Center for Understandable Performant Exascale Communication Systems

THE UNIVERSITY OF NEW MEXICO

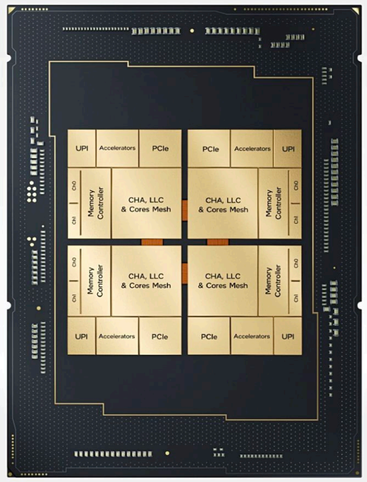
Understanding Data Movement on Many-Core Architectures

Jackson Wesley, Amanda Bienz | Scalable Solvers Lab

Introduction

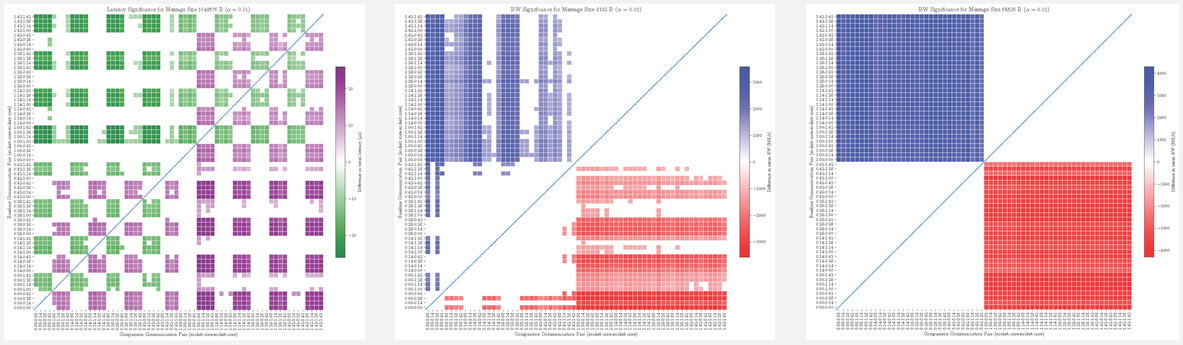
Initial research on many-core architectures such as Sapphire Rapids has shown interesting communication results. Our work analyzes these phenomenon closer with microbenchmarks and MPI tests over multiple Sapphire Rapids platforms. Specifically, we want to better understand if placement of communication within a node has an impact on internode communication.

We have collected microbenchmarking data from platforms at Los Alamos National Laboratory, Sandia National Laboratories, and Lawrence Livermore National Laboratory. Presented here are test results that we have obtained on Dane at Lawrence Livermore National Laboratory.



Single socket of a Sapphire Rapids architecture.

Results

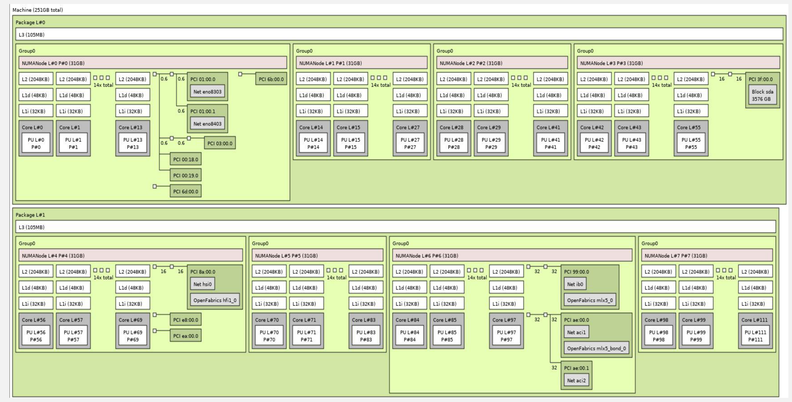


Left: Difference of latency at 1MiB; Center: Difference of bandwidth at 8KiB; Right: Difference of bandwidth at 65KiB

After running the benchmarks, we see common patterns in both latency and bandwidth. The heatmaps above display a few of the patterns that we have seen across platforms, each depending on the message size and type of measurement (bandwidth/latency). Every socket:core-socket:core communication pair is compared with every other communication pair by their mean in latency (green/purple) or bandwidth (blue/red).

Methods

We use one-sided bandwidth and latency tests from the OSU Microbenchmarks suite to generate our data about the architecture. Our scripts pin the host and client processes to cores on nodes, with the key distinction being to vary process by NUMA region and socket.

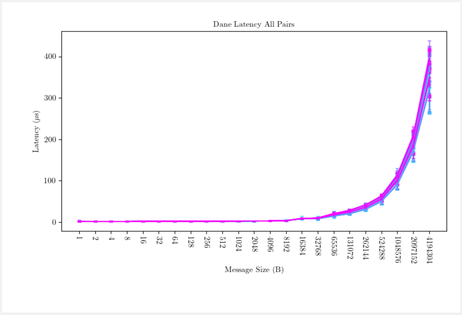


Topography of Dane (LLNL). Our microbenchmark method selects the first core within each NUMANode. Package indicates the socket that the group belongs to.

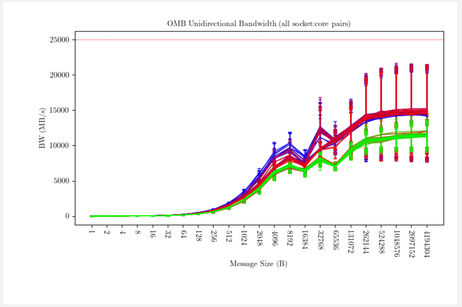
Conclusions

From the tests that we have run, we observe that there are differences in where you communicate from when using the Sapphire Rapids architectures. Looking at the patterns in the results, there is a trend in which socket 1 appears to be involved in scenarios that result in a higher latency and a lower bandwidth.

Our current work is to find ways to apply this knowledge. We are currently developing and testing an all-to-all hierarchical collective that will take advantage of relative on-node process locality to optimize where local data is best aggregated to.



