Simulation Modelling Practice and Theory xxx (2015) xxx–xxx

Contents lists available at ScienceDirect

Simulation Modelling Practice and Theory

journal homepage: www.elsevier.com/locate/simpat

Topology-oblivious optimization of MPI broadcast algorithms on extreme-scale platforms

Khalid Hasanov [a,*], Jean-Noël Quintin [b], Alexey Lastovetsky [a]

[a] University College Dublin, Belfield, Dublin 4, Ireland
[b] Extreme Computing R&D, Bull SAS, Paris, France

ARTICLE INFO

Article history:
Available online xxxx

Keywords:
MPI
Broadcast
BlueGene
Grid'5000
Extreme-scale
Communication
Hierarchy

ABSTRACT

Significant research has been conducted in collective communication operations, in particular in MPI broadcast, on distributed memory platforms. Most of the research efforts aim to optimize the collective operations for particular architectures by taking into account either their topology or platform parameters. In this work we propose a simple but general approach to optimization of the legacy MPI broadcast algorithms, which are widely used in MPICH and Open MPI. The proposed optimization technique is designed to address the challenge of extreme scale of future HPC platforms. It is based on hierarchical transformation of the traditionally flat logical arrangement of communicating processors. Theoretical analysis and experimental results on IBM BlueGene/P and a cluster of the Grid'5000 platform are presented.

1. Introduction

The Message Passing Interface (MPI) [1] is one of the core building blocks of scientific software libraries for parallel applications. For example, PETSc [2] library, which is used for simulation and modeling in different application domains such as Nanosimulations, Aerodynamics, Geosciences, Computational Fluid Dynamics and others, utilizes MPI for all inter-process data communication operations.

Depending on the application, MPI collective communication operations can provide significant performance improvements over MPI point-to-point communication routines. One of the commonly used collective operations, MPI broadcast, is used in a variety of basic scientific kernels such as parallel matrix–matrix multiplication, LU factorization and along with others. During a broadcast operation, the root process sends a message to all other processes in the specified group of processes. The implementations of the broadcast operation in MPICH [3] and Open MPI [4] are typically based on linear, binary, binomial and pipelined algorithms [5]. The linear algorithms are not good for large numbers of processes, the binary and binomial algorithms are not efficient for large data sizes. On the other hand, pipelined algorithms are more efficient for larger numbers of processes and data sizes. Other widely used broadcast algorithms are scatter-ring-allgather and scatter-recursive-doubling-allgather [6], which have been implemented in MPICH.

In addition, significant research has been done in optimizing MPI broadcast for some specific platforms. The research works in [7,8] present efficient implementations of MPI broadcast, which use native Infiniband multicast. The Cheetah

* Corresponding author.
E-mail addresses: khalid.hasanov@ucdconnect.ie (K. Hasanov), jean-noel.quintin@bull.net (J.-N. Quintin), Alexey.Lastovetsky@ucd.ie (A. Lastovetsky).

http://dx.doi.org/10.1016/j.simpat.2015.03.005
1569-190X/

framework offers a hierarchical collective communication framework that takes advantage of hardware-specific data-access mechanisms [9]. IBM BlueGene comes with its own platform specific optimizations of MPI collectives [10]. A comprehensive overview of optimization techniques for collectives on heterogeneous HPC platforms using broadcast as a use case can be found in [11]. A recent research work in [12] presents a generic framework to optimize the performance of MPI collectives on Intel MIC clusters.

Theoretically optimal MPI broadcast algorithms have been an active research subject as well. One of the early results in this area is the spanning binomial tree algorithm proposed by Jonson and Ho [13]. Later, the research work in [14] introduced another theoretically optimal broadcast algorithm based on fractional trees. The work in [15] is similar to the algorithm of Jonson and Ho when the number of processes is a power of two and extends it to an arbitrary number of processes.

The number of processors in HPC systems has increased by three orders of magnitude over the past two decades. This increase in scale is accompanied by the increase in complexity and diversity of communication layers in these platforms. Both these factors raise the communication cost of the execution of traditional message-passing data-parallel applications on modern and future HPC platforms. The main body of recent research in optimization of the communication cost of scientific algorithms and applications focuses on the complexity of particular platforms and proposes solutions specific to these platforms. In contrast to this approach, we focus on the scale of HPC platforms rather than their specific complexity and propose a solution that is universally applicable to all HPC platforms. The proposed solution is inspired by our previous study on parallel matrix multiplication on large-scale distributed memory platforms [16]. It provides a simple and general technique to optimize the legacy scientific MPI-based applications without redesigning them. The method is to design and implement simple hierarchical modifications of MPI collective operations significantly reducing their execution time, especially at extreme scale. This will lead to substantial acceleration of the applications that extensively use MPI collectives. In this article, we apply this method to the MPI broadcast operation as an initial step to achieve this goal. The article is a substantially extended version of our workshop paper [17].

The contributions of the presented work are as follows:

- A simple and general technique to optimize the MPI broadcast operation.
- The approach can be applied to any legacy applications using MPI broadcast with a marginal code modification.
- Theoretical and experimental study of the hierarchical modifications of eight existing broadcast algorithms in MPICH and Open MPI.

2. Preliminaries and previous work

In the rest of this paper the amount of data to be broadcast and the number of MPI processes will be denoted by m and p respectively. It is assumed that the network is fully connected, bidirectional and homogeneous. The cost of sending a message of size m between any two processes is modeled by the Hockney's model [18] as $\alpha + m \times \beta$. Here α is the startup cost or latency, while β is the reciprocal bandwidth.

