Run time optimization of the C++ OpenMPI library

Martin Söderén

Final thesis

Master of Science in Computer Science and Engineering
at Linköping University

Supervisor: Prof. The Punisher

14th January, 2016

Abstract

Your abstract goes here... ...

Acknowledgements

Thanks Mum!

Contents

1 Introduction

This chapter is meant to give a brief introduction and summary of this thesis and hopefully make you compelled to continue reading.

1.1 Motivation

Today HPC(High-performance computing) is used in a lot of different areas to make large computations possible. They can be used to recreate the big bang, understanding earthquakes, folding proteins, testing nuclear weapons and predicting climate change to name a few. Some of these are in great interest for mankind. By calculating how proteins fold necessary knowledge can be acquired that can be used to prevent diseases such as Cystic Fibrosis, Mad cow disease and Alzheimer's. By predicting climate change knowledge can be gain on how to prevent the increasing temperature of our planet.

The MPI(Message Passing Interface) is a standardized message-passing system to pass messages between processes. The processes can run on the same or different machines. MPI is just a standard and a lot of different implementations exists on different languages such as C++, C and FORTRAN. The de-facto implementation of MPI is MPICH but a more general implementation is OpenMPI which is the one this thesis will use as it basis.

The C++ OpenMPI implementation have many variables that can be changed to increase the performance of the computation. The optimal configuration can differ depending on the platform running the computation and the computation itself. There are many variables and to find the perfect configuration through exhaustive search is not an alternative. There exists auto tuning software that by using benchmarks and algorithms can find a pretty good configuration. There is also software that can analyse the data from the OpenMPI profiling tool after a computation and give a report on how well the system is utilized. This is important when you run a computation several times but what if you are only going to run one large computation one time. In that case you would need something that dynamically tunes the system during run time. This problem is the basis for this thesis.

1.2 Why is this needed

There are no run-time tuning software for the C++ OpenMPI. The only alternative are post-mortem analysis of the computation and this is no good if you are only running one large computation once or a lot of smaller ones. One use case of this dynamically run-time optimization can be that you are running a

lot of small computations and you don't want to tune the system for each computation since this takes a lot of time and resources. Instead the system will tune itself when you start the computation.

1.3 Aim

The goal of this thesis is to evaluate which parameters of an calculation distributed on several nodes can be monitored and adjusted during run time. The result will be a tool that can suggest how the computation can be optimized during run time. This will of course require some calculation and bandwidth to analyse the status of the the computation. The goal is that the overall performance will increase so that the performance gained is higher than the performance lost from resources used by the optimization tool.

1.4 Research questions

- Of the parameters that can be adjusted during run time in the OpenMPI library, for which of these can the result of a adjustment during run time be calculated
- Is the OpenMPI profiling tool capable of the measuring the data required for the calculation in question two or are there are need for a new tool running on all nodes

1.5 Delimitations

This thesis will only use the C++ OpenMPI library as it basis. No other programming languages or libraries will be evaluated. However OpenMPI has it roots on MPICH so there might be a possibility that it could work with MPICH and its derivatives but that will not be considered in this thesis. If the result is that such a tool can be created then the work on a base framework will start, however due to time constraints just the bare bone structure will be implemented and documentation will be created on how to further develop it. In the end there will only be some base functionality, not a complete tool.

The performance testing can only be done one a regular Ethernet and will not use standard HPC equipment such as Infiniband.

2

Background

MPI was first proposed in 1991 by a group of researchers. The first MPI standard was presented in 1993. Since then different libraries have been developed for many different platforms and programming languages.

The reason for this thesis is the authors interest in HPC performance and since MPI is used in a lot of the top 500 supercomputers in the world the seemed like a interesting topic. A proposition was made to professor The Punisher and he approved of it. The goal if the thesis is to create a plug-in for the OpenMPI profiling tool that can dynamically tune the OpenMPI variables during run time.

