



# **Facing the Four Algorithmic Frontiers of Exascale**

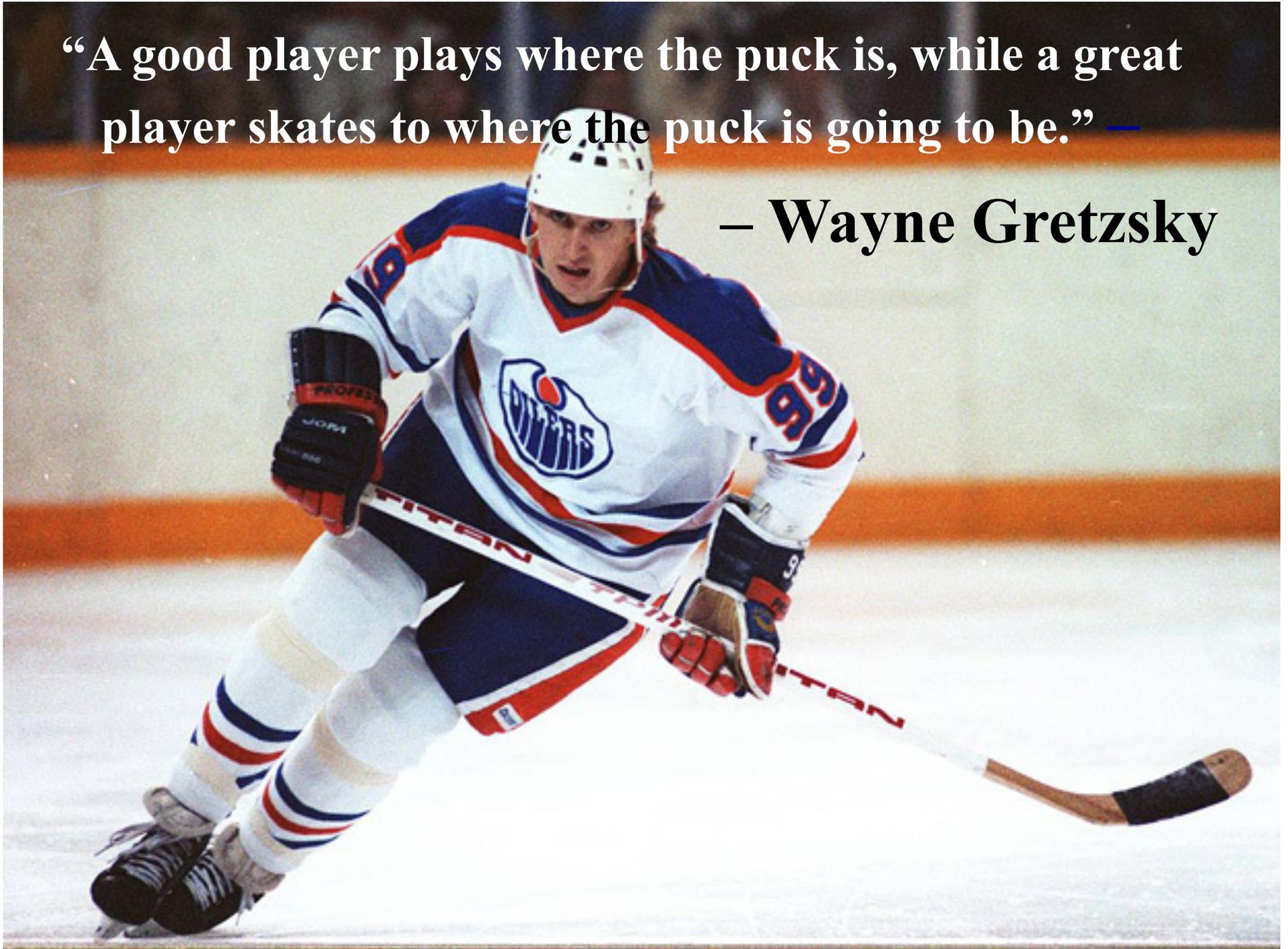
**David Keyes, Applied Mathematics & Computational Science**

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

**King Abdullah University of Science and Technology**

**“A good player plays where the puck is, while a great player skates to where the puck is going to be.” —**

**– Wayne Gretzky**



# Aspiration for this brief talk

To paraphrase Gretzky:

“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 their sweet spots – as the traditional objective of conserving flops is replaced with the new objective of reducing communication and synchronization – *for a new computational complexity calculus.*

# Reflections on algorithmic tuning from this morning's talks – a brave new world!

- **Classic trade-offs (pre-exascale/SPMD Bulk Synchronous)**
  - recompute on-the-fly for less storage/traffic
  - more expensive inner iterations for fewer outer iterations
  - redundant computation for less communication
  - less stability for less frequent global reductions
- **New trade-offs (energy-austere/emerging architectures)**
  - algorithmic responsibility for resilience for less power/energy
  - redundant solves for shorter executions
  - over-decomposition for less synchrony
  - less determinism for less synchrony
  - more flops per iteration for fewer iterations
  - more flops for more concurrency
  - more precision for fewer iterative corrections

# Some exascale trends: the four architectural frontiers

- 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*

# Authority for architectural roadmap: *www.exascale.org/iesp*

INTERNATIONAL  
**EXASCALE** ROADMAP 1.0  
SOFTWARE PROJECT



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

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  
Adoffy 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  
Mitsuhsa 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



ScalA15 | 16 Nov 2015

# Why exa- is different

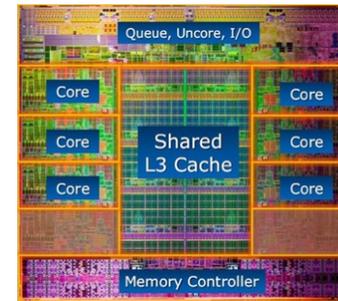
Which steps of FMADD take more energy?

64-bit floating-point fused multiply add

or

moving four 64-bit operands 20 mm across the die

$$\begin{array}{r} 934,569.299814557 \quad \text{input} \\ \times \quad 52.827419489135904 \quad \text{input} \\ \hline = 49,370,884.442971624253823 \\ + \quad 4.20349729193958 \quad \text{input} \\ \hline = 49,370,888.64646892 \quad \text{output} \end{array}$$



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)

# Today's power costs per operation

Operation	approximate energy cost
DP FMADD flop	100 pJ
DP DRAM read-to-register	4800 pJ
DP word transmit-to-neighbor	7500 pJ
DP word transmit-across-system	9000 pJ

Remember that a *pico* ( $10^{-12}$ ) of something done *exa* ( $10^{18}$ ) times per second is *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 ( $\$0.12/\text{KW-hr} \times 8760 \text{ hr/yr}$ )
  - We “use” 1.4 KW continuously, so 100MW is 71,000 people

# Why exa- is different

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

Table 1  
Scaling Results for Circuit Performance

Device or Circuit Parameter	Scaling Factor
Device dimension $t_{ox}, L, W$	$1/\kappa$
Doping concentration $N_a$	$\kappa$
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 $IR_L/V$	1
Line response time $R_L C$	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

# Simulation enabled by price and capability

By the Gordon Bell Prize, awarded since 1988, simulation *cost per performance* has improved by nearly a million times in two decades. Performance on *real applications* (e.g., mechanics, materials, petroleum reservoirs, gravitation) has improved *more than a million times*.

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

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

**Scientists and engineers plan along these trends ...**

# ... likewise, President Obama

THE WHITE HOUSE

Office of the Press Secretary

For Immediate Release

July 29, 2015

EXECUTIVE ORDER

- - - - -

## CREATING A NATIONAL STRATEGIC COMPUTING INITIATIVE

By the authority vested in me as President by the Constitution and the laws of the United States of America, and to maximize benefits of high-performance computing (HPC) research, development, and deployment, it is hereby ordered as follows:

Section 1. Policy. In order to maximize the benefits of HPC for economic competitiveness and scientific discovery, the United States Government must create a coordinated Federal strategy in HPC research, development, and deployment. Investment in HPC has contributed substantially to national economic prosperity and rapidly accelerated scientific discovery. Creating and deploying technology at the leading edge is vital to advancing my Administration's priorities and spurring innovation. Accordingly, this order establishes the

# Optimal hierarchical algorithms

- **Asymptotically in scale, one should restrict attention to algorithms with optimal scaling in memory requirements and flops,  $O(k^r N \log^p N)$**
- **$p$  is typically 0, 1, or at worst 2**
- **$k$  represents a local rank or polynomial expansion parameter**
- **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, KAUST PhD, 2013

# Shared memory strong scaling is the game

- **Expanding the number of nodes (processor-memory units) beyond  $10^6$  is *not* a serious threat for**
  - ◆ **optimal algorithms**
  - ◆ **with amortizable precise load balancing on performance-reliable nodes**
  - ◆ **connected with a log-diameter or high-radix switch network**
- **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**

## Main challenge at exascale

- **Almost all “good” algorithms in linear algebra, differential equations, integral equations, signal analysis, etc., require frequent synchronizing global communication**
  - ◆ inner products, norms, (global) transposes, and fresh (global) residuals are “addictive” idioms
  - ◆ can make algorithms fragile even for smaller concurrency, due to algorithmic load imbalance, hardware performance variation, etc.
- **Concurrency is heading into the billions of cores**
  - ◆ Temporarily leveled off at 3 million, but has to rise with bounded clock speeds

A hand holding a red baton, symbolizing energy-aware generation. The hand is positioned on the left side of the frame, with the baton extending towards the right. The background is a soft, blurred landscape with a warm, golden light, suggesting a sunset or sunrise. The text "Energy-aware generation" is overlaid on the hand in a white, serif font.

Energy-aware  
generation

Bulk synchronous  
generation

## The four algorithmic frontiers

- **Must adapt or substitute for favorite methods with new methods that have**
  - ◆ **reduced synchrony (in frequency and/or span)**
  - ◆ **greater arithmetic intensity**
  - ◆ **greater SIMD/SIMT-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**

# Mind the gap

- **Algorithms must adapt to span the gap between aggressive applications and austere architectures**
  - ◆ **full employment program for computational scientists and engineers**
  - ◆ **Like the new post-doc programs at Lawrence Berkeley and Oak Ridge National Labs**

# 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 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

# Dominant cycle burners

- **Poisson-like elliptic problems**
  - ◆ **Highest order operators in many PDE-based models**
    - fluids, solids, E&M, radiation, DFT, MD, etc.
- **Linear algebra on dense symmetric/Hermitian matrices**
  - ◆ **Generalized eigenproblems in chemistry/materials**
    - coming from various simplified models of Schroedinger in chemistry
  - ◆ **Covariance matrices**
    - coming from problems in spatial statistics
  - ◆ **Schur complement matrices**
    - coming from dimension reduction in PDEs
  - ◆ **Reduced Hessian matrices**
    - coming from problems in PDE-constrained optimization

# Hierarchical algorithms (esp. for PDEs)

- **Fast Fourier Transform**

- ◆ fragile (const. coeff.), but can be preconditioner for variable coeff.
- ◆ distributed multi-D version requires complete in-place exchange

- **Multigrid**

- ◆ sometimes hard to make robust for: anisotropy, inhomogeneity, indefiniteness, skew-symmetry, multiple components
- ◆ adaptive algebraic versions find coarse spaces automatically
- ◆ coarse space basis vectors can be  $O(N)$
- ◆ algebraic versions suffer from coarse grid stencil bloat
- ◆ can relax robustness requirements, use effectively as preconditioner
- ◆ frequent synchronization between levels
- ◆ main application vulnerability: Helmholtz-like indefiniteness
- ◆ *exascale features*: poor arithmetic intensity, but good algorithmic-based fault tolerance, new mult-add version less synchronous

# Hierarchical algorithms (esp. for PDEs)

- **Fast Multipole**

- ◆ fragile (needs Green's function), but interesting as a preconditioner
- ◆ as a high- $k$  solver typically loses to multigrid in BSP settings
- ◆ *exascale features*: good SIMT structure with high arithmetic intensity at leaves, concurrency across levels, not too synchronous

