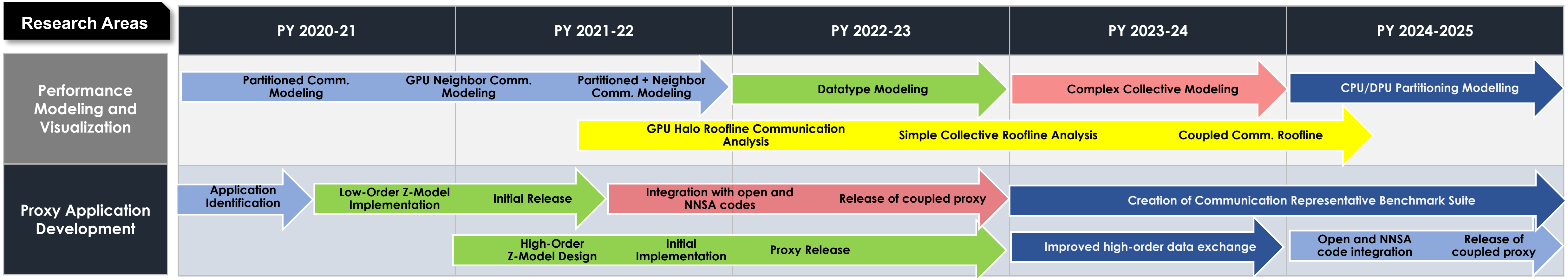
Benchmarking and Modeling

Prof. Patrick G. Bridges

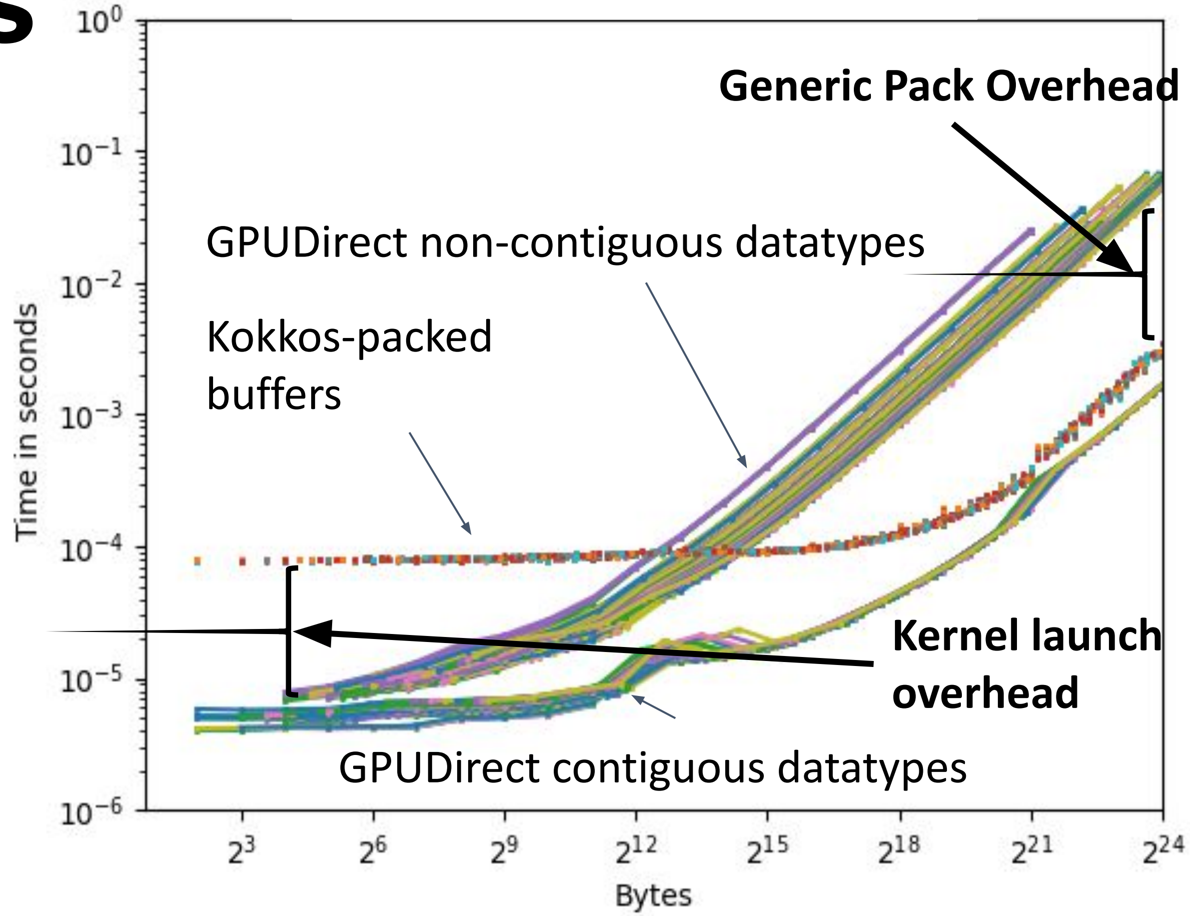
Roadmap for Modeling/Benchmarking



- Multiple modeling/assessment efforts
  - Key communication components: datatypes, simple collectives, scaling modeling
  - Irregular communication in production DOE applications
