SHIV SHAKTI

International Journal of in Multidisciplinary and Academic Research (SSIJMAR)

Vol. 4, No. 3, June 2015 (ISSN 2278 – 5973)

Dynamic Load Balancing In Web Server Systems

Ms. Rashmi
M.Tech. Scholar
Deptt. Of CSE
S (PG) I T M, Rewari
sharma.rashmi416@gmail.com

Impact Factor = 3.133 (Scientific Journal Impact Factor Value for 2012 by Inno Space Scientific Journal Impact Factor)

Global Impact Factor (2013)= 0.326 (By GIF)

Indexing:

Abstract

In this paper, I have tried to make a study to solve the problem of handling the load. Then, on the basis of the studies made we try to find out the best solution. An algorithm is then designed which covers the theoretical aspects explaining the details of how the load will be handled in our approach.

The implementation of the prescribed approach needs a lot of research and time as many aspects regarding the implementation, need to be carefully handled. But due to time constraints and limited knowledge -- in the implementation part, we just aim to propose a prototype scalable web server consisting of a load-balanced cluster of hosts that collectively accept and service TCP connections.

Lastly, I have evaluated the load testing concepts and analyze the performance of the web servers. Various parameters effecting the response time are summarized. Graphs and result generated later helped in verifying the results obtained in the theoretical studies.

Introduction

A scalable Web-server system needs to appear as a single host to the outside world, so that users need not be concerned about the names or locations of the replicated servers and they can interact with the Web system. This architecture provides scalability and transparency but requires some internal mechanism that assigns client requests to the node which in that moment can provide, possibly, the minimum response time.

DNS-dispatcher based systems can easily scale from locally to geographically distributed Web-server systems and are also used in other related distributed architectures, such as Content Delivery Networks. However, dispatching requests through the DNS has three problems that prevent load balancing among the Web nodes: routing decisions are content-blind, the different amount of load coming from various Internet regions may easily overload some Web nodes, and the address caching mechanism in the DNS causes the large majority of client requests to skip the A-DNS for address resolution, to the extent that the DNS dispatcher of a popular Web site controls only a small fraction of the requests reaching the Web site. These issues have been addressed through sophisticated DNS dispatching policies working on the TTL, or multiple tiers of proprietary name servers combined with very low values (few seconds) for the TTL.

We also examine an alternative solution that integrates the first-level request dispatching carried out by the A-DNS through a simple stateless algorithm (e.g., round-robin) with some distributed redirection mechanism managed directly by the Web nodes. We also need to recognize that load balancing goals are of key importance for the system administrator, but are worth for the user only if they contribute to reduce the response time.

Several circumstances call for load testing. Suppose, for instance, that you anticipate a significant traffic increase to your site following a marketing campaign. In place of what is now a peak of 3,000 session starts per hour, you're expecting twice that. Currently, your dial-up customers experience an average 6.5-second response time on search requests, the most critical e-business function. What will be the response time when the site's load increases to 6,000 sessions per hour? As another example, suppose that you're adding new functionality to the site or redesigning Web pages. You must know how this will affect response time before your customers find out; doing so lets you detect potential performance problems and fix them before they occur. Another good time to perform load testing is when you plan to implement IT infrastructure changes.

Methodlogy

How Load Testing Tool Works:

The load generator mimics browser behaviour: It continuously submits requests to the Web site, waits for a period of time after the site sends a reply to the request (the *think time*), and then submits a new request. The load generator can emulate thousands of concurrent users to test Web site scalability. Each emulated browser is called a *virtual user*, which is a key load-testing concept. A load test is valid only if virtual users' behaviour has characteristics similar to those of actual users. You must therefore ensure that your virtual users:-

1. Follow patterns similar to real users

2. Use realistic think times,

3. React like frustrated users, abandoning a Web session if response time is excessive.

Failure to mimic real user behavior can generate totally inconsistent results, because customers who abandon a session use fewer site resources than those who complete it.

