Toward Making HiCMA Hardware-Agnostic with SYCL

H. Ltaief[1], S. Abdulah[1], R. Alomairy[1], A. Dabah[1], Y. Hong[1], H. Ibeid[2], D. Keyes[1], M. Moawad[3], A. Nasr[3]

[1]Extreme Computing Research Center, KAUST
[2]Intel Corporation
[3]Brightskies

SIAM Conference on Computational Science and Engineering (CSE23)
February 26 - March 3, 2023
RAI Congress Centre | Amsterdam, The Netherlands

Acknowledgments

- **ECRC**: S. Abdulah, R. Alomairy, A. Dabah, M. Genton, Y. Hong, D. Keyes, L. Qu, M. Ravasi, Y. Sun **(Linear Algebra, Statistics, and Seismic)**
- **Mines Paristech**: W. Bader, Y. Mesri **(Mesh Deformations)**
- **Paris Observatory France, Australian National University - Research School of Astronomy & Astrophysic (ANU), National Astronomical Observatory of Japan**: D. Gratadour, Jesse Cranney, O. Guyon **(Computational Astronomy)**
- **ICL@UTK**: G. Bosilca, Q. Cao, J. Dongarra, Y. Pei **(Runtime Systems)**
- **INRIA Bordeaux / LABRI**: M. Faverge, S. Thibault **(Runtime Systems)**
- **HPC Resources**: KAUST, HLRS, ORNL, RIKEN **(Machines)**
- **Vendors**: A. Esposito (HPE/Cray), L. Gatineau (NEC), I. Said / S. Jones (NVIDIA), M. Sabony / A. Lashab (AMD), A. Al-Jeshi / A. Alabduljabbar (Intel), F. Dupros (ARM), T. Hoshiya (Fujitsu) **(Chips)**

Hardware Landscape

Intel #97

AMD #27

IBM/NVIDIA CPU/GPU #4

Fujitsu ARM A64FX #2

Hardware Landscape

Intel #97

AMD #27

IBM/NVIDIA CPU/GPU #4

Fujitsu ARM A64FX #2

intel

AMD

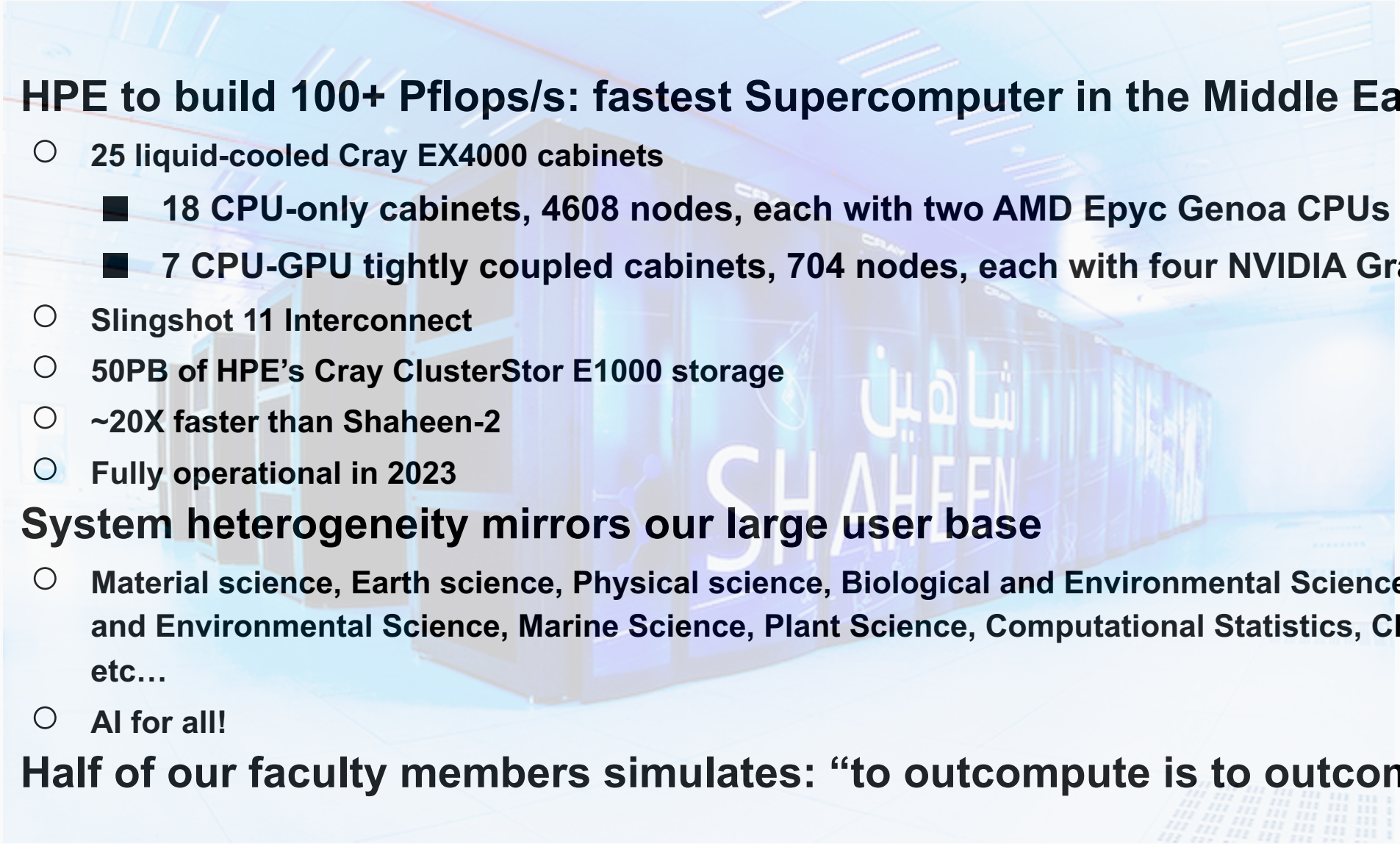
NVIDIA

FUJITSU

NEC

GRAPHCORE

Shaheen-3 Announcement

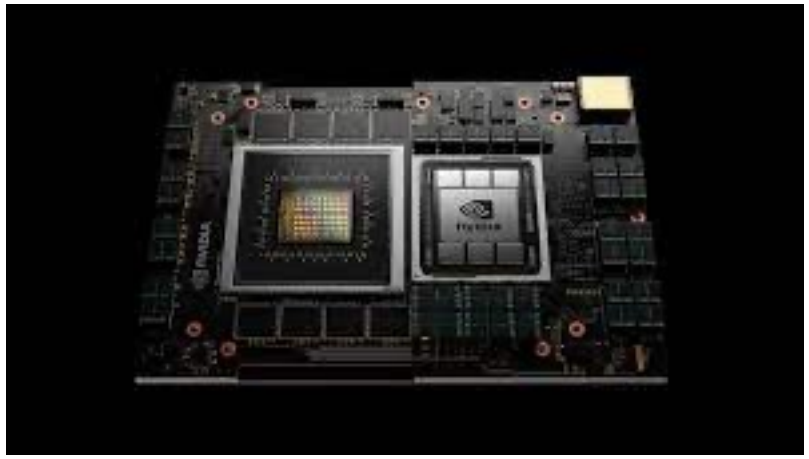


- HPE to build 100+ Pflops/s: fastest Supercomputer in the Middle East
  - 25 liquid-cooled Cray EX4000 cabinets
    - 18 CPU-only cabinets, 4608 nodes, each with two AMD Epyc Genoa CPUs
    - 7 CPU-GPU tightly coupled cabinets, 704 nodes, each with four NVIDIA Grace Hopper
  - Slingshot 11 Interconnect
  - 50PB of HPE's Cray ClusterStor E1000 storage
  - ~20X faster than Shaheen-2
  - Fully operational in 2023
- System heterogeneity mirrors our large user base
  - Material science, Earth science, Physical science, Biological and Environmental Science, Biological and Environmental Science, Marine Science, Plant Science, Computational Statistics, Chemistry, etc…
  - AI for all!
- Half of our faculty members simulates: "to outcompute is to outcompete!"

