# ATPESC

ARGONNE TRAINING PROGRAM ON EXTREME-SCALE COMPUTING

# Algorithmic Adaptations to Extreme Scale

**David Keyes, Applied Mathematics & Computational Science**

**Director, Extreme Computing Research Center (ECRC)**

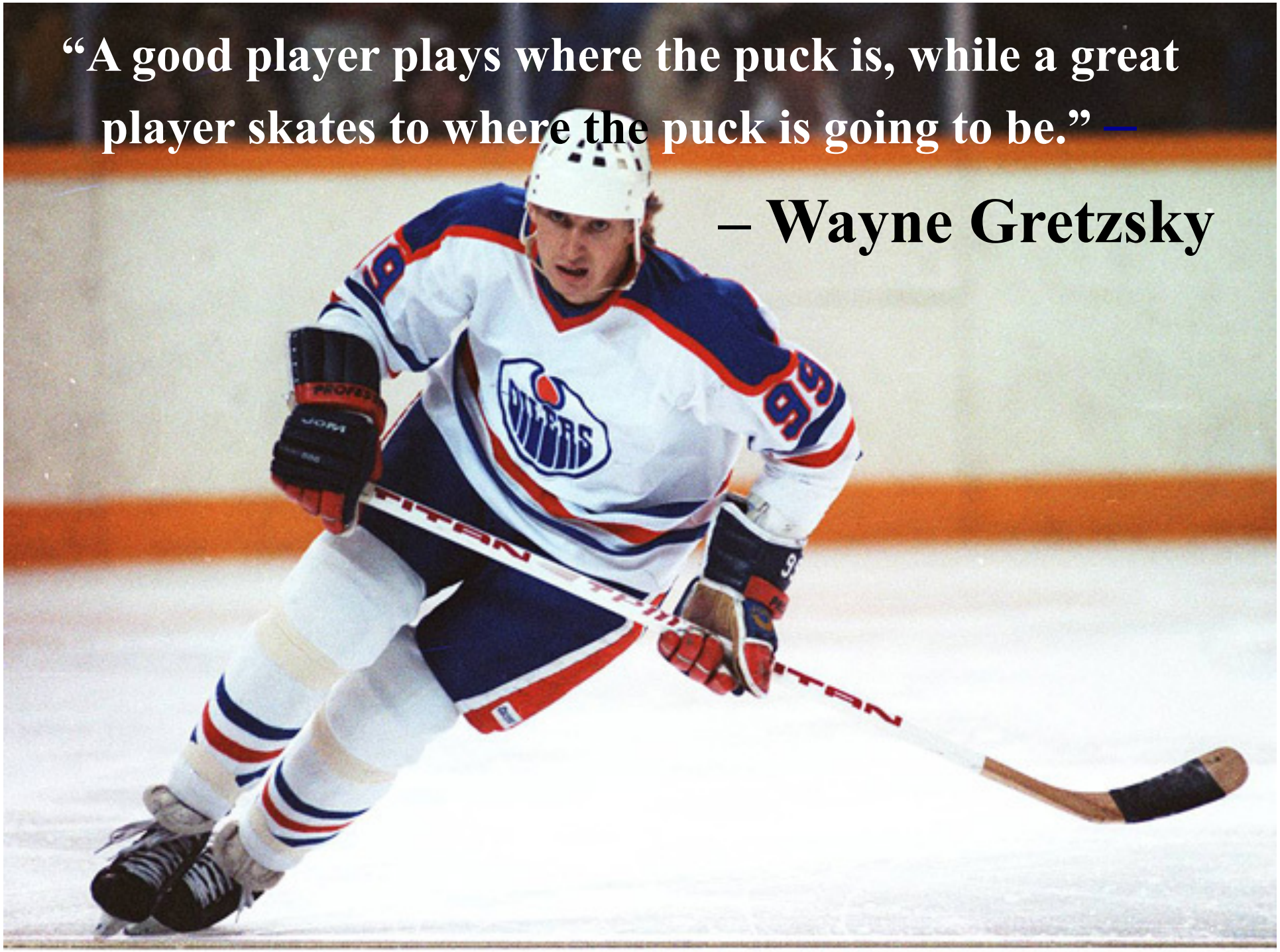
**King Abdullah University of Science and Technology**

# Tie-ins to other ATPESC'16 presentations

- **Numerous!**
  - **architecture, applications, algorithms, programming models & systems software, etc., form an *interconnected ecosystem***
  - **algorithms/software span diverging requirements in architecture (more uniformity) & application (more irregularity)**
- **To architecture presentations today**
  - **Intel, NVIDIA**
- **To programming models talks tonight thru Thursday:**
  - **MPI, OpenMP, Open ACC, OCCA, Chapel, Charm++, UPC++, ADLB**
- **To algorithms talks Friday and Monday:**
  - **Demmel, Diachin & FASTMath team, Dongarra**

# Shaheen I → Shaheen II

| IBM Blue Gene/P | Cray XC40 |
|---|---|
| **June 2009 (then #14)** | **July 2015 (then #7)** |
| **.222 Petaflop/s (peak)** | **7.3 Petaflop/s (peak, ↑ ~33X)** |
| Power: 0.5 **MW**<br>**0.44 GF/s/W** | Power: **2.8 MW (↑ ~5.5X)**<br>**~2.5 GF/s/Watt (↑ ~5X)** |
| Memory: **65 TeraBytes**<br>Amdahl-Case Ratio: **0.29 B/F/s** | Memory: **793 TeraBytes (↑ ~12X)**<br>Amdahl-Case Ratio: **0.11 B/F/s (↓ ~3X)** |
| I/O bandwidth: **25 GB/s**<br>Storage: **2.7 PetaBytes** | I/O bandwidth: **500 GB/s (↑ ~20X)**<br>Storage: **17.6 PetaBytes (↑ ~6.5X)** |
| Nodes: **16,384**<br>Cores: **65,536 at 0.85 Ghz** | Nodes: **6,192**<br>Cores: **198,144 at 2.3 Ghz** |
| Burst buffer:<br>**none** | Burst buffer:<br>**1.5 Petabytes, 1.2 TB/s bandwidth** |


"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"**

**Such algorithms may *or may not* be the best today; however, hardware trends can be extrapolated to infer algorithmic "sweet" spots.**

