RC 21881 (98348) 9 October 2000 **Computer Science**

IBM Research Report

A System for Resource Sharing and Control of a Cluster-Based Web Server Farm

Richard P. King, Junehwa Song, Daniel M. Dias
IBM Research Division
Thomas J. Watson Research Center
P. O. Box 704
Yorktown Heights, NY 10598

A System for Resource Sharing and Control of a Cluster-Based Web Server Farm

Richard P. King, Junehwa Song, Daniel M. Dias
IBM T.J. Watson Research Center,
P.O. Box 704
Yorktown Heights, NY 10598

Abstract

Web hosting services provide access to their server farms on behalf of large numbers of different customers. The users who connect to the Web sites of those customers drive the comsumption of a variety of resources, like network bandwidth, that are shared by the entire server farm, sometimes to the advantage of one customer's Web site and the detriment of others. A system has been devised that uses a front-end controller to monitor and throttle traffic, as necessary, to avoid such problems. Once contention for resources is detected, the controller uses admission controls to avoid overloads. It will also slow down delivery of data on individual network connections, if necessary.

Network traffic passes through this controller only when it arrives at the server farm, not when it leaves, and the control system requires no changes to the software on the servers themselves. These factors create problems in establishing control of the traffic, and the servers. At the same time, they reduce the load on the front-end controller and its performance impact on the system as a whole.

1 Introduction

With the increasing number of Web sites, the use of Web hosting services becomes more and more popular. Service providers host and manage a large number of Web sites for different customers. The customers are freed from the technical details required to establish and maintain their own Web sites and rely on the service providers to handle those details. From the service providers' point of view, maintaining separate hardware and software systems for individual customers is not cost-effective. They would rather establish a large system, referred to as a *Web server farm*, which can be shared by the Web sites of multiple customers. Such a high performance Web server farm is often constructed as a cluster of many smaller-sized systems for management- and cost-effectiveness.

A Web hosting service will generally be provided on a contractual basis. Each customer and service provider agree on and generate a service level contract. A core part of this service contract is a guaranteed level of performance. For example, customers may request a certain level of server and/or network bandwidth to be guaranteed for their Web sites. In other cases, they may want a certain number of users to be allowed to access specific applications on their sites at the same time.

While advantageous in many ways, providing a Web hosting service on a cluster-based Web server farm gives rise to new challenges in system management. Most of all, it should support dynamic sharing of the resources while enforcing the service level agreement. For this, it is necessary to collectively monitor and control the resources spread over different component systems. In this paper, we describe a resource monitoring and controlling system for a clustered Web server farm.

There have been many research efforts to provide guaranteed services on the Internet [4, 6, 14, 19, 25]. Our work makes an additional contribution to these efforts by providing control for resource sharing on the front end of a clustered system. Our system monitors the load on back-end servers by intercepting incoming packets at a front-end node. It throttles traffic to the servers by delaying or rejecting packets depending on the load and the resource utilization. Such front-end control makes it possible to collectively manage resources across different servers. This approach also avoids unnecessary waste of system resources, since the control decision is made at an early stage.

There are challenges in developing a front-end controller. Most of all, the control system suffers from lack of information. Since the controller resides on a separate node from back-end Web servers, it is hard to acquire the exact knowledge of resource utilization. In addition, the controller cannot extract detailed knowledge about the contents of packets or the connections because packets are intercepted before they fully go through the TCP/IP stack.

The rest of the paper is organized as follows. Section 1.1 delineates the contributions of this paper and describes its relationships to earlier works. Section 2 describes an architecture for a cluster-based Web server farm and front-end resource control. In section 3, we describe the policy for bandwidth sharing. In Section 4, we describe the system implementation.

1.1 Contributions and Relationship to Earlier Work

The main focus of our work is to provide a front-end resource controller for a cluster-based Web server farm. It should be able to globally manage the resources in a cluster. The performance requirement is specified per groups of connections to Web servers or applications (called a service group, which will be defined later). The resource controller should be able to increase resource utilization while guaranteeing the specified minimum level of performance for each service group.

In the context of a group of TCP flows, a *leaky bucket* rate control method has been proposed in [19]. It proposes a way to share a common token bucket among multiple TCP flows controlling and sharing the aggregate bandwidth within a group. Our study is different in two aspects. First, the method in [19] is a source-based control, which assumes detailed knowledge of the application and that of the TCP flow is available. On the contrary, our objective is to provide a controlling system which can be put on the front-end of a large, cluster-based web server farm. Source based control on each server (component node in a cluster) is not sufficient to orchestrate the bandwidth across the cluster system. Second, while the work in [19] focused on guaranteeing the aggregate bandwidth and fair sharing among each flow within a group, we have an additional objective of fair sharing of slack bandwidth across different service groups.

