Geographic Load Balancing for Scalable Distributed Web Systems *

Valeria Cardellini
University of Roma Tor Vergata
Roma, Italy 00133
cardellini@ing.uniroma2.it

Michele Colajanni
University of Modena
Modena, Italy 41100
colajanni@unimo.it

Philip S. Yu
IBM T.J. Watson Research Center
Yorktown Heights, NY 10598
psyu@us.ibm.com

Abstract

Users of highly popular Web sites may experience long delays when accessing information. Upgrading content site infrastructure from a single node to a locally distributed Web cluster composed by multiple server nodes provides a limited relief, because the cluster wide-area connectivity may become the bottleneck. A better solution is to distribute Web clusters over the Internet by placing content nodes in strategic locations. A geographically distributed architecture where the Domain Name System (DNS) servers evaluate network proximity and users are served from the closest cluster reduces network impact on response time. On the other hand, serving closest requests only may cause unbalanced servers and may increase system impact on response time. To achieve a scalable Web system, we propose to integrate DNS proximity scheduling with an HTTP request redirection mechanism that any Web server can activate. We demonstrate through simulation experiments that this further dispatching mechanism augments the percentage of requests with guaranteed response time, thereby enhancing the Quality of Service of geographically distributed Web sites. However, HTTP request redirection should be used selectively because the additional round-trip increases network impact on latency time experienced by users. As a further contribution, this paper proposes and compares various mechanisms to limit reassignments with no negative consequences on load balancing.

1. Introduction

The phenomenal growth of the Web is causing enormous strain on users, network service providers and content providers. Geographically distributed Web systems are the most scalable architectures to handle millions of accesses per day. In this paper, we consider a *Web site* that uses a single URL address to make the distributed nature of the service transparent to the users. The system architecture consists of various *Web clusters* placed in strategic Internet regions. Each Web cluster consists of one or more Domain Name System (DNS) servers and replicated back-end *Web server* machines[1] that are housed together in a location of an Internet region. Figure 1 shows an example of distributed Web site consisting of four Web clusters, each of them with multiple Web server nodes (WS) that are connected via a fast local network. Web clusters are typically interconnected via a high speed backbone to facilitate cooperation and information exchanges among the centers. To control the totality of the requests reaching the cluster and to mask the service distribution among multiple servers, each Web cluster provides a single virtual IP address that corresponds to the address of a front-end server. Independently of the mechanism that a Web cluster uses to route the internal load (see [12] for an overview), we refer to this front-end node as the *Dispatcher*. The Dispatcher acts as a centralized scheduler that receives the totality of requests reaching the Web cluster and routes them among the back-end servers in a client transparent manner. In a geographically distributed Web system, the decision on client request assignment can be taken at various network levels. A survey on distributed Web architectures can be found in [3].

The DNS servers of the Web site execute the *first-level assignment*. It acts on the address lookup phase of a client request that looks for an IP address corresponding to the URL hostname field. We assume that in a geographical context, it is appropriate to use an enhanced DNS server that implement some *proximity algorithm* to reply to the name resolution request. Through this mechanism, the DNS will reply with high probability with the IP address of the Web cluster Dispatcher closest to the client. DNS address caching mechanisms augment this probability because intermediate name servers of each Internet region will tend to get the resolution of the closest cluster. The concept of Internet proximity is still an open issue that will not be addressed in this paper, however many proposals exist for es-

[1] We consider systems with homogeneous Web clusters, where any server node owns or can access a replicated copy of the site content.