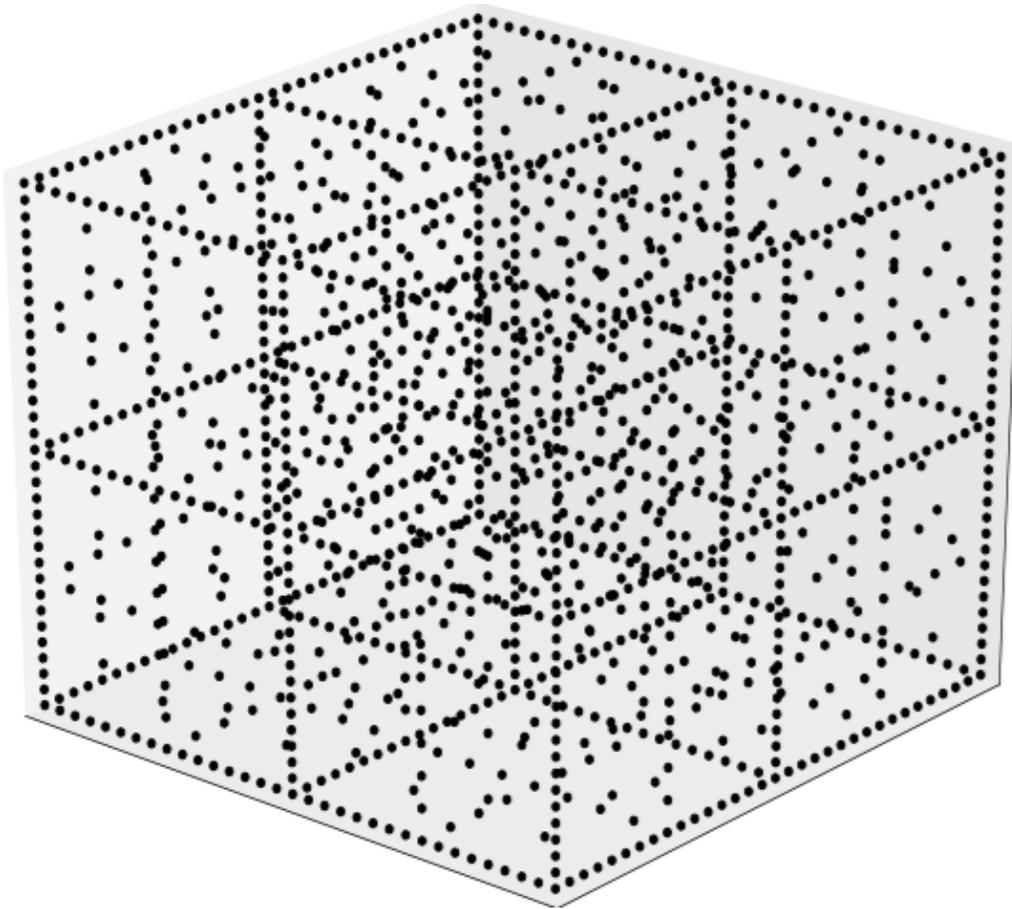
- **Sparse Grids**

- ◆ complicated data structure; but “combination” technique recovers some exploitable SIMT-like uniformity

- **$\mathcal{H}$  matrices**

- ◆ perhaps the least mathematically fragile optimal solver
- ◆ highly tunable by rank/admissibility to problem and to architecture
- ◆ somewhat complicated data structures
- ◆ *exascale features*: good SIMT structure with high arithmetic intensity at leaves

# Sparse grids for high-D spatial data



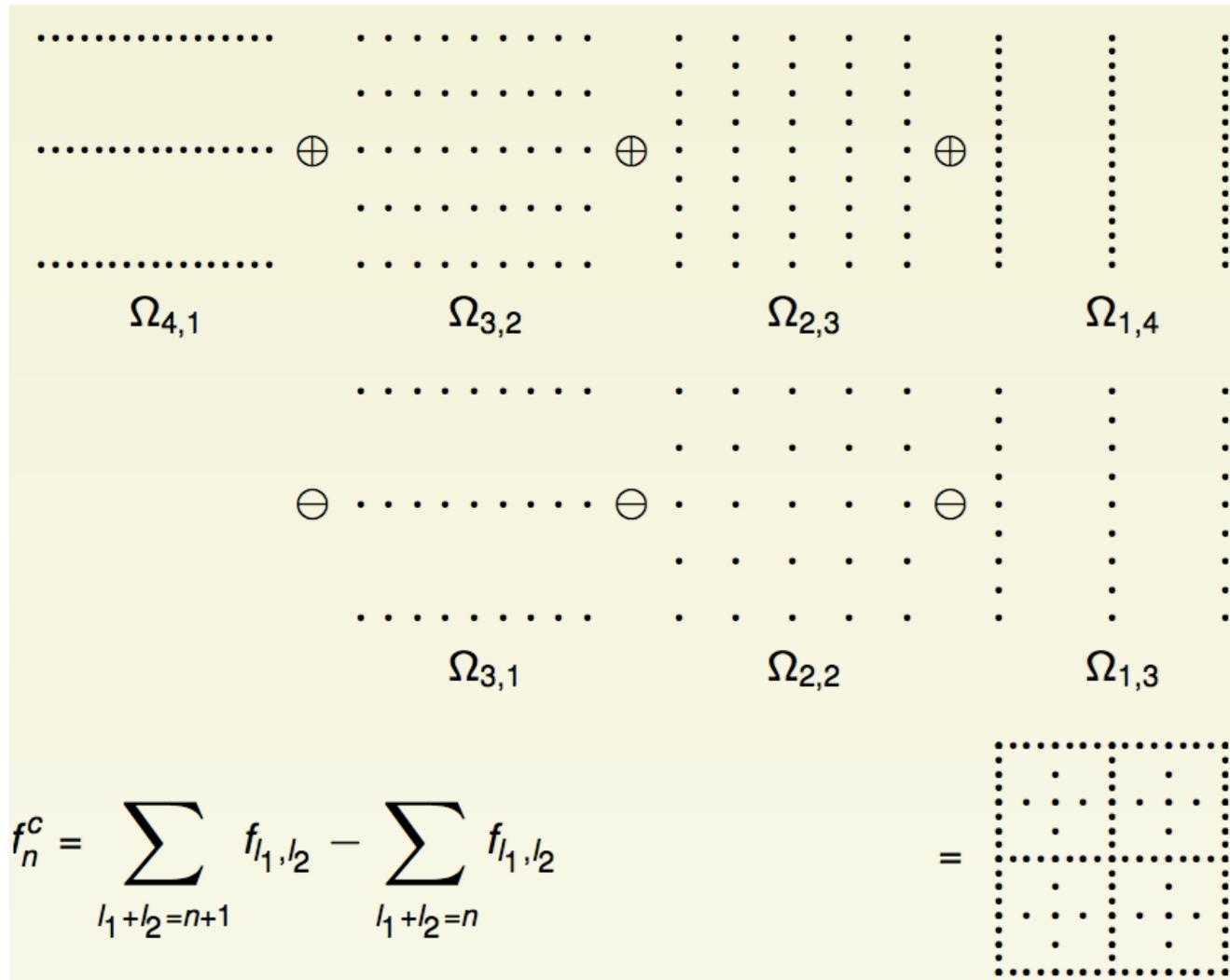
Storage complexity

$O(n^d) \rightarrow O(n \cdot (\log n)^{d-1})$   
(spatial dimension  $d$ )

Representation accuracy

$O(n^{-p}) \rightarrow O(n^{-p} \cdot (\log n)^k)$   
(order  $p$ ;  $k$  depends on  $p, d$ )

# Combination of simple coarse spaces



# Hierarchically low rank ( $\mathcal{H}$ ) matrices

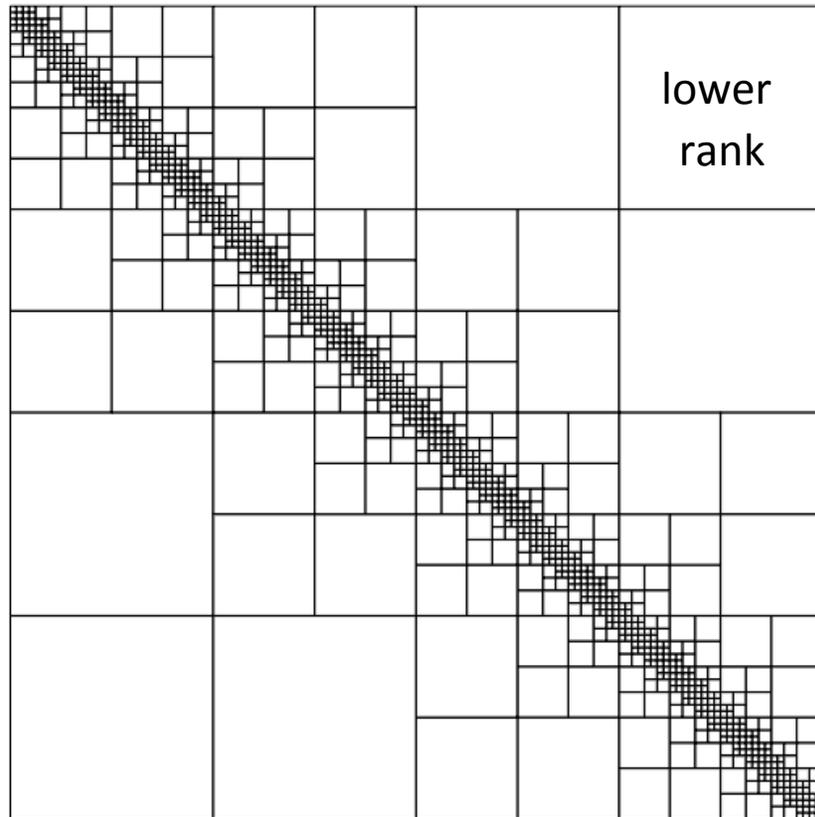
$$A = \left[ \begin{array}{ccc|ccc} 2 & -1 & & & & \\ -1 & 2 & -1 & & & \\ & -1 & 2 & & & \\ \hline & & -1 & 2 & -1 & \\ & & & -1 & 2 & -1 \\ & & & & -1 & 2 \end{array} \right] \iff = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 0 & 0 \end{bmatrix}$$

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

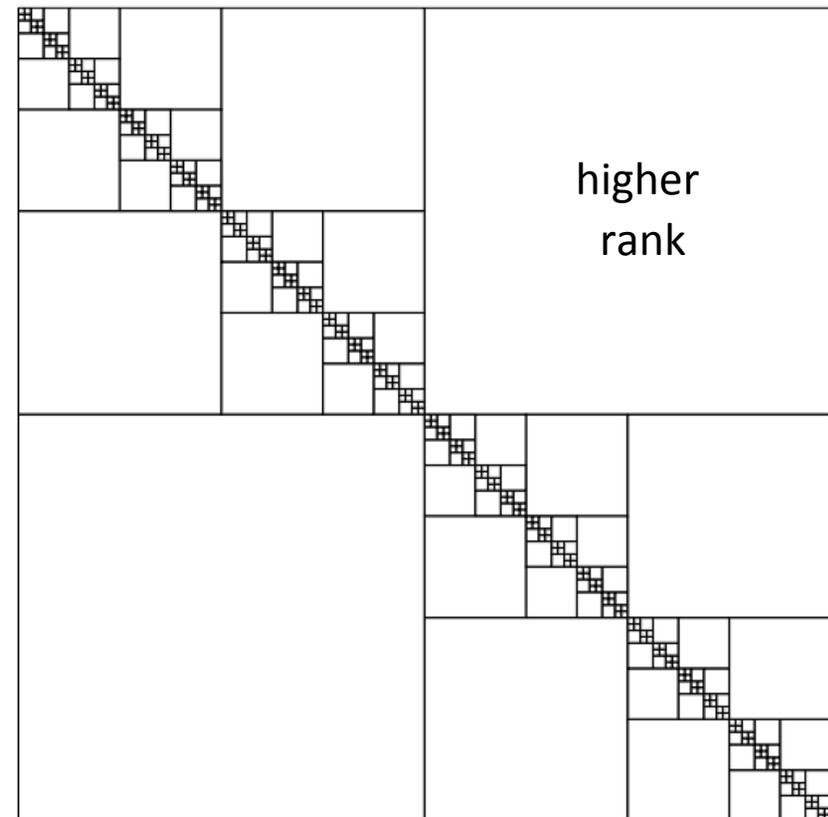
# General tool: hierarchical matrices

- [Hackbusch, 1999] : **off-diagonal blocks of typical differential and integral operators have low effective rank**
  - smoothness more critical than decay rate
- **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 in comparisons with multigrid
- **Such hierarchical representations navigate a compromise**
  - fewer blocks of larger rank (“weak admissibility”) or
  - more blocks of smaller rank (“strong admissibility”)

