Enabling Content-Based Load Distribution for Scalable Services

Guerney Hunt, Erich Nahum, and John Tracey
IBM T.J. Watson Research Center
Yorktown Heights, NY 10598
{gddh,nahum,jtracey}@watson.ibm.com

Abstract

We argue that using *information content* of individual client requests to distribute load across a cluster of workstations or PCs has many attractive properties, and describe how current Internet standards make content-based load distribution difficult. We discuss several approaches to achieve content-based load distribution for large scalable services such as WWW servers. We illustrate the advantages and disadvantages of these approaches and introduce one method, *connection hand-off*, which enables content-based load distribution. We describe a small backwards-compatible extension to the TCP protocol to allow connection hand-off and outline the OS implementation required.

1 Introduction

Networked information systems have seen explosive growth in the last few years. The information available via the global information infrastructure is growing rapidly as text-only information sources are augmented with voice, video and still image data. This growth places increasing demands on large scale information servers, which provide services such as digital libraries, video-on-demand, World-Wide Web and high-performance file systems. In addition, a single server may provide responses for potentially *thousands* of clients. If a server is connected to the global Internet, capacity planners may not even be able to accurately estimate the demand for it, since an unknown (and growing) number of clients may be requesting service. For example, Netscape's home page server currently receives over *one hundred million* requests a day [9].

Cluster-based computing has received a large amount of attention recently[2], both in the academic and commercial communities. Clusters attempt to replace an expensive high-end machine with a collection of cheaper workstations or PC's. The increasing performance and shrinking price of these machines make clusters an attractive alternative to traditional mainframes and supercomputers.

Clusters can be used as cost-effective scalable servers, for providing WWW, video, file, or other services. Indeed, several products have been released that make a cluster appear as a single server, establishing a single point-of-presence on the network at a particular IP address. This allows distributing requests across the cluster in a way that is transparent to clients.

Several approaches or *mechanisms* may be used to enable this distribution. In addition, many *policies* may be used to balance load across a cluster. Ideally, requests should be distributed in such a way as to maximize the number of requests that can be serviced by the cluster, or to minimize the response time as observed by a client. However, as we will demonstrate, the mechanism used to distribute the requests can limit the choice of policies.

In this paper, we argue that exploiting *information content* in client requests can greatly improve load distribution, using a cluster-based WWW service as an example application. We describe several approaches to distributing WWW requests across a cluster of machines, and show their limitations for exploiting information content. We propose a novel mechanism to migrate connections that enables content-based load distribution. We describe a backwards-compatible option to the TCP protocol [24] that allows migrating an established connection, and outline the OS changes required to implement this feature.

2 Mechanisms for Scalable WWW Servers

In this Section we outline several possible mechanisms for using clusters as scalable WWW servers, and describe the characteristics of each.

One possible mechanism is using a cluster with a single system image (SSI) or distributed shared memory (DSM)

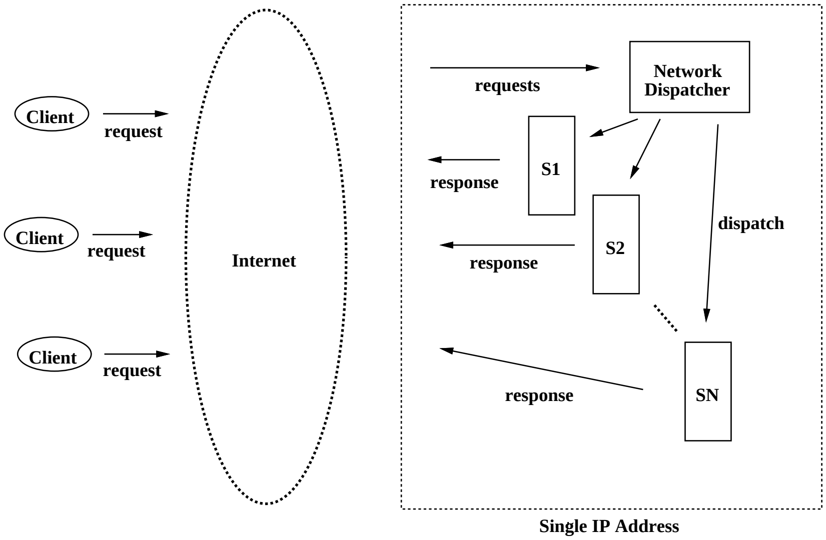

Figure 1: Connection Routing

[17, 20]. A great deal of research has been done in DSM, and Sun has demonstrated an SSI prototype [16]. The advantages to this approach are that it provides fine-grained sharing and load balancing across a cluster, and is transparent to the client. The disadvantage is that it requires support on the server system, and typically requires the cluster to be homogeneous. An attractive feature of cluster-based servers is that they can be grown incrementally as needed, preserving a customer's investment in hardware and software. This can lead to a heterogeneous environment, as machines that were most cost-effective at one point in time are augmented by machines that have the best current price/performance. Thus, we believe that heterogeneity limits the utility of DSM-based clusters as scalable servers.