- Creating new global and coupled communication benchmark
- Proposed focus of last two years
  - Curated benchmark suite based on assessment and benchmarking results
  - Modeling CPU/GPU/DPU performance tradeoff for coupled fluid/interface benchmark

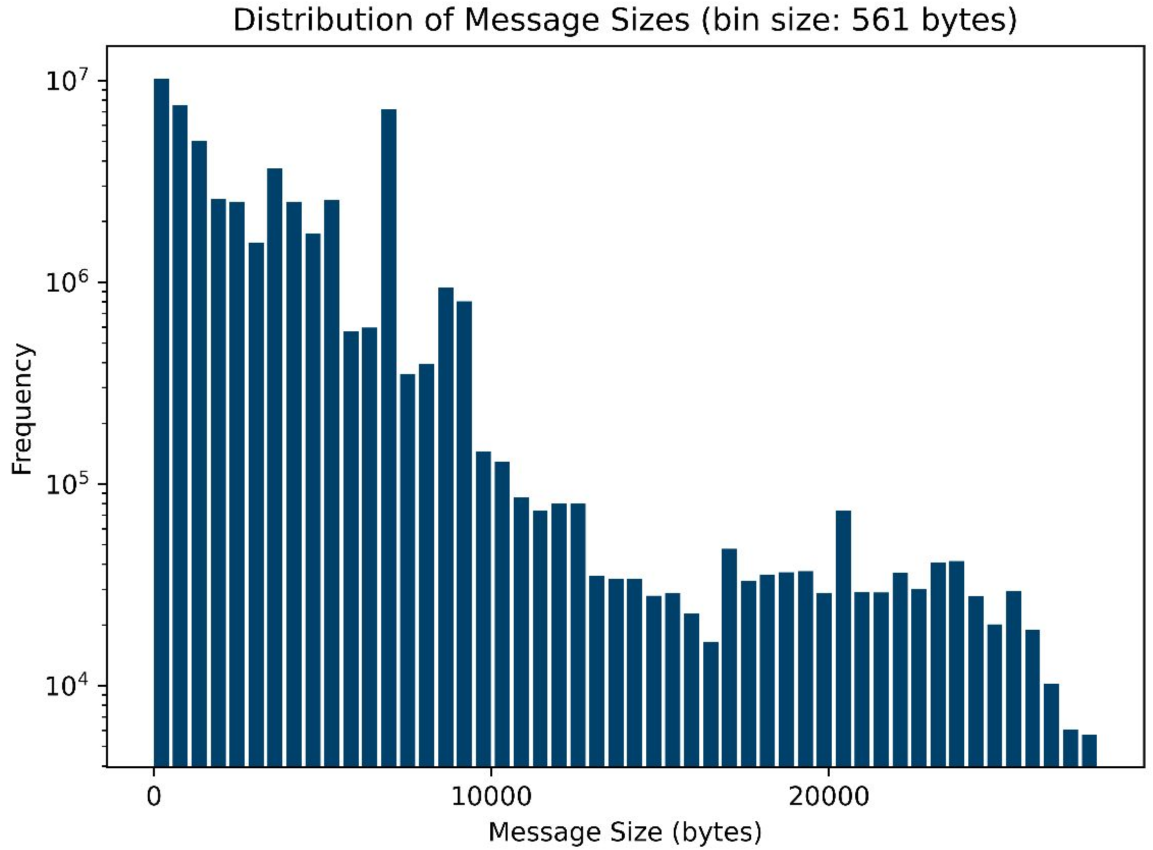
Modeling/Benchmarking Communication Primitives: Datatypes

