Evaluating MPI Collective Communication on the SP2, T3D, and Paragon Multicomputers

Kai Hwang, Choming Wang, and Cho-Li Wang
The University of Hong Kong, Pokfulam, Hong Kong

Abstract

We evaluate the architectural support of collective communication operations on the IBM SP2, Cray T3D, and Intel Paragon. The MPI performance data are obtained from the STAP benchmark experiments jointly performed at the USC and HKU. The T3D demonstrated clearly the best timing performance in almost all collective operations. This is attributed to the special hardware built in the T3D for fast messaging and block data transfer. With hardwired barriers, the T3D performs the barrier synchronization in 3 μs, at least 30 times faster than the SP2 or Paragon. The startup latency of collective operations increases either linearly or logarithmically in three multicomputers.

For short messages, the SP2 outperforms the Paragon in the barrier, total exchange, scatter, and gather operations. Various collective operations with 64 KBytes per message over 64 nodes of the three machines can be completed in the time range (5.12 ms, 675 ms). The Paragon outperforms the SP2 in almost all collective operations with long messages. We have derived closed-form expressions to quantify the collective messaging times and aggregated bandwidth on all three machines. For total exchange with 64 nodes, the T3D, Paragon, and SP2 achieved an aggregated bandwidth of 1.745, 0.879, and 0. 818 GBytes/s, respectively. These findings are useful to those who wish to predict the MPP performance or to optimize parallel applications by trade-offs between divided computation and collective communication.

Keywords: Collective communications, multicomputers, message passing, startup latency, aggregated bandwidth.

This research was supported in part by MIT Lincoln Laboratory and in part by the University of Hong Kong. The authors can be reached by Email: kaihwang@cs.hku.hk, choming@diana.usc.edu, clwang@cs.hku.hk

1: Introduction

Message Passing Interface (MPI) has become a commonly accepted communication standard for specifying message-passing functions in programming multicomputers or clusters of workstations [23]. A collective messaging operation involves a group of software processes, residing in the same or different nodes, to call the same collective communication routine, with matching arguments. Typical collective operations are *broadcast, gather, scatter, total exchange, barrier, reduce, scan (prefix),* etc. These operations provide a common interface for users to design their application codes.

In the past, benchmark results of MPI were mainly focused on point-to-point communications. Only limited amounts of timing data of collective operations were reported for workstation clusters [26, 29], SP2 [10, 31], Paragon [22], Convex SPP [8], and T3D [6]. Some benchmark suites [11, 14] have also report MPI performance results but lack of comparison among different machines.

In this paper, we evaluate the architectural support of collective communication operations on three multicomputers; namely the IBM SP2 at MHPCC [21], the T3D at Cray's Eagan Center [1], and the Intel Paragon at SDSC [28]. We measured the elapsed *collective messaging time*, $T(m, p)$, with various combinations of the *machine size*, p, and the *message length*, m. The machine size refers the number of nodes involved in a collective communication operation. The message size is measured by the number of bytes per message between a pair of nodes.

The collective messaging time consists of two components: the *startup latency*, $T_0(p)$ and the *transmission delay*, $D(m, p)$. The startup latency T_0 is a function of p, which captures the software overhead in establishing a collective operation over p processing nodes. The transmission delay D covers all the time needed for the message signals to flow through the hardware network and memory hierarchy. Therefore, D depends on both the message length m and the machine size p. We shall report the measured values of these terms in subsequent sections.

These two parameters are often used by application developers to estimate the communication overhead and to reason about the optimization or parallelization strategies. To determine the ultimate performance of a collective operation on a given network, the *aggregated bandwidth*, $R_\infty(p)$ in MByte/s is defined as the total number of bytes communicated over all nodes during the data transmission period of a collective operation, when the message size approaches the infinity. This metric reflects the saturated condition or the maximum capability of a data communication network.

The reminder of the paper is organized as follows: Section 2 provides the background on MPI collective operations and their testing conditions on three target machines. Section 3 formulates the performance model used. Section 4 shows the latency results and discuss their performance attributes. Section 5 presents the effects of message length on the collective performance. Section 6 discusses the effects of machine sizes. Results on collective messaging timing are presented in Section 7. Derived in Section 8 are the timing and aggregated bandwidth expressions obtained. Finally, we summarize the contributions of this work and offer suggestions for further work to improve MPI collective operations on message-passing multicomputers.

2: MPI Operations and Testing Conditions

Several implementations of MPI are available in the public domain, such as the CHIMP/MPI [2], LAM [24], mpi++ [17], MPICH [20]. Many MPP venders have modified from these MPI packages to have their own versions, optimized with respect to their own machine architecture. For examples, the CRI/EPCC MPI [6] is available on the T3D. The MPICH is currently ported on SP-2 and Paragon. These MPI implementations offer a rich set of collective operations as summarized in Table 1. In our experiments, the data structures used are always made small enough to fit in each node memory to avoid extensive page faults. The test program is written in standard C and the MPI primitives. No machine-specific library functions nor any assembly codes are used. The best compiler option to each given machine is always applied. Unless otherwise noted, the test programs were compiled with cc -O. The system resource is used in dedicated mode to avoid interference from other user processes.

We measure the wall clock time using `MPI_Wtime()` function. The test program is executed repeatedly for more than 22 times, with timing starting on the third iteration to exclude the warm-up effect. The minimal time, the maximal time, and the mean time from all processes are collected. To interpret the results, we focus on the maximal time, because we feel it reflects the condition that all processes involved in the machine have finished the operation.

Table 1. MPI Collective Operations Being Evaluated on Three Multicomputers

Operations	Function description
MPI_Bcast	Broadcasts a message from root process to all other processes in the same group.
MPI_Barrier	Blocks until all process have reached this routine.
MPI_Alltoall	Sends data from all to all processes.
MPI_Gather	Gathers together values from a group of processes.
MPI_Reduce	Reduces values on all processes to a single value.
MPI_Scan	Computes the prefix (partial reductions) of data on a collection of processes.
MPI_Scatter	Sends data from one task to all other tasks in a group.

