Recursive construction of an *H*-matrix



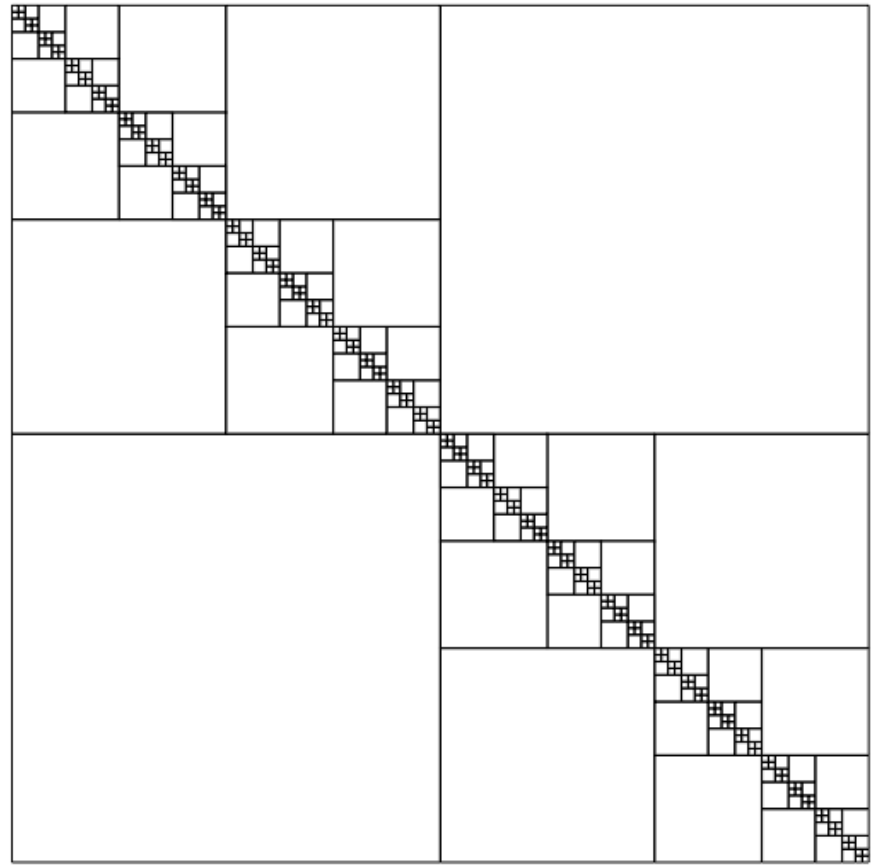


c/o W. Boukaram & G. Turkiyyah (KAUST)

“Standard (strong)” vs. “weak” admissibility



strong admissibility



weak admissibility

After Hackbusch, et al., 2003

Employ high-order discretizations

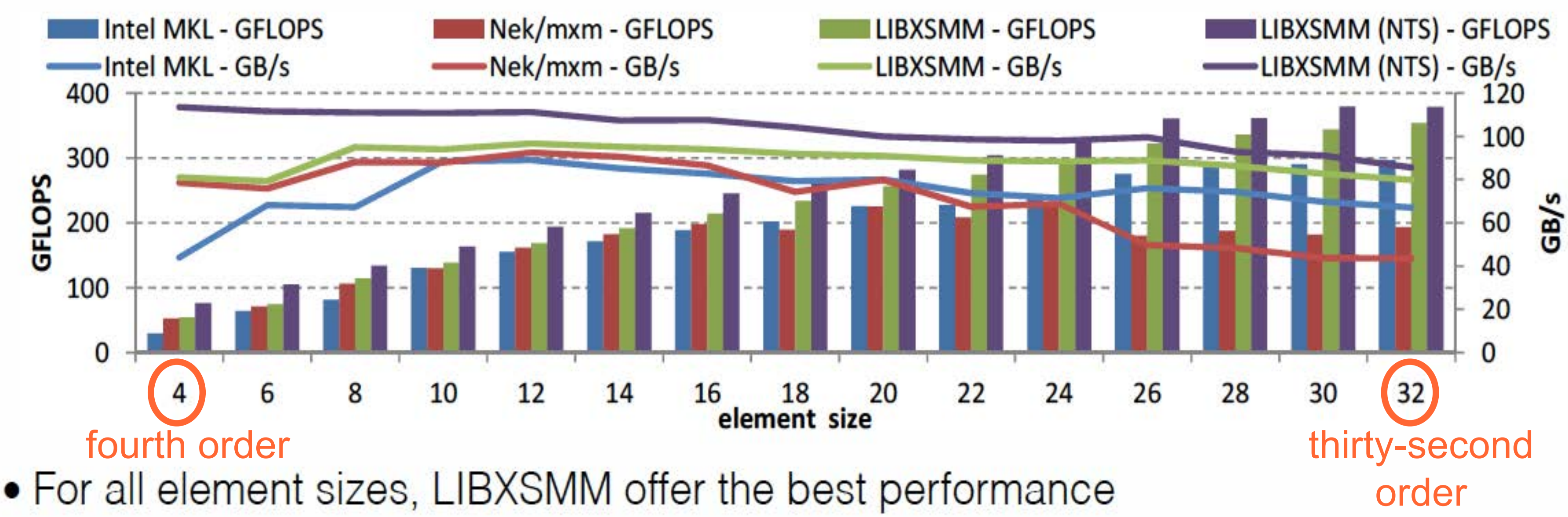
- **Advantages**
  - **shrink memory footprints to live higher on the memory hierarchy**
    - **higher means shorter latency**
  - **increase arithmetic intensity**
  - **reduce operation counts**
- **Disadvantages**
  - **high-order operators less suited to some solvers**
    - **e.g., algebraic multigrid, *H*-matrices***

