DNS-BASED LOAD BALANCING FOR WEB SERVICES

Alan Nakai, Edmundo Madeira and Luiz Eduardo Buzato
Institute of Computing, Unicamp - University of Campinas, Campinas, SP, Brazil

Keywords: Load balancing, Web services.

Abstract: A key issue for good performance of geographically replicated web services is the efficiency of the load balancing mechanism used to distribute client requests among the replicas. This work revisits the research on DNS-based load balancing mechanisms considering a SOA (Service-Oriented Architecture) scenario. In this kind of load balancing solution the Authoritative DNS (ADNS) of the distributed web service performs the role of the client request scheduler, redirecting the clients to one of the server replicas, according to some load distribution policy. This paper proposes a new policy that combines client load information and server load information in order to reduce the negative effects of the DNS caching on the load balancing. We also present the results obtained through an experimental tesbed built on basis of the TPC-W benchmark.

1 INTRODUCTION

With the increasing adoption of SOA (Service-Oriented Architecture), a new scenario arises, where highly accessed web applications are deployed as web services and clients are not web browsers accessing a web site, but other enterprises using the services of other providers.

This new kind of scenario may require higher levels of web service dependability because of the QoS contracted by the service consumers. Thus, providers replicate their applications in clusters geographically distributed linked via the Internet for the sake of fault-tolerance and performance. A key issue for good performance in these environments is the efficiency of the load balancing mechanism used to distribute client requests among the replicated services.

This work revisits the research on DNS-based load balancing mechanisms for gegraphically replicated web services. In this kind of load balancing solution the Authoritative DNS (ADNS) of the distributed Web service performs the role of the client request scheduler, redirecting the clients to one of the server replicas, according to some load distribution policy. Differently from previous works, that considered the simple browser-server scenario, in our work we consider a SOA scenario.

It is known that large Internet corporations – e.g. Google (Barroso et al., 2003) and Akamai (Pan et al., 2003; Su et al., 2006) – use DNS-based load balancing mechanisms. These mechanisms benefit from the existing DNS infrastructure, providing transparency for the Web clients. However, this kind of strategy has a main limitation: the low control of the ADNS over the load balancing caused by the DNS caching system, that prevents name resolution queries to reach the ADNS.

The main contribution of this paper is the proposal of a new DNS-based load balancing mechanism that uses client load information in order to better distribute the load among the replicated servers – the Current Relative Minimum Load (CRML). Besides, the mechanism reduces the negative effects of the DNS caching over the load balancing through the cooperation of the ADNS and the servers.

We also present the evaluation of our load balancing policy over an experimental testbed implemented on basis of the TPC-W (TPC, 2002), a well accepted E-Commerce benchmark. The experiments show that the CRML policy behaved as good as other policies in the scenario in which the ADNS had full control of the name resolution queries and behaved better than the others in a scenario were the ADNS had partial control.

The remainder of this text is organized as follows. Section 2 provides an overview of the DNS system and the DNS-based load balancing mechanisms. Section 3 presents related works. In Section 4 we de-

Nakai A., Madeira E. and Eduardo Buzato L.
DNS-BASED LOAD BALANCING FOR WEB SERVICES.
DOI: 10.5220/0002805000950100
In *Proceedings of the 6th International Conference on Web Information Systems and Technology (WEBIST 2010)*, page
ISBN: 978-989-674-025-2

scribe our new load balancing mechanism. Section 5 presents the testbed used for the evaluation of our policy and Section 6 shows the experimental results. Section 7 concludes the paper with our final comments and future work.

2 BACKGROUND

In the DNS-based load balancing mechanisms, the authoritative nameserver of the distributed Web server performs the role of the client request scheduler. When it receives a request for URL resolution, it replies the IP address of one of the server nodes, according to some load distribution policy.

The main advantage of this kind of load balancing mechanism is that it benefits from the existing DNS infrastructure. This makes these mechanisms immediately deployable in today's Internet (Pan et al., 2003).