# Examples being developed at KAUST's Extreme Computing Research Center

- **ACR($\varepsilon$), a new spin on 46-year-old cyclic reduction that recursively uses $\mathcal{H}$ matrices on Schur complements to reduce O($N^2$) complexity to O($N \log^2 N$)**
- **FMM($\varepsilon$), a 30-year-old O($N$) solver for potential problems with good asymptotic complexity but a bad constant (relative to multigrid) when used at high accuracy, used in low accuracy as a preconditioner**
- **QDWH-SVD, a 3-year-old SVD algorithm that performs more flops but generates essentially arbitrary amounts of dynamically schedulable concurrency by recursive subdivision, and beats state-of-the-art on GPUs**
- **MWD, a multicore wavefront diamond-tiling stencil evaluation library that reduces memory bandwidth pressure on multicore processors**
- **BDDC, a preconditioner well suited for high-contrast elliptic problems that trades lots of local flops for low iteration count, now in PETSc**
- **MSPIN, a new nonlinear preconditioner that replaces most of the global synchronizations of Newton iteration with local problems**

# Background of this talk:

## *www.exascale.org/iesp*

INTERNATIONAL EXASCALE SOFTWARE PROJECT ROADMAP 1.0



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


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.

Eight of these co-authors will speak to you this week

# Uptake from IESP meetings

- While obtaining the next order of magnitude of performance, we also need an order more Flop/s per Watt
  - target: 50 Gigaflop/s/W, today best is 6.7 Gigaflop/s/W
  - tendency towards less memory and memory BW per flop
- Power may be cycled off and on, or clocks slowed and speeded
  - based on compute schedules (user-specified or software adaptive) and dynamic thermal monitoring
  - makes per-node performance rate unreliable*
- Draconian reduction required in power per flop and per byte may make computing and moving data less reliable
  - circuit elements will be smaller and subject to greater physical noise per signal, with less space and time redundancy for resilience in the hardware
  - more errors should be caught and corrected in software

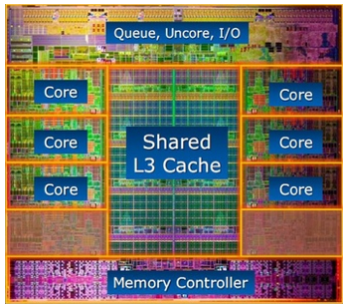
* "Equal work is not equal time" (Beckman, this morning)

# 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 will require 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)**

# Typical power costs per operation

| Operation | approximate energy cost |
|---|---|
| DP FMADD flop | 100 pJ |
| DP DRAM read-to-register | 5,000 pJ |
| DP word transmit-to-neighbor | 7,500 pJ |
| DP word transmit-across-system | 10,000 pJ |

**Remember that 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

## Moore's Law (1965) does not end yet 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 $N_a$ | $\kappa$ |
| 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, $R_L = \rho L/Wt$ | $\kappa$ |
| Normalized voltage drop $IR_L/V$ | $\kappa$ |
| Line response time $R_LC$ | 1 |
| Line current density $I/A$ | $\kappa$ |

Robert Dennard, IBM
(inventor of DRAM, 1966)

**Eventually, processing is limited by transmission, as known for > 4 decades**

# Some exascale architecture trends

- Clock rates cease to increase while arithmetic capability continues to increase dramatically w/ concurrency consistent with Moore's Law
- Memory storage capacity diverges exponentially below arithmetic capacity
- Transmission capability (memory BW and network BW) diverges exponentially below arithmetic capability
- Mean time between hardware interrupts shortens
- ➔ Billions of $ € £ ¥ of scientific software worldwide hangs in the balance until better algorithms arrive to span the architecture-applications gap

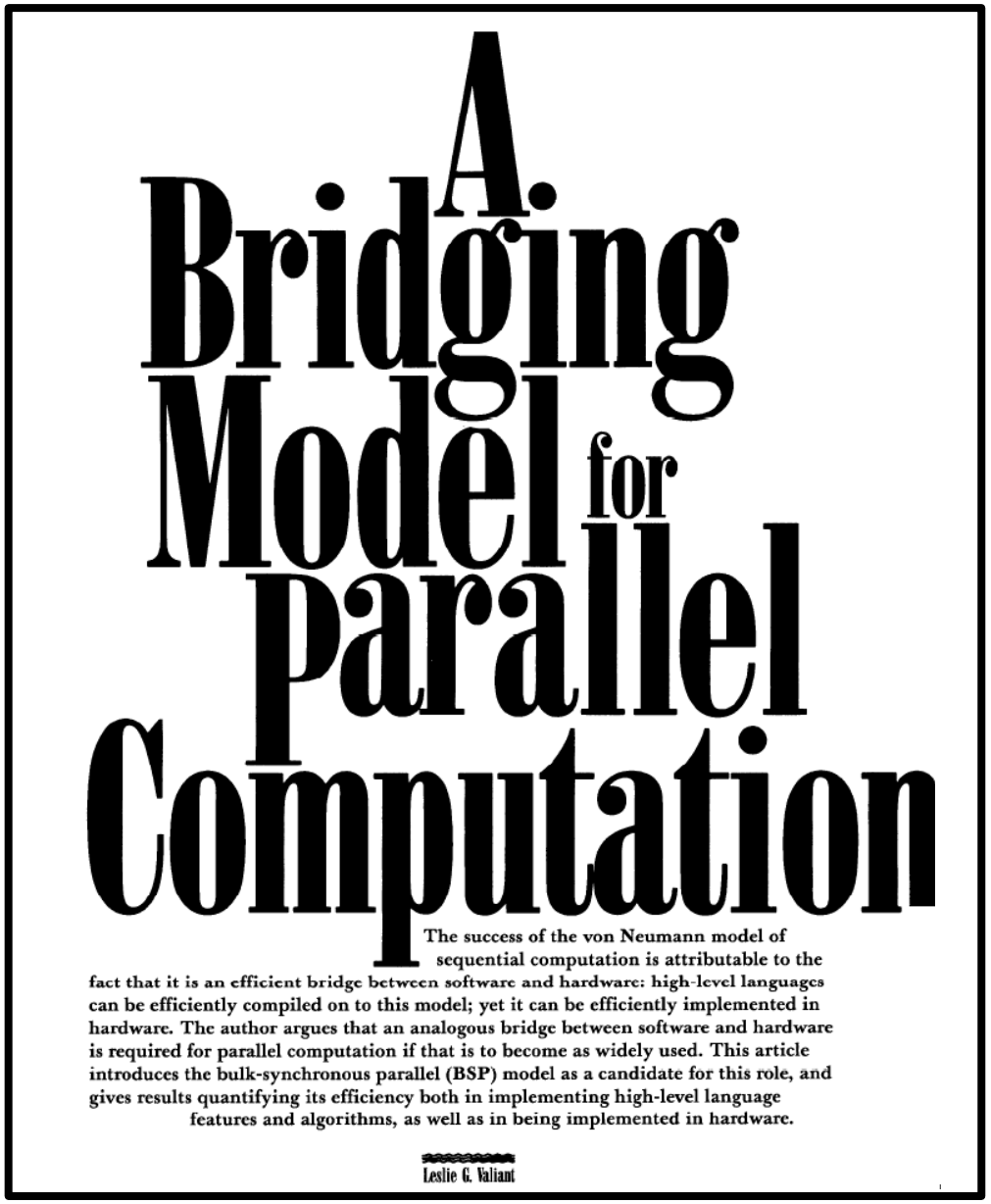
# Node-based "weak scaling" is routine; thread-based "strong scaling" is the game