Above: All pairs latency comparison
Below: All pairs bandwidth comparison



References

1. Bienz, Amanda, Schafer, Derek, and Skjellum, Anthony. 2023. MPI Advance: Open-Source Message Passing Optimizations. arXiv preprint arXiv:2309.07337.
2. Liu, J., Buntinas, D., Chandrasekaran, B., Yu, W., Kini, S., Wyckoff, P., Panda, D. K., & Wu, J. (2003). Micro-Benchmark Level Performance Comparison of High-Speed Cluster Interconnects. Computer and Information Science, The Ohio State University. Technical Report OSU-CISRC-5/03-TR30.

Acknowledgements

This work was performed with partial support from the National Science Foun-
dation under Grant No. CCF-2338077 and the U.S. Department of Energy's
National Nuclear Security Administration (NNSA) under the Predictive Science
Academic Alliance Program (PSAAP-III), Award DE-NA0003966.
Any opinions, findings, and conclusions or recommendations expressed in
this material are those of the authors and do not necessarily reflect the views of
the National Science Foundation and the U.S. Department of Energy's National
Nuclear Security Administration.

Center for Understandable Performant Exascale Communication Systems

Tennessee TECH

Leveraging Caliper and Benchpark to Analyze MPI Communication Patterns

Grace Nansamba, Evelyn Namugwanya, David Boehme, Olga Pearce, Riley Shipley, Derek Schafer, Anthony Skjellum

Introduction

- Understanding MPI communication patterns in HPC applications is important for optimizing HPC performance.
- Using Caliper's [2] special regions and profiling features, we analyze communication behaviors such as message volumes, sources/destinations, Irecv-gap, and communication patterns, such as halo exchanges in AMG2023, Kripke, and Laghos.
- We analyze communication patterns in Benchpark [1] applications and optimize performance based on the observed patterns.
- Our goal is to uncover bottlenecks and identify optimization strategies such as tuning message sizes and overlapping communication with computation to improve scalability and efficiency on both CPU and GPU systems.

Tools

- Benchpark is an open collaborative repository for reproducible specifications of HPC benchmarks. Benchpark enables cross-site collaboration by providing a mechanism for sharing reproducible, working specifications [1]. This software serves as the foundation for our enhancements, where we introduce new functionality to improve HPC application performance through detailed results analysis.
- Caliper is a program instrumentation and performance measurement framework. We used Caliper to annotate and profile halo-exchange regions in Benchpark applications [2].
- Thicket is a Python-based toolkit for analyzing ensemble performance data. We used Thicket to generate call trees and analyze results [3].

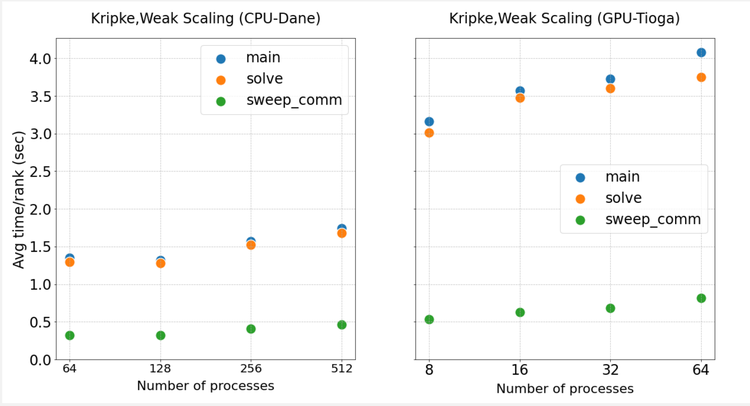
Applications

- Kripke is a 3D Sn deterministic particle transport code used to research the impact of data layout and programming paradigms on solver performance and parallelism [5].
- AMG2023 is a parallel algebraic multigrid solver for linear systems on unstructured grids, including a driver for 3D problems [4].
- Laghos is a high-order Lagrangian mini-app that solves compressible gas dynamics using unstructured meshes and explicit time integration. It supports full and partial matrix assembly, GPU acceleration, and high-performance operator application via MFEM [7] [8].

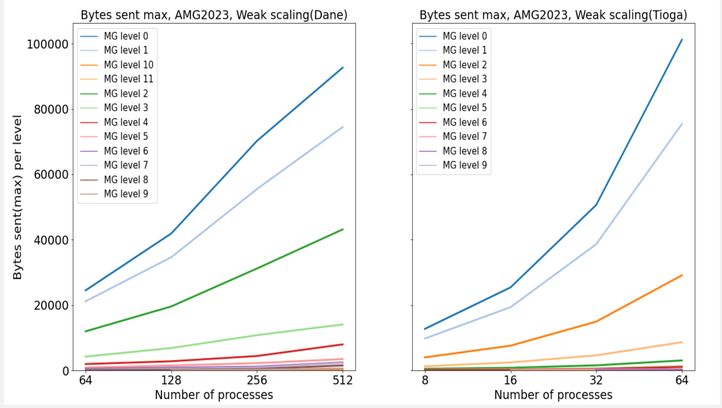
Methodology

- Extended Caliper with special regions to capture communication patterns, enabling profiling of send/receive volumes, message counts, and MPI timing data within defined computational regions.
- Added Irecv-gap revealing delays between posted and received non-blocking messages.
- Used the Caliper *modifier* (an abstract, reusable configuration object) in Benchpark to automate the capture of MPI communication patterns.
- Identified the communication regions in Kripke and AMG2023 (dependency on Hypre [6]), and Laghos(dependency on MFEM) and annotated with the new Caliper macros.

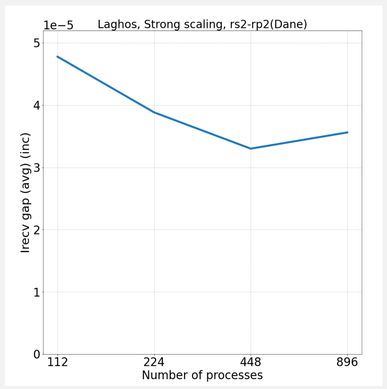
Results



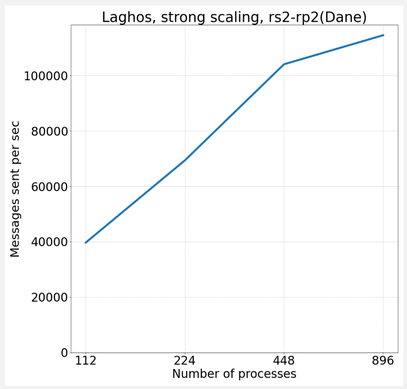
Avg time/rank for Kripke weak scaling on CPU and GPU



Bytes sent (max) per level, for AMG2023 weak scaling on CPU and GPU



Irecv gap (avg) (inc) for Laghos strong scaling on CPU



Messages sent per sec, Laghos strong scaling on CPU

Conclusions

- We extended Caliper with special regions to isolate and profile MPI communication patterns, enabling detailed analysis of halo exchanges, message sizes, and communication timing.
- Using Caliper + Benchpark + Thicket, we evaluated AMG2023, Kripke, and Laghos on CPU (Dane) and GPU (Tioga) systems, revealing architectural trade-offs in scalability and communication overhead.
- Distinct communication behaviors were observed across multigrid levels (AMG2023), particle sweeps (Kripke), and mesh updates (Laghos), allowing identification of bottlenecks and optimization opportunities.
- The Irecv-gap metric introduced a new way to measure non-blocking receive delays, aiding communication-computation overlap analysis.
- These insights inform optimization strategies, such as message size tuning, load balancing, and topology-aware mapping, supporting more efficient HPC application performance at scale.