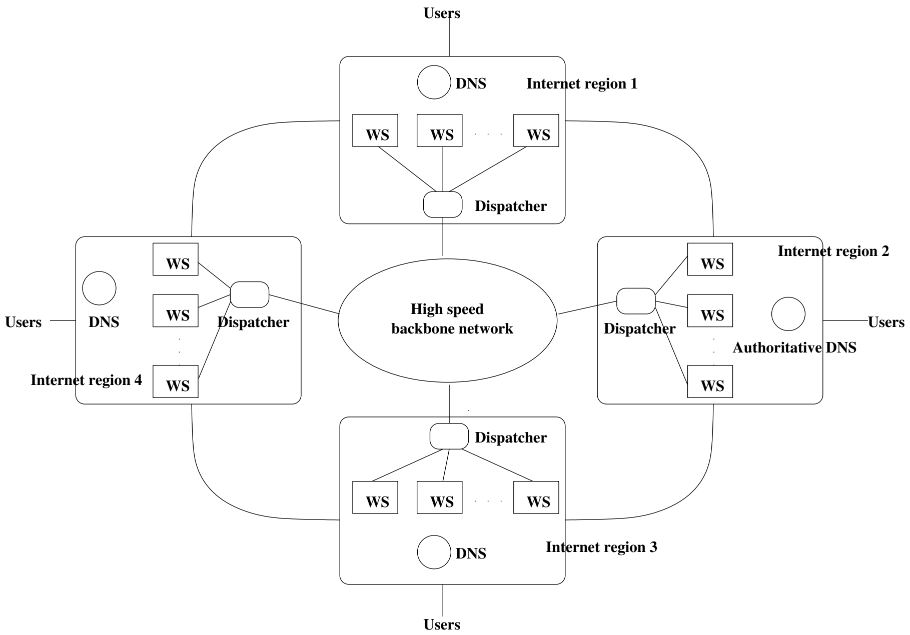

Figure 1. Architecture of the geographically distributed Web site.

timating static (e.g., network hops) or dynamic distances (e.g., network traffic) [5].

After the lookup phase, the page request arrives at the Dispatcher of a Web cluster that executes the *second-level assignment* among the server nodes. Dispatcher algorithms are out of the scope of this paper because we can assume that the Dispatcher is able to keep the load balanced among the servers of a Web cluster [8, 12].

The DNS assignment is a very coarse grain distribution of the load among the Web clusters, because proximity does not take into account heavy load fluctuations of Web workload that are amplified by the geographical context. Requests arrive in bursts, clients connected to the Web site are not uniformly distributed among the Internet domains, world time zones are another cause of heterogeneous source arrivals. To address these issues, we propose a distributed architecture that integrates DNS dispatching among the Web clusters with a *third-level (re)assignment* that can be activated by any Web server that is in critical load condition. The redirection mechanism is done through the HTTP protocol that allows a Web server to redirect a request by specifying the appropriate status code in the response header and indicating the alternative IP address from which the client can obtain the requested document [1, 4].

Through the redirection mechanism, an over-utilized server can get immediately rid of a fraction of the requests previously assigned by the DNS and the Dispatcher. Since this dispatching acts on individual requests of Web pages, it can achieve a fine grain control level. However, HTTP server redirection should be used selectively because it adds a round-trip latency to every re-assigned page request. The users may perceive or not an increase in the response time, depending on the load of the first contacted server. Hence, the second objective of this paper is to investigate whether it is possible to limit request redirection without affecting system load balancing. To this purpose, we propose and compare various mechanisms to limit request redirection. The goal is to guarantee that network overheads due to redirection have an impact on user latency time inferior to the system overhead due to an overloaded Web server. We show that strategies that limit redirection to heaviest requests can reduce substantially the amount of redirections and do not degrade Web cluster load balancing. The study is carried out through a simulation model of the Web system and network infrastructures. The system model details all characteristics of Web client/server interactions, while the network model is an approximate Internet vision that should provide a fair testbed to compare performance of different algorithms for geographic load balancing.

The paper is organized as follows. Section 2 analyzes related work. Section 3 presents various policies for request redirection by the Web servers and for limiting the percentage of redirected requests. Section 4 and 5 describe the system and network model, respectively. Section 6 discusses experimental results. Section 7 concludes the paper.

2. Related work

A considerable number of academic and commercial proposals regarding Web architectures with multiple nodes have focused on how to share the load evenly, especially in locally distributed Web systems [3, 12].

Two-level dispatching schemes, where client requests are initially assigned by the DNS, and each Web server may redirect a request to any other server of the system through the HTTP redirection mechanism, have been proposed for locally distributed Web systems in [1, 4]. Other request redirection strategies that use the built-in HTTP mechanism have been proposed for geographically distributed Web systems. For example, Cisco's DistributedDirector [6] uses a centralized single-level dispatching scheme: each request reaches a dispatcher that directs it to the Web server that is closest to the client. Most of the proposed geographically distributed Web systems place one or more Web clusters in different Internet regions, for example F5 Networks and Resonate products. The first-level assignment among the Web clusters is typically carried out by the Web site DNS that implements some proximity based dispatching strategy.

The originality of this paper is twofold. We consider not only sharing the load, but also minimizing the impact of WAN network delays on response times perceived by the users. Our proposal to augment the Quality of Service (QoS) of geographically distributed Web systems is to use a third-level dispatching through which over-utilized servers can immediately activate a redirection mechanism. Since request redirection may increase latency time experienced by the users, our second goal is to propose strategies that select the most suitable subset of requests to be redirected.

