High-Level Topology-Oblivious Optimization of MPI Broadcast Algorithms on Extreme-Scale Platforms

Khalid Hasanov[1], Jean-Noël Quintin[2], and Alexey Lastovetsky[1]

[1] University College Dublin
Belfield, Dublin 4, Ireland
khalid.hasanov@ucdconnecte.ie,
Alexey.Lastovetsky@ucd.ie
[2] Extreme Computing R&D, Bull SAS,
Paris, France
jean-noel.quintin@bull.net

Abstract. There has been a significant research in collective communication operations, in particular in MPI broadcast, on distributed memory platforms. Most of the research works are done to optimize the collective operations for particular architectures by taking into account either their topology or platform parameters. In this work we propose a very simple and at the same time general approach to optimize legacy MPI broadcast algorithms, which are widely used in MPICH and OpenMPI. Theoretical analysis and experimental results on IBM BlueGene/P and a cluster of Grid'5000 platform are presented.

Keywords: MPI, Broadcast, BlueGene, Grid'5000, Extreme-Scale, Communication, Hierarchy.

1 Introduction

Collective communication operations in the Message Passing Interface (MPI) [1] are very important building blocks for many scientific applications. In particular, MPI broadcast is used in a variety of algorithms and applications such as parallel matrix-matrix multiplication, LU factorization and so on. During a broadcast the root process sends a message to all other processes in the specified group of processes. The implementations of the broadcast operation in MPICH [2] and OpenMPI [3] are typically based on linear, binary, binomial and pipelined algorithms [5]. The linear algorithms are not good for a large number of processes, the binary and binomial algorithms are not efficient for large data sizes. On the other hand, pipelined algorithms try to be efficient for large 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, there has been a significant research in optimizing MPI broadcast for some specific platforms. The research work in [9] present efficient implementations of MPI broadcast with native Infiniband multicast. The Cheetah framework

L. Lopes et al. (Eds.): Euro-Par 2014 Workshops, Part II, LNCS 8806, pp. 412–424, 2014.

offers a hierarchical collective communication framework that takes advantage of hardware-specific data-access mechanisms [10]. IBM BlueGene comes with its own platform specific optimizations of MPI collectives [12]. The research work in [11] gives a comprehensive overview of optimization techniques for collectives on heterogeneous HPC platforms using broadcast as a use case.

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 [7]. Later, the research work in [8] introduced another theoretically optimal broadcast algorithm based on fractional trees. The work in [13] 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 in turn raises the cost of coordination and interaction of processes, namely, the communication cost in traditional message-passing data-parallel applications. Meanwhile, a lot of research in optimization of the communication cost of scientific algorithms and applications is going on. Very often such research works focus on specific platforms and propose a redesign of the existing scientific algorithms suitable for these platforms. In contrast to this approach, the goal of our work, which is inspired by our previous study of parallel matrix multiplication on large-scale distributed memory platforms [14], is to provide a simple and general technique to optimize the legacy scientific applications without redesigning them. In this paper, this idea is applied to the MPI broadcast operation as an initial step to achieve this goal.

The contributions of this work are as follows:

- A simple and general hierarchical technique to optimize the MPI broadcast operation, which 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 OpenMPI.

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. A process can simultaneously send and receive a message in $\alpha + m \times \beta$ time. Here α is the startup cost or latency, while β is the reciprocal bandwidth.

2.1 Previous Work

This section briefly summarizes the theoretical analysis of the performance of all the general-purpose MPI broadcast algorithms implemented in MPICH and OpenMPI. Namely, we recall the theoretical costs of linear, chain, pipelined,

binary, split-binary, scatter-ring-allgather, scatter-recursive-doubling-allgather and binomial tree broadcast algorithms. The first five and the binomial tree algorithm are implemented in OpenMPI and the last three algorithms are implemented in MPICH. Because of space limitation derivations of these algorithms are not provided in this work but can be found in [15].