- 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
- The real challenge is usefully 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)
  - don't need to wait for full exascale systems to experiment in this regime – the battle is fought on individual shared-memory nodes


BSP
generation
Energy-aware
generation

# Bulk Synchronous Parallelism

**Leslie Valiant, Harvard**
**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.

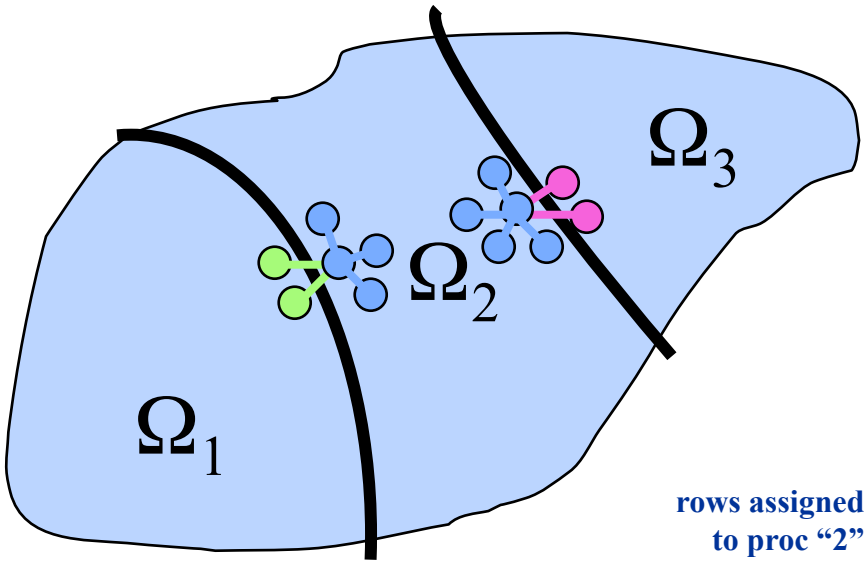
Leslie G. Valiant

**Comm. of the ACM, 1990**

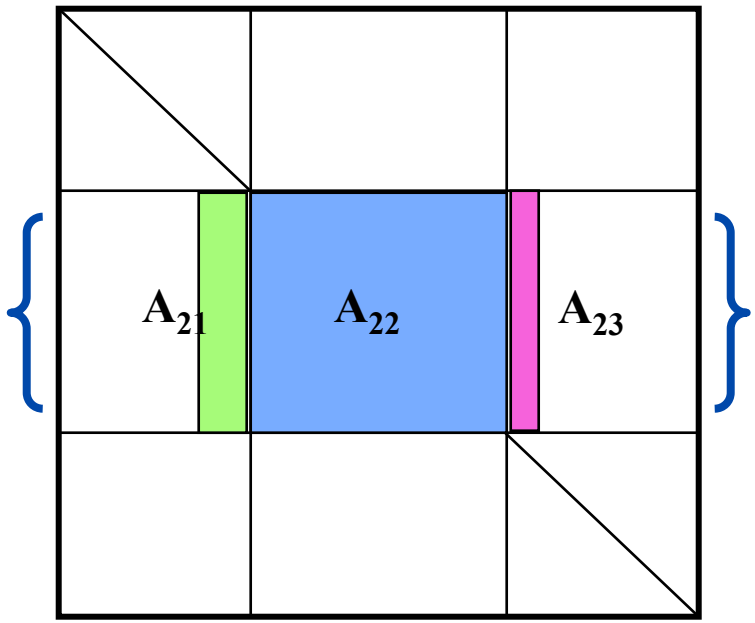
# How are most simulations implemented at the petascale today?

- **Iterative methods based on data decomposition and message-passing**
  - data structures (e.g., grid points, particles, agents) are distributed
  - each individual processor works on a subdomain of the original
  - 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 BSP/SPMD/CSP**
  - Bulk Synchronous Programming
  - Single Program, Multiple Data
  - Communicating Sequential Processes

# BSP parallelism w/ domain decomposition




rows assigned
to proc "2"




**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<br>Year | Gigaflop/s delivered to applications |
|---|---|
| 1988 | 1 |
| 1998 | 1,020 |
| 2008 | 1,350,000 |

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

# Extrapolating exponentials eventually fails

- **Scientific computing at a crossroads w.r.t. extreme scale**
- **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)**

# 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
  - not a "distraction," but an intellectual stimulus
  - potentially big gains in adapting to new hardware environment
  - 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., require frequent synchronizing global communication**
  - **inner products, norms, and fresh global residuals are "addictive" idioms**
  - **tends to hurt efficiency beyond 100,000 threads**
  - **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 10.6 million on the most powerful system today**

# Conclusions, up front

- **Plenty of ideas exist to adapt or substitute for favorite solvers with methods that have**
  - **reduced synchrony (in frequency and/or span)**
  - **greater arithmetic intensity**
  - **greater 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"**

# Bad news/good news (1)

- **One will have to explicitly control more of the 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 already**
  - **but vertical replication into caches and registers was (until GPUs) mainly 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 scientific and engineering users into more arithmetically intensive formulations than (mainly) PDEs**
  - **tomorrow's optimal methods will (by definition) evolve to conserve whatever is expensive**

