A World-Wide Web Server on a Multicomputer System *

Chun-Hsing Wu, Chun-Chao Yeh, and Jie-Yong Juang
Dept. of Computer Science and Information Engineering
National Taiwan University
Taipei, Taiwan, 10617
Fax:886-2-3628167
Email: juang@csie.ntu.edu.tw

Abstract

As the number of people browsing the world-wide web increases explosively, workload of popular web servers also increases rapidly. A multicomputer system that was designed for I/O intensive applications has been found to be quite suitable for serving as a web server. The system is composed of multiple clusters of multiprocessors interconnected by an interconnection network. The interconnection network is an ATM-like cell-based switching network which can be restructured so that the system can be scaled up to meet the increasing demands of web service. It is also found that the multicomputer system can support video streams effectively. Design of the system as well as porting of web servers on to it will be discussed in this paper.

1 Introduction

Since the initial World-Wide Web prototype was developed in 1990, it grows rapidly and becomes the most popular system on the Internet in recent years [2]. According to the Internet Domain Survey conducted in January 1996, about 76,000 systems now have the registered domain name *www*, up from only 600 in July, 1994 [9]. Due to the increasing demands of web requests, it becomes a critical issue for a web server of a popular site to offer high performance and guarantee high availability. A web server must be able to serve multiple simultaneous requests promptly even in its peak time. Besides, from the information providers' point of view, the provided information of each server will accumulate as the time passes by. Therefore, it is desirable for a web server to be scalable and to support better information searching capability. In the near future, supporting video streams will also become a basic requirement for a web server. Accordingly, highly-available, scalable machine with strong I/O capability will be necessary to run a web server.

To address these issues, we are developing a world-wide web server on top of a multicomputer machine designed and implemented in our laboratory. The machine uses a multistage switching network to connect multiple clusters of multiprocessors. Each processor can either be a simple CPU module, or with individual storage devices and/or network adapters attached depending on the needs of applications. As a result, multiple I/O devices can be accessed concurrently to provide higher I/O bandwidth than single-bus machines. An ATM-like cell-based interconnection network is designed to support more predictable and more efficient inter-processor communication. In addition, its restructurable architecture also makes the machine scalable. The web server on the machine will identify different kinds of requests and assign them to the processors optimized to handle the type of requests. The web server can also work well as a proxy web server. Furthermore, with our multicomputer, a simple replication strategy can be applied with little overhead to achieve high availability.

In the following sections, we will describe the characteristics of existing WWW systems and the availability issues first. Some related works are also discussed. Then, we present our multicomputer platform in Section 3. In Section 4, a general software structure of a web server proposed for multicomputers like ours is depicted. An implementation of the server is discussed in Section 5. Section 6 draws conclusions.

2 Issues of WWW Servers

In World-Wide Web, user agents use the application-level stateless Hypertext Transfer Protocol (HTTP) [3] to request documents or other kinds of objects from web servers (or from proxies/gateways to other Internet servers). In practice, a browser establishes a TCP connection to a web server before requesting a document. After fetching a document, it disconnects the connection immediately. The web server will feed the browser the document or just redirect it to contact other servers. A document may be a text file, either in plain format, in HTML, an image, or a motion-picture file. It also may be a virtual document, actual data of which is generated on-the-fly. Most servers support Common Gateway Interface (CGI) for virtual documents. Database queries and search requests can be implemented by this mechanism. There is also an extension to HTTP called server-push to handle dynamic documents consisting of multiple parts. Server-push allows implementation of simple animation, nevertheless it may occupy a con-

*This work was partially supported by the National Science Council under grants NSC84-2221-E-002-004, and NSC85-2221-E-002-029.

nection for a longer time.

To reduce the network traffic, a browser may contact a local proxy server (or caching server) for remote documents. The proxy server fetches remote documents for the browsers and then keeps a copy in its local disk. Next time if some request wants the same document, it may directly feed the browser with the local copy [8].

According to several trace analysis studies [4, 1, 5], small documents are accessed more frequently than large documents. This observation is consistent with the general web page design rule to keep the front page small. Most documents are read-only or not modified frequently comparing with ordinary files. For those virtual documents generated by database access, they usually invoke search queries which may involve large volume of read-only files. In summary, small files are accessed in most simple connections, and read-only file access contributes to most disk activities. The phenomena yields another opportunity for optimizing a web server.