- Flat tree broadcast algorithm.
  This is the simplest MPI broadcast algorithm, in which the root node sends the same message to all the nodes participating in the broadcast operation. This algorithm does not scale well for large communicators. By using the simple linear communication model its cost can be derived as follows:

$$(p-1) \times (\alpha + m \times \beta) \quad . \tag{1}$$

- 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 the flat tree algorithm:

$$(p-1) \times (\alpha + m \times \beta) \quad . \tag{2}$$

- Pipelined linear tree broadcast algorithm.
  By splitting and pipelining the message in the linear tree algorithm its performance can be improved. In this case each process can start sending a part of the message after it received the first part of the message.

$$(X + p - 2) \times \left(\alpha + \frac{m}{X} \times \beta\right) \quad . \tag{3}$$

  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.
- Binary and binomial tree broadcast algorithms.

$$\log_2(p) \times (\alpha + m \times \beta) \quad . \tag{4}$$

  Binary and binomial tree broadcast algorithms theoretically have the same cost. However, in practice, the binomial tree algorithm is more balanced than a binary tree broadcast.
- Scatter-ring-allgather broadcast algorithm.

$$(\log_2(p) + p - 1) \times \alpha + 2\frac{p-1}{p} \times m \times \beta \quad . \tag{5}$$

  This algorithm has two main phases: scatter and allgather. 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.

– Scatter-recursive-doubling-allgather broadcast algorithm.

$$2\times \log_2(p) \times\alpha + 2\frac{p-1}{p}\times m\times\beta \ . \tag{6}$$

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 for large message sizes because of its nearest-neighbor communication pattern [4].

– Split-binary tree broadcast algorithm [15].
The split-binary tree algorithm splits the original message into two segments and the segments are broadcast separately in two different binary trees. Finally, each process in both trees exchanges its message with the corresponding pair process from the other tree.

$$2\times (\log_2(p+1) - 2) \times (\alpha + m\times\beta) + \alpha + \frac{m}{2}\times\beta \ . \tag{7}$$

3 Hierarchical Optimization of MPI Broadcast Algorithms

This section introduces a simple but at the same time general optimization of the MPI broadcast algorithms. The idea was inspired by our previous study on the optimization of the communication cost of parallel matrix multiplication on large-scale distributed memory platforms [14].

The proposed optimization technique is based on the arrangement of the p processes participating in the broadcast into logical groups. For simplicity it is assumed 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. Figure 2 shows an arrangement of 12 processes in the original linear way and their hierarchical grouping 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 Figure 1 in red), and in the next step the leaders start broadcasting inside their own group (in this example among 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 happen in parallel. In general different algorithms can be used for broadcast operations between group leaders and within each group. This work focuses on the case where the same algorithm is employed for all broadcast operations. Algorithm 1 shows the pseudocode of the hierarchically modified broadcast algorithm. 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.

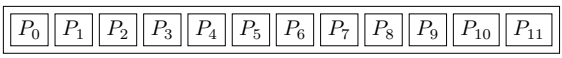

Fig. 1. Arrangement of processes in broadcast

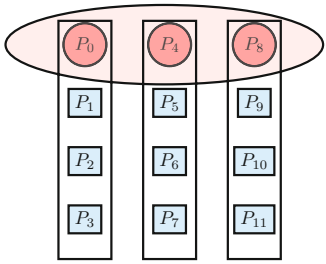

Fig. 2. Arrangement of processes in hierarchical broadcast. Processes in the ellipses are the group leaders. The rectangles show the processes inside groups. In the first step the broadcast is performed among the group leaders and in the next step it is performed among the processes inside each group.

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_outer, 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:

$$F(G) = (G-1)\times(\alpha+m\times\beta)+(\frac{p}{G}-1)\times(\alpha+m\times\beta) = (G+\frac{p}{G}-2)\times(\alpha+m\times\beta) \ . \tag{8}$$

Formula 8 is a function of G for a fixed p. Its derivative will be as follows:

$$F'(G) = \left(1-\frac{p}{G^2}\right)\times(\alpha+m\times\beta) \ . \tag{9}$$

We can see 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 formula 8 the optimal value of the broadcast will be as follows:

$$F\left(\sqrt{p}\right) = (2\sqrt{p}-2)\times(\alpha+m\times\beta) \ . \tag{10}$$

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:

$$F\left(G\right) = \left(2X+G+\frac{p}{G}-4\right)\times\left(\alpha+\frac{m}{X}\times\beta\right) \tag{11}$$

It can be shown that $G=\sqrt{p}$ is the minimum point again.