Hardware Landscape

AMD Epyc Milan-X

High cache capacity
High memory bandwidth
x86 programming env
Memory-bound workloads



NVIDIA Grace Hopper

High speed CPU-GPU interconnect
Memory coherency
Support for mixed precisions
Compute-bound workloads

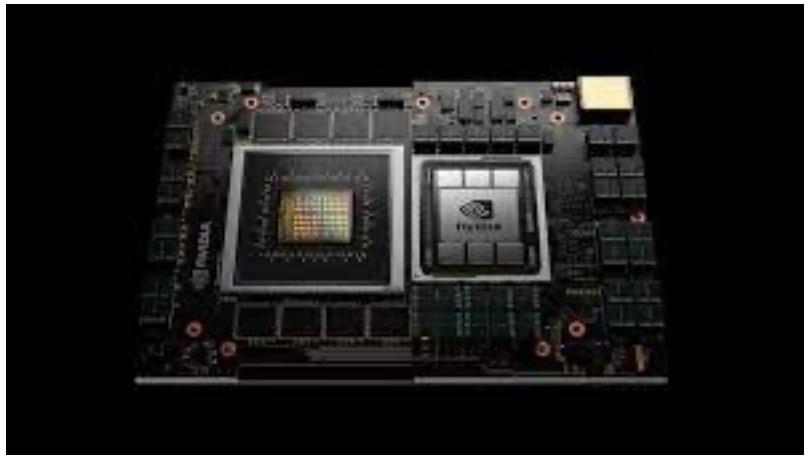
Graphcore IPU

AI-focused chip
Flat memory hierarchy
High SRAM bandwidth
Inference

Hardware Landscape

AMD Epyc Milan-X

High cache capacity
High memory bandwidth
x86 programming env
Memory-bound workloads



NVIDIA Grace Hopper

High speed CPU-GPU interconnect
Memory coherency
Support for mixed precisions
Compute-bound workloads

Graphcore IPU

AI-focused chip
Flat memory hierarchy
High SRAM bandwidth
Inference

How do we reconcile this hostile environment with HPC scientific applications?

HPC Scientific Applications

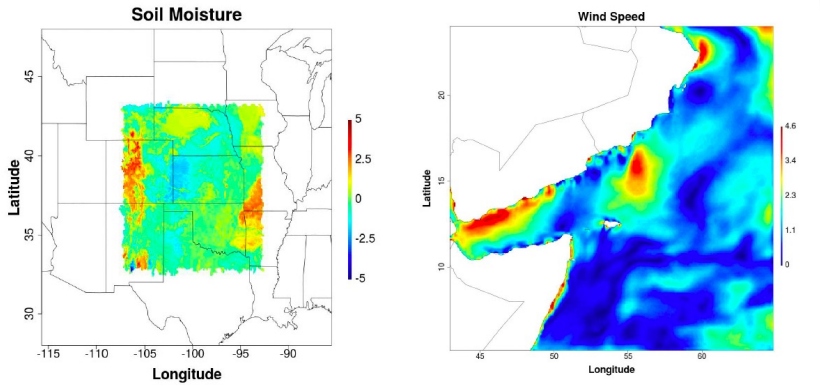
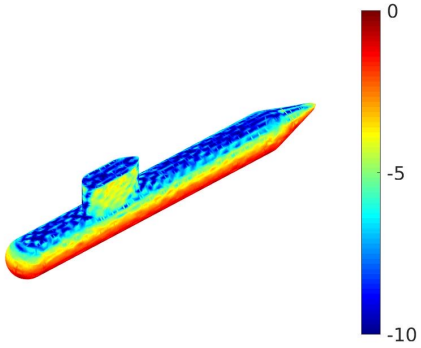
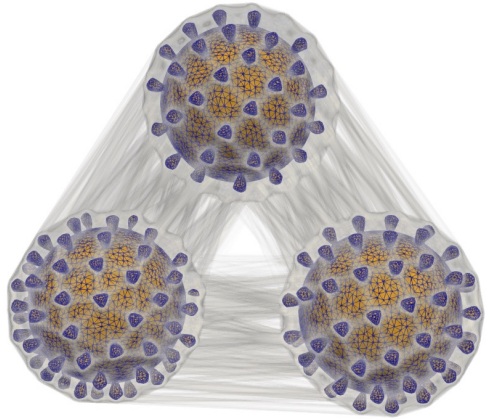


Fig. 4: **Left:** Soil moisture residuals at the topsoil of the Mississippi River basin. **Right**: Wind speed (m/s) in the Arabian Sea.