In a web server, disk write access is performed mainly in log operation, and in caching remote documents in local disks in case of a proxy server. Disk write in these cases is usually write-once, and the cached remote documents are always discardable. These features make the traditional weak consistency model of network file systems suitable for a web server and alleviate the consistency overhead in replicating files. Besides these, the link information in a hypertext document may give a hint that indicates which files will be requested soon. The link information may be used to design a more effective buffer replacement algorithm than that in a general file system.

Many browsers are multi-threaded. They are able to simultaneously send multiple requests for in-lined images within a HTML document. It can increase the concurrency of a web server, but it also reduces the number of users that a web server can serve during peak time. Besides, a request may occupy TCP connection for a long period of time if it's requesting for a large file or is connected from a low-speed network. For a proxy server, it will need more connection capability to serve local proxy clients and fetch remote documents while no cached file is available. However, there is limitation on the available connections of a server due to the shortage of operating system resources such as limited TCP ports, mbuf, process table, etc. Fine-tuning the system will solve the problem, but will not solve it definitely.

Existing redirection mechanism in HTTP protocol can be adopted to improve the availability and scalability [3]. Requests received by a central web machine can be redirected to a pool of web machines. This approach alleviates the workload of the central web machine, but it incurs network and connection overhead in redirection. In addition, the central web machine may still be the hot-spot of the machine groups. Furthermore, the same document returned by two different machines in the groups will be considered by clients or proxy servers as two different copies. It makes the global caching scheme ineffective.

In the design of NCSA's scalable web server [6], a Round-Robin DNS approach is designed for distributing requests among a cluster of web servers, which share the same alias host name. The authoritative DNS server in the cluster acts as a virtual router to distribute requests by rotating through the web servers that are alternately mapped to the shared alias name. This design eliminates the single point of failure, and it can dynamically increases the capacity of the virtual server. However, result of the name resolving will be cached in a client's local name server for a period of time. Any further resolving request to the same local name server, even from different clients, will reuse it before the mapping is expired. This may make the load distribution among the server cluster uneven. One method proposed to alleviate this effect is to shorten the time-to-live value for each resolving result, and then the name servers of clients will query again soon, but the DNS queries will increase the global traffic.

In the design of our proposed web server, it distributes the requests transparently and more evenly among several clusters of multiprocessors. It also can tolerate single point of failures. Before describing the design of the web server, we present the multicomputer architecture first.

3 The NTU Cost-effective Multicomputer Clusters

In this section we present a cost-effective multicomputer architecture, called SIGMA (System-Integrated Growable Multicomputer Architecture) , developed at National Taiwan University. The goal of the project was to develop a clustered machines to offer better cost-performance ratio than conventional supercomputers. In stead of using expensive custom design approach, the NTU SIGMA machine is developed with off-the-shelf components. It leverages the latest microprocessor technologies, and integrate computing components together with a proprietary interconnection network.

3.1 Architecture overview

Figure 1 shows an example of SIGMA multicomputer architecture with 64-node connection. The system consists of two major entities: computing nodes and network subsystem. The computing node can be as simple as a CPU module, or can be a complete computer with proper I/O capabilities. The network subsystem is a multistage interconnection network(MIN). For instance, in Figure 1, the MIN is a three-stage Clos network[7], in which each stage consists of sixteen four-by-four switching elements. Each computing node contains a SIGMA Network Interface(SNI) to connect its bus interface to a port of the MIN.

3.2 SIGMA computing nodes

Each computing node consists of a CPU module and some optional I/O modules. Connection between modules is via a standard I/O bus, and thus a vast array of commodity I/O adapters can be used in the node[10]. Each node is physically separated from each others. Communication between nodes is achieved by