A quite different solution to geographical distribution of Web content is provided by companies that offer global delivery services, such as Akamai and Mirror Image. When using this service, Web site administrators delegate the responsibility for content distribution to the service company that owns a set of geographically dispersed servers. They constitute the so-called *content distribution network* containing copies of the objects. The system is integrated with a mapping service where the Web site DNS or the Web site servers work in cooperation with the company servers. This mapping service aims to redirect the page requests to a nearby company server with low-medium utilization.

3. Server redirection strategies

We consider a Web architecture distributed over a wide area that does not make convenient to use centralized load redirection policies [4]. Hence, we focus on totally distributed mechanisms where redirection is activated by an alarm mechanism that checks CPU-disk utilization of each server node. Any overloaded server starts to redirect requests when its load exceeds a *threshold load* and ends when the load returns below the threshold. As a load metric, we use the server utilization evaluated during the last observation period, referred to as the *check-server-load interval*.

	Name	*Information*
Selection policies	R-all	none
	R-size	page size
	R-num	page hit number
Location policies	RR	none
	Load	Web cluster load
	Prox	network proximity

Table 1. Server redirection policies.

Once the server has decided to activate the redirection process, the *selection policy* determines which requests have to be redirected. We assume that only new requests for entire pages are eligible for redirection. The straightforward solution is to redirect every request reaching an overloaded server (i.e., redirect-all policy or **R-all**). We also investigate how it is possible to limit request redirection, because it increases the network impact on response time and incurs transfer overhead on the servers. To this purpose, we propose two schemes that redirect heaviest requests only. **R-size** redirects requests for Web pages larger than a certain size. The motivation is that Web workload (file sizes or dynamic requests) follow a *heavy-tailed* distribution [11]. Hence, a very small fraction of the largest files determines a large fraction of the load. We use the average size of a static Web page and its objects as the default size threshold for requests of static contents, while we use the mean magnitude of the processing cost for dynamic Web page requests. As a further policy, **R-num** considers for redirection only those pages consisting of a large number of embedded objects (*hits*). We use the average number of hits in a Web page as the default threshold for deciding about redirection.

Once selected the load to be redirected, the *location policy* selects the Web cluster that will receive the redirected requests. We consider three alternatives. The stateless **RR** policy redirects the selected requests to all Web clusters in a round-robin way. The **Load** policy uses some system load information; it redirects requests to the Web cluster which has the lightest load, as observed in the past *check-cluster-load* interval. The **Prox** policy uses some network load information; it redirects requests to the Web cluster that in the past *check-network-load* interval resulted best connected to the redirecting cluster. Information about Web cluster load and dynamic network proximity is provided through messages exchanged by the Dispatchers. Table 1 summarizes the server redirection strategies we analyze. We will denote the redirection algorithms by considering selection and location policy. For example, **R-size_Prox** is the algorithm that uses the size-based selection policy, and the network proximity location policy.

4. System and client model

We divide the Internet into K geographical regions located in different world areas. Each region contains a Web cluster, one or more DNS servers for the Web site (see Figure 1), and various *client domains*. The popularity of the domains in each region is described through a Zipf distribution with parameter $\alpha = 0.8$ that corresponds to a highly skewed function [11].

We define the following time-dependent model to represent the variability of traffic coming from Internet regions, so that the most popular region can change during the simulation runs. Let $cp_i(t)$ be the percentage of clients belonging to region i at time t, where $\sum_{i=1}^{K} cp_i(t) = 1$. This popularity function changes dynamically as in Figure 3a of [2]. To take into account world time zones, we assume that the time in region $(i+1) \bmod K$ is shifted of t_K hours forward with respect to region i.

Client arrivals to the Web system follow an exponential distribution [14], where the mean interarrival time is set to 0.05 seconds, if not otherwise specified. Each client is assigned to one Internet region with probability $cp_i(t)$, and assigned to one client domain in that region through the corresponding Zipf distribution. The period of visit of each client to the Web site is called *session*. The workload model incorporates the most recent results on Web characterization. The high variability and self-similar nature of Web access load is modeled through heavy-tailed distributions such as Pareto, lognormal and Weibull distributions [2, 11, 14].

The number of consecutive Web pages a user will request from the Web system (*page requests* per session) follows the inverse Gaussian distribution [11]. The client's silent time between the retrieval of two successive Web pages, namely the *user think time*, is modeled through a Pareto distribution [11]. The self-similarity of Web traffic requests is explained with the superimpositions of heavy-tailed ON-OFF periods. The number of objects that make up a whole Web page, including the base HTML object and its in-line referred files, also follows a Pareto distribution [11]. Web file sizes typically show extremely high variability in size. The function that models the distribution of the *object size* requested to the Web site varies according to the object type. For HTML objects, it is obtained from a hybrid distribution, where the body follows a lognormal distribution, while the tail is given by a heavy-tailed Pareto distribution [2, 11]. For in-line objects in a page, the size distribution is obtained from the lognormal function [11]. Table 2 shows the distributions, probability mass functions and parameter ranges for the workload model.