3D Geospatial Statistics

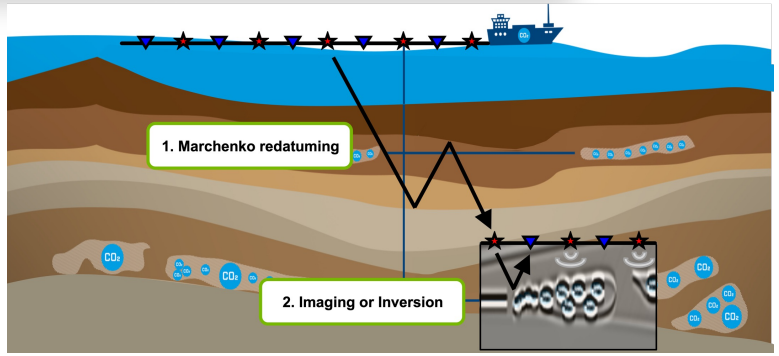


3D Computational Electromagnetics

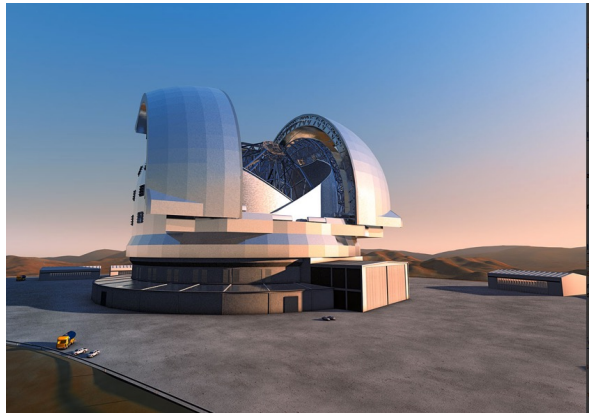


3D Mesh Deformations

Wireless Communications

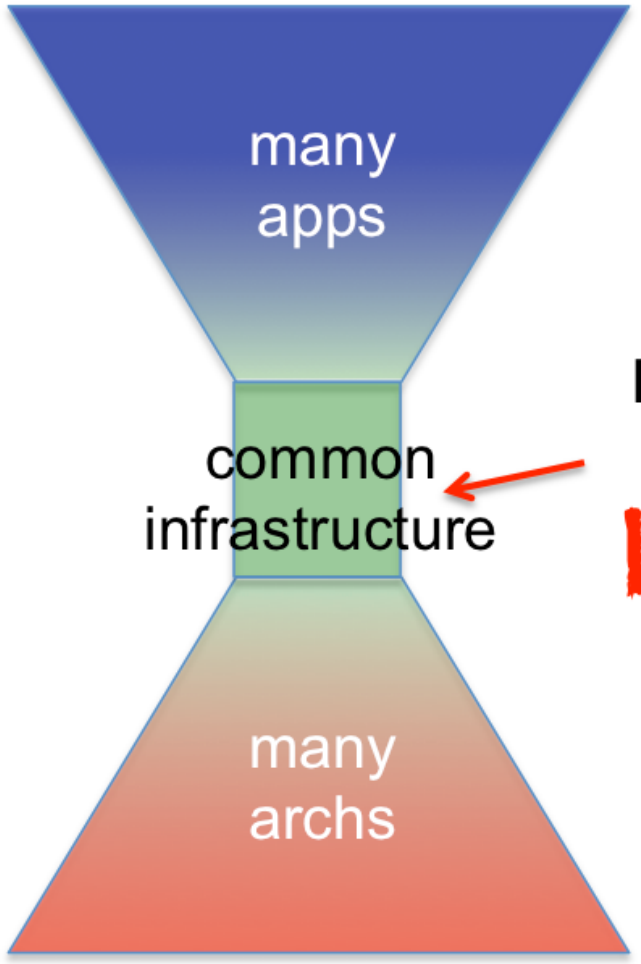


Seismic Imaging



Computational Astronomy

Revisiting the Hourglass



$$A\,x = b$$

ECRC is right

here

@KAUST_ECRC

https://www.facebook.com/ecrckaust

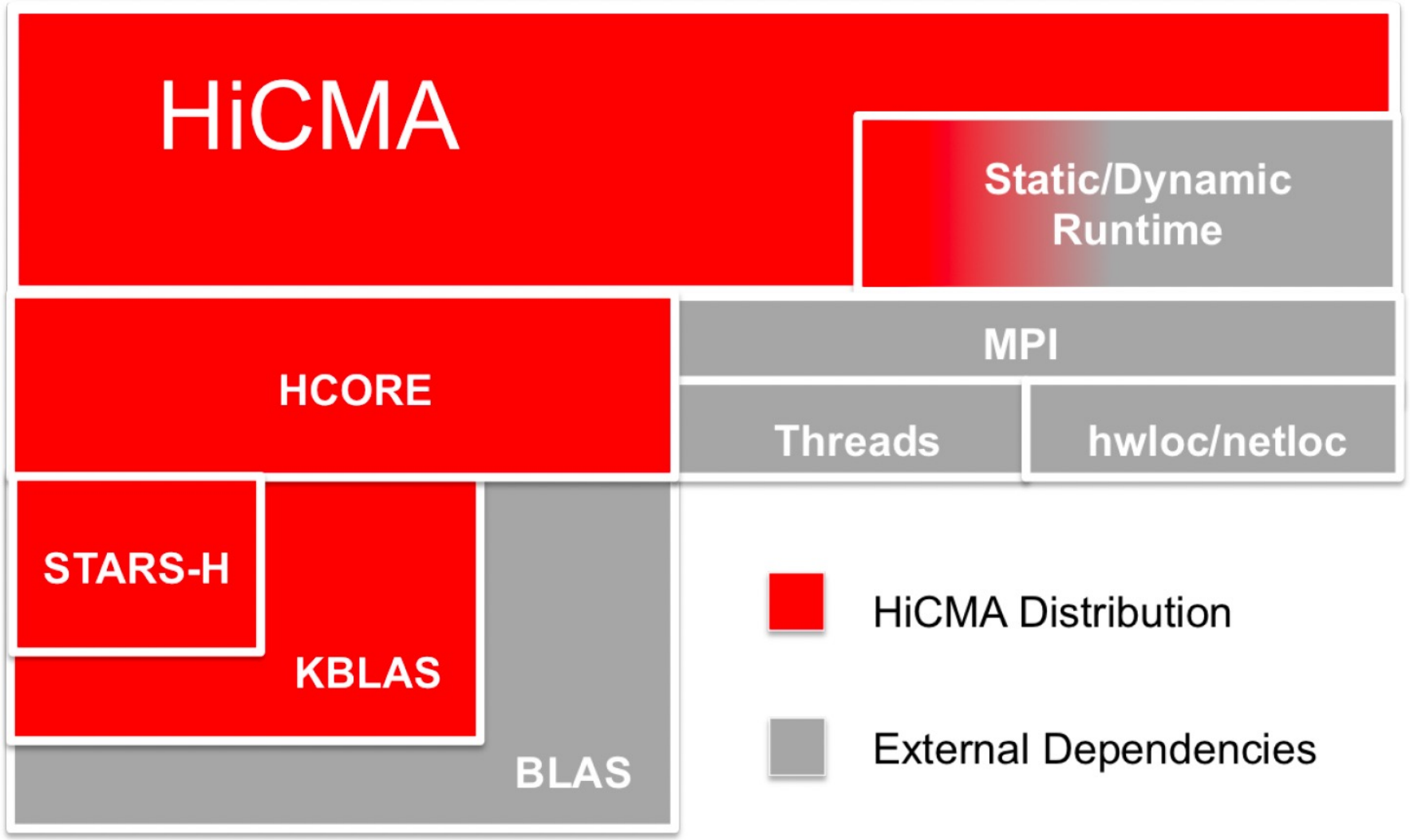
Reshaping Linear Algebra for Massively Parallel Architectures

- Expose parallelism using task-based programming models
- Enhance user-productivity using layers of abstraction
- Ensure scalability using asynchronous executions
- Mitigate synchronization overheads using fine-grained computations
- Reduce data motion using mixed precisions
- Exploit data sparsity using low-rank approximations
- Maintain code portability using standard basic blocks

Are you willing to redesign your algorithm?

One possible productive solution: **Matricization**

The HiCMA Library



Available at http://github.com/ecrc/hicma

HPC Scientific Applications

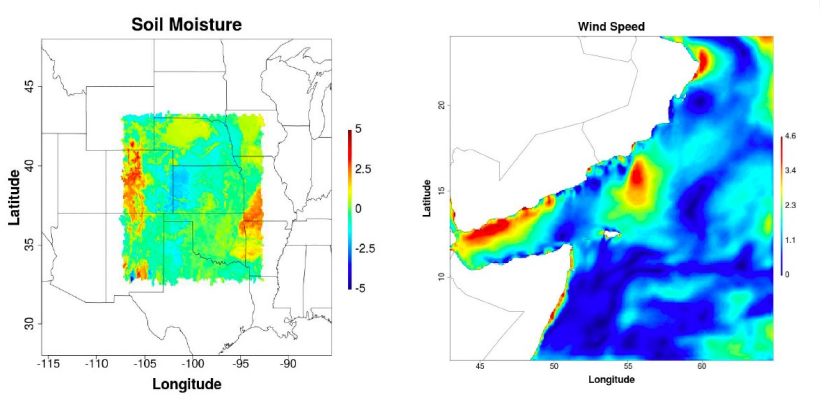
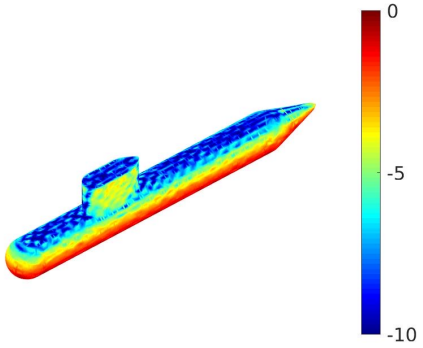
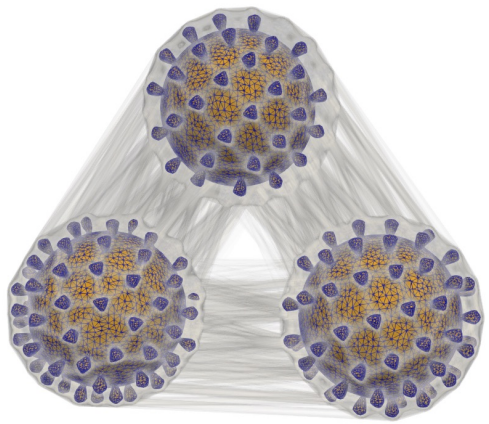


Fig. 4: **Left:** Soil moisture residuals at the topsoil of the Mississippi River basin. **Right**: Wind speed (m/s) in the Arabian Sea.

