Evaluating the Performance of MPI Java in FutureGrid

Nigel Pugh[2], Tori Wilbon[2], Saliya Ekanayake[1]
[1] Indiana University
[2] Elizabeth City State University

Abstract

Message Passage Interface (MPI) has been popular in developing tightly coupled parallel applications in the high performance computing (HPC) domain. The majority of such applications are based on either C, C++ or Fortran. The recent advancement in big data, however, has brought attention towards Java. Effort has also been put on Java's support for HPC with flavors of MPI such as OpenMPI Java and FastMPJ. We evaluate these against native C based MPI on a set of standard micro-benchmarks from Ohio State University. The results show a promising future with Java and MPI for HPC applications.

Introduction

This study serves as a proof of concept that Java based MPI communications can perform close to native implementations. We evaluate two MPI Java versions – OpenMPI [1] Java and FastMPJ [2] – against native OpenMPI. FastMPJ is a pure Java implementations whereas OpenMPI Java is a Java binding for the native OpenMPI. Our evaluations are based on benchmarking MPI kernel operations following the standard Ohio State University (OSU) Micro-Benchmark suite [3]. A detailed study extending this work is available at [4].

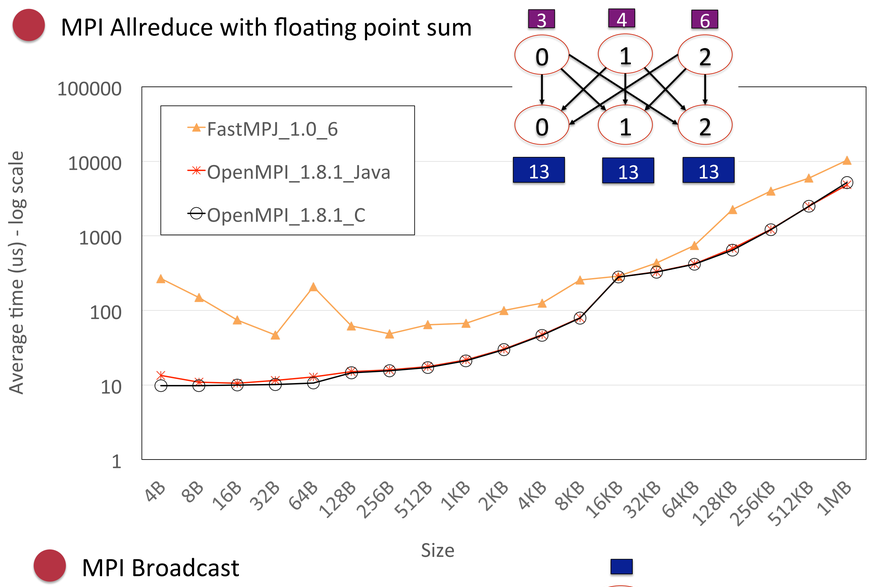
Evaluation

MPI collective primitives are evaluated for patterns, 1x1x8, 1x2x8, 1x4x8, and 1x8x8, where the digits represent threads per process, processes per node, and nodes respectively. Send and receive operations are timed for pattern 1x1x2 with different pairs of nodes. Averaged values of these tests are presented in graphs. Message sizes range from 0 bytes to 1 mega byte. We use FutureGrid as the HPC environment.

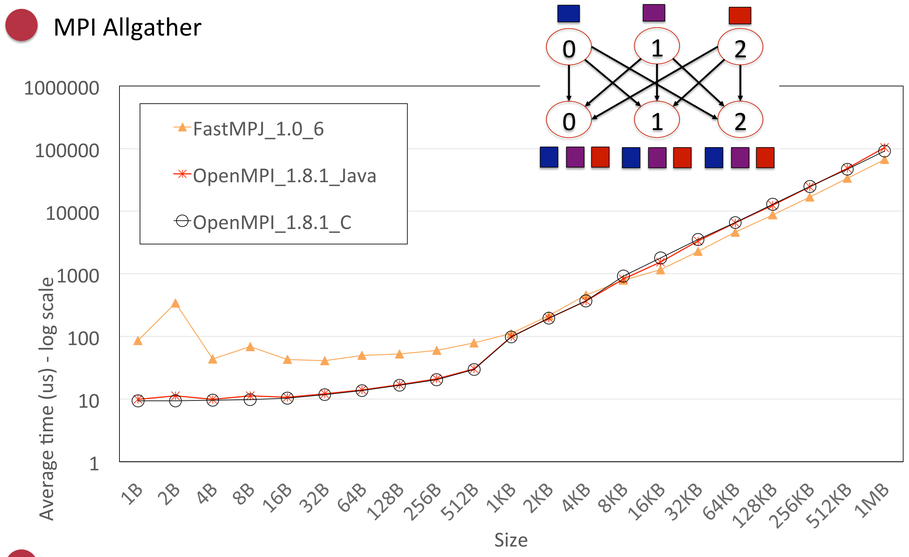
FutureGrid – 128 nodes, 2 Intel Xeon X5550 CPUs at 2.66 GHz with 4 cores, 8 cores per nodes 24 GB node memory and 20 Gbps IB.