# Bad news/good news (3)

- Fully hardware-reliable executions may be regarded as too costly/synchronization-vulnerable
- 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
- Several examples in direct and iterative linear algebra
- Anticipated by Von Neumann, 1956 ("Synthesis of reliable organisms from unreliable components")

# Bad news/good news (4)

- **Default 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 will have to 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, but also the last values, and forward just the deltas**
- **Equidistributing errors properly to minimize resource use will lead to innovative error analyses in numerical analysis**

# Bad news/good news (5)

- **Fully deterministic algorithms may be regarded as too synchronization-vulnerable**
  - **rather than wait for missing data, e.g., in the tail Pete showed earlier, 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.**

# Warning: not all accept the full 4-fold agenda

- **Non-controversial:**
  - **reduced synchrony (in frequency and/or span)**
  - **greater arithmetic intensity**
- **Mildly controversial, when it comes to porting real applications:**
  - **greater SIMD-style shared-memory concurrency**
- **More controversial:**
  - **built-in resilience ("algorithm-based fault tolerance" or ABFT) to arithmetic/memory faults or lost/delayed messages**

# The world according to algorithmicists

- **Algorithms must adapt to span the gulf between aggressive applications and austere architectures**
  - **full employment program for computational scientists and engineers**
  - **see, e.g., recent postdoc announcements from**
    - **Berkeley (8) for Cori Project (Cray & Intel MIC)**
    - **Oak Ridge (8) for CORAL Project (IBM & NVIDIA NVLink)**
    - **IBM (10) for Data-Centric Systems initiative**

  **for porting applications to emerging hybrid architectures**

# Required software at exascale

## 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

# Optimal hierarchical algorithms

- **At large scale, one must start with algorithms with optimal asymptotic scaling, $O(N \log^p N)$**
- **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)***

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

* **hierarchically low-rank matrices**

# Recap of algorithmic agenda

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

# Some algorithmic "points of light"

**Sample "points of light" that accomplish one or more of these agendas**

- **DAG-based data flow for dense symmetric linear algebra**
- **In-place GPU implementations of dense symmetric linear algebra**
- **Fast multipole preconditioning for Poisson solves**
- **Algebraic fast multipole for variable coefficient problems**
- **Nonlinear preconditioning for Newton's method**
- **Very high order discretizations for PDEs**

NVIDIA GPU RESEARCH CENTER

intel Intel Parallel Computing Centers

CRAY THE SUPERCOMPUTER COMPANY
CENTER OF EXCELLENCE

# For details: ATPESC 2015

- **Second half of my presentation last year briefly describes projects in all of these areas**
- **Or write**

  david.keyes@kaust.edu.sa **(repeated on last slide)**

# For closing minutes of ATPESC 2016

- **Our 2016 Gordon Bell submission**
- **CFD application, with emphasis on very high order**
- **Joint with:**
  - **U of Chicago: Max Hutchinson**
  - **Intel: Alex Heinecke**
  - **KAUST: Matteo Parsani, Bilel Hadri**
  - **Argonne: Oana Marin, Michel Shanen**
  - **Cornell: Matthew Otten**
  - **KTH: Philipp Schlatter**
  - **U of Illinois: Paul Fischer**

# HIGH-ORDER METHODS FOR PDES


Discretization order allows trades on *per-degree-of-freedom* basis

- Computational and memory cost *per DOF* rise with order
- For smooth solutions, accuracy improves with order
  - Fewer DOFs needed for a specified accuracy

High-order discretizations offer favorable extreme-scale properties

- Potentially less total memory required for a given accuracy
- Higher arithmetic intensity
- Better data locality

These claims deserve demonstration and quantification in various settings

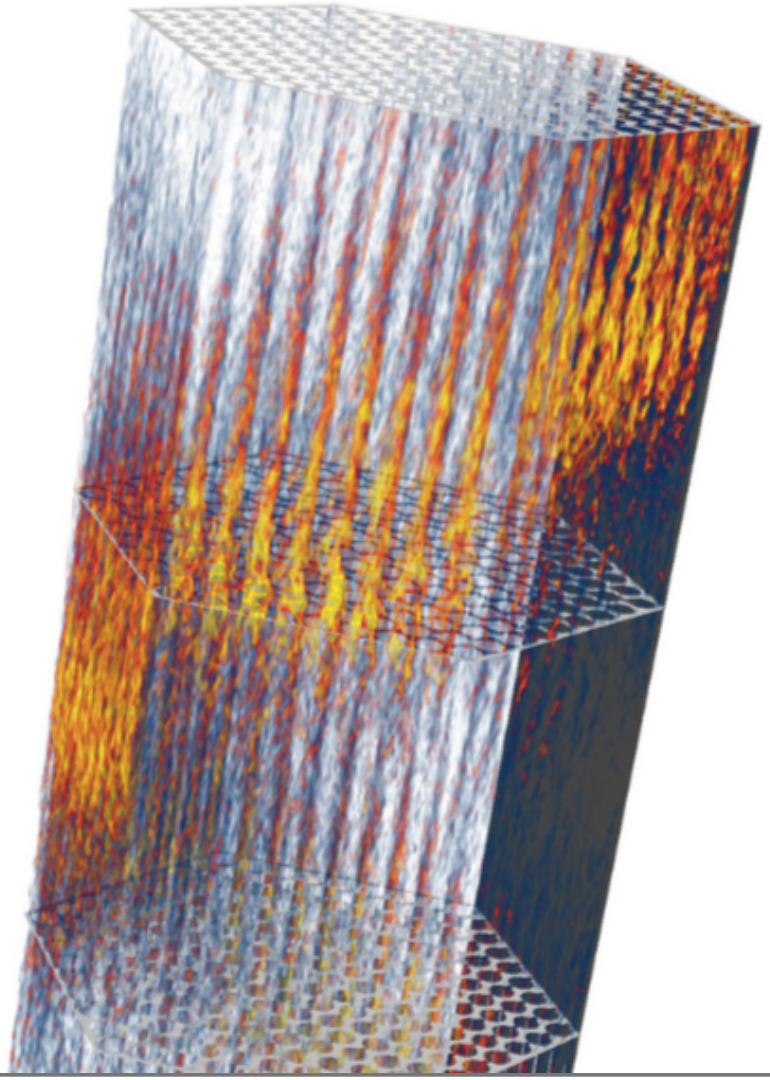
- Explore range of orders and number of DOFs
- Measure accuracy in terms of physical output quantities of interest (QOI)
- Measure cost in terms of core-hours consumed