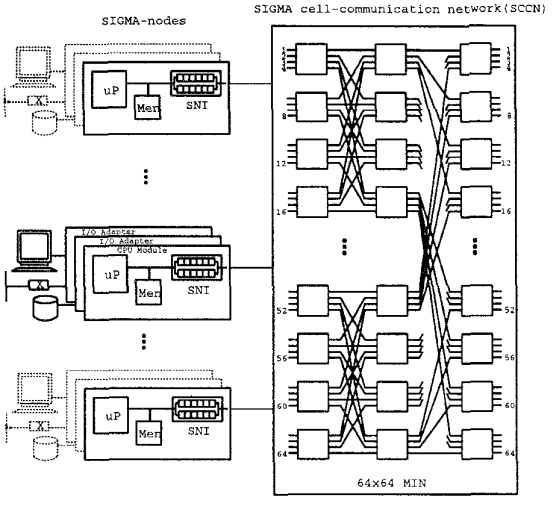

Figure 1: The SIGMA multicomputer architecture - A 64-node example

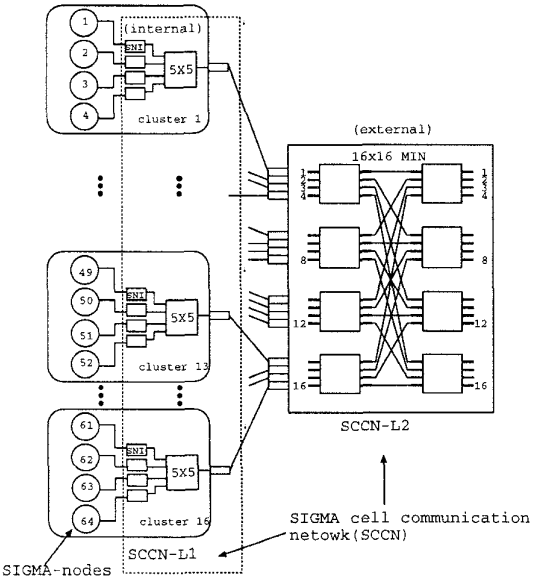

Figure 2: A configuration of SIGMA multicomputer system with 64 nodes

message passing through the internal network subsystem. Messages can also be delivered through conventional LAN(Local Area Network) facility if LAN devices are plugged into the nodes. The distributed nature of the architecture allows the system to survive device failures. All nodes are not necessary to be the same. Heterogeneous nodes can be in the system. Although most of system devices are separated located, system resources can be shared effectively through the communication facilities, the MIN or the plugged-in LANs.

3.3 SIGMA cell communication network

The network subsystem, SIGMA Cell Communication Network(SCCN), consists of two major parts: SIGMA network interface(SNI), and a cell switching network. A message is chopped into small fixed-sized data entities(cells), before it goes into the switching network, and the cells are reassembled at the destination nodes. The SNI take charges of: 1. network protocol conversion (data partition/reassembling), 2. data buffering, 3. cell header checking/generating, 4. network link serialization/de-serialization, 5. cell retransmission and link level flow control. In case of transmitting, packets are injected into SNI through the bus interface. Then, they are converted into cells and stored in cell transmitting buffer. As soon as cells go into the buffer, they will be fetched out and serialized for sending through the network immediately, cell by cell, whenever the requested channels are available. Upon receiving, similar operation steps in reverse direction will be performed on the cells. To overlay computing (protocol processing) and communication (cell sending/receiving), we use dual port RAM as cell buffer in the SNI. Also, we allow transmitting buffer and receiving buffer be manipulated concurrently. In addition, SCCN is a self-routing network based on the destination information carried in the cell header. Cells in the network can therefore be routed individually.

3.4 Cluster-based multicomputer system

The SIGMA multicomputer system is designed not only for parallel computing, but also for interactive computing. Therefore, each node occupies larger spaces than that in MPP(Massively Parallel Processor) systems. It is hard to put too many nodes all together in a PCB board. One common solution is to partition nodes into several clusters. In SIGMA multicomputer system, each cluster consists of several processor nodes and a on-board interconnection network with an architecture similar to that shown in Figure 1 but with fewer stages. Figure 2 shows an example of partitioning a system of sixty-four nodes network into sixteen clusters, with four nodes in each cluster.