---

* but see Gatto & Hesthaven, Dec 2016, on *H* for *hp* FEM

Performance effects of order in CFD

Helmholtz solve in spectral element code for incompressible Navier-Stokes

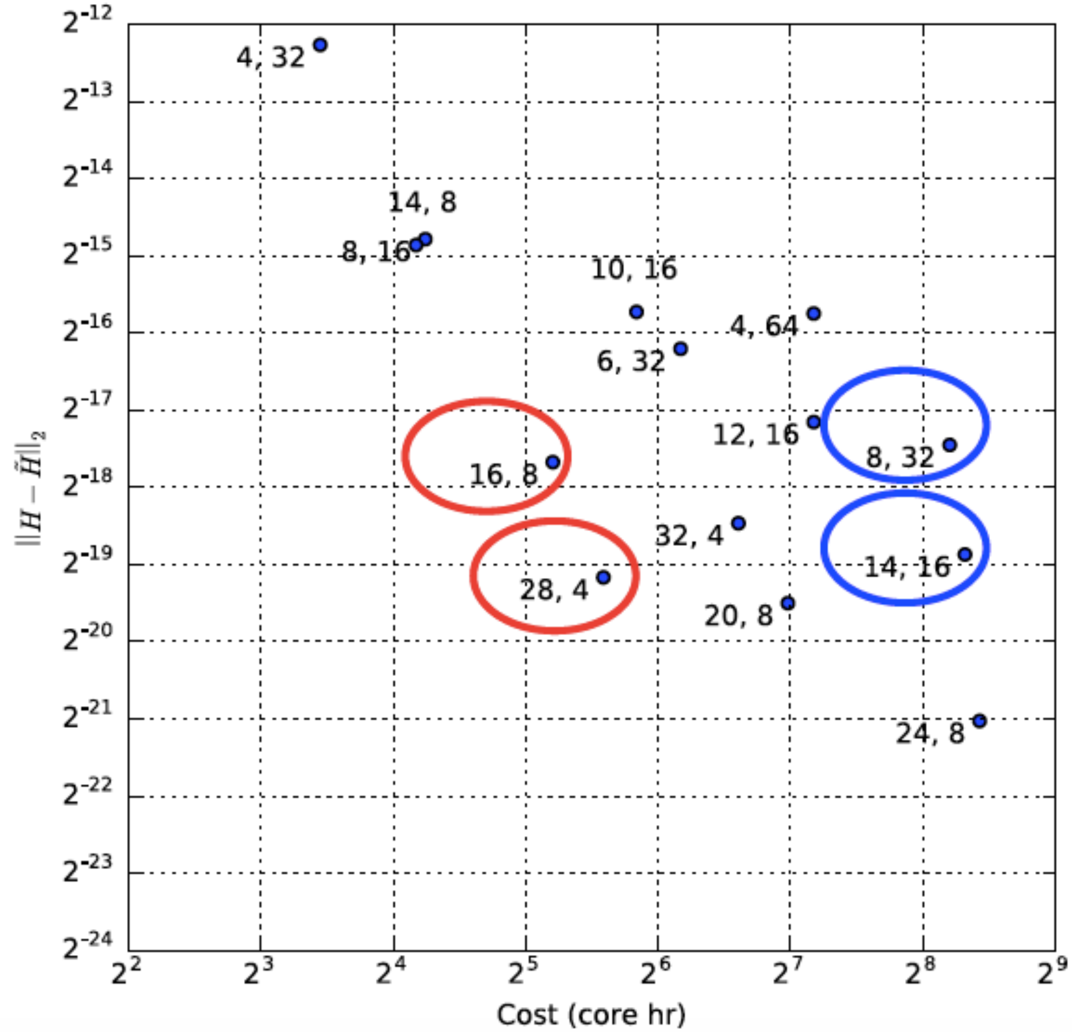


- For all element sizes, LIBXSMM offer the best performance
  - for order <= 16, the difference is small because the computation are memory bandwidth bound
  - for for order <= 16, a boost is possible with the non-temporal stores (101.6 GiB/s)
  - for order > 16, LIBXSMM ~ 2x is faster then Nek's `mxm_std` and up to 40% faster than Intel MKL

c/o Hutchinson *et al.* (2016) ISC'16

Runtime effects of order in CFD

Accuracy versus execution time as a function of order
Single-mode Rayleigh-Taylor instability

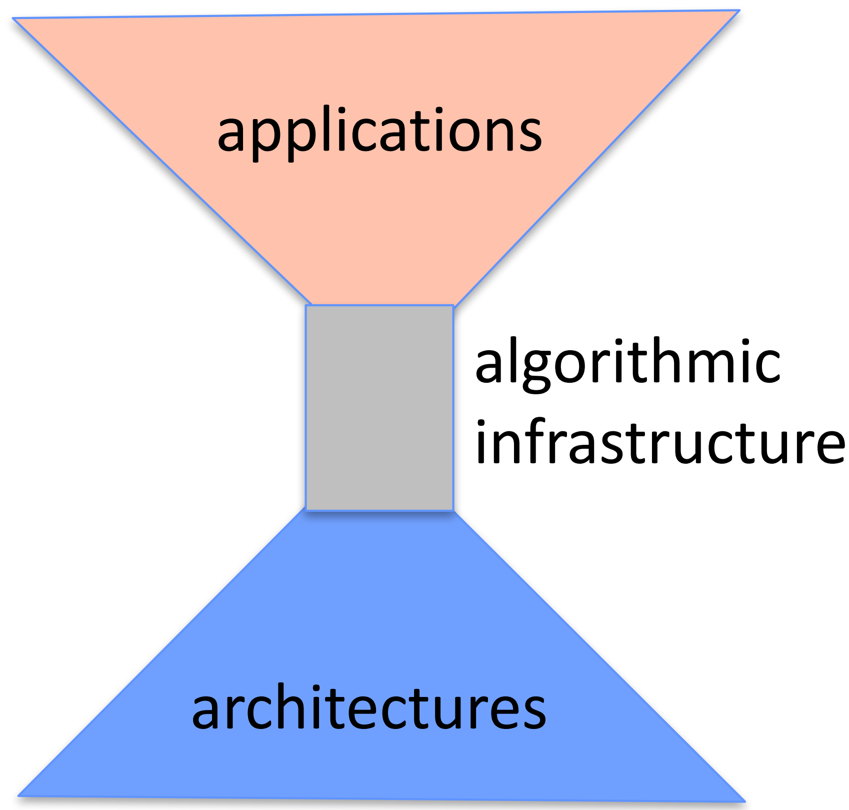


c/o Hutchinson *et al.* (2016) ISC'16

Code to the architecture

- Advantages
  - tiling and recursive subdivision create large numbers of small problems suitable for batched operations on GPUs and MICs
    - reduce call overheads
    - polyalgorithmic approach based on block size
  - non-temporal stores, coalesced memory accesses, double-buffering, etc. reduce sensitivity to memory
- Disadvantages
  - code is more complex
  - code is architecture-specific at the bottom

"Hourglass" model for algorithms
(traditionally applied to internet protocols)



Mapping algorithms to drivers

PhD thesis topics in the Extreme Computing Research Center at KAUST must address at least one of the four algorithmic drivers

Student	Algorithm/Kernel	Reduce Synchronization	Increase Intensity	Increase Concurrency	Algorithmic Resilience	New Capabilities
Abdelfattah	BLAS2		X	X		
Abduljabbar	FMM	X	X	X		
AlFarhan	Unstruct. PDEs	X		X		
AlHarthi	BEM	X	X	X		
AlOnazi	Multigrid	X		X	X	
Boukaram	H-BLAS		X	X		
Charara	BLAS2/3		X	X		
Chavez	H-Schur		X	X		
Ibeid	FMM precond.	X	X	X		
Liu	Nonlinear precond.	X			X	X
Malas	Stencil eval.		X	X		
Peng	Non-neg. mat. fact.			X		X
Sukkari	Eigen/SVD		X	X		