# NekBox
## Code Background


- Nek5000: a spectral element research code written in Fortran and C with MPI for incompressible Stokes or Navier-Stokes fluid flow, including heat transfer, species transport, and MHD
  - Developed at Argonne by Paul Fischer and collaborators in the late 1980s
  - Shared 1999 Gordon Bell Special Prize (ASCI Red, 0.319 TF/s)
  - Widely used open source code, ported at its current extremes to over 1M cores with one or more elements per core
  - Employs spectral element discretization with high spatial order and high semi-implicit temporal discretization
  - Equipped with fast solvers for the Poisson operators in the semi-implicit temporal discretization
- NekBox: simplified-geometry fork to study HPC adaptations and extend order upward
  - Ported to high performance on Intel many-core processors by Maxwell Hutchinson, PhD candidate, DOE CSGF, UChicago
  - Being equipped with high-order entropy-stable time stepping by Matteo Parsani, Assistant Professor, KAUST

# EXAMPLE NEK5000 APPLICATION

## INTERNAL INCOMPRESSIBLE FLOWS (FIG C/O A. R. SIEGEL, ANL)



# NEK FORMULATION – 3


- Spectral element Navier-Stokes solvers treat backwards Helmholtz term, $\alpha I - \nabla^2$, implicitly to remove most stringent CFL stability bound
- Nonlinear terms treated explicitly
- Numerical dispersion and dissipation are very small

# TESTS


NekBox on dual 16-core Haswell, 18-core Power PC A2, and KNL

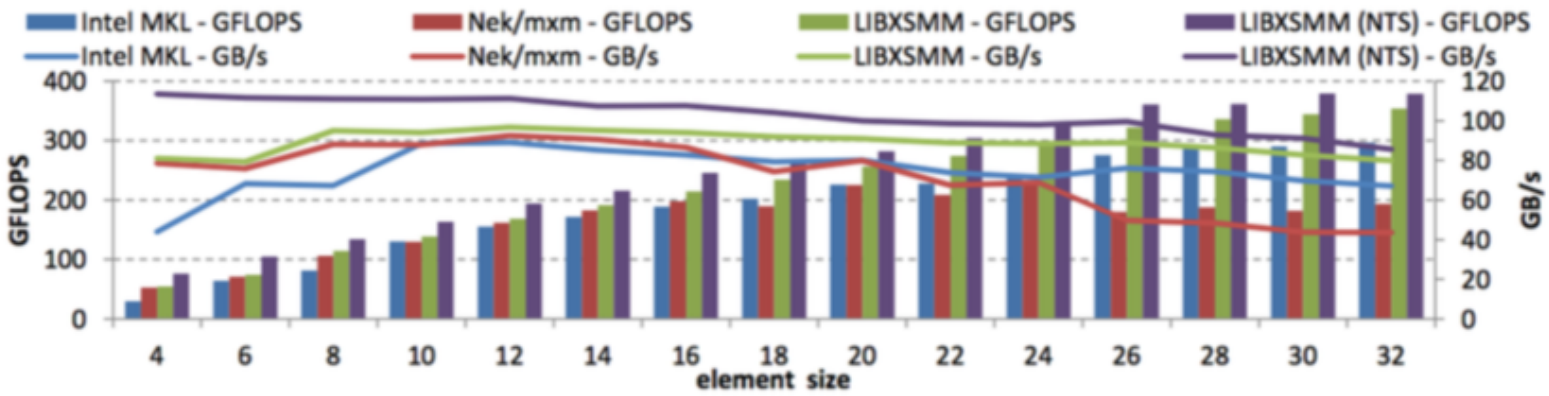
- Shaheen-2 (KAUST), ranked #10 on June 2016 Top500
- Mira (ANL), ranked #6
- Avalon, a pre-release testbed for KNL

Many details (optimizations based on just-in-time compilation with LIBXSMM, non-temporal stores, performance profiles by computational phase, etc.) left to the publications:

- *Efficiency of High Order Spectral Element Methods on Petascale Architectures*, M. Hutchinson, A. Heinecke, H. Pabst, G. Henry, M. Parsani, and D. Keyes, 2016, International Supercomputing Conference (ISC'16).
- *Sustained Petascale Direct Numerical Simulation of Secondary Flows in Square Ducts*, M. Hutchinson, A. Heinecke, M. Parsani, O. Marin, M. Schanen, M. Otten, B. Hadri, P. Schlatter, D. Keyes, and P. Fischer, 2016, submitted for Gordon Bell Prize at SC'16. [not a finalist]

# "SECRET SAUCES"


- LIBXSMM – creates a just-in-time specific kernel for each small matmult size, tuned for all recent Intel cores
- Non-temporal stores (NTS)




Evaluation of Helmholtz operator for single node of Shaheen: floating point performance (bars) and realized bandwidth (curves) for (1D) element sizes 4 through 32

# RAYLEIGH-TAYLOR INSTABILITY (RTI)

## Boussinesq form


$$\frac{\partial u}{\partial t} + u \cdot \nabla u = -\nabla p + \nu \nabla^2 u + \tilde{g} T$$

