Hierarchical Computations on Manycore Architectures: The HiCMA Library

David Keyes, Applied Mathematics & Computational Science
Director, Extreme Computing Research Center (ECRC)
King Abdullah University of Science and Technology
david.keyes@kaust.edu.sa
Intel Xeon Phi Users Group
First Middle East Meeting
Sun-Wed, 22-25 April 2018
@ KAUST, Thuwal
Mecca Province
Saudi Arabia
See you on the shores of the beautiful Red Sea
For hour-long version of this talk, see 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/
The Hierarchical Computations on Manycore Architectures (HiCMA) library aims to redesign existing dense linear algebra libraries to exploit the data sparsity of the matrix operator. Data sparse matrices arise in many scientific problems (e.g., in statistics-based weather forecasting, seismic imaging, and materials science applications) and are characterized by low-rank off-diagonal tile structure. Numerical low-rank approximations have demonstrated attractive theoretical bounds, both in memory footprint and arithmetic complexity. The core idea of HiCMA is to develop fast linear algebra computations operating on the underlying tile low-rank data format, while satisfying a specified numerical accuracy and leveraging performance from massively parallel hardware architectures.

**HiCMA 0.1.0**
- Matrix-Matrix Multiplication
- Cholesky Factorization/Solve
- Double Precision
- Task-based Programming Models
- Shared and Distributed Memory Environments
- Support for StarPU Dynamic Runtime Systems
- Testing Suite and Examples

**CURRENT RESEARCH**
- LU Factorization/Solve
- Schur Complements
- Preconditioners
- Hardware Accelerators
- Support for Multiple Precisions
- Autotuning: Tile Size, Fixed Accuracy and Fixed Ranks
- Support for OpenMP, PaRSEC and Kokkos
- Support for HODLR, H, HSS and H2

**KBLAS**
- Legacy Level-2 BLAS (f, d, s, c) SYMV, GEEmV, HEMV, HEMV
- Legacy Level-3 BLAS (f, d, s, c) TRSM, TRMM, GEMM (b = only)
- Batch Level-3 BLAS (f, d, s, c) TRSM, TRMM, SYTR
- Batch Triangular (f, d, s, c) TRTR, LCTR
- Batch Symmetric (f, d, s, c) POTRF, POTRS, POSV, POTTR, POTL
- Batch General (f, d, s, c) SYD, SYR
  - Standard precisions: f, d, c
  - Very small and large sizes

**CURRENT RESEARCH**
- Half Precision Legacy and Batch BLAS
- Tile Low-Rank (TLR) BLAS on GPUs
- TLR CPU-Resident Matrix Computations
- Adaptive Cross Approximation (ACA) on GPUs

**PERFORMANCE RESULTS**
- Cholesky Factorization – Double Precision – CRAY XC40 with Two-Socket, 16-Core HSW

**DOWNLOAD THE SOFTWARE AT** [http://github.com/ecrc/hicma](http://github.com/ecrc/hicma)

**DOWNLOAD KBLAS AT** [http://github.com/ecrc/kblas](http://github.com/ecrc/kblas)
STARSH is a high performance parallel open-source package of Software for Testing Accuracy, Reliability and Scalability of Hierarchical computations. It provides a hierarchical matrix market in order to benchmark performance of various libraries for hierarchical matrix compressions and computations (including itself). Why hierarchical matrices? Because such matrices arise in many PDEs and use much fewer memory, while requiring less flops for computations. There are several hierarchical data formats, each one with its own performance and memory footprint. STARSH-H intends to provide a standard for assessing accuracy and performance of hierarchical matrix libraries on a given hardware architecture environment. STARSH currently supports the tile low-rank (TLR) data format for approximation on shared and distributed memory systems, using MPI, OpenMP and task-based programming models. STARSH package is available online at https://github.com/ercr/starsh.

Matrix Kernels
- Electrodynamics (one over distance):
  \[ A_{ij} = \frac{1}{r_{ij}} \]
- Electrodynamics (cos over distance):
  \[ A_{ij} = \cos(kr_{ij}) \]
- Spatial statistics (Materen kernel):
  \[ A_{ij} = \frac{1}{\gamma^\beta} \left( \frac{2\gamma}{r_{ij}} \right)^\beta \]
- And many other kernels ...