Student placement, recent PhD graduates

Huda Ibeid
U Illinois UC/ DOE XPACC
US DOE

Gustavo Chavez
Lawrence Berkeley National Lab/ UC Berkeley
US DOE

Ali Charara
(offers at NVIDIA and Oak Ridge National Lab/ U Tennessee)
US DOE

Mustafa Abduljabbar
(offer at Oak Ridge National Lab/ U Tennessee)
US DOE

Student placement, recent PhD graduates

Ahmad Abdelfattah
Oak Ridge National Lab/ U Tennessee

US DOE

Tareq Malas
Lawrence Berkeley National Lab/ UC Berkeley

Now at Intel

Lulu Liu
Swiss National Supercomputer Center/ U Lugano

Chengbin Peng
Chinese Acad of Sciences/ Ningbo


The *other* baton pass
Paradigms Converged
3rd & 4th Paradigms Separate

"Convergence" background

www.exascale.org/bdec

10^{18} BDEC
BIG DATA AND
EXTREME-SCALE
COMPUTING

The BDEC "Pathways to Convergence" Report

Toward a Shaping Strategy for a Future
Software and Data Ecosystem for Scientific Inquiry

Edited by: Mark Asch and Terry Moore

Contributing Authors:

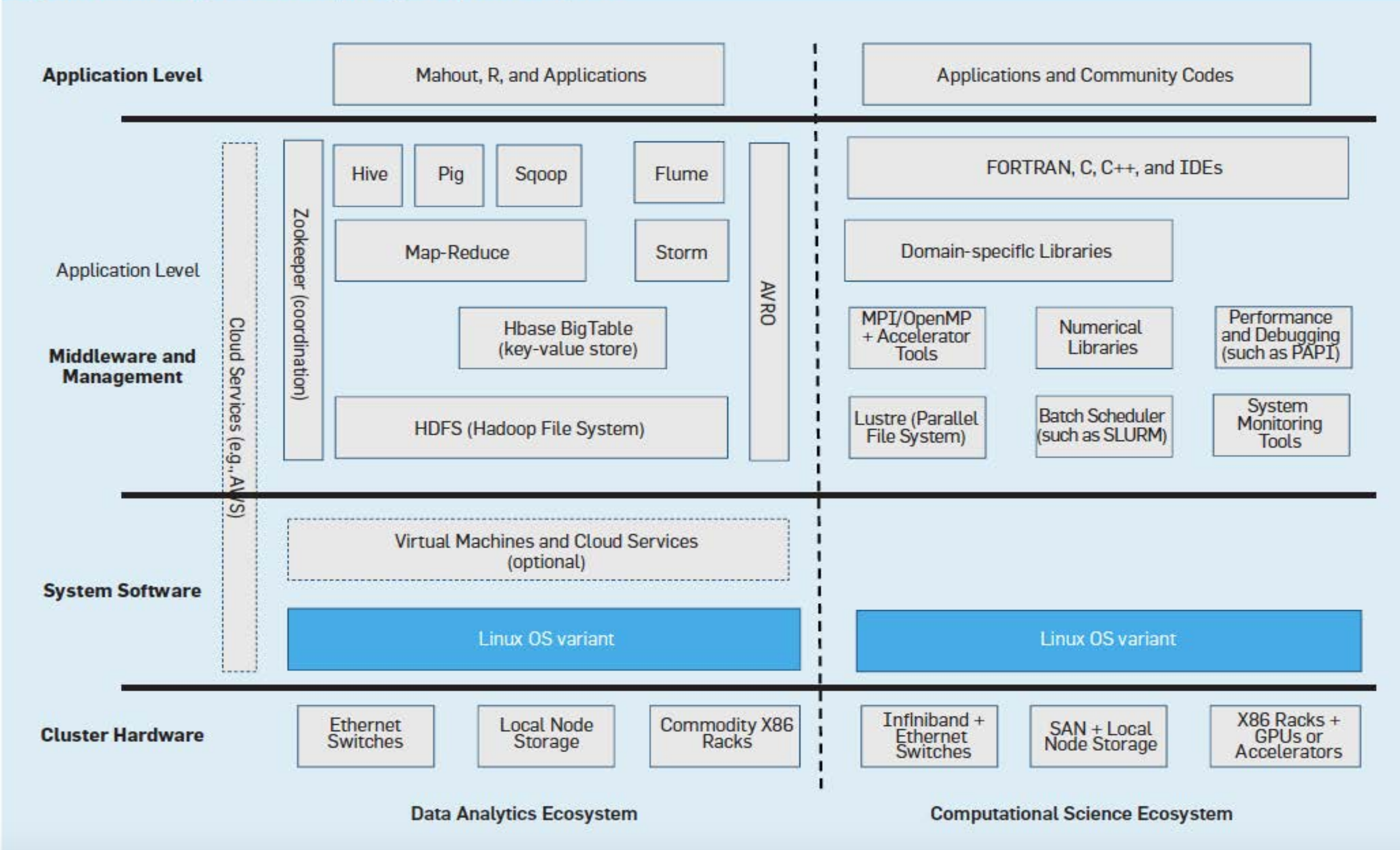
J.-C. Andre	G. Antoniu	M. Asch	R. Badia Sala	M. Beck	P. Beckman
T. Bidot	F. Bodin	F. Cappello	A. Choudhary	B. de Supinski	E. Deelman
J. Dongarra	A. Dubey	G. Fox	H. Fu	S. Girona	W. Gropp
M. Heroux	Y. Ishikawa	K. Keahey	D. Keyes	W. Kramer	J.-F. Lavignon
Y. Lu	S. Matsuoka	B. Mohr	T. Moore	D. Reed	S. Requena
J. Saltz	T. Schulthess	R. Stevens	M. Swany	A. Szalay	W. Tang
G. Varoquaux	J.-P. Vilotte	R. Wisniewski	Z. Xu	I. Zacharov	