3.3 Hierarchical Binary and Binomial Tree Broadcast

Let us apply formula 4 among G groups and inside the groups among $\frac{p}{G}$ processes. The cost of either the binary or the binomial broadcast algorithm among G groups and inside the groups will be $\log_2(G)\times(\alpha+m\times\beta)$ and $\log_2(\frac{p}{G})\times(\alpha+m\times\beta)$ respectively. If we add these two costs together and consider that $\log_2\left(\frac{p}{G}\right) = \log_2\left(p\right)-\log_2\left(G\right)$ then the cost of the hierarchical binary/binomial broadcast algorithm will be the same as that of the corresponding non-hierarchical broadcast algorithm.

3.4 Hierarchical Scatter-Ring-Allgather Broadcast

If we apply formula 5 in the same way we can get the following formula:

$$F\left(G\right) = \left(log_2\left(p\right)+G+\frac{p}{G}-2\right)\times\alpha+2\times m\times\left(2-\frac{1}{G}-\frac{G}{p}\right)\times\beta \ . \tag{12}$$

Let us find the optimal value of the $F(G)$ function.

$$F'(G) = \frac{g^2 - p}{G^2} \times \left(\alpha - \frac{2m\beta}{p} \right) \ . \tag{13}$$

Formula 13 shows that if

$$\frac{\alpha}{\beta} > \frac{2m}{p} \tag{14}$$

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:

$$F(\sqrt{p}) = (log_2(p) + 2\sqrt{p} - 2) \times \alpha + 2 \times m \times (2 - \frac{2}{\sqrt{p}}) \times \beta \ . \tag{15}$$

3.5 Hierarchical Scatter-Recursive-Doubling-Allgather Broadcast

$$F(G) = 2 \times log_2(p) \times \alpha + 2 \times m \times (2 - \frac{1}{G} - \frac{G}{p}) \times \beta \ . \tag{16}$$

The hierarchical modification of this algorithm has higher theoretical cost compared to the cost of the original algorithm (formula 6): the latency term is increased two times and the bandwidth term is increased as well.

3.6 Hierarchical Split-Binary Tree Broadcast

We take $p+1 \approx p$ in formula 7 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 formula 7):

$$2 \times (\log_2(p) + X - 4) \times (\alpha + \beta \times \frac{m}{X}) + 2 \times (\alpha + \beta \times \frac{m}{2}) \ . \tag{17}$$

3.7 Summary of Theoretical Analysis

This section can be summarized as follows: the hierarchical transformations of the flat, chain, pipeline and scatter-ring-allgather algorithms theoretically reduce the communication cost of the corresponding original algorithms. The communication costs of the binary, binomial, scatter-recursive-doubling-allgather and split-binary tree algorithms either do not change or slightly increase by a constant factor after the hierarchical transformation.

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 16384 nodes. Each node is equipped with four 32-bit, 850 Mhz PowerPC 450 cores and 4GB DDR memory. The BlueGene/P (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. Use of this network is integrated into the BG/P MPI implementation. BlueGene/P MPI implementation is based on MPICH. It is known that MPI broadcast operation in MPICH uses three different broadcast algorithms depending on the message size and the number of processes in a broadcast operation [4]:

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

Despite the referenced paper 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 this work we only present experiments with the corresponding hierarchical modifications of the scatter-ring-allgather algorithm. Experiments with the binomial and scatter-recursive-doubling-allgather algorithms showed only slight fluctuations which are expected theoretically. In addition to the algorithms in MPICH, the broadcast operation on BG/P uses different communication protocols and broadcast algorithms: if the communicator is MPI_COMM_WORLD then it uses the BG/P collective tree network and otherwise depending on the communicator shape either a rectangular broadcast algorithm or MPICH are used [12]. However, MPI_COMM_WORLD is not used in computational libraries. On the other hand the rectangular broadcast is used only for rectangular shaped sub-communicators. Depending on the allocated BG/P partition and the mapping of the processes into the physical topology, sub-communicators can be arbitrary shaped. On the other hand, the proposed optimization in this work is more general and topology-oblivious.