In the simulation experiments we assume that there are $K = 4$ regions. Each Web cluster has 4 homogeneous server nodes for a total of 16 Web servers. Each simulation run lasts for 24 hours, and the time difference among the regions is $t_K = 6$. Most experiments are carried out for a long-term system utilization kept around 0.6-0.65 of the capacity of the entire Web system. The time to serve each client request includes all delays at the Web cluster, such as dispatching time, parsing time, service time for the page request and all embedded objects or redirection time. If not otherwise specified, the check-server-load interval is set to 8 seconds, while check-cluster-load and check-network-load intervals are set to 16 seconds. The threshold value of server utilization that triggers the redirection mechanism is set to 0.75. No substantial changes in results were observed for thresholds equal to 0.80 or 0.85.

5. Network model

The network model aims at providing a controllable testbed where the transmission between Web clusters and clients has a cost, but the network does not represent the main bottleneck. This choice is motivated by the focus of this paper on system management algorithms. Moreover, network service providers are continuously improving network infrastructure to accommodate higher bandwidth.

The goal is to measure the impact of redirection on response time compared to request response time of not redirected requests. For this reason, we do not consider real Internet connections, network hierarchies, and narrow network bandwidth in the last mile. The model for communication delays is based on the following assumptions that, although simplified and subject to further improvements, introduce less arbitrary choices than pseudo-real network hierarchies and connections that could affect a fair comparison of the proposed algorithms.

In the model, we refer to the HTTP/1.1 protocol that uses persistent connections and pipelining that is, the connection is left open between consecutive objects transmissions or at least for 15 seconds and the browser can send multiple requests without waiting for a response. From [7] we have that the time to transmit n objects belonging to the same page between region i and j is given by

$$T_{tr,n} = 2rtt_{ij} + \sum_{k=1}^{n} (S_{req_k}/ab_{ij} + S_{res_k}/ab_{ji}) \qquad (1)$$

where rtt_{ij} and ab_{ij} are the *round-trip time* and the *available bandwidth* between region i and j, respectively; S_{req} and S_{res} are the size of the client request and server response for each object k, respectively. Equation 1 denotes that the time to transmit any message over the Internet is given by the time to establish a connection plus the ratio of the message size divided by the available bandwidth. Let us first discuss the message size by distinguishing client request and server response.

Although the traffic generated by Web clients is only about 6-8% of the global traffic (measured in bytes) that is,

Category	*Distribution*	*PMF*	*Range*	*Parameters*
Pages per session	Inverse Gaussian	$\sqrt{\frac{\lambda}{2\pi x^3}} e^{\frac{-\lambda(x-\mu)^2}{2\mu^2 x}}$	$x > 0$	$\mu = 3.86$, $\lambda = 9.46$
User think time	Pareto	$\alpha k^{\alpha} x^{-\alpha-1}$	$x \geq k$	$\alpha = 1.4$, $k = 1$
Objects per page	Pareto	$\alpha k^{\alpha} x^{-\alpha-1}$	$x \geq k$	$\alpha = 1.245$, $k = 2$
HTML object size	Lognormal Pareto	$\frac{1}{x\sqrt{2\pi\sigma^2}} e^{\frac{-(ln\, x-\mu)^2}{2\sigma^2}}$ $\alpha k^{\alpha} x^{-\alpha-1}$	$x > 0$ $x \geq k$	$\mu = 7.630$, $\sigma = 1.001$ $\alpha = 1$, $k = 10240$
In-line object size	Lognormal	$\frac{1}{x\sqrt{2\pi\sigma^2}} e^{\frac{-(ln\, x-\mu)^2}{2\sigma^2}}$	$x > 0$	$\mu = 8.215$, $\sigma = 1.46$

Table 2. Workload model.

$1/9$ of the traffic generated by Web servers [13], it is important to consider the client message component because of overheads of the HTTP redirection mechanism. The size of a client request S_{req} is exponentially distributed with mean equal to 358 bytes and typically fits a single packet. The server response consists of multiple files with various sizes following heavy-tailed distributions such as those described in Section 4.