3.5 Features of SIGMA multicomputer machine

Some features of the SIGMA multicomputer make SIGMA machine feasible to run a web server, although it can be applied to other applications as well. First, the system is expansible (scalable). To meet huge system resource demands of large scale web servers, size of a SIGMA machine is allowed to be incrementally increased. Upgrading of SIGMA system can be made on module-by-module basis. For instance, one can simply insert one CPU module to enhance computing power, instead of adding a whole computer(like PC/Workstation) to the system as it is needed in the case of PC/workstation clusters with conventional LAN interconnection. Besides, customized in-

terconnection network of the SIGMA machine provides higher bandwidth and better system resource sharing. Second, the system allows concurrent I/O operations. Different from scientific computing, Web service is more I/O-oriented, especially in disk I/O and networking. The SIGMA machine is a share-nothing architecture. Each node of it would be attached to disk modules and network modules. Consequently, it provides not only large aggregated computing power and system memory, but also large bandwidth of disk and network I/O. In addition, design of the SIGMA interconnection network also provides efficient communications to facilitate concurrent I/O.

3.5.1 hardware flow control

Flow-control supported at the hardware level contribute to the fast message-passing communication in SIGMA. It prevents data loss due to receiving buffer overflow (in hubs or in destination nodes). For a connection-oriented communication, any data loss would require re-transmission of the packet. This would waste bandwidth and cause significant communication delay. Although higher level flow-control protocols such as TCP/IP window-based flow-control can also alleviate the problems, it incurs larger overhead, and moreover it "avoids" the data loss problem, but not guarantees to "prevent" the problem from happening.

3.5.2 cell-switching communication

Another important feature is the cell-switching. Cell size in SIGMA is fixed at 64 bytes long. Four bytes of the cell is designated as cell header; two of them are hardware header, and the other two are cell adaption layer header. Sixty bytes of data payload can carry a complete ATM cell (53 bytes) or a minimum length of IP packet over Ethernet(60 bytes) which covers large portion of small control packets(e.g., ICMP, ARP, RARP packets) used in TCP/IP protocols. Sixty-byte packet fits one SIGMA cell without any waste, while it would need two ATM cells to carry such a packet. Also, we support multicasting in the hardware. Current version of SIGMA cell-switching network can achieve multicasting within a cluster (four nodes), and broadcasting(to all nodes) in the system. To respond to a urgent packet quickly, an emergency bit in the cell header can be set and will be identified by the hardware for immediate processing. Other benefits from cell switching versus packet switching are summarized as follows:

- simple architecture: Comparing with variable length(packet-based) architecture, cell-based architecture is simpler. Simplicity of the architecture results in better performance. It not only simplifies control logics but also eases the management of random access buffers.
- latency improvement: Small packet can interleave with large packets. As an example shown in Figure 4, small packets like *p2 and p5* can be sent out quickly by interleaving with other large packets. These packets may be blocked by the large packets in case of packet-switched communication. in case of pack. Furthermore, transmission of large packets can benefit from worm-hole effects of the network as shown in Figure 3.

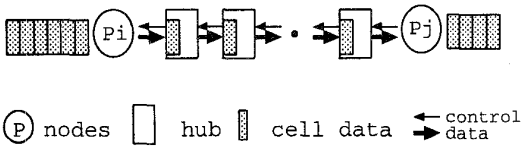

Figure 3: Worm-hole effects of transmitting a packet across multiple switching hubs

- more predictable transmitting time: Characteristics of cell-interleaving(Figure 4) in SIGMA network subsystem make it more like a TDMA (Time Division Multiple Access) network where network bandwidth is divided into a set of time slots. Consequently, the time to transmit a S-byte packet can be limited to $N*S/B$, where N is the number of nodes, and B is network bandwidth. Bounded transmitting time is important for real-time applications such as providing real-time video/audio streams in Web servers.

3.5.3 Cell pre-sink

To reduce communication latency, a cell *pre-sink* scheme was applied, which asks all nodes in a cluster receive(sink) all cells transmitted to the cluster in advance before the cells are determined which nodes they should go to exactly by the routing logics. Once the routing tags of the cells are resolved, all nodes in the clusters will be notified if they are the right destinations. If yes, it continues to receive the rest of the data segments of the cells; if not, it just flushes the pre-sink data of the cells and gets ready for next cells. As a result, data can be sent at a full speed without any delay due to routing tag processing.

4 Software Configuration of the Web Server on SIGMA

Since the SIGMA multicomputer is flexible in I/O device arrangement, it allows a large variety of software configurations for a web server. We propose a configuration based on the world-wide web's run-time behavior to take advantages of SIGMA architecture.