In all operations, single-precision (4-Byte) floating-point numbers are used. In MPI, the data type of the message elements is always MPI_FLOAT. In our testing runs, we always assign only one process to each single node. The number of nodes (processes) used ranges from 2, 4,... ., to 128 nodes. The message length m varies from 4, 16, . .., to 64 K Bytes. We were allocated with at most 64 T3D nodes from the Eagan Center. However, we were able to use up to 128 nodes from the SP2 in MHPCC and the Paragon at SDSC. Therefore, only data points up to 64 nodes are reported in subsequent sections for the T3D, and up to 128 nodes for the SP2 and Paragon.

One technical difficulty is that the allocated nodes are often not time synchronized, each having its own clock. In our experiment, the time delay of a collective communication operation is measured by the following procedure:

```
barrier synchronization
the_collective_routine_being_measured
get start_time
for (i=0; i < k; i++)
{
    the_collective_routine_being_measured
}
get end_time
local_time=(end_time-start_time)/k
communication_time=
    maximum reduce(local_time)
```

It is common that the first several runs of a test program take considerably longer time (sometimes 10 times higher) than the remaining execution runs. This is due to the warm-up effect, such as loading the routine and data into memory and cache, and initializing various buffers, etc. Therefore, results from the first two l run are ignored.

Each node process executes a barrier. After the barrier, the collective operation is executed k times by all p processes involved. The average time of the k executions

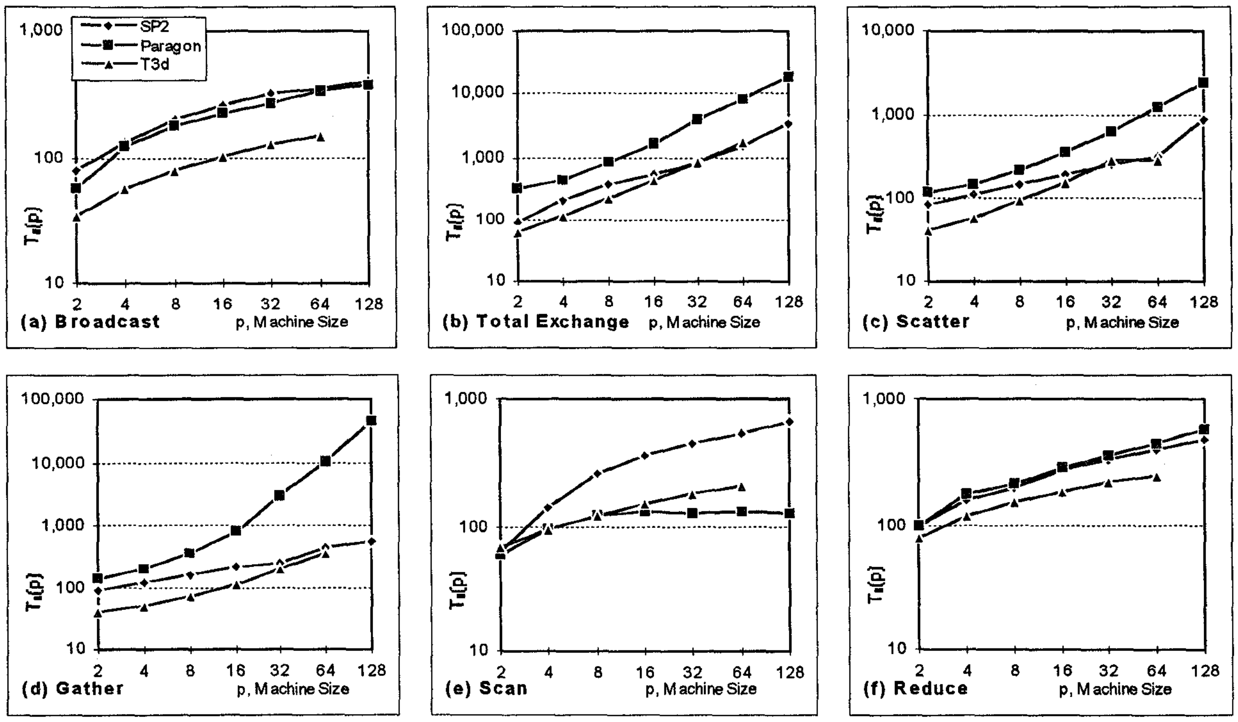


FIGURE 1. Startup latencies, $T_0(p)$, of six MPI collective operations over three multicomputers with 2 to 128 nodes.