Several proposals have been made for rate-based control of TCP data flow. Application-based rate control [17] provides high accuracy, but at the cost of modification to the server code itself. This limits it to those servers whose code happens to be available, and requires additional coordination when different servers are used, and even more so in a server farm. Most relevant to our work is the ACK-based pacing in [14]. In [14], enforcement of a rate allocated for each flow is achieved by modifying the ACK number and receiver window fields in the ACK header and by modulating the ACK rate. While our objective is to guarantee the aggregate rate for a service group, a similar ACK-pacing is used for each flow within each service group.

Many research efforts have been put on measurement-based admission control [9]. The traffic models in that work are usually different from ours in that each flow has a bandwidth requirement, possibly time-variant. Therefore, the focus in that work is to guarantee the bandwidth requirement by limiting the number of admitted flows while increasing system utilization. In contrast, in our work, the admission decision of each flow is made in a much simpler way since each flow does not have a bandwidth requirement. We use rate-controlling mechanisms to guarantee the bandwidth requirement given to groups of flows.

2 Architecture

Figure 1 shows a Web server farm on a cluster system. Web hosting service providers maintain such a web server farm and host multiple web sites for different customers.

Each customer establishes a Service Level Agreement (SLA) with a service provider. This SLA specifies that the provider guarantees for a service a certain level of performance. Here, the unit of a service can be a group of connections from clients and is referred to as a *service group.* A service group may be composed of a whole Web site, a group of URL's, or a group of applications. For example, an e-commerce site from a company may provide two different services, one for special promotions and the other for products on regular sale. In this case, the company may specify in SLA two service groups on the same Web site with different levels of service guarantees. For example, it may want at least 10 clients to be able to access the service on regular sale and at least 20 clients on the promotion sale.

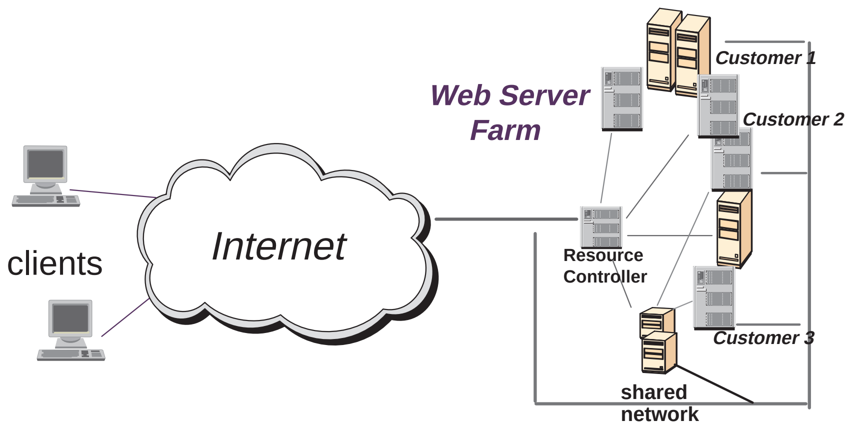

Figure 1: Web Server Farm on a Cluster System

There is also evidence that end users have expectations about what sort of performance a request should have, and that failing to meet those expectations can have a significant, negative impact on user satisfaction[1]. Requests perceived as more complex are granted greater leeway, while apparently simple ones are expected to be fast. Slow responses to requests buried in the middle of a sequence of requests leading to a purchase are especially annoying. Assignment of different types of requests to different service groups can therefore become critical to the success of a web site.

The service guarantee may be specified on different performance measures. Some of them specify the performance measures for bandwidth, such as guaranteed bandwidth per service group. Beyond this guaranteed level, more bandwidth, if available, can be allocated as a best-effort service. The SLA may specify different charges for guaranteed and best-effort services, with lower charges imposed, for example, for bandwidth that will only be provided at the convenience of the service provider. There is, as well, likely to be a maximum amount of bandwidth that the customer is willing to pay for. Server performance can also be specified. For example, a guaranteed connection rate may be set for a particular service group. As in bandwidth, best effort service can be specified when connection requests arrive at a rate higher than this guaranteed rate. This best effort service may further include a different priority level (e.g. higher priority to HTTPS). Also, the number of connections which it is guaranteed can be opened concurrently can be meaningfully included.