Sample Problem Setting
Spatial statistics problem for a quasi uniform distribution in a unit square (2D) or cube (3D) with exponential kernel:
\[ A_{ij} = e^{-\beta r_{ij}} \]
where \( \beta \) is a correlation length parameter and \( r_{ij} \) is a distance between \( i \)-th and \( j \)-th spatial points.

3D problem on different two-socket shared-memory Intel Haswell architecture
3D problem on different amount of nodes (from 64 up to 6084) of a distributed-memory shared-memory Intel Haswell architecture

Performance Results
A QDWH-Based SVD Software Framework on Distributed-Memory Manycore Systems

The KAUST SVD (KSVD) is a high performance software framework for computing a dense SVD on distributed-memory manycore systems. The KSVD solver relies on the polar decomposition using the QR Dynamically-Weighted Halley algorithm (QDWH), introduced by Nakatsukasa and Higham (SIAM Journal on Scientific Computing, 2013). The computational challenge resides in the significant amount of extra floating-point operations required by the QDWH-based SVD algorithm, compared to the traditional one-stage bidiagonal SVD. However, the inherent high level of concurrency associated with Level 3 BLAS compute-bound kernels ultimately compensates the arithmetic complexity overhead and makes KSVD a competitive SVD solver on large-scale supercomputers.

The Polar Decomposition
\[ A = U_s H, \quad A = H^{1/2}(Q) \] where \( U_s \) is an orthogonal Matrix, and \( H \) is a symmetric positive semi-definite matrix

Application to SVD
\[ \mathbf{A} = \mathbf{U} \mathbf{S} \mathbf{V} \] where \( \mathbf{U} \) is an orthogonal matrix, \( \mathbf{S} \) is a diagonal matrix with the singular values of \( \mathbf{A} \), and \( \mathbf{V} \) is the transpose of an orthogonal matrix.

The total flop count for QDWH depends on the condition number \( \kappa \) of the matrix.

The KAUST SVD (KSVD) is a high performance software framework for computing a dense SVD on distributed-memory manycore systems. The KSVD solver relies on the polar decomposition using the QR Dynamically-Weighted Halley algorithm (QDWH), introduced by Nakatsukasa and Higham (SIAM Journal on Scientific Computing, 2013). The computational challenge resides in the significant amount of extra floating-point operations required by the QDWH-based SVD algorithm, compared to the traditional one-stage bidiagonal SVD. However, the inherent high level of concurrency associated with Level 3 BLAS compute-bound kernels ultimately compensates the arithmetic complexity overhead and makes KSVD a competitive SVD solver on large-scale supercomputers.

The Polar Decomposition
\[ A = U_H A_n H^{1/2} \text{ (min), where } U_j \text{ is orthogonal Matrix, and } H \text{ is symmetric positive semi-definite matrix} \]

Application to SVD
\[ A = U_s H = U_s (Q) \]
\[ A = U_s (Q) \] where \( U_s \) is an orthogonal matrix, \( H \) is a symmetric positive semi-definite matrix.

The total flop count for QDWH depends on the condition number \( \kappa \) of the matrix.

The Polar Decomposition
\[ A = U_H A_n H^{1/2} \text{ (min), where } U_j \text{ is orthogonal Matrix, and } H \text{ is symmetric positive semi-definite matrix} \]

Application to SVD
\[ A = U_s H = U_s (Q) \]
\[ A = U_s (Q) \] where \( U_s \) is an orthogonal matrix, \( H \) is a symmetric positive semi-definite matrix.

The total flop count for QDWH depends on the condition number \( \kappa \) of the matrix.

The Polar Decomposition
\[ A = U_H A_n H^{1/2} \text{ (min), where } U_j \text{ is orthogonal Matrix, and } H \text{ is symmetric positive semi-definite matrix} \]

Application to SVD
\[ A = U_s H = U_s (Q) \]
\[ A = U_s (Q) \] where \( U_s \) is an orthogonal matrix, \( H \) is a symmetric positive semi-definite matrix.