3D Geospatial Statistics

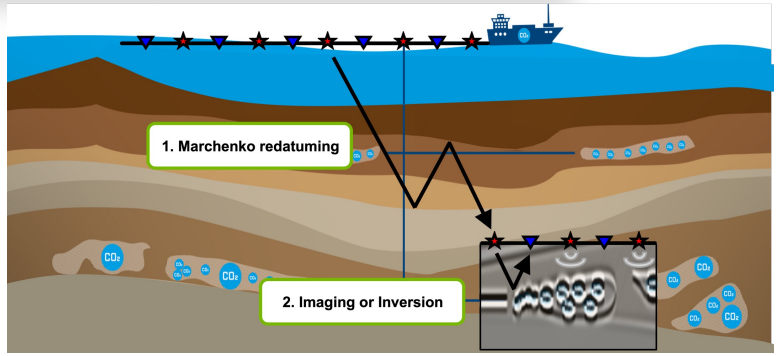


3D Computational Electromagnetics

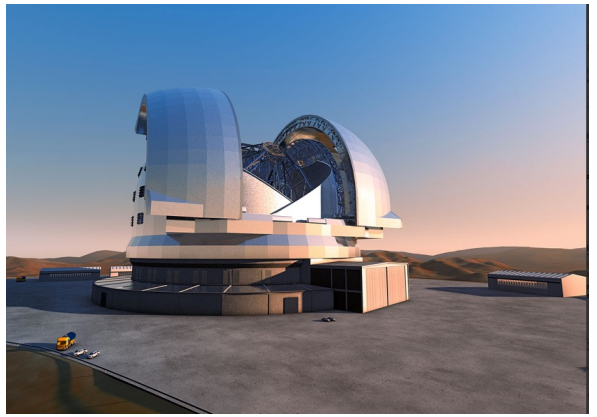


3D Mesh Deformations

Wireless Communications



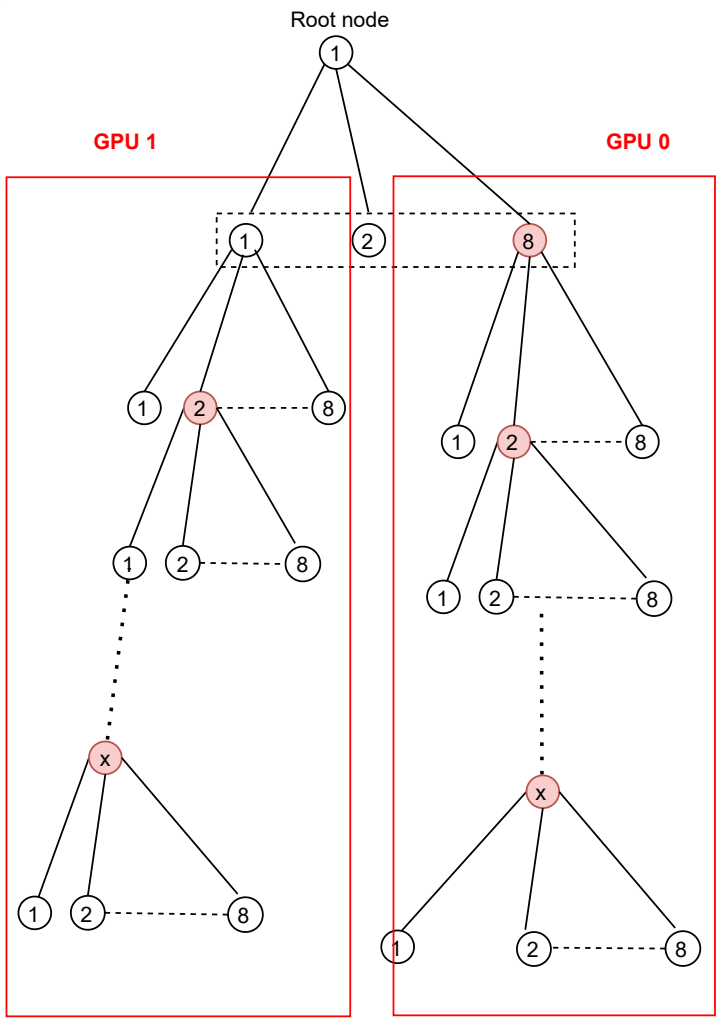
Seismic Imaging



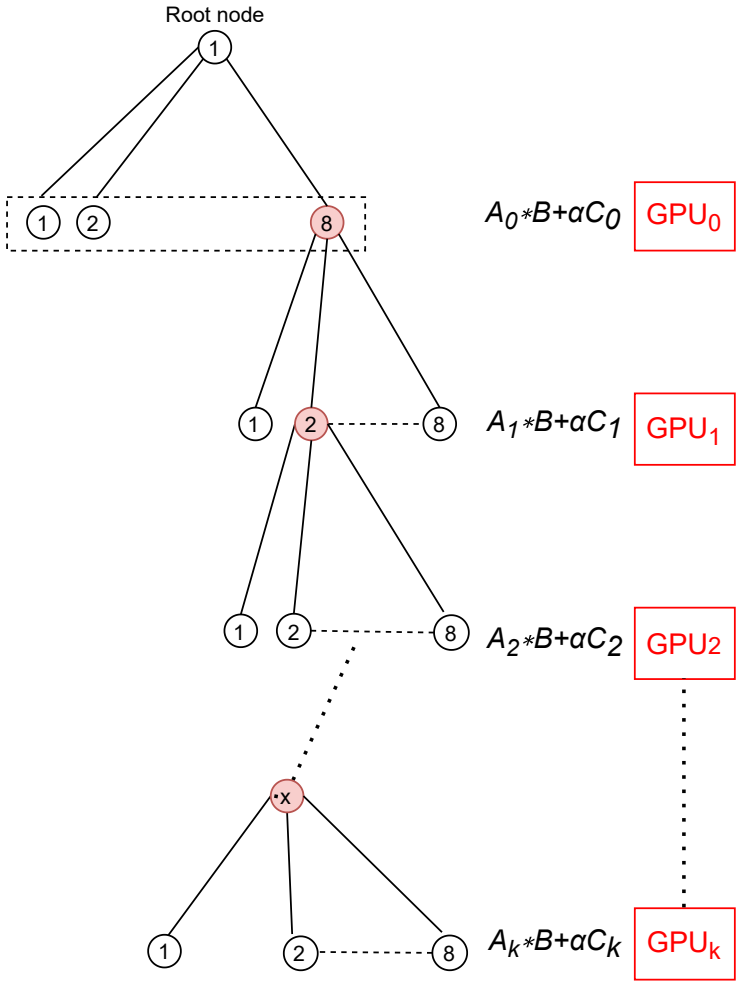
Computational Astronomy

Accelerating Multiple-Input Multiple-Output Wireless Communication Networks

- Implement the Sphere Decoder approach
- Matricize the tree traversal of the detection algorithm
- Design a multi-level approach to ensure high occupancy and increase accuracy
- Cast the computations in terms of batch GEMMs
- ***A. Dabah et al, accepted at ISC23***