2.1. Previous work

This section recalls eight MPI broadcast algorithms implemented in MPICH and Open MPI, namely flat, linear, pipelined, binary, split-binary, binomial tree, scatter-ring-allgather, and scatter-recursive-doubling-allgather broadcast algorithms. The first six algorithms are implemented in Open MPI and the last three algorithms are implemented in MPICH. The derivations of the theoretical costs of these algorithms are not the original contribution of this work and can be found in [19].

2.1.1. Flat tree broadcast algorithm

Flat tree is the simplest MPI broadcast algorithm, in which the root node sequentially sends the same message to all the nodes participating in the broadcast operation. This algorithm does not scale well for large communicators. Its cost is estimated as $(p-1) \times (\alpha + m \times \beta)$.

2.1.2. Linear tree broadcast algorithm

In this algorithm each node sends or receives at most one message. Since the root does not receive the message, it is called chain algorithm sometimes. Theoretically, its cost is the same as that of the flat tree algorithm: $(p-1) \times (\alpha + m \times \beta)$.

2.1.3. Pipelined linear tree broadcast algorithm

The performance of the linear tree algorithm can be improved by splitting and pipelining the message. In this case, each process can start sending a part of the message after it receives the first part of the message. The run time of the algorithm is

Please cite this article in press as: K. Hasanov et al., Topology-oblivious optimization of MPI broadcast algorithms on extreme-scale platforms, Simulat. Modell. Pract. Theory (2015), http://dx.doi.org/10.1016/j.simpat.2015.03.005

equal to $(X+p-2)\times(\alpha+\frac{m}{X}\times\beta)$. Here it is assumed that a broadcast message of size m is split into X segments and in one step of the algorithm a segment of size $\frac{m}{X}$ is broadcast among p processes.

2.1.4. Binary tree broadcast algorithm

Lets assume a full and complete binary tree of height h is given and the run time of the broadcast at the height h is denoted by $T(h)$. Then, the run time of the broadcast on a single node is zero, $T(0)=0$. The last node receiving the message at height h will send two messages to its children at height $h+1$, therefore $T(h+1)=T(h)+2(\alpha+m\beta)$. It can easily be shown that $T(h)=2\times h\times(\alpha+m\beta)$ and the number of nodes of the binary tree is $2^{h+1}-1$, therefore the overall run time will be $2\times(\log_2(p+1)-1)\times(\alpha+m\times\beta)$.

2.1.5. Split-binary tree broadcast algorithm

The split-binary tree algorithm [19] consists of forwarding and exchange phases. In the forwarding phase, the root process splits the original message in half, then each of the halves is sent down the left and right subtree respectively. In the exchange phase, each process in both trees exchanges its message with the corresponding pair process from the other tree. The time to complete a broadcast operation with this algorithm is equal to the sum of the times spent in forwarding and exchange phases. Thus, its cost on a full and complete binary tree, where the number of processes is one less than an exact power of two, will be as follows: $2\times(\log_2(p+1)-2)\times(\alpha+m\times\beta)+\alpha+\frac{m}{2}\times\beta$. A detailed analysis of this algorithm with segmented messages can be found in [19].

2.1.6. Binomial tree broadcast algorithm

Lets assume a binomial tree of height h is given and the number of nodes is 2^h. The number of nodes sending and receiving is doubled for each value of the height in the algorithm. For example, run time $T(h)$ will change as follows: $T(0)=0, T(1)=\alpha+m\beta, T(2)=2(\alpha+m\beta),\ldots T(h)=h(\alpha+m\beta)$. If we consider that $p=2^h$ then the run time of the algorithm will be $\log_2(p)\times(\alpha+m\times\beta)$.

2.1.7. Scatter-ring-allgather broadcast algorithm

The run time of this algorithm is as follows: $(\log_2(p)+p-1)\times\alpha+2\frac{p-1}{p}\times m\times\beta$. The algorithm consists of scatter and allgather phases. The message is scattered by a binomial tree algorithm in the first phase, and in the next phase a ring algorithm for allgather is used to collect all segments from all processes. It is used in MPICH for large message sizes.

2.1.8. Scatter-recursive-doubling-allgather broadcast algorithm

This algorithm is very similar to the previous one except the allgather uses a recursive doubling algorithm. It is used in MPICH for medium-size messages. However, the ring algorithm is more efficient than this one for large message sizes because of its nearest-neighbor communication pattern [20]. Its cost is estimated as $2\times\log_2(p)\times\alpha+2\frac{p-1}{p}\times m\times\beta$.

3. Hierarchical optimization of MPI broadcast algorithms

This section introduces a simple but general optimization of the MPI broadcast algorithms. This optimization technique is inspired by our previous study on the optimization of the communication cost of parallel matrix multiplication on large-scale distributed memory platforms [16].

The proposed optimization is based on the hierarchical arrangement of the processes participating in the broadcast into logical groups. For simplicity we assume that the number of groups divides the number of MPI processes and can change between one and p. Let G be the number of groups. Then there will be $\frac{p}{G}$ MPI processes per group. Fig. 1 shows an arrangement of 12 processes in a non-hierarchical way and a hierarchical grouping of 12 processes into 3 groups of 4 processes. The hierarchical optimization has two steps: in the first step a group leader is selected for each group and the broadcast is performed between the group leaders (see Fig. 2), and in the next step the leaders start broadcasting inside their own group (in this example between 4 processes). The grouping can be done by taking the topology into account as well. However, in this work the grouping is topology-oblivious and the first process in each group is selected as the group leader. The broadcasts inside different groups are executed in parallel. While in general different broadcast algorithms can be used inside and between groups, this work focuses on the case where the same broadcast algorithm is employed in both levels. Algorithm 1 shows the pseudocode of the optimized broadcast. Line 4 calculates the root for the broadcast inside the groups. Then line 5 creates a sub-communicator of G processes among the groups and line 6 creates a sub-communicator of $\frac{p}{G}$ processes inside the groups. Our implementation uses the MPI_Comm_split MPI routine to create new sub-communicators.

