Optimization of Collective Communications in HeteroMPI

Alexey Lastovetsky, Maureen O'Flynn, and Vladimir Rychkov

School of Computer Science and Informatics, University College Dublin,
Belfield, Dublin 4, Ireland
{Alexey.Lastovetsky,Maureen.OFlynn}@ucd.ie
http://hcl.ucd.ie

Abstract. HeteroMPI is an extension of MPI designed for high performance computing on heterogeneous networks of computers. The recent new feature of HeteroMPI is the optimized version of collective communications. The optimization is based on a novel performance communication model of switch-based computational clusters. In particular, the model reflects significant non-deterministic and non-linear escalations of the execution time of many-to-one collective operations for medium-sized messages. The paper outlines this communication model and describes how HeteroMPI uses this model to optimize one-to-many (scatter-like) and many-to-one (gather-like) communications. We also demonstrate that HeteroMPI collective communications outperform their native counterparts for various MPI implementations and cluster platforms.

Keywords: MPI, HeteroMPI, heterogeneous cluster, switched network, message passing, collective communications, scatter, gather.

1 Introduction

MPI [1] is the most widely used programming tool for parallel computing on distributed-memory computer systems. It can be used on both homogeneous and heterogeneous clusters, but it does not provide specific support for development of high performance parallel applications for heterogeneous networks of computers (HNOC).

HeteroMPI [2] is an extension of MPI designed for high performance computing on HNOCs. It supports optimal distribution of computations among the processors of a HNOC by taking into account heterogeneity of processors, network topology and computational costs of algorithm. The main idea of HeteroMPI is to automate the creation of a group of processes that will execute the heterogeneous algorithm faster than any other group. It is achieved by specifying the performance model of the parallel algorithm and by optimal mapping of the algorithm onto the HNOC, which is seen by the HeteroMPI programming system as a multilevel hierarchy of interconnected sets of heterogeneous multiprocessors. HeteroMPI is implemented on top of MPI, therefore it can work on top of any MPI implementation. HeteroMPI introduces a small number of

F. Cappello et al. (Eds.): EuroPVM/MPI 2007, LNCS 4757, pp. 135–143, 2007.

additional functions for group management and data partitioning. All standard MPI operations are inherited, so that the existing MPI programs can be easily transformed into HeteroMPI.

HeteroMPI inherits all MPI communication operations and solely relies on their native implementation. At the same time, our recent research on the performance of MPI collective communications on heterogeneous clusters based on switched networks [3] shows that MPI implementations of scattering and gathering are often very far from optimal. In particular, we observed very significant escalations of the execution time of many-to-one MPI communications for medium-sized messages. The escalations are non-deterministic but form regular levels. We also observed a leap in the execution time of one-to-many communications for large messages. Based on the observations, we suggested a communication performance model of one-to-many and many-to-one MPI operations reflecting these phenomena [3], which is applicable to both heterogeneous and homogeneous clusters. The paper presents a new feature of HeteroMPI, which is the optimized version of collective communication operations. The design of these optimized operations is based on the new communication model and can be summarized as follows:

- Upon installation, HeteroMPI computes the parameters of the model.
- Each optimized collective operation is implemented by a sequence of calls to native MPI operations. The code uses parameters of the communication model.

This high-level model-based approach to optimization of MPI communications is easily and uniformly applied to any combination of MPI implementation and cluster platform. It does not need to retreat to the lower layers of the communication stack and tweak them in order to improve the performance of MPI-based communication operations. This is particularly important for heterogeneous platforms where the users typically have neither authority nor knowledge for making changes in hardware or basic software settings.

The paper is structured as follows. Section 2 outlines the related work. Section 3 briefly introduces the communication model. Section 4 describes the implementation of the optimized collective operations in HeteroMPI. Section 5 presents experimental results, demonstrating that HeteroMPI collective communications outperform their native counterparts for different MPI implementations and clusters platforms.

2 Related Work

Vadhiyar et al. [4] developed automatically tuned collective communication algorithms. They measured the performance of different algorithms of collective communications for different message sizes and numbers of processes and then used the best algorithm. Thakur et al. [5] used a simple linear cost model of a point-to-point single communication in selection of algorithms for a particular collective communication operation. Pjesivac-Grbovic et al. [6] applied different point-to-point models to the algorithms of collective operations, compared

the predictions with measurements and implemented the optimized versions of collective operations based on the decision functions that switch between different algorithms, topologies, and message segment sizes. Kielmann et al. [7] optimized MPI collective communication for clustered wide-area environments by minimizing communication over slow wide-area links. There were some works on improving particular MPI operations [8,9].

