A RECONFIGURATION ALGORITHM FOR DISTRIBUTED COMPUTER NETWORKS

Chanan Glezer
Department of Information Systems Engineering, Ben Gurion University, Beer Sheva, Israel 84105

Moshe Zviran
Chair, Management of Technology and Information Systems Department
The Leon Recanati School of Business Administration,Tel Aviv University, Tel Aviv, Israel 69978

Keywords: computer networks, dependability, fault tolerance, load balancing

Abstract: This article presents an algorithmic reconfiguration model, combining mechanisms of load balancing and fault tolerance in order to increase utilization of computer resources in a distributed multi-server, multi-tasking environment. The model has been empirically tested in a network of computers controlling telecommunication hubs and is compared to previous efforts to address this challenge.

1 INTRODUCTION

Telecommunication systems as well as other mission-critical systems such as utility, banking, medical, military and transportation networks rely heavily on state-of-the–art computing and telecommunication technologies.

Fault tolerance in distributed computer networks refers in most cases to a hot-standby approach (Anderson and Lee, 1981), which is based on duplication of computer resources using check-pointing and message-logging techniques (Folliot and Sens, 1994). Nevertheless, during periods of normal operation the duplicated computer resources are underutilized.

Load Balancing in a Distributed Computing System (DCS) (Tiemeyer and Wong, 1988) refers to dynamically allocating and independently performing computation tasks across a heterogeneous network of processors.

Several experiences have been reported on combining load-balancing and fault-tolerance mechanisms, (e.g., Remote Execution Manager (Shoja et al., 1987), Paralex (Babaoglu et al., 1992), Condor (Litzkow et al., 1988), and DAWGS (Clark and McMillin, 1992), Coterie (Tiemeyer and Wong, 1988)). Nevertheless, these systems exhibit only limited fault tolerance capabilities. The most comprehensive attempt to constructs a reconfigurable, fault tolerant system was made in GATOSTAR (Folliot and Sens, 1994).

The goal of this article is to develop, illustrate and practically evaluate an algorithmic model that combines load sharing and fault tolerance using the prominent Hamilton method (Ibarkai and Katoh, 1988).

2 THE RECONFIGURATION MODEL

The proposed model is based on combining the mechanisms for fault tolerance and load balancing in a multi-server and multi-tasking computer network. Following are the assumptions underlying the model:

1. Each computer connected to the network can process several types of tasks concurrently based on the unique requirements of each task.
2. The tasks are processed from queues by (expert) servers operating under the computers connected to the network.
3. In case one of the servers becomes inoperative, the tasks in its incoming queue are routed to similar servers running concurrently on different computers.

Glezer C. and Zviran M. (2004).
A RECONFIGURATION ALGORITHM FOR DISTRIBUTED COMPUTER NETWORKS.
In *Proceedings of the Sixth International Conference on Enterprise Information Systems*, pages 491-494
DOI: 10.5220/0002592704910494
Copyright © SciTePress

4. Servers of a given type on different computers may have a different processing capacity.
5. The prototype derived from the conceptual model should accommodate safety mechanisms that will enable it to handle both crash-type and arbitrary (Byzantine) failures, resulting in a higher failure mode coverage (Laprie, 1995)

The challenge in providing fault tolerance in the scenario described above stems from the dynamic and uncertain nature of the network. As a case in point, computers can be installed or removed in real time, unexpected software/hardware crashes may occur. It is the need to provide end-users with quality service at a minimum level of response time that prompts the seeking and evaluation of mechanisms that will detect faults as well as rapidly adjust the performance of the network so that the desired quality standards are maintained. Effective synchronization and communication protocols are a critical asset for the success of such a system.
The proposed reconfiguration model is algorithmic and comprises the following elements:

Network Status: A set of vectors and matrices that capture the actual state of the network at any given point in time (termed *logical configuration*). These elements describe which servers and computers are active and which tasks are processed on each server at any point in time. It also includes operational instructions on what to do with the tasks running on a server in case the host computer becomes inoperative.
Task-Reconfiguration Algorithm: An algorithmic set of procedures that transform the network status elements so that they capture and react to changes in the state of the network (termed *events*) with minimal delay.