Please cite this article in press as: K. Hasanov et al., Topology-oblivious optimization of MPI broadcast algorithms on extreme-scale platforms, Simulat. Modell. Pract. Theory (2015), http://dx.doi.org/10.1016/j.simpat.2015.03.005

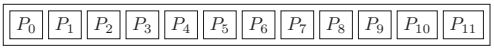

Fig. 1. Arrangement of processes in MPI broadcast.

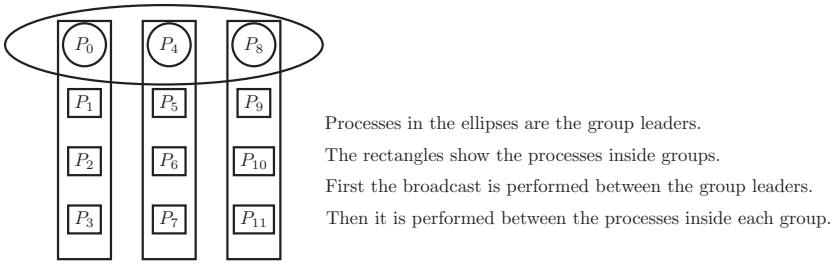

Fig. 2. Arrangement of processes in the hierarchical broadcast.

Algorithm 1. Hierarchical modification of an MPI broadcast algorithm.

Data: p - Number of processes
Data: G - Number of groups
Data: buf - Message buffer
Data: *count* - Number of entries in buffer (integer)
Data: *datatype* - Data type of buffer
Data: *root* - Rank of broadcast root
Data: *comm* - MPI Communicator
Result: All the processes have the message of size m
begin

```
1  MPI_Comm comm_outer            /* communicator among the groups */
2  MPI_Comm comm_inner            /* communicator inside the groups */
3  int root_inner             /* root of broadcast inside the groups */
4  root_inner = Calculate_Root_Inner(G, p, root, comm)
5  comm_outer = Create_Comm_Between_Groups(G, p, root, comm)
6  comm_inner = Create_Comm_Inside_Groups(G, p, root_inner, comm)
7  MPI_Bcast(buf, count, datatype, root, comm_outer)
8  MPI_Bcast(buf, count, datatype, root_inner, comm_inner)
```

3.1. Hierarchical flat and linear tree broadcast

If we group the processes in the hierarchical way and apply the flat or linear tree broadcast algorithm among G groups and inside the groups among $\frac{p}{G}$ processes then the overall broadcast cost will be equal to their sum:

$$F(G) = (G-1) \times (\alpha + m \times \beta) + \left(\frac{p}{G} - 1\right) \times (\alpha + m \times \beta) = \left(G + \frac{p}{G} - 2\right) \times (\alpha + m \times \beta) \quad (1)$$

Here $F(G)$ is a function of G for a fixed p. Its derivative is equal to $\left(1 - \frac{p}{G^2}\right) \times (\alpha + m \times \beta)$. It can be shown that $G = \sqrt{p}$ is the minimum of the function $F(G)$ as in the interval $(1, \sqrt{p})$ the function decreases, and in the interval $(\sqrt{p}, p)$ it increases. If we consider $G = \sqrt{p}$ in the derivative the optimal value of the broadcast cost will be as follows:

$$F(\sqrt{p}) = (2\sqrt{p} - 2) \times (\alpha + m \times \beta) \quad (2)$$

3.2. Hierarchical pipelined linear tree broadcast

In the same way, if we add two pipelined linear tree broadcast costs among G groups and inside the groups among $\frac{p}{G}$ processes then the overall communication cost for the hierarchical pipelined linear tree will be as follows:

$$F(G) = \left(2X + G + \frac{p}{G} - 4\right) \times \left(\alpha + \frac{m}{X} \times \beta\right) \quad (3)$$

It can be shown that $G = \sqrt{p}$ is the minimum point again and at this point the cost will be as follows:

Please cite this article in press as: K. Hasanov et al., Topology-oblivious optimization of MPI broadcast algorithms on extreme-scale platforms, Simulat. Modell. Pract. Theory (2015), http://dx.doi.org/10.1016/j.simpat.2015.03.005

$$F(\sqrt{p}) = (2X + 2\sqrt{p} - 4) \times \left(\alpha + \frac{m}{X} \times \beta\right) \tag{4}$$

3.3. Hierarchical binary and binomial tree broadcast

Let us apply the binary tree algorithm among G groups and inside the groups among $\frac{p}{G}$ processes. The broadcast cost among G groups and inside the groups will be $2\log_2(G) \times (\alpha + m \times \beta)$ and $2\log_2\left(\frac{p}{G}\right) \times (\alpha + m \times \beta)$ respectively. If we add these two costs and consider that $\log_2\left(\frac{p}{G}\right) = \log_2(p) - \log_2(G)$ then the cost of the hierarchical binary broadcast algorithm will be the same as the corresponding non-hierarchical broadcast algorithm.

Because of the same reason the hierarchical modification of the binomial tree algorithm does not improve the non-hierarchical binomial tree algorithm.