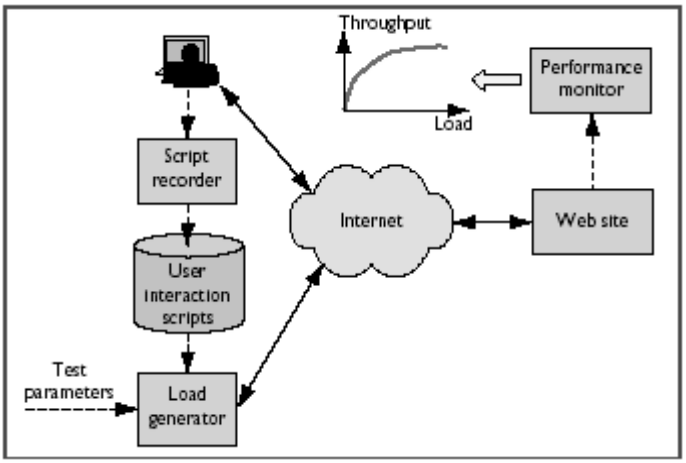


Figure 4.1.The load-testing process.

Testing Parameters and Results

Several circumstances call for load testing. Suppose that you're adding new functionality to the site or redesigning Web pages. You must know how this will affect response time before your customers find out; doing so lets you detect potential performance problems and fix them before they occur. Another good time to perform load testing is when you plan to implement IT infrastructure changes. You can use load testing to predict your Web site's performance at any load level by simply increasing the number of virtual users until you achieve the desired load. However, running load tests for numerous values with numerous virtual users can be time consuming and expensive. Consider a scenario in which several virtual users submit requests to a Web site, and let N_{VU} = number of virtual users.

N_C = number of concurrent requests a Web site is processing.

Z = average think time, in seconds.

R = average response time for a request, in seconds.

X_0 = average throughput, in requests per second.

Using the Response Time law, we get the following relationship:

$$R = \frac{N_{VU}}{X_0} - Z. \qquad (1)$$

A Web site's throughput is a function of the load level — the number *NC* of concurrently executing requests - and the service demands these requests make on individual site resources (processors, storage devices, and networks, for example). We define a request's service demand — *Di* — at resource *i* as the average total time the request spends receiving service from the resource. This time does not include queuing time, and is therefore independent of the load level. Given this, we can write that

$$X_0(N_C) = f(D_1 \ldots D_K, N_C) \qquad (2)$$

to indicate that throughput is a function of load level and the service demands on a Web site's *K* resources. Because the same is true for response time, we can write that:

$$R(N_C) = g(D_1 \ldots D_K, N_C). \qquad (3)$$

So, combining equations 1-3, we get

N_{VU} = [R (N_C) + □ □Z]*□X_0 (N_C). (4)

We can now use either an analytic or simulation model to predict response time and throughput for different values of the *NC* load level, and use equation 4 to estimate the number of virtual users we need to generate a given value of *NC*.

Load-testing tools are quite useful here. They can, for example, generate scripts for a few virtual users to measure service demands, which are load-independent. You can then use the service demands as input parameters to performance models.

