Practical Cluster Applications of Group Communication

Yair Amir, Yan Gu, Theo Schlossnagle, Jonathan Stanton
Department of Computer Science
The Johns Hopkins University
Baltimore, Maryland 21218 USA
{yairamir, yangu, theos, jonathan}@cnds.jhu.edu

Abstract

Group communication systems are used to build fault-tolerant and replicated services. However, they also can be very useful for building tools that support a highly available web cluster, or help manage a large collection of machines. This paper explores the features of one particular Group communication system called Spread^{TM}*that are useful in this environment and describes two specific applications that we built and actively use. These applications are a distributed system logging tool and an IP based high availability fail-over tool for clusters.*

1. Introduction

Group communication has been used for many years in high-availability application domains such as stock markets and command-and-control systems. However, despite the usefulness of high-performance group communications systems, they have not been widely used to build highly available and scalable cluster or system management tools.

The Spread group communication toolkit provides high performance and strong semantics. Spread provides all the traditional group communication services, as well as some unusual properties such as open-groups, inter-group message ordering, and good scalability.

In this short paper we will discuss two applications that we built using Spread as a group communication service which take advantage of the features built into Spread.

The first application is a distributed logging tool, implemented using the standard Unix syslog service. This tool allows multiple machines and processes to monitor the logs in real-time and simplifies creating replicas of logs either in whole or in part.

The second application is Wack-a-mole, a high-availability tool for clusters of servers. Wack-a-mole, like the annoying carnival game, ensures that a server handles the requests that arrive on any of the IP addresses you make publicly known, even if not all the actual servers are available. This goes significantly beyond the fail-over "Virtual Server" software commonly available since it allows lots of machines (not just 2) to dynamically reconfigure themselves to cover all the public IP addresses.

The Spread toolkit is available publicly. More details on the Spread system can be found at http://www.spread.org/.

2. Required Group Communication Properties

We list several important group communication properties required by cluster applications.

- *Scalability with the number of groups.* Spread can scale well with the number of groups used by the application, up to several thousand. Group naming and addressing is more flexible than IP-multicast (which uses IP multicast addresses) because Spread uses a large space of strings.
- *Open-group semantics.* This allows efficient implementation of the publish-subscribe model.
- *Membership Service with Strong Semantics.* The membership service provides a current list of alive members. The strong semantics guarantees a consistent state among all the connected members.
- *Efficient for Small Messages.* The common message size is very small (10-250 bytes) for these applications.
- *Low overhead.* Minimal use of network and CPU resources when message load is light.

The ability to support a large number of groups is especially useful for the logging application. Each service or application that wants to generate log messages might want its own group. Complex services such as web servers may need a number of groups for different types of data. In many practical cases each web server serves multiple domains, up to several thousand, multiplying the number of groups needed.

Traditionally group communication services support closed-group semantics in which only members of the group can send messages to the group. An alternative model, called open-group semantics, allows anyone to send to a group, even if they are not a member of that group. In cluster applications the senders are often distinct from the receiver processes (closer to a publish-subscribe model) and thus closed-groups are more complex and less natural to use than open-groups, and have an efficiency penalty.

3. Distributed Logging

Logging is necessary for effective systems administration. With today's networked clusters, dealing with individual log files on each machine is both ineffective and inconvenient. It is nearly impossible to determine cause and effect across machines. Merging large log files, which is required for most log analysis, is difficult, and monitoring systems are usually so complex that they themselves break.

A common system logging tool is "syslog" which is standard on most Unix machines. Syslog is a general logging interface used by many programs, ranging from the kernel, to ftp and network servers, to user applications. Currently, the syslog daemon (syslogd) logs each message to one of: a file on local disk, a pipe, or another syslogd process located remotely. The remote logging option is usually unreliable.

We modified syslogd to add a new logging destination–a Spread group. The standard syslogd configuration file defines which Spread group each type of event should be logged to. Additionally, a second program was created that reads the groups to monitor from a separate configuration file, and writes the received events to a file or the screen.

4. Dynamic Cluster N-way Fail-over

Clustering of servers, such as web and email, is a very common solution to high throughput requirements. Two common methods exist to make such a cluster "Highly Available": a gateway solution that hides all the actual servers behind a very smart switch or router, or a flat solution in which all the servers are visible to the clients and redundancy is built into the servers themselves.

A gateway solution has several advantages: it is available commercially today and is architecturally simple. It also has several significant disadvantages such as a high cost, a single point of failure, an inability to detect failed processes as opposed to failed links, and a general inflexibility making it hard to adapt. The single point of failure can be alleviated by doubling the infrastructure (which doubles the price).

Existing flat solutions can only provide pair-wise failover detection and masquerading. If both machines that are paired together fail, then this failure will not be masked and will be visible to the clients.

The main idea behind Wack-a-mole is to use the membership service built into Spread to generate an agreed-upon state of the existing servers. The properties of the membership service[5] guarantee that the decision made by all the members will be identical, consistent, and will have exactly one member covering every public IP address. As a result, Wack-a-mole can sustain any combination of any number of failures up to n-1.

One could think of building a distributed solution with "almost correct" consistency. However, an error may have significant costs. If an IP address is not assigned to a server, then obviously all access to that IP address will go unanswered. If an IP address is assigned to more than one server, then clients will experience failures such as TCP resets caused by arp contention.

5. Related Work

The only other group communication system that we know, which is commonly used and available for public use, is the Ensemble[3] system developed at Cornell University. Reliable IP-Multicast could be used to support a distributed logging tool. One reliable multicast protocol which has been developed is SRM[2].

Several implementations[4] of 2-way failover using IP address capture exist. Gateway solutions are available from a number of vendors such as Cisco, Alteon, Extreme Networks, Foundry and Arrowpoint, as well as an open source solution–the Linux Virtual Server. Some earlier work on distributed system management using group communication[1] showed how to use it in similar contexts.

References

[1] Y. Amir, D. Breitgand, G. V. Chockler, and D. Dolev. Group communication as an infrastructure for distributed system management. In *Proc. of the Int. Workshop on Services in Distr. and Networked Env. (SDNE)*, pages 84–91, June 1996.

[2] S. Floyd, V. Jacobson, C. Liu, S. McCanne, and L. Zhang. A reliable multicast framework for light-weight sessions and application level framing. *IEEE/ACM Transactions on Networking*, 5(6):784–803, December 1997.

[3] M. Hayden. *The Ensemble System*. PhD thesis, Cornell University, Department of Computer Science, 1998.

[4] S. Horman. Creating redundant linux servers. In *Procedings of the 4th Annual Linux Expo, Durham NC*, May 1998.

[5] L. E. Moser, Y. Amir, P. M. Melliar-Smith, and D. A. Agarwal. Extended virtual synchrony. In *Proceedings of the IEEE 14th International Conference on Distributed Computing Systems*, pages 56–65, June 1994.