$$\frac{\partial T}{\partial t} + u \cdot \nabla T = \alpha \nabla^2 T$$

$$\nabla \cdot u = 0,$$

Initial perturbation on temperature

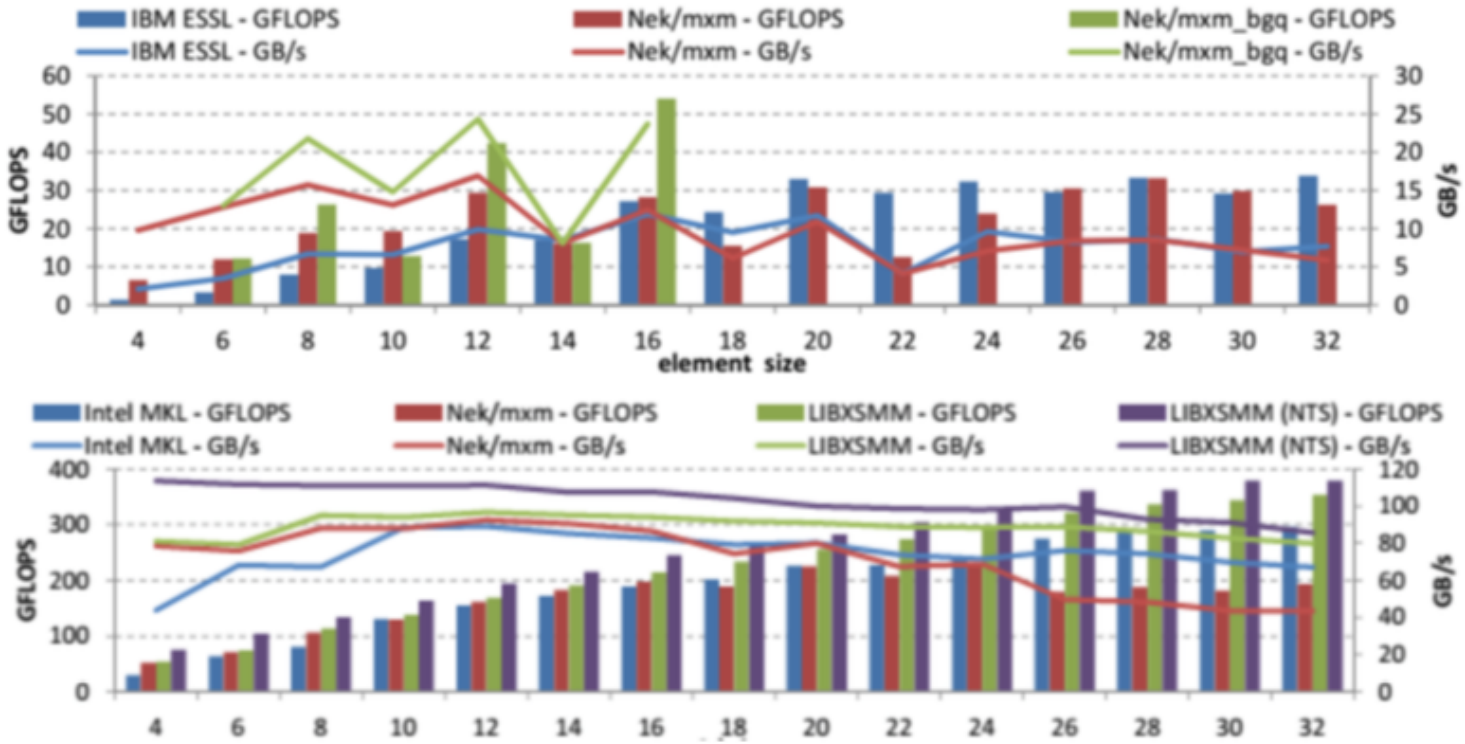
$$T(x, y, z, 0) = A \cdot \mathrm{erf}\left[\frac{z + a_0 \cos(2\pi x/\lambda) \cos(2\pi y/\lambda)}{\delta}\right]$$

Dimensionless parameters

$$\mathrm{Gr} = \frac{A\tilde{g}\lambda^3}{\nu^2}, \qquad \mathrm{Pr} = \frac{\nu}{\alpha}$$

# HELMHOLTZ ON MIRA AND SHAHEEN



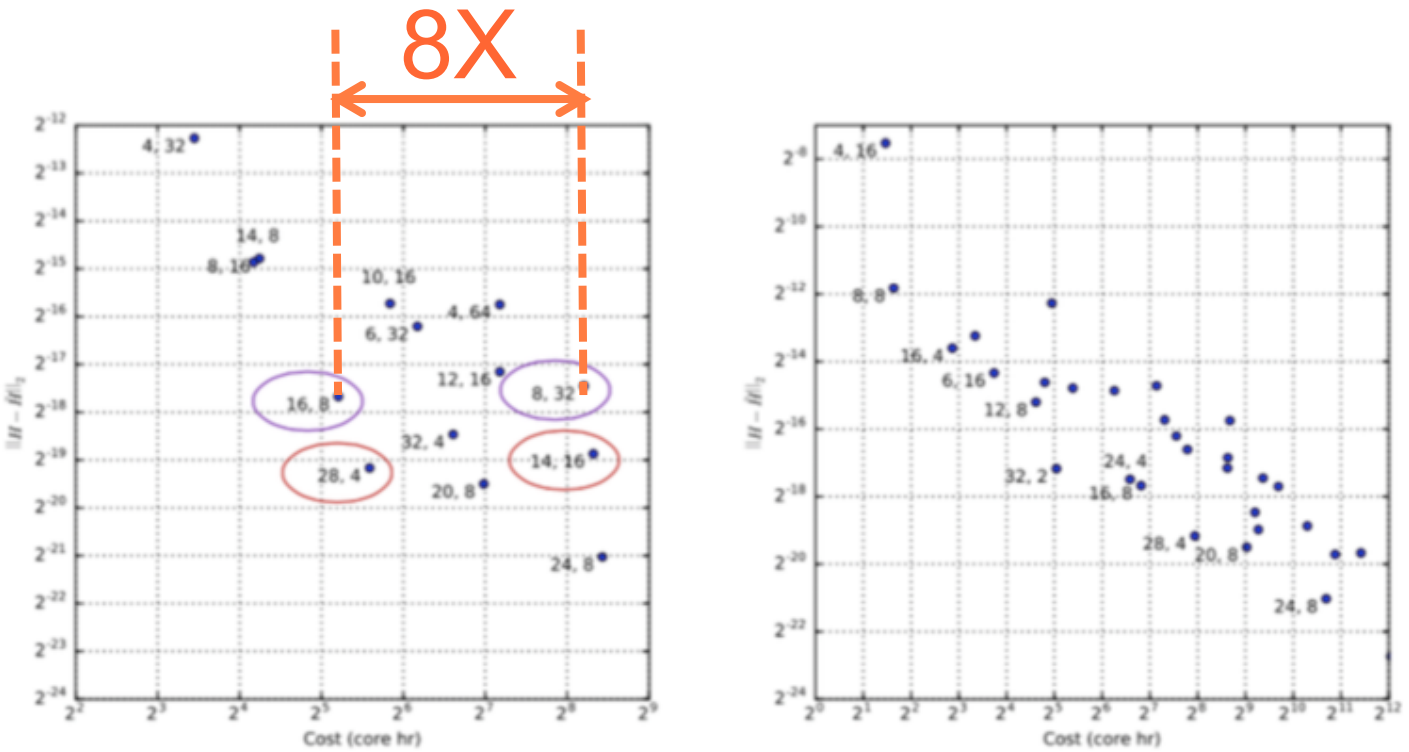



Performance of Helmholtz operator for single nodes of Mira (top, with ESSL and mxm_bgq) and Shaheen (bottom, with MKL and LIBXSMM)