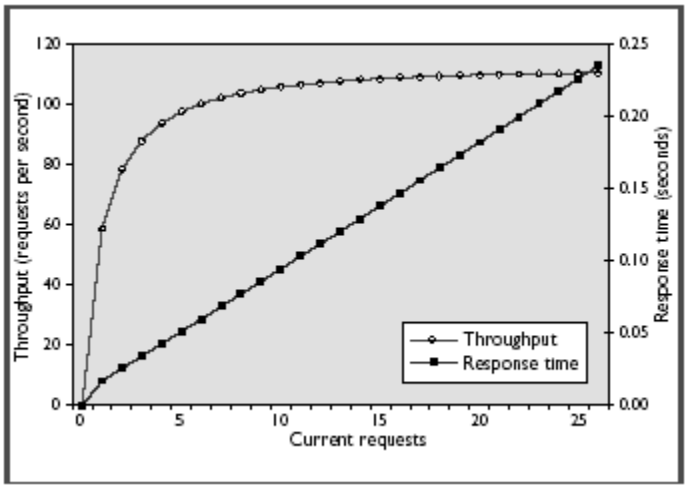


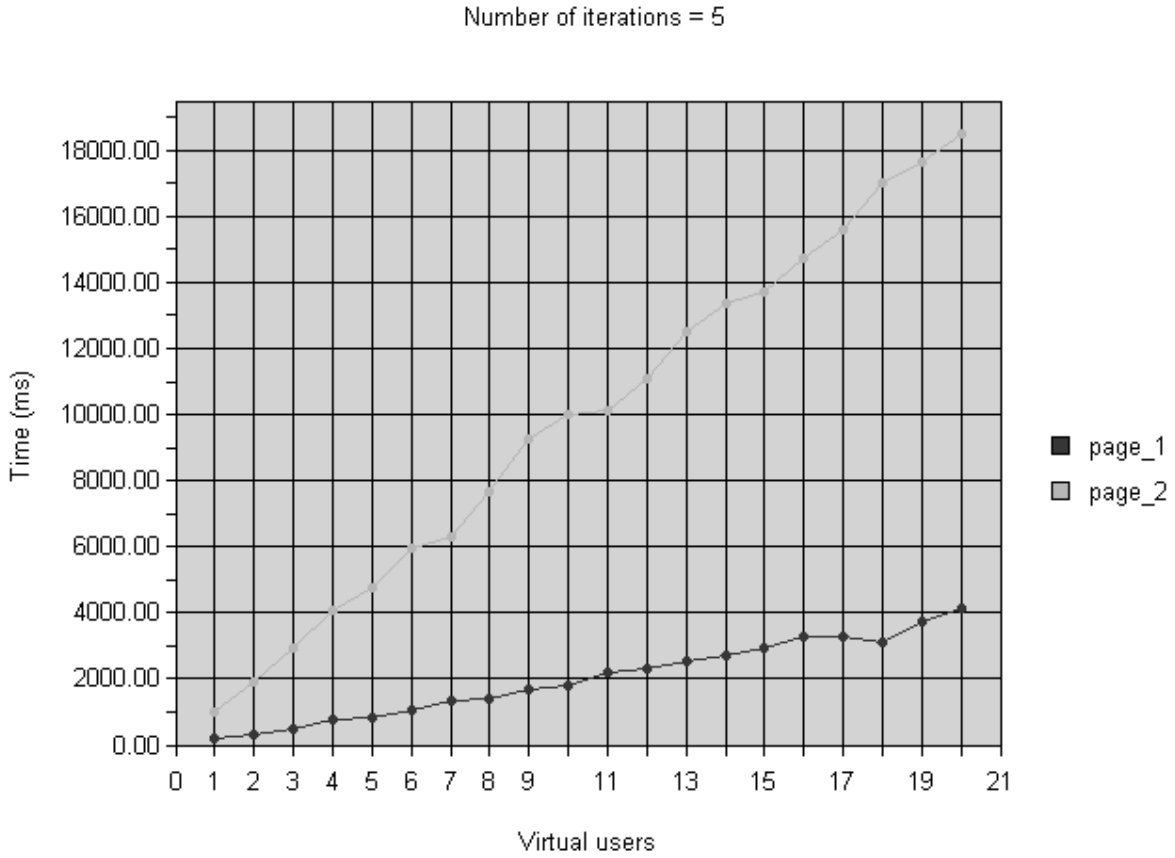
Figure 4.2: Graph of Throughput vs. No. of Request