# Tuning: 'strong' and 'weak' admissibility



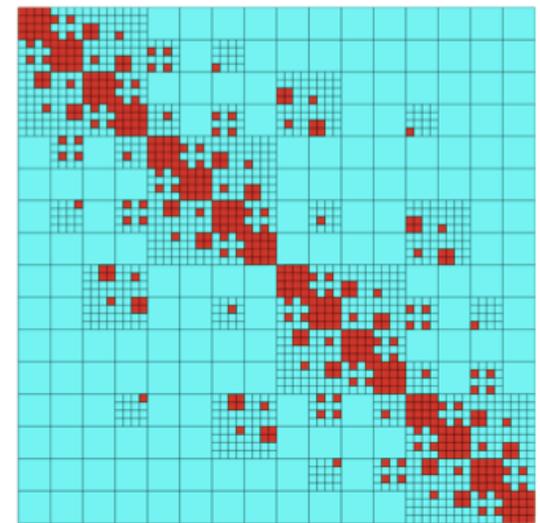
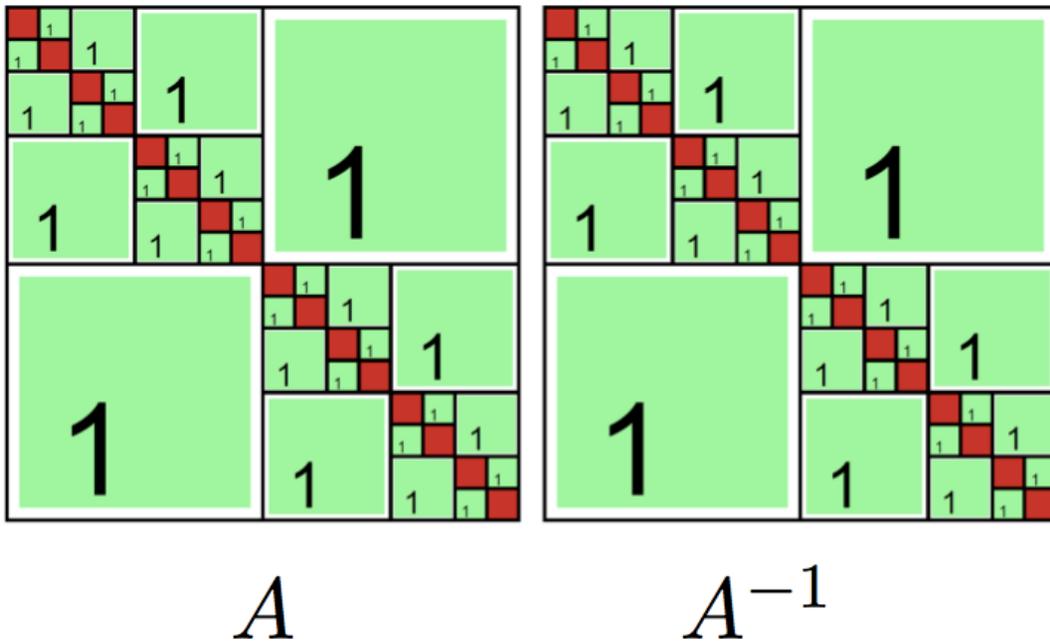
strong admissibility



weak admissibility

after Hackbusch *et al.*, 2003

# Graphical idiom for hierarchical structure



more general system

# Hierarchical matrix flavors

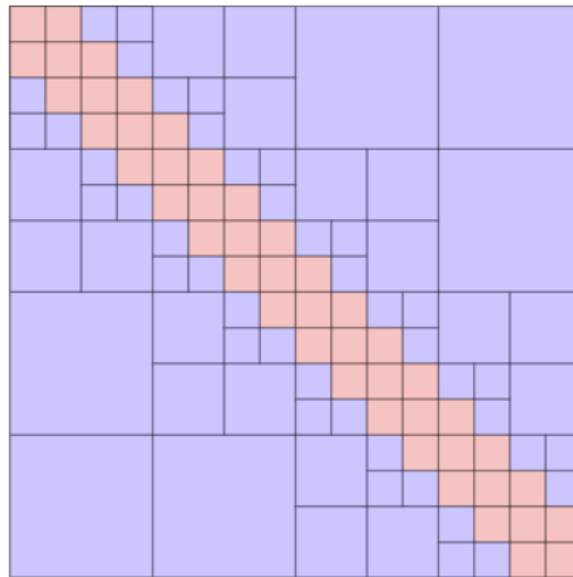
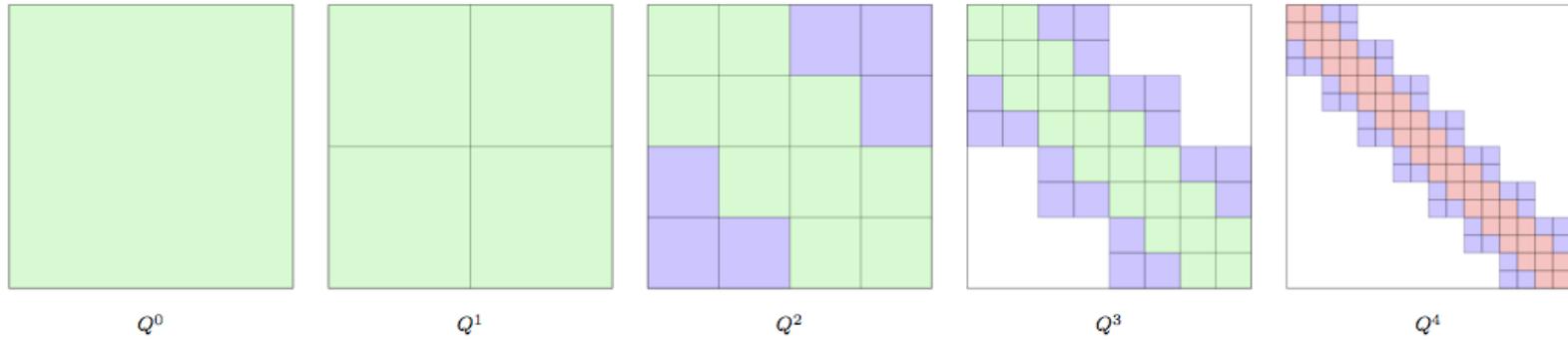
- [Hackbusch, 1999] :  $\mathcal{H}$ -matrices
- [Hackbusch & Khoromskij, 2000] :  $\mathcal{H}^2$ -matrices
- [Li *et al.*, 2009] : **hierarchically semi-separable matrices (HSS)**
- [Ho, *et al.*, 2012] : **recursive skeletonization**
- [Darve *et al.*, 2013] : **hierarchical off-diagonal low-rank matrices (HODLR)**
- [Yokota *et al.*, 2014] : **algebraic fast multipole**
- ...

# Applications in linear algebra

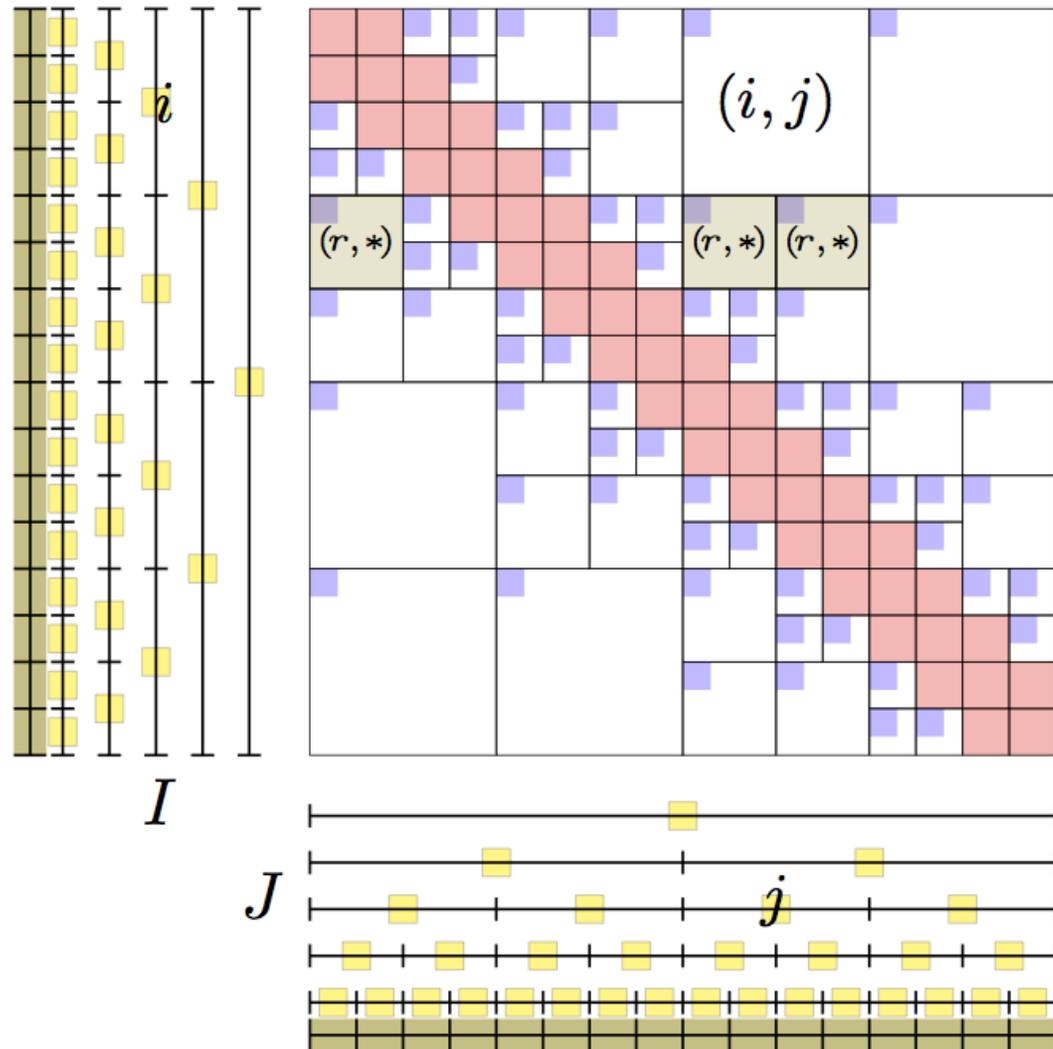
- When dense blocks arise in “routine” matrix operations, replace them with hierarchical representations
- Use high accuracy (high rank, but typically much less than full) to build “exact” solvers
- Use low accuracy (low rank) to build preconditioners
  - [Kriemann, *et al.*] : Hlib
  - [Li, *et al.*] : STRUMPACK
  - [Darve, *et al.*] : HODLR
  - [Poulson, *et al.*] : DMHM
  - [Amestoy, *et al.*] : MUMPS
  - [Ltaief, *et al.*] : HBLAS

Inversion Complexity		
	Operations	Storage
Dense	$\mathcal{O}(n^3)$	$\mathcal{O}(n^2)$
$\mathcal{H}$	$\mathcal{O}(k^2 n \log^2 n)$	$\mathcal{O}(kn \log n)$

# Hierarchical structure



# H<sup>2</sup> hierarchical structure



$$A_{ij} = U_i S_{ij} V_j^t$$

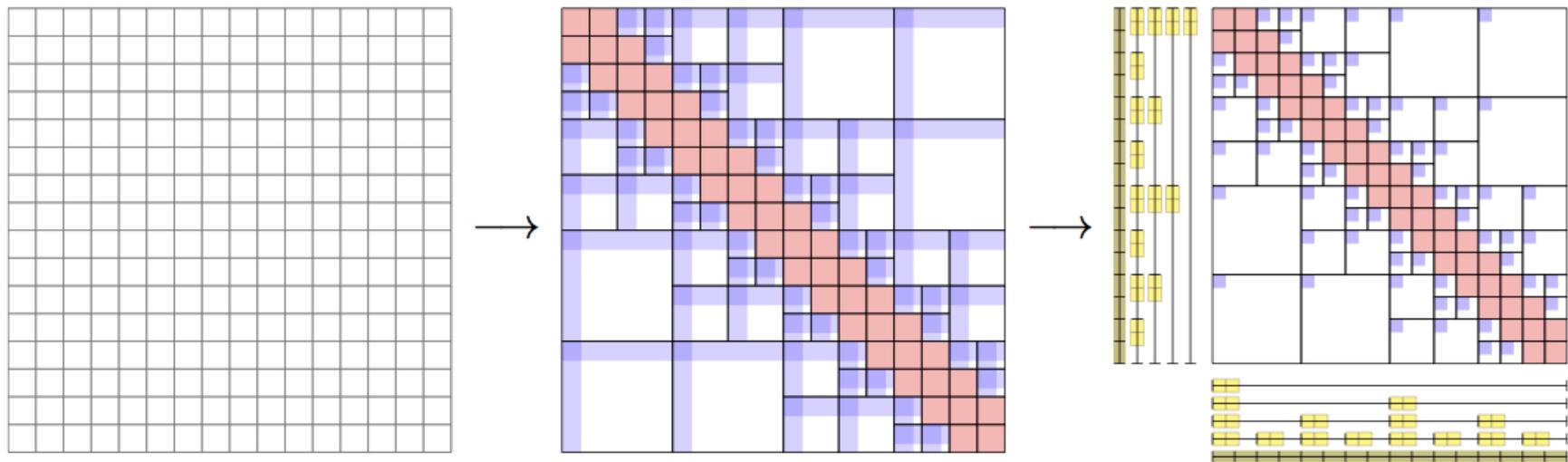
# Recursive construction of nested bases

$$\begin{bmatrix} U_i \end{bmatrix} = \begin{bmatrix} U_{i_1} & \\ & U_{i_2} \end{bmatrix} \begin{bmatrix} E_{i_1} \\ E_{i_2} \end{bmatrix}$$

