Fast Classification of MPI Applications Using Lamport's Logical Clocks

Zhou Tong
Florida State University
tong@cs.fsu.edu.edu

Scott Pakin
Los Alamos National Lab
pakin@lanl.org

Mike Lang
Los Alamos National Lab
mlang@lanl.org

Xin Yuan
Florida State University
xyuan@cs.fsu.edu

1. INTRODUCTION

Classification and identification of performance limiting factors in MPI applications is desired for code optimization, studies of network interconnect design and procurement of future systems. Various tracing tools such as the DUMPI library [4] have been developed for trace collection; many trace replay and/or analysis tools such as traceR[1] and Structured Simulation Toolkit (SST) [2] have been implemented, but most require high simulation overhead. Moreover, in one run, most simulators can only produce performance prediction under one given system configuration.

In this work, we propose a trace-driven, MPI-based classification tool that can identify the performance limiting factors of a parallel application targeted at production interconnect technologies. By maintaining multiple logical clocks, the tool can produce performance estimations for many different network configurations in one pass. By observing the application performance for a range of network configurations, we can determine whether the application is computation-bound, communication-bound, or load-imbalance-bound. Classification results on DOE's 9 applications show the performance characteristic of each application as well as its sensitivity to the speedup and slowdown of bandwidth speed and latency for the given interconnect technologies.

2. TECHNICAL DETAILS

The tool uses an extension of Lamport's logical clock [6] to keep track of time progress in the trace replay. Such a logical clock not only maintains the happen-before relationship, it also tracks the predicted application execution time with regards to the physical computation time observed in the trace and non-unit latency. The tool maintains multiple sets of logical clock counters, one set for each network configuration. Each set of counters record different types of predicted times including computation time, wait time, latency time, bandwidth time of the application for the particular interconnect configuration.

Our tool supports two models for point-to-point communication: a table look-up based model that provides more accurate estimation about the communication time on a system, and a simple Hockney's model [5] where an n bytes message takes $\alpha + n \times \beta$ time to complete, where α is the communication latency (the time to communicate a zero byte message) and β is the bandwidth term that is the reciprocal of the bandwidth. A collective communication is modeled as a global synchronization for all processes to be ready for the operation, and then the models in [7] are used to estimate the time for different collective operations.

During one pass of the trace replay, logical clocks for all network configurations are maintained simultaneously and all of the counters are computed. With the predicted execution times for carefully selected network configurations, application classification is performed based on the performance trend across a range of interconnect configurations instead of the predicted performance for a particular interconnect configuration. For example, to determine if MiniFE is computation-bound on 10G Ethernet which has an bandwidth speed (BW) of 10Gbps and latency (L) of 5 microseconds [3], we test the application's sensitivity to both bandwidth and latency by scaling 10G Ethernet's bandwidth and latency with the selected configurations, $(BW/8, 8L)$, $(BW/4, 4L)$, $(BW/2, 2L)$, (BW, L), $(2BW, L/2)$, $(4BW, L/4)$ and $(8BW, L/8)$. Figure 1 shows that the computation time accounts for more than 90% of total execution and its total execution is insensitive to the speedup and slowdown of bandwidth and latencies. Therefore, MiniFE on 1152 ranks is computation-bound on 10G Ethernet.

This approach can tolerate the inaccuracy in the prediction for individual network configuration and make the classification robust. In order to model multiple network configurations in one application run, we assume that all P2P and collective communication are performed under Eagle protocol with Hockney's model. By analyzing the performance trend, we can determine whether the application is computation-bound (comp.), latency-bound (Latency), bandwidth-bound (BW), or load-imbalance-bound (Imb.). Furthermore, if the application is not bounded, we further classify the applications as load-imbalance sensitive (Imb./s), bandwidth sensitive (BW/s), latency sensitive (latency/s), and communication sensitive (Comm./s).