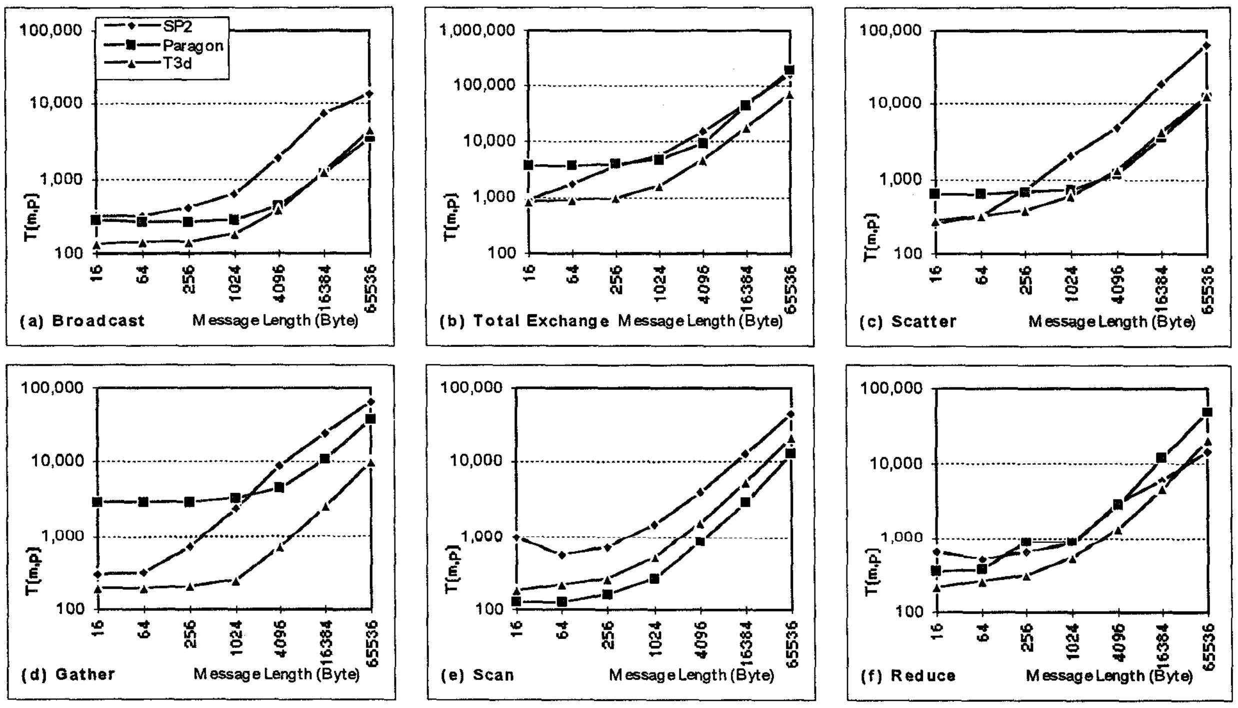


FIGURE 2. Collective messaging times, $T(m,32)$, of six MPI collective operations as a function of the message length.

is calculated on each process. The time for a collective operation is obtained through a reduction as the maximum of all p average timing values, one from each process. The test program is executed five times for each machine size p, with the value of k fixed at 20. A barrier only synchronizes the processes logically. It does not time-synchronize the processes. Thus the processes do not necessarily start to execute the *for* loop at the same time.

3: Collective Performance Metrics

To quantify the communication time of MPI collective operations, we present a model which is generalized from the model by Xu and Hwang [31]. The model considers the overhead from both hardware and software in collective communications. Through the generalized model, we hope to reveal both the strength and weakness of the underlying MPI implementations.

Let $f(m, p)$ be the *aggregated message length* in a collective operation. This equals to the sum of all messages being transmitted among all pairs of nodes in a collective communication operation. For example, in a *broadcast* operation, m is the length of the message broadcast from the source node, thus $f(m, p) = m(p-1)$, because $p-1$ destinations need to receive the same message. Similarly, $f(m, p) = m(p-1)$ for the *scatter*, *gather*, *reduce*, and *scan* operations. We have $f(m, p) = mp(p-1)$ for a *total exchange*.

The *collective messaging time*, $T(m, p)$, incurred with a collective operation is expressed as follows:

$$T(m,p) = T_0(p) + D(m,p) \qquad (1)$$

where $T_0(p)$ denotes the *startup latency* and $D(m, p)$ the *transmission delay*. Both terms can be obtained by a curve-fitting method to be described shortly. The term $D(m, p)$ is computed by:

$$D(m,p) = \frac{f(m,p)}{R(m,p)} \qquad (2)$$

where $R(m, p)$ is the *aggregated bandwidth* in MByte/s. As a matter of fact, the inverse term, $1/R(m,p)$, is the minimum time required to pass a single byte in a collective operation with an *aggregated message size* $f(m, p)$.

The $R_\infty(p)$ introduced in Section 1 is related to $R(m, p)$ as follows:

$$R_\infty(p) = \lim_{m \to \infty} R(m,p) = \lim_{m \to \infty} \frac{f(m,p)}{D(m,p)} \qquad (3)$$

This metric reflects the maximum aggregated communication capability of the network. Using this model, four performance metrics are summarized in Table 2 for the evaluation of collective operations, in general.

Table 2.Performance Metrics of Collective Communication Operations

Collective messaging time (μs)	$T(m,p) = T_0(p) + D(m,p)$
Startup latency (μs)	$T_0(p)$
Transmission delay (μs)	$D(m,p)$
Aggregated bandwidth (MByte/s)	$R_\infty(p)$

In Xu and Hwang's paper [31], an *asymptotic bandwidth*, $r_\infty(p)$, was defined. In fact, Eq. (1) can be used interchangeably with Eq.(2) in [31]. For example, in a point-to-point communication, $R_\infty = r_\infty$. In a *total exchange*, $R_\infty = p(p-1)r_\infty$. For other collective operations (*broadcast*, *gather*, *scatter*, *scan etc.*), $R_\infty = (p-1)r_\infty$

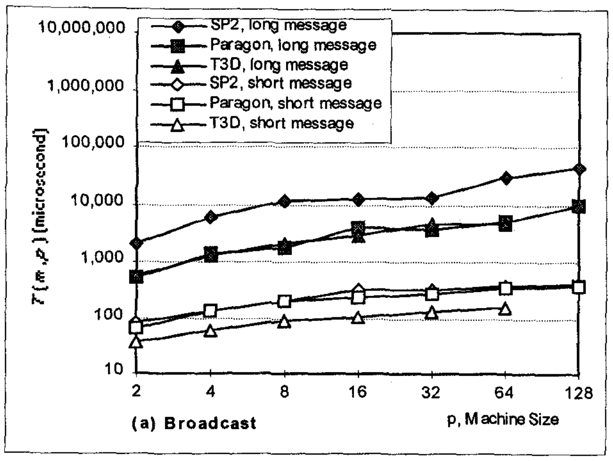
The time expressions for $T_0(p)$ and $D(m, p)$ are obtained in three sequential steps:

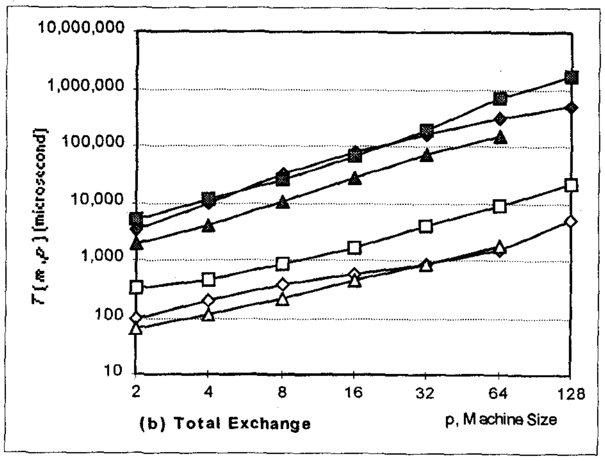
(1) Measure the *collective messaging time* values with various message sizes, m, over systems with increasing numbers of processing nodes, p.
(2) Curve-fit the measured timing data points to produce a closed-form expression for $T(m, p)$ as a function of p and m using Eq. (1).
(3) The startup latency, $T_0(p)$, expression is approximated by measuring the *collective messaging time* for a zero-byte or a short message, depending on the acceptability of the empty message definition in the target machine. This approximation is then used to obtain $D(m,p) = T(m,p) - T_0(p)$.