## Data structure components:

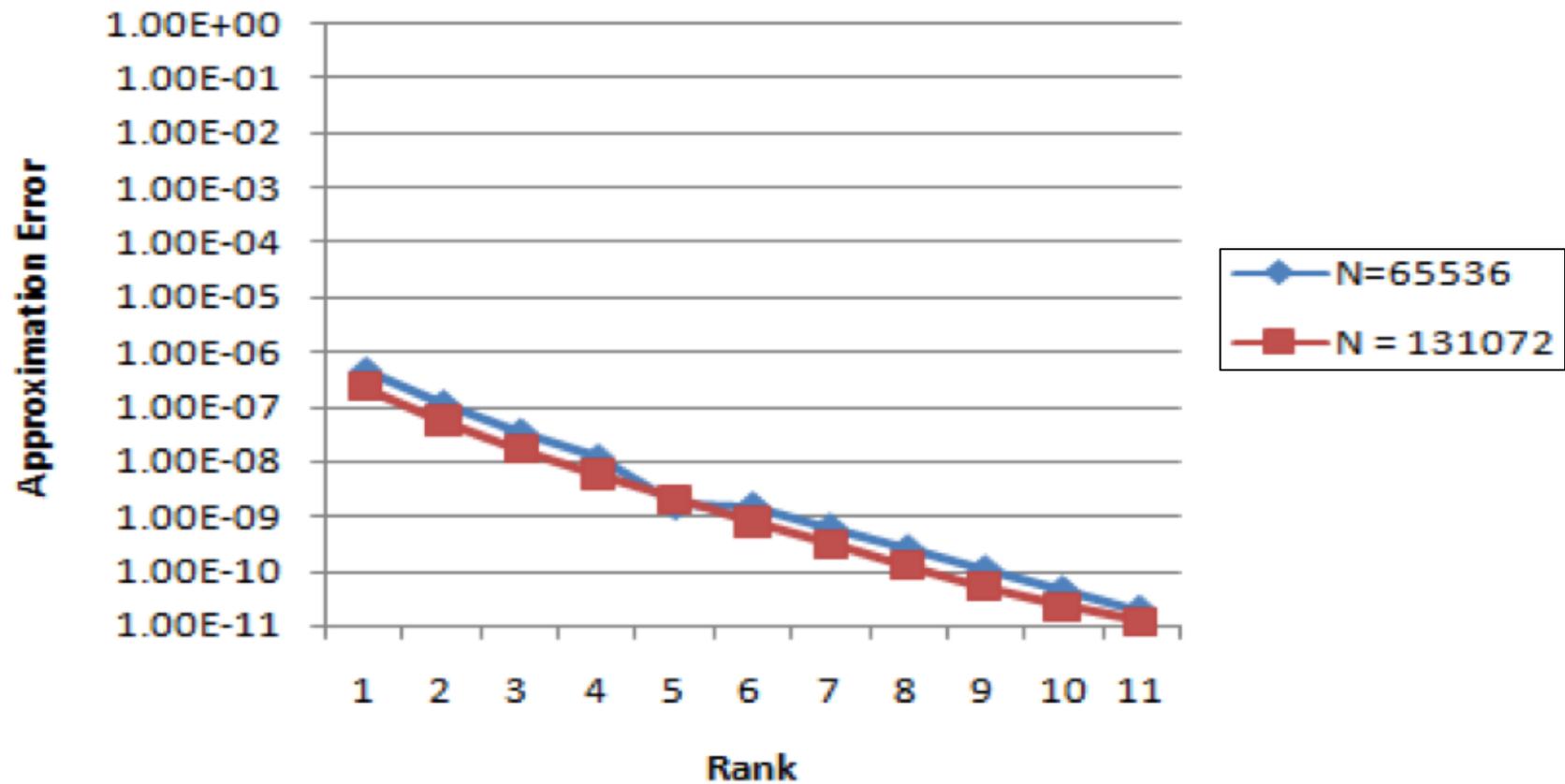
- dense blocks from original matrix
- diagonal blocks of low-rank matrices
- column bases and row bases for low-rank matrices
- maps between bases of different scales

# Two-stage compression procedure



# Accuracy vs. rank

exponential decay kernel in  $H^2$  form



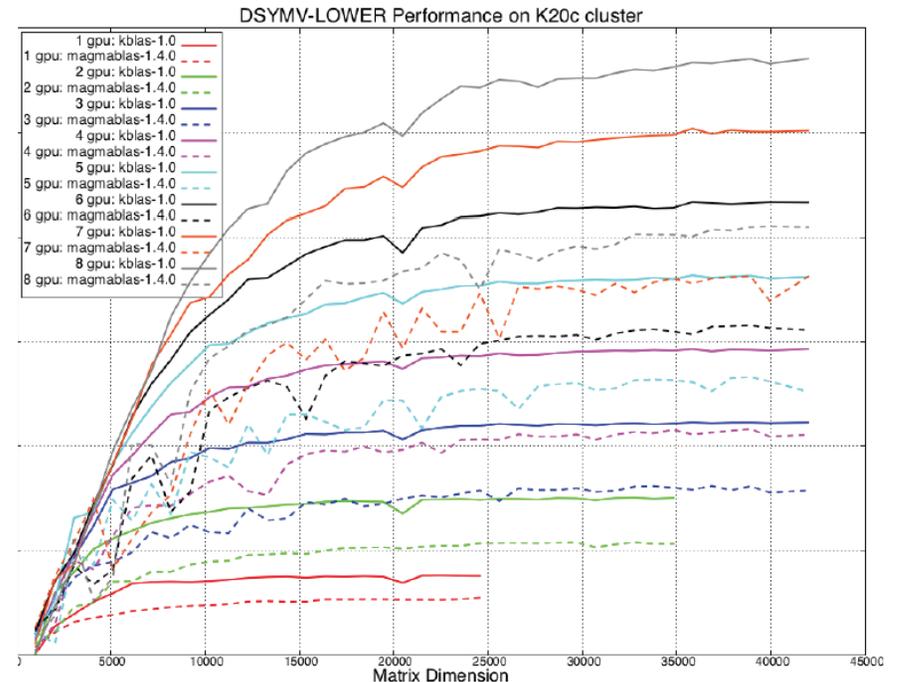
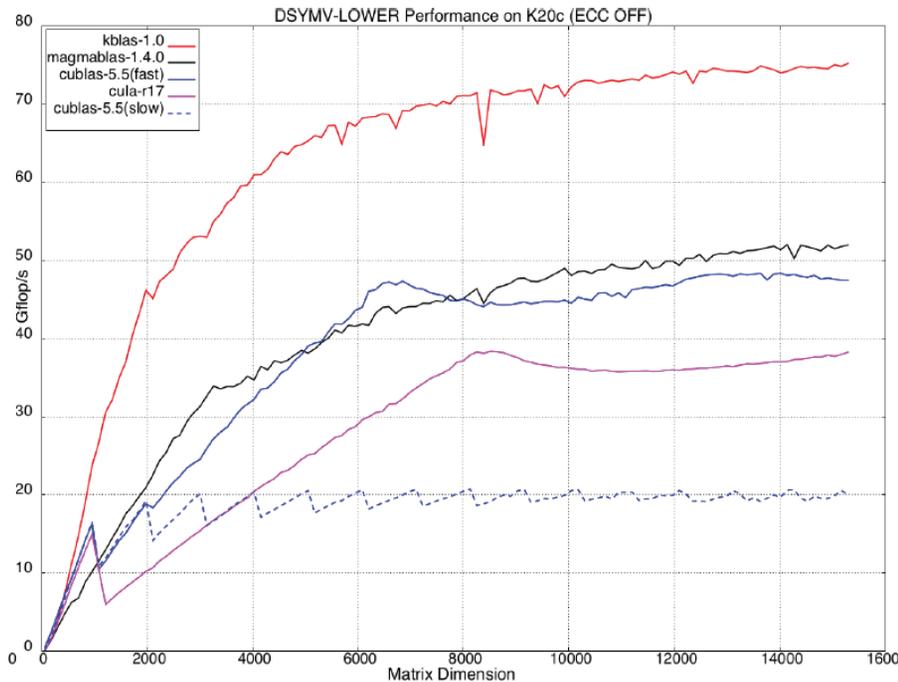
# To illustrate: stories of work in progress

- **Beginning of the “KBLAS”**
- **Beginning of the “HBLAS”**
- **Beginning of an H-based direct solver for Helmholtz**

# KBLAS: GPU implementations of dense linear algebra

- **Increase SIMT-style thread concurrency**
  - ◆ **overcome memory bandwidth limitations of the BLAS2 matrix-vector multiply,  $y = \alpha A x + \beta y$**
  - ◆ **coalesced memory accesses**
  - ◆ **double buffering**
  - ◆ **polyalgorithmic approach based on block size**

# Optimized GEMV/SYMV kernels



Single GPU and multi-GPU versions  
NVIDIA has adopted for CuBLAS 6.0



c/o A. Abdelfattah (Oak Ridge National Lab/UT)

ScalA15 | 16 Nov 2015

# “KBLAS inside”

**DEVELOPER ZONE** **CUDA TOOLKIT DOCUMENTATION** Search

**CUDA Toolkit v6.0**  
**cuBLAS**

- 1. Introduction
  - 1.1. Data layout
  - 1.2. New and Legacy cuBLAS API
  - 1.3. Example code
- 2. Using the cuBLAS API
  - 2.1. General description
    - 2.1.1. Error status
    - 2.1.2. cuBLAS context
    - 2.1.3. Thread Safety
    - 2.1.4. Results reproducibility
    - 2.1.5. Scalar Parameters
    - 2.1.6. Parallelism with Streams
    - 2.1.7. Batching Kernels
    - 2.1.8. Cache configuration
    - 2.1.9. Device API Library
  - 2.2. cuBLAS Datatypes Reference
    - 2.2.1. cublasHandle\_t
    - 2.2.2. cublasStatus\_t
    - 2.2.3. cublasOperation\_t
    - 2.2.4. cublasFillMode\_t
    - 2.2.5. cublasDiagType\_t
    - 2.2.6. cublasSideMode\_t
    - 2.2.7. cublasPointerMode\_t
    - 2.2.8. cublasAtomicsMode\_t
  - 2.3. cuBLAS Helper Function Reference
    - 2.3.1. cublasCreate()
    - 2.3.2. cublasDestroy()
    - 2.3.3. cublasGetVersion()
    - 2.3.4. cublasSetStream()

### C. Acknowledgements

NVIDIA would like to thank the following individuals and institutions for their contributions:

- Portions of the SGEMM, DGEMM, CGEMM and ZGEMM library routines were written by Vasily Volkov of the University of California.
- Portions of the SGEMM, DGEMM and ZGEMM library routines were written by Davide Barbieri of the University of Rome Tor Vergata.
- Portions of the DGEMM and SGEMM library routines optimized for Fermi architecture were developed by the University of Tennessee. Subsequently, several other routines that are optimized for the Fermi architecture have been derived from these initial DGEMM and SGEMM implementations.
- The substantial optimizations of the STRSV, DTRSV, CTRSV and ZTRSV library routines were developed by Jonathan Hogg of The Science and Technology Facilities Council (STFC). Subsequently, some optimizations of the STRSM, DTRSM, CTRSM and ZTRSM have been derived from these TRSV implementations.
- Substantial optimizations of the SYMV and HEMV library routines were developed by Ahmad Abdelfattah, David Keyes and Hatem Ltaief of King Abdullah University of Science and Technology (KAUST).

### Notices

#### Notice

ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, "MATERIALS") ARE BEING PROVIDED "AS IS." NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE.

Information furnished is believed to be accurate and reliable. However, NVIDIA Corporation assumes no responsibility for the consequences of use of such information or for any infringement of patents or other rights of third parties that may result from its use. No license is granted by implication or otherwise under any patent rights of NVIDIA Corporation. Specifications mentioned in this publication are subject to change without notice. This publication supersedes and replaces all other information previously supplied. NVIDIA Corporation products are not authorized as critical components in life support devices or systems without express written approval of NVIDIA Corporation.

#### Trademarks

NVIDIA and the NVIDIA logo are trademarks or registered trademarks of NVIDIA Corporation in the U.S. and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

#### Copyright

© 2007-2014 NVIDIA Corporation. All rights reserved.

This product includes software developed by the Syncro Soft SRL (<http://www.sync.ro/>).

