جامعة الملك عبدالله للعلوم والتقنية
King Abdullah University of Science and Technology

Extreme Computing Research Center

IXPUG
INTEL EXTREME PERFORMANCE USERS GROUP

# BEMFMM: An FMM-Accelerated Boundary Element Method-Based Solver for the 3D Helmholtz Equation

**Mustafa Abduljabbar**, Mohammed Al Farhan, Noha Al-Harthi, Rui Chen, Rio Yokota, Hakan Bagci, and David Keyes

April 24, 2018

King Abdullah University of Science and Technology (KAUST), Thuwal, Saudi Arabia
Tokyo Institute of Technology, Tokyo, Japan

# Highlights

- FMM-based solver for BEM discretizations of oscillatory operators.
  - Demonstrated for acoustics in Helmholtz formulation.
  - Scattering from a sphere (exact solution available).
- Three levels of parallelism: MPI + X + Y.
- Three contemporary Intel architectures: Haswell, Skylake, KNL.
- Up to 2 billion Degrees-of-Freedom and up to 0.2 million cores of Shaheen XC40.

# Outline

# Motivation

- Validating FMM-based horizontal and vertical parallelism techniques on modern many/multi-core architectures and 196,608 hardware cores of Shaheen XC40 supercomputer.
- Propose a performance model to estimate a near optimal granularity of recursive task creation during tree traversal.
- Describing the tradeoffs of using various modes of singularity treatment in the BEM.

# Problem Statement and Formulation [1/3]

- The time-dependent form of the wave equation is governed by the Helmholtz equation.

$$\nabla^2 U(r,t) - \frac{1}{c^2}\frac{\partial}{\partial t^2} U(r,t) = 0 \tag{1}$$

- The time-harmonic form as a result of plugging $U(r,t) = Re[U_0(r)e^{-jwt}]$ in Eq. 1 is:

$$\nabla^2 U_0(r) + k^2 U_0(r) = 0 \tag{2}$$

# Problem Statement and Formulation [2/3]

- The surface integral solution as a result of plugging the second form of Green's theorem looks like

$$p^{inc}(r) + \int_S [\frac{\partial G(r, r')}{\partial n'} p(r') - G(r, r') q(r')] dS' = \frac{1}{2} p(r), r \in S \tag{3}$$

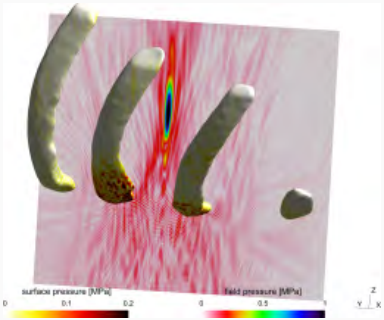
$$G(r, r') = \frac{e^{jkR}}{4\pi R} \tag{4}$$

- if we set $p = 0$, we obtain

$$\int_S G(r, r') q(r') = p^{inc}(r) \tag{5}$$

# Problem Statement and Formulation [3/3]

- Wave scattering and applications.




High-intensity focused ultrasound[1][2]

[1] T. Betcke, E. van 't Wout and P. Glat. *Computationally efficient boundary element methods for high-frequency Helmholtz problems in unbounded domains*, in: Modern Solvers for Helmholtz Problems, Springer (2017).

[2] E. van 't Wout, P. Glat, T. Betcke and S. Arridge. *A fast boundary element method for the scattering analysis of high-intensity focused ultrasound*, Journal of the Acoustical Society of America 138(5) (2015) pp2726-2737.

# System Overview

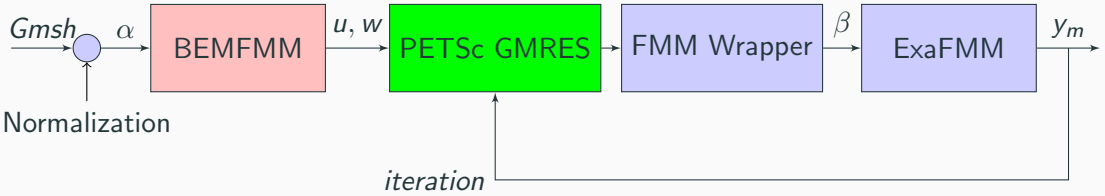
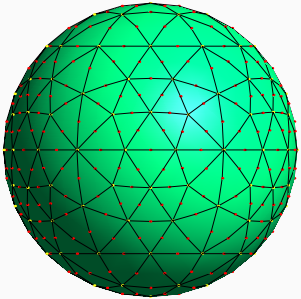