References

1. Olga Pearce, Alec Scott, Gregory Becker, Riyaz Haque, Nathan Hanford, Stephanie Brink, Doug Jacobsen, Heidi Poxon, Jens Domke, and Todd Gamblin. 2023. Towards Collaborative Continuous Benchmarking for HPC. In Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis (SC-W 2023), November 12–17, 2023, Denver, CO, USA. ACM, New York, NY, USA, 9 pages.
2. D. Boehme, T. Gamblin, D. A. Beckingsale, P.T. Bremer, A. Gimenez, M. P. LeGendre, O. Pearce, and M. Schulz. Caliper: Performance introspection for hpc software stacks. SC16: International Conference for High Performance Computing, Networking, Storage and Analysis, 2016.
3. S. Brink, M. Mckinsey, D. Boehme, C. Scully-Allison, I. Lumsden, D. Hawkins, T. Burgess, V. Lama, J. Luttgau, K. E. Isaacs, M. Taufer, and O. Pearce. Thicket: Seeing the performance experiment forest for the individual run trees. Proceedings of the 32nd International Symposium on High-Performance Parallel and Distributed Computing, 2023.
4. U. Yang, D. Boehme, R.Li. AMG2023. https://github.com/LLNL/AMG2023, 2024.
5. A.J. Kunen, T.S. Bailey, and P.N. Brown. Kripke - a massively parallel transport mini-app. In American Nuclear Society M&C (April 2015).
6. Hypre. hypre: High Performance Preconditioners. https://llnl.gov/casc/hypre. https://github.com/hypre-space/hypre, 2024.
7. Laghos: https://github.com/CEED/Laghos
8. MFEM: https://github.com/mfem/mfem

Acknowledgments

- This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344 (LLNL-POST-867473).
- This work was performed with partial support from the National Science Foundation under Grants Nos. 2405142, 2412182, and the U.S. Department of Energy's National Nuclear Security Administration (NNSA) under the Predictive Science Academic Alliance Program (PSAAP-III), AwardDE-NA0003966.

Lawrence Livermore National Laboratory

Enhancing Benchpark by integrating new Caliper features to capture and analyze communication patterns for optimizing HPC application performance.

LLNL-POST-2005864

Center for Understandable Performant Exascale Communication Systems

THE UNIVERSITY OF NEW MEXICO

Vernier: Generalizing Communication Pattern Tracing

Derek Schafer[1], Riley Shipley[2], Patrick Bridges[1] | [1]University of New Mexico, [2]Tennessee Technological University

Motivation

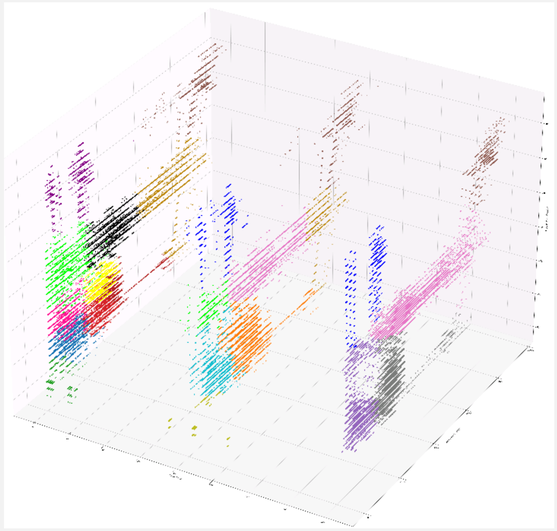
Current MPI tracing and logging tools provide:

- Frequency of each MPI call
- Time spent in MPI calls
- Time spent in annotated code regions

We had previously added manual, application-specific annotations to xRAGE, which allowed us to extract more detailed information such as:

- Detailed statistics about communication calls inside each pattern, such as per-rank distributions of peers and message sizes
- Changes in communication patterns over time
- Differences in communication calls carried out by each rank

This poster presents Vernier, our application-independent library encapsulating the approach we used in xRAGE.



Example data visualization using data captured from an xRAGE run. The axes feature message size, pattern frequency, and number of partners in the communication pattern.

Brief Comparison of Technical Details

	Manual	Vernier	Caliper
File Size	Large	Largest	Smallest
Data Captured	Most Specific	Fairly Specific	Most General
Per Rank?	Yes	Yes	Yes
Data Distributions?	Yes	Yes	No
Standard Interface?	No	Yes	Yes

Vernier Sample Output

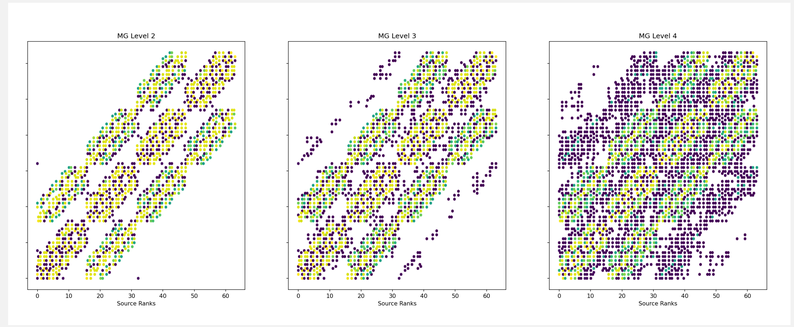
```
steady_state_comm:0,sweep_solver_comm:1
4|1{0},2{0},3{0;2},4{0;1},5{0},6{0;1;2;3},7{0;1;2;3}
0|*1*,AR:0:8:3,*1*,AR:0:8:3
<1>1|IR:0:2:131072:0,IR:0:1:131072:1,IR:0:2:131072:2,IR:0:1:131072:3, ...
<1>1|IR:0:2:131072:128,IR:0:1:131072:129,IR:0:2:131072:130, ...
```