- Needed to efficiently communicate and compute (e.g. reduce or gather) irregular data
- Measured data performance on Lassen with careful benchmarking
  - Generic datatypes can be faster for small irregular data (50 usec faster round trip)
  - Generic datatypes prohibitively slow for large irregular data (tens or hundreds of ms slower round trip!)
- Developed performance model to quantify GPU datatype behavior
  - LogP models effectiveness limited for GPUs
  - LogP models can quantify datatype implementation performance
  - See Nick Bacon's Poster, EuroMPI 2023 paper
- Working on new datatype abstraction to preserve advantages and eliminate disadvantages



Measuring and Modeling Irregular Communication

- Many applications rely on irregular communication
- This communication is not easy to measure or reproduce
- Created tools to extract irregular communication abstraction performance:
- Analyzing communication behaviors and optimization opportunities
- Creating benchmarks to *replay* these communication behaviors



Frequency of token P2P send by size on xRAGE Asteroid Problem (512 ranks, 50 bins, log Y scale)

Collecting Irregular Communication Patterns

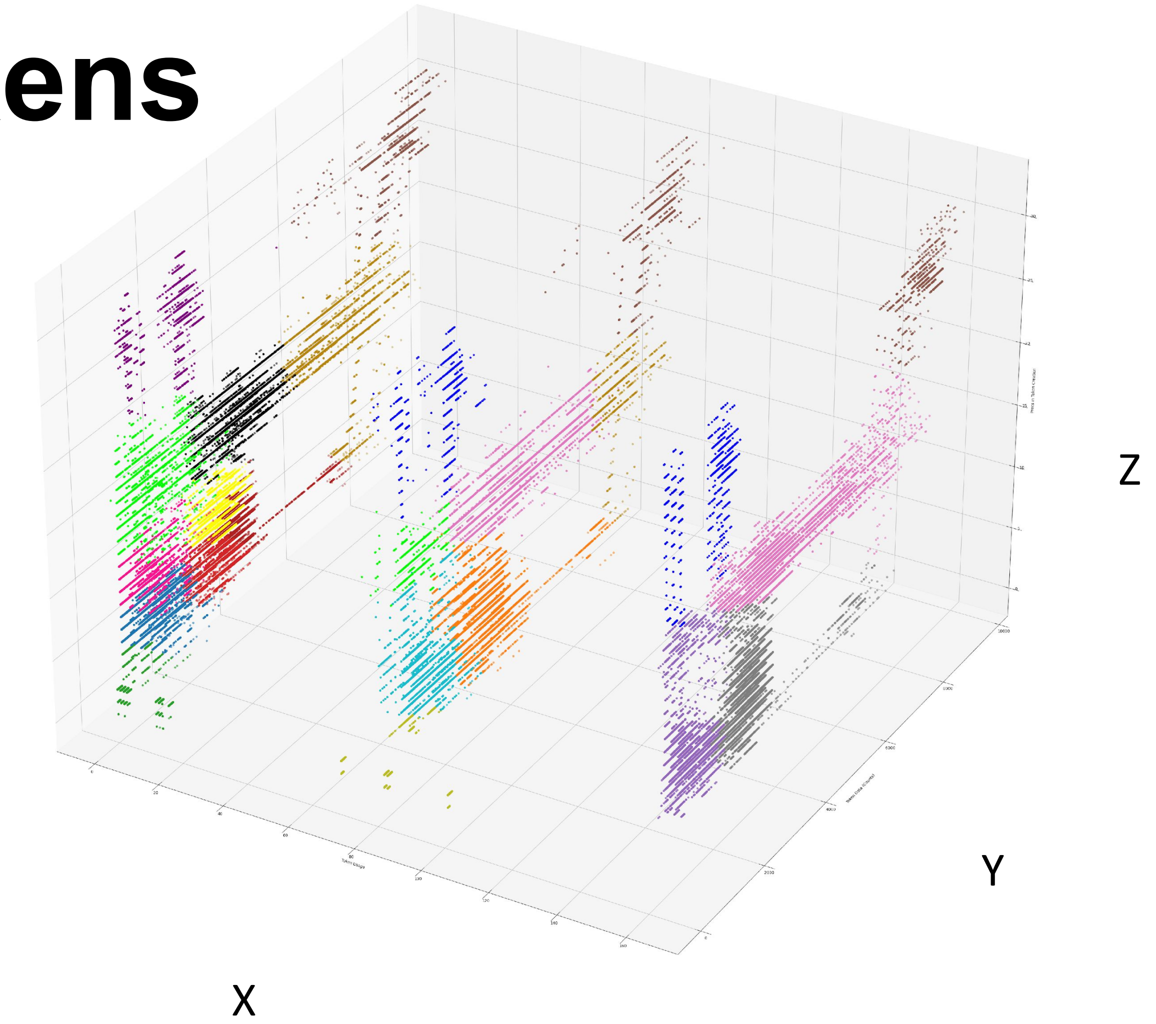
- xRAGE (LANL shock hydrodynamics code) uses **Tokens** to represent the communication pattern
  - Creation exchanges counts of data (no datatype info)
  - Usage of Token is the actual exchange (can flip direction)
- Multiple protocols for creating and using tokens with different performance
  - Creation: RMA-based sparse AllToAll, SomeToSome, MPI AllToAll; Amanda examining tradeoff
  - Communication:
    - Currently uses simple point-to-points to each process that is involved in the final pattern
    - MPI Neighbor collectives could optimize but MPI topology creation too expensive
    - Working on new abstractions to optimize this problem (see Amanda and Tony's talks)
- Research funded by both PSAAP III and LANL/UNM contract