The service provider should assign resources (e.g. servers or communication link bandwidth) on the web server farm to host different web sites. This allocation can be done in different ways. First, each resource can be *statically* partitioned and each part can be dedicated to different web sites. For example, a back-end node can be dedicated to Customer 1's site and another to Customer 2's. In contrast, the resources may be partitioned and shared on demand among different web sites and applications without static partitioning and allocation. This is refered to as *dynamic* partitioning and allocation. For example, each request can be directed to a back-end node which is dynamically chosen depending upon the resource availability. Lastly, a mixture of static and dynamic allocation is possible.

Comparing the different methods, static paritioning and allocation is easier to implement. However, the level of system utilization it can achieve will be less than the other two methods. That is because the unused resource capacity allocated to a specific customer can not be used by any other customers even when they lack the same resources. Static partitioning may also be very costly, as it would be to provide separate incoming communication lines for each customer. In addition, once the partition and allocation are done, the system is not resilient to environmental changes. Whenever new customers are added or existing customers withdraw their contracts, the system should go through reconfiguration. Similarly, the system may go through gradual upgrades. It is also likely that even a steady customer will change the level of guaranteed service specified in the SLA.

2.1 Front-End Cluster Control

Resource control on a clustered Web server farm is possible at multiple points (Figure 2). Control actions can be taken on incoming data packets or outgoing packets. The control on the incoming packets can be made at the front-end controller or on each server. Similarly, actions on the outgoing data packets can be taken at either a server or a common node, if any, on the data return path.

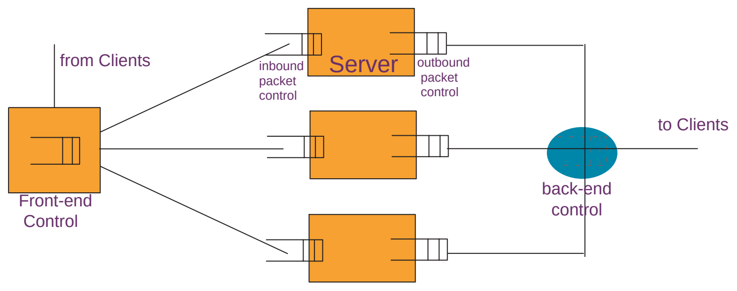

Figure 2: Control Points for a Web Server Farm

Control on the out-bound packets can achieve more accurate control than that on the in-bound packets. This is because by the time the out-going data packets are generated, the system has acquired all the related information for the requests, such as URL, data size, number of CPU cycles consumed, etc. However, it cannot be used to avoid an overload of resources, since it has already used up the required server resources. On the contrary, incoming packet control is effective to reduce the overload condition but lacks required information about the requests.

There have been many reseach efforts for the control on the server side; see, e.g., [19]. However, control on the servers is not sufficient to share resources on a clustered environment, since the resource sharing on a cluster system requires global knowledge and control of the system resource accross different servers. The focus of this paper is a front-end controller which makes possible such global orchestration of the system resources. Such a front-end controller is advantageous in avoiding unnecessary waste of resources because the control decision and action is made at an early stage of the system, before the data packets enter the servers and thereby consume any resources.

3 Bandwidth Sharing

Efficient bandwidth management while enforcing the SLAs in a clustered Web server farm results in two levels of bandwidth sharing. In the higher level, the farm's bandwidth is shared among different service groups. In the lower level, the different connections in the same service group share the bandwidth allocated to the service group. Therefore, separate bandwidth control mechanisms are provided for the two layers.

3.1 Sharing Among Service Groups: Macro-level Sharing

The sharing of bandwidth proceeds in three steps: forecasting future demand, allocating bandwidth among priority levels, allocating within priority levels. We now discuss each of those steps.

3.1.1 Demand forecasting

The traffic to Web sites has been successfully represented by an ARIMA model with daily, weekly, and annual trends [13]. We assume that our observation intervals will be short, on the order of a few seconds to, at most, a minute. This allows us to model the bandwidth demand during the upcoming observation interval as being equal to the demand during the previous interval plus some random noise ϵ.