The second parameter in Equation 1 is the *available bandwidth* ab_{ij} that measures the communication delays between two Internet regions. We assume that these delays are due to a static factor (*basic bandwidth*) and a dynamic factor (*traffic*). The basic bandwidth bb_{ij} between region i and j can be assumed deterministic. In particular, the basic bandwidth within a region and between this region and others is supposed to be *large*, *medium-large*, *medium-narrow* and *narrow*. The bandwidth values reported in Table 3 are taken in [7, 13]. We also carried out some sensitivity analysis (not shown due to space limits) as a function of the basic bandwidth. By halving or doubling the basic bandwidth among all Internet regions, we observed that this parameter does not affect the main conclusions of this paper.

Bandwidth type	*Basic bandwidth*	*Round-trip time*
Large	1.35 Mbps	[40, 70] msec
Medium-large	0.9 Mbps	[120, 150] msec
Medium-narrow	0.7 Mbps	[180, 210] msec
Narrow	0.4 Mbps	[270, 300] msec

Table 3. Network model.

We assume a good connection inside each region (that is, bb_{ii} = large), and the same bandwidth in both directions (that is, $bb_{ij} = bb_{ji}$). To determine the bandwidth between the four regions, in the simulation experiments we consider the following values: bb_{12} = large, bb_{13} = medium-large, $bb_{23} = bb_{14} = bb_{24}$ = medium-narrow, bb_{34} = narrow.

We model the Internet traffic as a random parameter that reduces the basic bandwidth. This parameter changes dynamically to take into account time zones and busy/quiet hours of each region. Let $\pi_{ij}(t)$ be a value between 0 and 1 that denotes how much fraction of the basic bandwidth can be used by a connection between region i and j. Therefore, the available bandwidth at time t is $ab_{ij}(t) = \pi_{ij}(t) bb_{ij}$. Since the traffic depends on the number of clients in the regions, we assume that the fraction of bandwidth available to a connection starting from a region and ending in another one, is related to the popularity of the two end-point regions. To model this assumption, we define a discrete random variable Z, that represents the connection type and can take one of the following three values: lc lucky connection or light network traffic, nc normal connection or mild network traffic, uc unlucky connection or heavy network traffic.

Let $\rho_i^Z(t)$ be the probability to have a connection of type $Z \in \{lc, nc, uc\}$ in region i, where $\rho_i^{lc}(t) + \rho_i^{nc}(t) + \rho_i^{uc}(t) = 1$. The distribution of $\rho_i^Z(t)$ is related to the popularity of region i; therefore, it is time-dependent. Let $\pi_i^Z(t)$ be the fraction of bandwidth for a connection of type Z starting from region i. We assume that a connection can experiment a fraction of bandwidth varying in the interval I_b^Z, defined as $I_b^{lc} = [0.75, 1.0]$, $I_b^{nc} = [0.5, 0.75)$, $I_b^{uc} = [0.2, 0.5)$. Hence, $\pi_i^Z(t)$ is a random variable uniformly distributed in the interval I_b^Z, and is obtained in the following way: first, Z is determined by using the distribution for $\rho_i^Z(t)$; then, the value for $\pi_i^Z(t)$ is selected randomly according to the Z value. The fraction of bandwidth for a connection between region i and j is given by $\pi_{ij}(t) = 0.5\pi_i^Z(t) + 0.5\pi_j^Z(t)$. This choice takes into account different levels of popularity that any Internet region may have at time t. For example, if a lot of requests will arrive from region i at time t, while region j has a low number of clients, then the traffic for a connection between region i and j will result low/medium with high probability. On the other hand, if both the regions are highly populated at time t, then the connection is unlucky with high probability. Moreover, this network model includes the possibility that a client connected to the server through a large basic bandwidth may experiment unlucky connections due to heavy traffic. Dually, a client far from the server may experience a medium bandwidth thanks to a lucky connection. Since Paxson points out that Internet paths are heavily dominated by a single prevalent route and that about two-thirds of the Internet paths have routes persisting for either days or weeks [10], we consider the same available bandwidth value for the entire client session.

The last parameter in Equation 1 to be discussed is the *round trip time* that denotes the time necessary to establish the TCP connection between the client and the server inside and between Internet regions [7]. We set four intervals of round-trip delays, one for each type of bandwidth, and we choose randomly the round-trip time in the corresponding set. The interval values are reported in Table 3.

6. Experimental results

In this section we consider the objective of minimizing the response time experienced by users that access a geographically distributed Web site. We use the cumulative distribution of the *page response time* as the main performance metric, because in a highly variable system it is more significant than average values. The page response time corresponds to the interval elapsed between the submission of the client request for a given page and the arrival at the client of all objects corresponding to the page request. It includes TCP connection time, delays at Web server, network transmission time, and possible redirection overheads. To verify which proposed load management algorithm guarantees Quality of Service, we use also the *90-percentile* of the page response time that is, the page response time limit that the Web site guarantees with 0.9 probability [9]. Another interesting performance parameter is the *percentage of redirected requests* because a further goal of this study is to propose and compare algorithms that minimize this value.