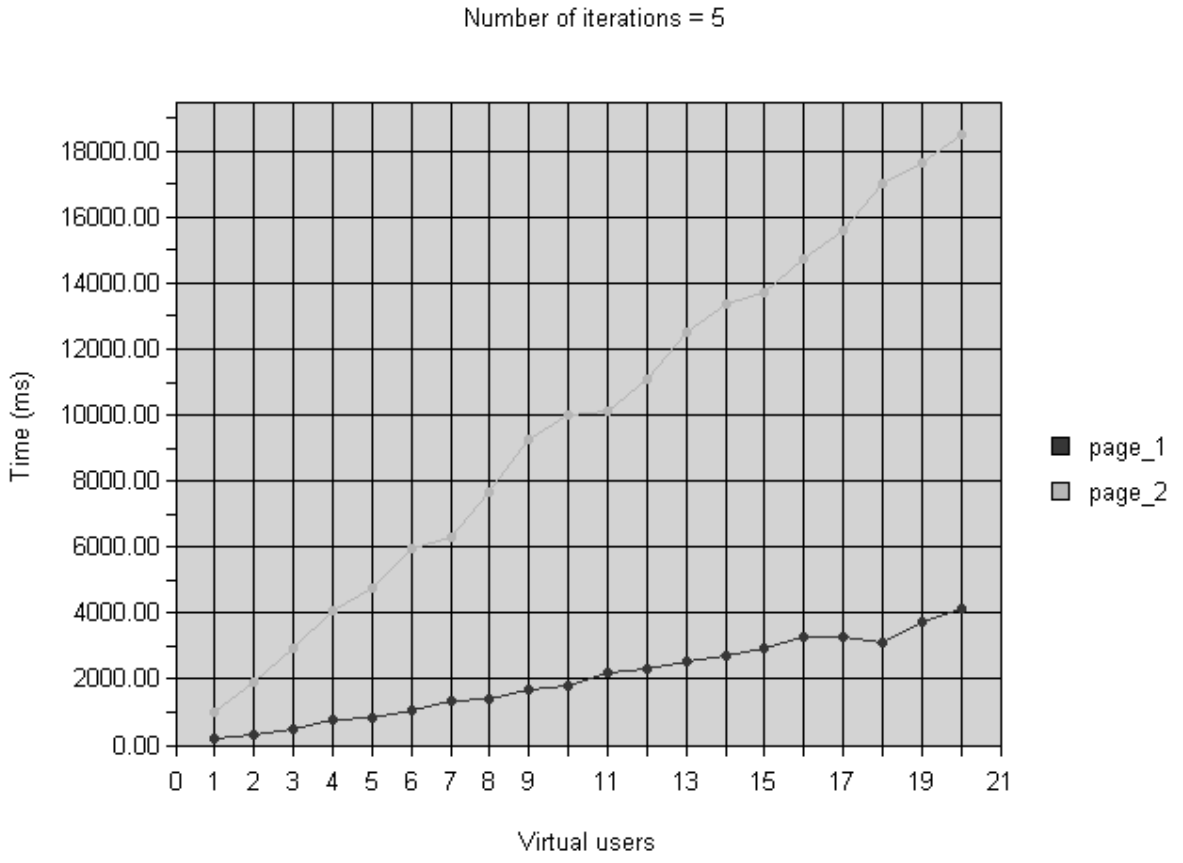
Workload intensity typically measured in session starts per hour.

