The Parallel Communication Protocol in BCL-4[*]

Xiaocheng Zhou, Zhigang Huo, Jie Ma, Dan Meng
National Research Center for Intelligent Computing Systems
Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100080, P.R. China
{zxc, zghuo, majie, md}@ncic.ac.cn

Abstract

As CLUMPS become the main stream of clusters and the number of nodes in a cluster increases, it requires enhancing the bandwidth performance and availability of the communication system used in clusters. Parallel communication based on multiple system area networks (SANs) can fulfill the requirements. This paper introduces the parallel communication protocol used in BCL-4, which is a high efficient communication system used in DAWNING-4000A, a large-scale LINUX cluster. It dispatches small messages and sub-messages stripped from large messages into multiple SANs and maintains the communication semantics as before. The parallel communication process is transparent to both users and the control program on network interface card (NIC). It also provides an efficient load balance mechanism. Using the parallel communication protocol, BCL-4 provides many key features, such as multiple throughput, high availability, and backward compatibility. The experimental results show that the peak bandwidth of BCL-4 over two Myrinet is 494.7MB/s, which is almost twice of that over one, and that there is only 0.02us overhead of short message at the same time.

1. Introduction

The recent advances in improved microprocessor performance and high speed networks are making clusters an appealing vehicle for cost effective parallel computing [1]. In the latest TOP500 list (published in November 2003), there are 208 clusters in total. Clusters have been widely used in the areas of scientific and engineering computing, Internet service, and database applications etc. Communication performance is one of the most critical factors determining the performance of a whole cluster system [2].

Nowadays CLUMPS has become the main trend of cluster systems. With the number of processors in a node and the number of nodes in a cluster increases, there are more and more traffic among nodes. This requires the communication system used in clusters to provide higher bandwidth performance. Since large-scale cluster will reduce system availability, high availability also becomes a key problem. However, in the clusters using only one SAN, the bandwidth performance of the communication system is limited by the performance of a single NIC and I/O bus. Although the performance of them has been improved a lot, it still can not satisfy the fast growing demand of the parallel computing. At the same time, because there is only one SAN, when there are faults in the network, the nodes can't communicate with each other. So it can't meet the high availability requirement, either.

Using multiple SANs is a promising technique to overcome bandwidth limitation and enhance availability of high performance clusters [3]. This technique implies the utilization of several NICs per SMP node, attached to one or more independent I/O buses. Many efforts [3, 4] have been put into various issues concerned with the utilization of multiple SANs, such as communication protocols, performance characteristics, fault tolerance, and implementation of system software and libraries.

In this paper, the parallel communication protocol used in BCL-4 is introduced. BCL-4 is a low-level parallel communication system, which will be adopted in the DAWNING-4000A. DAWNING-4000A is a grid-enabling CLUMPS consisting of more than 512 4-way SMPs of AMD Opteron processor. Powered by more than 2048 processors, its peak speed will be more than 10Tflops. All nodes will be connected with Gigabit Ethernet, Myrinet and DCnet. The last one is

[*] This work is supported by the National Natural Science Foundation of China (No. 60173019) and the National '863' High-Tech Program of China (No. 2002AA1Z2102)

high performance interconnect developed on our own. Both Myrinet and DCnet are supported by BCL-4. After using the parallel communication protocol, BCL-4 has the following key features.

Multiple throughput: Using multiple SANs, BCL-4 can provide multiple throughput. Under the test condition described in Section 4, the experimental results show that the peak bandwidth of BCL-4 over two Myrinet SANs is 494.7MB/s, which is almost twice of that over one.

High availability: The nodes can communicate with each other as long as there is one of the multiple SANs works well. At the same time, when the nodes concerned with the fault are not involved in the communicating process, BCL-4 can still make full use of all the available SANs of the system to get good performance.

Backward compatibility: The parallel communication protocol in BCL-4 is transparent to both users and the control program on NIC. The users can use the multiple SANs the same as there is only one. The middleware over BCL-4, such as MPI, VI, and Sockets, can be used with only minor changes after the parallel communication protocol is implemented.

This paper is organized as follows.

Section 2 and 3 introduce the architecture and the main design issues of the parallel communication protocol in BCL-4. Then the performance is presented and analyzed in Section 4. Finally, we present our conclusions and the ongoing research in Section 5.

2. Architecture

There are three parts in BCL-4, including the driver, the library, and the control program running on NICs. Over BCL-4, some middleware such as MPI, VI, and Sockets are implemented. Applications can use the middleware or the APIs implemented in the library directly to communicate. The parallel communication protocol is implemented in the library, so it is transparent to both the users and the control program.