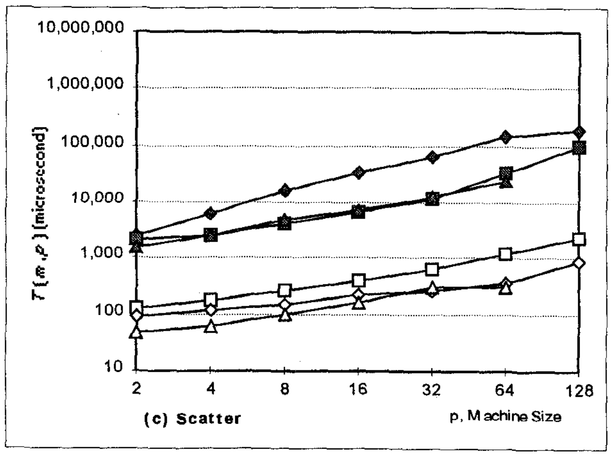
4: Startup Latencies and Attributes

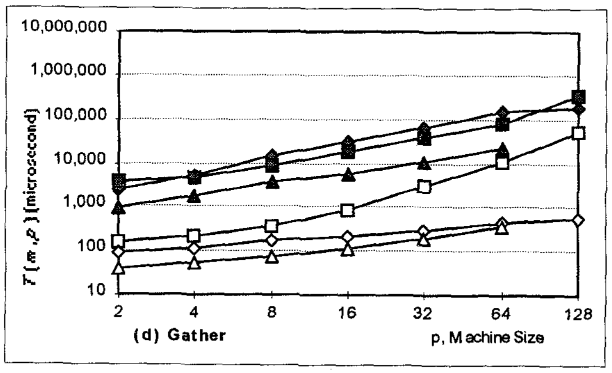
The *startup latency*, $T_0(p)$, is a monotonic increasing function of the machine size, p. The measured latencies of the three machines are plotted in Fig.1 for six collective functional types. Except the *scan* operation, the T3D has demonstrated the lowest *startup latency* in all collective operations. The lowest latency of using the T3D is 35 μs to broadcast a message to two nodes. On the 64-node T3D configuration, we measured a latency of 150 μs, 1700 μs, 298 μs, 365 μs, 209 μs, and 253 μs for the *broadcast, total exchange, scatter, gather, scan, and reduce* operations, respectively.

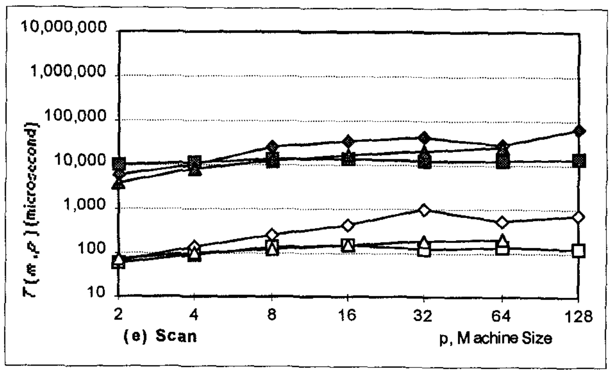
For *broadcast* operation, the SP2 has slightly higher

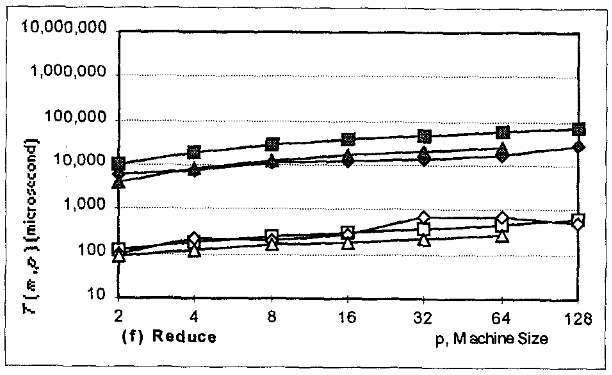












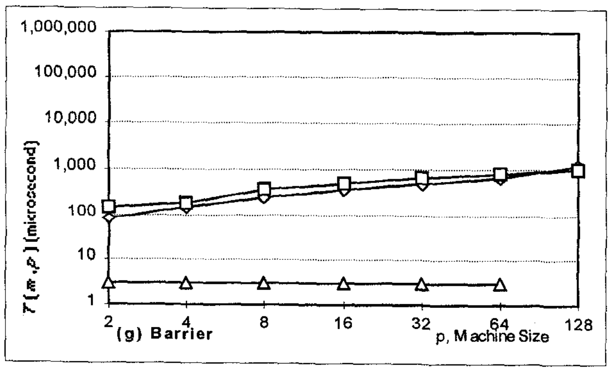


FIGURE 3. The collective messaging times, *T(m,p)*, as a function of machine size *p* for short messages (*m*=16 Bytes) and long messages (*m*=64 KBytes) for three message-passing computers.

latency than Paragon (Fig. 1a). The Paragon has the longest latency in *total exchange, scatter, gather,* and *reduce* operations. However, it performs the *scan* operation with even shorter latency than the T3D (Fig. 1e). For collective operations over a small number of nodes, the SP2 ranks a second place in latency for the *total exchange, scatter,* and *gather* operations.

Collectively messaging over large number of nodes, the Paragon has much greater latency than others. To summarize, we observe that the startup latency increases linearly with increase in machine size for the *gather, scatter,* and *total exchange* operations. The latency increases logarithmically for the *broadcast, scan, reduce,* and *barrier* operations, as larger machine configurations are used.

The startup latency includes the software overhead caused by executing the kernel subsystem for message passing. The T3D has a lower latency with special hardware support for fast messaging, lower network latency (20 ns per hop as opposed to 125 ns for the SP2, and 40 ns for the Paragon), and the use of *prefetch queue* and *remote processor store* to hide remote memory access latencies [1]. Although both SP2 and Paragon have ported the same MPICH version, the SP2 requires longer latency for lack of those hardware mechanisms in T3D [30]. The Paragon demonstrated the longest latency for two reasons: the longer NX messaging overhead and the routing delays in the 2-D mesh network [7].