The proposed software configuration of our web server is shown in Fig. 5. It is composed of several manager groups, each of which consists of several computing nodes. Number of nodes in each group depends on the workload of the web server and can be scaled up or down when it is necessary. Note that a computing node may run more than one kind of mangers at the same time. Communication between two managers is

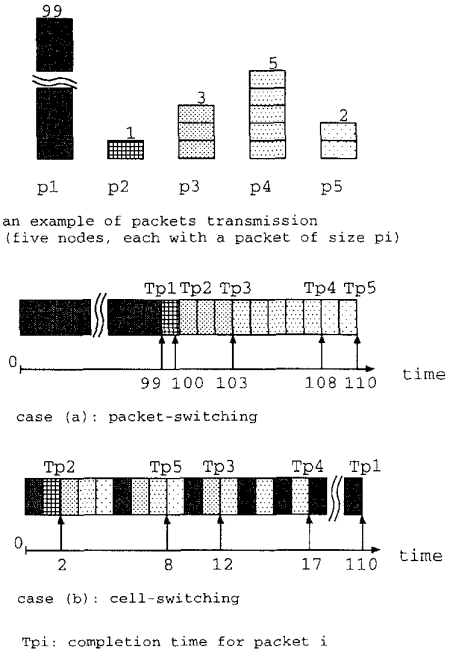

Figure 4: Cell interleaving

via the SIGMA SCCN switching network. However, request managers, stream managers and proxy managers, may also have connection to external LANs. Caching can be done more effectively with this architecture since each computing node is assigned with a specific task (or tasks) and it is easier to design effective caching schemes based on locality properties of individual tasks.

4.1 Load sharing among request managers

While a request arrives at the server, it is received by a request manager. Instead of using a single request manager, multiple request managers are asked to receive requests simultaneously. A distributed decision method is used to distribute workload evenly among them. In our design, all request managers share the same IP address. When a request packet arrives at a computing node where a request manager is running, the low-level network module will peek off its source IP address, and then apply a distributed decision algorithm to determine whether to receive the packet or not. It will be accepted directly by one of them and rejected by others. The distributed decision algorithm is implemented in the driver. So when an HTTP request is forwarded up to the high-level request manager, it's destined. Only one request manager will receive the request, others will not even see the request. This method reduces the overhead of the computing nodes. Note that, in the connection with broadcast networks

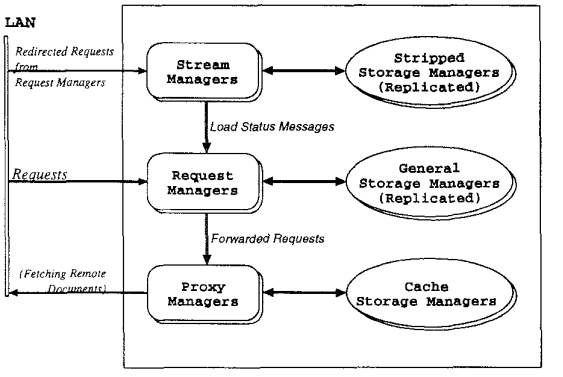

Figure 5: Software configuration of a world-wide web server on SIGMA multicomputer

such as Ethernet, the computing nodes sharing the same IP will also share the same physical network address.

4.2 Load redistribution

Three load redistribution strategies are used in our design to improve the web server performance. The first one is for the request manager to directly serve the request locally, the second is to redirect the request to another one, and the last is asking another computing node for help. These three strategies are applied to three different kinds of requests of different natures. Requests for small documents are better be served directly because the documents are often cached in the request manager's memory buffers. Requests for large files such as video streams usually take longer time to process, and processing of some requests will need to generate a virtual document such as database query results. Request managers will redirect the client sending these kinds of requests to some stream manager for help. This approach will alleviate the workload of request managers and improve the availability. The third strategy is good for proxy requests. Most current browsers are not able to redirect proxy requests. Moreover, it's difficult to transparently create another TCP connection between the client and another node to replace the existing connection (between the client and the request manager). Thus, a request manager must handle all proxy requests (requests to remote sites) by itself. This is fine if the requested documents are cached. If not, the request manager will create an extra TCP connection to the remote server to fetch the document. Usually, these two connections may stay for a long time. TCP connections are valuable resources in a web server, and should be used more effectively. To solve this problem, a request manager will route the request to a proxy manager if a local copy is not available. If, fortunately, the requested remote document is cached in the proxy manager, it will directly return the document to the request manager. Otherwise, it is the