3.4. Hierarchical scatter-ring-allgather broadcast

If we apply the scatter-ring-allgather algorithm in the same way we can get the following formula:

$$F(G) = \left(log_2(p) + G + \frac{p}{G} - 2\right) \times \alpha + 2 \times m \times \left(2 - \frac{1}{G} - \frac{G}{p}\right) \times \beta \tag{5}$$

Let us find the optimal value of the $F(G)$ function: $F'(G) = \frac{g^2 - p}{G^2} \times \left(\alpha - \frac{2m\beta}{p}\right)$. It is clear that if

$$\frac{\alpha}{\beta} > \frac{2m}{p} \tag{6}$$

then $G = \sqrt{p}$ is the minimum point of the $F(G)$ function in the interval $(1, p)$. The value of the function at this point will be as follows:

$$F(\sqrt{p}) = (log_2(p) + 2\sqrt{p} - 2) \times \alpha + 2 \times m \times \left(2 - \frac{2}{\sqrt{p}}\right) \times \beta \tag{7}$$

3.5. Hierarchical scatter-recursive-doubling-allgather broadcast

The hierarchical modification of this algorithm has a higher theoretical cost compared to the cost of the original algorithm (see Section 2.1.8). The latency term is increased two times and the bandwidth term is increased as well: $F(G) = 2 \times log_2(p) \times \alpha + 2 \times m \times \left(2 - \frac{1}{G} - \frac{G}{p}\right) \times \beta$. It can be shown that $G = \sqrt{p}$ is the extremum point. However the minimum point is either $G = 1$ or $G = p$, in which case the hierarchical algorithm will have exactly the same cost as the original algorithm.

3.6. Hierarchical split-binary tree broadcast

We take $p + 1 \approx p$ in the cost function of the split-binary tree algorithm to derive the cost of its hierarchical transformation. It can be shown that the overall cost will be slightly worse than that of the original algorithm itself (see Section 2.1.5) and with our simple theoretical model it does not depend on the number of groups: $2 \times (\log_2(p) + X - 4) \times (\alpha + \beta \times \frac{m}{X}) + 2 \times (\alpha + \beta \times \frac{m}{2})$. Therefore, the hierarchical algorithm should use 1 or p groups in this case.

3.7. Summary of theoretical analysis

Thus, the hierarchical transformation of the flat, chain, pipeline and scatter-ring-allgather algorithms theoretically reduces the communication cost of the corresponding original algorithms. The communication cost of the binary, binomial, scatter-recursive-doubling-allgather and split-binary tree algorithms get their best performance when the number of groups is one or equal to the number of processes.

4. Experiments

4.1. Experiments on BlueGene/P

Some of our experiments were carried out on the Shaheen BlueGene/P at the Supercomputing Laboratory at King Abdullah University of Science&Technology (KAUST) in Thuwal, Saudi Arabia. Shaheen has 16 racks with a total of 16,384 nodes. Each node is equipped with four 32-bit, 850 MHz PowerPC 450 cores and 4 GB DDR memory. The BlueGene/P

Please cite this article in press as: K. Hasanov et al., Topology-oblivious optimization of MPI broadcast algorithms on extreme-scale platforms, Simulat. Modell. Pract. Theory (2015), http://dx.doi.org/10.1016/j.simpat.2015.03.005

(BG/P) architecture provides a three-dimensional point-to-point BlueGene/P torus network which interconnects all compute nodes and global networks for collective and interrupt operations. The use of this network is integrated into the BG/P MPI implementation. BlueGene/P MPI is based on MPICH which uses three different broadcast algorithms depending on the message size and the number of processes in a broadcast operation [20]:

- binomial tree algorithm – when the message size is less than 12 kB or when the number of processes is less than eight.
- scatter-recursive-doubling-allgather algortihm – when the message size is less than 512 kB and the number of processes is a power-of-two.
- scatter-ring-allgather algorithm (we will call it SRGA) – otherwise, for long messages greater than or equal to 512 kB or with non power-of-two number of processes.

Despite the referenced paper [20] was published more than a decade ago it still reflects the current version of MPI broadcast operation implemented in MPICH according to its source code.

In addition to the algorithms implemented in MPICH, the broadcast operation on BG/P comes with different optimizations and algorithms specifically for the BG/P itself. Namely, if the communicator is MPI_COMM_WORLD it uses the BG/P collective tree network which supports hardware accelerated collective operations such as broadcast and all-reduce, and otherwise depending on the communicator shape either a rectangular broadcast algorithm or the broadcast algorithms from MPICH are used [10]. However, algorithms for some fundamental scientific applications such as parallel matrix multiplication, LU factorization does not use MPI_COMM_WORLD in their main communication steps, for example, it is more typical to use sub-communicators for rows and columns in a two-dimensional arrangement of processes. On the other hand, the rectangular broadcast is used only for rectangular shaped sub-communicators which strongly depends on the mapping of the processes into the physical topology and depending on the allocated BG/P partition can be arbitrary. Furthermore, the optimal mapping of processes to network hardware is not a trivial task and is a separate research area itself. The proposed optimization in this work is more general and topology-oblivious.

We present experiments with the corresponding hierarchical modifications of the scatter-ring-allgather algorithm and the native MPI broadcast operation. Experiments with the binomial and scatter-recursive-doubling-allgather algorithms demonstrated only slight fluctuations as expected theoretically.