Irregular Communication Profiling

- Maintain global string buffer that writes to file when the program finishes
  - Token Creation
    - Token ID (same across all ranks), Ranks involved
    - The count(s) and base direction(s) of data to be exchanged
    - Call site, Creation Time
    - Example:
      - Rank 0: 0|B1:1-200,T2:1100,F4:900,|1.23|T0
      - Rank 1: 0|B0:200-1,B2:300-400,B3:1500-1600,|0.97|T0
  - Token Usage:
    - Token ID, Direction
    - Size of Datatype involved (char, int, double, etc), Time
    - Example of three uses of tokens: 0:0:8|1.01, 1:0:8|2.45, 0:0:4|0.78,
- Minimal impact to problem completion time; total runtime slower (from I/O costs)
- Also examining HYPRE + AMG 2023, debugging collection/analysis

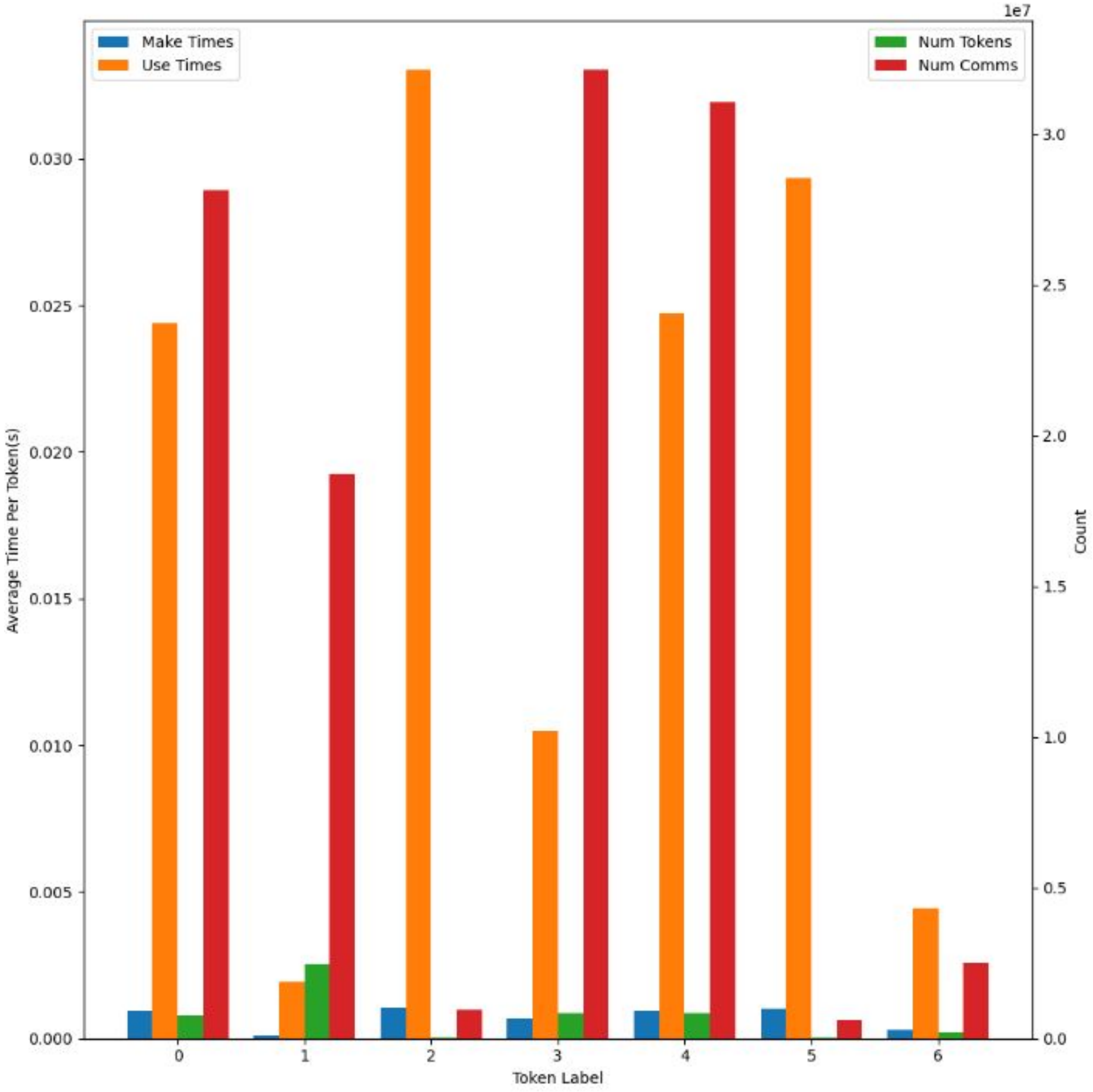
Finding Groups of Tokens

- Cluster tokens by usage characteristics:
  - X-axis: Exchanges performed
  - Y-axis: Items exchanged
  - Z-axis: Ranks exchanged with
  - Color = k-means group (18 groups)
- Pro: Clear sets of groups, unclear "right" number of clusters to make
- Con: Can't yet individual track how token characteristics change as scale changes
- Next steps:
  - Adding in call site to better track changes in token usage as scale changes
  - Examine optimization opportunities of difference token groups



How long do token operations take?

- Blue = average make time of token with label
- Orange = average use time of token with label
- Green = total number of tokens in this label
- Red = total number of scatter/gather calls made using tokens in that label
- Label 6 = the first 400 skipped tokens

