Distributed Peer-to-Peer Control in Harness *

C. Engelmann, S. L. Scott, G. A. Geist

Computer Science and Mathematics Division,
Oak Ridge National Laboratory, Oak Ridge, TN 37831-6367, USA
{engelmannc, scottsl, gst}@ornl.gov
http://www.csm.ornl.gov/dc.html

Abstract. Harness is an adaptable fault-tolerant virtual machine environment for next-generation heterogeneous distributed computing developed as a follow on to PVM. It additionally enables the assembly of applications from plug-ins and provides fault-tolerance. This work describes the distributed control, which manages global state replication to ensure a high-availability of service. Group communication services achieve an agreement on an initial global state and a linear history of global state changes at all members of the distributed virtual machine. This global state is replicated to all members to easily recover from single, multiple and cascaded faults. A peer-to-peer ring network architecture and tunable multi-point failure conditions provide heterogeneity and scalability. Finally, the integration of the distributed control into the multi-threaded kernel architecture of Harness offers a fault-tolerant global state database service for plug-ins and applications.

1 Introduction

Parallel and scientific computing is a field of increasing importance in many different research areas, such as fusion energy, material science, climate modeling and human gnome research. Cost-effective, flexible and efficient simulations of real-world problems performed by parallel applications based on mathematical models are replacing the traditional experimental research.

Software solutions, like Parallel Virtual Machine (PVM) [4] and Message Passing Interface (MPI) [5], offer a wide variety of functions to allow messaging, task control and event handling. However, they have limitations in handling faults and failures, in utilizing heterogeneous and dynamically changing communication structures, and in enabling migrating or cooperative applications.

The current research in heterogeneous adaptable reconfigurable networked systems (Harness) [1]-[3] aims to produce the next generation of software solutions for parallel computing. A high-available and light-weighted distributed

* This research was supported in part by an appointment to the ORNL Postmasters Research Participation Program which is sponsored by Oak Ridge National Laboratory and administered jointly by Oak Ridge National Laboratory and by the Oak Ridge Institute for Science and Education under contract numbers DE-AC05-84OR21400 and DE-AC05-76OR00033, respectively.

P.M.A. Sloot et al. (Eds.): ICCS 2002, LNCS 2330, pp. 720–728, 2002.

virtual machine (DVM) provides an encapsulation of a few hundred to a few thousand physical machines in one virtual heterogeneous machine.

A high availability of a service can be achieved by replication of the service state on multiple server processes at different physical machines [6]-[12]. If one or more server processes fails, the surviving ones continue to provide the service because they know the state. The Harness DVM is conceived as such a high-available service. It will survive until at least one server process is still alive.

Since every member (server process) of a DVM is part of the global state of the DVM service and is able to change it, a distributed control is needed to replicate the global state of the DVM service and to maintain its consistency. This distributed control manages state changes, state replication and detection of and recovery from faults and failures of members.

2 Distributed Peer-to-Peer Control

The distributed control symmetrically replicates the global state of the DVM, which consists of the local states of all members, to all members. A global state change is performed using a transaction, which modifies the local state of one or more members depending on their local and/or the global state during execution. These local state modifications are replicated to all members.

Such a transaction may join or split member groups, add or remove a member, start or shutdown a service at one member or at a member group, change states of services or report state changes of services. The global state is actively replicated (hot-standby) only at a subset of the DVM members, while all other members retrieve a copy of the global state only on demand.

The tradeoff between performance and reliability is adjustable by choosing the number and location of members with active replication. The number of hot-standby members affects the overall DVM survivability and robustness in terms of acceptable fault and recovery rate. The classic client-server model with its single point of failure is realized if there is only one.

The location of hot-standby members additionally addresses network bottlenecks and partitioning. If different research facilities collaborate using a large DVM, every remote site may choose its own hot-standby members. Additionally, any member can become a hot-standby member if a faulty hot-standby member needs to be replaced or the DVM configuration changes.

The hot-standby member group consistently maintains the replicated global state using group communication in an asynchronous connected unidirectional peer-to-peer ring network architecture. The members of this group are connected in a ring of TCP/IP peer-to-peer connections. Messages are transmitted in only one direction and received, processed and sent asynchronously.

3 Group Communication

The distributed control uses group communication services, such as Atomic Broadcast and Distributed Agreement, to simplify the maintenance of consis-

tently replicated state despite random communication delays, failures and recoveries. These services are based on reliably broadcasting messages to all members replicating the state in the hot-standby member group.

3.1 Reliable Broadcast

The Reliable Broadcast is the most basic group communication service. It ensures a message delivery from one member to all members in a group, so that if process p transmits a message m, then m is delivered by all non-faulty processes and all non-faulty processes are notified of any failures.

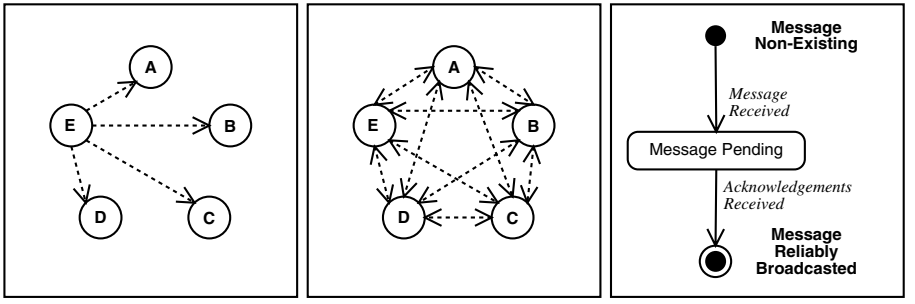

Fig. 1. Reliable Broadcast (Phase I, Phase II, State-Machine)

In order to reliably broadcast a message, one member broadcasts it to all members. Each member who receives it broadcasts an acknowledgement back to all members. A member knows that a message is reliably broadcasted if message and related acknowledgements of all members are received. The Reliable Broadcast performs two phases of broadcast communication.

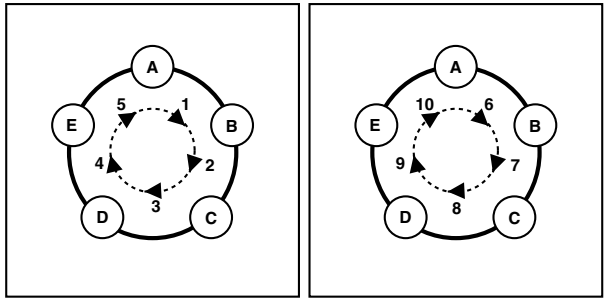

Fig. 2. Reliable Broadcast in a unidirectional Ring (Phase I, Phase II)

A straightforward implementation of this Reliable Broadcast algorithm results in high communication costs (n broadcasts) and is not scalable. What fol-

lows is a description of a much more efficient algorithm based on a unidirectional ring network which is used by the distributed control in Harness.

In a unidirectional ring, messages are forwarded from one member to the next to reliably broadcast a message. In phase one, a member sends a message to its neighbor in the ring until a member who has it already received before receives it. In phase two, a member sends the acknowledgement to its neighbor in the ring until a member who has it already received before receives it.

3.2 Atomic Broadcast

The Atomic Broadcast is an extension of the Reliable Broadcast. It additionally ensures a globally unique order of reliably broadcasted messages at all members in a group, so that if process p transmits a message *m1* followed by a message *m2*, then *m1* is delivered by all non-faulty processes before *m2* and all non-faulty processes are notified of any failures.

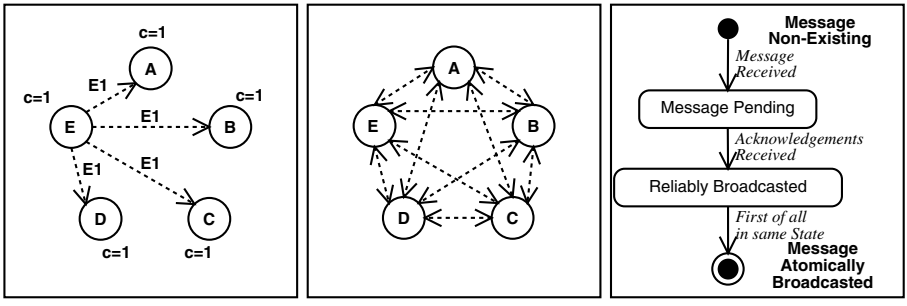

Fig. 3. Atomic Broadcast (Phase I, Phase II, State-Machine)

In order to atomically broadcast a message, the Reliable Broadcast is extended with a message numbering and sorting algorithm.

A global message number is maintained with the message broadcast, so that members cannot reuse it immediately. Every member has a local copy of a global message number counter, which is increased before every Atomic Broadcast and constantly updated with higher received values.

All simultaneously broadcasted messages have the same message number and all sequentially broadcasted messages have increasing message numbers. A static and globally unique member priority, which is assigned by the group during member admission, orders messages with equal numbers.

A message is broadcasted with its globally unique id (number/priority) and sorted into a list on receiving, so that it is atomically broadcasted if all previous messages are reliably broadcasted. One atomically broadcasted message may cause other reliably broadcasted messages to be in the atomically broadcasted state, so that blocks of messages may occur to be atomically broadcasted.

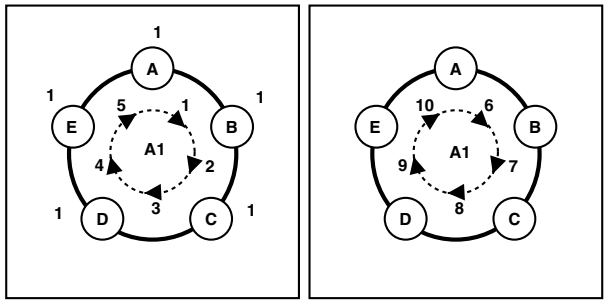

Fig. 4. Atomic Broadcast in a unidirectional Ring (Phase I, Phase II)

In a unidirectional ring, the message id is passed with the message in phase one from one member to the next until received twice. The message id is also passed with the acknowledgement in phase two from one member to the next until received twice. All messages with the same or a lower message number are known in phase two, so that the total order of all previously and all simultaneously broadcasted messages is known.

3.3 Distributed Agreement

The Distributed Agreement group communication service is based on multiple Reliable Broadcasts executed by all members in a group. It ensures that a decision is taken at all members by all members, so that each member knows the decision of every member and is able to evaluate it using the same voting algorithm.

In order to perform a Distributed Agreement on a decision, every member executes a Reliable Broadcast of its decision. A decision is taken at a member by all members if the decisions of all members are reliably broadcasted. Common voting algorithms are: majority, minority (at least one) and unanimous agreement.

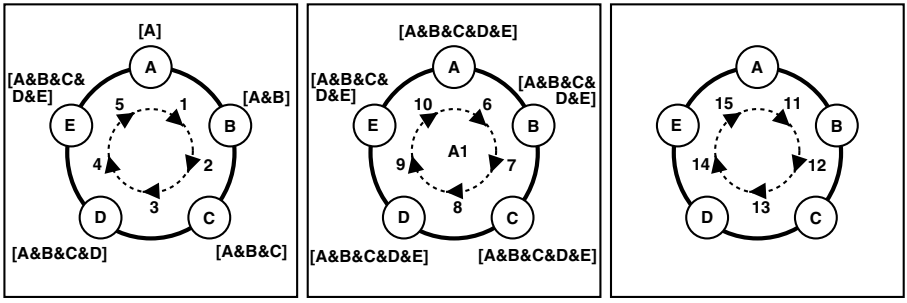

Fig. 5. Distributed Agreement in a unidirectional Ring (Phase I, Phase II, Phase III)

The Distributed Agreement in a unidirectional ring is based on interleaved Reliable Broadcasts. One message is passed around the ring twice to reliably broadcast information appended by all members in the first round. Collective communication is used to avoid n Reliable Broadcasts or n messages.

Passing a message from one member to the next until received twice collects the decisions of all members in phase one. Passing this message around the ring again in phase two broadcasts the final decision and ensures the decision collection of phase one. Passing an acknowledgement around the ring in phase three ensures the final decision broadcast of phase two.

The decisions of all members are collected in phase one using a counter for every class of decision. Their final count is broadcasted in phase two. The voting algorithm compares these values at every member in phase three. This anonymous voting may be replaced by collecting the member names for every class of decision or by a mixture of both.

4 Member Fault Recovery

A single, cascade or multiple member fault does not affect group communication services directly. However, they are based on acknowledgements for received messages from all members and the membership changes when a fault occurs. The underlying communication system notifies all indirectly affected members who are waiting for acknowledgements from faulty members.

Messages may get lost if a faulty member was used as a router. Since all messages are stored until acknowledged, they are sent again if a fault occurs to ensure delivery of messages at least once. Doubled messages are filtered at the receiving member based on the group communication service state.

Single or multiple faults in a ring are recovered by connecting the two correct neighbor members of the faulty member(s) to each other. The neighbor who had sent messages and acknowledgements to a faulty member assumes that all unacknowledged messages and all acknowledgements are lost.

All unacknowledged messages are sent again in the same order. A hop counter contained in every message which is increased when passed to the next member is used to distinguish doubled messages from messages seen twice. A message is ignored if the hop count is less than the stored one.

The last acknowledgement is always stored at a member and sent again. The receiving member regenerates all lost acknowledgements based on his last stored and based on the order the related messages passed previously.

Cascaded faults, which are single or multiple faults occurring during a recovery from previous ones, are handled by restarting the recovery.

5 Membership

The group membership requires an agreement on an initial group state and a linear history of group state changes at all members, so that each member has a complete and consistent replicated copy of the group state.

In order to achieve an agreement on an initial group state, every new member obtains a copy of the group state from a group member. In order to achieve an agreement on a linear history of state changes, every state change is performed as a transaction using an Atomic Broadcast of the state change followed by a Distributed Agreement on the state change execution result.

Transactions consistently replicate the list of members as part of the group state while admitting or removing members. In order to admit a new member, a member of the group connects to the new member and updates it with a copy of the group state. A member removal is based on a controlled member fault. In order to remove a member, the member to be removed disconnects itself from the group. All members expect the member fault and recover easily.

6 Transaction Types

The distributed control extends the management of a symmetrically distributed global state database by enabling transactions to be partially or completely committed or rejected depending on their execution result, which may be different at every member. Transactions may involve local state changes that may fail at one or more members, local state changes at one or more members that cannot fail and global state changes that may fail at all members.

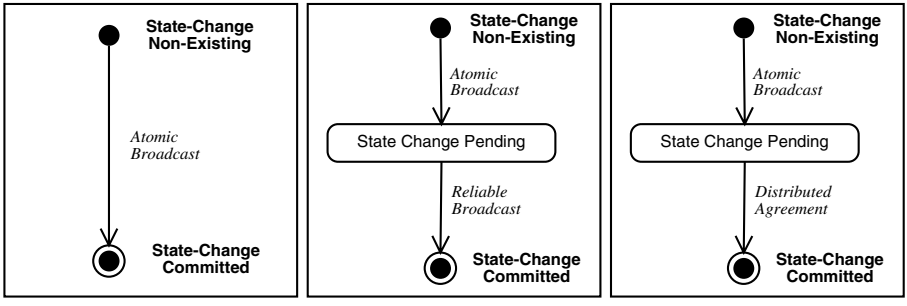

Fig. 6. Transaction Control State-Machines (Type I, II, III)

A linear history of state changes is realized using an Atomic Broadcast of a state change followed by a Distributed Agreement on the execution result, since a state change may involve multiple local state change failures (Type III).

A Reliable Broadcast of the execution result replaces the Distributed Agreement if only one local state change that may fail is involved (Type II). The Atomic Broadcast alone is performed if a transaction involves only local state changes that cannot fail or global state changes that may fail (Type I).

A transaction control algorithm decides the transaction type on a case-by-case basis and guarantees the total order of state change executions and state change commits, so that all state changes are executed and committed/rejected in the same order at all members.

7 Harness Kernel Integration

A first prototype was developed as a distributed stand-alone server application in C on Linux to test and experiment with different variations of the described group communication algorithms. The results were used to refine the algorithms and to define the requirements for the integration of the distributed control into the multi-threaded Harness kernel architecture.

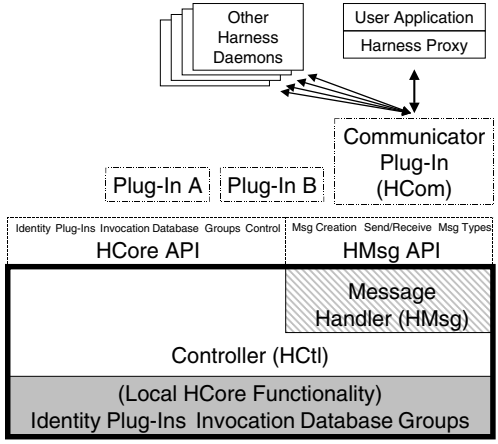

Fig. 7. Scheme of Harness Kernel

The transaction control algorithm manages all transactions in the controller module (HCtl). The central message handler (HMsg) delivers all messages to and from the controller. Communicator plug-ins (HCom) maintain the network connections to communicator plug-ins of other kernels.

Group communication messages are exchanged between a communicator plug-in and the controller using the message handler. State changes are executed by the controller or by a plug-in, while the controller maintains the state database. State change requests are delivered from a plug-in to the controller via the message handler or directly using the kernel API.

8 Conclusions

We have developed a distributed control that automatically detects and recovers single, multiple and cascaded member faults. Its design allows a trade-off between performance and robustness, i.e. tunable multi-point failure conditions. A strictly symmetric global state control and replication ensures fault-tolerance and protects members against denial-of-service and starvation.

The distributed control uses group communication services, such as Reliable Broadcast, Atomic Broadcast and Distributed Agreement, to simplify the maintenance of consistently replicated state despite random communication delays,

failures and recoveries. These algorithms were adapted to a peer-to-peer ring network architecture to improve scalability and heterogeneity.

Global state changes are transactions that are executed at all members or at a member group and consistently committed or rejected at the state database of all members depending on the execution results of all members. The transaction control algorithm classifies transactions into different types and chooses the most efficient combination of group communication services.

The integration into the multi-threaded Harness kernel architecture distributes the network communication, the message handling and the transaction control algorithm over different modules. The plug-in concept enables the distributed control to use different network protocols for every communication link, while providing its global database service to plug-ins and applications.

References

1. C. Engelmann: Distributed Peer-to-Peer Control for Harness. Master Thesis, School of Computer Science, The University of Reading, UK, Jan. 2001
2. W.R. Elwasif, D.E. Bernholdt, J.A. Kohl, G.A. Geist: An Architecture for a Multi-Threaded Harness Kernel. Computer Science and Mathematics Division, Oak Ridge National Laboratory, USA, 2001
3. G.A. Geist, J.A. Kohl, S.L. Scott, P.M. Papadopoulos: HARNESS: Adaptable Virtual Machine Environment For Heterogeneous Clusters. Parallel Processing Letters, Vol. 9, No. 2, (1999), pp 253-273
4. G.A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, V. Sunderam: PVM: Parallel Virtual Machine; A User's Guide and Tutorial for Networked Parallel Computing. MIT Press, Cambridge, MA, 1994
5. M. Snir, S. Otto, S. Huss-Lederman, D. Walker, J. Dongarra: MPI: The Complete Reference. MIT Press, Cambridge, MA, 1996
6. F. Cristian: Synchronous and Asynchronous Group Communication. Communications of the ACM, Vol. 39, No. 4, April 1996, pp 88-97
7. D. Dolev, D. Malki: The Transis Approach to High Availability Cluster Communication. Communications of the ACM, Vol. 39, No. 4, April 1996, pp 64-70
8. E. Moser, P. M. Melliar-Smith, D. A. Agarwal, R. K. Budhia and C. A. Lingley-Papadopoulos: Totem: a fault-tolerant multicast group communication system. Communications of the ACM 39, 4 (Apr. 1996), pp 54-63
9. F.C. Gaertner: Fundamentals of Fault-Tolerant Distributed Computing in Asynchronous Environments. ACM Computing Surveys, Vol. 31, No. 1, March 1999
10. S. Mishra, Lei Wu: An Evaluation of Flow Control in Group Communication. IEEE/ACM Transactions on Networking, Vol. 6, No. 5, Oct. 1998
11. T.D. Chandra, S. Toueg: Unreliable Failure Detectors for Reliable Distributed Systems. I.B.M Thomas J. Watson Research Center, Hawthorne, New York and Department of Computer Science, Cornell University, Ithaca, New York 14853, USA, 1991
12. M. Patino-Martinez, R. Jiminez-Peris, B. Kemme, G. Alonso: Scalable Replication in Database Clusters. Technical University of Madrid, Facultad de Informatica, Boadilla del Monte, Madrid, Spain and Swiss Federal Institute of Technology (ETHZ), Department of Computer Science, Zuerich