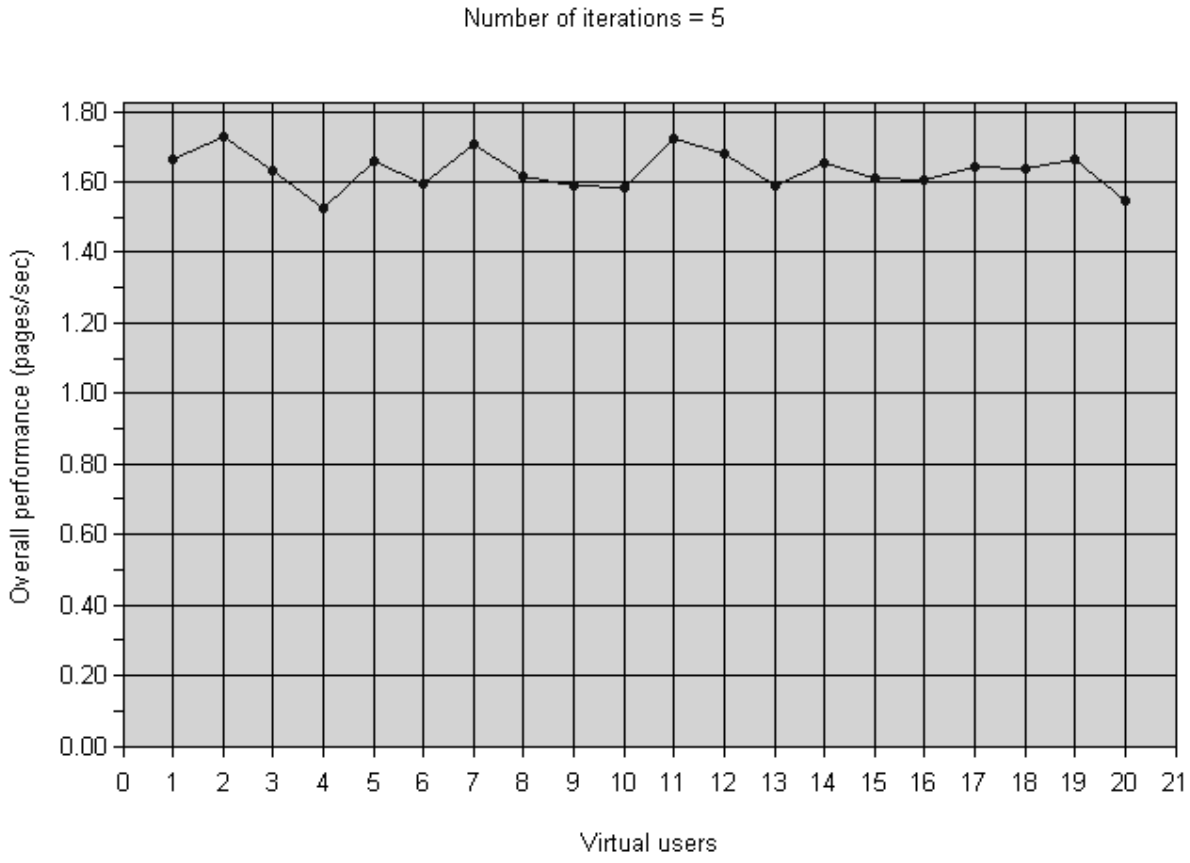
- **Workload mix**, described by the scripts, which define typical sessions and what customers do in each session type.
- **Customer behavior parameters**, including abandonment threshold and think time.

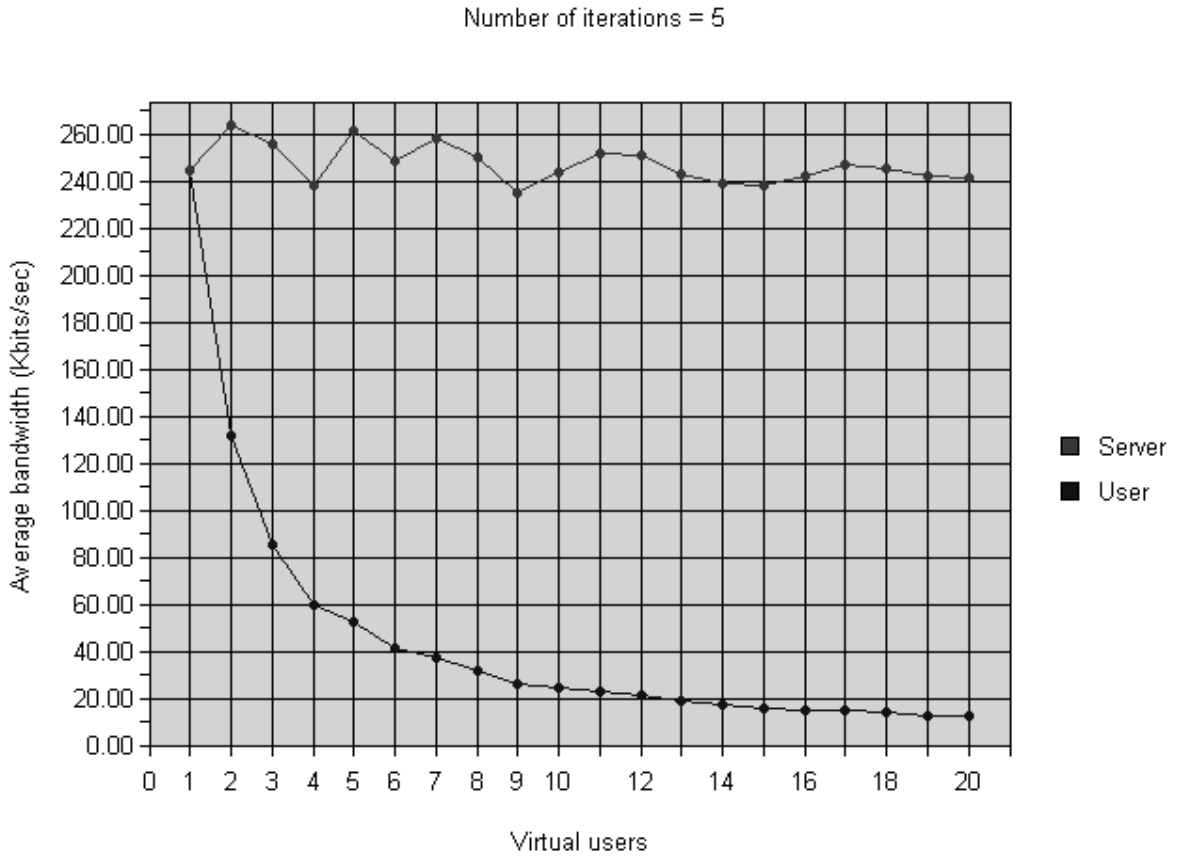
Typical load test results include:

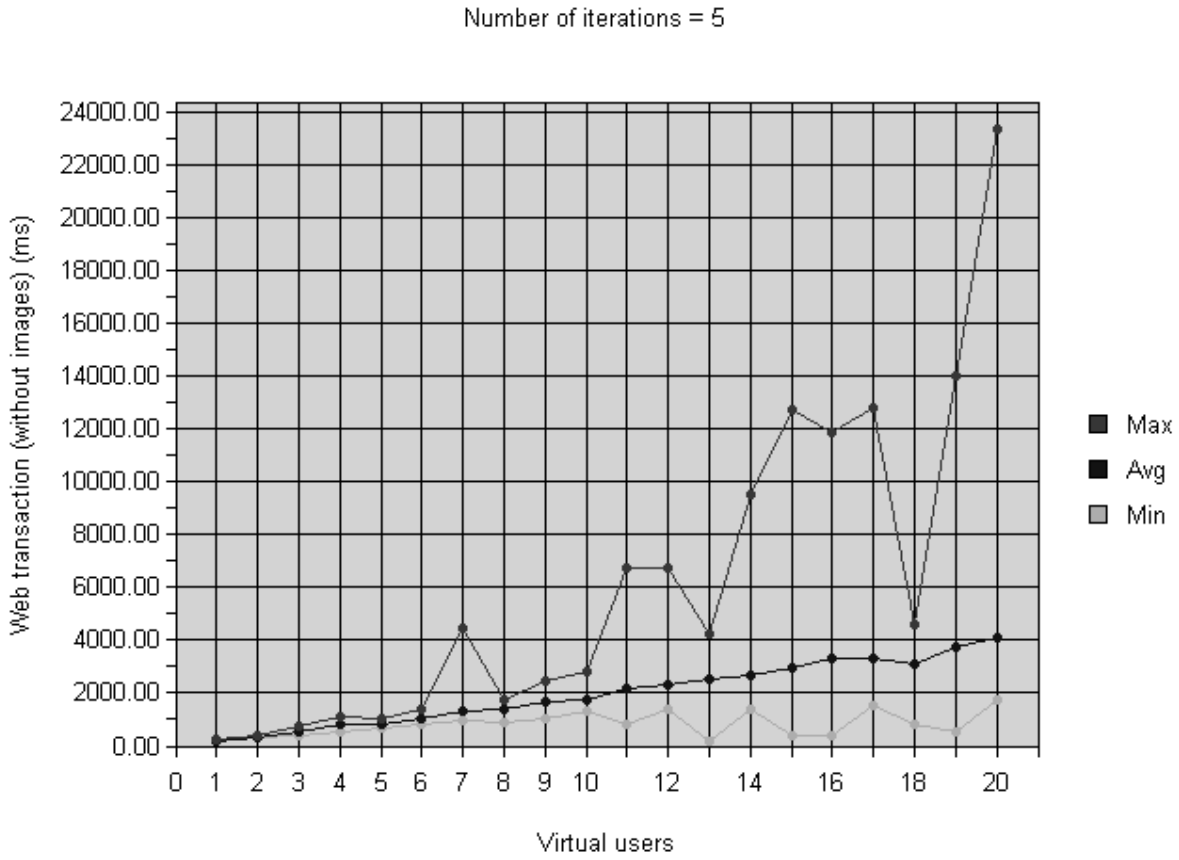
- Number of completed and abandoned sessions per hour, as a function of the number of started sessions per hour.
- Revenue and potential lost revenue throughput, as a function of the number of sessions started per hour.
- Individual page download times and transaction completion times versus the number of sessions started per hour.