Unfortunately, a limitation of DNS-based load balancing mechanisms is the weak control of the ADNS over the load balancing, caused by the DNS caching, that prevents a portion of DNS queries to reach the ADNS.

The simplest DNS-based load balancing mechanism is the *Round Robin* (RR) policy. In this policy, when a request for name resolution arrives at the ADNS, it responds with the address of the next replica of its list, in a rotative way. More sophisticated approaches apply information from server node utilization and/or client domain information to select a server replica.

3 RELATED WORK

The use of information about server node utilization in DNS-based load balancing mechanisms is exemplified in (Colajanni et al., 1998; Yokota et al., 2004; Moon and Kim, 2005). In these works, an agent monitors the states of the servers and reports this information to the ADNS. When a name resolution query arrives, the ADNS uses the utilization information of the server nodes to assign one of the replicas to the client. Many kinds of information can be used in the ADNS decision, such as request queue length, CPU, network, or memory usage.

There are two types of client domain information that can be used in the load balancing: client proximity and client domain load. In the first case, the ADNS tries to assign the nearest server to the client (Barroso et al., 2003; Pan et al., 2003; Su et al., 2006). A main concern in this kind of load balancing mechanism is to estimate the proximity between clients and servers. Since the ADNS does not have information about the client host, a solution for this problem is to assume that clients are located near to their local nameservers and estimate the distance between servers and local nameservers.

The capacity of estimating the load generated by client domains may be very useful for load balancing mechanisms. This information allows the ADNS to treat differently domains that generate high request rates (hot domains) from the others (cold domains). The works (Colajanni et al., 1998), (Colajanni and Yu, 2002), and (Chatterjee et al., 2005) present promising results using information about client domain load for: (i) avoiding the assignment of hot domains to the same servers; (ii) estimating the real load of each server; and/or (iii) applying different TTLs for name resolutions addressed to hot and cold domains.

In this work, we present a new DNS-based load balancing mechanism that combines information from clients and servers to alleviate the effects of the DNS caching on the load balancing.

4 THE CRML POLICY

This section presents our proposed load balancing policy, the Current Relative Minimum Load (CRML).

4.1 Rationale

The idea for the algorithm was motivated by the analysis of the three load balancing policies described in (Colajanni et al., 1998; Colajanni and Yu, 2002). Here is a summary of them:

- ***Least Utilized Node*** **(LUN):** the ADNS assigns a request to the less utilized node, based on the most recent server load information;
- ***Two Tier Round Robin*** **(RR2):** the ADNS divides client domains into two groups, hot and cold domains. Hot domains are those that generate high number of requests. The policy applies the round robin strategy separately to each group, trying to avoid the assignment of requests proceeding from hot domains to the same server nodes;
- ***Minimum Residual Load*** **(MRL):** the ADNS maintains a table containing all the assignments and their times of occurrence. Based on this table and on estimates of the request rate of each client domain, the policy calculates the load of each web server replica and assigns an incoming name resolution request to the least loaded one;

A deficiency of LUN is that the information used by the algorithm in decision making is often outdated. This happen because the algorithm considers only the last utilization information received from the servers, however, the state of the servers may have changed at the moment of a new assignment. In order to deal with this deficiency it should be necessary to make decisions based not only on the last utilization information but also considering the assignments performed after this information has arrived.

In our experiments, the RR2 policy presented a significative improvement in comparison to RR. This result shows us that to handle differently hot and cold domains is a good strategy. A deficiency of RR2 is that, even if a server is overloaded, the algorithm continues to assign new name resolution requests to that server because of the round robin strategy.

The MRL works quite fine because of its ability of estimating the total load of the servers based on previous name resolution assignments. Nevertheless, if the control of the ADNS decreases, by the reason of the caching of intermediary DNS servers, the MRL assignment table becomes incomplete and leads the ADNS to make wrong decisions. Moreover, if the number of clients is very high, it could be very expensive to maintain the assignment table.