Forecasting the traffic is thus reduced to estimating the distribution, or at least the moments, of ϵ. Although the expected value of ϵ should be near zero, it is estimated using an exponentially-smoothed moving average, in order to capture any short-term trends that the observation interval is not short enough to filter out. The variance of ϵ is not particularly convenient to estimate in a similar fashion. Instead, we have observed in various data sets that the standard deviation of ϵ is strongly correlated with $\max(0, \epsilon)$. We therefore use an exponentially-smoothed moving average of the $\max(0, \epsilon)$ as an estimate of the standard deviation of ϵ. Based on that, our estimate of the most bandwidth that a service group is likely to demand during the next observation interval is $\epsilon = \overline{\epsilon} + 1\frac{2}{3}\tilde{\epsilon}$, where $\overline{\epsilon}$ is the moving average of ϵ and $\tilde{\epsilon}$ is our approximation of the standard deviation.

3.1.2 Bandwidth allocation

Our system supports SLAs that specify, for each service group, the bandwidth guaranteed to be available whenever it is demanded. Given the SLAs in hand, we know by construction that the guaranteed level of service of all of the customers can be met simultaneously. Each service group that is demanding that much, or more, bandwidth will be initially allocated at least that much bandwidth. At the same time, those service groups forecasted to demand less bandwidth will receive a smaller initial allocation, covering just their forecasted demand. Thus, each service group will almost certainly at least get what it has been guaranteed, within the limits of accuracy of our forecasting.

The priorities specified in each SLA is then taken into account. We take the remaining, unallocated bandwidth, and give to each service group as much more than their initial allocation as they are forecasted to want, starting with the highest-priority group. Thus, service groups that have paid more for better best-effort service will get a better effort. If the SLA specifies a maximum allocation of bandwidth to a service group, that limit would be applied at this point.

There may then be some residue of unallocated bandwidth. This is split equally among all of the service groups, again within any limits on maximum bandwidth imposed by the SLAs. In recognition of the fallibility of the forecasting, an alternative that might offer better compliance with the guarantees would be to weight the distribution of this residue based on the sizes of the forecasted possible increases.

This is a somewhat rigid partitioning of all available bandwidth, at least for the duration of each observation interval. Similar decisionmaking could instead be made as each packet is processed, determining its fate on the basis of what has transpired up to the moment of its arrival. This would, unfortunately, greatly increase the cost of handling each packet in return for what we believe would be, at best, a marginal improvement in the fair use of available bandwidth.

3.1.3 Fairness

Among those groups of equal priority, the allocation of extra bandwidth over and above what had been guaranteed to those groups is all that has been forecasted that they might each desire, if there is sufficient bandwidth. When there isn't, the allocations are proportional to the amounts forecasted such that precisely all of the available bandwidth is consumed. This choice is actually fairly arbitrary, and max-min fairness could just as well have been used.

3.2 Admission Control

Once an aggregate bandwidth is assigned to a service group, the threshold on the number of admitted connections is decided. The main purpose of the admission control is to guarantee the minimum level of bandwidth for each flow. This minimum bandwidth may be specified in SLA. Otherwise, it is selected by a system administrator. Without such a minimum bandwidth for each connection, the system can nominally admit an infinite number of connections and still satisfy the aggregate bandwidth requirement. This may result in very slow connections opened for a long time, causing significant waste of resources in the system. The threshold can be computed from the assigned aggregate bandwidth and the chosen minimum bandwidth per connection.

In a more elaborate case, connections from different clients can be classified by looking at source IP addresses and assigned different minimum bandwidths.

3.3 Sharing within a Service Group: Micro-level Sharing

The objective of micro-level sharing is first to control the bandwidth of the service group to the bandwidth limit that was set at the macro level. At the same time, it should achieve fair sharing of the bandwidth among the different connections in the service group. Lastly, since micro-level sharing involves packet level control, the trade-off between the effectiveness of the control scheme and the system overhead should be considered.

At one extreme, we may simply focus on controlling the aggregate bandwidth without any separate treatment of each connection. This will result in a blind sharing of bandwidth. Although cheap, this simplistic method may result in unfair treatment of connections. At the other extreme, we may allocate bandwidth differently for each connection and realize a fair sharing of bandwidth. However, this will require separate control of each connection and induce an excessive system overhead.

One practical issue to be considered for a fair sharing in micro-level is the nature of the connections. When slow connections (for example, from clients using modems) and fast ones are mixed for a particular web site, equal allocation of bandwidth over the slow and fast connections are not meaningful. Also, when data rate excesses the allocated aggregate bandwidth, penalizing both slow and fast connections equally can be considered unfair.