The total flop count for QDWH depends on the condition number \( \kappa \) of the matrix.

The Polar Decomposition
\[ A = U_H A_n H^{1/2} \text{ (min), where } U_j \text{ is orthogonal Matrix, and } H \text{ is symmetric positive semi-definite matrix} \]

Application to SVD
\[ A = U_s H = U_s (Q) \]
\[ A = U_s (Q) \] where \( U_s \) is an orthogonal matrix, \( H \) is a symmetric positive semi-definite matrix.

The total flop count for QDWH depends on the condition number \( \kappa \) of the matrix.

The Polar Decomposition
\[ A = U_H A_n H^{1/2} \text{ (min), where } U_j \text{ is orthogonal Matrix, and } H \text{ is symmetric positive semi-definite matrix} \]

Application to SVD
\[ A = U_s H = U_s (Q) \]
\[ A = U_s (Q) \] where \( U_s \) is an orthogonal matrix, \( H \) is a symmetric positive semi-definite matrix.

The total flop count for QDWH depends on the condition number \( \kappa \) of the matrix.

The Polar Decomposition
\[ A = U_H A_n H^{1/2} \text{ (min), where } U_j \text{ is orthogonal Matrix, and } H \text{ is symmetric positive semi-definite matrix} \]

Application to SVD
\[ A = U_s H = U_s (Q) \]
\[ A = U_s (Q) \] where \( U_s \) is an orthogonal matrix, \( H \) is a symmetric positive semi-definite matrix.

The total flop count for QDWH depends on the condition number \( \kappa \) of the matrix.
The Multi-Object Adaptive Optics (MOAO) framework provides a comprehensive testbed for high performance computational astronomy. In particular, the European Extremely Large Telescope (E-ELT) is one of today’s most challenging projects in ground-based astronomy and will make use of a MOAO instrument based on turbulence tomography. The MOAO framework uses a novel compute-intensive pseudo-analytical approach to achieve close to real-time data processing on manycore architectures. The scientific goal of the MOAO simulation package is to dimension future E-ELT instruments and to assess the qualitative performance of tomographic reconstruction of the atmospheric turbulence on real datasets.

**THE MULTI-OBJECT ADAPTIVE OPTICS TECHNIQUE**

*Single conjugate AO concept.*  
*Open-Loop tomography concept.*  
*Observing the GOODS South cosmological field with MOAO.*  
*High res. map of the quality of turbulence compensation obtained with MOAO on a cosmological field.*

**MOAO 0.1.0**

- Tomographic Reconstructor Computation  
- Dimensioning of Future Instruments  
- Real Datasets  
- Single and Double Precisions  
- Shared Memory Systems  
- Task-based Programming Models  
- Dynamic Runtime Systems  
- Hardware Accelerators

**CURRENT RESEARCH**

- Distributed Memory Systems  
- Hierarchical Matrix Compression  
- Machine Learning for Atmospheric Turbulence  
- High Resolution Galaxy Map Generation  
- Extend to other ground-based telescope projects

**PERFORMANCE RESULTS**

**TOMOGRAPHIC RECONSTRUCTOR COMPUTATION - DOUBLE PRECISION**

- Two-socket 18 core Intel HSW - 64 core Intel KNL - 8 NVIDIA GPU P100s (DGX-1)

**ExaGeoStat**

The Exascale Geostatistics project (ExaGeoStat) is a parallel high performance unified framework for computational geostatistics on many-core systems. The project aims at optimizing the likelihood function for a given spatial data to provide an efficient way to predict missing observations in the context of climate/weather forecasting applications. This machine learning framework proposes a unified simulation code structure to target various hardware architectures, from commodity x86 to GPU accelerator-based shared and distributed-memory systems. ExaGeoStat enables statisticians to tackle computationally challenging scientific problems at large-scale, while abstracting the hardware complexity, through state-of-the-art high performance linear algebra software libraries.