ScalA15 | 16 Nov 2015

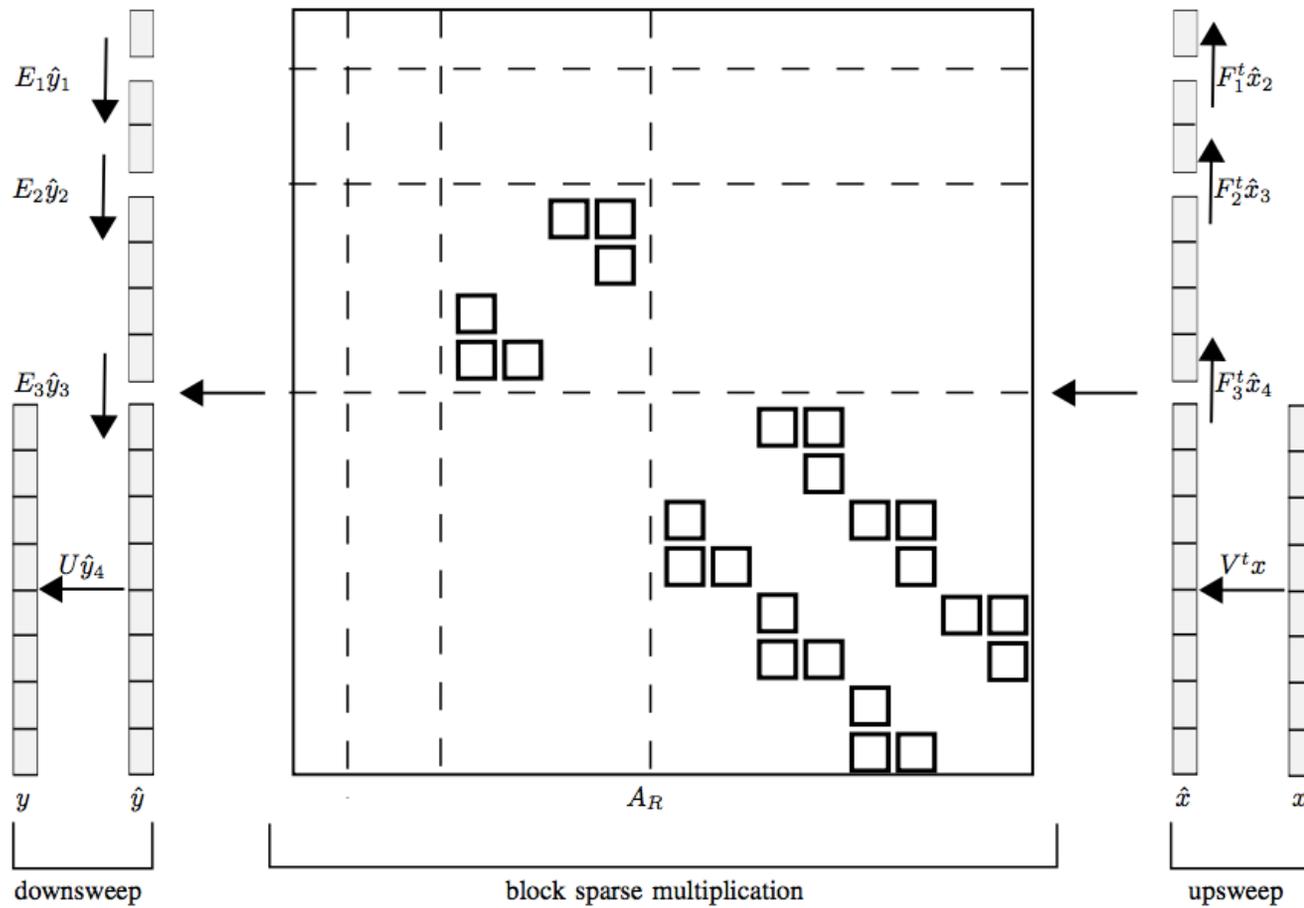
# HBLAS: GPU implementations of hierarchical linear algebra

- **Build on KBLAS to implement BLAS with hierarchically low rank matrix data structures on GPUs**
  - ◆ **matrix-vector multiply,  $y = Ax$**
  - ◆ **tree-based scheme, following fast multipole**

$$y = \left( \sum_{(i,j) \in D} A_{ij} \right) x + \left( \sum_{(i,j) \in L} U_i S_{ij} V_j^t \right) x = \underbrace{\sum_{(i,j) \in D} A_{ij} x_j}_{\text{Dense mat-vecs operations}} + \underbrace{\sum_{i \in I} U_i \sum_{(i,j) \in L} S_{ij} \underbrace{V_j^t x}_{\text{Upsweep}}}_{\text{Coupling phase}}_{\text{Downsweep}}$$

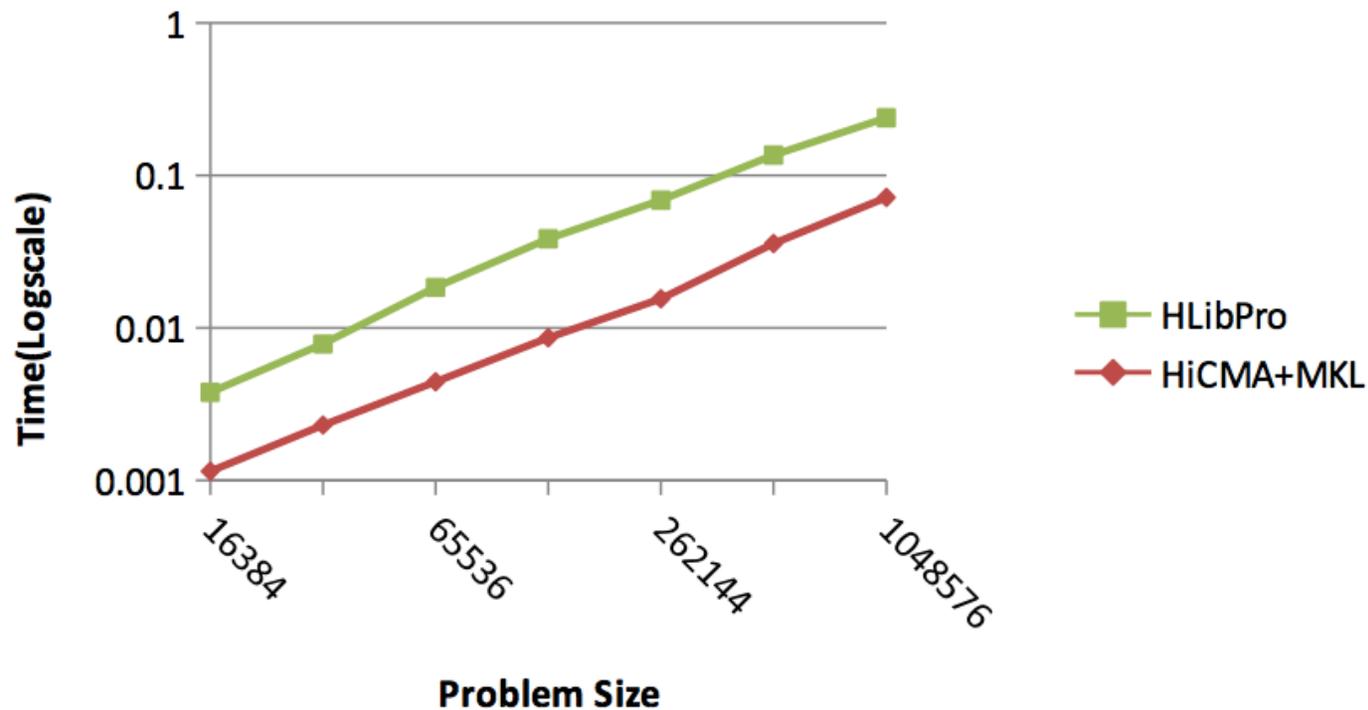


# Upsweep/downsweep phases



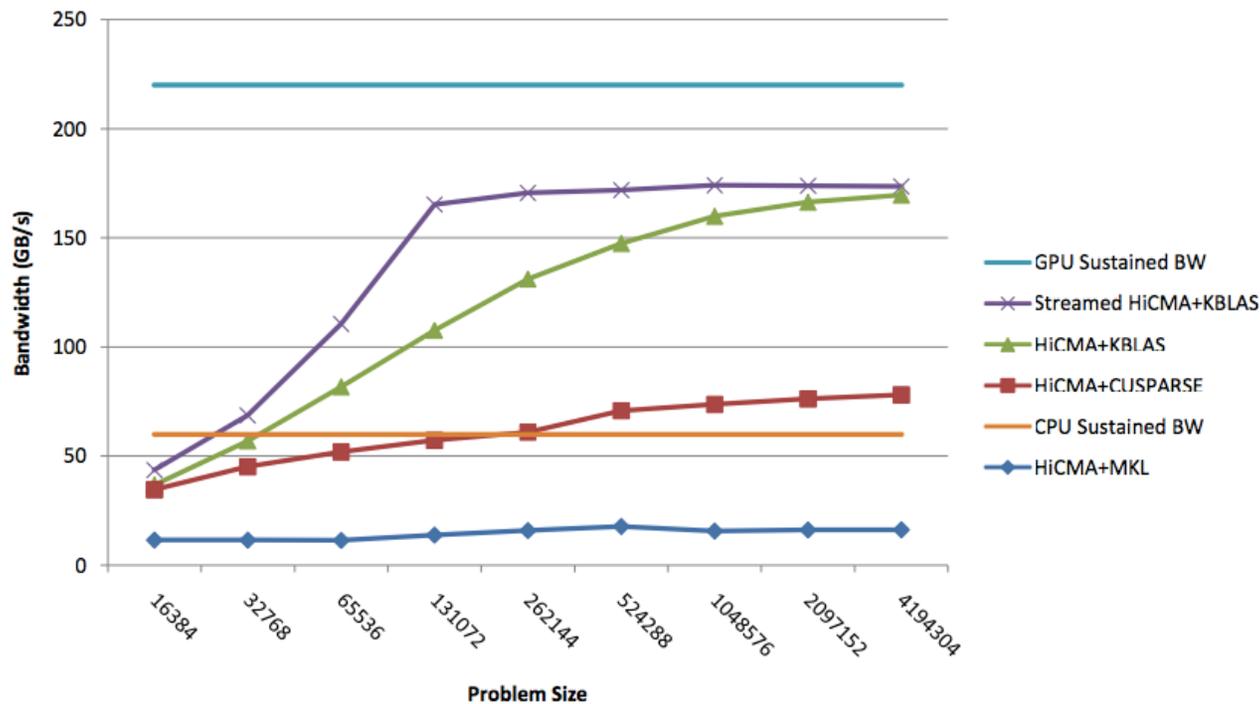
# Comparison with HlibPro

- **Problem: 1D integral equation (fully dense)**
- **Hardware**
  - ◆ **20-core Intel Xeon (2.8 GHz)**



# Preliminary GPU version

- Problem: 1D integral equation (fully dense)
- Hardware
  - ◆ Tesla K40m NVIDIA GPU



# Accelerated Cyclic Reduction

- **Build on HBLAS to convert cyclic reduction to an optimal complexity solver for non-constant coefficient operators**
  - ◆ **implement on multicore to compare against the full-featured HlibPro library of R. Kriemann until HBLAS is ready**
  - ◆ **compare with multigrid**

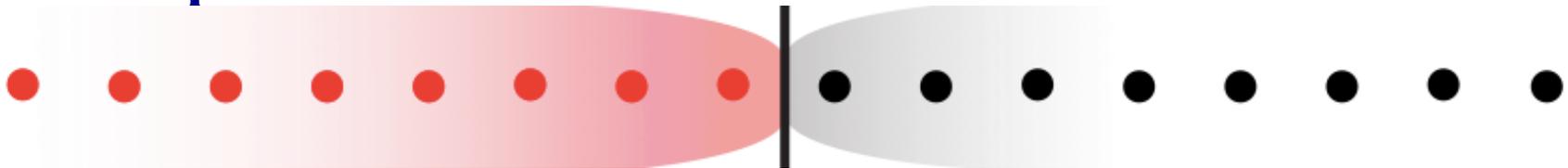


# Cyclic reduction complementary to nested dissection

1D mesh



First step of nested dissection



First step of cyclic reduction



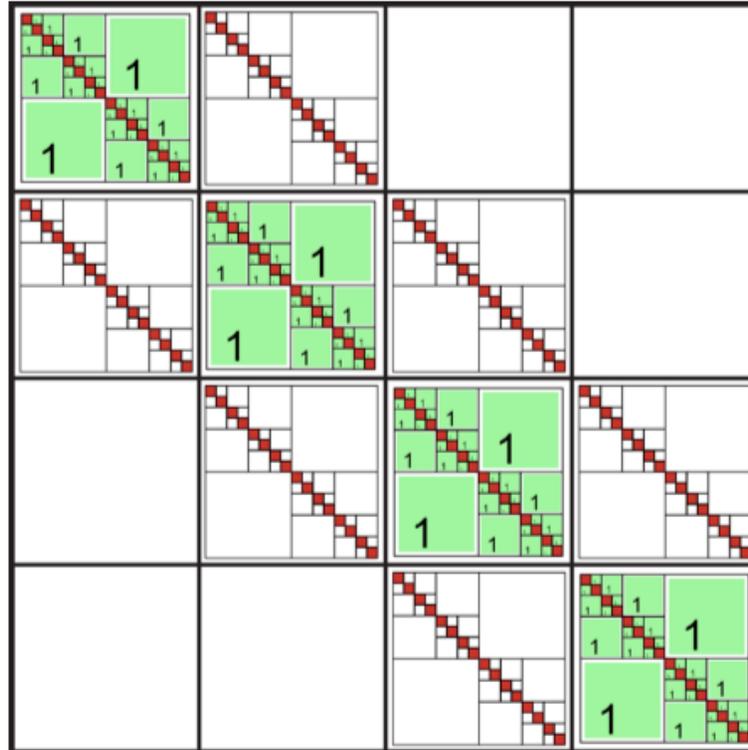
# Preservation of tridiagonal structure on Schur complementation

$$\begin{aligned}
 A &= \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \\
 &= \begin{bmatrix} x & x & & & \\ x & x & x & & \\ & x & x & & \\ & & & & \\ & & & & \end{bmatrix}
 \end{aligned}$$

$$\begin{aligned}
 S &= A_{22} - A_{21}A_{11}^{-1}A_{12} \\
 &= \begin{bmatrix} x & x & \\ x & x & x \\ & x & x \end{bmatrix} - \begin{bmatrix} x & \\ & x \end{bmatrix} \begin{bmatrix} x & x & x \\ x & x & x \\ x & x & x \end{bmatrix} \begin{bmatrix} x \\ & x \end{bmatrix} \\
 &= \begin{bmatrix} x & x & \\ x & x & x \\ & x & x \end{bmatrix} - \begin{bmatrix} x & \\ & x \end{bmatrix} \begin{bmatrix} x & \\ x & \\ x & \end{bmatrix} \\
 &= \begin{bmatrix} x & x & \\ x & x & x \\ & x & x \end{bmatrix} - \begin{bmatrix} x & \\ & x \end{bmatrix} \\
 &= \begin{bmatrix} x & x & \\ x & x & x \\ & x & x \end{bmatrix}
 \end{aligned}$$