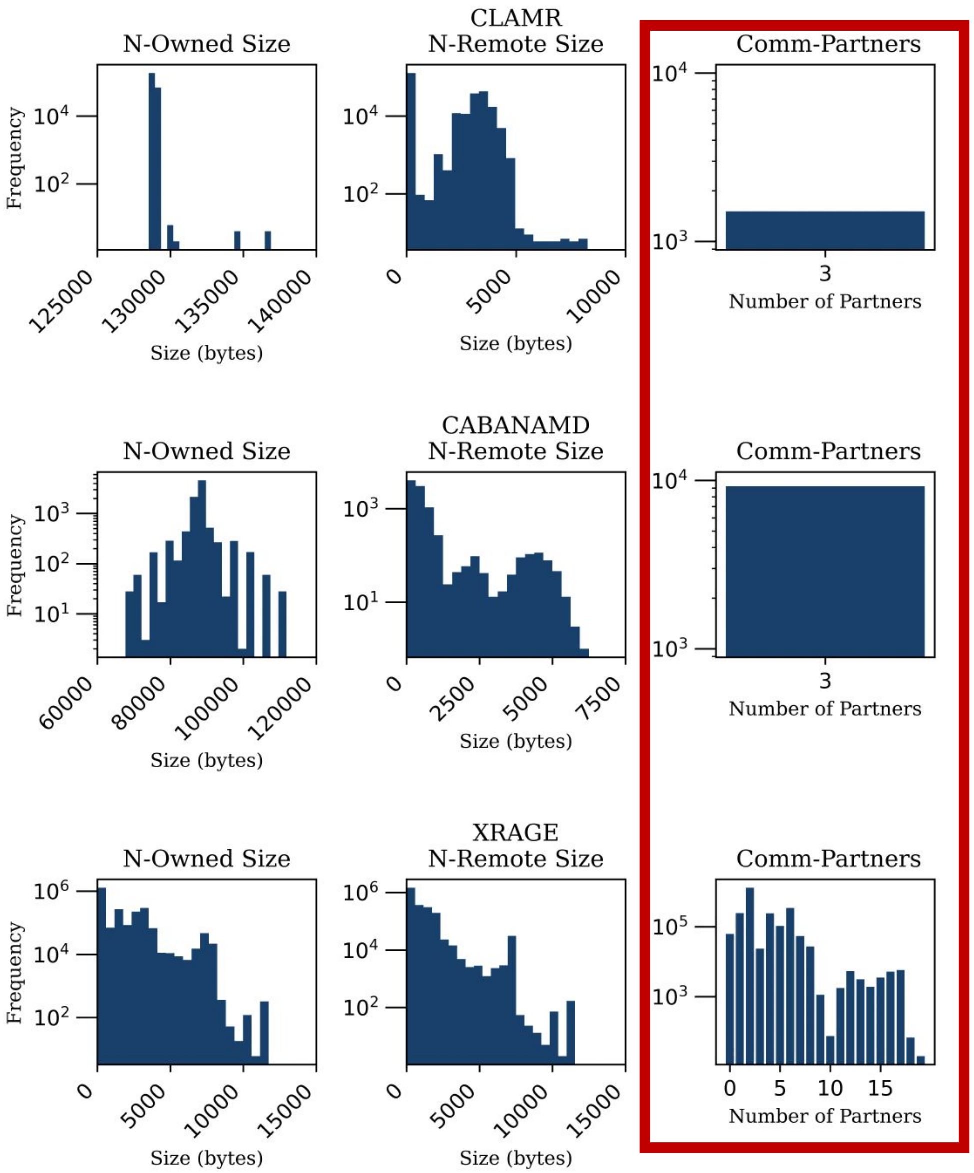


Can we compactly capture and replay these patterns?

- Goal: provide benchmarks that capture relevant characteristics of irregular application communication
  - Develop and test performance optimizations without prohibitively large communication traces
- Approach:
  - Extract distributions of communication partners, sizes, and data strides