# Fast Profiling-based Performance Modeling of Distributed GPU Applications

Jaemin Choi
University of Illinois Urbana-Champaign
Lawrence Livermore National Laboratory
jaemin@acm.org

Abhinav Bhatele (advisor)
Lawrence Livermore National Laboratory
bhatele1@llnl.gov

David Richards (advisor)
Lawrence Livermore National Laboratory
richards12@llnl.gov

## ABSTRACT

An increasing number of applications utilize GPUs to accelerate computation, with MPI responsible for communication in distributed environments. Existing performance models only focus on either modeling GPU kernels or MPI communication; few that do model the entire application are often too specialized for a single application and require extensive input from the programmer.

To be able to quickly model different types of distributed GPU applications, we propose a profiling-based methodology for creating performance models. We build upon the roofline performance model for GPU kernels and analytical models for MPI communication, with a significant reduction in profiling time. We also develop a benchmark to model 3D halo exchange that occurs in many scientific applications. Our proposed model for the main iteration loops of MiniFE achieves 6-7% prediction error on LLNL Lassen and 1-2% error on PSC Bridges, with minimal code inspection required to model MPI communication.

## KEYWORDS

performance modeling, GPU acceleration, MPI communication

## 1 INTRODUCTION

### 1.1 Roofline Modeling for GPU Kernels

The method we adopt to model performance of GPU kernels is the quantitative roofline model [2], where the performance is predicted from two parameter files; one for GPU parameters and the other for the application. We first obtain GPU parameters such as attainable maximum FLOPS and memory bandwidth by executing microbenchmarks on the target GPUs. Then we perform a profiling run to obtain parameters for the kernels within the application. Kernel performance predictions for a target GPU are then generated by combining the two parameter files.

### 1.2 MPI Communication Models

An analytical approach is widely employed to model MPI communication; latency-bandwidth model [1] for point-to-point messages and recursive doubling [3] for collective communication such as broadcast, reduction and allreduce. However, the communication pattern needs to be determined either by inspecting the application code or from knowledge of the developer.

*SC '19, November 17–22, 2019, Denver, CO*
2019. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

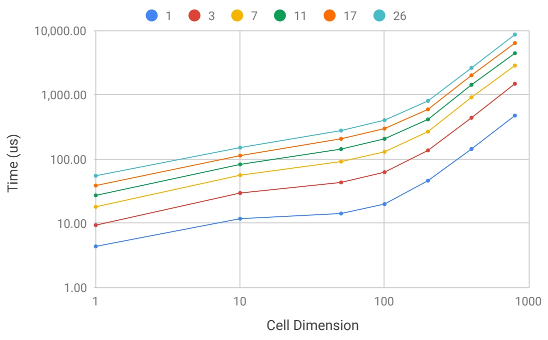

**Figure 1: MPI_Waitall Time in Halo Exchange Benchmark**

## 2 METHODOLOGY

### 2.1 Building End-to-end Performance Model

To build a performance model for the entirety of the application, we combine the GPU roofline model and MPI communication models and integrate the concept of *overhead*. Overhead is the time spent on operations other than kernels and communication, and it is assumed constant as we model weak scaling. Weak scaling also allows the model to reuse the estimated kernel times at larger scale.

Several improvements have been made to improve usability and accuracy; profiling time for kernels has been significantly reduced, and a benchmark was developed to better model the non-blocking halo exchange communication patterns found in many scientific applications. We reduced profiling time by aggregating multiple profiling runs, limiting the profiling scope to only the time stepping loop, and selectively profiling kernels that constitute most of the execution time. For MiniFE, discussed in the next section, profiling time went down from more than 4 hours to just 30 minutes.

The halo exchange benchmark was developed to model the `MPI_Waitall` time in the non-blocking 3D halo exchange; only up to 2 nodes are used and both intra-node and inter-node results are combined for more accuracy. Figure 1 shows the computed times from the benchmark used in the performance model.

### 2.2 Application Case Study: MiniFE

MiniFE[1] is a proxy application for unstructured finite element codes, part of the Mantevo project. We focus on the conjugate gradient