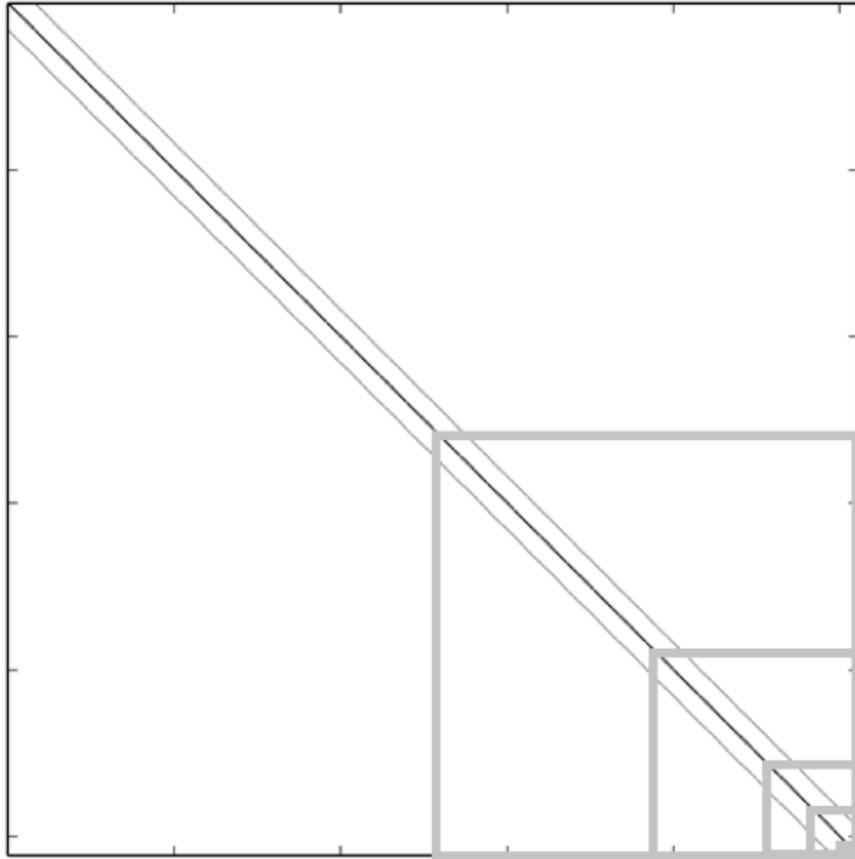
# Block structure in 2D



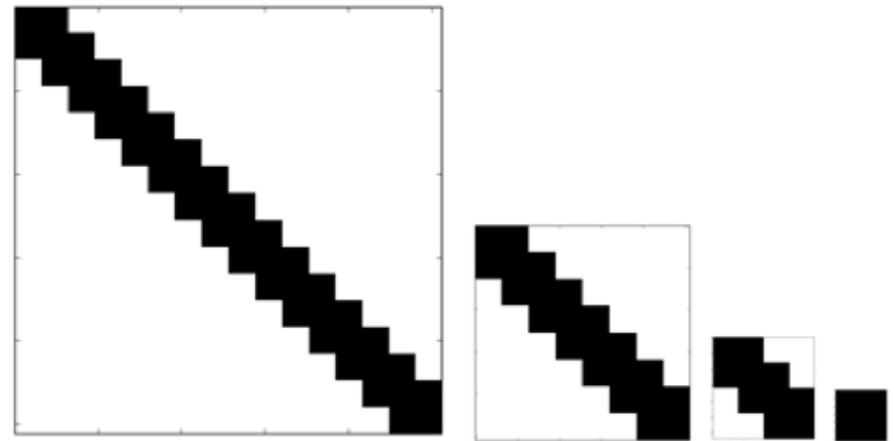
**Issue: diagonal block inversion is non-optimal when FFT does not apply**

**Enter hierarchical matrix arithmetic — how will rank grow?**

# Block recursive application



	Operations	Storage
CR	$\mathcal{O}(N^2)$	$\mathcal{O}(N^{1.5} \log N)$
ACR	$\mathcal{O}(N \log^2 N)$	$\mathcal{O}(N \log N)$



# Pseudocode

---

**Algorithm 1** Accelerated Cyclic Reduction (ACR)

---

- 1:  $n = 2^q, q \geq 2$
  - 2: Block-wise low-rank approximation of  $A$ :  $H^{(0)} = A$
  - 3: Set-up right-hand side  $f$
  - 4: **for**  $i = 1$  **to**  $q$  **do**
  - 5:      $H^{(i)} = PAP^T$
  - 6:      $H^{(i+1)} = H_{22}^{(i)} \ominus H_{21}^{(i)} \otimes \mathcal{H}\text{inverse}(H_{11}^{(i)}) \otimes H_{12}^{(i)}$
  - 7:      $f^{(i+1)} = b_{\text{odd}} - H_{21}^{(i)} \otimes \mathcal{H}\text{inverse}(H_{11}^{(i)}) \otimes b_{\text{even}}$
  - 8:      $u_{\text{odd}} = \mathcal{H}\text{inverse}(H_{11}^{(i+1)}) f^{(i+1)}$
  - 9: **end for**
  - 10: Perform back-substitution
  - 11: Permute solution vector back to natural ordering
-

# Numerical experiments

**Second-order finite difference discretization with homogeneous BCs on the unit square**

$$-\nabla \cdot \kappa(\vec{x}) \nabla u = f(\vec{x})$$

$$A = \text{tridiag}(E_i, D_i, F_i) = \begin{bmatrix} D_1 & F_1 & & & & \\ E_2 & D_2 & F_2 & & & \\ & \ddots & \ddots & \ddots & & \\ & & E_{n-1} & D_{n-1} & F_{n-1} & \\ & & & E_n & D_n & \end{bmatrix}$$

# Numerical experiments

## 2D experiments comparing ACR with H-LU and AMG

- Same forcing function and homogeneous boundary conditions:

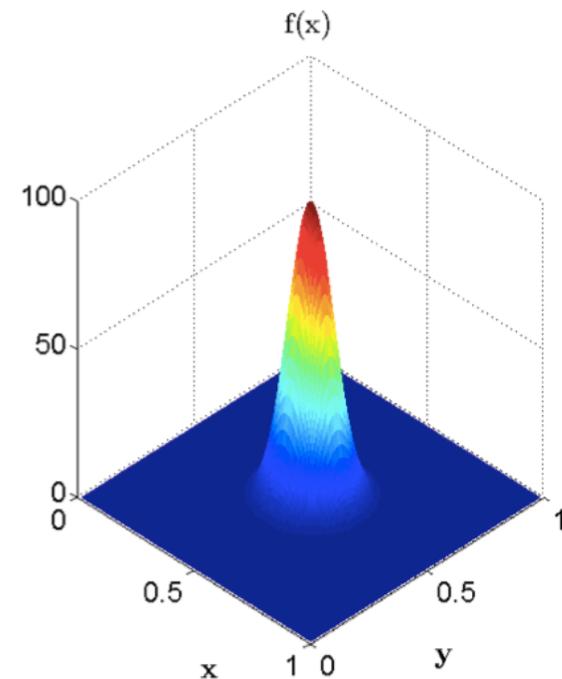
$$f(\mathbf{x}) = 100e^{-100((x-0.5)^2+(y-0.5)^2)}$$

- Both codes attempt a **direct solution** relative to the discretization error, using rank truncation:

$$\|u - \hat{u}^h\|_2 \sim \frac{\|Ax - b\|_2}{\|b\|_2}$$

Both codes are compiled with **icc15** and **O2** optimization enabled. In the same computer system.

- Benchmark is against the  $\mathcal{H}$ -LU multi-core implementation of **HLIBPro** by R. Kriemann et al.



# Hardware architecture

---

Metric	Westmere Xeon X5650	Ivy Bridge Xeon E5-2680 v2	Haswell Xeon E5-2699 v3	AMD Opteron 6376
Cores	12	20	36	64
Clock frequency (GHz)	2.66	2.9	2.3	2.6
L3 Total size (KB)	12,288	25,600	46,080	12,288
Cache/core (KB)	1,024	1,280	1,280	192
Bandwith (GB/s)	23.3	57.7	94.46	45.8

---

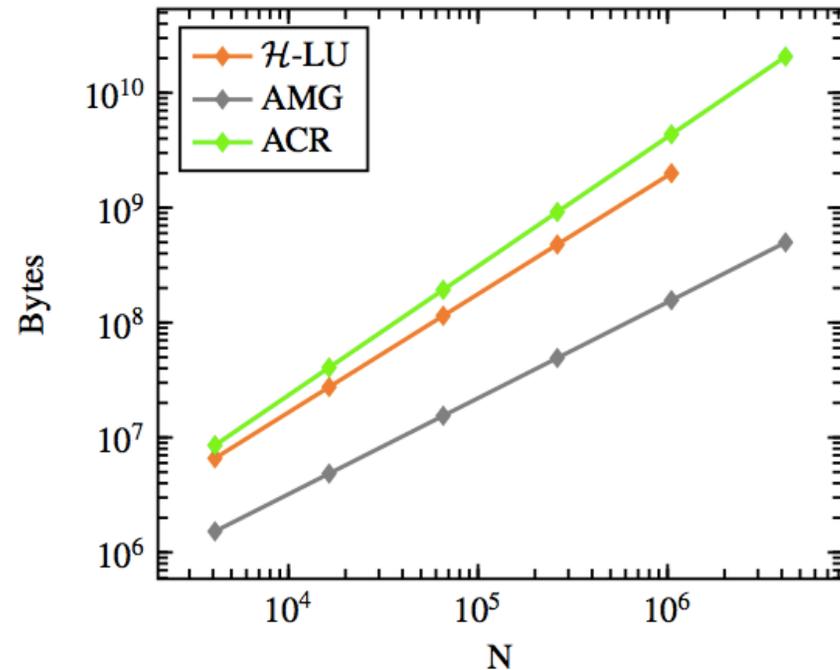
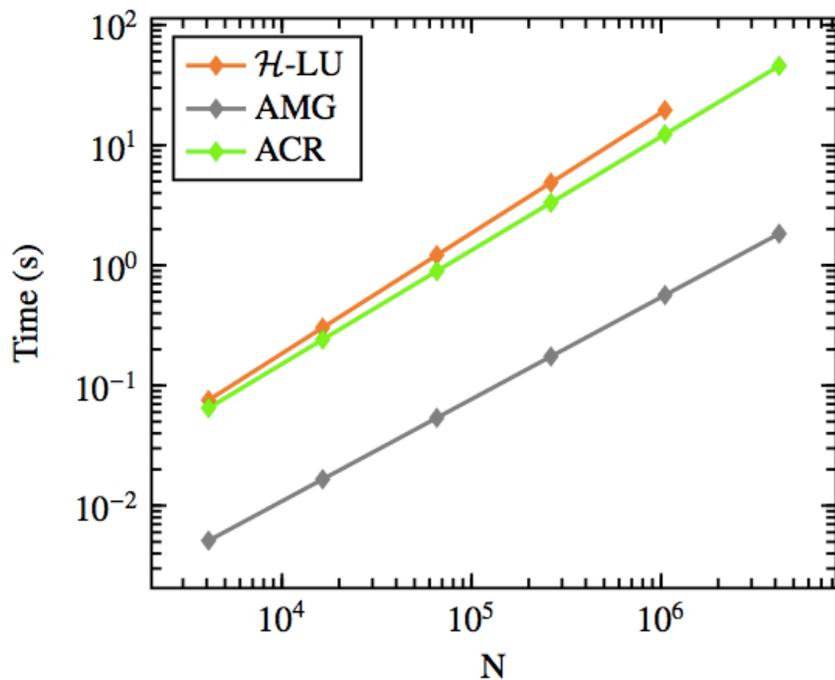
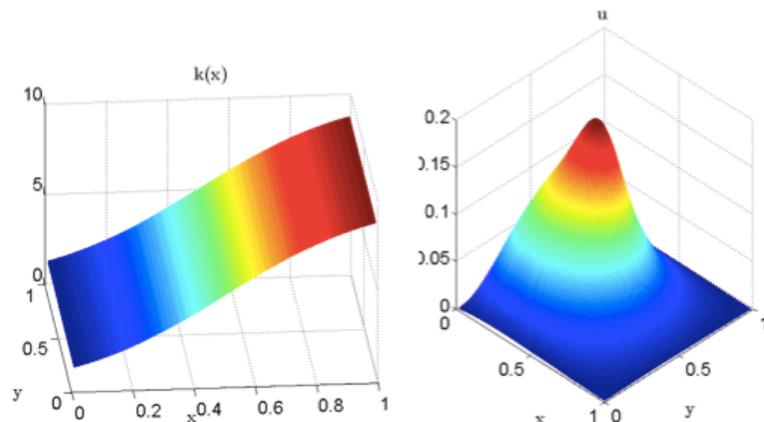
# Smoothly varying diffusivity