**Figure 1:** Dataflow across libraries (color-coded)

# Discretization of the Scatterer's Surface

- Divide the surface into curvilinear triangular patches.
- Each curvilinear patch has $N_i$ interpolation points.
- Using Nystrom method, The unknown velocity is expanded as an interpolation given by Equation:



$$q(r') = \sum_{n=1}^{N_p} \sum_{i=1}^{N_i} \vartheta^{-1}(r') L_{(i,n)}(\zeta, \eta) \{I\}_{(i,n)} \tag{6}$$

$$ZI = V^{inc} \tag{7}$$

$$[Z]_{(j,m)(i,n)} = \int_{\Delta n} G(r_{(j,m)}, r') \vartheta^{-1}(r') L_{(i,n)}(\zeta, \eta) dr' \tag{8}$$

# FMM in Numerical Science

- FMM looks for a solution that can be written as a BIE $u(x) = \int_\Gamma \Phi(x, y) q(y) dS(y)$ (to some extent)
- Examples of FMM use cases

| Application | Kernel | Single/Double |
|---|---|---|
| Gravitational, Potential | Laplace | Single |
| Electrostatic Field | Laplace | Double |
| Acoustics Scattering Field (Low Frequency) | Helmholtz | Single |
| Electromagnetic Scattering Field | Helmholtz | Double |

# FMM as an Accelerator for the 3D Helmholtz Solution

- FMM works as a matrix-free accelerator for the mat-vec multiplication (or IFMM as a pre-conditioner [Takahashi et al., 2017]).
- The resulting BIE results in a structured dense matrix.
- Example: impulse response due to a monopole source.

$$p(r) = \sum_{j}^{N_s} G(r, r')q(r') \tag{9}$$

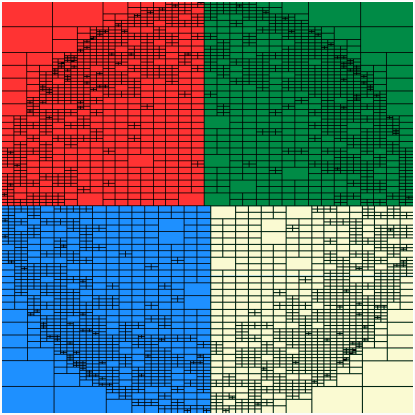
- Eq. 9 is expanded into a series of spherical harmonics.

$$p(r) = ik \sum_{n=0}^{\infty} \sum_{m=-n}^{n} S_n^{-m}(r_j) R_n^m(r), r \leq r_q \tag{10}$$

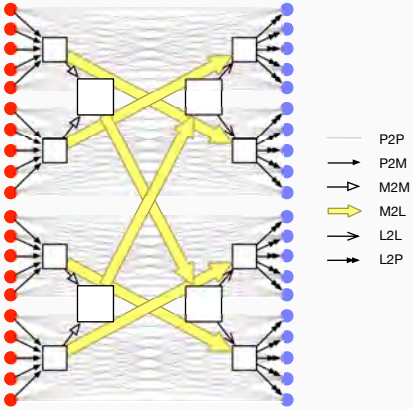
$$p(r) = ik \sum_{n=0}^{\infty} \sum_{m=-n}^{n} C_n^m R_n^m(r) \tag{11}$$

$$C_n^m = \sum_{r_q < R_{max}} Q_q S_n^{-m}(r_q) \tag{12}$$

# An Overview of FMM



Quad-tree partitioning on 4 processes

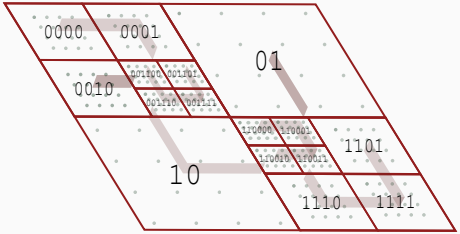



FMM Hierarchy (Upward, Horizontal, and Downward Sweeps)

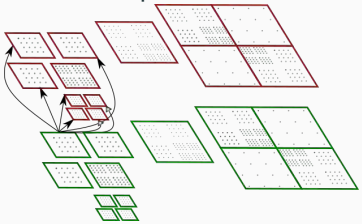
# Traversal Stage

Traditional FMM: Iterate over Hilbert/Morton orders and probe each cell for its neighbor [MS Warren et al., 1995].