The simulator, based on the Independent Replication Method, was implemented using the CSIM18 package. Each value is the result of ten or more simulation runs (each lasting 24 hours) with different seeds. The goal was that for all simulation results the 95% confidence interval resulted to be within 5-6% of the mean.

Figure 2 compares the cumulative distribution for the DNS proximity assignment with that of various redirection schemes based on the simplest selection policy (R-all) that lets an overloaded Web server to redirect all requests to a different Web cluster. This figure shows that even this naive server redirection approach achieves substantial performance improvement when compared to systems where scheduling is done by DNSes and Web cluster dispatchers only. All redirection policies guarantee that the maximum response time is below 20 seconds while analogous performance is guaranteed to only 80% of requests assigned by proximity-based DNS algorithms. The 90-percentile of the best redirection policy is equal to 7 seconds that is, about the value that many studies and surveys consider the service level acceptable to most Web users. The analogous service level is guaranteed to less than 70 percent of clients of Web systems based on two-level scheduling mechanisms (see DNS curve). If we compare the three location policies (RR, Load, Prox), we observe that the circular reassignment among all Web clusters achieves best performance, the policy that reassigns requests to the least loaded cluster performs slightly worse, while the Prox location policy redirecting all requests to the closest Web cluster shows significantly worse results. The motivation for this results is that when we reassign all requests, it seems preferable to spread the load among multiple Web clusters than concentrating all redirected load to one cluster, even if the receiver cluster is the lowest loaded or the closest. We observed that typically a burst of redirected requests improves performance on the sender cluster, but causes a temporary overload on the receiver cluster that, on its turn, activates the redirection mechanism. The consequence is that the Web system remains unstable for longer periods with tangible consequences on response time.

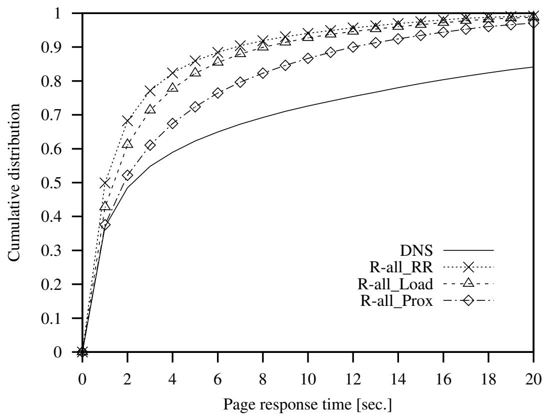

Figure 2. Cumulative distribution of page response time for the R-all selection policy.

We next consider selection policies that limit redirection on the basis of the estimated size of the requested page (R-size) or number of hits (R-num). Due to space limits, we plot the results of the former policy only, even because similar results are obtained for the latter algorithm. Figure 3 shows that redirecting only client requests having a size larger than the average page size improves for small response time values the performance achieved by the naive selection scheme. Now the relative performance order between Load and RR location policies is inverted, and R-size_Prox performs much better than corresponding R-all_Prox. This confirms that redirecting only a subset of requests reduces instability because the receiver cluster is not overwhelmed by bursts of additional requests. Hence, it becomes convenient to consider for redirection the least loaded cluster instead of proximity or circular assignments.

In the following experiments we compare the perfor-

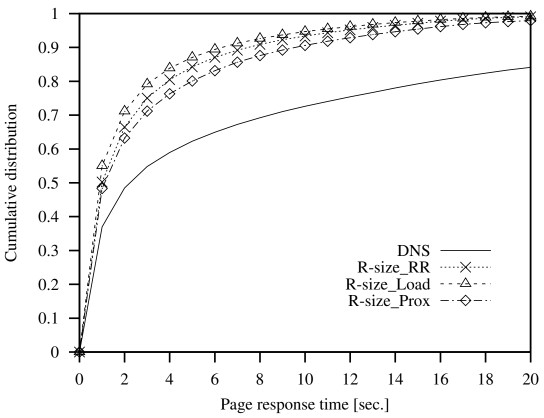

Figure 3. Cumulative distribution of page response time for the R-size selection policy.