4.2 Policy Description

In the CRML policy, we follow the hypothesis that the distribution of hot domains among the server replicas dictates the success of the load balancing mechanism. Thus, as well as RR2, the CRML policy divides the clients into two groups, hot and cold domains. Cold domains are distributed among the server replicas using the ordinary round robin strategy.

In order to better distribute the load generated by hot domains, the ADNS maintains an assignment table containing the assignments of servers to hot domains and their time of occurrence. Note that the set of assignments in this table is potentially incomplete, because many clients might have received name resolutions from intermediary DNS servers. Hence, the ADNS cooperates with the servers to compensate the effect of the DNS caching. Each server tracks the current set of hot domains from which it is receiving requests and report this information to the ADNS. Besides, servers also report estimates of the load (request rates) that hot domains are generating. Combining the information of the assignment table and the information proceeding from the servers, the ADNS can estimate the request rate each server is receiving from hot domains.

Let S be the set of web servers; let $l_i(a)$ be the load generated by the assignment a to the server i; let A_{Ki} be the set of assignments to the server i, known by the ADNS, and whose the TTL has not expired; and let A_{Hi} be the assignments reported by the server i. When the ADNS receives a name resolution request from a hot client it computes:

$$CRML = \min_{i \in S} \left\{ \sum_{a \in \{A_{Ki} \cup A_{Hi}\}} l_i(a) \right\}$$

and assigns it to the the corresponding server.

The efficiency of the CRML depends on how the set of assignments that the ADNS knows ($\{A_{Ki} \cup A_{Hi}\}$) is close to the reality. The ADNS knowledge is limited by the staleness of the information reported by the servers. The last sets of assignments (A_{Hi}) received from the servers may lack assignmets that were established after the information were sent and may contain assigments that are not valid anymore.

In a certain limit, the use of the ADNS assignment table reduces the staleness of the server-side information. Another way to reduce the effects of stale server-side information is to exclude any assignment of the client that is requesting a name-resolution from the CRML calculation. Previous assignments related to this client is obviously not valid anymore, since it is requesting a new one.

Since the assignment table of the CRML does not store the state of all clients, only the states of hot clients, the cost for maintaining the table is smaller than using MRL.

4.3 Software Architecture

This section presents the software architecture that supports the proposed load balancing policy. The architecture is illustrated in Figure 1.

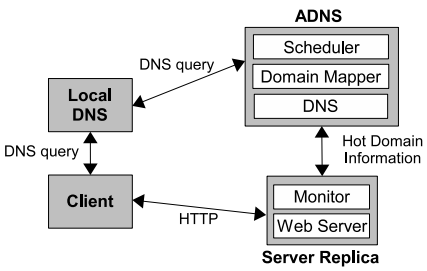


Figure 1: CRML Architecture.

In this architecture, the server replicas are composed of two modules: (i) the web server module, that processes the incoming HTTP requests, and (ii) the monitor module, that collects information about hot client domains and reports it to the ADNS.

The ADNS is a DNS server extended with two modules, the domain mapper and the scheduler. The former is responsible for identifying if an incoming request comes from a hot or a cold domain. The latter uses the information reported by the server replicas and the information stored in the DNS assignment table to decide the better replica to which redirect a client.

5 EXPERIMENTAL TESTBED

In order to evaluate our load balancing mechanism, we implemented an experimental testbed, the Lab4WS (Lab for Web Services). This section presents this testbed. More details about the Lab4WS Testbed can be found in (Nakai et al., 2009).

The testbed includes the implementation of a SOAP Web service based on the TPC-W benchmark (TPC, 2002), a transactional benchmark for E-Commerce web sites that is well accepted by the research community. The service consists of a set of 20 operations that allow clients to search and buy products. A proxy placed in front of each service replica collects client information and reports to the DNS the list of hot clients that are accessing the replica and the estimative of load generated by those clients.