from application runs using collected data (empirical or parametric distributions)
  - Create benchmark that generates irregular communication based on these distribution parameters
- Also supports system acquisition efforts – can provide benchmark and data from key applications to vendors

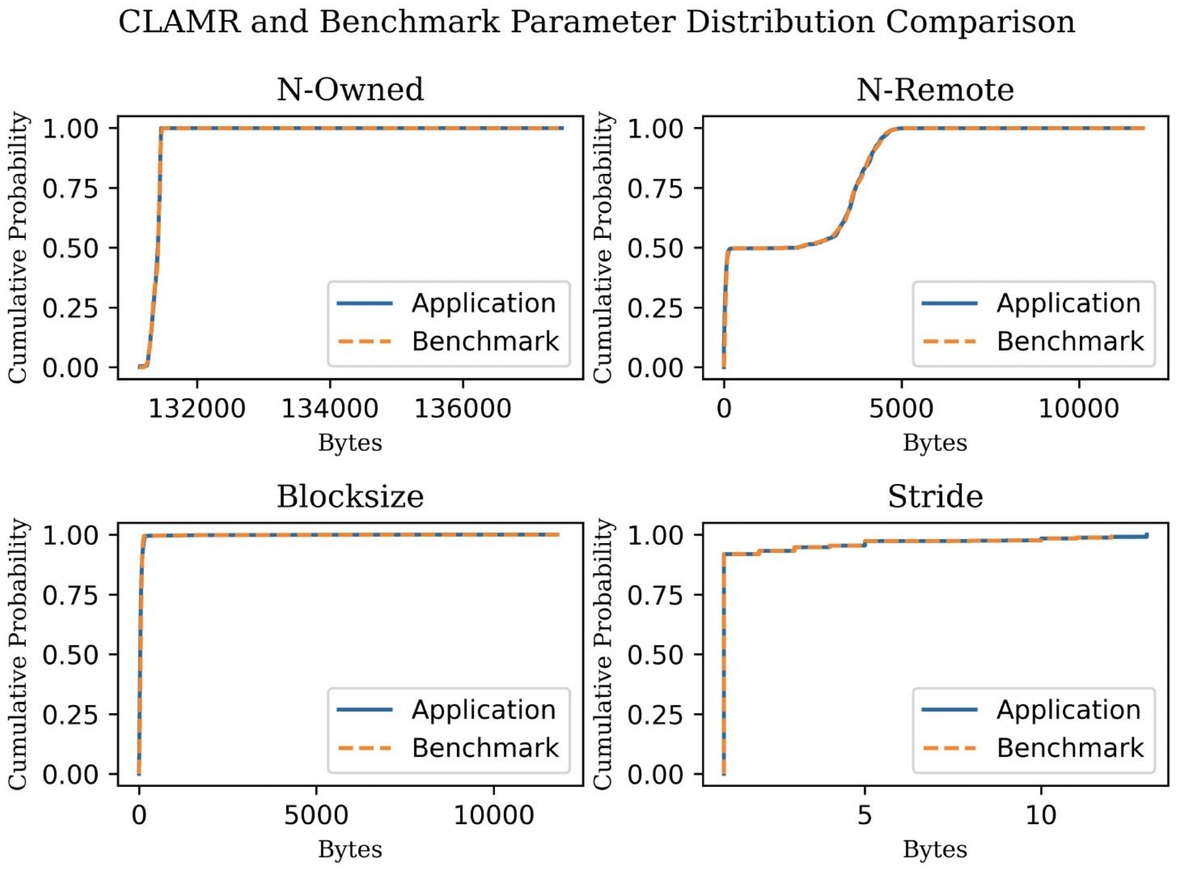
Distributions from CLAMR, CabanaMD, and xRage