First multi-GPU version
Multi-start



Second multi-GPU version
Multi-Gemm

Data Parallel C++ (DPC++)

- DPC++ = C++ and SYCL and **Extensions**.
  - Many of the DPC++ extensions (ex., Unified Shared Memory) are now merged in the new SYCL standard.
- Why SYCL?
  - Syntax for accelerators (device selection, offload, memory transfer).
  - Single source (host and device code).
  - Implicit or Explicit data-transfer.
  - Stack based on standards and open specifications (CLANG, LLVM, Level Zero).

```
sycl::queue q(cpu_selector{});
auto A = sycl::malloc_shared<float>(n, q);
auto B = sycl::malloc_shared<float>(n, q);
q.parallel_for( sycl::range<1>{n},
[=] (sycl::id<1> i) {
B[i] += A[i] * A[i];
}
).wait();
```

```
sycl::queue q(gpu_selector{});
auto A = sycl::malloc_shared<float>(n, q);
auto B = sycl::malloc_shared<float>(n, q);
q.parallel_for( sycl::range<1>{n},
[=] (sycl::id<1> i) {
B[i] += A[i] * A[i];
}
).wait();
```

oneMKL DPC++ APIs

```
sycl::queue Q{sycl::cpu_selector{}};
sycl::queue Q{sycl::gpu_selector{}};
sycl::queue Q{device};
```

Create queue attached to a given device or device type. All device execution goes through a queue object.

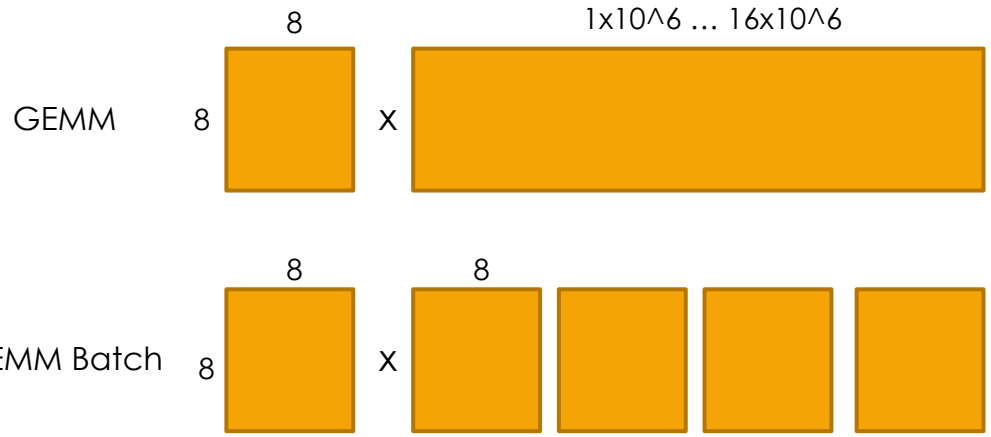
```
void *mem = sycl::malloc_shared(bytes, Q);
void *mem = sycl::malloc_device(bytes, Q);
```

Allocate device-accessible memory. Malloc_shared memory is also accessible from the host.

```
#include “oneapi/mkl.hpp“

int main() {

…

// Select device and create queue
sycl::device dev = sycl::device(sycl::gpu_selector());
sycl::queue queue(dev);

// Allocate arrays
a          = sycl::malloc_shared<double>(batch_size * sizeof(fp
*), queue);
…

// Initialize array
…

sycl::event gemm_batch_done;

gemm_batch_done = oneapi::mkl::blas::gemm_batch(main_queue, ta,
tb, m, n, k, alpha, (const fp **) a, lda, (const fp **) b, ldb,
beta, c, ldc, group_count, group_size, gemm_batch_dependencies);

gemm_batch_done.wait();

…
```

Intel Ponte Vecchio GPU Hardware Accelerator

GEMM_batch on PVC for MIMO Application



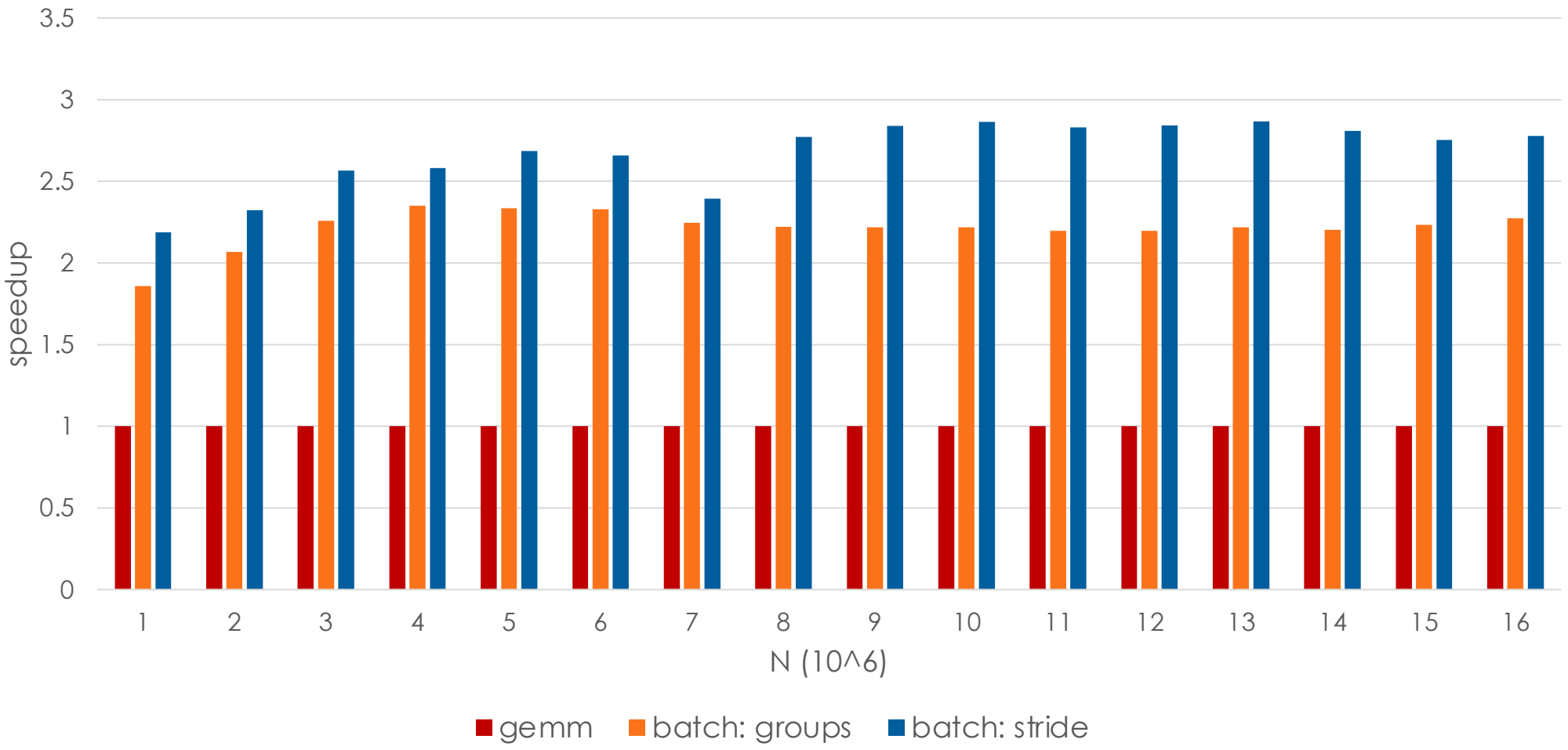
group API

```
idx = 0
for i = 0 … group_count – 1
    for j = 0 … group_size – 1
        A, B, and C are matrices in a[idx], b[idx] and c[idx]
        C := alpha[i] * op(A) * op(B) + beta[i] * C
        idx = idx + 1
    end for
end for
```

strided API

```
for i = 0 … batch_size – 1
    A, B and C are matrices at offset i * stridea, i * strideb, i * stridec in a, b and c.
    C := alpha * op(A) * op(B) + beta * C
end for
```