Dual Tree Traversal (DTT)[3]: Simultaneously parse source and targets in a pre-order depth-first manner.




Warren and Salmon original FMM



Dual-tree traversal

[3] Mustafa Abduljabbar, Mohammed Al Farhan, Rio Yokota and David Keyes. *Performance Evaluation of Computation and Communication Kernels of the Fast Multipole Method on Intel Manycore Architecture*, Proceedings of the European Conference on Parallel Processing (2017) 553–564.

# P2P/M2L: Data-level Parallelism

Auto-vectorization with `#pragma simd` may almost always win in embarrassingly parallel kernels with structured memory access; however,

- Check icc compiler's output with -qopt-report. The default was the inner-most loop (suboptimal here).
- Double-check vector loads for array-of-structs.
- Rewrite divisions and complex numbers to their polar form (experimental 20% reduction in vector moves).

```
1  for ( ; i<ni; ++i) {
2    vi_r = real(Bi.SRC); vi_i = imag(Bi.SRC);
3    for (j=0; j<nj; ++j) {
4      dX = xi-xj; R2 = norm(dX);
5    \\relay self-sigularity to PETSc callback
6      if(Bi.PATCH!=Bj.PATCH && R2!=0) {
7        real_t R=sqrt(R2);
8        if(R<=near_patch_distance) {
9          for (k=0; k<gauss_quad_points; ++k) {
10         \\ near patch singularity treatment
11         }
12       } else {
13         vj_r = real(Bj.SRC); vj_i = imag(Bj.SRC);
14         src2_r = vi_r*vj_r-vi_i*vj_i;
15         src2_i = vi_r*vj_i+vi_i*vj_r;
16         invR = 1.0/sqrt(R);
17         eikr = 1.0/exp(wave_i*R);
18         eikr *= invR;
19         eikr_r = cos(wave_r*R)*eikr;
20         eikr_i = sin(wave_r*R)*eikr;
21         pot_r += src2_r*eikr_r-src2_i*eikr_i;
22         pot_i += src2_r*eikr_i+src2_i*eikr_r;
23       }
24     }
25   }
26   Bi.TRG += complex(pot_r, pot_i);
27 }
```

# Traversal: Thread-level Parallelism

- Control cell size and task granularity by minimizing the difference between LLC and the interacting cell sizes (minimize cache-miss rate).

$$\begin{aligned} \min_{s,c} f(s,c) &= (M2L_{size} + P2P_{size}) - \text{L2/L3 Cache} \\ &= 2 \times \text{ c } \times \text{ task size } \times \text{ nthreads/core} \\ &\times [(csize \times \log \frac{s}{c}) + bsize] - \text{L2/L3 Cache} \end{aligned} \tag{13}$$

- Multiplier "2" is inclusive of source and target.
- *csize*/*bsize* is the cell/body struct size.
- $s$ is the task spawning parameter.
- $c$ is the number of bodies per leaf cell.
- $\log \frac{s}{c}$ is the depth of recursive branch.
- L2/L3 Last Level Cache (LLC) size.

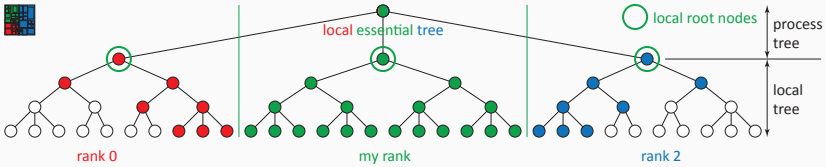
**Algorithm 2:** Interact($C_i$, $C_j$)

```
1  if Ci and Cj are leafs then
2  |  P2P(Ci,Cj)
3  else
4  |  if Ci and Cj satisfy MAC then
5  |  |  M2L(Ci,Cj)
6  |  else
7  |  |  if SizeOf (Ci,Cj)> nspawn then
8  |  |  |  Spawn (DualTreeTraversal (Ci,Cj))
9  |  |  else
10 |  |  |  DualTreeTraversal (Ci,Cj)
```

# Large Mesh Partitioning

- Pre-partitioning:
  - Create intermediate format with minimal indexed binary data (double-precision coordinates).
  - Map each rank to its region in the file.