Note that contrary to the logical configuration, the *physical configuration* of the network refers to the hardware profile (e.g., ratio of memory/CPU power, number of I/O devices etc.). Changes in the physical configuration are therefore less frequent than changes in the logical configuration. The former fall outside the focus of the model because they cannot affect the behavior of the model unless they are first reflected in the logical configuration (e.g., register a newly acquired computer in an appropriate status matrix).
The basic principle of the model is to dynamically redistribute tasks between servers available on the network in response to threatening events. When such an event occurs in the network (e.g., a computer crashes, or an arbitrary failure occurs), the model reallocates active tasks on running the stalled computer to other available computers according to a proportional ratio determined by the relative importance of the servers. The importance (vote) of a server is based on the system manager's perception of the relative processing capacity of all servers of a given type (running on different computers). In case there is a leftover task as a result of the above event, then this task is allocated to the computer that has the highest remainder, using the Hamilton method (Ibarkai and Katoh, 1988). This approach can be applied to the event of system initialization as well.

3 EVALUATION OF THE MODEL

The proposed reconfiguration model was evaluated on a large national digital telecommunications network comprising approximately 200 hubs of the following types: TX-1, TMX-10, and TMX-100 (manufactured by Northern Telecom) and System-12 (manufactured by Alcatel). The above hubs serve in the range of 1000 to 20,000 customers each. As an example, the System-12 hub is a complex hardware and software device running several tens of modules concurrently. The modules are responsible for various tasks (e.g., central control, connection with customers, message routing, connection bus with other hubs, distribution control and more). The System-12 hub uses approximately 100 types of status messages in order to monitor and coordinate the operation of the hub (e.g., detecting and handling malfunctions).
The model for controlling the network was implemented using the C programming language. The system operates over the VAX/OpenVMS operating system running on two VAX 4000-5000 computers and using the Digital RMS software for file management. The computers are connected in a cluster using the Digital Small Systems Interconnect (DSSI), which enables sharing of disks among computers, synchronization of events and transmission of data. Connection between the servers on the computers and the hubs they are serving is implemented using a X.25 packet switching network. This network transmits instructions from the servers to the hubs and events from the hubs to the servers. The performance of the network was measured and recorded using Digital's Monitor software package over a period of one month. Several measurements were performed during the day and an arithmetic average was used to summarize the results. The effect of the workload created by MONITOR on the results is

negligible compared to the other tasks running on the computers, and can therefore be ignored.

Table 1: Comparison of Cost/Utilization and Balance Factors

	Hot Standby	Reconfiguration Model
F	0.365	0.21
B	0.653	0.433

The benefit from using the proposed model was evaluated the theory of constraints (TOC), with or without a manufacturing focus, and on the cost/utilization model (Borovits and Ein-Dor, 1977) The idea underlying the method is to generalize the application of TOC combined with cost/utilization for performance analysis of a single processor, into a scenario of a distributed network composed of several processors. The method exploits a simple graphic display of the processing element (PE) components (e.g., CPU, Input/Output, Memory, Communication links) in order to pinpoint improper imbalances, fluctuations and bottlenecks. The model uses the following two main indicators for evaluating performance of a distributed system. The values of **F** (cost utilization factor) and **B** (balance) are between 0 and 1.

$$\mathbf{F}= \sum Pi * Ui \qquad (i=1\ldots.I)$$

$$\mathbf{B= Balance\ Factor} = 2*\sqrt{\sum[(F-Ui)**2 *Pi]}$$

Where **I** = Number of processing elements on a single processor
Pi= Relative cost of PE i
Ui= Utilization percentage level of PE i
The closer **F** gets to 1 the better the utilization of the network is in terms of the cost of its elements. The closer **B** gets to the less balanced the network becomes resulting in bigger variance in the utilization of its elements. Since the percentage of resource utilization in the original cost utilization model is replaced by the maximal resource utilization in the PE, it is better to have a system that is balanced (a smaller **B** is better). If there is a resource that is highly utilized in one of the PEs compared to the other resources in that PE, a moderate increase in the workload might cause a crash or bottleneck in that PE. This could affect the viability of the whole system.
The evaluation of the reconfiguration model was performed by comparing the B and F measures in two scenarios: hot standby, where a computer is used as a mirror backup (without routinely sharing the workload of the other computers); and a scenario, where the backup computer processes tasks and the load is balanced among all computers linked to the network (reconfiguration).
Table 1 depicts the values calculated for B and F in the two scenarios. In both cases the utilization of the two computers is not good. The cost of purchasing the backup computer is an imposed operational constraint, and therefore there is no option to alter the cost of the combined system. The reconfiguration model seems to be the preferred option because the system is more balanced ($0.433<0.653$) and can therefore handle peak processing volume with a better quality of service. In other words, the model enables avoiding bottlenecks which cause down time and impair service to end-users. In the hot standby option, the risk of a total malfunction, however, is higher because the operations relies only on a single computer which is more prone to crash.
Table 2 contrasts the proposed model with the GATOSTAR system (Folliot and Sens, 1994). The main theme of the reconfiguration model presented in this article is the application of the Hamilton method (Ibarkai and Katoh, 1988) to the task redistribution process. This article also analyses the effectiveness of the proposed method in a very large-scale industrial setting. A combination of the two approaches is recommended for covering all aspects of the dependability challenge

