March 29, 2001

Gale Meyer and Matt Bishop
NCISSE 2001 Paper Selection Committee
ncisse2001paper@cs.ucdavis.edu

Subject: GENERAL TRACK academic presentation paper

Dear General Paper Track Chairs:

We would like to submit our paper (**GENERAL TRACK**) to present in the 2001 NCISSE conference. Enclosed please find a copy of our manuscript.
The information of our paper follows:

Article Title:
Service recovery in a loosely coupled distributed computing environment
Authors:
James Garvin, Michael Baldwin, and Willie Chang
Affiliation:
New Mexico Institute of Mining and Technology
Corresponding Author and Contact Address:
Willie Chang
Computer Science Department
New Mexico Institute of Mining and Technology
Socorro, NM 87801-4682
Tel: (505) 835-5030 Fax: (505) 835-5587
E-mail: willie@cs.nmt.edu

Thank you and wish the conference a great success.

Respectfully,

Willie Chang
Assistant Professor
Computer Science Department
New Mexico Institute of Mining and Technology

Attachments: Abstract and paper (including author biography)

GENERAL TRACK NCISSE paper

Service Recovery in a Loosely Coupled Distributed Computing Environment

James Garvin, Michael Baldwin, and Willie Chang
New Mexico Institute of Mining and Technology
Socorro, NM 87801-4682
(505) 835-5030 Fax (505) 835-5587
willie@cs.nmt.edu

Abstract

We propose a new method to enforce the fault-tolerant and recovery capabilities of critical network services in a distributed computing environment. With our approach a service can be dynamically dispatched onto any available host, and at any time, each service is not only viable but also consumes the normal amount of resources without duplication. In the events or indications of system failures, services would reestablish themselves onto other hosts via a non-preemptive remote execution process. The basic simulation is to have a vital service reside on a primary host, with a secondary host designated as standing by. The primary host performs the service until the occurrence or indication of fatal faults in the system. Then, the secondary host resumes the service and becomes the primary host, with yet another host being designated as the new secondary host for that service. The cluster monitor and failover function is the center of our simulation and itself a critical service. Such a recovery scheme can improve the current failover and load-balancing network technology by reducing the resource usage and cluster-support overhead.

Backgrounds and Issues

As computer networks gain ubiquitous importance in the work place, operations of network services become vital and must be constantly guarded. In the events of network service failures, a set of contingency procedures must be carried out in a timely and effective manner [1-3]. The objective of our simulation and analysis is to find an efficient and effective method with a small implementation overhead for recovery of vital network

services of our usual concerns, namely, E-mail, World Wide Web, and domain name service (DNS).

E-mail service is widely considered as the most indispensable way of inter and intra-office communication. Without such service, office work will be greatly hindered. World Wide Web is the portal to an organization from the world. The discontinuation of Web service can be regarded as virtually shutting down an organization. DNS is responsible for providing road maps for all other network services in an organization, as well as for remote networks that wish to establish communication with that organization. Thus, DNS is essential to computer networks and without it, all network services may halt.

The commonly used recovery methods for network services are inefficient and sometimes ineffective. The cost associated with these methods is usually high. Two conventional solutions are immediately available: Periodical data backups and configuration of static secondary service hosts (secondary DNS hosts and mail exchange hosts.)

The first general recovery approach relies on performing periodical storage backups on all the file systems without discrimination or distinction made to selective network facilities. Since the storage requirement has been constantly growing in a network realm, the duplication and the restoration of such massive data have become an excessively expensive process in both time and money. Further, the recovery of data constitute the major network services does not guarantee the resumption of the operations in a seamless and continuous fashion. This is usually caused by the incremental and periodical natures of the backup schedule [4]. In general, a periodical file system backup is usually the last resort since besides having inefficiency and storage overhead, it suffers discontinuity of service.

Designation of secondary service hosts (mail exchange and secondary DNS hosts) as the second general recovery approach does not guarantee that the services, continued on the new hosts, will not be susceptible to the same factors that brought down the primary hosts. In addition, DNS registry is a set of fixed information that cannot be dynamically altered [5-6]. We need a recovery system which is flexible in nature, namely, has the capability to select and assign each vital network service to any available candidate on the intranet of the organization within a very short time frame in order to resume services seamlessly [7-10].