- Partitioning:
  - Separate Global/Local trees using modified ORB [Abduljabbar et al., 2017].
  - Graft tree in one step when doing global traversal.




LET Grafting

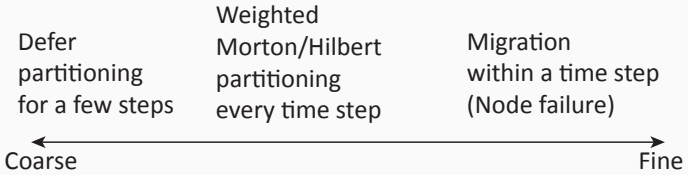
# Load-balancing (Repartitioning)

- Load-imbalance harms GMRES performance due to long wait on reduction of each step.
- Optimal partitioning for random distributions is NP-hard.
- Consider previous work-load and **communication**.

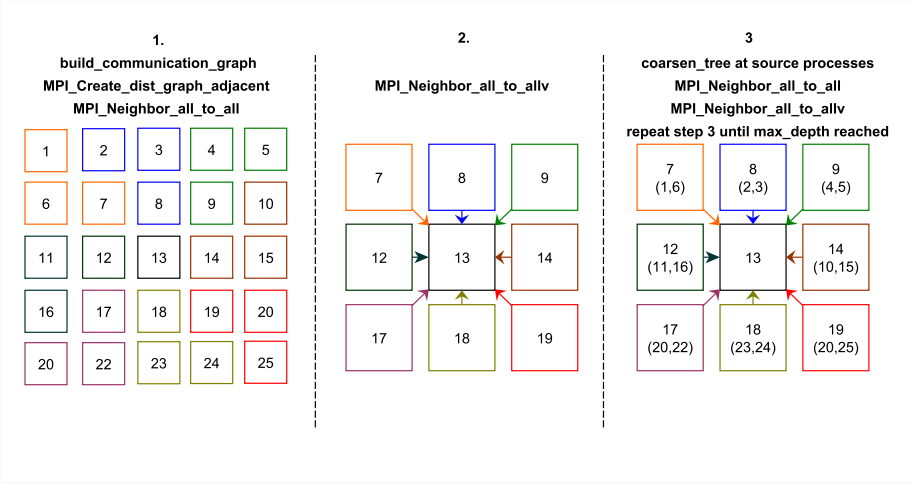
$$w_i = l_i + \alpha * r_i \tag{14}$$

- The variables $l_i$ and $r_i$, and the total runtime are already measured in the present code, so the information is available with negligible cost.

- Control granularity of partitioning based on load-imbalance.



# A Neighborhood-based Communication Protocol ($\mathcal{HSDX}$) [4]

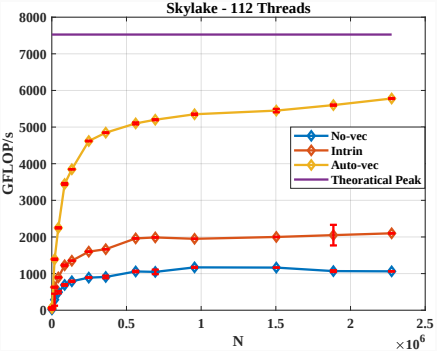


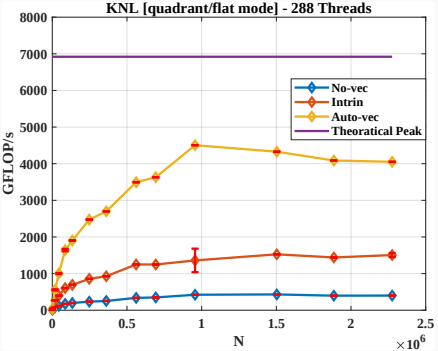

[4] Mustafa Abduljabbar, George S. Markomanolis, Huda Ibeid, Rio Yokota and David Keyes. *Communication Reducing Algorithms for Distributed Hierarchical N-Body Problems with Boundary Distributions*, Proceedings of the International Supercomputing Conference (2017), 79–96.