Performance modeling and analysis of the BG/P specific broadcast algorithms and optimizations are beyond the scope of this paper. However, we also present experiments with the default BG/P broadcast operation as an initial research in that direction. The experiments have been done with different configurations, message sizes from 1kB up to 16MB and the number of MPI processes from 8 up to 5000. Here we used less number of MPI processes than the allocated BG/P nodes as we created sub-communicators to avoid the case with MPI_COMM_WORLD. Because of space restrictions we provide the results only for 2048 and 5000 processes and message sizes 512kB and 2MB. It is worth mentioning that BG/P is quite stable in terms of reproducibility if the configuration

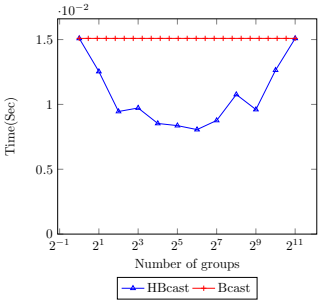

Fig. 3. Hierarchical SRGA bcast on BG/P. m=512kB and p=2048.

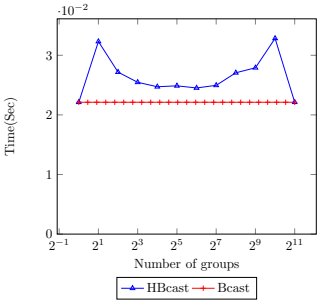

Fig. 4. Hierarchical SRGA bcast on BG/P. m=2MB and p=2048.

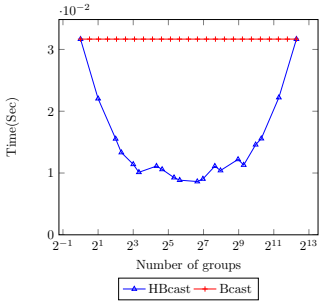

Fig. 5. Hierarchical SRGA bcast on BG/P. m=512kB and p=5000.

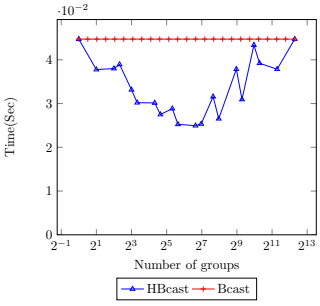

Fig. 6. Hierarchical SRGA bcast on BG/P. m=2MB and p=5000.

is kept the same. The allocated BG/P shapes were 2x1x2 and 2x3x2 in the the experiments with 2048 and 5000 processes respectively. Figure 3 and Figure 4 show experiments with the scatter-ring-allgather broadcast with message sizes of 512kB and 2MB respectively. The improvement with 512kB on 2048 nodes is 1.87 times, however with a message size of 2MB there is a performance drop. On the other hand, according to formula 14 (i.e. $\frac{\alpha}{\beta} > \frac{2m}{p}$) if we fix the message size, for a larger number of nodes the hierarchical transformation should improve the performance. This is validated with the experiments: Figure 5 shows that the performance with the message size 512kB increases up to 3.67 times on 5000 nodes. Moreover, Figure 6 shows that on 5000 nodes the hierarchical algorithm is better even with the message size of 2MB. In addition, if we put the platform and algorithm parameters in formula 12 the plots of the hierarchical algorithm will be parabola-like as well (Figure 7 and Figure 8). Figure 9 and Figure 10 show the experiments with the default BG/P MPI broadcast.

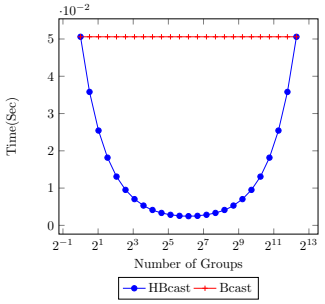

Fig. 7. Theoretical plot of SRGA bcast. m=512kB and p=5000.

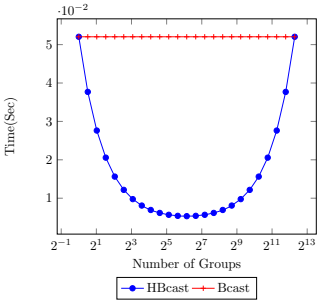

Fig. 8. Theoretical plot of SRGA bcast. m=2MB and p=5000.