Applications of Tracing Data

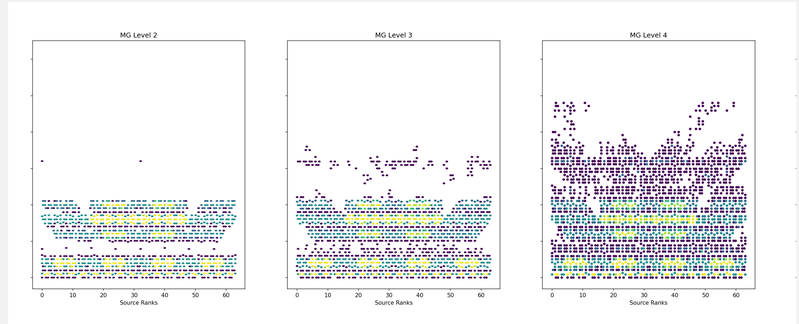
By using Vernier to capture communication patterns across an application, we can collect:

- Full traces of a given communication pattern (for a given problem)
- Detailed numerical analyses of each communication region (see figures on right)
- Histograms of various metrics per rank, per region of code (such as number of neighbors or average message size to each rank)
- Rank participation metrics (for evaluation of more optimal collective layouts)

We are also working to feed this data into our irregular communication replication benchmark that works off these distributions to re-create the communication pattern.



The above communication heatmap, created from AMG2023 data, plots the frequency of communication between each pair of ranks. White means no messages, and yellow indicates a larger number of messages.



Another visualization of AMG2023 data. Here, the y-axis has been changed to show the "distance" between the source and destination ranks. Colors maps are the same as the above graph.

Summary

Vernier tries to walk the balance between collecting necessary data and being application independent. In contrast to other tools, Vernier captures the unaltered attributes for each occurrence of the desired MPI calls across all ranks. This data provides deeper insights into the underlying communication patterns within an application and can be applied to optimize the communication structures (e.g., use of more optimal MPI operations). Similarly, communication pattern can be extracted into a smaller micro-benchmark that re-creates the pattern for more extensive study. In the future, we envision this approach being possible from Caliper itself.

Acknowledgements

This work was performed with partial support from the U.S. Department of Energy's National Nuclear Security Administration (NNSA) under the Predictive Science Academic Alliance Program (PSAAP-III), Award DE-NA0003966. We'd also like to thank Jered Dominguez Trujillo, Nick Bacon, and Ryan Marshall for helping us squash bugs in the code! Lastly, we'd like to thank David Boehme for sharing his experiences and technical expertise from Caliper.

Center for Understandable Performant Exascale Communication Systems

THE UNIVERSITY OF NEW MEXICO

Analyzing and Predicting Irregular Communication Performance Improvements

Nicholas Bacon and Patrick Bridges | Department of Computer Science

Research Topic

Irregular communication is common modern HPC applications, but expensive, hard to quantify, and hard to optimize. The goal of this research is to develop tools to quantify, model, and predict the performance of various irregular communication primitives

Analyzing and Modeling Costs of Irregular Data Movement

MPI's derived datatypes were developed to support to ease programmability and allow the communication system to efficiently send, receive, and compute on (e.g., reduce) complex irregular data structures. Even highly-optimized versions of MPI's derived datatypes have wildly-varying performances when using realistic application data structures on modern GPU-based systems. In our initial tests, even highly-optimized datatype engines such as MPICH/Yaksa, TEMPI, and MVAPICH often performed significantly (5%-50%) worse than simple Kokkos application data

To understand this better, we modified the existing Netgauge communication performance measurement tool and the LogGOPS model to characterize the communication behavior of modern hardware and MPI implementations. With these tools we were able to evaluate and compare the performance of different MPIs, systems and data layouts. Our results showed that the LogGOP model is often unable to capture the complexity of irregular data communication on modern systems.

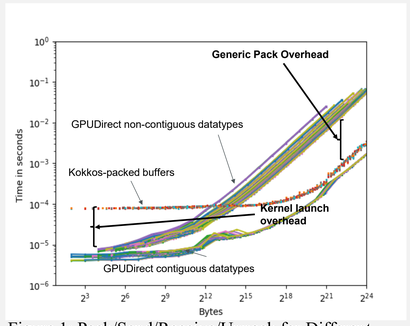


Figure 1: Pack/Send/Receive/Unpack for Different Contiguous and Non-Contiguous Data on LLNL Lassen with MVAPICH

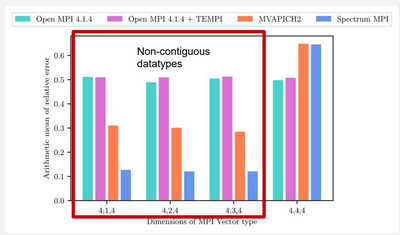


Figure 2: LogGOP accuracy with non-contiguous (red box) and contiguous datatype ping pong latency averaged across all buffer sizes

Using Caliper/Hatchet and LogGOPSim to Predict Network Speedup

We next examined using Caliper and Hatchet to predict how simple hardware and software changes would impact application performance, focusing on the irregular communication in the MiniAero, MiniFE, and AMG2023 benchmarks, using the workflow shown in Figure 3.

For this research, we tried to predict an infinite network speedup with drastically different amounts of success using Hatchet-based analysis of Caliper data. Our initial approach was to simulate an infinitely-fast network by removing all MPI communication costs. This oversimplification removed synchronization costs, resulting in over 100% speedup at larger node counts. We then studied revised approaches that examined removing portions of the minimum, maximum and average MPI communication times, and compared the results to LogGOPSim-based simulations of dramatically increased network performance.