Another mechanism is called *connection routing* [12]. In this approach, a front-end machine, called the *network dispatcher*, routes incoming requests to server nodes. When a TCP connection request arrives, the dispatcher determines which server in the cluster to forward the packet to, usually based on an estimate of load on each server. The dispatcher routes the packet to the server, and adds an entry in its internal state that allows it to forward subsequent packets for that connection to the same server. The dispatcher must process all external client packets entering the cluster, but need not see packets from the server to the client. Several announced products use this or similar methods [8, 10, 13, 18, 25], which is illustrated in Figure 1.

IBM's product is used for the main IBM web site (www.ibm.com) and was used for the Summer 1996 Olympic Web Server [3, 7]. In IBM's implementation, the network dispatcher includes a manager which is used to implement load balancing policies. The network dispatcher uses a weighted round-robin policy for the distribution of requests. The manager monitors metrics which indicate the load on the servers. These distribution weights are adjusted by the manager to keep the load balanced on the servers in the cluster. How metrics are combined is user configurable.

Connection routing has compelling advantages: it is transparent to both the client and the server, and it scales well with the number of servers in the cluster. For example, HTTP requests for the Olympic Web Server were distributed over a cluster of 56 machines. One disadvantage is that, because of the nature of TCP and HTTP, the connection must be established between the client and server node before the *type* of request can be ascertained. This precludes using the content of the request as a factor in making a connection routing decision. TCP was designed as a bulk data transfer protocol, and the problems of using it for request-response communication are well-known [22, 23]. Specifically, TCP requires a three-way handshake to establish a connection, in order to detect duplicate packets and reject spurious connection requests, and data typically does not flow across the connection until after the handshake completes. Although other connection management mechanisms are possible, replacing TCP is extremely unlikely, given the large (and growing) investment in the Internet, and in critical applications such as the WWW that use TCP as their transport protocol.

This leads to a third approach, which is to use Transactional TCP (T/TCP) [5, 6], an option in TCP designed to improve the protocol's usefulness for request-response communication. T/TCP uses a timer-based connection management mechanism that enables data to be sent in the first packet along with the SYN bit. A network dispatcher for a cluster-based WWW server could examine the request in the data to use information content in its routing decision, before the connection is actually established. Unfortunately, this approach requires modifying both the client and the server to use T/TCP. This is likely to hinder deployment, especially since the transport protocol is typically implemented in the kernel. While individual vendors and users can upgrade their servers and browsers to take advantage of T/TCP, a large scale web server cannot *rely* on the presence of clients using T/TCP to make its load balancing decision.

This leads us to introduce a novel mechanism, which we call *connection hand-off*. In this approach, the network dispatcher completes the connection establishment with the client *before* deciding to which server to forward the request. Once the three-way handshake has completed and the client has sent the request, the dispatcher can examine the request to aid its routing decision, enabling content-based load distribution. By adding an option to the TCP protocol specification, we can enable connection hand-off *in a way transparent to the client*. Thus, only the server cluster needs to be modified, and clients are unaffected. We describe the connection hand-off mechanism in more detail in the Section 4.

3 Content-Based Load Distribution

In this Section we describe how load distribution policies could exploit information content of client requests.

The connection routing approach described in the previous Section generally routes requests to servers based solely on an estimate of current load on each server. The next logical step is to further refine the routing decision based on the content of the request, which until now has been ignored. Exploiting this has several advantages:

- Requests to the same page can be routed to the same server, since that server is likely to have that page cached. If a page is extremely popular, a subset of the servers can be used to distribute load while maximizing the cache usage.
- Recent work has shown that *spatial locality* exists in WWW reference streams [1]. A dispatcher may be able to exploit this locality by forwarding related streams to the same server node, again improving cache utilization.
- The content type may used to forward to servers specialized for the content type [15]. For example, requests for video clips may be sent to a server with an operating system and file system customized for video playback.

In essence, we are allowing for a finer granularity of distribution of requests. Other methods, in particular the connection routing approach, require that all the services associated with an IP address and port number be homogenous with respect to what they offer. In other words, they require all nodes in the cluster to be capable of providing the same content. Distribution based on content eliminates this requirement and allows servers to be heterogeneous.

The *policy* as to exactly how to route based on content is a research topic that remains to be addressed, but it is clear that a mechanism which enables routing based on content allows choosing from a larger set of policies.

4 Migrating Established Connections

In this Section we describe the mechanism for doing connection hand-off in detail.