Performance Results

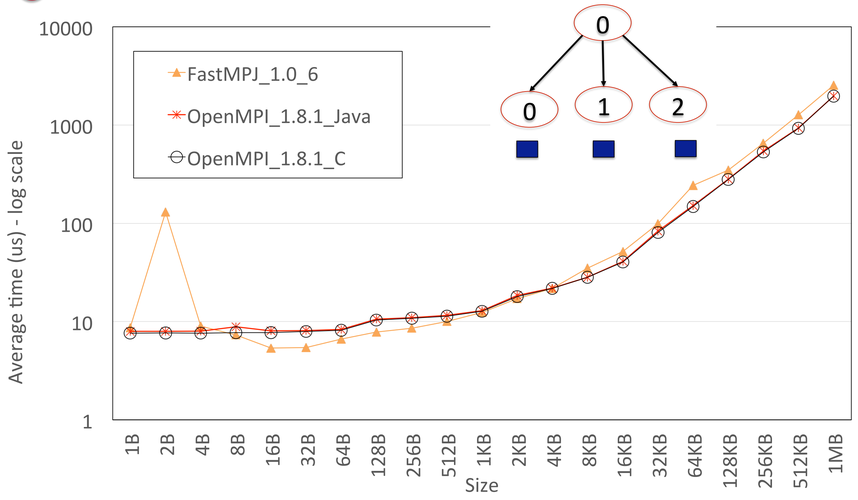
MPI Allreduce with floating point sum



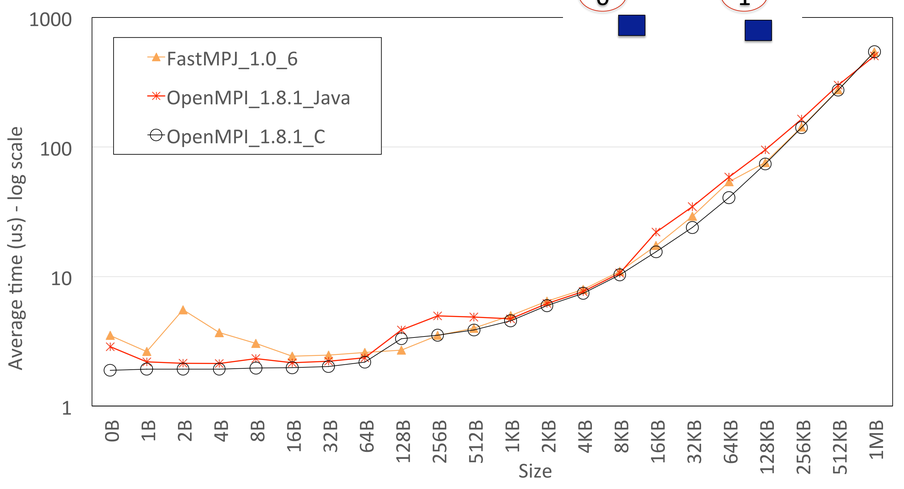
MPI Allgather



MPI Broadcast



MPI Send and Receive



Acknowledgements

We like to express our gratitude to Dr. Geoffrey Fox for giving us the opportunity to work in his lab this summer. We also would like to thank the School of Informatics at Indiana University Bloomington and the IU-SROC Director Dr. Lamara Warren. We are equally thankful Dr. Guillermo López Taboada for giving us commercially available FastMPJ. This material is based upon work supported in part by the National Science Foundation under Grant No. 0910812.

Primary Contact

Saliya Ekanayake, Indiana University, sekanaya@cs.indiana.edu
Dr. Geoffrey Fox, Indiana University, gcf@indiana.edu

References

[1] Gabriel, E., Fagg, G., Bosilca, G., Angskun, T., Dongarra, J., Squyres, J., Sahay, V., Kambadur, P., Barrett, B., Lumsdaine, A., Castain, R., Daniel, D., Graham, R. and Woodall, T. *Open MPI: Goals, Concept, and Design of a Next Generation MPI Implementation*. Springer Berlin Heidelberg, City, 2004.
[2] Expósito, R., Ramos, S., Taboada, G., Touriño, J. and Doallo, R. FastMPJ: a scalable and efficient Java message-passing library. *Cluster Computing*(2014/02/06 2014), 1-20.
[3] Laboratory, T. O. S. U. s. N.-B. C. and (NBCL) *OMB (OSU Micro-Benchmarks)*. City.
[4] Ekanayake, S. *Evaluation of Java Message Passing in High Performance Data Analytics*. City, 2014.
[5] von Laszewski, G.; Fox, G. C.; Wang, F.; Younge, A. J.; Kulshrestha; Pike, G. G.; Smith, W.; Voeckler, J.; Figueiredo, R. J.; Fortes, J.; Keahey, K. & Deelman, E. Design of the FutureGrid Experiment Management Framework, Proceedings of Gateway Computing Environments 2010 (GCE2010) at SC10, IEEE, 2010