All works on the optimization of collective operations are based on deterministic linear communication models. Implementation of the optimized versions of collective operations in HeteroMPI uses the performance model that takes into account non-deterministic escalations of the execution time of many-to-one MPI communications for medium-sized messages and the leap in the execution time of one-to-many communications for large messages.

3 The Performance Model of MPI Communications

This section briefly introduces the new performance model of MPI communications [3], which reflects the phenomena observed for collective communications on clusters based on switched networks. The basis of the model is a LogP-like [11] model of point-to-point communications for heterogeneous clusters [10]. The parameters of the point-to-point model represent the heterogeneity of processors and are also used in construction of the models of collective communications. Apart from the point-to-point parameters, the models of collective communications use parameters reflecting the observed non-deterministic and non-linear behavior of MPI collective operations.

Like any other ***point-to-point communication model***, except for PLogP our model is linear, representing the communication time by a linear function of the message size. The execution time of sending a message of M bytes from processor i to processor j on heterogeneous cluster is estimated by $C_i + Mt_i + C_j + Mt_j + \frac{M}{\beta_{ij}}$, where C_i, C_j are the fixed processing delays; t_i, t_j are the delays of processing of a byte; β_{ij} is the transmission rate. Different parameters for nodal delays reflect heterogeneity of the processors. For networks with a single switch, it is realistic to assume $\beta_{ij} = \beta_{ji}$.

There are two components in the models of one-to-many and many-to-one communications. The first one is built upon the model of point-to-point communications by representing each collective communication, MPI_Scatter or MPI_Gather, by a straightforward combination of point-to-point communications

```
if (rank==root) {
  memcpy(recvbuf, sendbuf, recvcount);
  for (i=1; i<n; i++) {
    if (scatter)
      MPI_Isend(sendbuf+sendcount*i, sendcount, i);
    if (gather)
      MPI_Irecv(recvbuf+recvcount*i, recvcount, i);
  }
```

```
  MPI_Waitall(n-1);
}
else {
  if (scatter)
    MPI_Recv(recvbuf, recvcount, root);
  if (gather)
    MPI_Send(sendbuf, sendcount, root);
}
```

This approach obviously results in a linear predictive model, which is quite accurate for relatively small message sizes but does not reflect the observed phenomena of non-deterministic and non-linear escalations of the execution time of many-to-one communications for medium-sized messages and the significant leap in the execution time of one-to-many communications for large messages. The second component in the collective communication models addresses the issues.

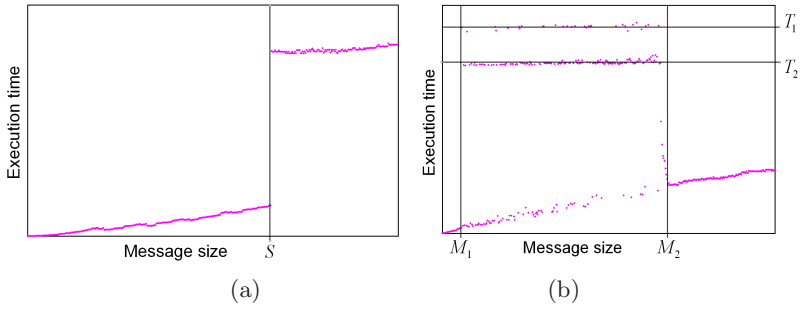

Fig. 1. The execution time of collective communications against the message size: (a) one-to-many and (b) many-to-one

Fig. 1 shows the typical behavior of one-to-many and many-to-one communications on a switch-based cluster, given the operations are implemented via the described straightforward combination of point-to-point communications. One can see a distinctive leap in the execution time for the one-to-many operation as well as non-deterministic and non-linear escalations of the execution time of the many-to-one operation for medium-sized messages. These phenomena were observed on the MPI implementations over TCP communication layer but not over Myrinet-MX. So, they may be caused by some TCP features. At the same time, we have not had a chance to experiment with a reasonably large Myrinet-based cluster and, therefore, cannot guarantee that MPI over Myrinet-MX does not have such irregularities.

The ***one-to-many model*** [10,3] reflects the leap in the execution time and categorizes the small and large messages. Parameter S is a message size threshold, separating small and large messages. It is different for different combinations of clusters and MPI implementations. The estimated time of scattering

messages of size M from node 0 to nodes $1, 2, ..., n$ is given by $C_0 + t_0 \times n \times M + \max_{1 \le i \le n} \left\{ C_i + t_i M + \frac{M}{\beta_{0i}} \right\}$, if $M < S$ or $C_0 + t_0 \times n \times M + \sum_{i=1}^{n} (C_i + t_i M + \frac{M}{\beta_{0i}})$, if $M \ge S$, where C_0, t_0, C_i, t_i are the fixed and variable processing delays on the source node and destinations. The one-to-many model displays parallel communication for small messages and a serialized communication for large messages.