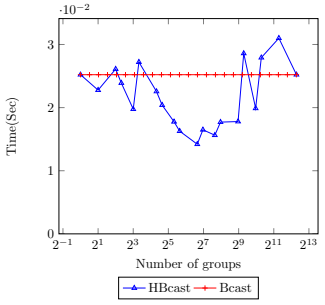

Fig. 9. Hierarchical bcast on BG/P. m=512kB and p=5000.

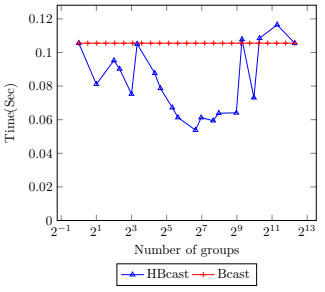

Fig. 10. Hierarchical bcast on BG/P. m=2MB and p=5000.

4.2 Experiments on Grid'5000

The next part of the experiments was carried out on the Graphene cluster of Nancy site of the Grid'5000 infrastructure in France. The platform consists of 20 clusters distributed over 9 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 OpenMPI 1.4.5 which provides a few broadcast implementations. Among those implementations there are several general broadcast algorithms such as flat, chain(linear), pipelined, binary, binomial, split-binary tree and platform/architecture specific algorithms some of which are broadcast algorithms for Infiniband networks, and the Cheetah framework for multicore architectures. In this work we do not consider the broadcast algorithms for the specific platforms. Furthermore, experiments with the binary and binomial tree broadcasts are not presented here because of space restrictions. Because of the same reason we present experiments only with 128 nodes (one process per node). We have used the same approach as presented in MPIBlib [16] to benchmark the performance.

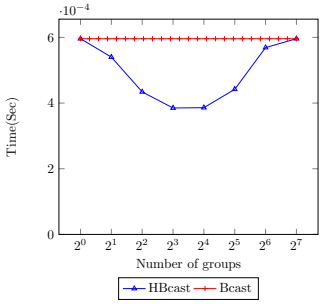

Fig. 11. Hierarchical chain broadcast on Grid'5000. m=16kB and p=128.

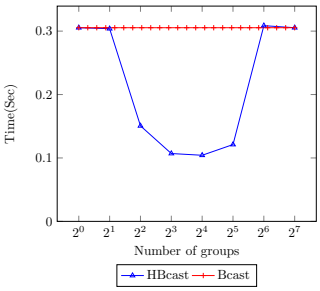

Fig. 12. Hierarchical chain broadcast on Grid'5000. m=16MB and p=128.

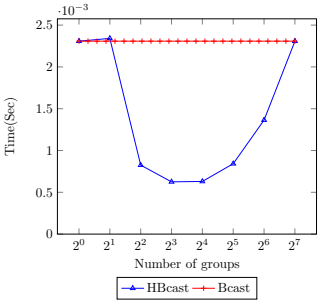

Fig. 13. Hierarchical pipeline broadcast on Grid'5000. m=16kB and p=128.

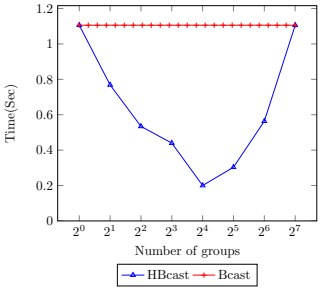

Fig. 14. Hierarchical pipeline broadcast on Grid'5000. m=16MB and p=128

Figure 11 and Figure 12 represent experiments with the chain broadcast algorithm and its hierarchical transformation with message sizes 16kB and 16MB respectively. The speedup with the message size 16MB is three times and with 16kB is 1.5 times. During the experiments with smaller message sizes up to 1kB the overhead from the two MPI_Comm_split operations were higher than the chain broadcasts itself. Still, 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. Figure 14 and Figure 13 show experiments with the pipeline broadcast algorithm and its hierarchical transformation. The trend is similar to the chain algorithm. This time the improvement is even higher, 5.5 times with the message size 16MB and 3.69 times with the message size 16kB.

5 Conclusion

Our hierarchical approach to optimize MPI broadcast algorithms is more general and simpler than many existing broadcast optimizations. The idea itself

does not break up any existing broadcast algorithms, is not limited to some specific platforms and can be used as a standalone library on top of any MPI implementations. Some broadcast algorithms have been improved more than five times even on a relatively small number of processors.