**ExaGeoStat Dataset Generator**

- Generate 2D spatial Locations using uniform distribution.  
- Matérn covariance function: $C(r; \theta) = \frac{1}{2} \sum_{n=0}^{N} \theta_n \left( \frac{r}{\rho_n} \right)^n K_n \left( \frac{r}{\rho_n} \right)$  
- Cholesky factorization of the covariance matrix: $\Sigma(\theta) = V \cdot V^T$  
- Measurement vector generation ($Z$): $Z = X \cdot \Sigma^{-1}(0,1)$

**ExaGeoStat Maximum Likelihood Estimator**

- Maximum Likelihood Estimation (MLE) learning function: $f(\beta) = -\frac{1}{2} \log(x^T \Sigma(X)) - \frac{1}{2} \log(|\Sigma(X)|) - \frac{1}{2} z^T \Sigma^{-1}(X) z$  
- Where $\Sigma(X)$ is a covariance matrix with entries $\Sigma(X)_{ij} = C(X_i - X_j)$, $i, j = 1, ..., n$

**ExaGeoStat Predictor**

- MLE prediction problem  
- Tile Low Rank (TLR) approximation  
- NetCDF format support  
- PyHSEC runtime system  
- In situ processing

**Performance Results (MLE)**

**Parallel high performance unified framework for geostatistics on many-core systems.**

**A high performance multi-object adaptive optics framework for ground-based astronomy.**

**MOAO**

The Multi-Object Adaptive Optics (MOAO) framework provides a comprehensive testbed for high performance computational astronomy. In particular, the European Extremely Large Telescope (E-ELT) is one of today’s most challenging projects in ground-based astronomy and will make use of a MOAO instrument based on turbulence tomography. The MOAO framework uses a novel compute-intensive pseudo-analytical approach to achieve close to real-time data processing on manycore architectures. The scientific goal of the MOAO simulation package is to dimension future E-ELT instruments and to assess the qualitative performance of tomographic reconstruction of the atmospheric turbulence on real datasets.

**THE MULTI-OBJECT ADAPTIVE OPTICS TECHNIQUE**

*Single conjugate AO concept.*  
*Open-Loop tomography concept.*  
*Observing the GOODS South cosmological field with MOAO.*  
*High res. map of the quality of turbulence compensation obtained with MOAO on a cosmological field.*

**MOAO 0.1.0**

- Tomographic Reconstructor Computation  
- Dimensioning of Future Instruments  
- Real Datasets  
- Single and Double Precisions  
- Shared Memory Systems  
- Task-based Programming Models  
- Dynamic Runtime Systems  
- Hardware Accelerators

**CURRENT RESEARCH**

- Distributed Memory Systems  
- Hierarchical Matrix Compression  
- Machine Learning for Atmospheric Turbulence  
- High Resolution Galaxy Map Generation  
- Extend to other ground-based telescope projects

**PERFORMANCE RESULTS**

**TOMOGRAPHIC RECONSTRUCTOR COMPUTATION - DOUBLE PRECISION**

- Two-socket 18 core Intel HSW - 64 core Intel KNL - 8 NVIDIA GPU P100s (DGX-1)

**Download the library at** http://github.com/erci/exageoast:  

**A collaboration of**  
**With support from**  
**Sponsored by**  

Pick up flyers at KAUST Booth #1671

Download the software at http://github.com/erci/moaao
The Girih framework implements a generalized multi-dimensional intra-tile parallelization scheme for shared-cache multicore processors that results in a significant reduction of cache size requirements for temporally blocked stencil codes. It ensures data access patterns that allow efficient hardware prefetching and TLB utilization across a wide range of architectures. Girih is built on a multicore wavefront diamond tiling approach to reduce horizontal data traffic in favor of locally cached data reuse. The Girih library reduces cache and memory bandwidth pressure, which makes it amenable to current and future cache and bandwidth-starved architectures, while enhancing performance for many applications.