- 256 process runs of CabanaMD, CLAMR (LLNL Quartz), and xRage (LANL Darwin)
- Collected information on owned and remote size, communication partners
- Significant difference between benchmark number of peers and production code number of peers!



Benchmark recreates these distributions

- Compared CDFs between benchmark and application runs
- Graph shows of owned and remote sizes, block sizes, and block strides for 512 process CLAMR run
- Examining correct statistical equivalence test to use - outliers in real data make simple statistical tools inaccurate
- Want to extend these results to additional applications, benchmarks, and input decks

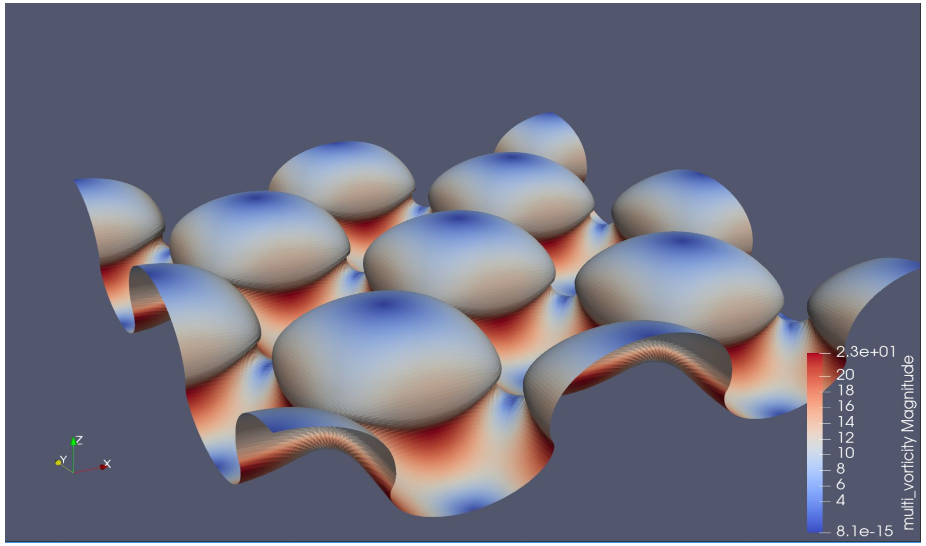


Goal: Create suite of similar tools and inputs for production codes

- Finding: Significant gap between production communication patterns and proxy/benchmark communication patterns
  - Most Benchmarks use communication patterns that are highly unrealistic
  - Some counterexamples exist but mostly do not use GPUs (MiniAMR, SNAP)
- Need suite of proxies, benchmarks, and input data that mimic production runs to drive research, development, and acquisition
  - Provide better input decks for existing proxies, macro- and micro- benchmarks when possible
  - Will require continued work on new benchmarks and production application data
- Goal for remaining two years: develop and curate this suite in collaboration with national lab partners
  - Partners at labs already identified, SIAM PP mini-symposium with personnel already planned
  - Already working toward data on xRAGE, HOSS, SPARC
  - Discussions needed on EMPIRE/MiniEM, other applications to target