The load generation of our testbed is performed by a set of client emulators. Each load generator emulates the load of an E-Commerce Web site that serves an entire Web domain and generates requests to the TPC-W Web Service. The generated load follows the TPC-W specification, which specifies three kinds of workloads that vary according to the percentage of read and write operations. As in previous works – e.g. (Colajanni and Yu, 2002) – the load generation is divided between different clients according to the Zipf's distribution, where the probability of a client to belong to the ith domain is proportional to $1/i^x$. The testbed user can vary the skewness of the distribution changing the exponent x of the function. This solution was motivated by previous works that demonstrated that if one ranks the popularity of client domains by the frequency of their accesses to a Web site, the size of each domain is a function with a big head and a very long tail.

The testbed also contains a DNS emulator that materializes the ADNS of the web system. This emulator allows the testbed user to deploy new load balancing policies in a friendly way. The effect of the DNS caching is implemented as a mechanism that controls the percentage of name resolution requests that reach the ADNS. The testbed user can define this percentage. The client emulators randomly decide what name resolution requests are sent to the ADNS. When a name resolution request is not sent to the ADNS, the client emulator reuses the last name resolution it received, emulating a caching effect.

6 CRML EVALUATION

In our experiments, we consider a scenario in which a set of retailers form partnerships with a large E-Commerce enterprise to outsource the application logic of their e-store Web sites. Each e-store is visited by a great number of end customers, and accesses the E-Commerce enterprise services via Web Services. The E-Commerce enterprise needs to distribute the load incoming from the e-stores among its geographically distributed replicas of servers.

6.1 Methodology

For these experiments, our testbed was deployed on the Emulab[1] network testbed. Emulab is a user-configurable lab environment that allows users to model and emulate network topologies on a cluster, varying parameters such as latency and bandwidth. We ran the experiments using 5 TPC-W Web Service replicas and 16 machines running Emulated Clients with a load equivalent to 300 requests/second (75 requests/second for each secondary replica). All machines were Pentium Xeon 64, 3GHz, with 2GB of memory. A latency of 100 ms was introduced into the links between the machines, to emulate a wide area network latency.

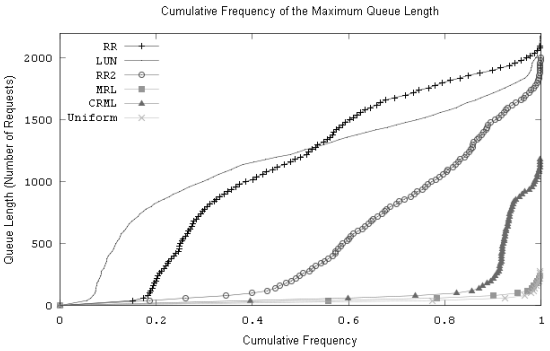
In order to evaluate the different load balancing mechanisms, we adopted two main metrics: the maximum system queue length and the response time observed by the clients. The maximum system queue length is the largest number of requests waiting to be answered on a service replica, observed in a given instant, among all service replicas.

If the incoming load exceeds the service replica capacity, the response rate becomes lower than the request rate and the incoming requests tend to accumulate at the server. Thus, the request queue grows. Since the requests take longer to be served, the response times observed by the client increase, and the system throughput goes down.