3

Theory

1 HPC in general

An HPC is normally a massively parallel computer using thousands of nodes interconnected by high bandwidth, low latency networks such as Infiniband[1]. This is however when the cluster computing approach is used. A HPC can also use a GRID structure where computers does not necessary need to be in the same building or even in the same country. You can for example have two clusters in different countries and spread the work on both clusters and then use a GRID/cluster approach.

Today there are many algorithms that scale well horizontal, meaning that adding a extra node to the cluster will decrease computing time. Today hardware is also relatively cheap, the most expensive part of a computing is mostly the electricity required to run the nodes. Most benchmarks available focuses on performance but it could also be possible to tune for efficiency to reduce the energy required.

The optimization of a HPC can be done on multiple levels. For example in libraries using auto tuning library generators such as the one in ATLAS(Auto-tuning linear algebra library) which as the name suggests is a library for linear algebra. The optimization can be done in the compiler using iterative compiling. It can also be done using dynamic composition meaning that there is a exposed tuning interface for the programmer but it could also mean that the program is using learning-algorithms during run-time to tune the parameters exposed.

2 OpenMPI

OpenMPI is a standard message passing interface. It purpose is to abstract away all the underlying transport for the program that is needed to send data from one process to another. The other process can be on the same node or on another node in another country. OpenMPI can however do much more than just send messages between different processes, it supports sending complex data structures, starting, monitoring and killing processes and also debugging them.

2.1 MCA in OpenMPI

The design of the OpenMPI library is called MCA(Modular Component Architecture). This design pattern makes the library very modular and the exchange for other modules and addition of newer easier. The application is also split up into abstraction layers:

- OMPI

- ORTE
- OPAL

Where the OMPI(Open MPI) is the highest abstraction layer and the which communicates with the application. The ORTE(Open MPI Run-Time Environment) handles the execution and killing of processes on nodes.

The MCA in OpenMPI is a series of frameworks,components and modules that not only can be combined during compilation but can also be exchanged during runtime. This allows for runtime tuning by exchanging different modules. Also variables can be adjusted during runtime if it considered a MCA variable, a example is the boundary between long and short messages in the TCP protocol. If one node has two connections, one GigaBit and one 10 GigaBit, it might wants to change these values when switching between connections.[2]

3 OTPO

Mostly used for tuning OpenMPI as a whole cluster. Also uses a combination of brute-force and heuristics before runtime. This requires a lot of compilation or optimization time and are sensitive to input data [3].

4 Runtime optimization

There have been evaluation of dynamic control over MCA parameters and the end result where that these could be implemented without any considerabke overhead. [4].

5 Profiler organisation

The PMPI (MPI standard profiling interface) is used to gather information from the running code using several different modules. For example it can catch calls to the MPI_Send (actually PMPI_Send when setup to capture the event) and monitor the time spent in that function. According to the OpenMPI documentation [5] the OMPI source base which PMPI is a part of can be used to capture performance data but it needs to be analyzed by an external program. The PMPI has been used to create a real time visulization tool[6].

6 Benchmarking tools

There exists a lot of different benchmarking tools for OpenMPI. One or several of these will be used to measure the change in performance by runtime tuning the parameters.

6.1 SKaMPI

SKaMPI consists of three parts the benchmarking program, the postprocessing tool and the report generation tool. It can benchmark induvidual MPI operations and compare different configurations. It's very configurable. [7]

6.2 NAS Parallel Benchmarks

NAS is a collection of several different benchmarks. It was developed in 1991 by NASA to benchmark their HPC. The benchmarks are eight programs relative to science and are supposed to mimic programs normally run on HPC.[8]

6.3 OSU benchmarks

The OSU runs many microbenchmarks and reports back data such as latency and sending time for packages of different size.

6.4 Intel® MPI Benchmarks

The Intel MPI Benchmark focuses on point-to-point and global communcation operations.