proxy manager's duty to fetch the remote document. The remote IP address is used as the hash key so that all proxy requests to the same remote site are handled by the same proxy manager. Furthermore, the proxy manager can prefetch or clean the cached documents from the hyperlink information of the documents for optimization. By carefully applying these three strategies, the machine's performance and availability could be enhanced optimistically.

4.3 Video stream service

If all requests to the same video are served by the same stream manager, it will have better locality. However, this may cause a problem. when a popular video program solicits a large number of requests in the same period. The load will not be distributed evenly among the stream manager groups. So, the hashing algorithm in a request manager will first redirect all requests for the same video program to the same stream manager, and each stream manager will periodically broadcast its loading status to the request managers. Once the number of requests to a stream manager is larger than a threshold, request managers will then take the source IP address into consideration to choose a second stream manager. In addition to supporting load distribution, a status message can also act as a probe message to check whether a request manager or a stream manager is still alive. Status broadcasting can be implemented by the low-level multicast mechanism of the SIGMA switching network.

In SIGMA, accessing remote memories is faster than accessing local disks. To improve the performance of video stream service, a large file is partitioned into pieces and stored in a set of stripped storage managers. While a video file is requested, all of the stripped storage managers will access their local disks concurrently and then return the video program to the agent via a single stream manager. Each stripped storage manager simply keeps pieces of the file and they can access disks concurrently. So the total time to request a whole video file can be reduced.

4.4 Storage managers

There are three kinds of storage managers in the system, namely cache storage managers, general storage managers, and stripped storage managers. They work together with proxy managers, request managers, and stream managers respectively, and maintain remote documents, local files, and video files correspondingly. A SIGMA computing node running one of these managers should have local disks attached.

Except for cached remote documents, local documents and video files are replicated so that serving requests can continue even when disk failures occur. Some of the reasons not to replicate remote documents are:

1. Remote documents can be fetched again later after the cache storage manager recovers from crashes.

2. Leaving more disk spaces for caching other remote documents results in better cache hit rate.

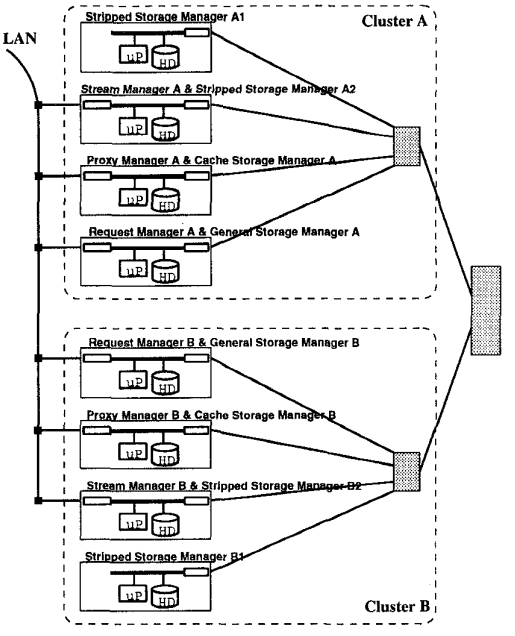

Figure 6: An implementation of the proposed world-wide web server on an eight-node SIGMA multicomputer

3. Replicating files dynamically incurs some overhead.

At present, replicating local files and video files is done by a simple file duplication scheme since web files are rarely modified by requests and inconsistency during the period of file transfer can be tolerated.

5 Implementation

In this section we present an implementation of the proposed world-wide web server on an eight-node SIGMA multicomputer. The server is composed of two mirrored clusters as shown in Fig. 6. Request managers, proxy managers and stream managers run in association with corresponding general storage managers, cache storage managers, and stripped storage managers. There are two stripped storage managers in each cluster so that a large file can be segmented into two stripes. Local files are replicated in different clusters. Currently we are porting the reference library and the HTTP server released by the World-Wide Web Consortium onto our eight-node SIGMA prototype. Each node runs FreeBSD Unix. All nodes are also connected to an Ethernet LAN except for the ones dedicated to stripped storage managers.