12 cores of Westmere

$$-\nabla \cdot \kappa(\mathbf{x}) \nabla u = f(\mathbf{x})$$

$$\kappa(x, y) = \kappa^- + (\kappa^+ + \kappa^-) \frac{1 + \tanh\left(\frac{x-0.5}{\epsilon}\right)}{2}$$

$$\kappa^- = 0.1, \quad \kappa^+ = 10, \quad \epsilon = 0.5$$

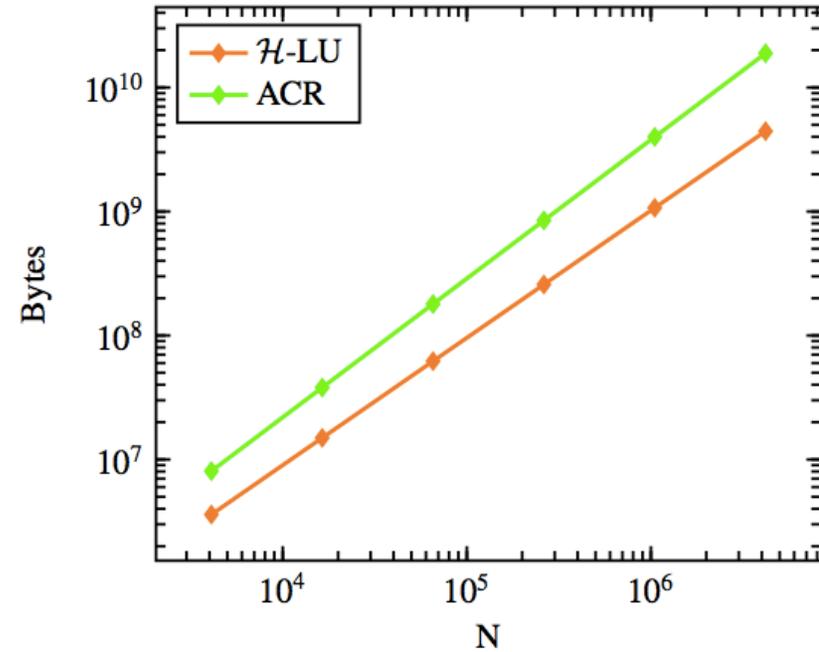
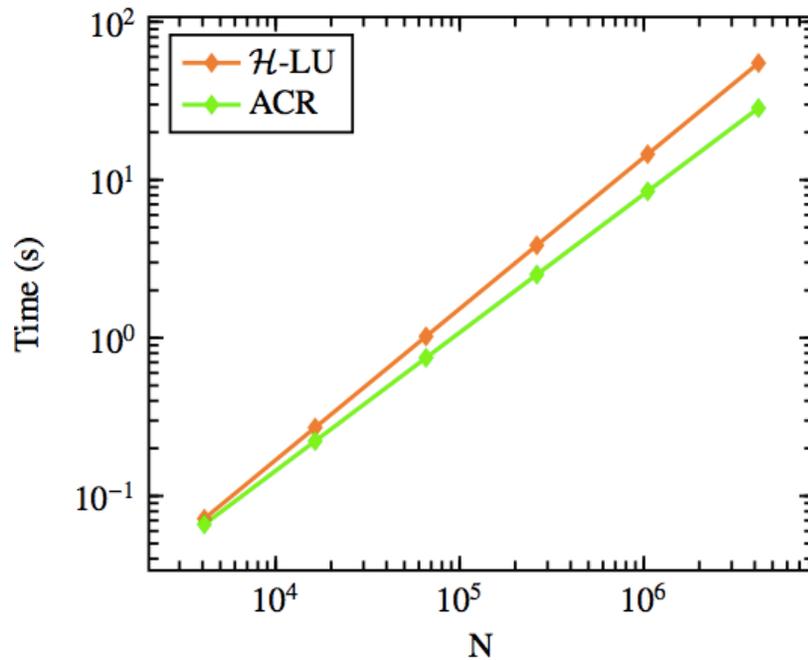
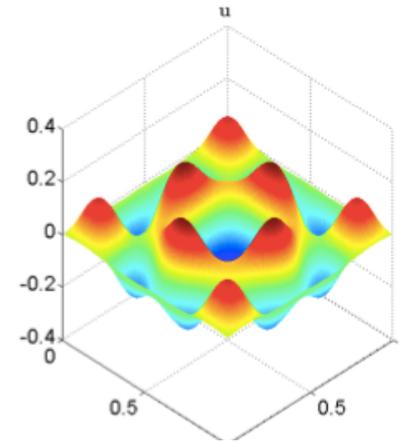


# Helmholtz: decreasing $h$ , fixed $k$

12 cores of Westmere

$$\nabla^2 u + k^2 u = f(\mathbf{x})$$

$$k = 20.$$



# Helmholtz: fixed $h$ , increasing $k$

12 cores of Westmere

ACR

$n$	$k$	Error	Memory (Bytes)	Time (sec)
1,024	1	$2.41 \times 10^{-6}$	$3.81 \times 10^9$	8.22
1,024	25	$3.30 \times 10^{-6}$	$3.81 \times 10^9$	8.32
1,024	50	$1.96 \times 10^{-6}$	$3.81 \times 10^9$	8.38
1,024	100	$4.14 \times 10^{-6}$	$3.83 \times 10^9$	8.62

H-LU

$n$	$k$	Error	Memory (Bytes)	Time (sec)
1,024	1	$8.26 \times 10^{-6}$	$1.07 \times 10^9$	14.35
1,024	25	$5.10 \times 10^{-6}$	$1.04 \times 10^9$	14.48
1,024	50	$9.12 \times 10^{-6}$	$1.03 \times 10^9$	14.04
1,024	100	$1.41 \times 10^{-6}$	$1.04 \times 10^9$	14.52

4x memory      half the time

# Helmholtz: fixed $kh$

12 cores of Westmere

ACR

$n$	$k$	Error	Memory (Bytes)	Time (sec)
64	5	$5.72 \times 10^{-6}$	$5.12 \times 10^6$	0.09
128	10	$7.41 \times 10^{-6}$	$2.21 \times 10^7$	0.33
256	20	$4.71 \times 10^{-6}$	$9.53 \times 10^7$	1.07
512	40	$7.24 \times 10^{-6}$	$4.04 \times 10^8$	3.67
1,024	80	$7.87 \times 10^{-6}$	$1.72 \times 10^9$	14.37