While performance modeling and analysis of the BG/P-specific broadcast algorithms and optimizations are beyond the scope of this paper, we present some experiments with the native BG/P broadcast operation as an initial research in that direction. The experiments have been done with different configurations, message sizes from 1 kB up to 16 MB and the number of MPI processes from 8 up to 6142. The number of the allocated BG/P nodes was 6144, however we deliberately excluded two of them and used 6142 nodes by creating sub-communicators to avoid the case with MPI_COMM_WORLD. Because of space restrictions we present results mainly for 2048 and 6142 processes and message sizes of 512 kB and 2 MB. Figs. 3 and 4 show experiments with the scatter-ring-allgather broadcast with message sizes of 512 kB and 2 MB respectively. The improvement with 512 kB on 2048 nodes is 1.87 times, however with a message size of 2 MB there is a performance drop. On the other hand, according to the formula (6) $\left(\text{i.e.}\,\frac{\alpha}{\beta} > \frac{2m}{p}\right)$ if we fix the message size, for a larger number of processes the hierarchical transformation should improve the performance. This is validated with the experiments: Fig. 5 shows that for a message size of 512 kB the speedup increases up to 3.09 times on 6142 nodes and unlike on 2048 nodes, the hierarchical algorithm outperforms the original algorithm with a message size of 2 MB as well. In addition, if we put the platform and algorithm parameters in formula (5), we will see that the theoretically expected plots of the hierarchical algorithm will be parabola-like as well (Fig. 6). Experiments with the native BG/P MPI broadcast operation are given in Figs. 7 and 8. As it is already mentioned that during these experiments a sub-communicator of size 6142 was created from an MPI_COMM_WORLD of size 6144 to disable BG/P optimizations for MPI_COMM_WORLD. As a result the native BG/P MPI broadcast operation is worse than the scatter-ring-allgather broadcast with a message size of 2 MB.

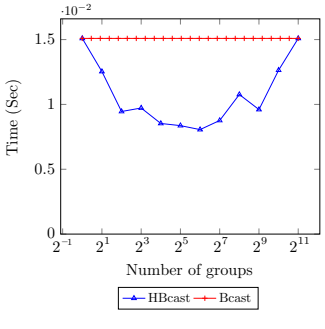

Fig. 3. Hierarchical SRGA bcast on BG/P. m = 512 kB and p = 2048.

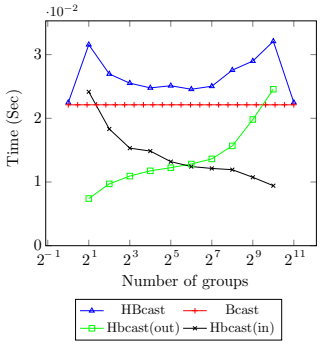

Fig. 4. Hierarchical SRGA bcast on BG/P. m = 2 MB and p = 2048.

Please cite this article in press as: K. Hasanov et al., Topology-oblivious optimization of MPI broadcast algorithms on extreme-scale platforms, Simulat. Modell. Pract. Theory (2015), http://dx.doi.org/10.1016/j.simpat.2015.03.005

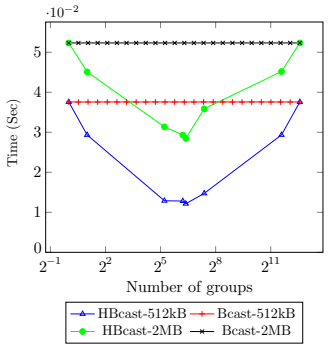

Fig. 5. Hierarchical SRGA bcast on BG/P. $m = 512$ kB and $m = 2$ MB. $p = 6142$.

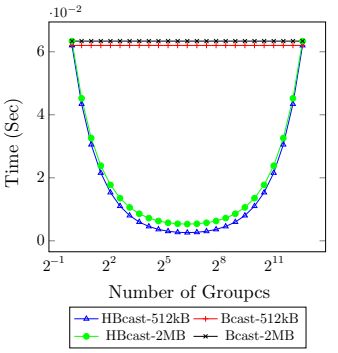

Fig. 6. Theoretical plot of SRGA bcast. $m = 512$ kB and $m = 2$ MB. $p = 6142$.

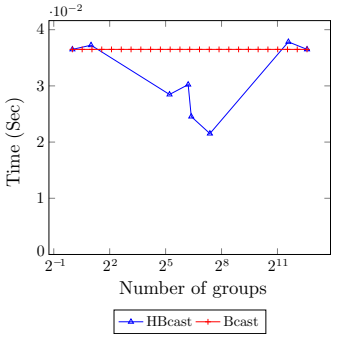

Fig. 7. Hierarchical bcast on BG/P. $m = 512$ kB and $p = 6142$.

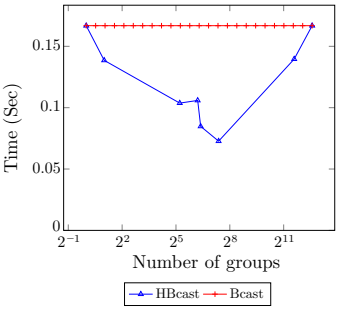

Fig. 8. Hierarchical bcast on BG/P. $m = 2$ MB and $p = 6142$.

It is obvious that the time between the groups will increase if the number of groups increases, however it is the opposite inside the groups. Fig. 4 confirms that by showing the broadcast times separately spent inside and between groups as well.