Conceptually, we wish to have the network dispatcher for the cluster hand off the connection to an individual server machine in a way that is transparent to the client. We want to perform the hand-off after the connection has been established, so that the URL can be examined to distribute load based on information content, but before any transaction occurs. Since an exchange of bytes may have almost any meaning (e.g., executing a CGI/bin program), an arbitrary change of state can occur at the server (such as debiting a bank account). It is infeasible to transfer this state without a large process migration or DSM mechanism; thus it is difficult transfer a connection at any arbitrary point in time.

However, until the client receives any bytes in response to a request, or even an acknowledgment of the bytes sent, the client cannot infer that any action has been taken in response to the request. We take advantage of these TCP semantics, and propose to transfer the connection right at the point the connection has been established, but before any data or acknowledgments have been received by the client. We can do this by adding an option to the TCP protocol.

An initial TCP connection request, as received by the network dispatcher contains the following information:

- Source and destination port numbers
- The SYN control bit set
- An initial send sequence number (ISS)
- A flow control window
- Possible options (e.g., segment size)

The response seen by the client contains the following:

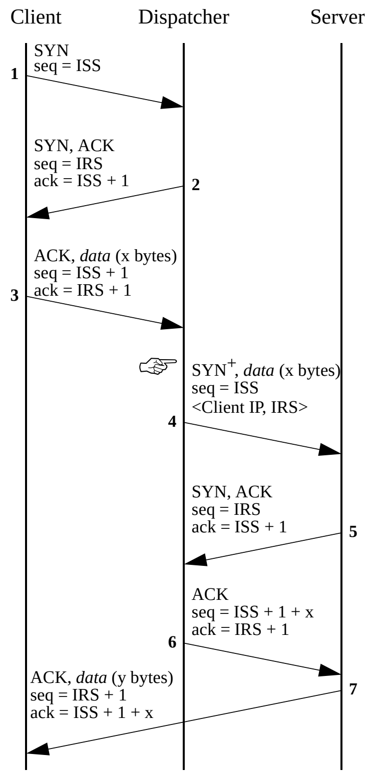

Figure 2: TCP Connection Handoff

- Destination and source port numbers
- The SYN and ACK control bits set
- An initial receive sequence number (IRS)
- An acknowledgment of the client's ISS
- The server's flow control window
- Possible options (e.g., segment size)

The client, in turn, sends a packet acknowledging the server's receive sequence number (IRS), completing the three-way handshake. Once the first data packet is received by the dispatcher, the data can be examined to allow content-based load distribution, and then the connection hand-off can be performed. The key idea here is that the individual server node can masquerade as the dispatcher as long as it has the appropriate connection state. Almost all of this state is contained in the original SYN packet; the exception is the initial receive sequence number (IRS) that the dispatcher has selected. To transfer the state, the dispatcher forwards the original SYN to the selected server node, with one addition: a connection hand-off option that contains the IRS selected by the dispatcher and the IP address of the client.

Figure 2 shows the mechanism of the handoff in detail. The numbers correspond to events in the handoff, proceeding as follows:

- The client establishes a connection with the dispatcher, using a three-way handshake as before, as shown in Figure 2 with messages 1, 2, and 3.
- The dispatcher forwards a SYN packet to the server node, illustrated with message 4. The SYN is similar to the SYN sent by the client, except that it uses the dispatcher's address as the source IP address, includes the client data received, and contains an option which has the IP address of the client and the IRS selected by the dispatcher.
- The server node goes through the normal connection establishment process to the dispatcher, except that it uses the passed IRS, rather than selecting its own. The server node sends a SYN/ACK packet to the dispatcher, as shown by message 5, notifying it that the server has accepted the connection.
- The dispatcher completes the three-way handshake, illustrated with message 6, notifying the server it is completing the hand-off. The dispatcher frees any extra state associated with the connection, such as buffered data, and subsequently acts in the same fashion as the original connection router.
- The server node changes its connection state to use the client's IP address rather than the dispatcher's, and responds to the the request directly, as shown by message 7, sending any resulting data to the client and acknowledging the receipt of the request.

After this point, the client and server perform the normal TCP data transfer and connection termination. The only change in the TCP protocol state is to understand the connection hand-off option, and to complete the handshake with the dispatcher rather than the client.

Attention must be paid to several subtleties that arise in connection hand-off due to the TCP protocol. For example, the dispatcher cannot accept any options (such as the selective acknowledgment or SACK option [21]) from the client that the server nodes are not capable of understanding. Similarly, since the TCP protocol specification requires that an advertised flow control window can never be retracted, the dispatcher must choose an initial flow control window that each server can satisfy.

With these changes, the network dispatcher can be modified to allow additional load balancing policies that can take advantage of information content. When content is moved, content is added, or new servers are added to the cluster, the dispatcher must be notified so that the correct load distribution decision will be made.