STENCIL COMPUTATIONS
- Hot spot in many scientific codes
- Appear in finite difference, element, and volume discretizations of PDEs
- E.g., 3D wave acoustic wave equation

MULTIDIMENSIONAL INTRA-TILE PARALLELIZATION
- Thread binding to cores with sched
- Memory affinity with numatcl
- Thread assignment in space-time dimensions

SOFTWARE INFRASTRUCTURE
- GIRIH 1.0.0
  - MPI + OpenMP
  - Single and double precision
  - Autotuning
  - Short and long stencil ranges in space and time
  - Constant/variable coefficients
  - LIKWID support for profiling

CURRENT RESEARCH
- Matrix power kernels
- Overlapping domain decomposition
- GPU hardware accelerators:
  - OpenACC / CUDA
  - Out-of-core algorithms
  - Dynamic runtime systems
  - Extension to CFD applications

Performance results:
- Domain size 512 x 512 x 512
- # of time steps: 500
- 25-point star stencil
- Dirichlet boundary conditions
- Two-socket systems (Mem./L3):
  - 8-core Intel SNB (64GB/20MB)
  - 16-core Intel HSW (128GB/40MB)
  - 28-core Intel SKL (256GB/38MB)
- Intel compiler suite v17 with AVX512 flag enabled
- Autotuning
- LiKWID support for profiling

Download the software at [GitHub](https://github.com/ecrc/girih)
END ADVERTISING
Architectural background

www.exascale.org/iesp

The International Exascale Software Roadmap

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*
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
The familiar

Taihu Light

Shaheen

Sequoia

K computer
The challenge
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
Some algorithmic imperatives

- Reduce communication and 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

- See Amani & Dalal’s talks in the 3:45pm session
Widely applicable strategies

1) Employ dynamic runtime systems based on directed acyclic task graphs (DAGs)
   - e.g., Charm++, Quark, StarPU, Legion, OmpSs, HPX, ADLB, Argo

2) Exploit data sparsity of hierarchically low-rank type
   - meet the “curse of dimensionality” with the “blessing of low rank”

3) Code to the architecture, but present an abstract API
1) 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
2) Hierarchically low-rank operators

- **Advantages**
  - shrink memory footprints to live higher on the memory hierarchy
    - higher means quick access (↑ arithmetic intensity)
  - reduce operation counts
  - tune work to accuracy requirements
    - e.g., preconditioner versus solver

- **Disadvantages**
  - pay cost of compression
  - not all operators compress well
3) 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
Reducing over-ordering and synchronization through DAGs, ex.: generalized eigensolver

\[ Ax = \lambda Bx \]

<table>
<thead>
<tr>
<th>Operation</th>
<th>Explanation</th>
<th>LAPACK routine name</th>
</tr>
</thead>
<tbody>
<tr>
<td>( B = L \times L^T )</td>
<td>Cholesky factorization</td>
<td>POTRF</td>
</tr>
<tr>
<td>( C = L^{-1} \times A \times L^{-T} )</td>
<td>application of triangular factors</td>
<td>SYGST or HEGST</td>
</tr>
<tr>
<td>( T = Q^T \times C \times Q )</td>
<td>tridiagonal reduction</td>
<td>SYEVD or HEEVD</td>
</tr>
<tr>
<td>( Tx = \lambda x )</td>
<td>QR iteration</td>
<td>STERF</td>
</tr>
</tbody>
</table>
Loop nests and subroutine calls, with their over-orderings, can be replaced with DAGs:

- Diagram shows a dataflow ordering of the steps of a $4 \times 4$ 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
Reducing memory footprint and operation complexity with low rank

• When dense blocks arise in matrix operations, replace them with hierarchical representations
• Use high accuracy (high rank, but typically less than full) to build “exact” solvers
• Use low accuracy (low rank) to build preconditioners
• Block structure and rank provide useful tuning parameters for migration onto variety of hardware configurations
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

\[ A = \begin{bmatrix} 2 & -1 & \text{ } \\ -1 & 2 & -1 \\ -1 & 2 \end{bmatrix} \]

\[ A^{-1} = \frac{1}{8} \times \begin{bmatrix} 7 & 6 & 5 & | & 4 & 3 & 2 & 1 \\ 6 & 12 & 10 & | & 8 & 6 & 4 & 2 \\ 5 & 10 & 15 & | & 12 & 9 & 6 & 3 \\ 4 & 8 & 12 & | & 16 & 12 & 8 & 4 \\ 3 & 6 & 9 & | & 12 & 15 & 10 & 5 \\ 2 & 4 & 6 & | & 8 & 10 & 12 & 6 \\ 1 & 2 & 3 & | & 4 & 5 & 6 & 7 \end{bmatrix} \]

\[ = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 0 & 0 \end{bmatrix} \]

\[ = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \begin{bmatrix} 4 & 3 & 2 & 1 \end{bmatrix} \]
Recursive construction of an $H$-matrix
“Standard (strong)” vs. “weak” admissibility