# SPATIAL ERROR VERSUS CORE HOURS

(order,element#) pairs

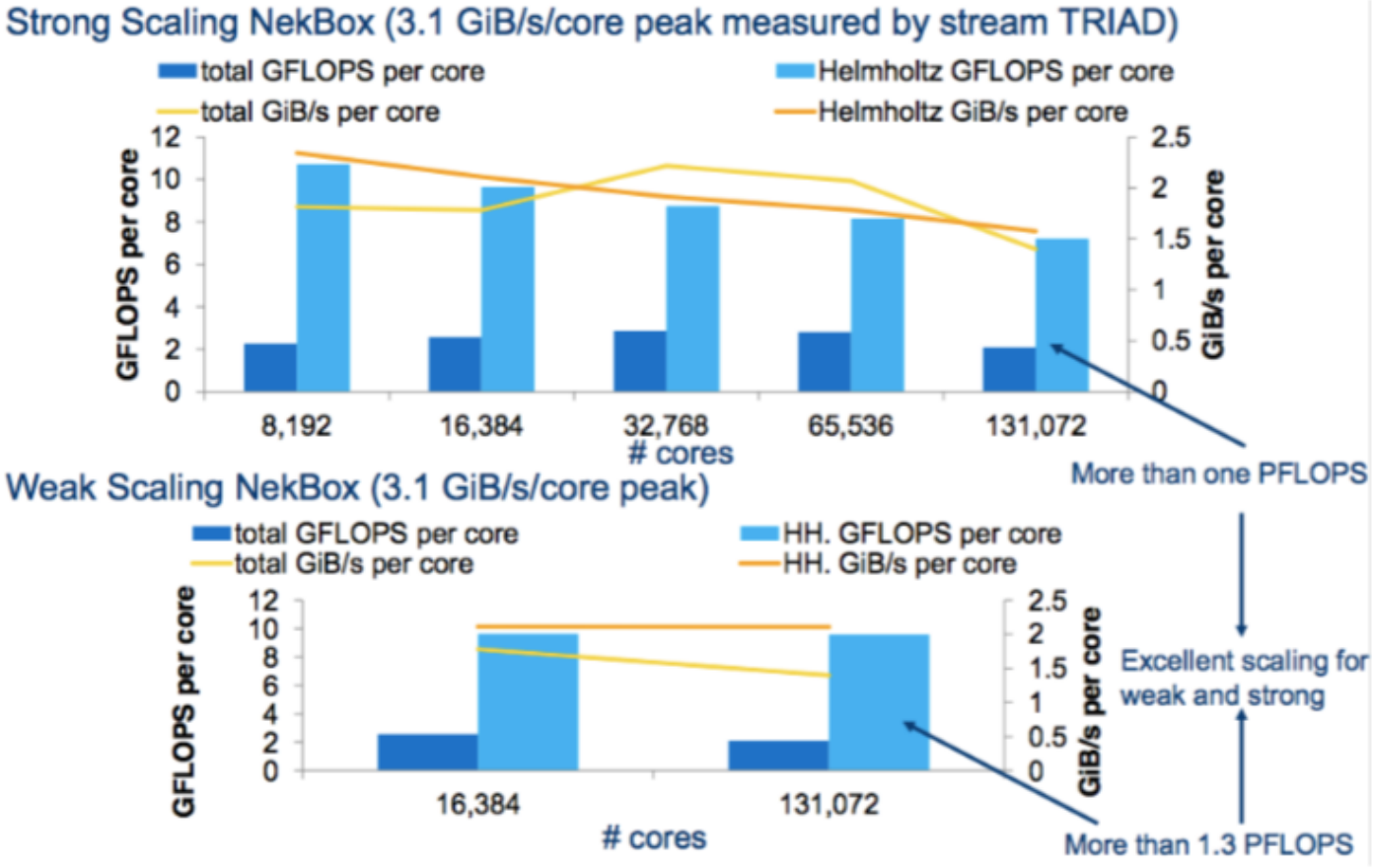





Accuracy (bubble height) versus cost (core-hours) as a function of element order on Shaheen (left) and Mira (right); labeled by (order,element#) pairs