The architecture of parallel communication in BCL-4 is illustrated in Figure 1. Communication is occurred between ports. Each port has a sending request queue (SRQ), a receiving queue (RQ) in each NIC and corresponding event queues (EQs) in the library. After opening a port, users can communicate through multiple SANs with each other by calling the communication APIs.

Sending requests are submitted by users to send messages. After being stripped, they are dispatched into the selected SRQs of the available NICs. As the result, the corresponding messages are stripped into multiple sub-messages, which are dispatched to multiple SANs and transmitted to the destination respectively. When all the sub-messages stripped from the same message arrived, the corresponding receiving events in the receiver and the sending completion events in the sender will be assembled orderly and made visible to the users.

The node id table, which is maintained by a daemon, is used when stripping and dispatching sending requests. The function of the daemon is to explore the connection condition of the SANs.

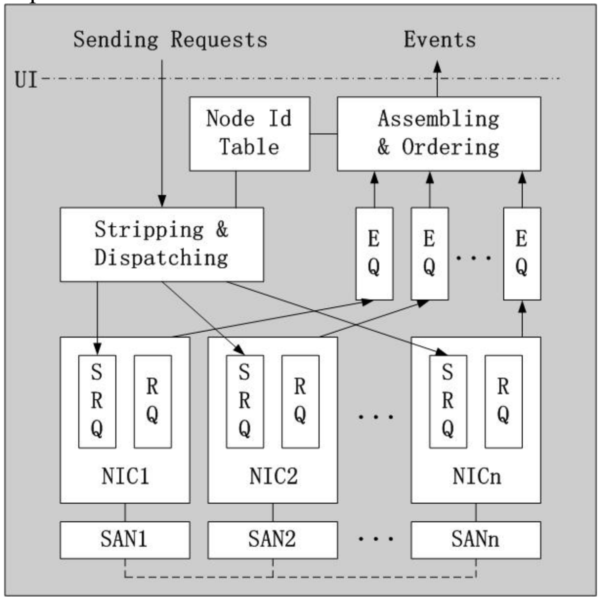

Figure 1 Architecture of parallel communication in BCL-4

The multiple NICs of each node reside in multiple SANs. To employ the load balance mechanism mentioned in section 3.4, all the SANs should be connected with each other.

3. Design issues

3.1. Multi-layer naming mechanism

In BCL-4, nic_id is used as the identifier of a NIC and node_id is used as the identifier of a node. Because there are multiple NICs in each node, nic_id is in many-one correspondence with node_id. When sending APIs are called to send messages, only the target_node_id is designated. However, when messages are transmitted by the NIC ultimately, the target_nic_id must be appointed. So there need a mechanism to convert node_id to nic_id.

In BCL-4, this is done by a node id table. It is a two-dimensional array. In the array, node_id is used as the subscript of the first dimension and the serial number of the NICs in a node is used as the subscript of the second dimension. The context of the array is the nic_id. When messages need be transmitted by a NIC, target_node_id is appointed by the user. Which NIC of the target node will the message be transmitted to is decided by our parallel communication protocol. With the two subscripts, the target_nic_id can be found in the table.

3.2. Stripping & dispatching mechanism

By stripping messages to multiple sub-messages and dispatching them to multiple SANs, BCL-4 can get multiple bandwidth performance. There are several key issues to be solved.

The first key issue is to determine an appropriate threshold length of messages, which will be stripped. As shown in Figure 2, there is about 0.5us overhead when 2 bytes long message are stripped and dispatched over two SANs. This overhead is mainly introduced by request stripping and event assembling operations. Because the time spent in transmitting messages through SANs is halved, as the length of messages increases, the latency penalty becomes lower and lower. Finally, it is outweighed by multiple throughput when the length of message reaches 128 bytes. So in BCL-4, 128 bytes is used as the threshold when there are two SANs to get the best performance. If the length of a message is less than this threshold, it is called a short message; or else a large message. Only large messages are stripped into sub-messages in BCL-4. The value of threshold will change as the number of the SANs increases.