# Hardware Specifications

| | KNL | Haswell | Skylake |
|---|---|---|---|
| Family | x200 | E5V3 | Scalable |
| Model | 7290 | 2670 | 8176 |
| Socket(s) | 1 | 2 | 2 |
| Cores | 72 | 32 | 56 |
| GHz | 1.50 | 2.60 | 2.10 |
| Watts/socket | 245 | 120 | 165 |
| DDR4 (GB) | 192 | 128 | 264 |
| Frequency Driver | acpi-cpufreq | acpi-cpufreq | acpi-cpufreq |
| Max GHz | 1.50 | 2.60 | 2.10 |
| Governor | conservative | performance | ondemand |
| Turbo Boost | ✓ | ✓ | ✓ |

# Data-level Parallelism

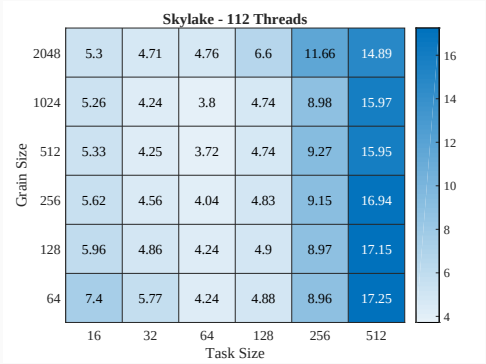


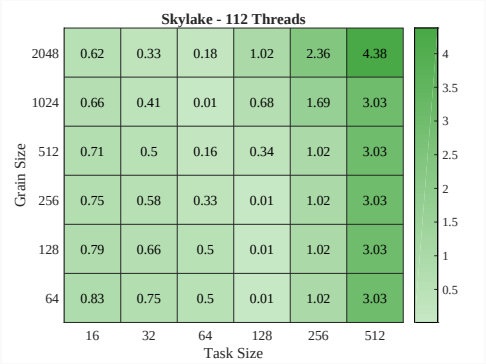




Single Precision Floating Point Performance

- Lower latency and higher throughput for AVX512 in Skylake.
- Increased cache miss penalty in KNL.

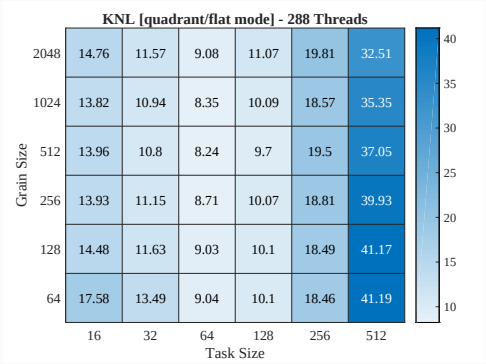
# Thread-level Parallelism [1/2]

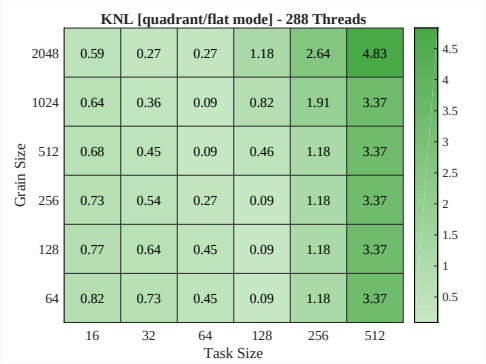



Experimental results vs. performance model on Skylake

# Thread-level Parallelism [2/2]




Experimental results vs. performance model on KNL

# Cray XC40 Characteristics

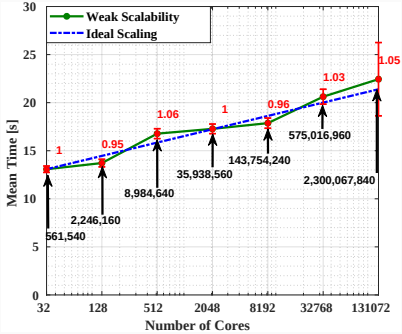
- Distributed experiments are run on Shaheen XC40, which hosts 196,608 Haswell cores, and has a linpack performance of 5.5 PFlop/s.
- Strong scaling is challenging in traditional FMM codes.