5: Effects of Message Length

In Fig. 2, the total messaging time, $T(m, p)$, is plotted as a function of the message length, m, for all three systems with $p = 32$ nodes. The time increases slowly when the message length is shorter than 1024 bytes for all three machines. Increasing the message length beyond 4 KBytes, the *transmission delay*, $D(m, p)$, becomes the dominant factor in the collective messaging time. For long messages, the total messaging time increases almost linearly with respect to the increase in message length. The T3D clearly shows the lowest messaging time in all collective functions, except the Paragon performs the *scan* operation with less time (Fig. 2e).

For short messages, the Paragon performs the worst in *total exchange, scatter,* and *gather* operations, because of excessive latency encountered. However, the Paragon performs the *broadcast, total exchange, scatter, gather* faster than the SP2 for longer messages. The SP2 requires much longer time in passing long messages than either T3D or Paragon.

To *reduce* long messages beyond 64 KBytes, the SP2 shows the lowest messaging time (Fig. 2f). The ranking order of T3D, Paragon, and SP2 in total messaging time have something to do with the reported network bandwidths of 300 MBytes/s, 175 MBytes/s, and 40 MBytes/s, respectively. For example, in 64 node total exchange the SP2 requires 317 ms to transmit messages of 64 KBytes each. The total message exchanged is 256 MBytes and the aggregated bandwidth is 847 MBytes/s. Only 33% of the raw aggregated bandwidth (2.56 Gbytes/s = 40 MBytes/$s \times 64$ nodes) was consumed. On the Paragon and T3D, it takes much less time to perform the same messaging operation because of higher network bandwidth provided.

Another reason that the T3D outperforms the others in handling long messages is the use of the *block transfer engine* (BLT) [18]. This feature significantly reduce the amount of time to *total exchange* or *gather* large amounts of data among the nodes. It is interesting to observe that the SP2 is faster than Paragon in handling short messages. But for longer messages, the Paragon outperforms the SP2 in almost all operations except the *reduce* operation. This crossover in the relative performance of the two machines is attributed to the fact that the Paragon uses a higher bandwidth network and a dedicated message processor (i860) per node to process long messages more effectively. The long time to process short messages on the Paragon is mainly due to the longer startup latency experienced.

6: Effects of Machine Size

The relationship between the total messaging time and machine size is plotted in Fig. 3. The lower three curves correspond to a short message of 16 Bytes. The top three curves are for a long message of 64 KBytes. The time gaps between short and long messages reflects essentially the transmission delays in each functional type.

In this section, we focus on the effects of machine size on the messaging time. In general, the messaging time grows linearly or logarithmically with respect to the growth of the machine size. The curves in Fig. 3 are obtained by actual measurements of the total messaging time including the startup latency.

For short messages, these timing curves show a steady growth pattern and a relative ranking among the three machines very similar to the latency curves in Fig. 1. This is because the latency portion of the total communication time is much higher than the transmission delays measured for such a short message of 16 Bytes. However, the total messaging time grows much closer to a linear function of the machine size, when long messages are involved in the collective operations.

For the *broadcast* operation in Fig. 3a, the Paragon performs about the same as the T3D for long messages. But the Paragon performs as worse as the SP2 for short messages. In the case of *total exchange* (Fig. 3b), the SP2

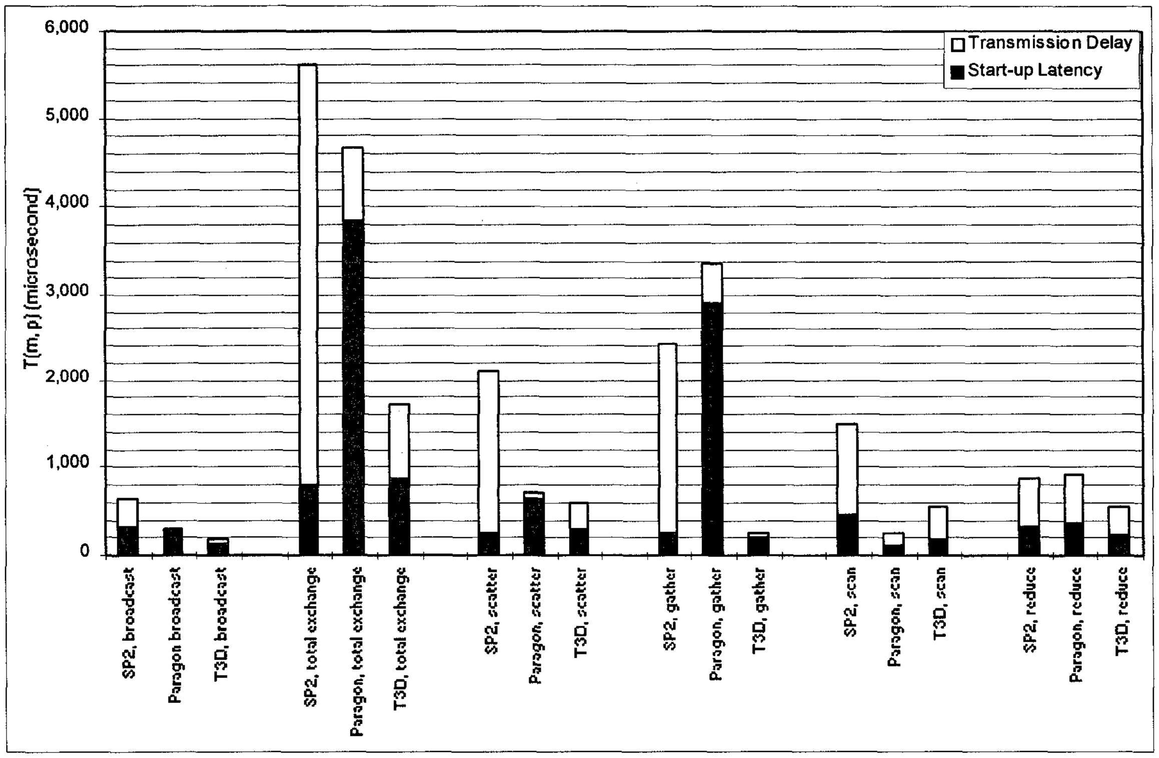


FIGURE 4. Breakdown of timing results in six MPI collective operations over p=32 nodes with m=1 KBytes per message.