# STRONG AND WEAK SCALING ON SHAHEEN






21% of theoretical peak

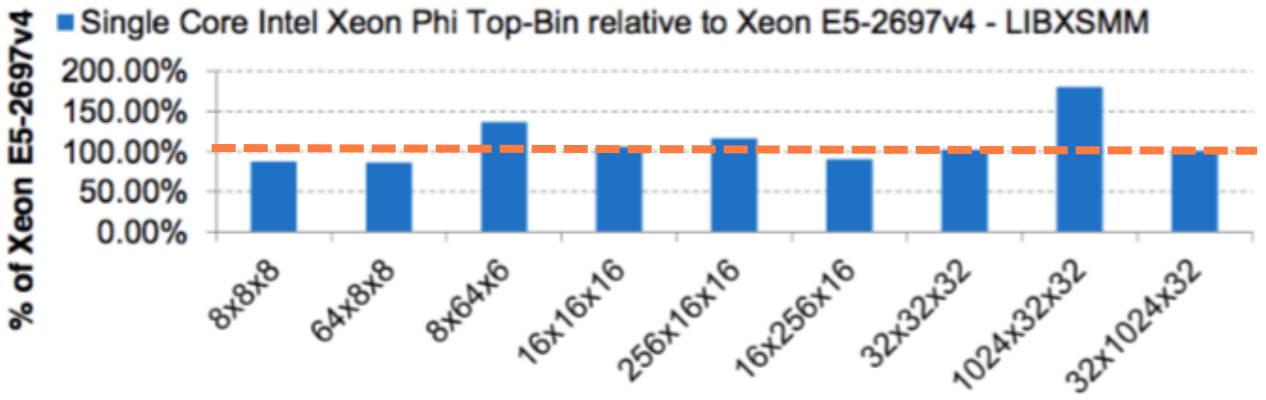
27% of theoretical peak

Tests range from 16K to 256K points per MPI rank, 1M to 4B DOFs

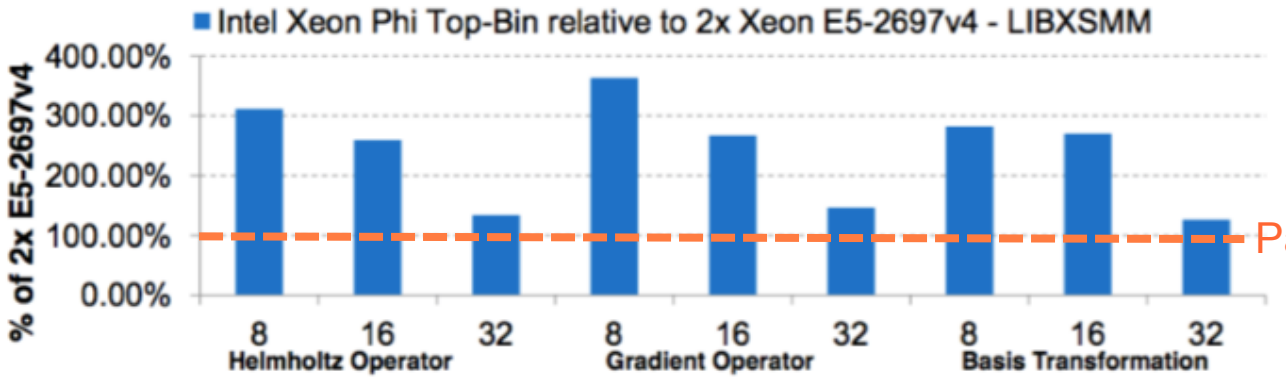
Grashof number: 17,234 | Prandtl number: 1

# KNIGHTS LANDING RESULTS






Parity with Haswell (1 core each)

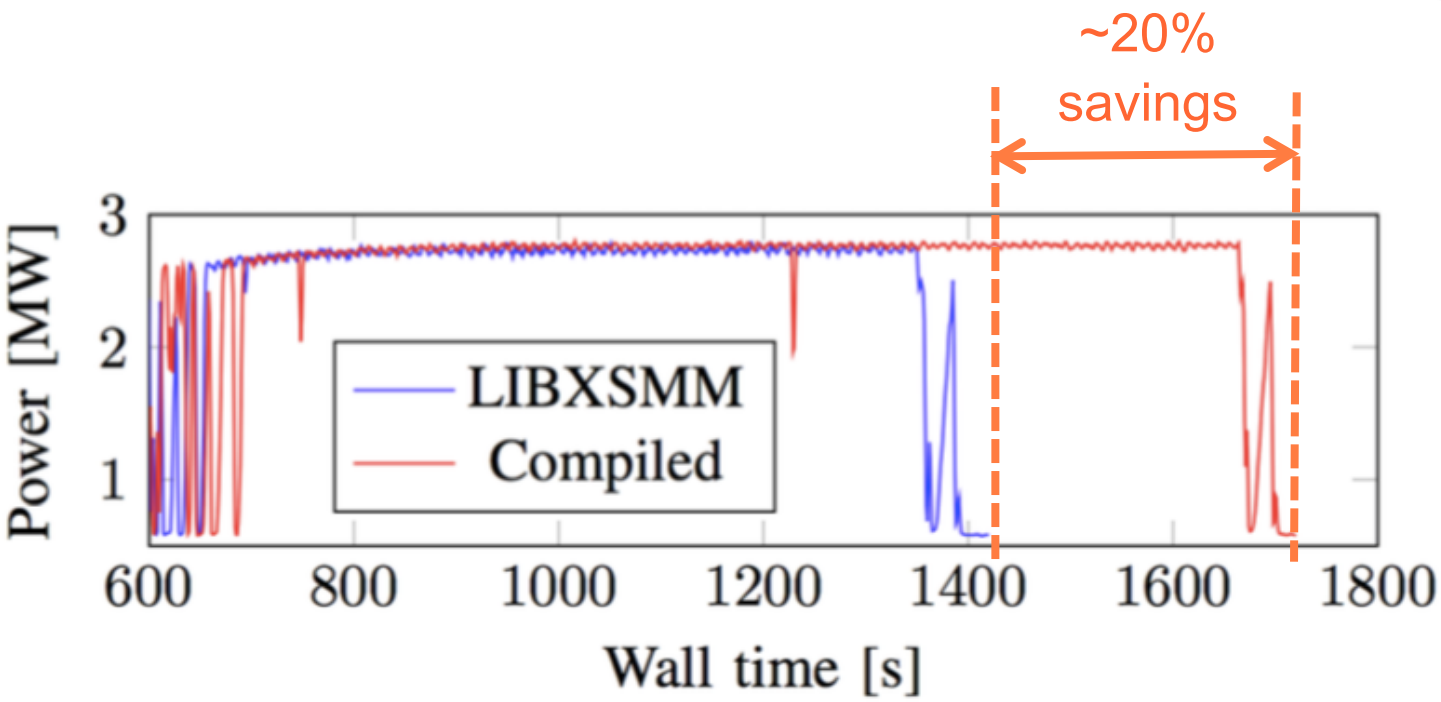



Parity with Haswell (full node each)

# POWER PROFILE ON SHAHEEN

Fortran and LIBXSMM





# CONCLUSIONS
## Choice of Spatial Order


- High order methods are able to take advantage of wider vectors and higher compute to memory ratios to reach higher order *at little to no marginal cost on a per-step basis*.
- Increases in cost can come in through coupling to the choice of time-step. In cases where the time-step is chosen to be smaller than that required for stability, the number of time steps is decoupled from the order and the marginal cost in higher order at fixed degrees of freedom is exclusively through an increase in arithmetic intensity.
- The order should be chosen to be at least large enough to saturate the floating point capabilities of the architecture in the order-dependent kernels, because increasing the order to that point significantly improves accuracy at no marginal computational cost. On BlueGene/Q, this mark is order 15, while on the Cray XC40 it is 31.


BSP
generation
Energy-aware
generation
Skate to where the puck is going to be!


جامعة الملك عبدالله
للعلوم والتقنية
King Abdullah University of
Science and Technology
Thank you
شكرا
david.keyes@kaust.edu.sa

# Extra Slide

# Philosophy of software investment