XC40 Network Hierarchy

| Level | Hardware/Network Unit | Nodes | Cores | Overhead | Hops |
|---|---|---|---|---|---|
| 1 | Socket | 1 | 16 | 32 | N/A |
| 2 | NUMA Node | 1 | 32 | 64 | N/A |
| 3 | Blade | 4 | 128 | 256 | 1 |
| 4 | Chassis | 64 | 2,048 | 4,096 | 1 |
| 5 | Cabinet | 192 | 6,144 | 8,192 | 1 |
| 6 | Local alltoall G | 384 | 12,288 | 16,384 | 1 |
| 7 | Global alltoall G1 | 2,304 | 74,728 | 131,072 | 2 |
| 8 | Global alltoall G2 | 6,174 | 197,568 | N/A | 3 |

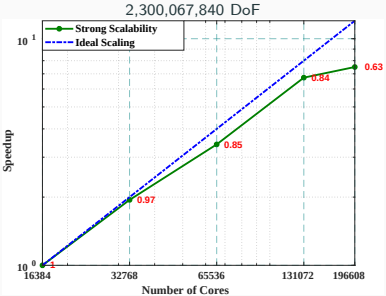
# Weak Scalability

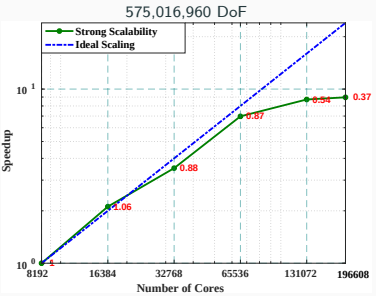
- A single scattering sphere, to validate the concept and the convergence.
- We fix the problem size while keeping 10 points per wavelength ($1.0e^{-4}$ accuracy vs. analytical).

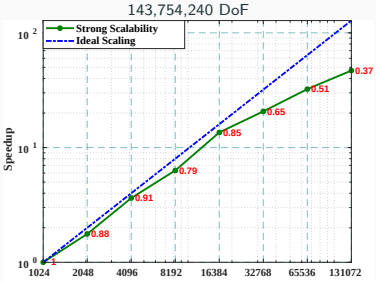



| Cores | Mem. [GB] | Freq. [KHz] | N (DoF) | T [S] | Iters |
|---|---|---|---|---|---|
| 512 | 2,048 | 24 | 8,984,640 | $3.1e3$ | 185 |
| 2,048 | 8,192 | 96 | 35,938,560 | $3.3e3$ | 190 |
| 8,192 | 32,768 | 384 | 143,754,240 | $3.6e3$ | 200 |
| 32,768 | 131,072 | 1,536 | 575,016,960 | $4.6e3$ | 223 |
| 131,072 | 524,288 | 6,144 | 2,300,067,840 | $5.7e3$ | 256 |

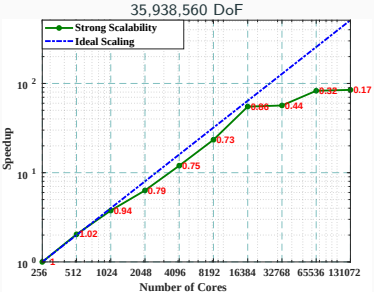
# Strong Scalability [1/2]