GEMM_batch on PVC for MIMO Application



HPC Scientific Applications

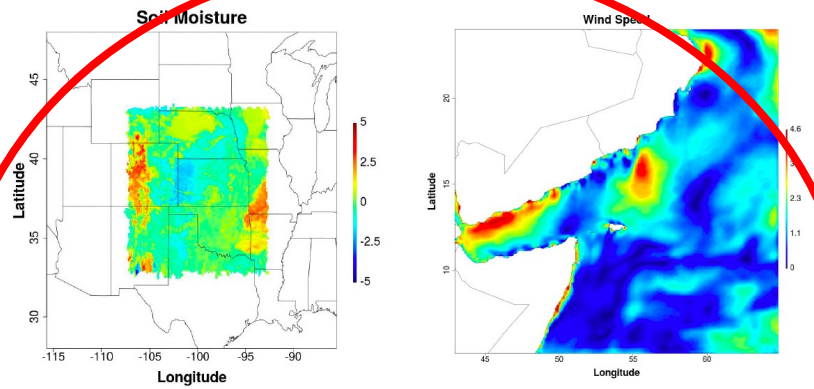
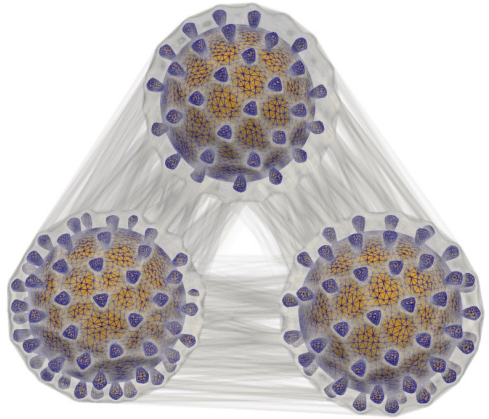
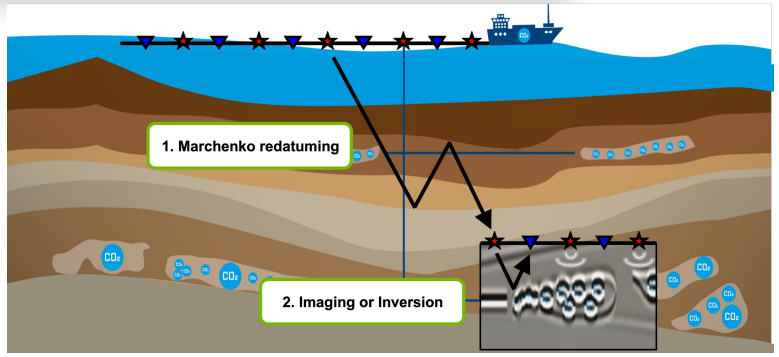


Fig. 4: **Left:** Soil moisture residuals at the topsoil of the Mississippi River basin. **Right**: Wind speed (m/s) in the Arabian Sea.