November 2017

Successor to *The International Exascale Software Roadmap*, by many of the same authors and new authors from big data

Opportunity for applications: merging software for 3rd and 4th paradigms

Figure 1. Data analytics and computing ecosystem compared.



c/o Reed & Dongarra, Comm. ACM, July 2015

Interactions between application archetypes

Increasingly, there is scientific opportunity in pipelining

➔ *Convergence is ripe*

		To Simulation	To Analytics	To Learning
3rd	**Simulation provides**	–	**Physics-based "regularization"**	**Data for training, augmenting real-world data**
4th (a)	**Analytics provides**	**Steering in high dimensional parameter space; *In situ* processing**	–	**Feature vectors for training**
4th (b)	**Learning provides**	**Smart data compression; Replacement of models with learned functions**	**Imputation of missing data; Detection and classification**	–

How will complex PDE codes adapt?

- Programming model will still be dominantly 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) through dynamic languages or runtimes
- Noncritical parts will be made available for NUMA-aware work-stealing in economically sized chunks

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

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; …

- 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 analytics

Sources of nonuniformity

- System
  - *Already* important: manufacturing, OS jitter, TLB/cache performance variations, network contention,
  - *Newly* important: dynamic power management, more soft errors, more 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 of both types are similar when it comes to waiting at synchronization points
- Possible solutions for system nonuniformity will improve programmability for nonuniform problems, too ☺

Conclusions

- **Plenty of ideas exist to adapt or substitute for favorite solvers with methods that have:**
  - **reduced synchrony (in frequency and/or span)**
  - **higher residence on the memory hierarchy**
  - **greater SIMT/SIMD-style shared-memory concurrency**
  - **built-in resilience ("algorithm-based fault tolerance" or ABFT) to arithmetic/memory faults or lost/delayed messages**
- **Programming models and runtimes may have to be stretched to accommodate**
- **Everything should be on the table for trades, beyond disciplinary thresholds ➔ "co-design"**

Thanks to:

NVIDIA
GPU
RESEARCH
CENTER

intel
Intel Parallel Computing Centers

CRAY
THE SUPERCOMPUTER COMPANY
CENTER OF EXCELLENCE


Intel Xeon Phi Users Group
First Middle East Meeting
Sun-Wed, 22-25 April 2018
@ KAUST, Thuwal
Mecca Province
Saudi Arabia
ADVERT

Sponsors:
intel
Intel Parallel Computing Centers
CRAY
THE SUPERCOMPUTER COMPANY
Generation "Energy Aware"
Generation "Bulk Synchronous"
Meeting Themes:
Applications in Energy and Environment
Convergence of Large-scale Simulation and Big Data Analytics
Co-design for Exascale: Architecture / Algorithms / Applications
From Bulk-synchronous to Dynamic Task-based Algorithm Design


جامعة الملك عبدالله
للعلوم والتقنية
King Abdullah University of
Science and Technology
See you on the shores of
the beautiful Red Sea
david.keyes@kaust.edu.sa
Mediterranean Sea
IRAN
EGYPT
SUDAN
Arabian
Peninsula
SAUDI ARABIA
YEMEN
ETHIOPIA
Gulf of Aden
Persian Gulf