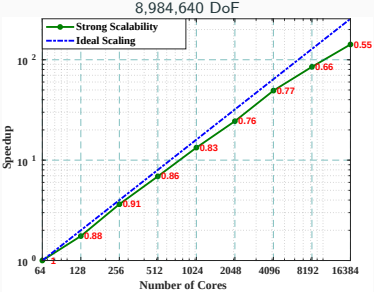


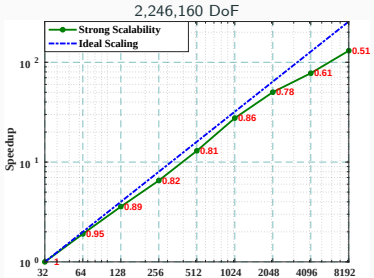



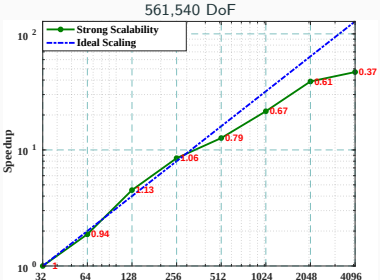



# Strong Scalability [2/2]

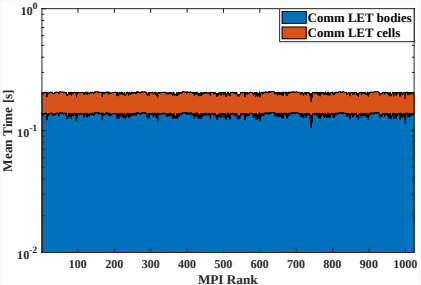


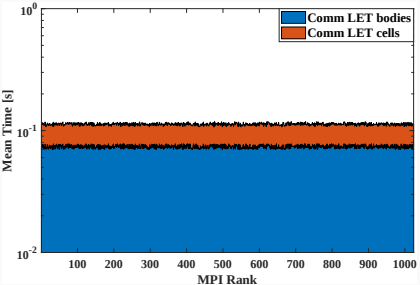



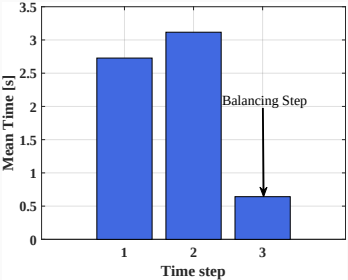



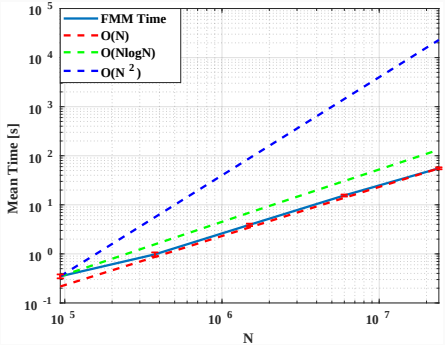
# Communication Load Balancing



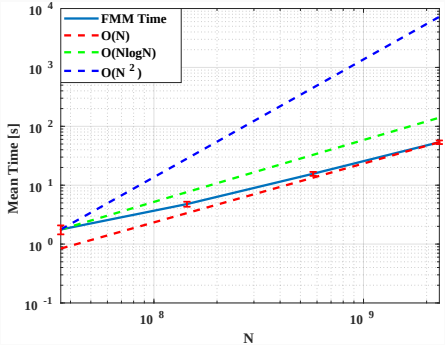



# Data Scalability

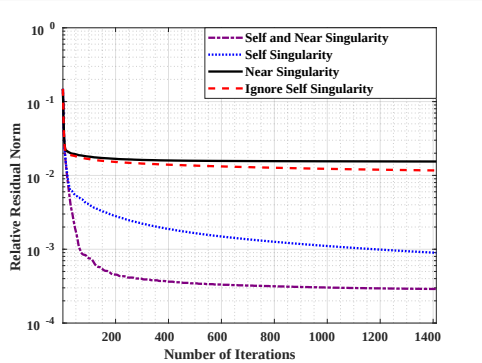



64 compute nodes of Shaheen




1,024 compute nodes of Shaheen

# Convergence Aspects and Numerical Error




Convergence behavior of 1m DoF using different singularity correction modes

# Conclusion

- Auto-vectorization of FMM kernels by the Intel C/C++ Compiler 2017 can achieve significant performance boost with given precautions.
- Heavy tuning parameters for recursive task parallelism of FMM's tree traversal can be avoided using our performance model with a small prediction penalty.
- Communication balancing results in 6 times faster global communication time for 100m DoF.
- Solution of a 2 billion DoF systems of high-order curvilinear triangular patches of a spherical mesh in about 60 minutes time-to-solution and an accuracy of $1.0e^{-4}$ compared to the analytical solution.
- Near-optimal parallel efficiency on Shaheen for both weak and strong scalability studies.

# Future Direction

- Explore and address performance challenges of more heterogeneous HPC architectures.
  - GPU-based supercomputers.
- Study different network topologies of various supercomputer architectures, and make $\mathcal{HSDX}$ aware of the underlying network units.
  - Intel Omni-Path network architecture.
- Further extend our solver implementation to include different highly nonuniform domains and more complex geometries.
  - Wing-fuselage configuration emulated by two intersecting ellipsoids.

Thank you