The scheme we propose considers the nature of each connection as mentioned above. At the same time, it can be realized with relatively little overhead. We impose a delay that varies with the size of the packets. This is done to all packets for all connections in the service group whenever the currently measured bandwidth exceeds the assigned aggregate bandwidth for the service group. For this, we maintain the integral of the errors, *i.e.*, E = E + (S - B), where S is the current bandwidth consumption and B is the target rate of bandwidth consumption. Then a delay per byte of a(S-B) + b*E, with an appropriate selection of constants a and b, gives a type 1 feedback control system [23]. (The integration term (bE) is for the elimination of steady-state error.) This will result in reasonable treatment of connections, because slow connections will see sufficiently small additional delays that it will be in the noise. In contrast, fast connections will see the delays that are really keeping things under control.

This method does not require the maintaining of separate packet queues for individual connections, resulting in a significant reduction in system overhead. However, it still requires some computation for every packet. For a further reduction in the overhead, one might let through immediately those packets whose round-trip time, based on the time of the last acknowledgement packet, seems large enough relative to the delay being imposed.

3.4 Efficiency

Figure 3 shows the diurnal variation of workload at a commercial web site, which we will refer to as Customer 1. (The gap at 17 is due to a break in the trace, not to a break in the activity at the web site.) As seen in the

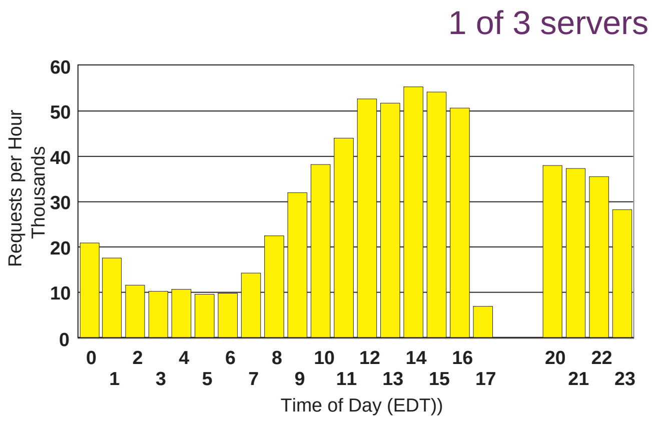

Figure 3: Customer 1's Diurnal Workload Variation

figure, the number of requests varies significantly over time, ranging from a few thousands per hour to more than 50 thousand.

To evaluate the efficiency of our bandwidth allocation method, we compare it to the worst and best possible alternatives. The worst method of allocating bandwidth is to statically assign as much bandwidth to each service group as that service group will ever want. As can be seen in figure 3, even during the course of a single day that will leave large amounts of network capacity idle most of the time. The best possible method is to employ an oracle that can predict the precise amounts of bandwidth that will be consumed and to assign bandwidth budgets that precisely match those predictions. Our method must fall somewhere in between, since we track the actual consumption, but must leave some cushion in our budgets to allow for errors in our predictions.

Figure 4 shows the efficiency of the bandwidth forecasting and allocation described above for the log data from Customer 1. In addition, it shows the results from another commercial web site, which we will refer to as Customer 2. The figure plots the amount of unused bandwidth bandwidth made available to other service groups. The results are scaled such that the best possible results, as provided by the oracle, have a value of 1, while the worst possible method, static assignment, never frees unused bandwidth and has an efficiency of 0. For example, when the polling interval is 5 seconds, about

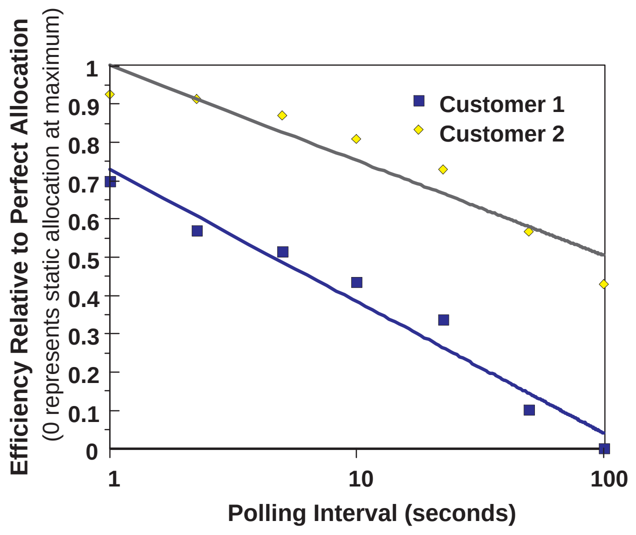

Figure 4: Efficiency of the Bandwidth Forcasting and Allocation

55% efficiency is achieved for Customer 1 and about 90% for Customer 2.