Conclusion

We provide taxonomy of the redirection schemes in distributed Web systems including centralized vs. distributed control algorithms for the activation of the mechanism, for the localization of the destination nodes, and for the selection of the requests to be redirected. These control policies can be based on local, partial or global state information. A thorough investigation has led to the proposal of centralized and distributed schemes that not only achieve good results, but also guarantee stable performance. The stability of performance under different system scenarios is crucial when considering real systems that operate in the extremely variable Web environment. Even the DNS round-robin dispatching that performs very poorly under skewed request load, when combined with a distributed redirection mechanism carried out by the Web nodes, can achieve good and stable performance with percentages of redirections lower than 15%.

By comparing fully centralized, fully distributed and hybrid control schemes, we have found that in some cases, a distributed redirection algorithm may achieve slightly worse performance than the best centralized alternatives, but it has three major advantages that make its use preferable: **it is easier to implement, imposes much lower computational and communication overheads**, and allows the use of content-aware redirection policies that are gaining so much importance in the Web.

At last, we hope that our inferences may prove to be successful on practical implementation and may help in solving the problem of load balancing.

Advantages of Load Sharing on Web Servers

Pros and Cons of different types of Load Balancing:

Server load balancing is only one of many ways to accomplish the goal of multiple servers responding as one. Let's look at some different possibilities, and the pros and cons of each:

DNS load balancing
As the name implies, balancing is done with DNS. A single name resolves to multiple other names or IP addresses. These real names or IP addresses are then hit in a round-robin manner.

Pro:

- Very simple to configure and understand.

Cons:

- No intelligence other than round-robin.
- No way to guarantee connection to the same server twice if needed (sticky connections).
- DNS cannot tell if a server has become unavailable.
- Load may not be evenly distributed, as DNS cannot tell how much load is present on the servers.
- Each server requires a public IP address, in the case of publicly available web servers.

Bridged load balancing
Load balancing at layer two, or bridged load balancing, is a very simple model. A virtual IP address is created in the same IP network as the real servers. Packets destined for the virtual IP address are forwarded to the real servers.

Pros:
•Can be inserted into an existing network, with no additional IP networks required.
•Possibly easier to understand for simple networks.
•Usually less expensive than a routed model.

Cons:
•Layer-2 issues including loops and spanning-tree problems can arise if the solution is not designed carefully.
•Can be harder to understand for people used to layer-3 environments.
•Usually limited to a single local network.

Routed load balancing

Load balancing at layer three, or routed load balancing, is slightly more complex than bridged load balancing. In this model, the virtual IP address exists on one network, while the real servers exist on one or more others.

Pros:

- Expandability. Routed models allow for the real servers to be geographically diverse. The possibilities here are almost limitless.
- Easier to understand for people used to layer-3 environments.
- No spanning-tree issues.