Recent network technology introduced two new recovery methods: A virtual router redundancy protocol (VRRP) to perform a failover and load-balancing dynamic network routing scheme, and an IP teaming based on a Beowulf cluster [11-13]. Both methods are again costly due to the redundant structure in the configuration (Figure 1). Also, each VRRP channel provides only a limited recovery support such as a pair of hosts responsible

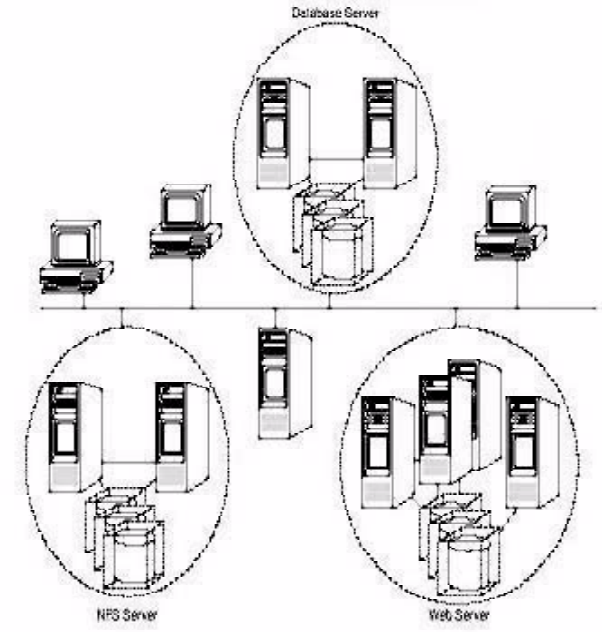

Figure 1. A service is performed by a group of hosts in which each has duplicate resources.

for all vital services. It is possible to configure more VRRP channels, but the redundancy lends itself impractical in cost. The same can be said with the IP Teaming. What we truly need is a dynamic dispatch of services across hosts, so that at any given time, each critical service consumes only a single set of (or much limited) resources without duplication.

Recovery Methodology

The basic version of our proposed recovery strategy can be described in the following: In a distributed or clustered computer network configuration each major network service is established dynamically on a single host. At any given time, each selective network service resides on one of the hosts as its primary server, with a secondary host in a stand-by mode. The primary host performs the service until events indicating the occurrence of fatal faults in the host's system. Then, the secondary host resumes the service and becomes the primary host, with yet another host designated as the new secondary host for that service.

Major issues of this strategy may include:

1. Fault detection of the service to initiate the process migration. This can be accomplished via a collection of fault detection processes running either on the service host or a remote system (preferably, the secondary or stand-by host) to verify the server status [1].

2. Service quality and coherency. A service failover can be contrived as process migration between hosts on the network. Therefore, the process context along with other resources must be provided to start the service process on the secondary host (to become the primary host of that service.) Service quality is usually measured by the responsiveness and coherency is the resumption of the service. An example of coherency is the mail queue and mail aliases that should also be transferred to the new host besides the mailboxes [4], [9], [14].
3. Designating a succeeding secondary host and establishing its readiness. Triggered by the fault detection processes, a normal working host can be selected as the new secondary host. A scheduling table can serve the purpose. The table can be updated at intervals as the cluster is monitored and during each service switch.
4. The functions which monitor and switch a service are themselves a critical service (typically called a cluster monitor [13]) which must also be monitored (by itself) and designated with a primary and a secondary service hosts (in case of a failover.)

Our method is suitable for a common network of workstations supported by reliable file systems. The file service is usually a redundant mirror storage system capable of recovering itself. The recovery scheme presented here can tolerate transient processor failures and other types of failures with a low overhead.

A set of functions monitor and test services periodically, and dispatch services if needed. These functions constitute a cluster monitor which is itself vital and needs be tested and dynamically dispatched. When the cluster monitor detects lacking or slowing down of a service. The following procedures are taken:

1. Shutdown the service process or deactivate the IP address in the network interface of the questionable primary host.
2. Initiate the new primary host (from the secondary host):

   2.1 Ready the required resources for the service (e.g., mount NFS directories to include the needed config and doc files such as mail aliases, mail queue, web/CGI docs, DNS records, etc.)