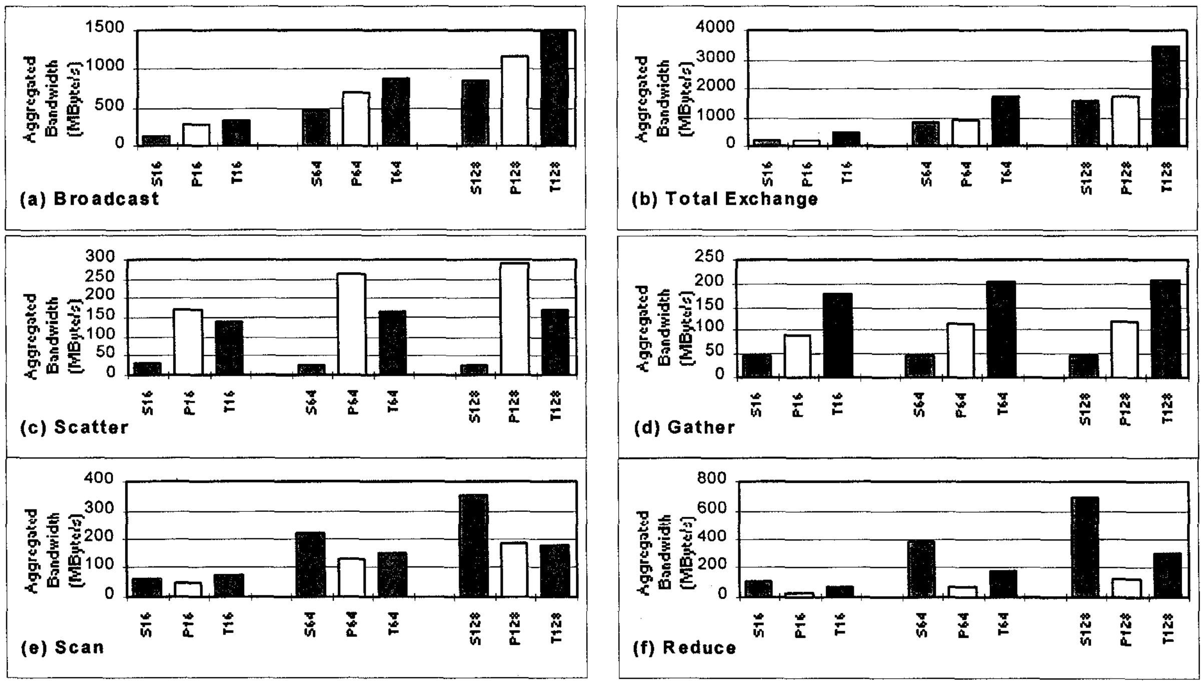


FIGURE 5. Aggregated bandwidths in performing different collective MPI operations on three different machine sizes.

demonstrates that it performs better than the Paragon in short messages. But the SP2 performs equally bad as the Paragon in handling long messages.

It is interesting to observe the different ranking of the three machines, as the message length changes from one extreme to another. The switching in performance ranking is also sensitive to the machine size. For an example, the SP2 and Paragon switch in their performance ranking in the *scatter* operation (long message curves in Fig. 3d) as the machine size increases from 4 to 8.

The most dramatic change in machine ranking is demonstrated in Fig. 3f, where the ordered ranking of SP2, T3D, and Paragon for long messages is replaced by the new order of T3D, Paragon, and SP2 for short messages. From these results in Fig. 3, we conclude that the total messaging time is more sensitive to the rapid increase in message length than to the slow change in machine size.

7: Breakdown of Timing Results

The relative performance of the six collective functions are further illustrated in Fig. 4 based on a case study of machines with 32 nodes and 1 KByte per message. We divide the total messaging time into two portions in each bar: the darkened bar showing the startup latency and the white bar showing the transmission delay. For the same collective operation, the performance ranking of the three machines is inversely proportional to the height of the bars. Obviously, the *total exchange* demands the longest time to complete.

The T3D shows the lowest startup latency in *broadcast, gather,* and *reduce* operations. The latency in Paragon varies dramatically with different collective functions performed. For example, in the *total exchange* and *gather* operations, the Paragon experienced 3857 μs and 2918 μs latencies, about 4 to 15 times greater than the SP2 and T3D counterparts.

On the other hand, Paragon experienced a comparable low latency as others in the remaining operations. In the case of *scan*, the Paragon even shows a lower latency. Our explanation of this phenomena is attributed to different collective algorithms used. The Paragon used the least efficient schemes to implement the *total exchange* and *gather* operations through the NX messaging subsystem in the node kernel. As the message length increases, the transmission delays (the white portions of the bars) will increase linearly.

8: Timing and Bandwidth Expressions

We have measured the messaging times of seven collective operations. The curve-fitted timing formulas for these collective operations are given in Table 3. The timing formula, $T(m,p)$, can be used to compute the actual execution time of the collective operation. For example, the *total exchange* time on the T3D is expressed by $(26p + 8.6) + (0.038p - 0.12)m$ in μs. Given $m = 512$ Bytes and $p = 64$, the time to perform the *total exchange* is calculated as 2.86 ms using this timing expression.

These formulas assist us in quantifying the total execution time of different optimization strategies in parallel program development. To optimize the application code, possible combinations of (m, p) should be tested to achieve a shorter execution time or a better efficiency for a given problem size.

The first part of the formula in Table 3 is the startup latencies of collective operations for three parallel machines. We observe O(log p) startup latency in the *barrier, scan, reduce,* and *broadcast* operations. Most algorithms for implementing collective operations were aimed at minimizing the number of messages sent. A treelike algorithm is usually employed to deliver the message. In EPCC/MPI, an unbalanced tree is formed among the participating processes to perform a *barrier* or a *broadcast* command; while a *binary tree* is formed to perform *reduce* operation [6].