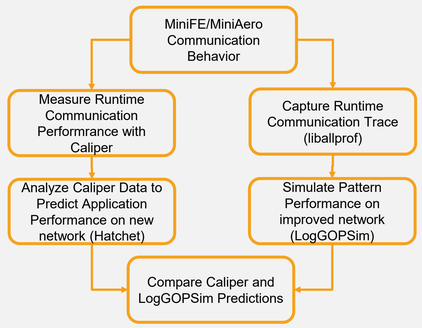


Figure 3: Workflow for comparing Caliper/Hatchet and LogGOPSim Performance Predictions

Caliper/Hatchet and LogGOPSim Prediction Results

Using this approach, we compared Caliper/Hatchet and LogGOPSim predictions by running MiniFE and MiniAero on the Sandia Glinda GPU system. These runs collected Caliper data for MiniFE for analysis and LogGOPSim traces for simulation, enabling us to compare the predictions of the two systems. As shown in Figure 4, our results showed that a simpler Hatchet-based analysis that removed the per-rank minimum MPI time from halo exchanges resulted in improved Caliper prediction of the impact of network improvements on application performance, but also demonstrated the difficulty of generalizing this approach and comparing Caliper and LogGOPSim performance predictions.

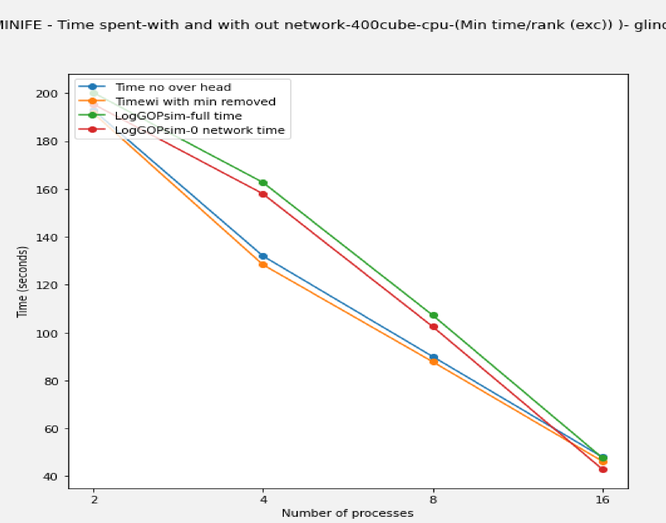


Figure 4: Comparison of Hatchet Analysis and LogGOPSim prediction of infinite bandwidth network in the performance of MiniAero, using data collected from the Sandia Glinda testbed

Predicting Optimization Impact on Production Applications

Based on lessons from the previous studies, we are now examining an improved workflow for predicting the impact of neighbor discovery and neighbor exchange primitives on Sparse Matrix/Vector Multiply (SpMV) operations, with the goal of being able to predict how such optimizations would improve the performance of codes like AMG2023, MueLU, and MiniEM. This approach uses a combination of Vernier pattern capture, a Cabana-based irregular communication benchmark, and Python-based analysis scripts to predict the performance impact of these optimizations, using the workflow shown in Figure 5. We have implemented the irregular pattern benchmark and are currently gathering data from the SpMV benchmark using Vernier.

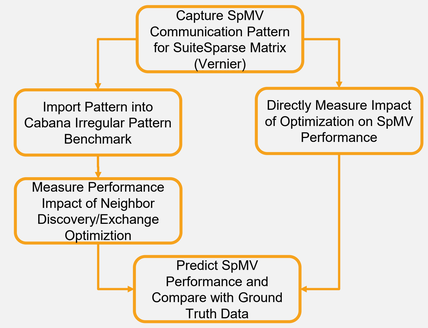


Figure 5: Workflow for estimating impact of neighbor discovery and exchange primitives on SpMV performance

Acknowledgements

- National Science Foundation OAC-2103510
- U.S. Dept. of Energy Award DE-NA0003966
- Sandia National Laboratories Mentors Scott Levy and Kurt Ferreira

Center for Understandable Performant Exascale Communication Systems

THE UNIVERSITY OF NEW MEXICO

Adding Communication Benchmarks to the Benchparks Experiment Managment Framework

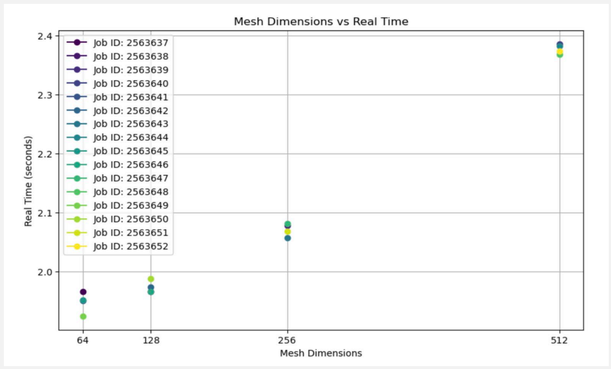
Abdalaziz Raad and Patrick Bridges | Dept. of Computer Science

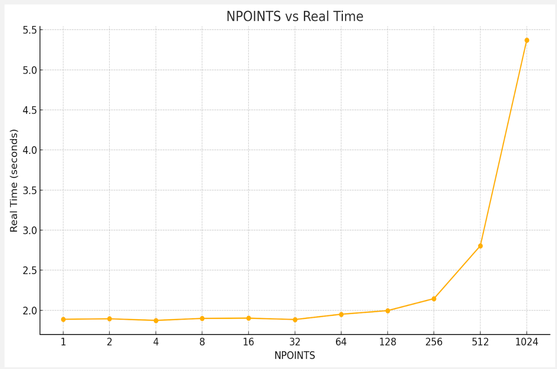
Motivation

Goal: Demonstrate the ability to manage complex communication experiments

- Use **Beatnik** as driving benchmark, including building, running, benchmarking, and analyzing data
- Initially began with **ReFrame**, but we ran into several bottlenecks. While ReFrame supports environment modules and system configurations, it doesn't offer fine-grained experiment tracking—such as linking inputs, outputs, configuration versions, and metadata. ReFrame can collect output, but you have to write additional logic to extract and store performance metrics like runtime and memory usage.
- Recently transitioned to use **BenchPark.** BenchPark makes it easier to:
  - Run and manage performance benchmarks by providing a structured framework for defining, executing, and analyzing application workloads.
  - Integrates seamlessly with **Ramble** for benchmark execution.
  - Ensures reproducibility by consistently tracking all experimental configurations, outputs, and performance metrics in an organized and automated way.