Resonate Inc has announced a product [14] which performs 'connection hop,' which may be similar to our approach. However, they do not describe the mechanism by which connections are transferred; their system could use a single system image.

5 Summary

In this paper, we described the current state of the art in building scalable web servers using clusters and illustrated the utility of distributing load based on information content. We described a mechanism to hand off active connections and how this mechanism enables routing based on content. We have begun implementing the protocol and OS changes described, and plan future research evaluating different policies for content-based load distribution.

References

[1] Virgilio Almeida, Azer Bestavros, Mark Crovella, and Adriana de Oliveira. Characterizing reference locality in the WWW. In *Proceedings of PDIS'96: The IEEE Conference on Parallel and Distributed Information Systems*, Miami Beach, Florida, December 1996.

[2] Thomas Anderson, David Culler, and David Patterson. A case for NOW (networks of workstations). *IEEE Micro*, 15(1):54–64, Feb 1995.

[3] Hari Balakrishnan, Srinivasan Seshan, Mark Stemm, and Randy Katz. Analyzing stability in wide-area network performance. In *Proceedings of the ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems*, Seattle, WA, June 1997.

[4] D. Borman, R. Braden, and V. Jacobson. TCP extensions for high performance. Request for Comments (Proposed Standard) RFC 1323, Internet Engineering Task Force, May 1992.

[5] Robert Braden. Extending TCP for transactions – concepts. In *Network Information Center RFC 1379*, November 1992.

[6] Robert Braden. T/TCP – TCP extensions for transactions functional specification. In *Network Information Center RFC 1644*, July 1994.

[7] IBM Corporation. 1996 summer olympic web site. http://www.atlanta.olympics.org.

[8] IBM Corporation. IBM interactive network dispatcher. http://www.ics.raleigh.ibm.com/ics/isslearn.htm.

[9] Netscape Communications Corporation. 2.9 million visitors propel netscape internet site to over 100 million hits a day. http://www.netscape.com/newsref/pr/newsrelease238.html, September 1996.

[10] Rad Network Devices. Web server director. http://www.radinc.com.

[11] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, and T. Berners-Lee. Hypertext transfer protocol – HTTP/1.1. In *Network Information Center RFC 2068*, January 1997.

[12] German Goldszmidt and Guerney Hunt. Scalable servers using the TCP connection router. Technical report, IBM T.J. Watson Research Center, Hawthorne, NY, 1997. In preparation.

[13] Cisco Systems Inc. LocalDirector. http://www.cisco.com.

[14] Resonate Inc. Resonate dispatch. http://www.resonateinc.com.

[15] M. Frans Kaashoek, Dawson R. Engler, Gregory R. Ganger, and Deborah A. Wallach. Server operating systems. In *1996 SIGOPS European Workshop*, Connemara, Ireland, Sept. 1996.

[16] Yousef A. Khalidi, Jose M. Bernabeu, Vlada Matena, Ken Shirriff, and Moti Thadani. Solaris MC: A multi computer OS. In *Proceedings of the USENIX Annual Technical Conference*, pages 75–85, San Diego, CA, Jan 1996.

[17] Povl T. Koch, Robert J. Fowler, and Eric Jul. Message-driven relaxed consistency in a software distributed shared memory. In *Proceedings of the First USENIX Symposium on Operating Systems Design and Implementation*, pages 75–85, November 1994.

[18] F5 Labs. BIG/ip. http://www.f5.com.

[19] Samuel J. Leffler, Marshall Kirk McKusick, Michael J. Karels, and John S. Quarterman. *The Design and Implementation of the 4.3BSD UNIX Operating System*. Addison Wesley, Reading, Massachusetts, 1989.

[20] Kai Li and Paul Hudak. Memory coherence in shared virtual memory systems. *ACM Transactions on Computer Systems*, 7(4):321–359, Nov 1989.

[21] Matthew Mathis, Jamshid Mahdavi, Sally Floyd, and Allyn Romanow. TCP selective acknowledgment options. In *Network Information Center RFC 2018*, October 1996.

[22] Jeffrey C. Mogul. The case for persistent-connection HTTP. In *ACM SIGCOMM Symposium on Communications Architectures and Protocols*, Cambridge, MA, August 1995.

[23] Jeffrey C. Mogul. Network behavior of a busy web server and its clients. Technical Report 95/5, Digital Equipment Corporation Western Research Lab, Palo Alto, CA, October 1995.

[24] Jon Postel. Transmission Control Protocol. *Network Information Center RFC 793*, pages 1–85, September 1981.

[25] HydraWEB Technologies. HydraWEB. http://www.hydraweb.com.

[26] Gary R. Wright and W. Richard Stevens. *TCP/IP Illustrated Volume 2*. Addison Wesley, Reading, Massachusetts, 1995.