4 Implementation

The system has been prototyped on the AIX operating system for the IBM RS/6000 as a combination of a kernel extension based on the IBM Network Dispatcher and a user-level program. It is mainly composed of three modules: *Connection Monitor, Resource Controller, and Resource Manager.* Also two types of queues, *Connection Queue* and *Acknolwedgement Queue*, are maintained in the system. These queues are governed by the Resource Controller.

4.1 Major Data Structures

The following kernel data structures are maintained in the system.

Connection Record: Connection Records hold the dynamic information about each connection in the system. For each connection, it identifies the address of the back-end node to which the connection is dispatched. It also keeps track of the connection status, such as *Active*, *Finished*, *Pending*, etc and holds the list of Service Groups to which each connection belongs.

Connection Queue: For each service group, a Connection Queue is maintained for pending connection requests (SYN packets). A connection request can belong to multiple Service Groups and therefore to multiple Connection Queues.

Acknowledgement Queue: Acknowledgement Queue is used to pace ACK packets to enforce the allocated rate. The way in which an Acknowledgement Queue is maintained depends on the micro-level bandwidth sharing policy. When rate enforcement is imposed per flow, the Acknowledgement Queue is maintained for each flow, whereas if rate enforcement is done on a service group basis, a single Acknowledgement Queue is maintained for each service group.

4.2 Resource Controller

The Resource controller is the major component of the system and is realized as kernel extension modules. When a new SYN packet arrives, it looks into the destination IP address and the port number and identifies the service groups which the requested connection belongs to. If the corresponding Connection Queues are empty, it selects and dispatches the connection request to one of the back-end nodes. The selection is done in a weighted round-robin fashion [10], to balance the load over those nodes that are assigned to that service group. More requests are assigned to nodes with higher weights, which, by their static setting, accounts for nodes of differing speeds and, by their dynamic adjustment, accounts for load imbalances due to differences among the requests. Then, a Connection Record is constructed and used to forward any follow-up packets belonging to the same flow. The SYN packet is queued in the corresponding Connection Queues, if any of them are not empty.

When a received packet is an ACK packet, the controller identifies the corresponding Acknowledgement Queues of the appropriate Service Groups. It then either queues or fowards the packet to the back-end node depending on if the Acknowledgement Queues are empty.

4.3 Connection Monitor

The Connection Monitor is another component implemented in the kernel space. The main function of the connection monitor is to keep track of the status of each connection as well as that of resource utilization on each back-end node. It allocates a Connection Record whenever a new connection is opened between a client and a server, and deallocates it whenever the connection is terminated. It also maintains the statistics such as connection rate, number of open connections, etc, for each service group, and the bandwidth consumption for each open connection and service group. This is done by observing the acknowledgement numbers in the arriving packets. The bandwidth consumption rate of each service group is periodically reported to the Resource Manager.

Apart from the Connection Monitor, the system also gathers related information from the respective back-end nodes. The connection and resource information can be more accurately monitored in back-end nodes than in front-end nodes. In addition, the information gathered at the front-end can also be obtained by combining data from the respective back-end nodes. However, collecting part of it directly from the front-end is still meaningful. The report from a back-end node incurs significant cost to both back-end and front-end nodes due to the requirement of inter-node communication. Therefore, the reporting frequency is limited, possibly resulting in untimeliness of the information.

4.4 Resource Manager

The Resource Manager is the only user-level program in the sytem. The statistics being gathered by the Connection Monitor are read, periodically, by the Resource Manager. They are then used in the macro-level sharing