Since the nodes running Request Manager A and Request Manager B have the same IP address and physical Ethernet address, all incoming packets destined to the shared IP address will reach both Ether-

net drivers. The drivers have been hacked to support distributed decision method. In the current implementation, they only accept the packets with the least significant bits of the IP address match their unique node ID. The number of bits to be checked depends on the number of request managers in the system. In this way, all incoming requests are dispatched to Request Manager A and Request Manager B intrinsically.

A request is routed to one of three manager groups, and there are two managers in each group that can accept and serve requests concurrently. A video request, on the other hand, will be served by two stripped storage managers concurrently. This arrangement can achieve high degree of load sharing and improve the system availability.

In addition to serving requests concurrently and accessing disks in parallel, request managers are designed to utilize time locality. Small general documents are requested more frequently, and they tend to be cached in the memories of request managers. Large files or video streams won't be cached by request managers; they are redirected to stream managers. Stream managers with replicated stripped storage managers are designed to explore space locality. Large disk block, sequential block placement and prefetching disk blocks for read are implemented in the file system of stripped storage managers.

The system can be scaled up in different ways to adopt the change of request distribution. If the total number of requests increases, more computing nodes can be added to run as request managers. If proxy service becomes heavy, we should add more computing nodes to run proxy managers and cache storage managers. If demands for video stream service increase, more computing nodes for stream managers and stripped storage managers are required. On the other hand, the system can also be scaled down gracefully when a manager or a cluster of nodes fails to work. The system will be able to continue to work with degraded performance.

6 Summary and Conclusions

In this paper we demonstrate that Internet computing is a suitable application of a multicomputer system. The proposed web server is composed of several groups of different managers. Each manager is arranged to run on one node of the multiprocessor clusters. Using a distributed decision method and hashing algorithms for dispatching request and redistributing workload, computing nodes can share workload, concurrently handle requests, tolerate single point of failure, and act as a virtual server transparently to clients. By optimizing the processing of the three kinds of requests independently, performance of the web server can also be improved significantly, even for video streams and caching remote documents. Thus, we believe that a scalable multicomputer like ours is suitable for serving as a web server.

References

[1] M. Abrams, C. R. Standridge, G. Abdulla, S. Williams, and E. A. Fox, "Caching Proxies: Limitations and Potentials," *Fourth International World Wide Web Conference*, 1995.

[2] T. Berners-Lee, R. Cailliau, J. Groff, and B. Pollermann, "World-Wide Web: The Information Universe," *Electronic Networking: Research, Applications, and Policy*, Vol. 1, No.2, 1992.

[3] T. Berners-Lee, R. Fielding, and H. Frystyk, "Hypertext Transfer Protocol - HTTP/1.0," *Internet Draft*, Nov. 1995.

[4] H. Braun, and K. Claffy, "Web Traffic characterization: an assessment of the impact of caching documents from NCSA's web server," *Second World Wide Web Conference*, Oct. 1994.

[5] R.J. Clark, and M.H. Ammar, "Providing Scalable Web Service Using Multicast Delivery," *Second International Workshop on Services in Distributed and Networked Environments*, 1995.

[6] E.D. Katz, M. Butler, and R. McGrath, "A Scalable HTTP Server: The NCSA Prototype," *Computer Networks and ISDN Systems*, Vol. 27, No. 2, 1994.

[7] Y. J. Lin, J. M. Ho, C. C. Yeh, and J. Y. Juang, "Design of a Switching Module for Large-Scale ATM Switch," *International Conference on Parallel and Distributed Systems*, Taiwan, pp. 399-408, 1993.

[8] A. Luotonen, and Kevin Altis, "World-Wide Web Proxies," *First International World Wide Web Conference*, 1994.

[9] Network Wizards, "Internet Domain Survey," *URL:http://www.nw.com/zone/WWW/top.html*, January 1996.

[10] C. C. Yeh, J. T. Lin, W. C. Kao, C. H. Wu, and J. Y. Juang, "A Multicomputer Server for I/O-Intensive Applications," *12th IASTED International Conference on Applied Informatics*, Austria, 1995.