   2.2 Wake up the stand-by service process via remote procedural calls (RPC).

   2.3 Configure the network interface and activate the service IP address (e.g., ifconfig[1]).
3. Select a new secondary host from a pool of available hosts. Preferably, a host with

[1] The network interface configuration we perform here is simply assigning an additional IP address on top of the IP address currently used by the host, and not the usually lengthy reconfiguration in which a router cache flush is occasionally required. For more detail, refer to UNIX administration documentation.

adequate resources and light overall loads, and not currently designated as either a primary or a secondary host of any service. A scheduling table can be used for this purpose. However, the currently condition of a candidate must be verified.

4. Initiate the new secondary host:

4.1 Test the availability of the required resources.

4.2 Create a new service process via RPC but sleep the process in order to reduce the processor overhead.

The requirement of starting up the process in 4.2 may require mounting certain file systems to obtain the executable and configuration files (typically in the case of a web server.) Although the service process can be started in the primary host stage (instead of creating it and put it to sleep in 4.2, and then wake it up in 2.2,) a sleeping process would provide better response time especially with its resources ready (file systems mounted).

The network interface hardware can accommodate multiple IP addresses. The IP address of a service defined in DNS can be added into the network interface of the primary host. This IP address can be the DNS server address itself. Thus, the DNS can also be served dynamically using our approach. In case of a slow network interface configuration, we can prepare the work by adding an extra step:

4.3 Configure the network interface to include the service IP address but keep it inactivated.

So that in 2.3 we need only activate the IP address instead of having to configure it at the same time. However, the recent improvement in the network hardware seems to have eliminated such performance concern.

Simulation Results and Analysis

The results of the average completion time for detection and failover in our simulation implemented on a workstation cluster is shown in tables 1 and 2. There are four stand-alone network services (as oppose to inetd-spawned processes,) namely, Apache httpd for web service, Berkeley sendmail for mail service, the system-included named for DNS, and our cluster monitor service.

Table 1 is based on the event of a transient failure in the primary host[2]. At each monitor interval a *ping* utility program is deployed to detect if there is a total failure of the primary host. If there is, the service process migration is carried out and the statistics are recorded. The deterministic time, is the period from the failure to when migration starts, and

[2] We simply just disconnect the host from the network to simulate system down time.

Table 1. Average response time at transient failures (in seconds)

Stand-alone Services	Monitor Interval	Deterministic	Migration	Response
Apache httpd	4	2.3	1.4	3.7
	8	4.7	1.4	6.1
	16	8.9	1.4	10.3
	32	16.8	1.4	18.2
Berkeley sendmail	4	2.3	0.9	3.2
	8	4.7	0.9	5.6
	16	8.9	0.9	9.8
	32	16.8	0.9	17.7
DNS named	4	2.3	0.7	3.0
	8	4.7	0.7	5.4
	16	8.9	0.7	9.6
	32	16.8	0.7	17.5
Cluster Monitor	4	2.2	0.5	2.7
	8	4.7	0.5	5.2
	16	8.8	0.5	9.3
	32	16.7	0.5	17.2

Table 2. Average response time at system overloads (in seconds)

Stand-alone Services	Monitor Interval	Deterministic	Migration	Response
Apache httpd	4	2.5	2.2	4.7
	8	5.1	2.2	7.3
	16	9.3	2.2	11.5
	32	17.6	2.2	19.8
Berkeley sendmail	4	2.6	1.7	4.3
	8	5.3	1.7	7.0
	16	9.5	1.7	11.2
	32	18.0	1.7	19.7
DNS named	4	2.4	1.5	3.9
	8	5.1	1.5	6.6
	16	9.1	1.5	10.6
	32	17.3	1.5	18.8
Cluster Monitor	4	2.4	1.3	3.7
	8	5.1	1.3	6.4
	16	9.1	1.3	10.4
	32	17.2	1.3	18.5