After Hackbusch, et al., 2003
Some solvers that leverage data sparsity

<table>
<thead>
<tr>
<th>Package</th>
<th>Technique</th>
<th>Format</th>
<th>Contact</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACR</td>
<td>Cyclic reduction</td>
<td>$\mathcal{H}$</td>
<td>G. Chávez, G. Turkiyyah, D. Keyes [118]</td>
</tr>
<tr>
<td>AHMED</td>
<td>$\mathcal{H}^{-1}$ &amp; $\mathcal{H}$-LU</td>
<td>$\mathcal{H}$</td>
<td>M. Bebendorf [120]</td>
</tr>
<tr>
<td>BLR PaStiX</td>
<td>Supernodal</td>
<td>BLR</td>
<td>G. Fichon, M. Faverge [44]</td>
</tr>
<tr>
<td>CE</td>
<td>$\mathcal{H}^2$-LU</td>
<td>$\mathcal{H}^2$</td>
<td>D. Sushnikova, I. Oseledets [57]</td>
</tr>
<tr>
<td>DMHIF</td>
<td>Multifrontal</td>
<td>ID</td>
<td>Y. Li, L. Ying [43]</td>
</tr>
<tr>
<td>DMHM</td>
<td>Newton-Schulz</td>
<td>$\mathcal{H}$</td>
<td>Y. Li, L. Ying [121]</td>
</tr>
<tr>
<td>ExaFMM</td>
<td>Fast multipole</td>
<td>$\mathcal{H}$</td>
<td>H. Ibeid, R. Yokota, D. Keyes [122]</td>
</tr>
<tr>
<td>H2Lib</td>
<td>$\mathcal{H}^{-1}$ &amp; $\mathcal{H}^2$-LU</td>
<td>$\mathcal{H}^2$</td>
<td>S. Christophersen, S. Börm [123]</td>
</tr>
<tr>
<td>HLib</td>
<td>$\mathcal{H}^{-1}$ &amp; $\mathcal{H}$-LU</td>
<td>$\mathcal{H}$</td>
<td>L. Grasedyck, W. Hackbusch [124]</td>
</tr>
<tr>
<td>HLibPro</td>
<td>$\mathcal{H}^{-1}$ &amp; $\mathcal{H}$-LU</td>
<td>$\mathcal{H}$</td>
<td>R. Kriemann, W. Hackbusch [54]</td>
</tr>
<tr>
<td>LoRaSp</td>
<td>$\mathcal{H}^2$-LU</td>
<td>$\mathcal{H}^2$</td>
<td>H. Pouransari, E. Darve [56]</td>
</tr>
<tr>
<td>MF-HODLR</td>
<td>Multifrontal</td>
<td>HODLR</td>
<td>A. Aminfar, E. Darve [30]</td>
</tr>
<tr>
<td>MUMPS-BLR</td>
<td>Multifrontal</td>
<td>BLR</td>
<td>T. Mary, P. R. Amestoy [22]</td>
</tr>
<tr>
<td>Structured CHOLMOD</td>
<td>Supernodal</td>
<td>BLR</td>
<td>J. Chadwick, D. Bindel [46]</td>
</tr>
<tr>
<td>STRUMPACK-Sparse</td>
<td>Multifrontal</td>
<td>HSS</td>
<td>F.-H. Rouet, P. Ghysels, X.S. Li [36]</td>
</tr>
</tbody>
</table>

Please notify if you have released one that is not here:
gustavo.chavez@kaust.edu.sa

---

c/o G. Chavez (KAUST)
“Hourglass” model for algorithms
(borrowed from internet protocols)
Hierarchical Computations on Manycore Architectures: HiCMA*

* “Hikmah” is the Arabic word for wisdom
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

• **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:
Thank you!

شكرا

https://github.com/ecrc/
david.keyes@kaust.edu.sa