Our theoretical models do not take sub-communicator creation overheads into account. However, MPI_Comm_split is also a collective operation and depending on the number of groups it makes different contributions into the overall time of the hierarchical broadcast. For example, in Fig. 4 we do not have the expected upside-down parabola-like shape because of the additional overhead from MPI_Comm_split operations. The reason is that when the number of groups is 2 or 1024 the total cost of creation of the two sub-communicators exceeds the gains due to the hierarchical optimization. Fig. 9 demonstrates these results. If the reduction of the execution time due to the optimization is greater than the overhead itself then the sub-communicator creation times will be well compensated by the reduction. It is the case in the experiments with 512 kB (Fig. 3).

Fig. 10 presents the results of experiments with scatter-ring-allgather and native MPI broadcast operations. It shows the speedup due to the hierarchical optimization of these operations. Here the number of processes is fixed to be equal to 6142, and the message size changes from 1 kB up to 16 MB. The experiments with 2048 processes have more data points than that of 6142 processes. The reason for that is we take only the factors of the number of processes as group numbers.

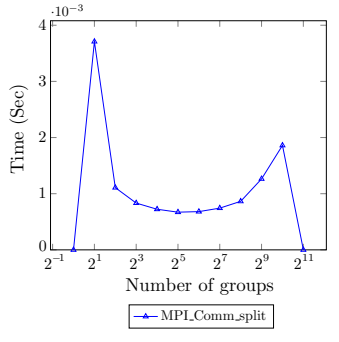

Fig. 9. MPI_comm_split time on BG/P. $p = 2048$.

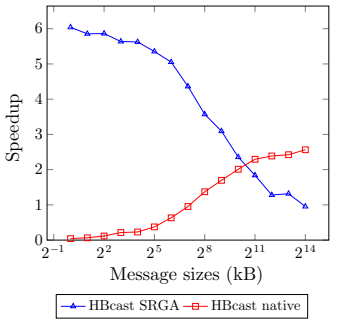

Fig. 10. Speedup of hierarchical bcast on BG/P, $p = 6142$.

Please cite this article in press as: K. Hasanov et al., Topology-oblivious optimization of MPI broadcast algorithms on extreme-scale platforms, Simulat. Modell. Pract. Theory (2015), http://dx.doi.org/10.1016/j.simpat.2015.03.005

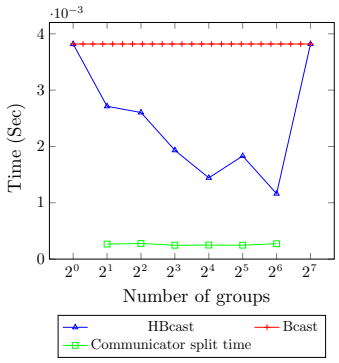

Fig. 11. Hierarchical native MPI broadcast on Grid'5000. m = 16 kB and p = 128.

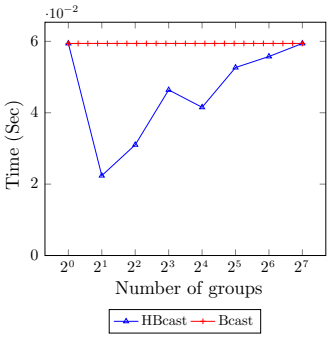

Fig. 12. Hierarchical native MPI broadcast on Grid'5000. m = 16 MB and p = 128.

4.2. Experiments on Grid'5000

The next part of the experiments was carried out on the Graphene cluster of the Nancy site of the Grid'5000 infrastructure in France. The platform consists of 20 clusters distributed over nine sites in France and one in Luxembourg. The Grid'5000 web site (http://www.grid5000.fr) provides more comprehensive information about the platform.

The experiments on Grid'5000 have been done with Open MPI 1.4.5 which provides a few broadcast implementations, such as flat, chain(linear), pipelined, binary, binomial, split-binary tree. Because of space restrictions we only present experimental study with Open MPI native broadcast operation, the chain and pipeline broadcast algorithms. During the experiments the hierarchical transformations of the binary and binomial tree algorithms had the same performance as the original algorithms. The same technique as described in MPIBlib [21] has been used to benchmark the performance.

4.2.1. Experiments on Grid'5000: One process per node

An experimental study with the Open MPI native broadcast operation is given in Figs. 11 and 12. The first measurement was performed with a message size of 16 kB where there is more than 3 times improvement. The experiment with 16 MB showed 2.6 times reduction of the broadcast time. In the experiments with smaller message sizes up to 1 kB and 128 processes the overhead from the two MPI_Comm_split operations was higher than the broadcast itself. However, with message sizes larger than 1 kB the overhead from the split operations was negligible, for example, Fig. 11 shows the split time on 128 nodes. We had the same trend with larger processes, unfortunately for a lack of space we could not include all of them in this text. Figs. 13 and 14 show the results of experiments with the chain broadcast algorithm and its hierarchical transformation for message sizes 16 kB and 16 MB respectively. The speedup with the first setting is more than 8 times and with 16 kB there is about 3 times improvement. In such situations an implementation of the algorithm could check the message size beforehand and fall back to use the regular MPI_Bcast for short messages to reduce the overhead even further.

Figs. 15 and 16 show experiments with the pipeline broadcast algorithm and its hierarchical transformation. This time the speedup can be more than 30 times with a message size of 16 kB and more than 5 times with 16 MB. Fig. 17 shows the speedup for different numbers of processes for a fixed message size of 16 MB, and Fig. 18 shows the speedup for different message sizes on 128 nodes.

