REPORT

LogGP based Analysis of Collective Communication Algorithms

FOR B.TECH SIXTH SEMESTER

MINI PROJECT

BY

Geetika Bakshi (RIT2015003)
Shivangi Singhal (RIT2015011)
Bobbili Vineela Moses(RIT2015041)
Megha Mishra (RIT2015074)

UNDER THE SUPERVISION OF
Dr. JAGPREET SINGH
IIIT-ALLAHABAD

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY, ALLAHABAD

(A UNIVERSITY ESTABLISHED UNDER SEC.3 OF UGC ACT, 1956 VIDE NOTIFICATION NO. F.9-4/99-U.3 DATED 04.08.2000 OF THE GOVT. OF INDIA)

A CENTRE OF EXCELLENCE IN INFORMATION TECHNOLOGY ESTABLISHED BY GOVT. OF INDIA

CANDIDATES' DECLARATION

We hereby declare that the work presented in this report entitled LogGP based Analysis of Collective Communication Algorithms , submitted towards fulfillment of sixth semester mini project of B.Tech. (IT) at Indian Institute of Information Technology, Allahabad, is an authenticated record of our original work carried out under the guidance of Dr.Jagpreet Singh.Due acknowledgements have been made in the text to all other material used.The project was done in full compliance with the requirements and constraints of the prescribed curriculum.

Place : Allahabad
Date: / /

Geetika Bakshi (RIT2015003) :

Shivangi Singhal (RIT2015011) :

Bobbili Vineela Moses (RIT2015041) :

Megha Mishra (RIT2015074) :

CERTIFICATE FROM SUPERVISOR

This is to certify that the statement made by the candidates is correct to the best of my knowledge and belief. The project titled ”LogGP based Analysis of Collective Communication Algorithms ” is a record of candidates’ work carried out by them under my guidance and supervision. I do hereby recommend that it should be accepted in the fulfilment of the requirements of a sixth semester mini project at IIIT Allahabad.

Date: / /

(Dr. JAGPREET SINGH)

ABSTRACT

We describe our work as enhancing the runtime performance of collective communication operations. For every collective operation, we can utilize multiple algorithms based on the message size(to be operated on), with the goal of minimizing latency for short messages and minimizing bandwidth use for long messages. We have analyzed few of the algorithms for MPI collective operation Broadcast and Reduce.

In this project, we propose, develop and implement a tool for choosing an appropriate algorithm for a particular collective operation under particular parameters.We will be using a new model of parallel computation i.e. the LogGP model and use it to analyze a number of algorithms.It models communication performance through the use of four parameters: the communication latency (L), overhead (o), bandwidth (g), and the number of processors (P). Communication is modeled by point-to-point messages of some fixed short size. Thus, the model has implicitly a fifth parameter, the message size w.

All the results used for the comparisons are calculated under one allocation of resources, so as to avoid any errors. Our analysis could lead to results which might help to develop other collective operations or algorithms.We likewise have proposed and executed another algorithm(Random2Tree) which is superior to every single other algorithm examined in our work. This report examines about how much and why a algorithm is more qualified than some other in a specific situation.

Contents

Chapter 1

Introduction

Parallel programs for distributed memory machines can be structured as sequential computations plus calls to a small number of (collective) communication primitives. The newest version of the Message Passing Interface (MPI) standard, the de-facto standard for distributed-memory parallel programming, offers a set of commonly-used collective communications.

Analysis is done under the LogGP model wherein the processors communicate by point-to-point messages and communication performance is characterized by the following parameters:

- L: an upper bound on the Latency, incurred in sending a message from its source processor to its target processor.
- o: the overhead, defined as the length of time that a processor is engaged in the transmission or reception of each message; during this time the processor cannot perform other operations.
- g: the gap between messages, defined as the minimum time interval between consecutive message transmissions or consecutive message receptions at a processor. The reciprocal of g corresponds to the available per processor communication bandwidth for short messages.
- G: the Gap per byte for long messages, defined as the time per byte for a long message. The reciprocal of G characterizes the available per processor communication bandwidth for long messages.
- P: the number of processor/memory modules

Chapter 2

Motivation

Many algorithms and the optimized implementation has been designed for various collective communication purposes, However none of them can be declared as optimal as the performance of these algorithms is dependent on many parameters. An algorithm optimal in some situation might not be optimal in some other situation. So the task of choosing an algorithm for a particular operation is tricky.

A generic framework to design close-to-optimal schedules for predefined as well as neighborhood collective operations would be a valuable contribution to the state of the art and will surely help to choose an appropriate algorithm for the particular collective operation.

Chapter 3

Problem definition and Objectives

3.1 Problem definition

To implement and to perform a LogGP based analysis for different collective algorithms.Also to provide a framework which would help to choose an appropriate algorithm based on different parameters.

3.2 Objectives

- To analyse and implement broadcast and reduce collective algorithms.
- To analyze the algorithms for different operations on different values of parameters such as message size and number of processors.
- To give a LogGP based analysis of the algorithms, which is better in what conditions.
- To point out gaps and define future research topics based on our analysis of the algorithms.