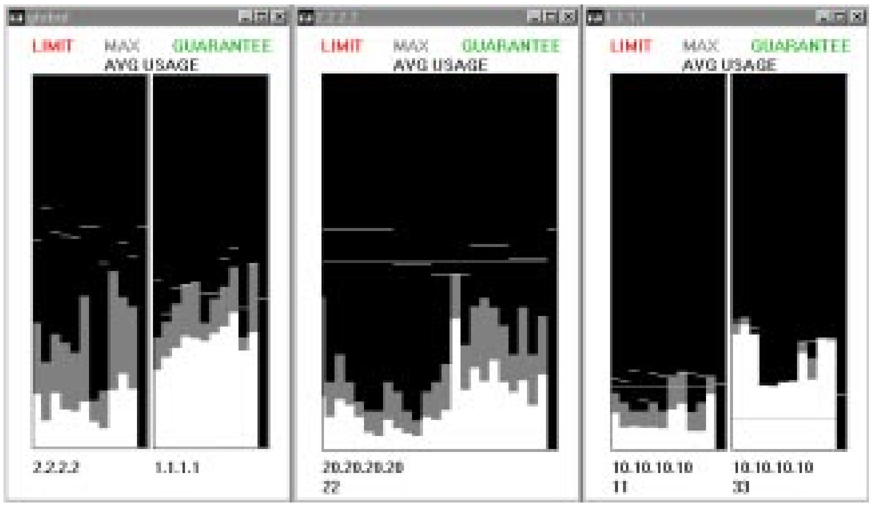

Figure 5: Resource Manager in Operation

calculations performed by the Resource Manager. The resulting bandwidth budgets and admission control settings are passed back down to the Resource Controller.

In a server farm that includes multiple front-end controllers, each has its own Resource Manager. They all communicate with a global Resource Manager, which is responsible for allocating bandwidth among all of the front-end controllers. It polls the local Resource Managers at each front-end controller periodically, computes the new, global budget, and passes back to each local manager its own budget. The polling by the global Resource Manager, using TCP/IP, is much more expensive than that by the local Resource Managers of their own Connection Monitors, using a system call exported by the kernel extension, and is therefore significantly less frequent.

This is illustrated in the screen shot in Figure 5. The window for the global manager, on the left, shows the average bandwidth consumption of each local manager (white), the peak consumption (gray), the bandwidth guaranteed to be available (green), and the bandwidth budget given to that local manager (red). Time progresses from left to right, each bar representing an observation and update cycle. The other two windows show the operation of the local managers, showing the activity and limits on each service group. In this example, 10.10.10.10:11 has the highest priority but the lowest consumption; 20.20.20.20:22 has the next lower priority; and 10.10.10.10:33 has the lowest priority. Because the demands of 10.10.10.10:11 are so low, the excess, available bandwidth is given to 20.20.20.20:22, whose demands are sufficiently large and erratic that it receives a generous cushion of excess bandwidht, leaving little for 10.10.10.10:33, which often only gets exactly what it was guaranteed and no more.

As indicated in section 3.4, the efficiency of the bandwidth sharing mechanism increases as the observation interval is shortened. This must be traded off against the overhead of running the resource manager more often. The performance of the resource manager was therefore measured on a relatively slow machine (160MHz). These measurements indicate that the overhead for the resource manager grows to about 1observation interval is about 2 milliseconds. This is significantly shorter than the time necessary for the estimation of bandwidth consumption, and so is of no further concern.

To see if admission control alone was effective, the system was run with an artificial workload based on httperf[21]. One change was made to httperf, to limit the bandwidth consumption of each client. In this test, it was set to 5KB per second per client, of which there were 10, while the resource manager was given a bandwidth budget of only 20KB per second. As such, there was certainly always sufficient demand to consume the entire budget. Therefore, underutilization is as much of an error as overutilization, which makes the square root of the mean-squared-error an interesting measure of success. Use of admission control alone led to an MSE error of about 5KB per second. This is about as good as we had hoped, given the poor granularity of the individual clients relative to the budget. It does, however, point up the value of adding ACK pacing. This allows us to admit enough more requests to ensure that a drop in demand per client will not leave us underutilized (we add 50 percent to the connection limit) while also keeping increases in demand per client from blowing consumption way past the budgeted amount.

5 Conclusions

The use of a front-end control node in a Web server farm is effective in the management of shared resources like bandwidth. Admission control at the front-end node limits consumption of resources at the server load while allowing a sufficient number of jobs into the system to keep overall utilization high. Providing the front-end node with TCP rate control allows the actual bandwidth consumption among the admitted connections to be fine-tuned to the prescribed levels. Adding some simple forecasting allows occasional, periodic adjustments to the control settings to be made with reasonable accuracy and fairness without undue cost.

As it stands, our front-end control node operates solely on the basis of information directly available to it. In the future, we intend to integrate it with other sources of information, which may influence the sharing of resources in a variety of ways. We are also considering how this system will interact with other system management tools, to which reports must be made and whose control directives must be followed.

References

[1] N. Bhatti, A. Bouch, A. Kuchinsky. "Integrating User-Perceived Quality into Web Server Design," *Proceedings of the WWW9 Conference*, May 2000.