Results





Goal: Benchmark Beatnik to understand how performance scales with input size by collecting initial performance data by manually building and running **Beatnik**, before using ReFrame or BenchPark

- **Left graph**: Shows run time vs. **mesh size** (64 to 512)
  - Run time increases slightly (1.9 to 2.4 seconds)
  - Indicates Beatnik scales well with larger meshes
  - Tightly grouped dots show consistent results across runs
- **Right graph**: Shows run time vs. **NPOINTS** (system size)
  - Run time stays steady under 2 seconds for NPOINTS 1–128
  - Increases rapidly beyond 128, reaching over 5 seconds at 1024
  - Suggests higher computational demand with larger inputs

Current Work

Goal: Make Beatnik easy to run and benchmark across multiple systems using Ramble and BenchPark

- Got **STREAM** and **Kripke** working in BenchPark by setting them up through **Ramble**, which made it easy to define how each application should run and what parameters to vary. Both applications are both included in the **BenchPark** application library, I was able to start with existing configurations and customize them to fit my benchmarking needs.
- Integrating **Beatnik**, a small HPC application designed to test communication patterns and simulate fluid dynamics.
- Using **Ramble** to define how Beatnik is built, how its parameters are configured, and how jobs are submitted to the system.
  - Ramble automates environment setup and job control, ensuring that each run is consistent and repeatable.
  - **BenchPark** stores the performance data in a structured and searchable format and tracks results, compares performance across different setups.

Future Work and Bigger Picture

Goal: Create a portable, reusable configuration for running Beatnik with Ramble and BenchPark

- Developing a full configuration that defines how **Beatnik** runs, which parameters can be adjusted (such as mesh size or number of points), and how performance data is collected from each run
- Setting up the workflow so **Beatnik** can be easily executed across different systems without needing to manually edit scripts or settings
- Ensuring compatibility with any system that supports **Ramble** and **BenchPark**, regardless of hardware or environment
- Running tests on multiple platforms—including personal machines, university clusters, and HPC systems—to verify portability and consistency
- Planning to contribute the completed configuration to the **BenchPark** application library to make Beatnik readily available for the community
- This will make it easier for researchers and developers to benchmark Beatnik, compare performance across platforms, and use it in larger automated workflows

Acknowledgements

- Ryan Marshall for collaboration with ReFrame.
- Jason Stewart for helping me with understanding ramble.
- Grace Nansamba for helping me with getting Kripke running through benchpark.
- The U.S. Department of Energy's National Nuclear Security Administration (NNSA) under the Predictive Science Academic Alliance Program (PSAAP-III), Award DE-NA0003966.

Center for Understandable Performant Exascale Communication Systems

Scaling Laws for the Workload Throughput of Emerging Heterogeneous Clusters

Akhil Alasandagutti, Patrick Bridges, Trilce Estrada | UNM Computer Science
Joshua Suetterlein, Jesun Firoz, Stephen Young, Joseph Manzano, Kevin Barker | Pacific Northwest National Laboratory

Introduction

In the post-Moore's law era, the rapid evolution of high-performance computing (HPC) architectures is increasingly driven by the emergence of diverse and complex workloads that comprise a mixture of computationally intensive and communication-bound application, including, but not limited to advanced simulation, machine learning, and graph analytics. As transistor scaling slows, today's HPC systems incorporate heterogeneous components, such as GPUs and FPGAs, to continue to deliver performance improvements. The next generation of HPC systems will likely integrate a broader array of specialized accelerators such as quantum processors [1], neuromorphic accelerators [2], and dataflow accelerators [3], all co-located within the same HPC cluster [4]. To maximize their potential, these next-generation HPC systems must effectively integrate diverse accelerators with classical computing while addressing job scheduling, system utilization, and resource allocation.

To address these challenges, we propose a new analytical model for understanding the performance of workloads on an HPC cluster incorporating multiple co-located accelerators. Our model projects the improvement in throughput as accelerators are to the system, taking into account the following factors: whether the accelerators are shared or user exclusively, and the amount of time a job spends on classical computing units and accelerators.

Methods

- We extend the SST/macro simulation framework [5] to simulate an HPC system with multiple accelerators at the MPI application level.
- A driver application delegates one MPI rank for each accelerator and compute node.
- Driver application configuration file contains list of applications in the queue, including number of processes, scheduling priority, scheduling mode, and which accelerators they use.
- Two possible network topology configurations – Dragonfly, and Fat Tree. Application pool: MiniVite, ResNet-512, Cosmoflow, DLRM, Sweep3D, Halo3D, FFT.
- For each scheduling mode – exclusive and multitenant (shared accelerators), pre-determined job-type composition, accelerator task time (ATT), and accelerator count, we generate 50 random configuration files and simulate them on a real system and on SST.
- We then compute a γ for each scheduling mode, workload, and topology configuration as follows:

$$S = \frac{T_1}{T_2} \qquad \gamma = \frac{2S-2}{S}$$

  where T1 is the queue clear time of the workload on one accelerator, and T2 is the queue clear time on two accelerators.
- Projections from the analytical model are then compared to real system and SST runs.

Analytical Models for Job Drain Time and TCO

We now present an analytical model to estimate the speedup or slowdown of workloads in future heterogeneous HPC systems as the number of accelerators varies. These models offer insights into the trade-offs between total cost of ownership (TCO) for multi-tenant and exclusive scheduling modes, taking into account the composition of the workload.

$$\frac{1}{1-\gamma+\frac{1}{k}\gamma}$$

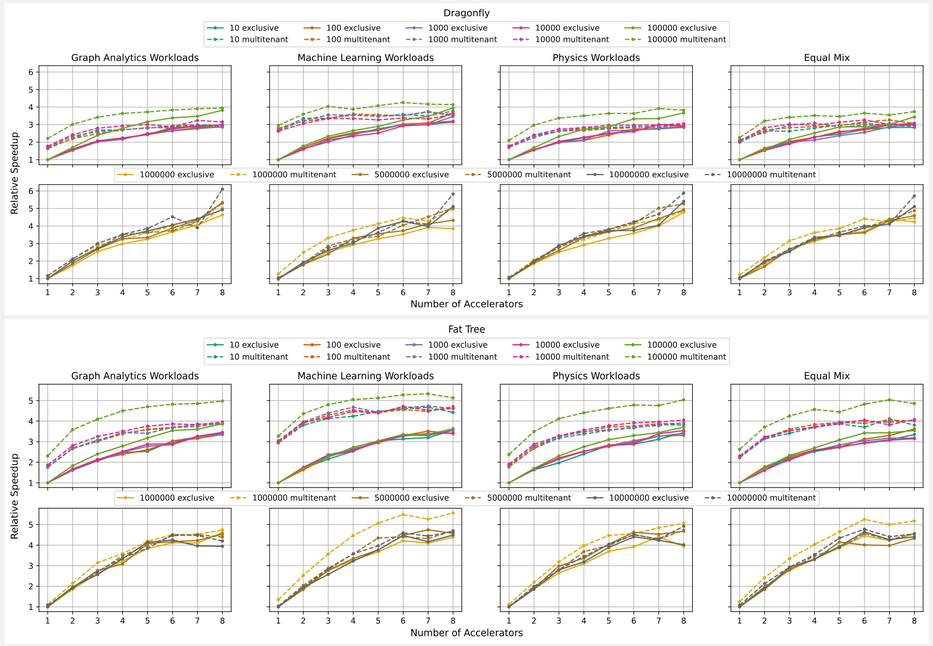
Where k is the total number of accelerators in the system, and γ is some workload and scheduler dependent parameter that quantifies the proportion of the overall workload which benefits from accelerator parallelism. We note that this speedup formulation can be used to identify the point of diminishing returns in terms of adding accelerators. Adding a new accelerator will increase the speedup by less than a factor of $(1+\epsilon)$ if the number of accelerators, k, satisfies:

$$\frac{1-\gamma+\frac{1}{k}\gamma}{1-\gamma+\frac{1}{k+1}\gamma} < (1+\epsilon)$$

Which can be solved algebraically as follows:

$$k = \frac{-1+\sqrt{1+4\frac{\gamma(1-\gamma)}{\epsilon}}}{2(1-\gamma)}$$

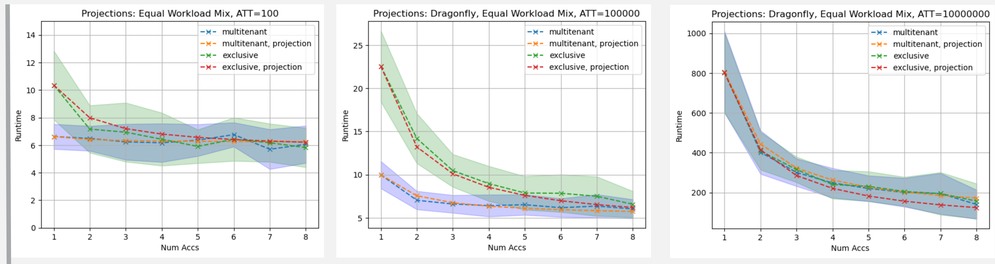
Observed Relative Speedups



Top: Relative Speedups of workloads with increasing number of accelerators and various Accelerator Task Times (ATTs), running on a system with a Dragonfly Topology.

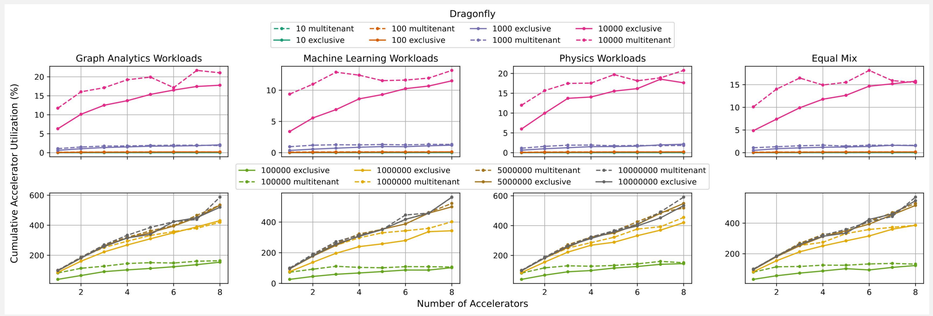
Bottom: Relative Speedups of workloads with increasing number of accelerators and various Accelerator Task Times (ATTs), running on a system with a Fat Tree Topology.

Speedup Projections



Analytical Model Projections vs Actual Runs on SST with varying ATTs on SST and Real Systems.

Accelerator Utilization



Cumulative accelerator utilization on Dragonfly. Percentages larger than 100% are indicative of usage saturating multiple accelerators.

Results and Discussion

- Low ATT workloads can achieve up to 2x improvement in throughput with multi-tenant scheduling while using a fraction of the accelerators. However, this advantage over exclusive mode diminishes with an increase in ATT.
- There is no observable difference in performance between Dragonfly and Fat Tree Topologies.
- Our analytical models predict performance well, with all of the predicted points falling within the confidence interval of 1 standard deviation of the observed runs.
- The performance spread for the real system runs is much more erratic, resulting in a larger standard deviation. Our analytical models however still predict performance just as well.
- This increased variability in the real system runs can be attributed to network interference and system effects not present in simulations.

Conclusion

Emerging HPC systems are increasing incorporating a variety of accelerators beyond traditional GPUs and FPGAs. To our knowledge, this is the first study to propose an analytical model for emerging multi-accelerator systems, demonstrating how various factors influence job performance. Our model aligns well with both simulation data and real-world experiments, showing slightly higher variance in real scenarios. This work paves the way for future research, including efforts to reduce prediction errors in the model and expand the testbed scale, enhancing our understanding of workload optimization and cluster balance.

Acknowledgements

This research was supported in part by an award from the National Science Foundation under grant numbers OAC-2103510 and OAC-1807563, and by the U.S. Department of Energy's National Nuclear Security Administration (NNSA) under the Predictive Science Academic Alliance Program (PSAAP-III), Award DE-NA0003966. This research used funding and resources supported by the U.S. DOE Office of Science, Office of Advanced Scientific Computing Research, under award 66150: ``CENATE - Center for Advanced Architecture Evaluation". The Pacific Northwest National Laboratory is operated by Battelle for the U.S. Department of Energy under contract DE-AC05-76RL01830. PNNL information release number: PNNL-SA-208649.

References

1. IBM, "Ibm debuts next-generation quantum processor, ibm quantum system two, extends roadmap to advance era of quantum utility," IBM News Room, 2023. [Online]. Available: https://newsroom.ibm.com/2023-12-04-IBM-Debuts-Next-GenerationQuantum-Processor-IBM-Quantum-System-Two,-Extends-Roadmap-to-Advance-Era-of-Quantum-Utility
2. "A new brain-based supercomputer could revolutionize scientific discovery," Sandia National Laboratories News Releases, 2023.
3. R. Prabhakar et al., "Sambanova sn40l: Scaling the ai memory wall with dataflow and composition of experts," arXiv:2405.07518, 2024.
4. RIKEN, "Riken selects ibm's next-generation quantum system to be integrated with the supercomputer fugaku," 2023. [Online]. Available: https://newsroom.ibm.com/2023-12-14-IBM-announcescollaboration-with-RIKEN-to-integrate-IBM-quantum-system-two-withsupercomputer-fugaku
5. J. Wilke et al., "Structural Simulation Toolkit (SST) Macroscale Element Library," https://github.com/sstsimulator/sst-macro, 2023, accessed: 2023-01-05.