Table 3.Timing Expressions for Collective Communications on Three MPPs

Operation	SP2	T3D	Paragon
Barrier	$123 \log p - 90$	$0.011 \log p + 3$	$147 \log p - 66$
Broadcast	$(55 \log p + 30) + (0.014 \log p + 0.053)m$	$(23 \log p + 12) + (0.013 \log p - 0.0071)m$	$(52 \log p + 15) + (0.019 \log p - 0.022)m$
Gather	$(3.7 p + 128) + (0.022 p - 0.011)m$	$(5.3 p + 30) + (0.0047 p + 0.0084)m$	$(48 p + 15) + (0.0081 p + 0.039)m$
Scatter	$(5.8 p + 77) + (0.039 p - 0.12)m$	$(4.3 p + 67) + (0.0057 p + 0.16)m$	$(18 p + 78) + (0.0031 p + 0.039)m$
Reduce	$(63 \log p + 26) + (0.016 \log p + 0.071)m$	$(34 \log p + 49) + (0.061 \log p - 0.00035)m$	$(77 \log p + 3.6) + (0.16 \log p - 0.028)m$
Scan	$(100 \log p - 43) + (0.0010 p + 0.23)m$	$(28 \log p + 41) + (0.0046 p + 0.12)m$	$(10 \log p + 73) + (0.0033 p + 0.28)m$
Total Exchange	$(24 p + 90) + (0.082 p - 0.29)m$	$(26 p + 8.6) + (0.038 p - 0.12)m$	$(97 p + 82) + (0.073 p - 0.10)m$

For the *gather*, *scatter*, and *total exchange* operations, we observe O(p) latency time, since they are either many-to-one, one-to-many, or many-to-many communication. In these cases, O(p) stages of data communication are required to move the data to its final destinations.

The second part of the formula in Table 3 is the *transmission delay*, *D*(m,p). The *aggregated bandwidth*, $R_\infty(p)$ for various collective operations is derived using the following formula:

$$R_\infty(p) = \lim_{m \to \infty} \frac{f(m,p)}{T(m,p) - T_0(p)} \qquad (4)$$

This function indicates the ultimate execution rate of a collective operation on a machine, when m becomes significantly long. In Fig. 5, we plot the aggregated bandwidth, $R_\infty(p)$, against the machine size p. The aggregated bandwidth monotonically increases for all collective operations. However, their growth rates vary from function to function dramatically. The aggregated bandwidth can set the limit on the relative performance of machines with different sizes. For example, the *scatter* operation on the Paragon results in the highest aggregated bandwidth (Fig. 5c). Therefore, Paragon is expected to have shorter messaging time compared with the SP2 and T3D of the same size. For 64 nodes, the aggregated bandwidths of *total exchange* for T3D, Paragon, and SP2 are 1.745, 0.879, and 0.818 GBytes/s, respectively.

For different collective functions, the ranking of their aggregated bandwidths changes from function to function dramatically. For example, the bandwidth ranking of the *broadcast* operation is T3D, Paragon, and SP2 in descending order. However, the machine ranking is changed to a new order, SP2, T3D, and Paragon, for the *reduce* operation. The message here is that one should not use the machine ranking for one collective operation to predict the relative machine performance of another collective operation.

9: Conclusions

Overall, we rank the T3D the highest in collective messaging performance, compared with the SP2 and Paragon. The T3D does uniformly best in all collective functions, with the only exception of trailing the Paragon in performing the *scan* operation on 16 nodes or more.

Considering message length, the SP2 outperforms the Paragon in any short messages less than 1 KBytes. The Paragon performs better than the SP2 in long messages, except the *reduce* operation. For short messages, the SP2 and Paragon perform about the same in the *broadcast* and *barrier* operations. For long messages, the T3D and SP2 have approximately the same performance in the *broadcast, scatter*, and *reduce* operations.

The T3D is well supported by some special hardware features, which helped lowering the startup latency as well as the transmission delay of large number of messages in a collective communication. The hardwired barriers are effective to reduce the synchronization time on the T3D significantly.

The Paragon is weak in handling short messages for its long latency from NX overhead in collective messaging passing. This surge in latency is especially true in performing the *total exchange* and *gather* operations on the Paragon. The SP2 is weak in handling long messages for its limited network bandwidth compared with the other two machines.

The accuracy of our measured timing results could be offset slightly by the following six factors: First, the resolution of the timer affects the accuracy of measurement. Second, the interference from other users in the multicomputer environment affects the measurement of the dedicated applications. Third, the warm-up effect forces us to throw away the results from initial runs due to cold caches or lack of data locality. Fourth, we experienced the limitation of a small local memory, such as 16 MBytes per node in some of the Paragon nodes. This may lead to excessive page faults or disk crushes. Fifth, the runtime node allocation affects the implementation of a collective communication pattern. Finally, none of the three target machines is supported by a truly real-time operating system, our measurements must bear some runtime conditions.

The aggregated bandwidth introduced by Xu and Hwang [31] offers a better metric to quantify the data transfer rate in a collective message passing operation. The asymptotic bandwidth by Hockney [13] is only effective in characterizing point-to-point communications. We did not apply the active messages (Culler, *et al.* [25]) or the MPI-FM (Chien, *et al.* [19]) in our benchmark experiments, because they have not being widely ported on those target machines we have tested. We suggest extended research be conducted in evaluating the use of active messages or fast messages in MPI applications.

Our results should be useful to those who are developing parallel applications on message-passing multicomputers. The collective latency and messaging time expressions and reported numerical data can be applied in trade-off studies of SPMD or MPMD computations using collective message passing. The latency and messaging delays can be used to predict MPP performance as reported in [32]. These results on communication overhead can be also used in the optimization of parallel applications on multicomputers as reported in [15, 16]. We shall report the entire STAP benchmark performance results in a separate paper.

Acknowledgments

This work was carried out mainly at the University of Hong Kong, with part of the results generated at the University of Southern California. We want to thank Wei-Chung Cheng and Shihg-Liang Ou of USC in producing some of the timing data. We would like to thank David Martinez and Robert Bond at MIT Lincoln Laboratory for their sponsorship of this work. We want also to thank the computing support groups at HKU and USC for their technical assistance in the benchmark experiments.

References

[1] D. Adams; "Cray T3D System Architecture Overview Manual," Cray Research, Inc., (http://www.cray.com/PUBLIC/product-info/mpp/CRAY-T3D.html), September 1993.