[1]https://github.com/Mantevo/miniFE

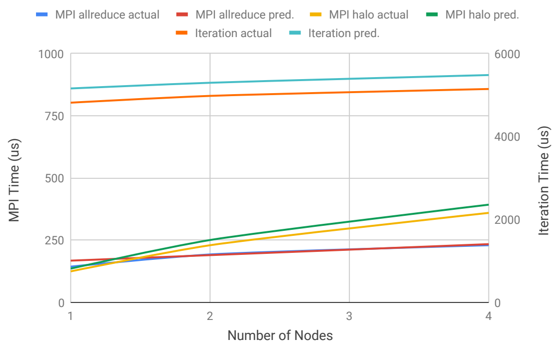

**Figure 2: Actual and Predicted Times for MPI Communication and Iteration on LLNL Lassen**

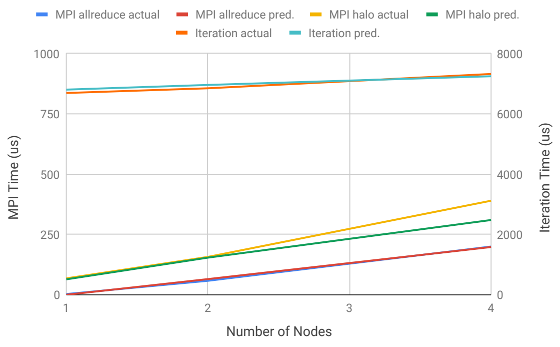

**Figure 3: Actual and Predicted Times for MPI Communication and Iteration on PSC Bridges**

(CG) solve portion of MiniFE, where most of the computation and communication occur.

It should be noted that the author had little experience or knowledge of MiniFE; its code was minimally inspected to determine the MPI communication pattern.

## 3 RESULTS

We evaluate the performance model for MiniFE created with our methodology on 2 HPC machines: (1) Lassen, a CORAL-class supercomputer deployed at Lawrence Livermore National Laboratory, and (2) Bridges, a XSEDE supercomputer deployed at Pittsburgh Supercomputing Center. Each compute node of Lassen has 4 NVIDIA Tesla V100 GPUs, whereas Bridges has nodes each consisting of 2 NVIDIA Tesla P100 GPUs. Prediction error plots for GPU kernels are omitted from the summary in favor of space; please refer to the poster.

### 3.1 LLNL Lassen

From the weak scaling runtimes, we see that the overhead and GPU kernel times stay mostly constant as expected. On the other hand, time spent in communication increase with more nodes. To generate the performance predictions, we use the measured overhead (820 $\mu$s), kernel times from the GPU roofline model, and communication times from the MPI models.

Prediction errors for kernels vary between 8% and 10%. The actual and predicted times for MPI communication (both allreduce and halo exchange) and iteration are shown in Figure 2. The prediction error for MPI allreduce is between 1% and 17%, with the largest error at 1 node (4 ranks); for the halo exchange the error is consistently 9%. With these combined, the predicted iteration time has a prediction error between 6% and 7%, with the MATVEC kernel contributing the largest error due to the relative scale of its execution time.

### 3.2 PSC Bridges

On Bridges, no profiling run aside from the GPU microbenchmarks was necessary; the application parameters obtained from Lassen were used for GPU performance predictions. The measured overhead was 1001 $\mu$s.

Prediction errors for kernels vary between 2-6%, and for MPI communication they are between 1-21% with the largest error (21%) from halo exchange at 4 nodes, as shown in Figure 3. The error for iteration time is only 1-2%, however, due to the fact that predictions for GPU kernels are very accurate, especially for `MATVEC` which constitutes the majority of the time.

## 4 CONCLUSION

We have seen that our performance modeling methodology requires user input only for the communication models, as the rest of the program are modeled via profiling. This allows our approach to be easily administered to various applications and avoid extensive code inspection. We have also improved usability by significantly reducing profiling time, and developed a benchmark that models halo exchange with minimal resources. The accuracy of the performance model has been evaluated with MiniFE on two separate platforms, LLNL Lassen and PSC Bridges, which resulted in relatively small prediction errors.

## ACKNOWLEDGMENTS

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).

## REFERENCES

[1] Torsten Hoefler, William Gropp, Rajeev Thakur, and Jesper Larsson Träff. 2010. Toward Performance Models of MPI Implementations for Understanding Application Scaling Issues. In *Proceedings of the 17th European MPI Users' Group Meeting Conference on Recent Advances in the Message Passing Interface (EuroMPI'10)*. Springer-Verlag, Berlin, Heidelberg, 21–30. http://dl.acm.org/citation.cfm?id=1894122.1894126

[2] Elias Konstantinidis and Yannis Cotronis. 2017. A quantitative roofline model for GPU kernel performance estimation using micro-benchmarks and hardware metric profiling. *J. Parallel Distrib. Comput.* 107 (2017), 37–56.

[3] Rajeev Thakur and William D. Gropp. 2003. Improving the Performance of Collective Operations in MPICH. In *Recent Advances in Parallel Virtual Machine and Message Passing Interface*, Jack Dongarra, Domenico Laforenza, and Salvatore Orlando (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 257–267.