The ***many-to-one model*** [3] differentiates small, medium and large messages by introducing parameters M_1 and M_2. For small messages, $M < M_1$, the execution time has a linear response to the increase of message size. Thus, the execution time for the many-to-one communication involving n processors ($n \le N$, where N is the cluster size) is estimated by $n(C_0 + t_o M) + \max_{1 < i \le n} \left\{ C_i + t_i M + \frac{M}{\beta_{0i}} \right\} + \kappa_1 M$, where κ_1 is a fitting parameter for correction of the slope. For large messages, $M > M_2$, the execution time resumes a linear predictability for increasing message size. Hence, this part of the model has the same design but a different slope of linearity and greater value due to overheads: $n(C_0 + t_0 M) + \sum_{i=1}^{n} (C_i + t_i M + \frac{M}{\beta_{0i}}) + \kappa_2 M$. The additional parameter κ_2 is a fitting constant for correction of the slope. For medium messages, $M_1 \le M \le M_2$, we observed a small number of discrete levels of escalation, remaining constant as the message size increases. The model describes the probability of escalation to each of the levels as a function of message size and the number of processors involved in the operation. If no escalation occurs, the linear model used for small massages will accurately predict the execution time.

The presented model accurately describes the performance of many-to-one and one-to-many operations for all combinations of MPI implementations and cluster platforms, which we used in our experiments, if the collective operations were implemented via point-to-point MPI operations (as described by the pseudo-code above). For LAM [12] and Open MPI [13], MPI_Scatter and MPI_Gather display exactly the same performance pattern as their straightforwardly implemented counterparts. Therefore, such MPI implementations need no further extension of the communication model in order to describe native collective communication operations. At the same time, for some MPI implementations (mainly, some versions of MPICH [14]), the native collective communications perform differently (better or worse) than their straightforward counterparts. To deal with such MPI implementations, HeteroMPI uses an extended communication model, additionally including a separate model for each native collective operation. Due to space limitations, we do not include in the paper considerations related to this extended model.

In implementation of the optimized scatter and gather collective operations, we use the message size thresholds S, M_1 and M_2 to fragment the messages and to avoid the message sizes for which the irregularities are observed. These parameters are found experimentally for a parallel platform. The building of the analytical part of the communication model is out of the scope of this paper. We do not describe the communication experiments and the measurement techniques required to find the rest of parameters.

4 Optimization of Collective Operations in HeteroMPI

This section describes the implementation of two newly introduced HeteroMPI operations, HMPI_Scatter and HMPI_Gather, which are optimized versions of native MPI_Scatter and MPI_Gather respectively. The implementation uses the communication performance model presented in Section 3 in order to avoid the MPI_Gather time escalations and the MPI_Scatter leap in the execution time. More precisely, only the message size thresholds S, M_1 and M_2 are used in the implementation. These parameters are computed by the HeteroMPI programming system upon its installation on the parallel platform. In the implementation, neither point-to-point nor low-level communications are used, but only the native MPI counterparts.

The implementation of HMPI_Gather re-invokes the native MPI_Gather for small and large messages. The gathering of medium-sized messages, $M_1 \leq M \leq M_2$, is implemented by an equivalent sequence of m MPI_Gather operations with messages of the size that fits into the range of small messages: $\frac{M}{m} < M_1$ and $\frac{M}{m-1} \geq M_1$. Small-sized gatherings are synchronized by barriers, which removes communication congestions on the receiving node. The barriers are marked bold in the pseudo code:

```
if (M1<=M<=M2) {
  find m such that M/m<M1 and M/(m-1)>=M1;
  for (i=0; i<m; i++) {
    MPI_Barrier(comm);
    MPI_Gather(sendbuf + i*M/m, M/m);
  }
}
else MPI_Gather(sendbuf, M);
```

Note. If MPI_Barrier is removed from the code, the resulting implementation will behave exactly as the native MPI_Gather. It means that this synchronization is essential for preventing communication congestions on the receiving side.

The implementation of HMPI_Scatter uses parameter S of one-to-many communication model. For small messages, $M < S$, the native MPI_Scatter is re-invoked. The scattering of large messages is implemented by an equivalent sequence of MPI_Scatter operations with messages of the size less then S: $\frac{M}{m} < S$ and $\frac{M}{m-1} \geq S$. The pseudo code of the optimized scatter is as follows:

```
if (M>=S) {
  find m such that M/m<S and M/(m-1)>=S;
  for (i=0; i<m; i++)
    MPI_Scatter(recvbuf + i*M/m, M/m);
}
else MPI_Scatter(recvbuf, M);
```

As the presented approach does not use the communication parameters reflecting the heterogeneity of the processors, it can be applied to both homogeneous and heterogeneous switch-based clusters.