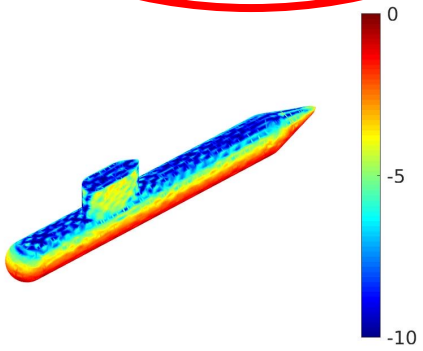
3D Geospatial Statistics



3D Mesh Deformations

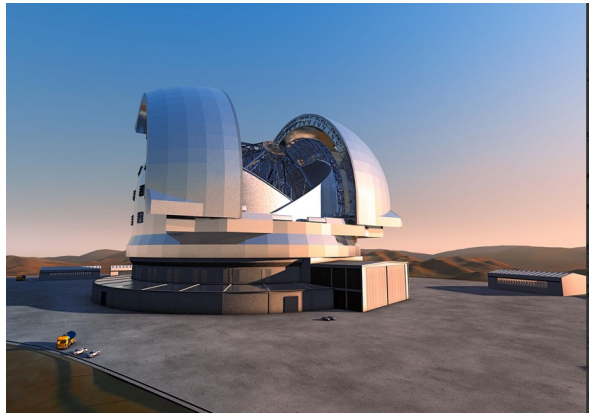


Seismic Imaging



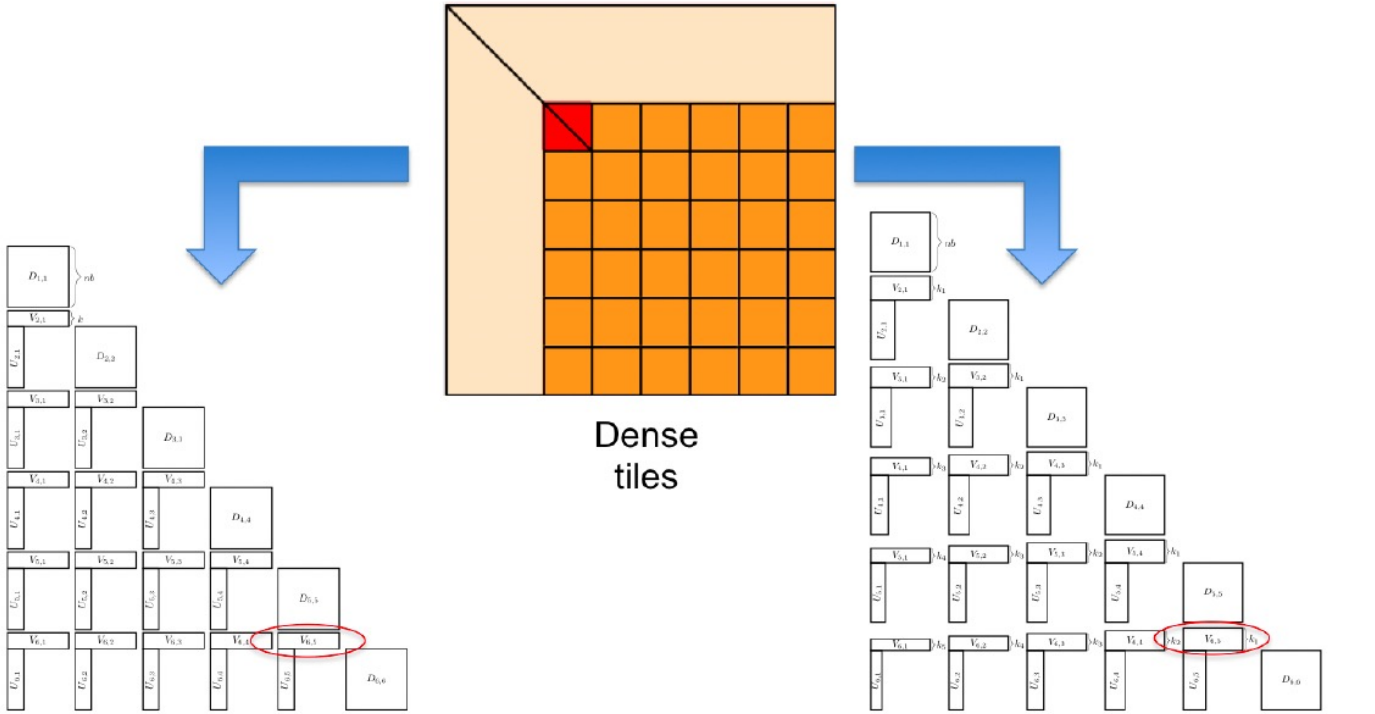
3D Computational Electromagnetics

Wireless Communications



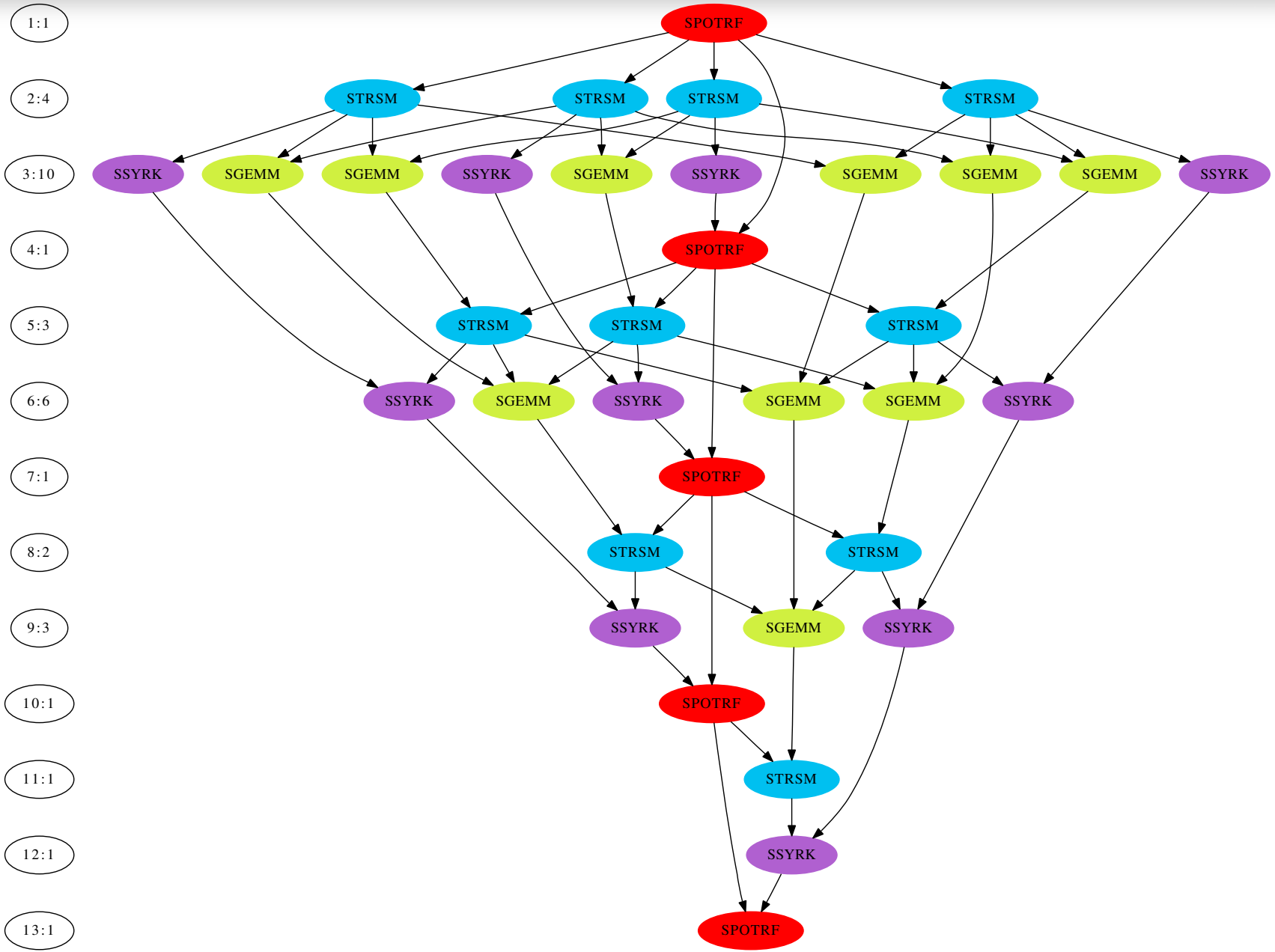
Computational Astronomy

Tile Low-Rank as a Pragmatic Approach

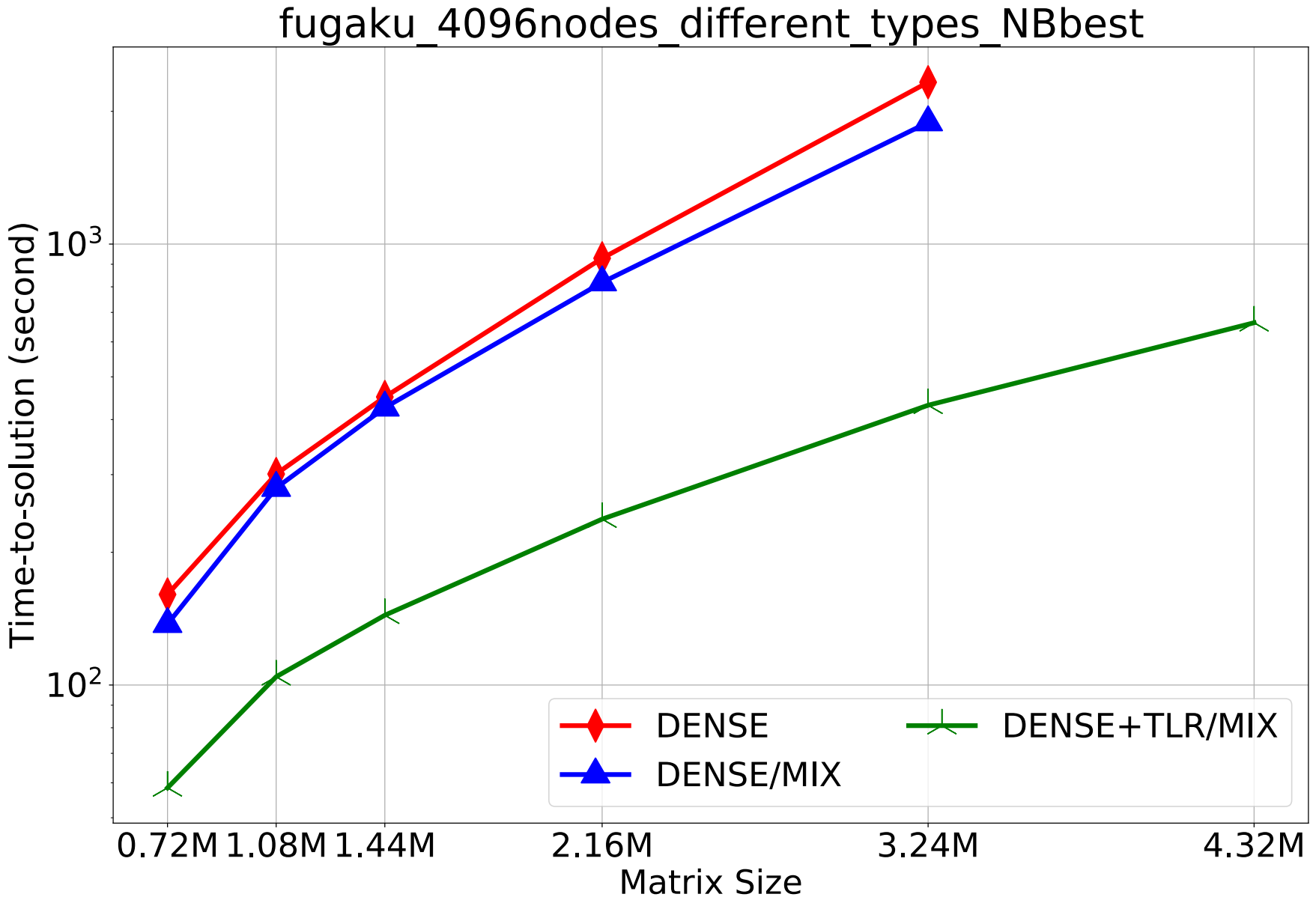


T. Mary, PhD Dissertation, Block Low-Rank multifrontal solvers: complexity, performance, and scalability, 2017.
C. Weisberger, PhD Dissertation, Improving multifrontal solvers by means of algebraic Block Low-Rank representations, 2013.

Directed Acyclic Graph – Cholesky Factorization



Incremental performance improvements on Fugaku



SC22 GB finalist

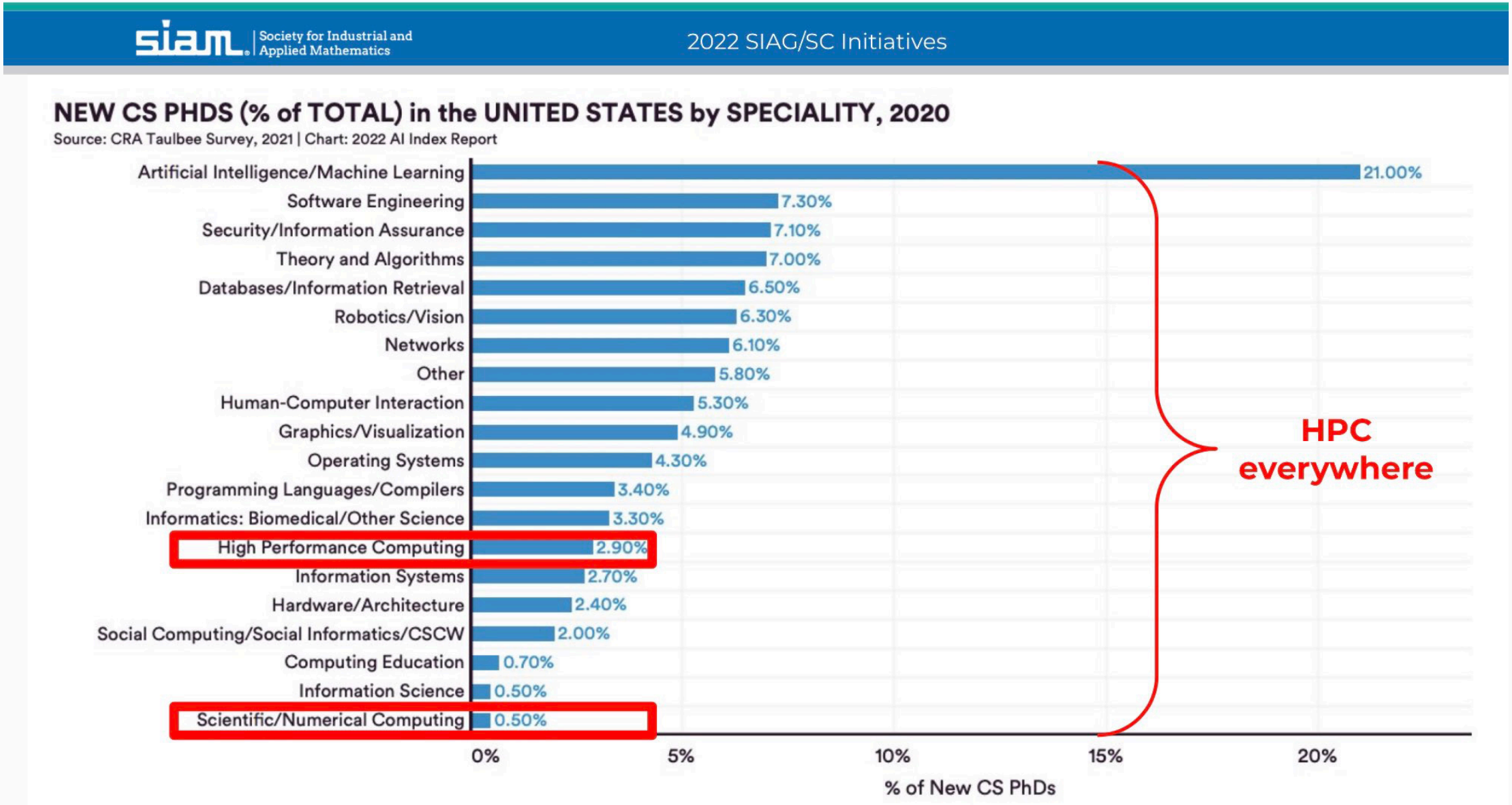
Most Time-Consuming Kernel: HCORE-TLR-GEMM

- Perform matrix-matrix multiplication on compressed tile low-rank data structures
- Call successive BLAS/LAPACK kernels, i.e., QR, RandSVD, GEMM
- Require recompression to restore the low rankness property of the tiles
- Encapsulate all these operations in a single HCORE-TLR-GEMM kernel call
- Limit opportunities for performance analysis and improvements

Challenges with SYCL

- The new rank calculation is only known when the HCORE-TLR-GEMM kernel is executed, which blocks the asynchronous execution, until the rank is known.