4.2.2. Experiments on Grid'5000: One process per core

This section presents experiments with a one-process-per-core configuration, or equivalently four processes per node. Figs. 19 and 20 show experimental results for the chain and pipeline broadcasts on 512 cores with message sizes of

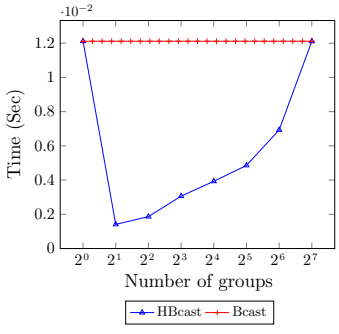

Fig. 13. Hierarchical chain broadcast on Grid'5000. m = 16 kB and p = 128.

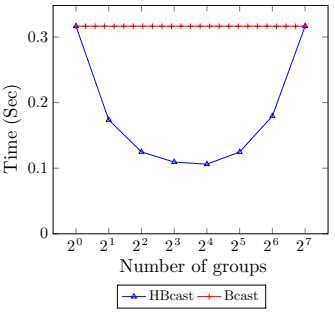

Fig. 14. Hierarchical chain broadcast on Grid'5000. m = 16 MB and p = 128.

Please cite this article in press as: K. Hasanov et al., Topology-oblivious optimization of MPI broadcast algorithms on extreme-scale platforms, Simulat. Modell. Pract. Theory (2015), http://dx.doi.org/10.1016/j.simpat.2015.03.005

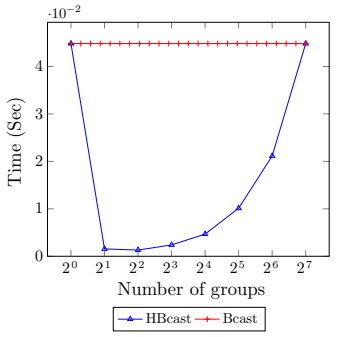

Fig. 15. Hierarchical pipeline broadcast on Grid'5000. m = 16 kB and p = 128.

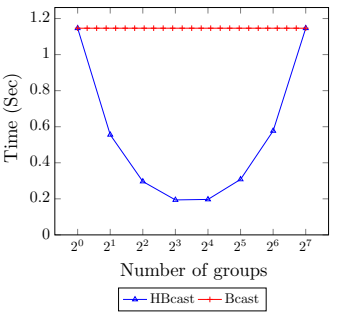

Fig. 16. Hierarchical pipeline broadcast on Grid'5000. m = 16 MB and p = 128.

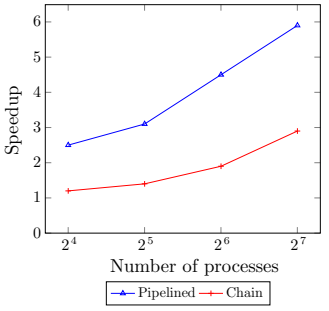

Fig. 17. Speedup of Hbcast over Bcast on Grid'5000. m = 16 MB.

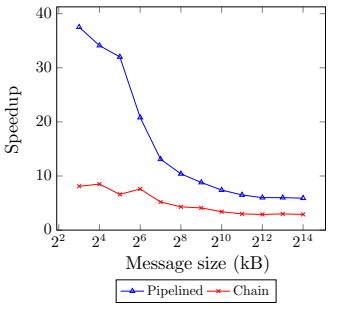

Fig. 18. Speedup of Hbcast over Bcast on Grid'5000. p = 128.

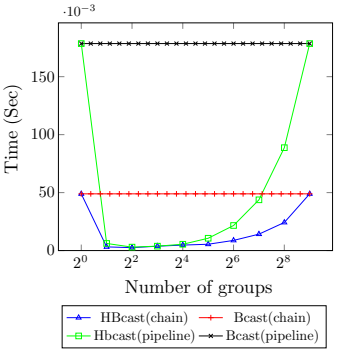

Fig. 19. Hierarchical broadcast on Grid'5000. m = 16 kB and p = 512.

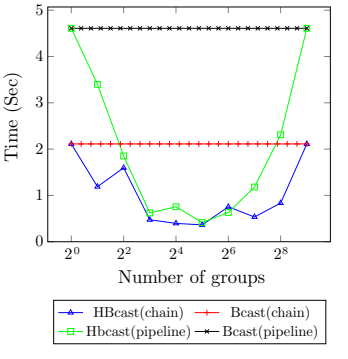

Fig. 20. Hierarchical broadcast on Grid'5000. m = 16 MB and p = 512.

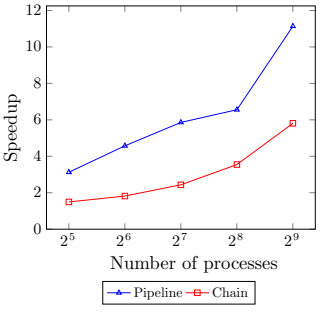

Fig. 21. Speedup of Hbcast over Bcast on Grid'5000. m = 16 MB.

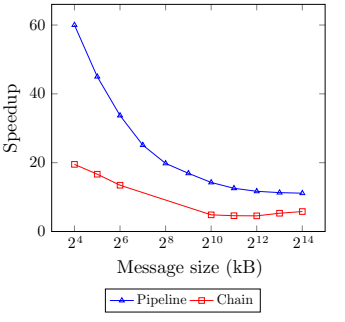

Fig. 22. Speedup of Hbcast over Bcast on Grid'5000. p = 512.

16 kB and 16 MB respectively. Fig. 22 shows the speedup of the hierarchical chain and the hierarchical pipeline broadcast algorithms for different message sizes from 16 kB up to 16 MB on 512 cores. Another experiment with a fixed message size of 16 Mb and a power-of-two number of processes varying from 32 to 512 is given on Fig. 21.