Beatnik: A High-Performance Parallel Fluid Interface Benchmark

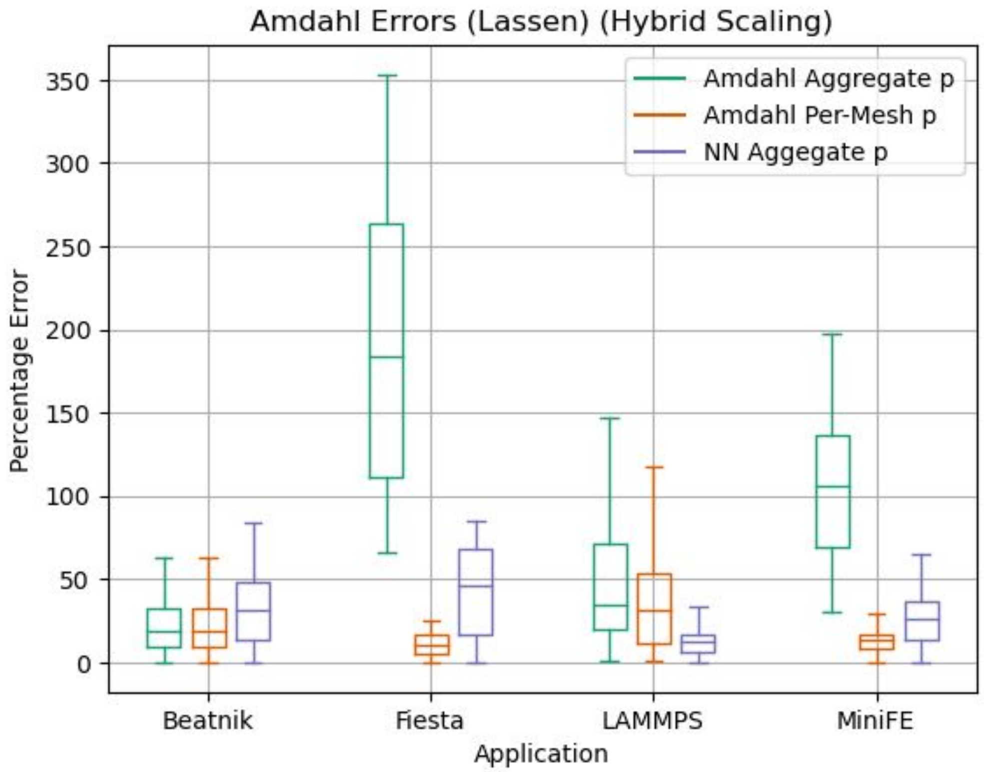
- Benchmark for methods requiring global communication
  - Uses Raag and Shkoller's fluid interface motion formulation
  - Implementation in ECP-Copa's Cabana/Kokkos framework
  - Low-order model is FFT-intensive
  - High-order model requires far-field force solver
- Scalable low-order implementation and brute force high order parallel solvers done; cutoff-based high-order solver in progress
- Next steps:
  - Benchmark and optimizing global communication algorithms
  - Examining octree for FMM using Cabana's sparse grid abstractions
  - Waiting to integrate with a fluid solver (HIGRAD or Fiesta/Fury) until Spring 2024 after discussions with Jon Reisner
- Source available: https://github.com/CUP-ECS/beatnik
- For more information, see Jason Stewart and his poster



Beatnik Low-order Multi-mode Rocket Rig Simulation

Modeling Performance Scaling and Communication Impacts

- Initial approach: train machine learning model to predict application runtime based on communication rooflines.
  - Predictions were *very* poor with limited data
  - Training data needed for black box modeling was prohibitive
- New approach: Adopt a gray box approach based on performance laws to reduce training data needed
  - Generalized Amdahl's law as baseline prediction
  - Learn *overhead* function to correct Amdahl for each application
- Step one: directly train overhead from application runtime
  - Compare neural network trained *overhead* with Amdahl percent parallel estimated from exact problem or aggregate of problems
  - Result: Neural network competitive with Amdahl's law with high-quality information on amount of parallelism in the problem
- Next: learn how changes in communication rooflines impact the parallelism and overhead in this model
- Submission to IPDPS or ICS in preparation



Goal: Model communication tradeoffs for coupled codes on DPU and APU systems

- Driving question: How to partition, communicate, and program coupled codes in current and upcoming systems?
  - When should we program and locate bandwidth/latency sensitive algorithms on DPUs?
  - How to partition communication primitives and algorithms between GPU and CPU cores in APU systems?
- Application: Assume a coupled code comprised of codes similar to Beatnik and HIGRAD/Fiesta (e.g. the Fury code underway at LANL)
- Step two: Predict how bandwidth changes impact scaling overhead in Beatnik low order and Fiesta/HIGRAD performance (see approach on previous slide)
- Step three: Predict how changes in latency impact scaling overhead performance of sorting/tree methods
- In parallel: Understand performance characteristics of DPU and APU systems.
  - Have started (non-NDA) discussions with NVIDIA.
  - Will be working with AMD under NDA as well.

Summary

- Wide range of findings, opportunities and some limitations and areas on modeling and benchmarking across a range of communications
- Specific modeling and benchmarking goals for final two years of project to inform abstraction development
- Benchmarking and modeling results driving and closely integrated with abstraction development and optimization