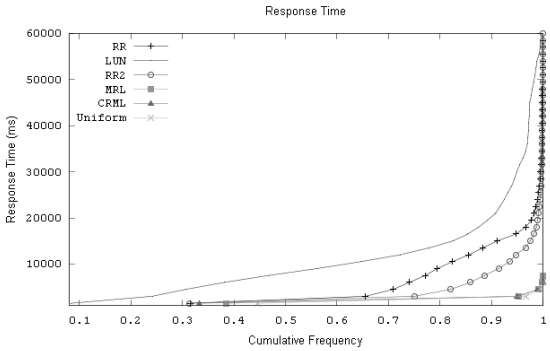
6.2 Experimental Results

We have evaluated the CRML policy on two scenarios. In the first, we compare five load balancing mechanisms (RR, LUN, MRL, RR2, and CRML) with

[1] https://www.emulab.net/

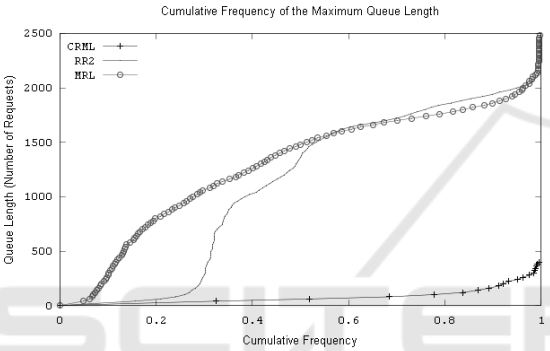


(a) Cumulative frequency of the maximum system queue length.

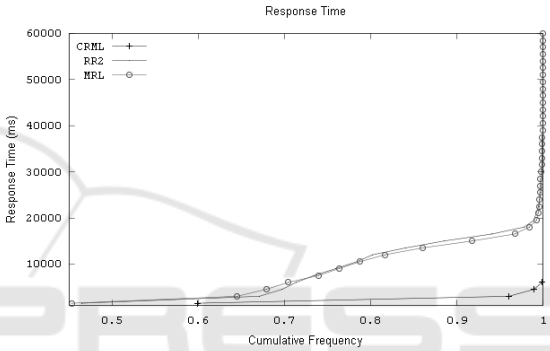


(b) Cumulative frequency of response times.

Figure 2: Experimental results: 100% of ADNS control.



(a) Cumulative frequency of the maximum system queue length.



(b) Cumulative frequency of response times.

Figure 3: Experimental results: 30% of ADNS control.

a uniform distribution, which represents the "ideal" load balancing, assuming that the ADNS has total control over the name resolution requests. In the second scenario, we compare MRL, RR2, and CRML, which presented the better performances in the first scenario, assuming that only 30% of the name resolution requests reach the ADNS.

In these experiments we used the DNS TTL=60s and interval for server information dissemination of 10s. The threshold adopted to identify hot clients was a load equal to 4 requests/second for MRL and CRML, and a load equal to 10 requests/second for RR2. Load was generated using Zipf distribution with $x = 1.0$.

Figure 2(a) shows the cumulative frequency of the maximum system queue length for the first scenario. The MRL and CRML mechanisms presented good performances, showing maximum system queue lengths close to the uniform distribution. This result was expected because in this scenario, where the ADNS has full control over the name resolution requests, these two policies are able to compute the states of the servers with a high precision. The RR2 does not performed as good as CRML and MRL, but its performance was better than RR, showing that the use of client load information improved the round robin strategy. The LUN policy presented the worst performance, showing that the simple use of information about server utilization is not effective for load balancing.

The response times obtained in the experiments of the first scenario reflected the results of the maximum system queue length measurement. Figure 2(b) shows the cumulative frequency of response times for the *subjectSearch* operation, which performs a search for items by the subject in the book store. While, using CRML and MRL, 95% of the operation requests were answered in less then 3s, using RR2, about 25% of the requests were answered in more than 3s. Using RR, about 35% of the operation requests were answered in at least 3s, and, using LUN, more than 80% of the requests were answered in at least 3s.

The results obtained in the experiments of the second scenario are shown in Figure 3. As expected, due to the low control of the ADNS, the RR2 and MRL policies presented worse results than in the first scenario. In more than 50% of the time, these policies showed maximum system lengths larger than 1500 (Figure 3(a)). Differently from the others, CRML worked quite fine in the second scenario, presenting maximum system lengths smaller than 250 in 95% of the time.