This work presents the application of the proposed technique to general MPI broadcast algorithms implemented in MPICH and OpenMPI. Among these algorithms there are the two most used algorithms: scatter-ring-allgather and pipelined algorithms. Our initial observation showed that BlueGene/P default broadcast operation can be optimized by the hierarchical transformation as well. Therefore, one of our future plans is to study the hierarchical modifications of the broadcast algorithms optimized for IBM BlueGene/P and Infiniband networks. A similar kind of approach can also be applied to other MPI collective operations.

We are working on a software library/tool which can be incorporated into any application which uses MPI broadcast. The software will let users easily transform any broadcast algorithm into a two-level hierarchy and predict their performance.

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 number EPSG/2011/188 and Science Foundation Ireland, grant number 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. MPICH-A Portable Implementation of MPI, `http://www.mpich.org/`
3. 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., Woodall, T.: Open MPI: goals, concept, and design of a next generation MPI implementation. In: Proceedings of the 11th European PVM/MPI Users' Group Meeting, pp. 97–104 (2004)
4. Thakur, R., Gropp, W.D.: Improving the Performance of Collective Operations in MPICH. In: Dongarra, J., Laforenza, D., Orlando, S. (eds.) EuroPVM/MPI 2003. LNCS, vol. 2840, pp. 257–267. Springer, Heidelberg (2003)
5. Watts, J., Van de Geijn, R.: A Pipelined Broadcast for Multidimensional Meshes Parallel Processing Letters 05, 281 (1995)
6. Barnett, M., Gupta, S., Payne, D., Shuler, L., Van de Geijn, R., Watts, J.: Interprocessor collective communication library (InterCom). In: Proceedings of the Scalable High Performance Computing Conference, pp. 357–364. IEEE (1994)

7. Johnsson, S.L., Ho, C.-T.: Optimum Broadcasting and Personalized Communication in Hypercubes. IEEE Transactions on Computers 38(9), 1249–1268 (1989)
8. Sanders, P., Sibeyn, J.F.: A bandwidth latency tradeoff for broadcast and reduction. Information Processing Letters 86(1), 33–38 (2003)
9. Hoefler, T., Siebert, C., Rehm, W.: A practically constant-time MPI Broadcast Algorithm for large-scale InfiniBand Clusters with Multicast. In: Proceedings of the 21st IEEE International Parallel and Distributed Processing Symposium, vol. 232 (March 2007)
10. Graham, R., Venkata, M.G., Ladd, J., Shamis, P., Rabinovitz, I., Filipov, V., Shainer, G.: Cheetah: a framework for scalable hierarchical collective operations. In: Proceedings of CCGrid, pp. 73–83 (2011)
11. Dichev, K., Lastovetsky, A.: Optimization of collective communication for heterogeneous HPC platforms. In: High-Performance Computing on Complex Environments, Wiley, pp. 95–114. Wiley (2014)
12. Kumar, S., Dozsa, G., Almasi, G., Heidelberger, P., Chen, D., Giampapa, M.E., Blocksome, M., Faraj, A., Parker, J., Ratterman, J., Smith, B., Archer, C.J.: 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 (ICS), pp. 94–103 (2008)
13. Träff, J.L., Ripke, A.: Optimal Broadcast for Fully Connected Processor-node Networks. Journal of Parallel Distributed Computing 7(68), 887–901 (2008)
14. Hasanov, K., Quintin, J., Lastovetsky, A.: Hierarchical Approach to Optimization of Parallel Matrix Multiplication on Large-Scale Platforms. The Journal of Supercomputing 24 (2014)
15. Pješivac-Grbović, J.: Towards Automatic and Adaptive Optimizations of MPI Collective Operations. PhD Thesis, University of Tennessee, Knoxville (2007)
16. Lastovetsky, A., Rychkov, V., O'Flynn, M.: MPIBlib: Benchmarking MPI communications for parallel computing on homogeneous and heterogeneous clusters. In: Lastovetsky, A., Kechadi, T., Dongarra, J. (eds.) EuroPVM/MPI 2008. LNCS, vol. 5205, pp. 227–238. Springer, Heidelberg (2008)