- HCORE-TLR-GEMM needs to be free from intermediate memory allocations.
- A significant redesign of the main kernel is necessary to expose the internal function calls in favor of a batch mode of execution.
- This is required anyway for getting performance on hardware accelerators.
- For memory buffers, it not clear how this can be done optimally.

Summary


siam Society for Industrial and Applied Mathematics
2022 SIAG/SC Initiatives
NEW CS PHDS (% of TOTAL) in the UNITED STATES by SPECIALITY, 2020
Source: CRA Taulbee Survey, 2021 | Chart: 2022 AI Index Report
Artificial Intelligence/Machine Learning 21.00%
Software Engineering 7.30%
Security/Information Assurance 7.10%
Theory and Algorithms 7.00%
Databases/Information Retrieval 6.50%
Robotics/Vision 6.30%
Networks 6.10%
Other 5.80%
Human-Computer Interaction 5.30%
Graphics/Visualization 4.90%
Operating Systems 4.30%
Programming Languages/Compilers 3.40%
Informatics: Biomedical/Other Science 3.30%
High Performance Computing 2.90%
Information Systems 2.70%
Hardware/Architecture 2.40%
Social Computing/Social Informatics/CSCW 2.00%
Computing Education 0.70%
Information Science 0.50%
Scientific/Numerical Computing 0.50%
HPC everywhere
0%
5%
10%
15%
20%
% of New CS PhDs

We are recruiting!