5 Experiments

To compare the performance of the optimized HeteroMPI collective operations with their native MPI counterparts, we experimented with various MPI implementations and different clusters. Here we present the results for the following two platforms:

- **LAM-Ethernet:** 11 x Xeon 2.8/3.4/3.6, 2 x P4 3.2/3.4, 1 x Celeron 2.9, 2 x AMD Opteron 1.8, Gigabit Ethernet, LAM 7.1.3,
- **OpenMPI-Myrinet:** 64 x Intel EM64T, Myrinet, Open MPI 1.2.2 over TCP.

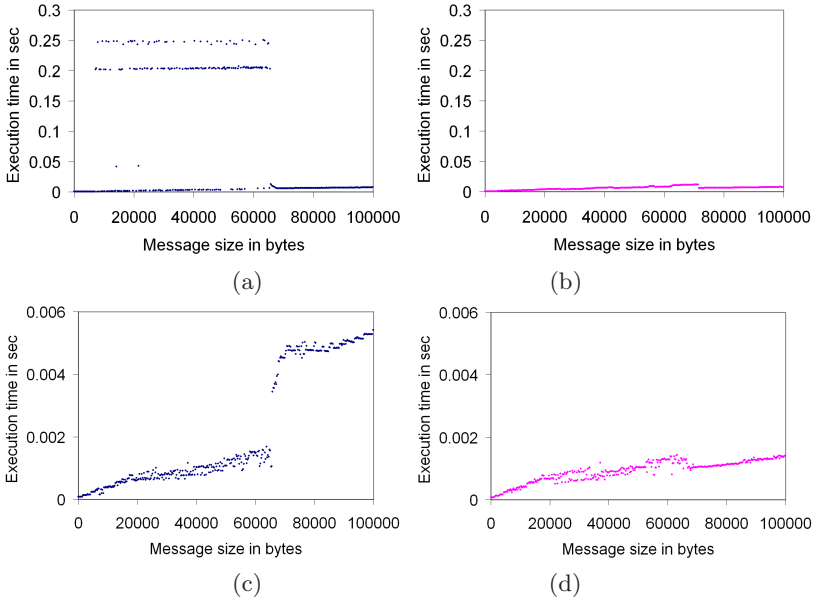

Fig. 2. Performance of (a) MPI_Gather, (b) HMPI_Gather, (c) MPI_Scatter, (d) HMPI_Scatter on 16-nodes heterogeneous cluster LAM-Ethernet

Fig. 2 shows the results for the heterogeneous LAM-Ethernet cluster, with all nodes in use. The message size thresholds for this platform are $M_1 = 5KB$, $M_2 = 64KB$, $S = 64KB$. Similar results are obtained on the 64-node homogeneous OpenMPI-Myrinet cluster (Fig. 3). For this platform, $M_1 = 5KB$, $M_2 = 64KB$, $S = 64KB$. The results show that the optimized HeteroMPI versions outperform their native MPI counterparts, avoiding the escalations and

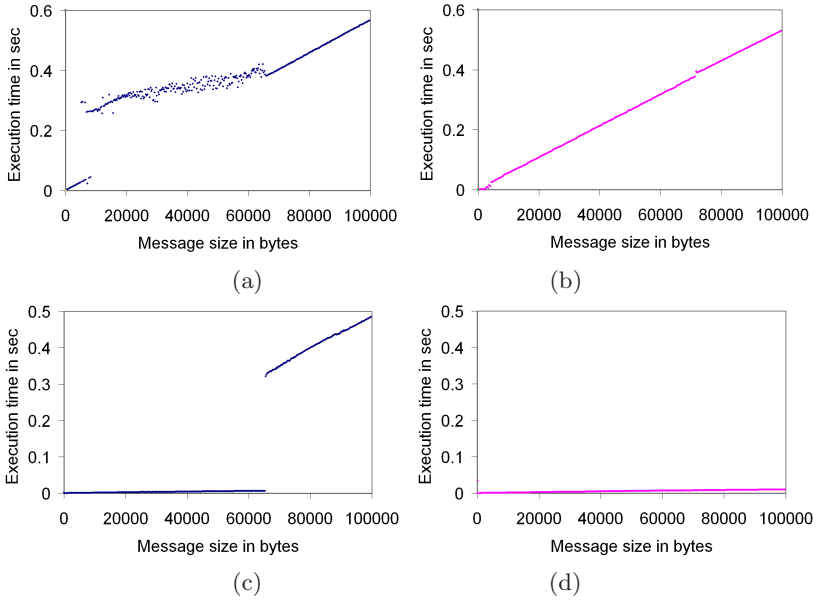

Fig. 3. Performance of (a) MPI_Gather, (b) HMPI_Gather, (c) MPI_Scatter, (d) HMPI_Scatter on 64-nodes OpenMPI-Myrinet cluster

restoring the linear dependency of the communication execution time on message size. On all platforms we observed $S = M_2$.