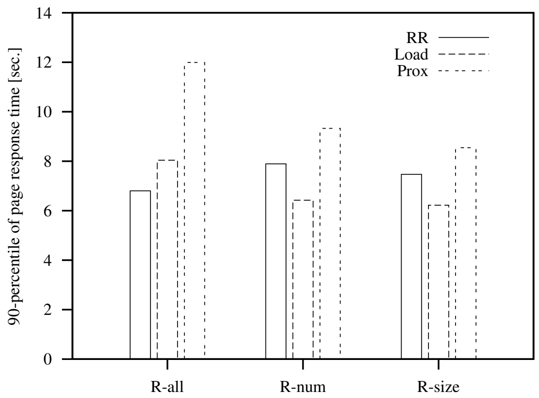

Figure 4. 90-percentile of page response time for various selection and location policies.

mance of various selection and location policies by taking into account the 90-percentile of the page response time that is considered the most important parameter to evaluate QoS of a Web system [9]. In Figure 4 the results are grouped according to the selection policy; different location policies are considered inside each group. We observe that the Prox location policy remains the worst solution for all selection policies. RR location is fine to spread evenly the load when all requests are reassigned by overloaded servers, but this effect is less important when only a subset of requests is reassigned. When redirection is activated only on the most resource consuming requests, it seems useful to use some state aware location policy to select the most appropriate Web cluster. The Load location policy performs better than the stateless RR for both R-num and R-size selection schemes. Finally, it is important to observe that the 90-percentile of page response time for Web systems based on two-levels scheduling (that is, DNS proximity and Web cluster dispatching) is equal to 32 seconds. This is more than six times the value achieved by the best redirection policies.

Performance shown in Figures 2, 3 and 4 are useful for algorithm comparison and must not be considered as absolute time values. The proposed network model is incomparable to the complexity of real Internet. In particular, we believe that redirection may have an impact on performance even greater than that shown in previous figures when a real geographical environment is considered. Consequently, the reduction of redirections achieved by R-num and R-size selection policies may limit network impact on latency time even more than that shown in Figure 4. To this purpose, in Figure 5 we show the redirection percentages for the selection and location policies considered in Figure 4. We can see that the naive selection policy tends to redirect close to 20% of the requests reaching the Web site, while the analogous metrics is below 5% for R-num and R-size selection policies. This difference is consistent if we consider overheads that each redirection causes on Web servers (e.g., time lost to open and close a TCP connection) and network (e.g., round-trip and new TCP connection delays).

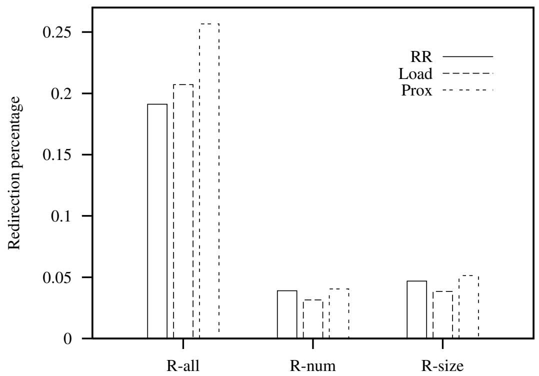

Figure 5. Redirection percentage for various selection and location policies.

To verify the robustness of the results, in the last experiments we analyze the sensitivity of the selection and location strategies to some system parameters. In Figure 6 we analyze the 90-percentile of the response time as a function of the check-server-load interval for the best selection and location policies. As expected, performance of all policies tends to improve for lower values of this interval. It is reasonable to use relatively short periods such as 8 seconds, because server load evaluation does not necessarily imply the trigger of the redirection mechanism.

Figure 7 shows the 90-percentile of the page response

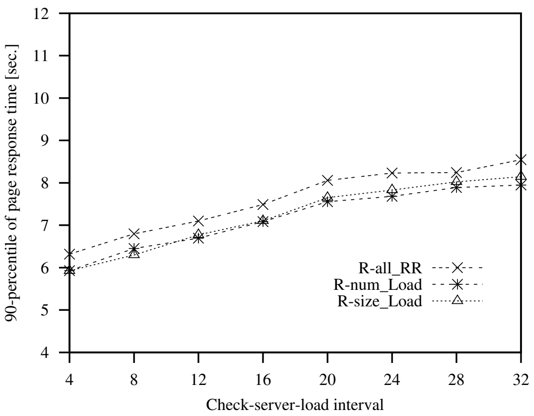

Figure 6. Sensitivity to the frequency of the check-server-load interval.