Table 3. Cost-effectiveness: (1/R)/(1/M) R: Response M: Monitor Interval

	Apache httpd				Berkeley sendmail				DNS named				Cluster Monitor			
Moniter Interval	4	8	16	32	4	8	16	32	4	8	16	32	4	8	16	32
Transient Failure	1.08	1.31	1.55	1.76	1.25	1.43	1.63	1.81	1.33	1.48	1.70	1.83	1.48	1.54	1.72	1.86
Overload	0.85	1.10	1.39	1.62	0.93	1.14	1.43	1.62	1.03	1.21	1.51	1.70	1.08	1.25	1.54	1.73

the response time is the period from the failure to the completion of the migration, equals to the deterministic time plus the migration time [16].

Table 2 is based on the event of system overloads. Such a condition is usually heuristic and based on the number of service processes and the overall load of the system. For the mail service, we also take into account the size of the mail queue[3]. Again, a *ping* process is first used before looking into the system and service loads. Also the migration process must include shutting down the current service or disable the service IP address. All these extra requirement lengthen both the deterministic and migration time.

The secondary host establishment is carried out after the critical service migration, and hence, it is not counted toward the response time. In general, services of smaller footprints migrate faster. Our cluster monitor has the least resource consumption among the four services, and resides on a separate host (as other services). Hence, its migration time tends to be the smallest of the four.

The analysis can be summarized in the following:

1. Our proposed recovery scheme can achieve predictable performance. The results also reveal the impact of process migration latency on the quality of the service. To our knowledge, no analysis on the performance impact from process migration latency has been carried out previously.
2. With performance measurement of service responsiveness and coherency (resuming the mail service including the mail queue) in mind, we test the simulation from the user ends and find the desirable cluster monitor configuration and recovery strategies.
3. By taking the inverse of the response time over the monitor frequency, we obtain the distribution of the cost effectiveness of our approach (Table 3). It appears to be almost linear. This indicates that the performance of our approach is highly predictable and expectable [14].

Concluding Remarks

Our failover and recovery approach performs well in simulations with very little overhead and resource waste. The ease of implementing our method is due to the simplicity of non-preemptive remote execution [17], and having only one host per service. However, the advantage of preemptive process migration and having multiple hosts per service should also be considered to extend our approach.

---

[3] Fortunately, the mail queue can also be recovered in the new host by mounting the corresponding directories.

The basis of service process switching is a policy of distributed process scheduling, with the enhancement from the process and thread migration, for achieving service responsiveness and coherency [18-19]. The obvious tradeoffs are the performance versus computing resources [20]. In order to achieve better response time and service coherency through process migration, more computing resources must be granted in the entire procedure, namely, process priority, shared memory space, etc [21]. Currently, the analysis of this resources-performance tradeoff is not clearly known, and the cost-to-efficiency factor has not been analyzed.

The loads of the network services have to be simulated in a considerably large scale of heavy random user requests in order to obtain more accurate analysis.

Based upon the aforementioned criteria, our future work may include the following:

1. Extend our simulation to accommodate a set of hosts per service for load sharing. In order to assign the same service IP address redundantly on multiple hosts, we need to incorporate the VRRP dynamic routing technology. We would like to retain the capability to dynamically dispatch services onto any set of hosts.
2. Incorporate preemptive process and thread migration instead of remote execution that interrupts and terminates the current service processes.
3. Employ multiple service process migration upon simultaneous failures of multiple services, with the complexities of having multiple services switched among multiple hosts, and the combinations of single/multiple-services primary/secondary hosts.

References

[1] L. Barnett and M. K. Malloy, “ILMON: A UNIX network monitoring facility,” Proceedings of USENIX Conference, pp. 133-144, Washington, District of Columbia, Winter, 1987.

[2] S. Simmons, “Making a large network reliable,” Proceedings of Large Installation Systems Administration (LISA) Workshop, Monterey, California, November 17-18, 1988.

[3] K.I. Mandelberg and V. S. Sunderam, “Process migration in UNIX networks,” Proceedings of USENIX Conference, pp. 357-363, Winter, 1988.