The communication execution time was measured on the root node. The barrier was used to ensure that all processes have finished the scatter-like operations. The communication experiments for each message size in a series were carried out only once. The repeated measurements gave similar results. To avoid the pipeline effect in a series of the experiments for different message sizes, the barriers were included between collective operations.

6 Conclusion

The paper introduced a new feature of HeteroMPI, which is the optimized versions of MPI collective communications for switched-based computational clusters. The optimized collective operations were implemented on top of the corresponding MPI functions and based on the communication performance model. We also presented experimental results demonstrating that the optimized functions outperformed the native ones.

The proposed approach to optimization of MPI communications is based on the use of a high-level communication performance model. Therefore, it can be easily and uniformly applied to any combination of MPI implementation and cluster platform. It needs no retreat to the lower layers of the communication stack for tweaking them in order to improve the performance of MPI-based

communication operations. This is particularly advantageous for heterogeneous platforms where the users typically have neither the authority nor the knowledge for changing hardware and basic software settings.

Acknowledgments. The work was supported by the Science Foundation Ireland (SFI). We are grateful to the Innovative Computing Laboratory, University of Tennessee, for providing with computing clusters.

References

1. Dongarra, J., Huss-Lederman, S., Otto, S., Snir, M., Walker, D.: MPI: The Complete Reference. The MIT Press, Cambridge (1996)
2. Lastovetsky, A., Reddy, R.: HeteroMPI: Towards a message-passing library for heterogeneous networks of computers. J. of Parallel and Distr. Comp. 66, 197–220 (2006)
3. Lastovetsky, A., O'Flynn, M.: A Performance Model of Many-to-One Collective Communications for Parallel Computing. In: Proc. of IPDPS 2007, Long Beach, CA (2007)
4. Vadhiyar, S.S., Fagg, G.E., Dongarra, J.: Automatically tuned collective communications. In: Proc. of Supercomputing 99, Portland, OR (1999)
5. Thakur, R., Rabenseifner, R., Gropp, W.: Optimization of Collective Communication Operations in MPICH. Int. J. of High Perf. Comp. App. 19, 49–66 (2005)
6. Pjesivac-Grbovic, J., Angskun, T., Bosilca, G., Fagg, G.E., Gabriel, E., Dongarra, J.J.: Performance Analysis of MPI Collective Operations. In: Proc. of IPDPS 2005, Denver, CO (2005)
7. Kielmann, T., Hofman, R.F.H., Bal, H., Plaat, A., Bhoedjang, R.A.F.: MagPIe: MPI's collective communication operations for clustered wide area systems. In: Proc. of PPoPP 1999, pp. 131–140. ACM Press, New York (1999)
8. Iannello, G.: Efficient algorithms for the reduce-scatter operation in LogGP. IEEE Transactions on Parallel and Distr. Systems 8(9), 970–982 (1997)
9. Benson, G.D., Chu, C-W., Huang, Q., Caglar, S.G.: A comparison of MPICH allgather algorithms on switched networks. In: Dongarra, J.J., Laforenza, D., Orlando, S. (eds.) Recent Advances in Parallel Virtual Machine and Message Passing Interface. LNCS, vol. 2840, pp. 335–343. Springer, Heidelberg (2003)
10. Lastovetsky, A., Mkwawa, I., O'Flynn, M.: An Accurate Communication Model of a Heterogeneous Cluster Based on a Switch-Enabled Ethernet Network. In: Proc. of ICPADS 2006, Minneapolis, MN, pp. 15–20 (2006)
11. Culler, D., Karp, R., Patterson, R., Sahay, A., Schauser, K.E., Santos, R.S.E., von Eicken, T.: LogP: Towards a realistic model of parallel computation. In: Proc. of the 4th ACM SIGPLAN, ACM Press, New York (1993)
12. LAM/MPI User's Guide, `http://www.lam-mpi.org/`
13. Open MPI Publications `http://www.open-mpi.org/`
14. MPICH/MPICH-2 User's Guide, `http://www-unix.mcs.anl.gov/mpi/`