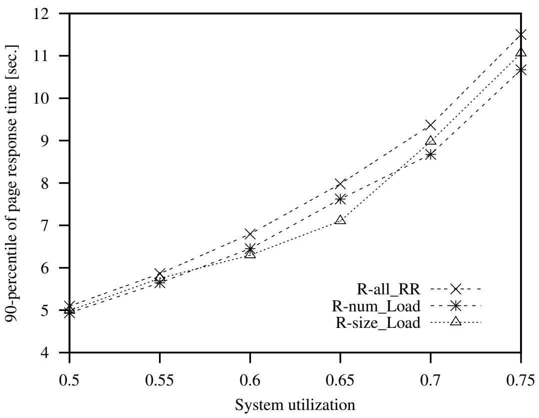

Figure 7. Sensitivity to the long-term system utilization.

time when the long-term utilization of the Web system varies from 0.5 to 0.75. All the policies that redirect requests show a similar behavior when the load increases. Although the limit for guaranteed response time doubles, in the Web environment subject to heavy-tailed arrivals, a 0.75 sustained utilization corresponds to a highly loaded system that would require more resources. The need of the third-level redirection mechanism is well motivated by the results of a Web site using DNS proximity algorithm only: the 90-percentile of the page response time for 0.5, 0.6, and 0.7 system utilization is equal to 10, 32 and 56 seconds, respectively. This shows that DNS only is not able to guarantee request response time even when the system utilization is subject to little increases.

7. Conclusions

Highly popular Web sites require a distribution of content and servers over geographical regions to avoid network bottlenecks. Various proposals consider a geographically distributed architecture where the DNS of the Web site evaluate network proximity and requests reach the closest Web cluster, where a Dispatcher executes second-level scheduling. We demonstrate that serving closest requests only may cause unbalanced servers and may increase system impact on response time, because arrivals from each Internet region is highly variable, depending on population, time zones, and day hour. To achieve a scalable and balanced Web system, we integrate DNS proximity and Dispatcher scheduling with an HTTP redirection mechanism that any Web server can activate at its need. We show that this third-level dispatching, when integrated with some limitation on request redirection, is a powerful mechanism to enhance QoS of geographically distributed Web sites.

References

[1] D. Andresen, T. Yang, and O. H. Ibarra. Towards a scalable distributed WWW server on networked workstations. *J. of Parallel and Distributed Computing*, 42:91–100, 1997.

[2] M. F. Arlitt, R. Friedrich, and T. Jin. Workload characterization of a Web proxy in a cable modem environment. *ACM Performance Evaluation Review*, 27(2):25–36, Aug. 1999.

[3] V. Cardellini, M. Colajanni, and P. S. Yu. Dynamic load balancing on Web-server systems. *IEEE Internet Computing*, 3(3):28–39, May 1999.

[4] V. Cardellini, M. Colajanni, and P. S. Yu. Redirection algorithms for load sharing in distributed Web-server systems. In *Proc. of IEEE 19th Intl' Conf. on Distributed Computing Systems*, pages 528–535, Austin, TX, June 1999.

[5] R. L. Carter and M. E. Crovella. Server selection using dynamic path characterization in wide-area networks. In *Proc. of IEEE Infocom 1997*, pages 1014–1021, 1997.

[6] Cisco. Cisco DistributedDirector. http://www.cisco.com/warp/public/cc/cisco/mkt/scale/distr/.

[7] J. Heidemann, K. Obraczka, and J. Touch. Modeling the performance of HTTP over several transport protocols. *IEEE/ACM Trans. on Networking*, 5(5):616–630, Oct. 1997.

[8] G. D. H. Hunt, G. S. Goldszmidt, R. P. King, and R. Mukherjee. Network Dispatcher: A connection router for scalable Internet services. *Computer Networks*, 30:347–357, 1998.

[9] D. Krishnamurthy and J. Rolia. Predicting the QoS of an electronic commerce server: Those mean percentiles. In *Proc. of Workshop on Internet Server Performance*, Madison, WI, June 1998.

[10] V. Paxson. End-to-end routing behavior in the Internet. *IEEE/ACM Trans. on Networking*, 5(5):601–615, Oct. 1997.

[11] J. E. Pitkow. Summary of WWW characterizations. *World Wide Web*, 2(1-2):3–13, Mar. 1999.

[12] T. Schroeder, S. Goddard, and B. Ramamurthy. Scalable Web server clustering technologies. *IEEE Network*, pages 38–45, May 2000.

[13] K. Thompson, G. J. Miller, and R. Wilder. Wide-area Internet traffic patterns and characteristics. *IEEE Network*, 6(11):10–23, Nov. 1997.

[14] W. Willinger and V. Paxson. Where Mathematics meets the Internet. *Notices of the American Mathematical Society*, 45(8):961–970, Aug. 1998.