7 Optimizing using machine learning

There has been work done in the field of using machine learning algorithms to make the HPC learn which parameters work best for different kinds of calculations. The two algorithms mostly used is decisions trees and neural networks. This works by analysing work being done and the application being run. Using different algorithms to test settings the system learns which settings fits what application best. These could be changed during runtime but mostly the system runs one instance of the application and acquires data, selects the appropriate parameter values and starts the complete computation.[9][10]

8 Parameters relevent in shared memory applications

According to [9] the following parameters affect performance in shared memory application:

- sm_eager_limit
- mpi_paffinity_alone
- coll_sm_control_size
- coll_tuned_use_dynamic_rules
  [insert descriptions]

4

Method

9 Theoretical basis

This thesis results will be based empirical studies and published theories that have been through a peer-review process[11].

10 Types of research questions answered

The first kind of question is in the form of means of development, meaning how to implement this in the current implementation of OpenMPI. The second kind of question is a method for analysis which is required to evaluate the quality of the implementation.

11 Pre-study

Information gathered during this phase will be used to further delimit the thesis. Several different field will be studied such as OpenMPI, MCA in OpenMPI and general dynamic runtime optimization.

In the end of the pre-study a list of variables that can be adjusted during runtime and the effect of changes in these variables can be calculated will be present. All of these variables will of course have to affect the performance. This list will be the basis for the implementation phase.

12 Implementation

13 Evaluation

The primary goal of the implementation is to result in a general speed-up over the most problems which uses OpenMPI. To measure the results one or more of the benchmarking tools listed in the theory will be used with and without the implementation. These benchmarks runs several tests that are designed to mimic real world problems that OpenMPI is used for.

To assess the change in resource utilization the benchmarks tools will test the system without the run-time optimization enabled, the results will be documented and the same test will be run with the optimization enabled and hopefully we will see a improvement in performance.

5

Results

hello from results

6

Discussion

hello from discussion

6.1 Results

6.2 Method

6.3 The work in a wider context

7

Conclusions

hello from conclusions

Bibliography

[1] Infiniband trade association. `http://www.infinibandta.org/`. Accessed: 2015-12-16.

[2] General run-time tuningn. `https://www.open-mpi.org/faq/?category=tuning`. Accessed: 2015-11-29.

[3] S. Pellegrini, R. Prodan, and T. Fahringer. Tuning mpi runtime parameter setting for high performance computing. In *Cluster Computing Workshops (CLUSTER WORKSHOPS), 2012 IEEE International Conference on*, pages 213–221, Sept 2012.

[4] Graham E. Fagg, Jelena Pjesivac-grbovic, George Bosilca, Jack J. Dongarra, and Emmanuel Jeannot. Flexible collective communication tuning architecture applied to open mpi. In *In 2006 Euro PVM/MPI*, 2006.

[5] Performance analysis tools. `https://www.open-mpi.org/faq/?category=perftools`. Accessed: 2015-11-30.

[6] Thomas RICHARD. *MPI Visualisation*. PhD thesis, The University of Edinburgh, 2011.

[7] Ralf Reussner, Peter Sanders, Lutz Prechelt, and Matthias Müller. Skampi: A detailed, accurate mpi benchmark, 1998.

[8] David H. Bailey. NAS parallel benchmarks. In *Encyclopedia of Parallel Computing*, pages 1254–1259. 2011.

[9] Simone Pellegrini, Jie Wang, Thomas Fahringer, and Hans Moritsch. Optimizing mpi runtime parameter settings by using machine learning. In Matti Ropo, Jan Westerholm, and Jack Dongarra, editors, *Recent Advances in Parallel Virtual Machine and Message Passing Interface*, volume 5759 of *Lecture Notes in Computer Science*, pages 196–206. Springer Berlin Heidelberg, 2009.

[10] Simone Pellegrini, Thomas Fahringer, Herbert Jordan, and Hans Moritsch. Automatic tuning of mpi runtime parameter settings by using machine learning. In *Proceedings of the 7th ACM International Conference on Computing Frontiers*, CF '10, pages 115–116, New York, NY, USA, 2010. ACM.

[11] W. P. Thurston. On proof and progress in mathematics. *ArXiv Mathematics e-prints*, March 1994.