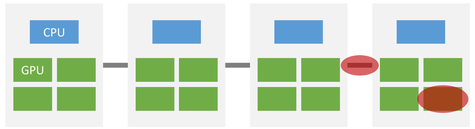
Fast Profiling-based Performance Modeling of Distributed GPU Applications

Jaemin Choi[1,2], **Abhinav Bhatele**[2] (advisor), **David Richards**[2] (advisor)

[1]University of Illinois Urbana-Champaign, [2]Lawrence Livermore National Laboratory

Overview



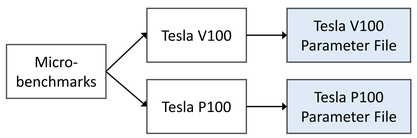
- Increasing number of GPU-accelerated applications targeting execution at scale
- Existing performance models
  - Only focus on either GPU kernels or MPI communication
  - Or are too specialized for one application
- **Goal:** "Develop a **general**, **accurate**, **fast end-to-end performance model** for distributed GPU applications"

Methodology

- Profiling-based, black-box approach

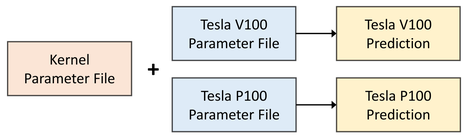
- **Part 1: Modeling GPU kernels**[1]

1. Obtain GPU parameters with microbenchmarks



2. Obtain kernel parameters with a profiling run

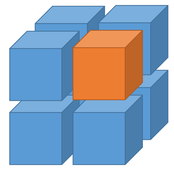
3. Generate performance predictions



- **Reduction in profiling time** (compared to [1])
  - Aggregated metric profiling runs
  - Limited profiling scope to time stepping loop
  - Selective profiling of kernels that constitute 99% (or some chosen value) of total GPU time
  - **4+ hours** → **30 minutes** for MiniFE

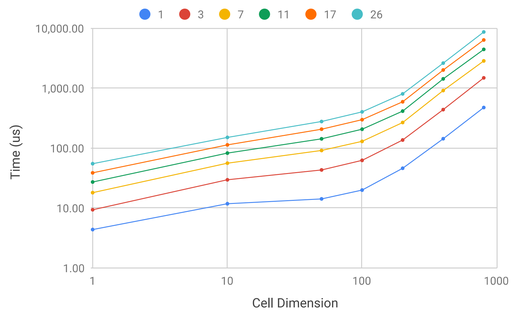
Methodology (cont.)

- **Part 2: Modeling MPI communication**
  - Inspect code & identify communication pattern
  - **Latency-bandwidth model**[2] for simple P2P
  - **Recursive doubling**[3] for collective communication



<Fig 1. 3D Halo Exchange>
(8 MPI ranks, 7 neighbors)

  - Modeling **halo exchange** communication
    - Occurs in many scientific applications
    - MPI_Irecvs & MPI_Isends (for all neighbors) -> **MPI_Waitall** (takes up most of the time)
    - Cannot be modeled with pingpong benchmark



<Fig 2. MPI_Waitall Time in 3D Halo Exchange Benchmark>
(LLNL Lassen, 2 ranks, 1 per node, lines: maximum number of neighbors)

  - Developed **3D halo exchange benchmark**
    - Uses only up to 2 nodes, combining both intra-node and inter-node results
    - Generalized equation to be derived
    - # of neighbors: 1 (2 ranks) → 26 (64+ ranks)

- **Part 3: Build end-to-end performance model**
  - **Overhead**
    - Time spent on operations other than profiled GPU kernels and MPI communication
    - Assume it stays constant as we scale
  - **Application runtime = Overhead + GPU kernel time + MPI communication time**
  - Model can be applied to larger scale to predict scalability
  - Currently limited to weak scaling
    - GPU kernel times do not change as we scale
    - Only MPI communication affects scalability

Target Application

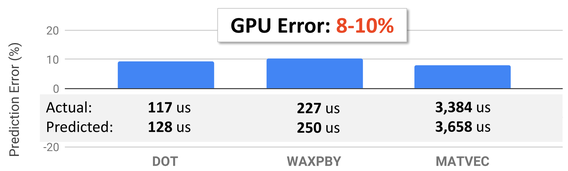
- **MiniFE**: unstructured implicit finite element code
- Main iteration loop

- dot(): vector dot product
  - **GPU kernel (DOT)**
  - Reduction kernel
  - Device-to-host memcpy
  - **MPI_Allreduce**
- waxpby(): $w = ax + by$
  - **GPU kernel (WAXPBY)**
- matvec(): matrix-vector product
  - Packing kernel
  - Device-to-host memcpy
  - **Halo communication**
  - **GPU kernel (MATVEC)**