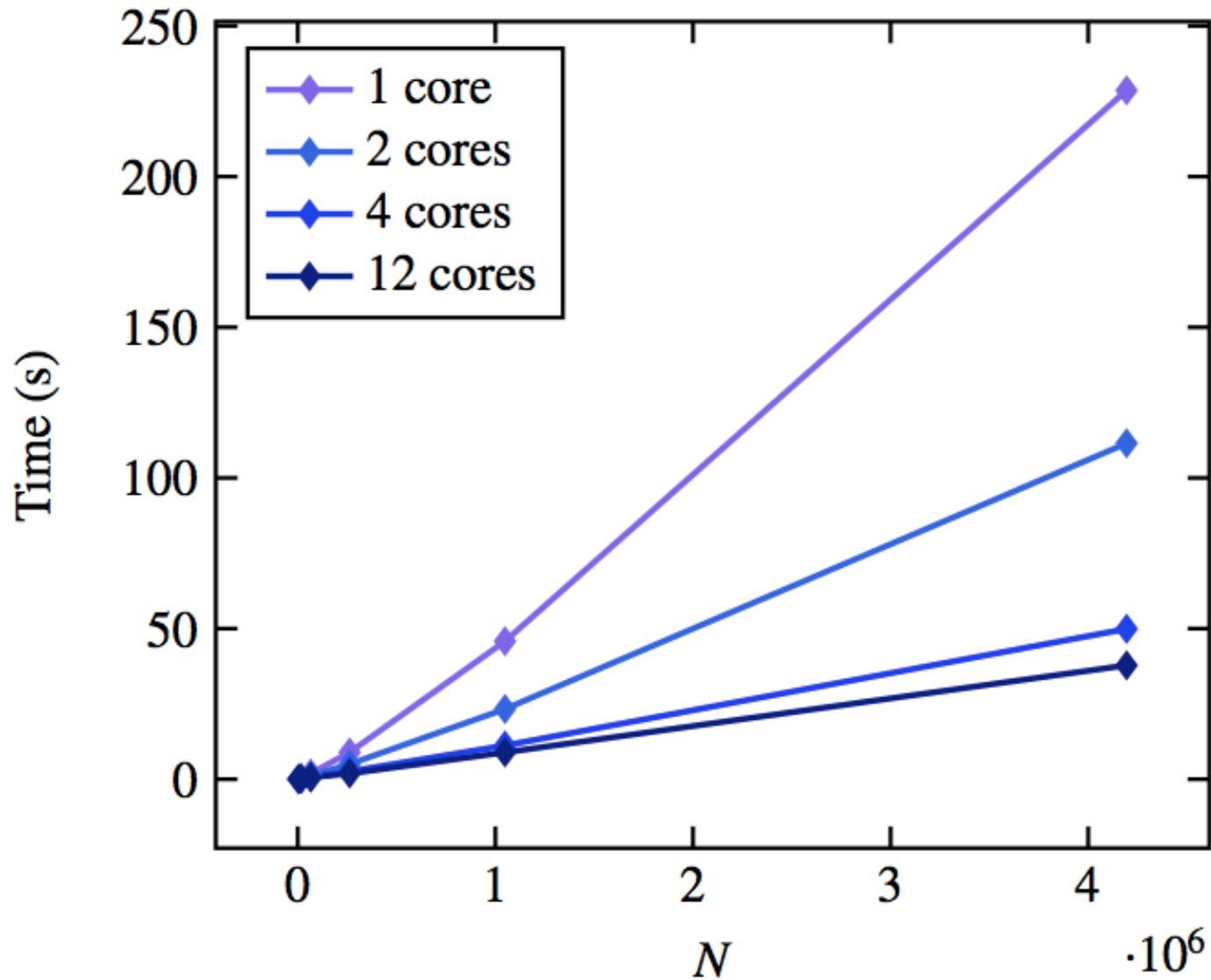
H-LU

2x memory

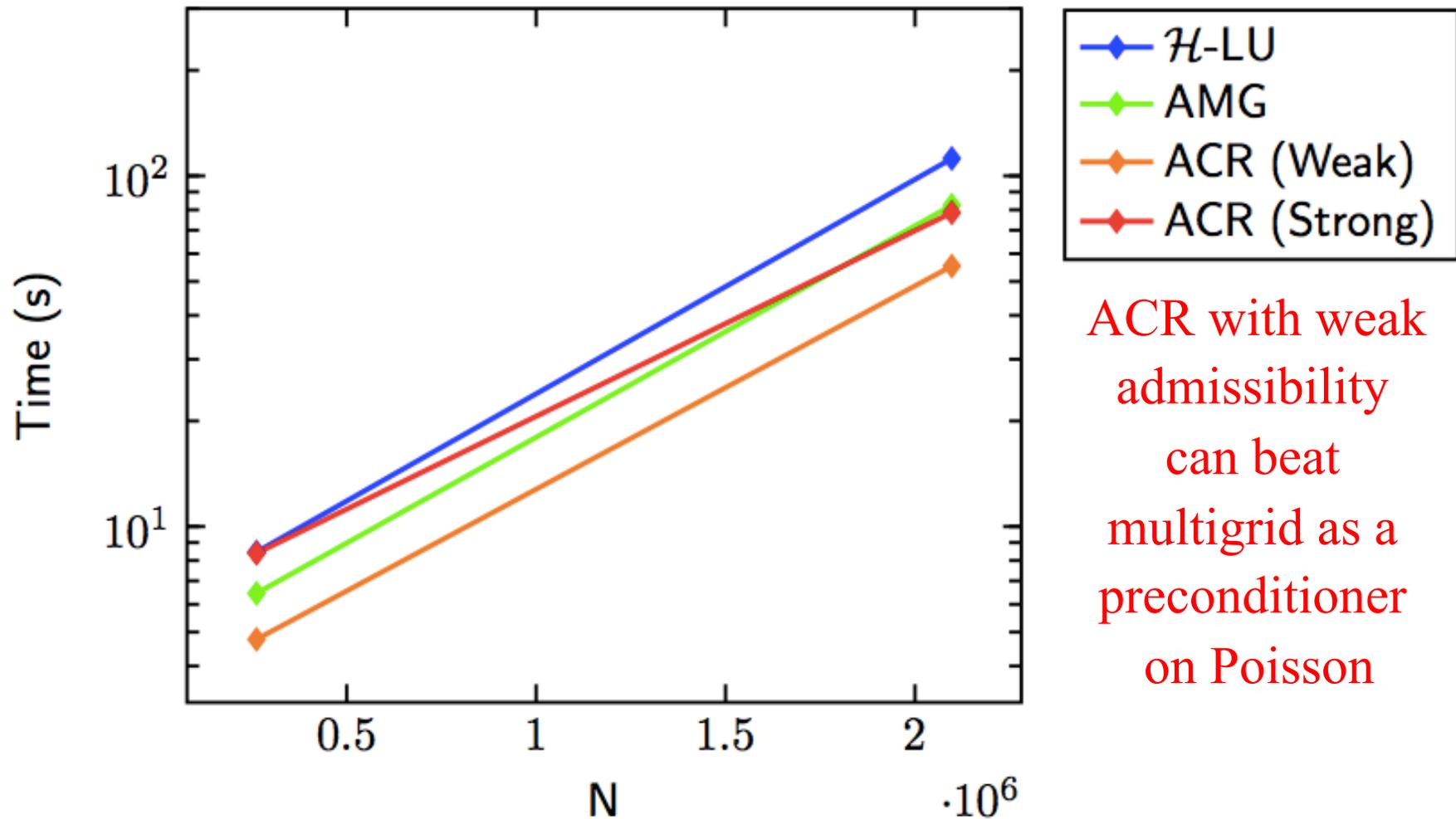
same time

$n$	$k$	Error	Memory (Bytes)	Time (sec)
64	5	$7.80 \times 10^{-6}$	$3.64 \times 10^6$	0.08
128	10	$4.03 \times 10^{-6}$	$1.57 \times 10^7$	0.21
256	20	$4.12 \times 10^{-6}$	$6.20 \times 10^7$	0.94
512	40	$7.42 \times 10^{-6}$	$2.56 \times 10^8$	3.88
1,024	80	$9.12 \times 10^{-6}$	$1.04 \times 10^9$	14.61

# Near linear complexity in problem size



# Radically truncated-rank ACR as preconditioner for 3D Poisson

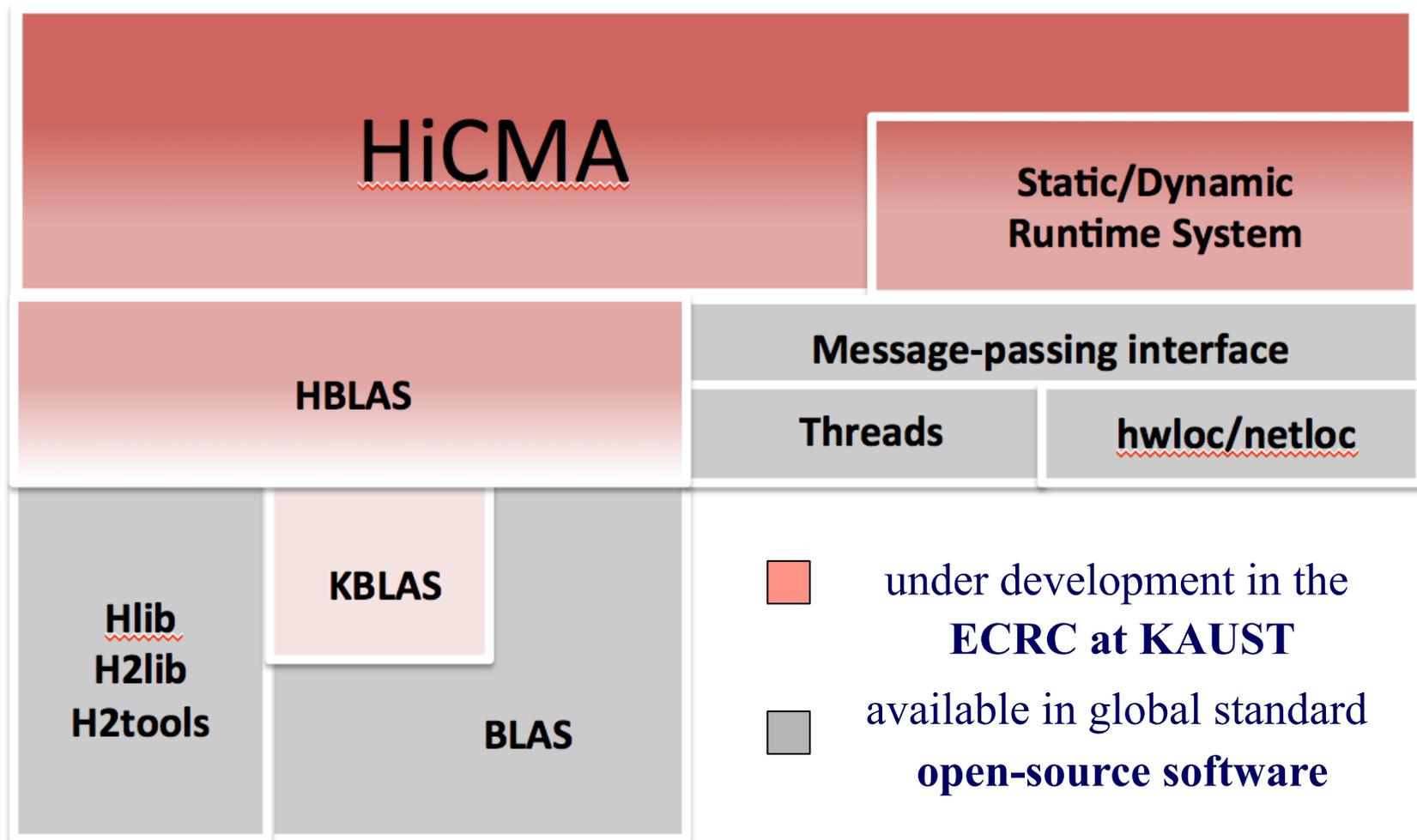


ACR with weak  
admissibility  
can beat  
multigrid as a  
preconditioner  
on Poisson

# Comments

- **ACR is not in competition with other hierarchical matrix libraries; it is a customer**
  - algorithmic outer wrapper
  - may unlock additional exploitable structure for concurrency and reuse
- **Internal H arithmetic engine is modular**
  - here, HLibPro<sup>®</sup>
  - ride emerging software for each emerging hardware environment

# Hierarchical Computations on Manycore Architectures (HiCMA – Arabic for “wisdom”)



# KAUST's Extreme Computing Research Center

The image is a collage representing KAUST's Extreme Computing Research Center. It features several key elements:

- Server Racks:** A photograph of a server room with rows of black server racks. The racks are labeled with "CRAY" and "SHARAD" logos.
- Network Diagram:** A diagram showing a network topology with nodes and connections. Labels include "P2M", "M2M", "M2L", "L2L", and "L2P".
- Flowchart:** A process flowchart for performance optimization. It starts with "Initialize: allocate arrays, etc." and "Machine info.: Threads #, cache block size range, etc.". It then branches into "User selected parameters" and "Generate all possible threads ( $T_x, T_y, T_z$ ) combinations". A loop "For each ( $T_x, T_y, T_z$ )" leads to "Find best  $D_w$  using hill climbing search" and "Find best  $W_w$  using hill climbing search". A "Solution function" block is used for both. The process concludes with "Choose best operating point" and "Best params.:  $D_w, W_w, T_x, T_y,$  and  $T_z$ ".
- Neural Network:** A diagram of a neural network with multiple layers of nodes (red, blue, yellow) and connecting lines.
- Data Visualization:** A matrix visualization with a grid of colored cells (red, blue, yellow) and axes labeled  $i, j$  and  $r, s$ .
- Text Banner:** A teal banner at the bottom right with the text "Embracing the opportunities of exascale".

# Mapping algorithms to architecture challenges

Student	Algorithm/Kernel	Reduce Synchronization	Increase Intensity	Increase Concurrency	Algorithmic Resilience	New Capabilities
Ahmad	BLAS2		X	X		
Ali	BLAS2/3		X	X		
Amani	Multigrid	X		X	X	
Chengbin	Non-neg. mat. fact.			X		X
Dalal	Eigen/SVD		X	X		
Gustavo	H-Schur		X	X		
Huda	FMM precondition.	X	X	X		
Lulu	Nonlinear precondition.	X			X	X
Mohammed	Unstruct. PDEs	X		X		
Mustafa	FMM	X	X	X		
Noha	BEM	X	X	X		
Tareq	Stencil eval.		X	X		
Wajih	H-BLAS		X	X		

PhD thesis topics in the ECRC all attempt to impact scaling of important algorithmic infrastructure to the exascale

# Mapping algorithms to application challenges

TOOLS	SIMULATION						ANALYTICS	
	Fluid dynamics	Adaptive Optics	Chemistry	Electro magnetics	Reservoirs	Seismic Imaging	Spatial Statistics	Graph Algorithms
Dense Linear/Eigen Solvers		Hatem, Ahmad, Ali	Hatem, Dalal				Hatem	
Sparse Nonlinear/Iter Solvers	Matteo, Mohamed				Stefano, Amani, Lulu	Stefano		Chengbin
Fast Multipole	Rio, Huda		Rio, Mustafa	Rio, Noha				
Hierarchical Matrices				George	George, Gustavo	George, Wajih	Alex, George	
Stencil Evaluation				Hatem		Daniel, Hatem, Tareq		
Current and Planned Collaborators	Samtaney, Stanford, NASA, Intel	Observatoire de Paris, NVIDIA	GROMACS, Japan CREST	Bagci	Sun, Aramco, NVIDIA	Schuster, Total, Aramco	Genton, NVIDIA	Shanghai Jiaotong University

Students and scientists in the ECRC are motivated by at least one “grand challenge” application requiring scaling

# ECRC: skating to where the puck will be

<b>MWD</b> multicore wavefront diamond-tiling stencil algorithm	reduces memory bandwidth pressure in shared-cache multicore processors	Girih
<b>BDDC</b> parallel preconditioner for high-contrast elliptic PDEs	reduces synchronization by doing lots of local flops	PETSc
<b>MSPIN</b> nonlinear preconditioner for Newton methods	replaces most global synchronizations with local problems	PETSc
<b>FMM(<math>\epsilon</math>)</b> preconditioner for elliptic problems	has good asymptotic complexity but a high constant; we use in low accuracy (low constant) as a preconditioner	HiCMA
<b>QDWH-SVD</b> singular value decomposition	generates arbitrary amounts of dynamically schedulable concurrency (already beats state-of-the-art on GPUs even though it requires many more flops)	HiCMA
<b>Multi-Add MG</b> multigrid algorithm	creates additional concurrency without the usual convergence penalty of additive MG through elaborate inter-grid transfers	Hypre

## 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 recently with 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
- Examples already 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, 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.

# National Strategic Computing Initiative (NSCI)\*

- Accelerate delivery of an exascale computing system that integrates hardware and software capability to deliver approximately 100 times the performance of current systems across a range of applications
- Increase coherence between the technology base used for modeling and simulation and that used for data analytic computing
- Establish a path forward for HPC systems after the limits of current semiconductor technology are reached (the “post-Moore’s Law era”)
- Increase the capacity and capability of a **national HPC ecosystem** by employing a **holistic approach including** networking technology, **foundational algorithms and software**, and **workforce development**
- Develop a public-private collaboration to ensure that the benefits of the R&D advances are shared between government, industrial, and academic sectors

*“Not since the signing of legislation in 1991 for the HPCC initiative has the nation articulated as bold and specific a goal for the advancement of HPC and the benefits to be derived.”*

– Thomas Sterling and William Gropp, on NCSI in *HPCWire*

---

\* Slightly condensed for for slide display

# Exascale algorithms “mind the gap”



# Thank you



# شكرا

[david.keyes@kaust.edu.sa](mailto:david.keyes@kaust.edu.sa)