The performance of the three policies is also perceived in the response time graph (Figure 3(b)). While the CRML presented response times lower than 3s in more than 95% of the requests, RR2 and MRL presented response times higher than 3s in 35% of the requests.

The experimental results show that, in the scenario where the ADNS had full control of the name resolution queries, CRML performed as well as the best policy (MRL) considered, presenting response times close to the ideal distribution. Moreover, CRML presented a better performance than RR2 and MRL in the scenario where the ADNS had partial control. This result shows that the cooperation between the ADNS and the servers compensated the effect of the DNS caching on the calculation of the server load states, allowing a good load balancing.

7 CONCLUSIONS

In this paper, we have presented a new DNS-based load balancing policy, the CRML. This policy combines information from clients and servers to alleviate the negative effect of the DNS caching over the load balancing mechanism.

The experimental results showed that our load balancing policy worked as good as other DNS-based load balancing policies in the scenario where the ADNS had full control over name resolution requests. Furthermore, CRML outperformed RR2 and MRL in the scenario where the ADNS control was limited.

Future work includes: (i) the evaluation of the sensitivity of our policy to different combinations of DNS TTLs, time interval of server information propagation, and the threshold for identifying hot domains; and (ii) the evaluation of the combination of CRML with server redirection mechanisms and dynamic TTL policies.

ACKNOWLEDGEMENTS

The authors would like to thank Fapesp (n. 07/56423-6), CAPES, and CNPQ for the financial support, and Schooner, Emulab, PlanetLab, and RNP (National Education and Research Network) for the infrastructure support.

REFERENCES

Barroso, L. A., Dean, J., and Hölzle, U. (2003). Web search for a planet: The google cluster architecture. *IEEE Micro*, 23(2):22–28.

Chatterjee, D., Tari, Z., and Zomaya, A. Y. (2005). A task-based adaptive ttl approach for web server load balancing. In *Proceedings. 10th IEEE Symposium on Computers and Communications, 2005. ISCC 2005.*, pages 877–884. IEEE Computer Society.

Colajanni, M. and Yu, P. S. (2002). A performance study of robust load sharing strategies for distributed heterogeneous web server systems. *IEEE Transactions on Knowledge and Data Engineering*, 14(2):398–414.

Colajanni, M., Yu, P. S., and Dias, D. (1998). Analysis of task assignment policies in scalable distributed web-server systems. *IEEE Transactions on Parallel and Distributed Systems*, 9(6):585–600.

Moon, J.-B. and Kim, M. H. (2005). Dynamic load balancing method based on dns for distributed web systems. In *E-Commerce and Web Technologies. EC-Web*, volume 3590 of *Lecture Notes in Computer Science*, pages 238–247. Springer.

Nakai, A. M., Madeira, E., and Buzato, L. E. (2009). Lab4ws: A testbed for web services. In *Proceedings of the 2nd International Conference on Computer Science and its Applications (CSA'09)/2nd IEEE International Workshop on Internet and Distributed Computing Systems (IDCS'09)*, pages 647–652.

Pan, J., Hou, Y. T., and Li, B. (2003). An overview of dns-based server selections in content distribution networks. *Computer Networks*, 43(6):695–711.

Su, A.-J., Choffnes, D. R., Kuzmanovic, A., and Bustamante, F. E. (2006). Drafting behind akamai (travelocity-based detouring). *SIGCOMM Comput. Commun. Rev.*, 36(4):435–446.

TPC (2002). TPC Benchmark W - Specification 1.8. http://www.tpc.org/tpcw/spec/tpcw_V1.8.pdf.

Yokota, H., Kimura, S., and Ebihara, Y. (2004). A proposal of dns-based adaptive load balancing method for mirror server systems and its implementation. In *AINA '04: Proceedings of the 18th International Conference on Advanced Information Networking and Applications*, page 208, Washington, DC, USA. IEEE Computer Society.