4 DISCUSSION

This study proposed and evaluated an algorithmic model for combining hot standby and load balancing in a network of computers where tasks are processed concurrently and re-allocated by servers running concurrently on different computers.
The research found support for the claim that a combination of fault tolerance and load balancing mechanisms is more effective than software-based fault tolerance alone. The combined approach is also better than implementing a purely hardware-based fault tolerant system, which is a much more expensive solution because it requires the purchase of specialized, synchronized, fault-tolerant computers.

Table 2: Comparing the HS/LB model with the GATOSTAR system

Criterion/ System	HS/LB Reconfiguration (Model and Prototype)	GATO-STAR
Locus of model	Specification of a redistribution mechanism to increase utilization	Seamless unification of GATOS and STAR
Impleme-ntation constructs	Network of computers, each with servers that handle processes	Ring of hosts composed daemons (LSM, FTM, RM)
Algorithm	Hamilton method (Ibarkai and Katoh, 1988)	Overload, migration, reception thresholds
Network status information	Matrices and vectors	Local shared memory
Prototype	Hubs serving a national telecommunica-tion network	Workstatio-ns in a LAN of a university
Evaluation criteria	Balance (B) and Utilization (U) factors	Overhead of process allocation, logging.
Conclusions	Combining load balancing with fault tolerance recommended for increasing potential of dependable computer networks	Useful for increasing dependa-bility of LANs). Need to reduce overhead

A major advantage of the model is its flexibility and scalability. The model can operate on various hardware platforms and has a great effect on both real-time and Electronic Data Processing (EDP) applications.

The model can be expanded in the future to include an internal feedback system that changes the vote (relative importance) of different servers automatically to achieve an optimal balance in the network. Such a system will invoke a quantitative model, suggest a modification to the human administrator, and enable "what-if" analysis regarding the effects caused by various changes in the logical configuration of the network.

REFERENCES

Anderson, T., and Lee, P.A., 1981. *Fault Tolerance: Principles and Practice*, Prentice Hall International, Englewood Cliffs, N.J.

Babaoglu, O., Alvisi, L., Amoroso, A., and Davoli, R., 1992. Paralex: An environment for parallel programming in distributed systems, *Proc. of International Conference on Supercomputing,* Washington D.C.

Borovits, I., and Ein-Dor, P., 1977. Cost/utilization: A measure of system performance, *Communications of the ACM*, 20 (3), pp. 185-191.

Clark, H., and McMillin, V., 1992. DAWGS – A distributed computer server utilizing idle workstations, *Journal of Parallel Distributed Computing*, 14, pp. 175-186.

Folliot, B., and Sens, P., 1994. GATOSTAR: A fault tolerant load sharing facility for parallel applications, *Proc. of the first European dependable computing conference*, Berlin.

Ibarkai, T. and Katoh, N., 1988. Resource Allocation Problems: Algorithmic Approaches, MIT Press, *Foundations of Computer Series*, Cambridge, MA, (Chap. 6: The apportionment problem: the Hamilton Method, pp. 106-126)

Laprie, J.C., 1995. Dependable computing: Concepts, limits, challenges, invited paper *FTCS-25*, pp. 42-54.

Litzkow, M.J., Livny, M., and Mutka, M.W., 1988. Condor - A hunter of idle workstations, *Proc. of the 8th International Conference on Distributed Computing Systems*, San Jose, CA.

Shoja, C.G., Clarke, G., and Taylor, T., 1987. REM: A distributed facility for utilizing idle processing power of workstations, *Proc. of the IFIP Conference on Distributed Processing*, Amsterdam.

Tiemeyer, M.P., Wong, J.S.K, 1998. A task migration algorithm for heterogeneous distributed computing systems, *Journal of Systems and Software*, 41 (3), pp. 175 – 188.