Cons:

- Layer-3 load balancing can be costly. The CSM-S module for the 6500

switch is one of the most expensive modules Cisco produces.

- Requires network design and additional IP address space to implement. Because the real servers must be on a different broadcast domain from the virtual server, a routed load balancer cannot be dropped into an existing flat network without redesigning the network.

References

1. V. Cardellinie, M. Colajannie, and P.S. Yu, "Redirection Algorithm for Load Sharing in Distributed Web-Server Systems," Proc. 19th IEEE Int'l Conf. Distributed Computing,

2. D. Mosedale, W. Foss, and R. McCool, "Lessons Learned Administering Netscape's Internet Site," IEEE Internet Computing, Vol. 1, No. 2, March-April 1997, pp. 28-35.

3. M. Baentsch, L. Baum, and G. Molter, "Enhancing the Web's Infrastructure: From Caching to Replication," IEEE Internet Computing, Vol. 1, No. 2, Mar.-Apr. 1997, pp. 18–27.

4. T.T. Kwan, R.E. McGrath, and D.A. Reed, "NCSA's World Wide Web server: Design and Performance," Computer, Vol. 28, No. 11, Nov. 1995, pp. 68–74.

5. M. Colajanni, P.S. Yu, and D.M. Dias, "Analysis of Task Assignment Policies in Scalable Distributed Web-Server Systems," IEEE Trans. Parallel and Distributed Systems, Vol. 9, No. 6, June 1998, pp. 585–600.

6. D.M. Dias et al., "A Scalable and Highly Available Web-Server," Proc. 41st IEEE Computer Soc. Int'l Conf., IEEE Computer Soc. Press, Los Alamitos, Calif., Feb. 1996, pp. 85–92.

7. R.J. Schemers, "lbmnamed: A Load Balancing Name Server in Perl," Proc. 9th Systems Administration Conf., Usenix Assoc., Berkeley, Calif., Sept. 1995.

8. M. Beck and T. Moore, "The Internet2 Distributed Storage Infrastructure Project: An Architecture for Internet Content Channels," Proc. 3rd Workshop WWW Caching,

9. V. Cardellini, M. Colajanni, and P.S. Yu, "DNS Dispatching Algorithms with State Estimators for Scalable Web-Server Clusters," World Wide Web J., Baltzer Science, Bussum, Netherlands, Vol. 2, No. 2, July 1999.

10. G.D.H. Hunt et al., "Network Dispatcher: A Connection Router for Scalable Internet Services," J. Computer Networks and ISDN Systems, Vol. 30, Elsevier Science, Amsterdam, Netherlands, 1998.

11. O.P. Damani et al., "ONE-IP: Techniques for Hosting a Service on a Cluster of Machines," J. Computer Networks and ISDN Systems, Vol. 29, Elsevier Science, Amsterdam, Netherlands, Sept. 1997, pp. 1,019–1,027.

12. M. Garland et al., "Implementing Distributed Server Groups for the World Wide Web," Tech. Report CMUCS-95-114, School of Computer Science, Carnegie Mellon Univ., Pittsburgh, Pa., Jan. 1995.

13. A. Bestavros et al., "Distributed Packet Rewriting and its Application to Scalable Web Server Architectures," Proc. 6th IEEE Int'l Conf. Network Protocols, IEEE Computer Soc. Press, Los Alamitos, Calif., 1998.

14. DNS-based approaches M. Colajanni, P.S. Yu, V. Cardellini, "Dynamic load balancing in geographically distributed heterogeneous Web servers," Proc. 18th

Intl. Conf. Distributed Computing Systems, 1998, pp. 295-302

15. TCP splicing Chu-Sing Yang, Mon-Yen Luo, "An effective mechanism for supporting content-based routing in scalable Web server clusters," Proc. Intl.

16. TCP handoff Wenting Tang, L. Cherkasova, L. Russell, M.W. Mutka, "Customized

library of modules for STREAMS-based TCP/IP implementation to support content-aware request processing for web applications," 3rd Intl. Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems, 2001, pp. 202-211.