[2] CacheFlow Inc., *CacheFlow,* http://www.cacheflow.com/Default.html.

[3] Cisco Systems Inc., *Local Director,* http://www.cisco.com/warp/public/751/lodir/index.shtml.

[4] J. Crowcroft and P. Oechslin, "Differentiated End-to-End Internet Services Using a Weighted Proportional Fair Sharing TCP," *Computer Communications Review*, pp 53-69, July 1998.

[5] D. Dias, W. Kish, R. Mukherjee, and R. Tewari. "A Scalable and Highly Available Web Server," *Proceedings of the 1996 IEEE Computer Conference (COMPCON)*, February 1996.

[6] W. Feng, D. Kandlur, D. Saha, and K.G. Kang, “Understanding TCP Dynamics in an Integrated Services Internet,” *Proceedings of NOSSDAV '97*, May 1997.

[7] S. Floyd and V. Jacobson, “Random Early Detection Gateways for Congestion Avoidance,” *IEEE/ACM Trans. on Networking*, August 1993.

[8] A. Fox, S. D. Gribble, Y. Chawatbe, E. A. Brewer, and P. Gautbier, “Cluster-Based Scalable Network Services,” *Proc. 1997 Symposium on Operating Systems Principles*, October 1997.

[9] M. Grosslauser and D. N. C. Tse, “A Time-Scale Decomposition Approach to Measurement-Based Admission Control,” *Proc. IEEE INFOCOM '99*, March 1999.

[10] G. Hunt, G. Goldszmidt, R. King, and R. Mukherjee, “Network Dispatcher: A Connection Router for Scalable Internet Services,” *Proceedings of the 7th International World Wide Web Conference*, 1998.

[11] IBM, *Web Traffic Express*, http://www.software.ibm.com/webservers/wte.

[12] Inktomi Corp., *Traffic Server*, http://www.inktomi.com/products/traffic/technology.html.

[13] A. Iyengar, M.S. Squillante, L. Zhang, “Analysis and Characterization of Large-Scale Web Server Access Patterns and Performance,” *World Wide Web*, vol. 2, pp 85-100, Baltzer, 1999.

[14] S. Karandikar, S. Kalyanaraman, and P. Bagal, “TCP Rate Control,” *ACM Computer Communication Review*, January 2000.

[15] R. Lee, *A Quick Guide to Web Server Acceleration*, http://www.novell.com/bordermanager/accel.html.

[16] E. Levy, A. Iyengar, J. Song and D. Dias, “Design and Performance of a Web Server Accelerator,” *Proc. IEEE INFOCOM'99*, March 1999.

[17] K. Li, S. Jamin, “A Measurement-Based Admission-Controlled Web Server,” *Proc. IEEE INFOCOM 2000*, March 2000.

[18] A. Luotonen, *Web Proxy Servers*, Prentice Hall, 1997.

[19] A. Mehra, R. Tewari, and D. Kandlur, "Design Considerations for Rate Control of Aggregated TCP Connections," *Proc. NOSSDAV '99*, June 1999.

[20] Mindcraft, *Mindcraft - WebStone Benchmark Information*, http://www.mindcraft.com/webstone/.

[21] D. Mosberger and T. Jin, "httperf–A Tool for Measuring Web Server Performance," Hewlett Packard, Palo Alto, CA, http://www.hpl.hp.com/personal/David_Mosberger/httperf.html

[22] Network Appliance, *Netcache: Highly Scalable Network and Web Caching Solutions.*, http://www.netapp.com/products/internet.html.

[23] K.Ogata *Modren Control Engineering, third edition*, Prentice Hall, 1997.

[24] V. Pai, M. Aron, G. Banga, M. Svendsen, P. Druschel, W. Zwaenepoel, and E. Nahum, "Locality-Aware Request Distribution in Cluster-based Network Servers," *Proceedings of ASPLOS-VIII*, 1998.

[25] R. Satyavolu, K. Duvedi, and S. Kalyanaraman, "Explicit Rate Control of TCP Applications," *ATM-Forum/98-0152R1*, February 1998.

[26] W. Richard Stevens, *TCP/IP Illustrated,* Vol 1, Addison-Wesley, 1997.

[27] J. Song, E. Levy, A. Iyengar, and D. Dias, "Design Alternatives for Scalable Web Server Accelerator," *Proc. ISPASS'2000*, April 2000.