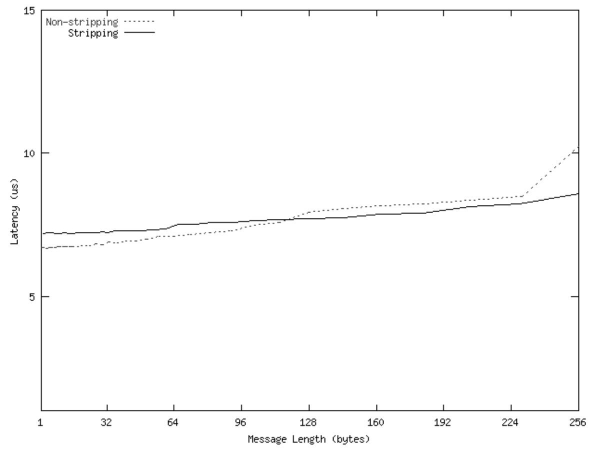

Figure 2 Latency comparison of stripping messages over two SANs and non-stripping

The second issue to be concerned is how to strip large messages into multiple sub-messages. To get good load balance and high performance, large messages should be evenly stripped. However, when large messages distribute on multiple pages, there will be multiple DMA operations to transmit messages between main memory and NIC, so the performance may be reduced. So in BCL-4, large messages are evenly stripped and sub-messages are made to distribute on as least pages as possible.

The final issue is how to dispatch sub-messages and short messages to multiple SANs. In BCL-4, the sub-messages stripped from the same large messages will be dispatched to different SANs. The method to dispatch short messages will be present in section 3.4 in detail.

3.3. Parallel semantics

To make the parallel communication protocol transparent to the middleware and the application, all the communication semantics in BCL-4 is kept as before.

First, the semantic of events. In BCL-4, sending completion events are used to notify the success or failure of sending requests. When the status is success, it indicates that the corresponding messages have been successfully received. Receiving events are used to notify the arrival of message. When messages are stripped to sub-messages and transmitted through multiple SANs, all the events of the same messages should be combined to one. How to record the events of the same messages and how to combine them to one are the two problems to do this work. BCL-4 adopts different approaches to deal with sending completion events and receiving events respectively.

To sending completion events, it is done by signs in the sending request structures. All the NICs used to send messages are parted as main sending NIC and bound sending NICs. A count is added in the structure of the main sending NIC to store the number of the NICs used to transmit the same message. In the structure of bound sending NICs, a field is added to store the position of the corresponding structure of the main sending NIC. When a successful sending completion event is received in the library layer, if it comes from the main sending NIC, the count of the structure will be decreased by 1; if it comes from the bound sending NICs, the structure of the main sending NIC will be found first. Then the count of the structure of the main sending NIC will be decreased by 1. When the count is 0, it shows that all the sending events of the message have been received, so the event can be passed to user to notify the whole message has been sent successfully. When there is an error in one of the

NICs, however, it can't be done like this. The method to handle this situation will be discussed in Section 3.5.

To receiving events, it is done differently. The NICs used to receive messages are also divided to main receiving NIC and bound receiving NICs. In receiving events of BCL-4, there is some information about the sender such as send_node_id and send_port_id. A tag in the receiving events is used to separate sub-messages from non-stripped short messages. There are counters in each receiving NICs to count unsettled sub-messages received from each port. When a sub-message is received from a NIC, the counter of the sending port in this NIC is increased by 1. If it is 1, the counters of the same sending port in all NICs will be looked. If they are all not zero, we can say that all sub-messages stripped from the same message have been received. So the receiving event of the main receiving NIC can be passed to the user and all the counters are decreased by 1. Otherwise, if it is the main receiving NIC, the event will be stored in a queue, which will be discussed below.

The second semantics need to be concerned is the ordered delivery of messages from the same source port to the same destination port, which is called as a link in this paper. Since large messages are stripped to sub-messages and transmitted through multiple SANs and short messages aren't stripped and transmitted through only one SAN, when some of the sub-messages stripped from large message and some following short messages on the same link have reached, other sub-messages stripped from the same large message transmitted through other SANs may have not reached. To preserve the order of them, all the subsequent short messages on the same link can't be provided to the user until all sub-messages of the large message have arrived.

In BCL-4, an event queue is introduced in the main receiving NIC to preserve the sequence semantics of messages on the same link. According the dispatching mechanism of BCL-4, all short messages and one sub-message of large messages will be received by the main receiving NIC. So the sequence of the short messages and sub-messages in the main receiving NIC is used as the sequence of messages. When a receiving event is received by the main receiving NIC and one of the sub-messages transmitted to other NICs is not arrived, the receiving event and all the following receiving events of the same link must be stored in the queue to keep the order of them. When all the sub-messages of the same large message have arrived, those receiving events stored in the queue will be passed to the user at the order in which they received.