[2] R. Alasdair, A. Bruce, J. G. Mills, and A. G. Smith; "CHIMP Version 2.0 User Guide," University of Edinburgh, March 1994.

[3] V. Bala, J. Bruck, R. Cypher, P. Elustondo, A. Ho, C.-T. Ho, S. Kipnis, and M. Snir; "CCL: A Portable and Turnable Collective Communication Library for Scalable Parallel Computers," *IEEE Transactions on Parallel and Distributed Systems*, 1995, 6(2): pp 154-164.

[4] J. Bruck, L. de Coster, N. Dewulf, C.-T. Ho and R. Lauwereins, "On the Design and Implementation of Broadcast and Global Combine Operations Using the Postal Model," IEEE Trans. on Parallel and Distributed Systems, Vol. 7, No. 3, March 1996, pp. 256-265.

[5] G. Burns, R. Daoud, and J. Vaigl; "LAM: An Open Cluster Environment for MPI," Ohio Supercomputer Center, (http://www.osc.edu/lam.html), May 1994.

[6] Kenneth Cameron, Lyndon J. Clarke, A. Gordon Smith; "CRI/EPCC MPI for CRAY T3D," *1st European Cray T3D Workshop*, September 1995.

[7] Thomas H. Dunigan; "Beta Testing the Intel Paragon MP," *Technical Report*, ORNL/TM-12830, Oak Ridge National Lab., June 1995.

[8] Stephen Fleischman and Dan Golan; "MPI Collective Communication on the Convex Exemplar SPP-1000," *MPI Developers Conference*, University of Notre Dame, June 1995.

[9] H. Franke, *et al.*; "MPI-F Programming Environment for SP1/SP2," *15th International Conference on Distributed Computing Systems*, Vancouver British Columbia, CANADA, May 1995.

[10] Vasilios Georgitsis, and John S. Sobolewski; "Performance of MPL and MPICH on the SP2 System," *MPI Developers Conference*, University of Notre Dame, June 1995.

[11] Ian Glendinning; "The GENESIS Distributed-Memory Benchmark Suite Release 3.0," (http://www.hpcc.soton.ac.uk/RandD/genesis/genesis.html), 1994.

[12] B. Groop, R. Lusk, T. Skjellum, and N. Doss; "Portable MPI Model Implementation," Argonne National Laboratory, *Technical Report*, July 1994.

[13] Roger W. Hockney; "The Communication Challenge for MPP: Intel Paragon and Meiko CS-2," *Parallel Computing*, 1994, Vol. 20, pp. 389-398.

[14] Roger W. Hockney and M. Berry; "Public International Benchmarks for Parallel Computers: PARKBENCH Committee Report No. 1," *Scientific Computing*, Vol. 3, No. 2, February 1994, pp. 101-146.

[15] K. Hwang and Z. Xu; "Scalable Parallel Computers for Real-Time Signal Processing," *IEEE Signal Processing Magazine*, July 1996, pp. 50-66.

[16] K. Hwang, Z. Xu, and M. Arakawa; "Benchmark Evaluation of the IBM SP2 for Parallel Signal Processing," *IEEE Trans. Parallel and Distributed Systems*, May 1996, pp. 522-536.

[17] Dennis Kafura, Liya Huang; "mpi++: A C++ Language Binding for MPI," *MPI Developers Conference*, University of Notre Dame, June 1995.

[18] R. Kent Koeninger, Mark Furtney, and Martin Walker; "A Shared Memory MPP from Cray Research," *Digital Technical Journal*, Vol. 6, No 2, 1994.

[19] Mario Lauria and Andrew Chien; "MPI-FM: A High Performance MPI for Workstation Clusters," (http://www-csag.cs.uiuc.edu/papers/index.html), March 1996.

[20] Rusty Lusk, Nathan Doss, Anthony Skjellum, William Gropp; "MPICH: A High-Performance Portable Implementation of the MPI Standard," *MPI Developers Conference*, University of Notre Dame, June 1995.

[21] MHPCC, MHPCC 400-Node SP2 Environment, Maui High-Performance Computing Center, Maui, HI, October 1994.

[22] Prasenjit Mitra, Robert van de Geijn, David G. Payne, and Lance Shuler; "Fast Collective Communication Libraries, Please," *Proceeding of the Intel Supercomputing Users' Group Meeting*1995.

[23] MPI Forum, "MPI: A Message-Passing Interface Standard," *International Journal of Supercomputer Applications*, 1994, 8 (3/4).

[24] N. Nevin; "The Performance of LAM 6.0 and MPICH 1.0. 12 on a Workstation Cluster," *Technical Report*, OSC-TR-1996-4, Ohio Supercomputer Center, 1996.

[25] NOW Project; "Efficient and Portable Implementation of MPI using Active Messages," University of California, Berkeley, (http://now.cs.berkeley.edu/Fastcomm/mpi.html), July 10, 1996.

[26] Natawut Nupairoj and Lionel M. Ni; "Performance Evaluation of Some MPI Implementations on Workstation Clusters," *Proceedings of the 1994 Scalable Parallel Libraries Conference*, October 1994.

[27] Natawut Nupairoj and Lionel M. Ni; "Benchmarking of Multicast Communication Services," Technical Report MSU-CPS-ACS-103, Dept. of Computer Science, Michigan State University, April 1995.

[28] SDSC, SDSC's Intel Paragon, San Diego Supercomputer Center, (http://www.sdsc.edu/Services/Consult/Paragon/paragon.html), 1995.

[29] Anthony Skjellum, Paula Vaughan, Christopher Roberts; "UNIFY: Interoperable MPI and PVM Programming in a Workstation-Network Environment," *MPI Developers Conference*, University of Notre Dame, June 1995.

[30] Graig B. Stunkel, *et al.*; "The SP2 Communication Subsystem," *IBM Technical Report*, August 22, 1994.

[31] Zhiwei Xu, and Kai Hwang; "Modeling Communication Overhead: MPI and MPL Performance on the IBM SP2," *IEEE Parallel & Distributed Technology*, Spring 1996, pp. 9-23.

[32] Zhiwei Xu and Kai Hwang; "Early Prediction of MPP Performance: SP2, T3D, and Paragon Experiences," *Parallel Computing*, October 1996.