[4] S. Shumway, “Issues in on-line backup,” Proceedings of Large Installation Systems Administration (LISA) Conference, pp. 81-88, San Diego, California, September 30 - October 3, 1991.

[5] D. J. Fiander, “Review: Sendmail,” Proceedings of USENIX, vol. 19, no. 2, pp. 1-48,

1994.

[6] P. Collinson, “Review: DNS and BIND,” Proceedings of USENIX, vol. 18, no. 6, pp. 1-44, 1993.

[7] K. Hwang, W. J. Croft, G. H. Goble, et al., “A Unix-based local computer network with load balancing,” IEEE Computer, vol. 15, no. 4, pp. 55-66, 1982.

[8] W. E. Leland and T. J. Ott, “Load-balancing heuristics and process behavior,” Proceedings of Performance and ACM SIGMETRICS, Vol. 14, pp. 54-69, 1986.

[9] M. Aron, P Druschel, and W. Zwaenepoel, “Efficient support for P-HTTP in cluster-based web servers,” Proceedings of USENIX Annual Technical Conference, pp. 185-198, Monterey, California, June 6-11, 1999.

[10] B. A. Kingsbury and J. T. Kline, “Job and process recovery in a UNIX-based operating system,” Proceedings of USENIX Conference, pp. 355-364, San Diego, California, Winter, 1989.

[11] Dell Computer Corp., “Network teaming,” Technical Brief of Dell Computer, no. 9195P, 2000.

[12] Cisco Systems, Inc., “What is VRRP?” Tech Notes of Cisco Systems, no. Public-471, 2000.

[13] Sun Microsystems, Inc., “Sun Clusters,” White Paper of Sun Microsystems, no. 97234-001, 1997.

[14] N. H. Vaidya, "Another two-level failure recovery scheme: performance impact of checkpoint placement and checkpoint latency,” Technical Report, no. 94-068, Texas A & M University", 1994.

[15] M. Harchol-Balter and A. B. Downing, "Exploiting process lifetime distributions for dynamic load balancing," Proceedings ACM SIGMETRICS, May 1996.

[16] W. Stallings, Operating Systems, 4th Ed., Prentice-Hall, 2001.

[17] A. B. Downey and M. Harchol-Balter, “A note on the limited performance benefits of migrating active processes for load sharing,” Technical Report, no. UCB/CSD-95-888, University of California at Berkeley, November 1995.

[18] P. Gopinath and R. Gupta, “A hybrid approach to load balancing in distributed systems,” Proceedings of Symposium on Experiences with Distributed and Multiprocessor Systems (SEDMS II), pp 133-148, Atlanta Georgia, March 21-22, 1991.

[19] P. Lockwood and D. Agrawal, “A fault-tolerant client-server transaction model,” Proceedings of UNIX Transaction Processing Workshop, pp. 63-71, Pittsburgh,

Pennsylvania, May 1-2, 1989.

[20] P. Krueger and M. Livny, “A comparison of preemptive and non-preemptive load distributing,” Proceedings of The 8th International Conference on Distributed Computing Systems, pp. 123-130, June 1988.

[21] M. R. Barber, “Increased server availability and flexibility through failover capability,” Proceedings of The 11th Large Installation Systems Administration (LISA) Conference, pp. 89, San Diego, California, October 26-31, 1997.

Author Biography

James Garvin has senior standing in the Computer Science Program at the New Mexico Institute of Mining and Technology. His focus is on Microsoft products and developing in the Windows platform. Networks and streaming media are his new fascination. James is working with the Multimedia Productions Department at New Mexico Tech to develop a streaming media platform. He can be contacted at boot@nmt.edu.

Michael Baldwin is senior computer science major at New Mexico Institute of Mining and Technology. Application development and networking are his biggest interests. Mike is a long time Macintosh fan that is also heavily appreciates Unix. Mike is currently working on radar and weather analysis programs for New Mexico Tech. He can be reached at mbaldwin@nmt.edu.

Willie Chang is an assistant professor of the computer science department at the New Mexico Institute of Mining and Technology. His current research interest includes network service management, data visualization, data mining, and signal processing.