Center for Understandable Performant Exascale Communication Systems

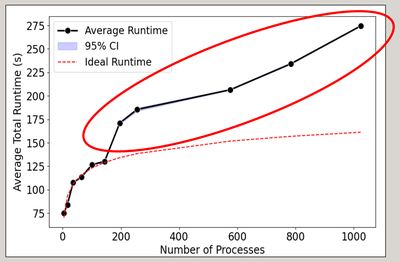
THE UNIVERSITY OF NEW MEXICO

Beatnik: A Prototype High Performance Parallel Interface Benchmark

Jason Stewart and Patrick Bridges | Computer Science

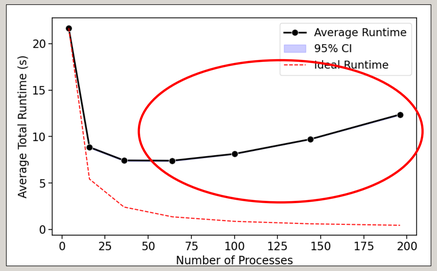
Benchmarking Results

Beatnik is limited by network bandwidth when weak scaling

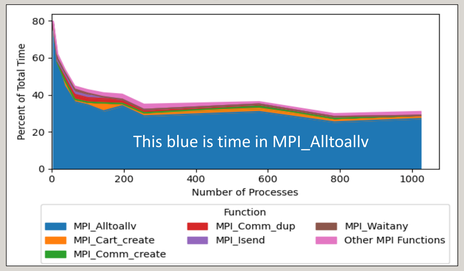


- Amount of data per GPU stays constant
- **Runtime initially follows log trend, but begins to grow linearly as network bandwidth limits are met**

Latencies associated with small all-to-all communication and small GPU FFTs form bottleneck when strong scaling

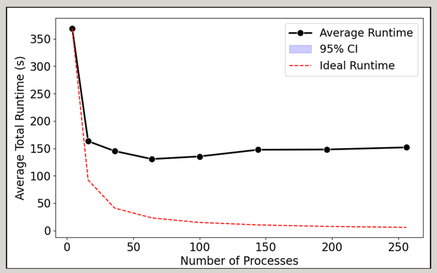


- Runtime increases as problem size remains constant
- **Startup latencies of GPU kernels form bottleneck**

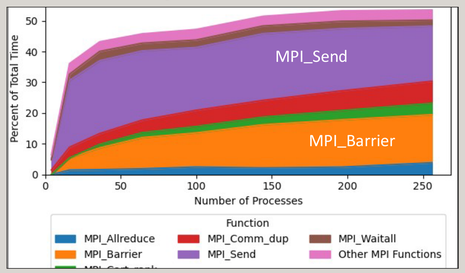


- **Percentage of time spent in MPI_Alltoallv and in compute stays the same as total runtime grows**
- Both compute and communication overheads are growing

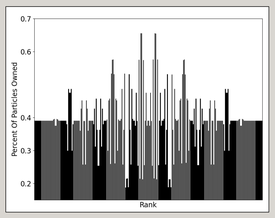
Cutoff solve strong scaling: Load imbalances develop as particles are pulled into the rollup



- Scaling from 4 to 64 GPUs reduces runtime by 3x
- **No further improvement with more GPUs**



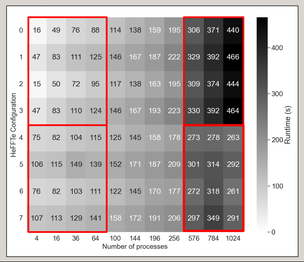
- MPI_Send and MPI_Barrier dominate total runtime
- **Some processes must halo and compute on more particles while other processes wait on them**



At timestep 340, with 256 processes:

- Min particles owned: **0.2% (~525)**
- Max particles owned: **0.65% (~1705)**

Beatnik highlights tradeoffs of HeFFTe parameter choice

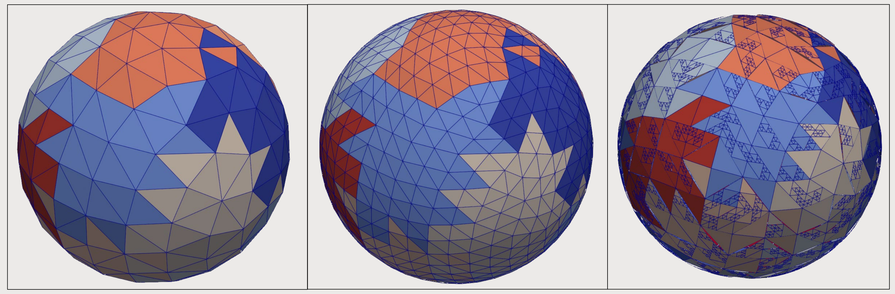


Parameter Settings

Configuration	AllToAll	Pencils	Reorder
0	False	False	False
1	False	False	True
2	False	True	False
3	False	True	True
4	True	False	False
5	True	False	True
6	True	True	False
7	True	True	True

- **On small numbers of processes, heFFTe performance is better when using its custom communication routines** and not using Spectrum MPI's MPI_Alltoall primitive.
- In contrast, on **large numbers of processes, heFFTe performance improves if the AllToAll parameter is true.**

Support for closed interface surfaces and adaptive mesh refinement



16 processes, no face refinement | **16 processes, uniform face refinement** | **16 processes, non-uniform face refinement**

Colors represent rank of ownership

- Enable research of communication patterns associated with dynamic, non-uniform meshes
- Data can be attached to any set of elements (vertices, edges, or faces)
- 1-deep element and data haloing
- Hierarchal structure of parent and child faces and edges

Future work

- Incorporate Z-Model mathematics into unstructured and adaptive mesh.
- Implement distributed, fast-multipole-like algorithm for far-field force calculations on the unstructured mesh, taking advantage of hierarchal structure.
- Implement dynamic load balancing of mesh
- Performance profiling using different communication spaces in Cabana

Acknowledgements

This work was funded by the PSAAP program. We would like to thank the UNM Center for Advanced Research Computing, and the Department of Energy for providing the research computing resources used in this work.