5. Conclusion

The proposed hierarchical approach to optimize MPI broadcast algorithms is more general and simpler than many existing broadcast optimizations. The method does not break up any existing broadcast algorithms, is not limited to some specific platforms and can be realized as a standalone library on top of any MPI implementations. The experiments show multifold performance improvements. This approach can be applied to other MPI collective operations as well.

This paper presents optimization results of the general MPI broadcast algorithms implemented in MPICH and Open MPI, including two most widely used algorithms, scatter-ring-allgather and pipelined algorithms. Our initial observations indicate that BlueGene/P default broadcast operation can also be optimized by the hierarchical transformation.

Acknowledgements

This work has emanated from research conducted with the financial support of IRCSET (Irish Research Council for Science, Engineering and Technology) and IBM, Grant No. EPSPG/2011/188 and Science Foundation Ireland, Grant No. 08/IN.1/I2054.

Some of the experiments presented in this publication were carried out using the Grid'5000 experimental testbed, being developed under the INRIA ALADDIN development action with support from CNRS, RENATER and several Universities as well as other funding bodies (see https://www.grid5000.fr).

Another part of the experiments were carried out using the resources of the Supercomputing Laboratory at King Abdullah University of Science&Technology (KAUST) in Thuwal, Saudi Arabia.

References

[1] Message Passing Interface Forum. <http://www.mpi-forum.org/>.
[2] S. Balay, S. Abhyankar, M.F. Adams, J. Brown, P. Brune, et al., PETSc Web Page, 2014. <http://www.mcs.anl.gov/petsc>.
[3] MPICH – A Portable Implementation of MPI. <http://www.mpich.org/>.
[4] E. Gabriel, G. Fagg, B. Bosilca, et al., Open MPI: goals, concept, and design of a next generation MPI implementation, in: Proceedings of the 11th European PVM/MPI Users Group Meeting, 2004, pp. 97–104.
[5] J. Watts, R. Van de Geijn, A pipelined broadcast for multidimensional meshes, in: Parallel Processing Letters, 1995.
[6] M. Barnett, S. Gupta, et al, Interprocessor collective communication library (InterCom), in: Proceedings of the Scalable High Performance Computing Conference, IEEE, 1994.
[7] J. Liu, A.R. Mamidala, D.K. Panda, Fast and scalable MPI-level broadcast using InfiniBand's hardware multicast support, in: Proceedings of IPDPS, 2003.
[8] T. Hoefler, C. Siebert, W. Rehm, A practically constant-time MPI broadcast algorithm for large-scale InfiniBand clusters with multicast, in: Proceedings of IPDPS, 2007.
[9] R. Graham, M.G. Venkata, et al., Cheetah: a framework for scalable hierarchical collective operations, in: Proceedings of CCGrid, 2011, pp. 73–83.
[10] S. Kumar, G. Dozsa, G. Almasi, et al., The deep computing messaging framework: generalized scalable message passing on the blue gene/P supercomputer, in: Proceedings of the 22nd Annual International Conference on Supercomputing, 2008, pp. 94–103.
[11] K. Dichev, A. Lastovetsky, Optimization of collective communication for heterogeneous HPC platforms, High-Perform. Comput. Complex Environ. (2014) 95–114.
[12] K. Kandalla, A. Venkatesh, et al., Designing optimized MPI broadcast and allreduce for many integrated core (MIC) InfiniBand clusters, in: Proceedings of HOTI, 2013, pp. 63–70.
[13] S.L. Johnsson, C.-T. Ho, Optimum broadcasting and personalized communication in hypercubes, IEEE Trans. Comput. 38 (9) (1989) 1249–1268.
[14] P. Sanders, J.F. Sibeyn, A bandwidth latency tradeoff for broadcast and reduction, Inform. Process. Lett. 86 (1) (2003) 33–38.
[15] J.L. Träff, A. Ripke, Optimal broadcast for fully connected processor-node networks, J. Parall. Distrib. Comput. 7 (68) (2008) 887–901.
[16] K. Hasanov, J.N. Quintin, A. Lastovetsky, Hierarchical approach to optimization of parallel matrix multiplication on large-scale platforms, J. Supercomput. (2014) 1–24.
[17] K. Hasanov, J.N. Quintin, A. Lastovetsky, High-level topology-oblivious optimization of MPI broadcast algorithms on extreme-scale platforms, in: Euro-Par 2014: Parallel Processing Workshops, Lecture Notes in Computer Science, vol. 8806, 2014, pp. 413–425.
[18] R.W. Hockney, The communication challenge for MPP: Intel Paragon and Meiko CS-2, Parall. Comput. 20 (3) (1994) 389–398.
[19] J. Pješivac-Grbović, Towards Automatic and Adaptive Optimizations of MPI Collective Operations, Ph.D. thesis, University of Tennessee, Knoxville, 2007.
[20] R. Thakur, R. Rabenseifner, W. Gropp, Optimization of collective communication operations in MPICH, IJHPCA 19 (1) (2005) 49–66.
[21] A. Lastovetsky, V. Rychkov, M. O'Flynn, MPIBlib: benchmarking MPI communications for parallel computing on homogeneous and heterogeneous clusters, in: Recent Advances in Parallel Virtual Machine and Message Passing Interface, Lecture Notes in Computer Science, vol. 5205, 2008, pp. 227–238.

Please cite this article in press as: K. Hasanov et al., Topology-oblivious optimization of MPI broadcast algorithms on extreme-scale platforms, Simulat. Modell. Pract. Theory (2015), http://dx.doi.org/10.1016/j.simpat.2015.03.005