Chapter 4

Literature Survey

S.NO.	Title	Authors	Summary
1.	Two-Tree Algorithms for Full Bandwidth Broadcast, Reduction and Scan	Peter Sanders and Jochen Speck and Jesper Larsson Traff	Presents a new algorithmic idea to concurrently utilize bandwidth with two tree implementation.
2.	Optimization of Collective Communication Operations in MPICH	Rajeev Thakur,Rolf Rabenseifner,William Grop	Gives an overview of basic collective algorithms
3.	LogGP: Incorporating Long Messages into the LogP Model	Albert Alexandrov, Mihai F. Ionescu, Klaus E. Schauser, and Chris Scheiman	Presents a LogGP based analysis of algorithms

Chapter 5

Methodology

5.1 Operations

The newest version of the Message Passing Interface (MPI) standard, the de-facto standard for distributed-memory parallel programming, offers a set of commonly-used collective communications. These operations cover most use-cases discovered in the last two centuries and we thus use them as a representative sample for our analyses. In general, collective patterns reflect key characteristics of parallel algorithms at large numbers of processing elements, for example, parallel reductions are used to implement parallel summation and all-to-all is a key part of many parallel sorting algorithms and linear transformations.

Few of the Collective operations are :

- BroadCast[2]: A single message of size s is distributed (copied) from a designated root process to all other P-1 processes. Can be implemented with non-personalized tree algorithms. Binomial and binary trees are commonly used for implementations of small-message broadcast and reduction. Large- message operations can be implemented with double trees. We will discuss these algorithms in next section.
- Reduction [2]: Each process contributes a message of size s. The associative operator combines all P messages into a single result of size s at a designated root process.
$$r = m_1 + m_2 + ... + m_P$$
- Scan [2]: Each process specifies a message of size s and received the partial sum of all messages specified by processes with a lower id than itself.
- Neighborhood Collectives [5]: The programmer can specify any communication pattern and in this way build his own collective communication operation.

5.2 Algorithms

Every collective algorithm exploits a specific virtual topology, i.e., a directed graph which represents message propagation between the processes. Trees can be utilized to actualize any collective communication, processes are organized in a tree shape and messages are flowing from parents to children or the other way around, depending on the type of collective operation.

Trees are regularly utilized for imparting little messages because in most cases, leaf processes just recieve messages and thus are not able to use their own send bandwidth. Simple pipelines that reduces the number of leaves provides better and easy solutions for very large message sizes. Also double-tree algorithms can be used which improve the latency over such simple pipelines.

Various algorithms have been designed for the purpose. Few of them are :

- **Recursive Doubling Algorithm** In the first step, pairs of neighboring processes exchange data; in the second step, pairs of processes at distance 2 apart exchange data; in the third step, processes at distance 4 apart exchange data; and so forth. In step 1, processes exchange all the data except the data needed for their own result (n n p); in step 2, processes exchange all data except the data needed by themselves and by the processes they communicated with in the previous step (n 2n p); in step 3, it is (n 4n p); and so forth. We use this algorithm for very short messages (¡ 512 bytes).

- **Reduce and Allreduce** : MPI Reduce performs a global reduction operation and returns the result to the specified root, whereas MPI Allreduce returns the result on all processes. The old algorithm for reduce in MPICH uses a binomial tree, which takes lg p steps, and the data communicated at each step is n.

- **Pipelined Tree Algorithm** : Pipeline calculations depend on the plan to partition a vast message into numerous little pieces(chunks) and to appropriate these pieces among processors in a pipeline fashion. Diverse virtual topologies can be used for transmitting the information.

  Linear pipelines are least complex while tree pipelines permit to diminish latencies. Figure 5.1 shows the topology for Pipelined Binary Tree algorithm. We accept for this situation that the send and get overheads(o) can be charged at the same time. Pipelines are frequently utilized as building hinders for more mind boggling calculations.

- **Two Trees Algorithm[1]** : While pipelined trees improve the overall bandwidth utilization, they are still not optimal. The reason for this is that the leaves in the tree never transmit messages and thus do not contribute their bandwidths. To use the leaves bandwidth, one can employ two trees with different structure (leave nodes) such that each node sends eventually. Figure 5.2 demonstrates how a Two Tree topology is constructed.

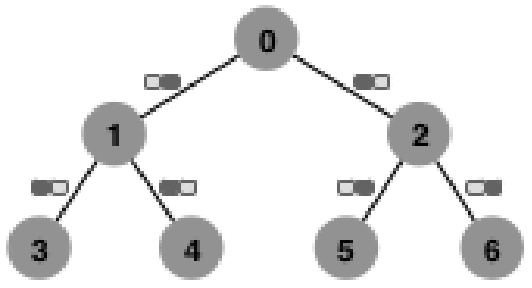

Figure 5.1: Pipeline Binary Tree

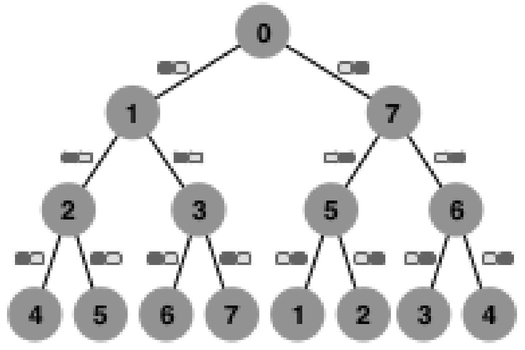

Figure 5.2: Binary Two Tree

A simple lower bound for the run-time of all algorithms is ((o log P) + sG) [2] because data needs to reach all processes and data must be sent at least once. Apart from these several researchers developed algorithms that are tuned to particular properties of the machine, several algorithms that specialize to the network topology exist. Some others utilize special hardware features.