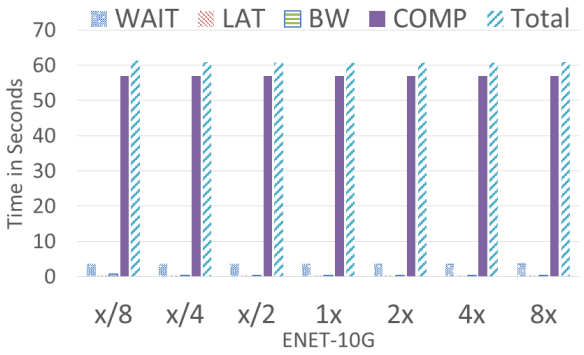

Figure 1: Sensitivity of MiniFE on 1152 ranks to communication (bandwidth and latency) on 10G Ethernet

Table 1: Classification results (number of MPI ranks shown in parenthesis)

	ENET-1G	ENET-10G	QDR
AMG(8)	Comp.	Comp.	Comp.
AMG(27)	Imb.-s	Imb.-s	Imb.-s
AMG(216)	Imb.-s	Imb.-s	Imb.-s
AMG(1728)	Imb.	Imb.	Imb.
AMR(64)	Latency-s	Comp.	Comp.
BigFFT(100)	Comm.	Comm.	Imb.
CLAMR(64)†	Latency	Latency	Imb.
CR(64)†	Comm.	Comm.	Comp.
FB(64)†	BW-s	Comp.	Comp.
FB(125)	BW-s	Imb.-s	Imb.-s
MG(1000)	Imb.-s	Comp.	Comp.
MiniFE(1152)	Comp.	Comp.	Comp.
PARTISN(168)	Imb.	Imb.	Imb.
IS(64)†	Comm.	Imb.-s	Imb.-s

3. PRELIMINARY RESULTS

First, we compare the predicted overall execution time of CLAMR, CR and FB with the observed performance. The results in the poster show that the prediction error of our proposed model with simple Hockney's model does is not as accurate as the look-up table approach. However, it is more than sufficient to classify applications since our classification is based on the performance trend over a range of network configurations, which alleviates the problems caused by the inaccurate modeling.

Table 1 summaries classification results of our selected DOE[1] full application and miniapps targeted at 1G Ethernet, 10G Ethernet and InfiniBand QDR. As can be seen from the experiments, by examining the performance trend for an application on a range of network configurations, many important performance characteristics of the application is revealed. By this approach, our tool is very effective in classifying applications even with the inaccuracy in the performance prediction for individual networks.

[1]Department of Energy, Design Forward Project, http://portal.nersc.gov/project/CAL/designforward.htm

Furthermore, We study the performance of our classification tool using NAS's benchmarks with 64 and 4096 ranks. Form the figures in the poster, we show the simulation time speed up ranges from 3 to 45 in comparison to the application time for 64-rank runs and 2 to 14 time speedup for 4096 ranks runs. In addition, we show that the execution time increases only slightly when the number of network configurations increases from 16 to 256 as a result of the extra communication cost of synchronizing timestamps based on our MPI-based implementation. This indicates that the number of network configurations (up to 256) is not a major performance limiting factor in the simulation; and the tool can effectively predict the performance for many network configurations in one simulation.

4. CONCLUSIONS

We have presented a trace-based and communication-centric fast classification tool for MPI programs. By simultaneously model performance over multiple network configurations, the tool can perform many network simulations in comparable time to a single simulation by existing simulators. The tool is able to effectively uncover performance characteristics of the applications and identify the individual contributors which include computation, latency, bandwidth speed and load-imbalance. Based on the application's sensitivity to each contributor, it can classify application into either bounded or sensitive. The tool can be used to aid parallel application programmers to study the performance of their experiments. It will also help the design and procurement of future HPC systems for DOE laboratories by narrowing down the number of key applications that requires more detailed performance investigation.

5. REFERENCES

[1] B. Acun, N. Jain, A. Bhatele, M. Mubarak, C. D. Carothers, and L. V. Kale. Preliminary evaluation of a parallel trace replay tool for hpc network simulations. In *Workshop on Parallel and Distributed Agent-Based Simulations*, Aug. 2015.

[2] H. Adalsteinsson, S. Cranford, D. A. Evensky, J. P. Kenny, J. Mayo, A. Pinar, and C. L. Janssen. A simulator for large-scale parallel computer architectures. *Int. J. Distrib. Syst. Technol.*, 1(2):57–73, Apr. 2010.

[3] Q. Corp. Introduction to ethernet latency. an explanation of latency and latency measurement. 2014. [Online; accessed 6-Oct-2015].

[4] S. Corporation. Sst: The structural simulation toolkit. 2014. [Online; accessed 14-DEC-2012].

[5] R. W. Hockney. The communication challenge for mpp: Intel paragon and meiko cs-2. *Parallel Comput.*, 20(3):389–398, Mar. 1994.

[6] L. Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7):558–565, 1978.

[7] R. Thakur and W. D. Gropp. Improving the performance of mpi collective communication on switched networks. 11/2002 2002.