Grid Gateway: Message-Passing between Separated Cluster Interconnects *

Cui Wei, Ma Jie, and Huo Zhigang

Institute of Computing Technology, Chinese Academy of Sciences
P.O. Box 2704, Beijing 100080, P.R. China
Graduate School of the Chinese Academy of Sciences, Beijing 100039
{cw, majie, zghuo}@ncic.ac.cn

Abstract. Geographically distributed computing requires high-performance clusters to be integrated to solve problems in computational Grid. Because cluster interconnect is isolated, its low-level communication protocol doesn't exchange messages with others directly. This paper presents a plug-in, Grid Gateway, which enables separated low-level communication protocols to communicate with each other. Grid Gateway can be used in many topologies of inter-cluster network. It has some dynamic features, such as support for multi-gateway mechanism to enhance communication performance. Grid Gateway allows low-level communication protocol to involve in the high-performance Grid computing. Thus it is expected to support the implementation of Grid-enabled tools over it, such as Grid-enabled MPI. This paper describes its architecture and implementation, and presents some design issues.

1 Introduction

Computational Grid provides more computational power than present machines, which meets the needs of many fields in science and commerce [1]. Clusters with lots of applications and deployed plentifully play an important role in the Grid environment. Currently, cooperative computing among clusters is discussed widely.

Because cluster interconnect is isolated, low-level communication protocol can't exchange messages with each other directly. Grid Gateway (GGW) is introduced to enable separated low-level communication protocols to communicate with each other. There is a gateway process each on the nodes, namely gateway nodes, in each cluster. When a node attempts to send messages to the node in a remote cluster, messages are transferred to one gateway node of local cluster at first. Then gateway process sends them to the remote gateway node, which forwards the messages to the destination at last. This data movement is transparent to users, thus user has a single image to the involved clusters. GGW extends the functionality of low-level communication protocol. Grid-enabled tools such as Gird-enabled MPI[3] can be implemented over it.

* This work is supported by the National '863' High-Tech Program of China (No. 2002AA104410, 2002AA1Z2102)

Grid Gateway introduces four key design issues as follows, which are described in the paper.

- Global ID. Communication between cluster low-level communication protocols requires that sender and receiver could identify each other uniquely. Designing a hostname and address space is the first consideration.
- Communication semantic. Extension of communication path and data buffering at Gateway nodes result in some semantic issues of protocol, such as how to notify sender the send success, what sender does when the receiving buffers of gateways are unavailable.
- Multi-routing mechanism. To send a message out of local cluster, sender chooses a gateway node to transfer the message. The gateway node also needs to choose a route to the target gateway node. In GGW a multi-routing mechanism is used to deal with the things.
- Flow control. Gateway stores messages of all the nodes before sending successfully, but the number of buffers is limited. Therefore a flow control mechanism is designed to avoid buffer overflow.

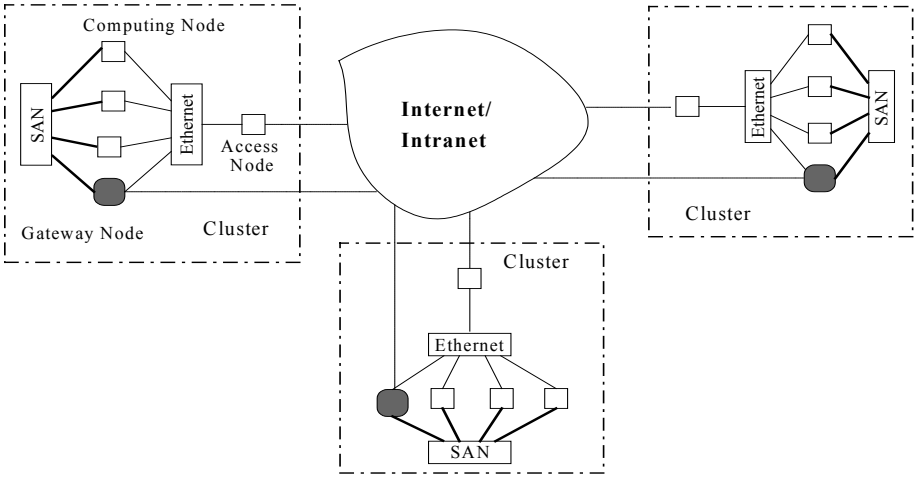

Fig. 1. Grid Gateway Background Topology

This paper is organized as follows. Section 2 presents the architecture of GGW. Section 3 describes some design issues. Then the design and implementation of GGW is described in section 4. In the final section the paper is summarized briefly.

2 Architecture of Grid Gateway

As shown in figure 1, several clusters are interconnected through Internet or Intranet (also called inter-cluster network) and compose of a computational Grid. Data moving

between clusters reaches computing nodes through Grid Gateway node, which is in the cluster system area network (SAN). On the gateway node, there is a Grid Gateway process responsible for forwarding data the inter-cluster network. When a computing node sends messages to a node in the remote cluster, messages are sent to the local gateway node through SAN at first. Then the gateway node transfers the messages to the remote gateway node through inter-cluster network, which forwards the messages to the destination through SAN at last.

Gateway node connects directly with inter-cluster network. If not, it still can communicate with remote clusters through access node via Ethernet. Access nodes actually act as Ethernet gateways to forward IP packets. The flexibility makes Grid Gateway independent of inter-cluster network, which means that we can connect clusters through an Intranet in a building or the Internet.

Every node can act as a gateway node as well as a computing node. A computing node requires that its tasks be assigned enough CPU time. Gateway process has blocking-receiving function and supports intra-node communication. It sleeps when it is idle, so doesn't affect the task running on the node.

There can be more than one gateway nodes in a cluster. At the extreme every computing node is a gateway node, which only transfer itself messages to remote clusters. Static strategy is used in the system to balance loads if the number of gateway nodes is more than one but less than that of computing nodes. This multi-gateway mechanism is expected to enhance the performance of inter-cluster communication when lay grid-enabled MPI implementation over it.

In the Grid, it is possible that the clusters become unavailable suddenly during the time of executing tasks. In order to prevent the local gateway process and tasks from their affect, GGW allows every cluster to join or leave the computational Grid dynamically.

3 Design Issues

Grid Gateway enables low-level communication protocol to pass messages between each other directly. The domain of the protocol is extent to include several clusters. Gateway process has complex buffer management and flow control mechanisms because of the performance gap between cluster interconnect and inter-cluster network. These factors result in several design issues in the Grid Gateway. For instance, how to design the hostname and address space that identify communication end-points for all nodes of clusters; how to cope with unavoidable semantic change caused by the external communication of low-level communication protocol, and so on.

3.1 Global ID

In a low-level communication protocol every end-point needs a unique node ID in a cluster. To identify the sender or receiver when communicating between clusters, the node ID should be unique or global in all the involved clusters. That means all the end-points of the clusters will share a hostname and address space.

In GGW, net adapter is the communication end-point. Every net adapter is assigned a global ID to identify itself. The global ID is derived from the net adapter's MAC address that can identify the net adapter globally. Gateway process knows all the node's global IDs that are gotten from the remote clusters during initialization. Every computing node has a name table containing all the nodes' global ID, which is filled by the data gotten from gateway process if it's empty when opening a port.

3.2 Communication Semantic

In the low-level communication protocol, after sending a message sender will receive a send event to notify the result of the send. A send-completed event suggests that the message get to the receiver. In Grid Gateway, the communication path is extended to include local and remote gateway nodes. If the operation of sending messages to a remote cluster conforms to the semantic of the low-level communication protocol, high latency will result in a great number of return events blocking in the event queue to waiting for ACK/NACK from receiver. At last the blocking-event number will reach the capacity of low-level communication protocol so that the node will be unable to send and receive. GGW changes the semantic of the send operation. After sending a message to a gateway successfully, sender will receive a send event notifying that the message arrives at the gateway other than the remote cluster node.

In addition, the performance gap between SAN and Ethernet makes the buffer of Grid Gateway process often full of data. In order to avoid buffer overflow, Gateway process returns send-failed event. Therefore sender probably encounters more failures when sending messages to a remote cluster than to a local cluster. But for some low-level communication protocols this effect is slight. For instance, in GM2.0 if target node has no receiving buffers available, messages stay in the send queue for about one minute until the buffers is provided. This waiting time is long enough to allow gateway process to release a few buffers and provide them for receiving messages.

3.3 Multi-routing Mechanism

Since communication involves the nodes at remote clusters, every computing node has a multi-routing mechanism. When a computing node sends a message, it judges whether the destination is in local the cluster or a remote cluster at first, then chooses a route to the nodes in local cluster, or to the local gateway if the destination is at a remote cluster. To forward the message, the local gateway also needs to choose a route to the remote gateway node for the message.

Grid Gateway uses global ID as the parameter of send and receive operations. Hostname and global ID of all the nodes are stored in a name table on every host. Routing operation on the computing node and gateway just is to retrieve the table. But the retrieving operation may be inefficient when name table is very large. We assign each cluster an ID. The name table catalogs global IDs according to the cluster IDs. The first byte of a global ID word is set as a cluster ID, and local cluster ID is zero. It is very fast to locate a node by its cluster ID.

3.4 Flow Control

For the sack of the performance gap between cluster interconnect and inter-cluster network, it's possible that the Gateway buffers are full of data and cannot receive messages. In case that buffer overflow results in data loss and out-of-order of messages, an efficient flow control mechanism is needed between computing node and gateway node.

Grid Gateway use tokens to regulate sends between computing node and gateway node. A computing node port may send a message to gateway node only if when it possesses a token for that port, and so does gateway. Receiver returns tokens after it deals with the messages.

Each send token imply these is a corresponding receiving buffer on the gateway. In fact the number of send tokens is so large that the number of buffers is far less than it. As a solution, in GGW only if it is confirmed that a send is successful, the next message can be sent.

4 Design and Implementation

Grid Gateway implementation is based on the DAWNING 4000A[4] high-performance Grid-enabling computer, which will consist of 512 SMP nodes based on the AMD 64-bit processor and connected with the 1024-port Myrinet. In current version of Grid Gateway we choose GM2.0[2] as low-level communication protocol.

4.1 Communication Path

Communication end-point is identified by global ID. All global IDs and hostnames are store in the global ID table. If it is empty gm_open() will get it from gateway. To interact with gateway node, some control messages is transferred by the single-side operation (or called RMA operation). The base address of RMA buffers is exchanged in gm_open(). When a port is closed, gm_close() must notify the gateway the state change of the port in order to stop the gateway passing any message to the port..

When a computing send a message to a remote cluster's node through gateway, gateway process should know the destination's information, such as global ID, port ID, and size etc. To tell the gateway the destination's information, we may contain it in the message body, or send an additional control message. GGW uses the latter method, because by the former method excessive data may causes receiver's buffer overflow. The additional control message is sent by a RMA operation, gm_put(), shown as dotted line in the figure 2. Since the order of the transfer of gm_put() is preserved relative to messages of the same priority sent by gm_send(), it is easy to keep the mapping between the additional control message and data message. Every opened port of each node in the local cluster has two RMA areas both on the gateway and local node.

Gm_put() writes data into a target node's buffer specified by the parameter of virtual memory address. The first message can't by sent by this means in gm_open()

because local node haven't that address. Fortunately the gateway can recognize it for it's the first message that the gateway receives from that node's port.

As shown in figure 2, we implement gm_send() and gm_put() functions mainly used at first. In gm_put() we use gm_send() to pass data to the gateway. If we use gm_put() to do so, to notify the gateway process the data's arrive will complicate the gateway process. The type of messages is implied by the additional RMA control message. When receiving a message from GM, Gateway process deals with it according to its type. The message is packed with gateway protocol form and then sent to other gateway. When receiving a message from other gateway, the message will be sent to target node by gm_send() or gm_put() according its type. For messages from the gateway gm_receive() fills the additional message's information into the receive event.

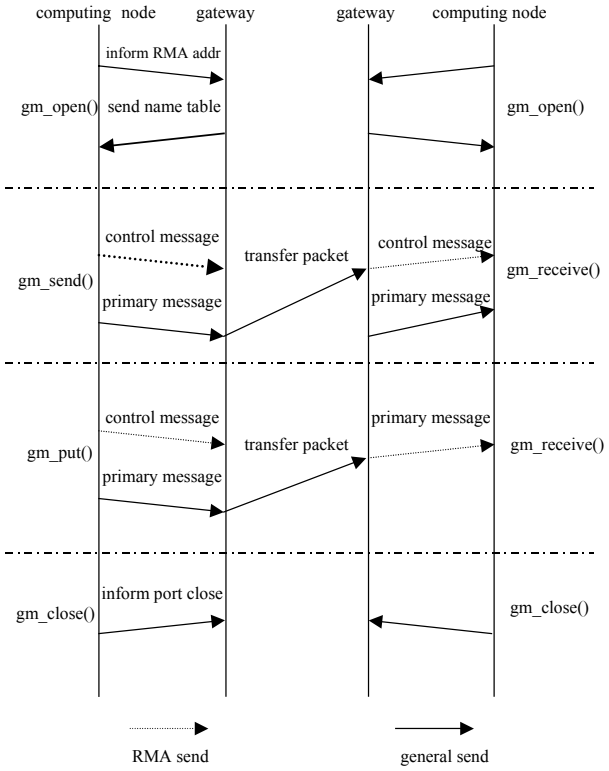

Fig. 2. Message Sequence in the Functions

4.2 Gateway Process

Gateway process maintains connections with other gateway, and receives messages into local cluster's nodes or sends out to remote clusters. We use two threads to deal

with the receive-in and send-out operation respectively. Gateway process has to be able to sleep to wait for messages from GM and sockets simultaneously; otherwise busy waiting will waste much of CPU time so that gateway node can't run as computing node. So only one single process isn't suitable. Multi-process is capable of realizing gateway's functions, but to share complicated buffers and data queues between processes is challenging. Based on the above analysis multi-thread structure is an ideal choice. Gm_blocking_receive() and socket select() calls enable gateway process to sleeping receiving in different threads; and careful dividing data structure to diminish shared data and locks may make program very simple but effective.

So gateway process consists of three parts: main procedure and receive-in thread and send-out thread. Main procedure collects information of local cluster and remote, initiate those two threads, and bock itself to accept new clusters' connection requests. Receive-in thread is in charge of the responsibility shown at the upper part of Figure 3. It repeatedly receives data from TCP/IP, enqueue it into receiving FIFO, and send it to its destination through GM. Send-out thread is responsible for Figure 3's lower part, and do at the same steps with receive-in thread but in the contrary direction.

The two threads don't share buffers so that it is not necessary to consider mutex locks. Both threads have message FIFO organized under the consideration for picking out packets efficiently. Sending FIFO is a two-dimensional array, with item divided by its target gateway. Receive-in thread's FIFO is a list, because target port number is very huge that array structure may be inefficient. Sending a message to a specific gateway or nodes' GM ports may be not permitted when the socket's buffer is full or GM port's buffer is unavailable. To pick out another item to send can speed up proceeding instead of to block at the point. Logically GM2.0 can transfer 2^{31} bytes one time maximally, but MPICH-GM divides long messages into several fragments with 1M bytes length each at most. So a buffer block about 1M bytes length is enough.

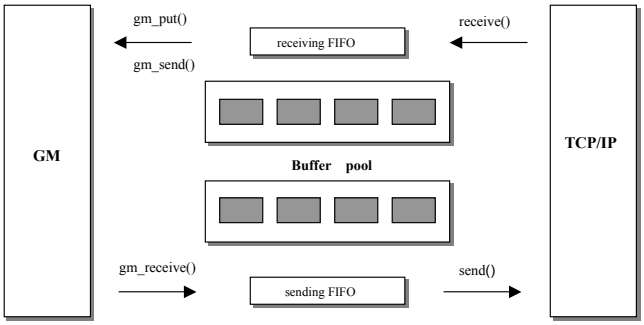

Fig. 3. Gateway Process Structure

5 Summary

Grid Gateway connects separated low-level communication protocols, and enables messages to move between distributed cluster interconnects. This paper describes its architecture and implementation. Several design issues are presented, including global ID, message passing semantic, multi-layer routing mechanism, and flow control. Now the project is under way. We expect to present its performance evaluation later.

GGW can be applied to different structures of inter-cluster networks flexibly. It allows clusters to be inter-connected through the Intranet or the Internet. A node can be a computing node as well as a gateway node, which suggests that the number of gateway node can be as many as that of the computing nodes. GGW extends the functionality of low-level communication protocol so that it might facilitate the implementation of Grid-enabled tools.

References

[1] I. Foster, C. Kesselman, S. Tuecke: The Anatomy of the Gird: Enabling Scalable Virtual Organizations. International Journal of Supercomputer Applications, 2001, 15(3): 200-222.
[2] Myricom, Inc: GM Reference Manual 2.0.3, http://www.myri.com.
[3] I. Foster, N. Karonis: A Grid-enabled MPI: Message Passing in Heterogeneous Distributed Computing System, Proc. 1998 SC Conference.
[4] Zhiwei Xu, Ninghui Sun, Dan Meng, etc: Cluster and Grid Superservers: The Dawning Experience in China, Proc of the 3rd IEEE Int'l Conf on Cluster Computing, 2003.
[5] R. A Bhoedjang: User-level Network Interface Protocols, IEEE Computer, November 1998.
[6] N. Karonis, B. Toonen, I. Foster: MPICH-G2: A Grid-enabled Implementation of the Message Passing Interface, Journal of Parallel and Distributed Computing, 2003, 63(5): 551-563.
[7] T. Imamura, Y. Tsujita, H. Koide, H. Takemiya: An Architecture of Stampi: MPI Library on a Cluster of Parallel Computers, LNCS Vol. 1908, 200-207, Springer, 2000.
[8] M. Muller, M. Hess, E. Gabriel: Grid enabled MPI solutions for Clusters, The 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid, 2003.
[9] E. Gabriel, M.Resch, T. Beisel, R. Keller: Distributed computing in a Heterogeneous Computing Environment, LNCS Vol. 1497, 180-187, Springer, 1998
[10] I foster, C Kesselman: The Grid: Blueprint for a New Computing Infrastructure, Morgan Kaufmann Publishers, 1998
[11] Zhiwei Xu, Wei Li: Research on VEGA Grid Architecture, Journal of Computer Research and Development, 2002, 39(8)
[12] V. G. Cerf, R .E. Kahn: A Protocol for Packet Network Intercommunicaiton, IEEE Trans on Comms, Vol Com-22, No5, 1974
[13] D. E. Comer, D. L. Stevens: Internetworking With TCP/IP Vol I: Principles, Protocols, and Architecture, 3rd edition, Prentice-Hall, 1995