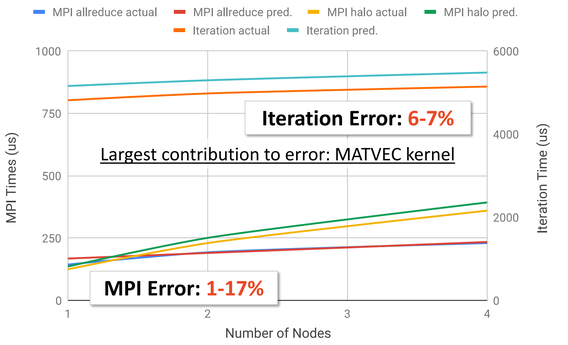
⚑: overhead

Results

- **Platform 1: LLNL Lassen**
  - 4 NVIDIA Tesla V100 GPUs/node, up to 16 GPUs
  - Weak scaling MiniFE: **measured times**
    - Constant overhead and GPU kernel times
    - Increase in allreduce and halo communication times



<Fig 3. GPU Kernel Prediction Error on Tesla V100>

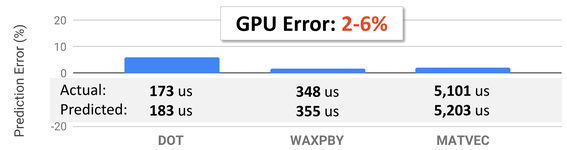


<Fig 4. Actual & Predicted Times for MPI Communication and Iteration>

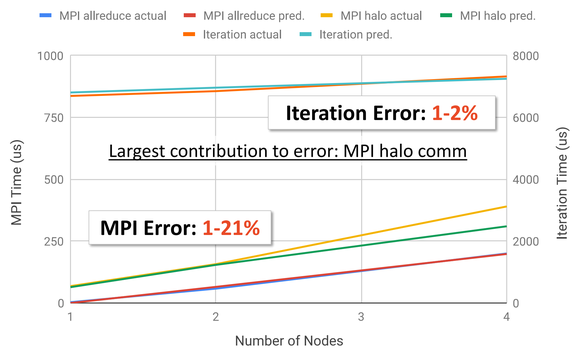
Results (cont.)

- **Performance predictions**
  - Overhead assumed constant: 820 us
  - GPU kernel times using roofline model
  - Allreduce time = 22.1 * N_{nodes} + 146
  - Halo exchange time computed from benchmark
  - $T_{Iteration} = T_{overhead} + T_{kernels} + T_{allreduce} + T_{halo}$

- **Platform 2: PSC Bridges**
  - 2 NVIDIA Tesla P100 GPUs/node, up to 8 GPUs
  - Similar weak scaling behavior & good prediction accuracy



<Fig 5. GPU Kernel Prediction Error on Tesla P100>



<Fig 6. Actual & Predicted Times for MPI Communication & Iteration>

  - **Performance predictions**
    - Overhead: 1001 us
    - Allreduce time = 66.4 * N_{nodes} – 68 (0 at 1 node)

Future Work

- Improve GPU modeling
  - Integrate data transfer model
  - Support use of CUDA-aware MPI and GPUDirect
  - Incorporate L1 cache effects
- Improve modeling for MPI collectives and halo communication
  - Generalize halo exchange benchmark and improve accuracy
- Model strong scaling behavior
- Evaluate more applications on more platforms

[1] Konstantinidis, E. and Cotronis, Y. (2017). A quantitative roofline model for GPU kernel performance estimation using micro-benchmarks and hardware metric profiling. Journal of Parallel and Distributed Computing, 107, pp.37-56.
[2] Hoefler, T., Gropp, W., Thakur, R., Träff, J. L. (2010). Toward Performance Models of MPI Implementations for Understanding Application Scaling Issues. Recent Advances in the Message Passing Interface. EuroMPI 2010. Lecture Notes in Computer Science, pp.21-30.
[3] Thakur R., Gropp W. (2003). Improving the Performance of Collective Operations in MPICH. Recent Advances in Parallel Virtual Machine and Message Passing Interface. EuroPVM/MPI 2003. Lecture Notes in Computer Science, vol 2840.

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. This work also used the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation grant number ACI-1548562. Specifically, it used the Bridges system, which is supported by NSF award number ACI-1445606, at the Pittsburgh Supercomputing Center (PSC).