The last semantics is the semantics of the short messages sent after a RMA operation. In BCL-4, no receiving events will be generated when messages are sent in RMA mode. To indicate the completion of a RMA operation, a short message on the same link will be sent in normal sending mode subsequently if needed. Because they are transmitted orderly, when the receiver receives the receiving event of the short message, it means that the message transmitted by RMA has arrived, too. Since large messages are transmitted through multiple SANs, to keep this semantics, the short messages following them must be transmitted through multiple SANs, too. In BCL-4, this type of short messages is called as sync-message to distinguish from other short messages. They will be stripped and dispatched to multiple SANs no matter whether the length is larger than the threshold.

3.4. Load balance mechanism

When there are multiple SANs used in communication, the load on each SANs need be balanced to acquire the best communication performance. By stripping and dispatching large messages, BCL-4 can make the load of them be evenly distributed on multiple SANs. However, the load balance of short messages is gained by a different method.

In BCL-4, short messages are transmitted from main sending NIC to main receiving NIC. To balance the load, the main sending NIC and main receiving NIC of different links are different. Since receiver can't forecast the source node of messages that will be received, main receiving NIC on each link should be one and only and decided by the receiver to provide the receiving buffer correctly. It is selected by the receiving port_id. The main sending NIC of each link is decided by the node_id of destination. When there are more than one communication links, the load of short messages will be evenly distributed on all the SANs.

3.5. Fault handling mechanism

By using the ACK/NAK mechanism and other mechanisms, BCL-4 can automatically handle transient network errors such as packet drop and corruption. However, when there is a catastrophic error on a link, BCL-4 cannot handle this all by itself. It will simply abort the delivery of all messages on the same link and inform the user. Two strategies can be used to handle this situation. One is dropping all subsequent sending requests on the same link, the other is resending them repeatedly for a predefined duration. However, large messages are transmitted through multiple SANs in

BCL-4. When there is an error in one SAN, sub-messages stripped from large messages transmitted through other SANs may have been reached the receiver, thus it can't be dropped. So only the latter can be used in BCL-4.

4. Performance and analysis

The testing platform consists of 8 Linux workstations, which is a 2-way 1.6GHz AMD (Opteron Processor 242) SMP with 133MHz PCIX Bus. Two Myrinet M3F-PCIXD NICs are used on each host and interconnected by M3F-SW16M switches.

In this section, the communication performance of BCL-4 over one Myrinet NIC and two is compared. Figure 3 shows the latency performance comparison. The minimum latency of BCL-4 over one is about 6.70us. The value over two networks is almost the same. When the length of messages is shorter than 128 bytes, there is average 0.02us overhead for each message. However, when it is longer than 128 bytes, for the messages are transmitted through two Myrinet NICs and halve the time spent on network, just as analyzed in section 3.2, the latency over two Myrinet NICs is much shorter than that over one Myrinet NIC.

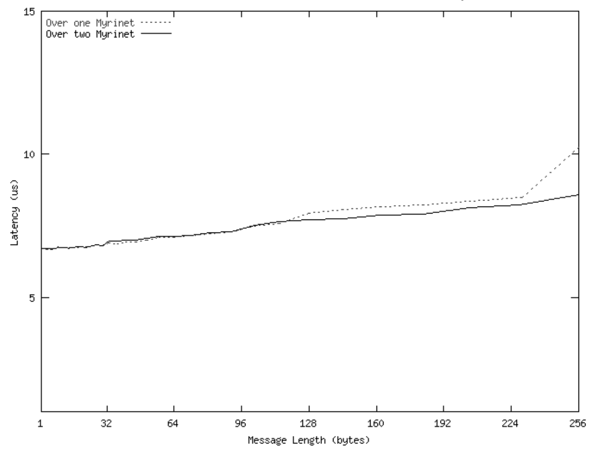

Figure 3 Latency comparison of BCL-4 over one Myrinet NIC and two Myrinet NICs

The bandwidth performance is compared in Figure 4. As we can see, the peak bandwidth of BCL-4 over one Myrinet NIC is 247.4MB/s. When the message is 2048 bytes long, it reaches the peak bandwidth performance. The half-bandwidth is reached with less than 1024 bytes message. It is 494.7MB/s over two and the peak point is about 4816 bytes. With about 1856 bytes message, the half-bandwidth is reached. When two Myrinet NICs are used, the bandwidth performance is almost doubled.

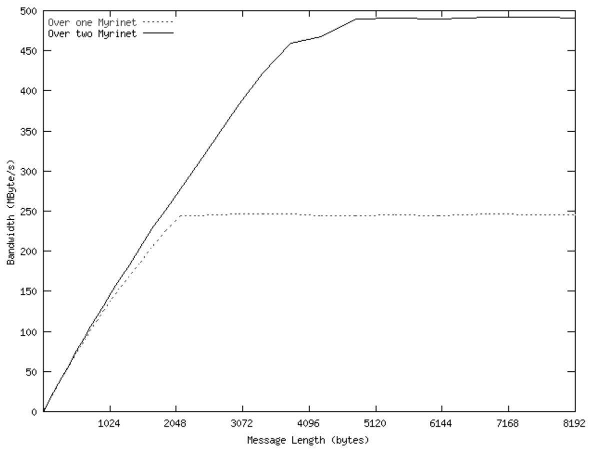

Figure 4 Bandwidth comparison of BCL-4 over one Myrinet NIC and two Myrinet NICs

The performance of MPI and VI over BCL-4 is showed by Table 1. The minimal latency of MPI over BCL-4 is 7.05us and the peak bandwidth is 461.7MB/s. The minimal latency of VI over BCL-4 is 8.44us and the peak bandwidth is 469.4MB/s.

Table 1 Performance of MPI and VI over BCL-4

	Latency (us)	Bandwidth (MB/s)
BCL-4	6.72	494.7
MPI	7.05	461.7
VI	8.44	469.4

5. Conclusion and Future work

To overcome bandwidth limitation and enhance availability of high performance clusters, one of the promising methods is to apply the parallel communication protocol based on multiple SANs. In this paper, the parallel communication protocol used in BCL-4 is present. It can make the bandwidth performance over two Myrinet NICs improve almost twice and the latency of short messages only reduced 0.02us. At the same time, all the semantics are kept so that the users can use them as they used them before. With the parallel communication protocol, BCL-4 can provide multiple throughput, high availability, and backward compatibility, etc.

At the future work, we will analyze the performance of BCL-4 when the number of NICs increases. It is helpful for us to find the bottleneck in communication systems to analyze the performance change with the change of the number of I/O buses and NICs. The performance of large-scale clusters will also be studied to see the impact of host number on the performance of BCL-4. We also plan to make the protocol running on heterogeneous SANs, mainly Myrinet and Dcnet. We will analyze the impact on the mechanism used for load balance and other mechanisms when the performance of SANs used to communication is different.

6. References

[1] "Cluster Computing White Paper", Available at http://www.dsg.port.ac.uk/mab/tfcc/WhitePaper/final-paper.pdf

[2] S. Araki, A. Bilas, C. Dubnicki, J. Edler, K. Konishi, and J, Philbin, "User-space communication: A quantitative study", *In Proceedings of SC'98*, Nov 1998.

[3] Fabrizio Petrini, Salvador Coll, Eitan Frachtenberg, Adolfy Hoisie, Leonid Gurvits, Using Multirail Networks in High-Performance Clusters, In Proceedings of IEEE Cluster 2001, October 2001.

[4] Dan Meng Jie Ma Zhigang Huo Fan Gao, User-Level Zero-Copy Parallel Communication Protocol for Clusters, In Proceedings of CNCC 2003.

[5] Ma Jie, He Jin, Meng Dan and Li Guo-jie. BCL-3: A High Performance Basic Communication Protocol for Commodity Superserver DAWNING-3000. Journal of Computer Science and Technology, 2001, 16(6): 522-530.

[6] Meng Dan, Ma Jie, et al. Semi-User-Level Communication Architecture. In Proceedings of IPDPS 2002.

[7] Von Eicken, D. E. Culler, S. C. Goldstein and K. E. Schauser. Active Message: A Mechanism for Intergrated Communication and Computation. In Proceeding of the 19th International Symposium on Computer Architecture, May 1992.

[8] V. Eiken, A. Basu, V. Buch, and W. Vogels. U-net: A user-level network interface for parallel and distributed computing. In Proceeding of the 15th ACM Symposium on Operating Systems Principles, 1995.

[9] S. Pekin, Karamcheti and A. Chien. Fast Message (FM): Efficient, Portable Communication for Workstation Cluster and Massively-Parallel Processors, IEEE Concurrency, 1997.

[10] N. J. Boden, D. Cohen, R. E. Felderman, A. E. Kulawik, C. L. Seitz, J. N. Seizovic, and W. K. Su. Myrinet: A gigabit-per-second local area network. IEEE Micro, Feb 1995.