Chapter 6

Requirements

6.1 Hardware Used

- **Piz Daint (supercomputer)**: Piz Daint is a supercomputer in the Swiss National Supercomputing Centre, ranked 8th on the TOP500 ranking of supercomputers until the end of 2016, higher than any other supercomputer in Europe.
  Processors Used - 512.

6.2 Software Requiremnt

- Linux Operating System.
- MPICH-3.2 library.
- c9, cloud service provider.
- SublimeText.

Chapter 7

Implementation

The algorithms discussed in the methodology are the few of the algorithms for collective communication. We use the single-ported, full-duplex variant of this model where a PE can simultaneously send data to one PE and receive data from a possibly different PE (sometimes called the simultaneous send -receive model.

7.1 Broadcst

A broadcast sends a message from a specified root PE (processing element) to all other PEs.Assume PE i wants to send a message to everybody. See also Figure 2. We build the trees T1 and T2 for the remaining PEs (renumbering them 0..p 2). The tree edges are directed from the root towards the leaves. T1 broadcasts the first half of the message using pipelined binary tree broadcasting. Concurrently, T2 broadcasts the second half using the same algorithm. PE i takes turns dealing out one of k pieces of the left and right halves to the roots of T1 and T2 respectively.

7.2 Reduce

Reduction is the mirror-image of broadcasting. Assume PE i wants to compute the sum over all messages. We build the trees T1 and T2 for the remaining PEs. The tree edges are directed from the leaves towards the roots. T1 sums the first half of the message for PEs with index from 0, . . . , p 1 i and T2 sums the second half. PE i takes turns receiving pieces of summed data from the roots of T1 and T2 and adds its own data. Disregarding the cost of arithmetic operations, we have the same execution time bound as for broadcasting.

7.3 Pipeline Binary Tree

As discussed above, this algorithm divides the message into chunks of small messages. Then reduce these chunks through the binary tree in pipeline fashion. All the sends and receives are non-blocking.

Steps involved in this implementation are:

- Set up all sends for the leaf process.
- Set up all receives for non-root process.
- Whenever a chunk is received, set up all non-blocking sends for that process.
- Repeat previous step until all sends are completed.

7.4 Two Tree(Complete Non Blocking)

This implementation of Two Tree Algorithm described above uses non-blocking send and non-blocking receives for the communication process. Thus it does not provide the synchronization in the broadcast process.One tree will be involved in the exchange of even chunks while the other will process odd chunks. Steps involved in this implementation are:

- Set up all sends for the leaf process.
- Set up all receives for non-leaf process.
- Whenever a chunk is received, set up all sends for that process.
- Repeat previous step until all sends are completed.

This implementation may not guarantee complete usage of bandwidth, but reduces the time for synchronization.

Chapter 8

Conclusion and Future scope

8.1 Conclusion

1. This study provides an overview of existing collective algorithms and implementations.
2. We have derived run-time results for each algorithm for different number of processors for message size 1MB.
3. Our implementation, results and comparisons provides fundamental insights into the nature of these algorithms.
4. The implementations needs to be tested on missing values to get a clear picture.

8.2 Future scope

1. The analysis can be used to design other algorithms for collective operations.More emphasis will be laid on the implementation of all reduce algorithms.
2. Other models such as memory model and energy model can also be used to analyze the algorithms.
3. Can establish a discussion basis for the fundamental trade-offs between run-time, energy, and memory consumption.

References

[1] Peter Sanders, J. Speck, Jesper Larsson, *Two-Tree Algorithms for Full Bandwidth Broadcast, Reduction and Scan*. Parallel Computing, Vol-35, No.-12, 2009.

[2] Torsten Hoefler, Timo Schneider, Andrew Lumsdaine, *LogGP in Theory and Practice - An In-depth Analysis of Modern Interconnection Networks and Benchmarking Methods for Collective Operations* Simulation Modelling Practice and Theory, Volume-17, 2009

[3] Torsten Hoefler and Dmitry Moor, *Energy, Memory, and Runtime Tradeoffs for Implementing Collective Communication Operations* Supercomputing Frontiers and Innovations, Volume-1, 2014

[4] MPI Forum. MPI: A Message-Passing Interface standard. Version 3.0, 2012.

[5] Torsten Hoefler and Timo Schneider, *Optimization Principles for Collective Neighborhood Communications* SC '12 Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis