HIGH AVAILABILITY

Ira Pramanick
SUN MICROSYSTEMS

The International Journal of High Performance Computing Applications, Volume 15, No. 2, Summer 2001, pp. 169-174

1 Introduction

System availability has traditionally been a requirement for mission-critical applications. Nowadays, it is also becoming increasingly important in most commercial and many academic arenas where the goal of providing minimal downtime to users is receiving significant attention. System downtimes can be for scheduled maintenance or due to the occurrence of a failure of the components in the system.

Broadly speaking, there are three flavors of system availability: continuous availability, fault tolerance, and high availability. Continuous availability implies nonstop service, representing an ideal state. To achieve continuous availability, the system would need to be made up of perfect components that could never fail, either in hardware or in software. Hence, by definition, no actual systems can provide this type of availability.

Fault tolerance (Jalote, 1998) masks the presence of faults in a system by employing redundancy in hardware, software, and/or time. Hardware redundancy consists of adding replicated custom hardware components to the system to mask hardware faults. Software redundancy includes the management of redundant hardware components and ensuring their correct use in the event of failures in the system. Time redundancy refers to the repetition of the execution of a set of instructions to ensure correct behavior even if a failure occurs.

High availability (HA) (http://www.dhbrown.com; http://www.imexresearch.com) provides a cheaper alternative to fault tolerance. Here, the system uses off-the-shelf hardware components in a redundant fashion together with software solutions that mask system failures to provide uninterrupted services to a user. Typically, highly available solutions guarantee survival from single points of failure in the system. An HA system is usually a cluster (Pfister, 1998) of machines or computer nodes.

This article focuses on HA in cluster computing (Buyya, 1999a, 1999b). It starts with an overview of the main features of a typical HA solution. This is followed by a brief description of the salient features of several HA research projects and then by a listing of several current commercial HA solutions. The next section discusses the main issues that come up when designing and using an HA system.

2 Background and Overview

A typical HA solution consists of software modules that run on a cluster consisting of two or more off-the-shelf computer nodes that together form the HA system. Clus-

"System availability has traditionally been a requirement for mission-critical applications. Nowadays, it is also becoming increasingly important in most commercial and many academic arenas where the goal of providing minimal downtime to users is receiving significant attention."

ter applications or services run on a subset of the cluster nodes where configuration parameters of the application as well as that of the cluster determine the nodes in this subset. At one end of the spectrum are active-standby configurations where the applications run on some of the nodes (the active nodes), with the remaining nodes acting as redundant backups for those services. At the other end of the spectrum are active-active configurations where each node acts as an active server for one or more applications and potentially serves as a backup for applications that are running on the other nodes. When a node fails in the system, applications running on that node are migrated to one of their configured backup nodes. Under such circumstances, these backup nodes may get overloaded with applications, leading to suboptimal performance. This is generally acceptable, since degradation in performance is preferable to the total unavailability of the service in question.

An HA system can also be a single-node system. HA in a single-node scenario refers to providing uninterrupted services in the wake of intranode component failures. Just as in the general-cluster HA case, software techniques are used in conjunction with hardware redundancy of intranode components to mask these component failures. Masking the failure of a network adapter is an instance of providing intranode HA.

Thus, HA and clustering are closely related. Except for intranode availability that applies to a single computer node as well as to nodes of a cluster, HA of services directly implies a clustering solution. The basic feature of an HA system is that the failure of a node results in all HA applications that were running on that node being migrated or *failed over* to another node in the system. Similarly, a failure of other components in the system results in appropriate fail-overs for that resource, provided the system is configured to have redundancy in that resource and there is support in the HA software for masking the failure of that resource. In addition to this automatic fail-over in the event of a component failure, HA systems also typically provide the user with commands to manually migrate services to another component. This is commonly referred to as a *switch-over.*

Current HA solutions range from clusters consisting of two to eight nodes, although a few commercial products are advertised to work on much larger clusters. Because the target market for most HA applications involves nodes that are servers, the need for much larger than 32-node clusters for HA is not deemed very important currently. This has an implicit impact on the choice of underlying algorithms

used in an HA solution where scalability is not the main driving force. This is in contrast to algorithms used in high performance computing (HPC), which assumes that a cluster will consist of a large number of nodes. Recent market trends indicate that even for the HA arena, there is now a demand for application availability on large clusters consisting of small nodes. This is leading to a greater emphasis on scalability during the design of HA solutions.

Broadly speaking, an HA solution consists of two parts: an HA infrastructure and HA services. The HA infrastructure consists of software components that cooperate with each other and enable the cluster to appear as a single system to a user. Their functions include monitoring cluster nodes, monitoring cluster-related processes on the cluster nodes, controlling access to shared cluster resources, and enforcing resource constraints both to ensure data integrity and to satisfy user requirements. The infrastructure needs to provide these functions when the cluster is in a steady state and, more important, while nodes in the cluster are going up and down.

The HA services are clients of the HA infrastructure and use the facilities exported by the latter to provide semiseamless service availability to users. There is usually performance degradation when a system resource such as a cluster node fails and a service is failed over to another resource in the system, but there is no service interruption per se. There is a plethora of HA applications available on various HA products in the marketplace. Common examples include the various databases, file systems, mail servers, and web servers. Many of these applications are prepackaged with the HA product itself, while the others are supplied by various application vendors.

An application that executes on a single node needs to be made cluster-aware so that it can be run as an HA service. This typically involves adding wrapper scripts around the application in question such that it fits the basic HA framework (i.e., it can be started and stopped on any cluster node and can be monitored for liveness on any cluster node). This conversion of an application into an HA version is done via the HA Application Programming Interfaces (HA APIs). Currently, there is no industry-wide standard for HA APIs, and each HA product comes with its own version. All the APIs contain methods that will enable an application to switch over or fail over as the case may be, and these include methods to start an application, stop an application, and monitor an application-specific resource set, to mention a few.

3 Research Projects

There have been several notable research projects on fault tolerance and availability. This section discusses a representative sample of these research efforts, which are as follows:

1. ISIS Project (Birman and Cooper, 1991)
2. Horus Project (van Renesse, Birman, and Maffeis, 1996)
3. Solaris MC Project (Khalidi et al., 1996)
4. High Availability Linux Project (http://linux-ha.org)
5. Linux Virtual Server Project (Zhang, 2000)

ISIS and its successor, Horus, were a result of research work done at Cornell University on fault tolerance in distributed systems. Both systems implemented a collection of techniques for building software for distributed systems that performed well, was robust despite both hardware and software crashes, and exploited parallelism. Both systems provided a toolkit mechanism for distributed programming, whereby a distributed system was built by interconnecting fairly conventional nondistributed programs, using tools drawn from the kit. They included tools for managing replicated data, synchronizing distributed computations, automating recovery, and dynamically reconfiguring a system to accommodate changing workloads.

ISIS became very successful with several companies and universities employing the toolkit in settings ranging from financial trading floors to telecommunications switching systems. It was moved from Cornell University to Isis Distributed Systems, a subsidiary of Stratus Computer, Inc., in 1993, where it was sold as a product for 5 years.

The Horus Project was originally launched as an effort to redesign the Isis group communication system. It, however, evolved into a general-purpose communication architecture with advanced support for the development of robust distributed systems in settings for which Isis was unsuitable, such as applications that have special security or real-time requirements. Besides the practical uses of this software, the project contributed toward the theory of virtual synchrony, a runtime model used for some implementations of data replication and fault tolerance. Horus was also much faster and lighter than the Isis system.

Solaris MC was a prototype distributed operating system designed for a cluster of computer nodes that provided a single system image (SSI) such that users and applications could treat the cluster as a single computer running the Solaris operating system. It was implemented as a set of extensions to the base Solaris UNIX system and provided the same ABI/API as Solaris, thereby running unmodified applications. The components of Solaris MC were implemented in C++ through a CORBA-compliant object-oriented system with all new services defined by the Interface Definition Language (IDL). Objects communicated through a runtime system that borrowed from Solaris doors and Spring subcontracts. Solaris MC was designed for HA: if a node failed, the remaining nodes remained operational. It had a distributed caching file system with UNIX consistency semantics based on the Spring virtual memory and file system architecture. Process operations were extended across the cluster, including remote process execution and a global `/proc` file system. The external network was transparently accessible from any node in the cluster. The Solaris MC project has been the basis of Sun Microsystems' next-generation Sun Cluster product line.

The High Availability Linux Project and Linux Virtual Server are both HA solutions for the Linux operating system, built via a community development effort. Both these groups are beginning to collaborate their efforts, and although these HA solutions are currently not at par with the commercial products as far as features go, they are rapidly moving toward that goal. The HA Linux Project also includes efforts to port some commercial HA products to Linux, a notable example of this being SGI's FailSafe product. The Linux Virtual Server is a highly scalable and highly available server built on a cluster of real servers. The architecture of the cluster is transparent to end users, and the users see only a single virtual server. It uses various forms of Internet Protocol–level load balancing and is advertised to work for up to 100 servers. It is released under a GNU license.

4 Commercial Products

There are several commercial HA solutions available in the marketplace today. Each HA product comes with its set of features, details of which can be found at the corresponding vendor's web site. These points of reference will also have the latest publicly available information regarding the products. The reader is encouraged to visit these web sites for the most up-to-date information on these products. The following is a list of some of these products. It is not a comprehensive list.

Compaq's TruClusters (http://www.digital.com/info/SP4417/SP4417HM.HTM)
Data General's DG UX Clusters (http://www.dg.com)
HP's MC/ServiceGuard (http://www.corporatebusiness.hp.com/index.html)
IBM's HA Products (http://www.ibm.com/servers/clusters)
Microsoft Cluster Services (http://www.microsoft.com/ntserver/)
NCR's LifeKeeper (http://www3.ncr.com/support/nt/lknt/lkntsol.htm)
Novell's HA solutions (http://www.novell.com/whitepapers/ha/hawp.html)
RSi's Resident Server Facility (http://www.rsi.co.uk/)
Sequent's ptx/CLUSTERS (http://www.sequent.com/products/software/layered/ptx_clusters.html)
SGI's IRIS FailSafe (http://www.sgi.com/Technology/FailSafe.html)
Siemens Reliant Monitor Software (http://www.siemens.de/servers/rms/rms_us.htm)
Stratus's CA Solution (http://www.stratus.com/docs/service/products/index.htm)
Sun Clusters (http://www.sun.com/clusters)
TurboCluster Server (http://community.turbolinux.com/cluster/)
VERITAS Cluster Server (http://www.veritas.com)

5 Technological Scope

The design of an HA solution presents many technical challenges, from the perspectives of both infrastructure design and its use by an application. This section will touch on some of these issues.

An HA infrastructure can be part of the base operating system or a separate layer that sits on top of the operating system. There are examples of both kinds of design in the commercial products available to the cluster community today. Integrating an HA solution into the underlying operating system has several advantages, including better performance and greater ease of use. The main advantages of an HA solution that sits on top of an operating system include independence of the solution with respect to the underlying operating system, leading to better portability across various hardware platforms and often a smaller development cycle. Both from a user's perspective and that of HA applications that sit on top of an HA

framework, the SSI capability of an HA solution is critical to its ease of use. SSI is necessary for providing a single-entity view of the cluster and for easing the task of administering a cluster. This is a rapidly growing area of research for cluster computing in general. For an HA solution, the most important elements of SSI include global file systems, global devices, and global networking in the cluster.

As with any multiresource system, load balancing becomes important in a clustering solution and assumes additional dimensions of complexity when its integration with HA is taken into account. The main facet of load balancing in an HA product includes decisions about the node to which services should be failed over if the node on which these were running goes down. The decision for this can be either resource driven, in which only a subset of the cluster nodes have the appropriate resources, or it can be user driven, in which the user defines the fail-over order a priori. The issue of load balancing on a quiescent cluster (where the nodes are in steady state, some up and others down) is similar to that on a non-HA cluster with resource constraints.

Scalability of HA solutions has not received its due attention, since typical HA cluster nodes tend to be *fat* server nodes. Hence, there does not seem to be a compelling reason to support very large (greater than 32 nodes) HA clusters. That is, algorithms used in the design and implementation of the HA infrastructure do not necessarily scale. As HA becomes increasingly popular and begins to be used by many different kinds of applications, a need for supporting larger clusters will become important. This has already started happening in the marketplace, where customers are demanding highly available applications on nodes that are *thin* nodes, which are clustered together with other thin or fat nodes. This may also happen if highly scalable HPC solutions need to be integrated with HA solutions.

As HA becomes increasingly important, there is a significant growth in the number of applications that will need to be made highly available. The traditional "how difficult is it to program" issue that has been a great source of extensive research in parallel and distributed computing and clustering in the HPC arena begins to loom in the HA world as well. There is ongoing work in the HA area to establish easier ways for users to write custom HA applications.

Ease of cluster management is extremely important in general and for HA in particular. HA products are often difficult to understand from a user's perspective, as users often prefer to treat their HA cluster as a "system" that acts as a server or a server group and provides seamless access to services in the wake of different types of failures. Cluster

"An HA infrastructure can be part of the base operating system or a separate layer that sits on top of the operating system. There are examples of both kinds of design in the commercial products available to the cluster community today."

"As with any multiresource system, load balancing becomes important in a clustering solution and assumes additional dimensions of complexity when its integration with HA is taken into account."

administrators do not want to be burdened with details of how the HA software works. The success of an HA solution is largely dependent on how easy it is to administer and maintain. A Graphical User Interface for cluster administration, management, and monitoring (and sometimes even controlling) is a good starting point. However, the real enabler here is SSI, which allows administration tools, graphical or not, to present a unified picture of the cluster to an administrator. All this becomes particularly critical in an HA environment, where incorrect cluster management and/or user error should minimally affect system downtime.

One of the advantages of cluster computing and a reason for its growing success is the fact that a cluster can consist of heterogeneous nodes. This is in contrast to multicomputers in which the nodes are usually homogeneous nodes. Support for heterogeneous nodes in a cluster is, in fact, almost always a necessity for a user. Users often test out applications on small test/development nodes before deploying these on bigger nodes in the cluster. Heterogeneity support and ways to effectively use a heterogeneous cluster have been actively researched in the HPC area for the past two decades, and these topics are now receiving attention in the HA area as well. For an HA system, these issues are further complicated by considerations such as the degree of integration with the underlying operating system, vendor support in terms of the APIs used by HA applications, portability of the APIs, and support for rolling upgrades.

6 Conclusions

HA is becoming increasingly important, since there is an increasing demand for minimal downtime of systems. Clustering provides the basic infrastructure, both in hardware and software, to support HA. Typical HA solutions span clusters of sizes ranging from two to eight server nodes.

There are many different HA products available to the cluster community, each with several HA applications enabled for them in a prepackaged form and others being supported by various vendors. HA is in its infancy stage and presents exciting opportunities for research and development. This is especially true with respect to its seamless integration with other facets of cluster computing such as HPC, scalability, and standardization of APIs.

BIOGRAPHY

Ira Pramanick received a B.Tech. in electrical engineering from the Indian Institute of Technology, Kharagpur, India, and a Ph.D. in electrical and computer engineering from the University of Iowa. She was an assistant professor in the Electrical and Computer Engineering Department at the University of Alabama in Huntsville for a year. She then worked at IBM and Silicon Graphics and is currently at Sun Microsystems. Her research interests include parallel processing and distributed computing, clusters, grids, algorithms and applications, and highly available systems. She has served on the program committees for several IEEE conferences related to parallel, distributed, cluster, and grid computing. She was a program vice chair for IEEE CLUSTER 2000 and is the tutorial chair for IEEE Cluster 2001. She is a member of the Executive Committee of the IEEE Task Force on Cluster Computing, being the coordinator for the High Availability Technical Area. She is a holder of three U.S. patents.

REFERENCES

Birman, K., and Cooper, R. 1991. The ISIS Project: Real experience with a fault tolerant programming system. *Operating Systems Review*, 103-107.

Buyya, R., ed. 1999a. *High Performance Cluster Computing: Architectures and Systems*. Vol. 1. Upper Saddle River, NJ: Prentice Hall.

Buyya, R., ed. 1999b. *High Performance Cluster Computing: Programming and Applications*. Vol. 2. Upper Saddle River, NJ: Prentice Hall.

Jalote, P. 1998. *Fault Tolerance in Distributed Systems*. Upper Saddle River, NJ: Prentice Hall.

Khalidi, Y. A., et al. 1996. Solaris MC: A multi computer OS. In *Proceedings of the 1996 Usenix Conference*, January.

Pfister, G. 1998. *In Search of Clusters*. Upper Saddle River, NJ: Prentice Hall.

van Renesse, R., Birman, K. P., and Maffeis, S. 1996. Horus, a flexible group communication system. *Communications of the ACM*, April.

Zhang, W. 2000. Linux virtual servers for scalable network services. Paper presented at the Ottawa Linux Symposium.