RC22477 (W0206-013) June 5, 2002
Computer Science

IBM Research Report

Analysis of Design Alternatives for Reverse Proxy Cache Providers

Bruno Ciciani, Daniel M. Dias
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

F. Quaglia
DIS, University of Rome "La Sapienza"
Via Salaria 113, 00198 Rome Italy

Research Division
Almaden - Austin - Beijing - Delhi - Haifa - India - T. J. Watson - Tokyo - Zurich

Analysis of Design Alternatives for Reverse Proxy Cache Providers

B. Ciciani, D. Dias
IBM T.J. Watson Research Center
Yorktown Heights, NY 10598

F. Quaglia
DIS, University of Rome "La Sapienza"
Via Salaria 113, 00198 Rome Italy

Abstract

Reverse proxy caches are used to provide scalability and improved latency to popular sites on the Web. In this paper we provide analytical performance models for distributed reverse proxy cache architectures, and study the trade-offs between various design alternatives. Specifically, we consider static and dynamic assignment of proxy cache nodes to Web sites, with different levels of sharing of proxy caches among Web sites. Innovative analytical modeling contributions have been introduced to handle real design constraints, such as bounded cache size and bounded processing power, and of different characteristics related to the hosted objects, including reference rates, popularity distributions and update rates. In the analysis we have modeled both system steady state as well as transient interaction between Proxy sites and Web sites. We have found different trade-offs between various design alternatives depending on characteristics of the Web site workloads.

1 Introduction

With the growth of traffic to popular sites on the World Wide Web (Web), various Web caching techniques have been developed to improve the client response time, and to offload traffic from the heavily loaded Web sites. In one technique, so-called "reverse proxy (Web) caches" retain the hot pages from specific Web sites. These reverse proxy caches may be co-located with the Web site itself, be distributed but owned and hosted by the Web site owner/provider, or may be provided by third party reverse proxy caching services [11, 15, 16, 17, 18, 20, 25]. Various architectures have been used for distributed reverse proxy caches, in terms of the number and location of proxy cache sites, how the cache nodes are allocated or related to the Web sites supported, how cache misses are handled, among other design alternatives. In this paper we provide an analytical model for estimating the performance of distributed reverse proxy cache architectures, and study the trade-offs between various design alternatives.

Reverse proxy caches differ from forward proxy caches (typically referred to as proxy caches without qualification) in that the former cache Web objects from specific Web sites only, while the latter cache objects from all Web sites. Typically, reverse proxy caches are associated with,

and payed for, by the end Web sites. On the other hand forward Web proxy caches are typically owned by enterprises for caching Web requests from browsers within the enterprise, or by Internet Service Providers (ISPs) to cache requests from their client Web browsers. In this paper we focus on reverse proxy caches exclusively. Initially, reverse proxy Web caches were co-located at the Web sites, in order to reduce the load on the Web servers, and to improve the throughput [5, 6]. In order to provide better scaling, high availability and lower latency to clients, the Web caches were distributed and often co-located at network access points (NAPs), such as for Sports and Events Web sites [4, 13]. Finally shared reverse proxy caching services were provided to cache objects from multiple Web sites [1, 8]. The basic organization of the reverse proxy caches is one in which there are a number of geographically distributed locations at which a cluster of Web cache nodes are located. The locations are typically either co-located with ISPs (e.g. [1]), or located at NAPs (e.g. [8]), in order to reduce the latency to clients. The number of locations varies from the tens, typically for the case with the caches at the NAPs, to hundreds, typically for the case with caches co-located at ISPs. As study of the measured performance of reverse proxy caching services using each of these architectures can be found in [12]. In this paper, we focus on the former case, where there are on the order of tens of sites.

There are various alternatives for distributed reverse proxy cache architectures; this is discussed in further detail in Section 2, and summarized here. In the simplest case, each of the geographically distributed site independently caches objects from all of the Web sites, and specific nodes at each site are statically assigned to cache objects from specific sites. The static assignment could be random, or based on measured load for each Web site. Multiple nodes could cache objects from the same Web site for scaling, and the assignment could be static or dynamic. Similarly, there could be sharing across geographically distributed sites to minimize the miss ratio to the Web sites (actually this solution is not analyzed in this paper, it will be object of a following contribution). There are trade-offs between each of these alternatives, which are examined in detail in this paper. For example, static assignment of a small number Web sites per node can leads to higher hit ratios in RAM because fewer sites share the same cache, but can lead to significantly worse performance during traffic surges. These and other trade-offs are the subject of this paper. We analyze the behavior of the different architectures with variation of the popularity of the objects accessed, object request rates, object update probability, available RAM size in each cache node, available processing capability in each cache node, among other parameters.

The organization of the paper is as follows. Section 2 describes the reverse proxy cache architectures considered and provides a qualitative comparison. Section 3 describes the model for the various architectures and design alternatives. Quantitative comparisons from the model appear in Section 4. Finally, Section 5 has concluding remarks, and a summary of the contributions of the paper.

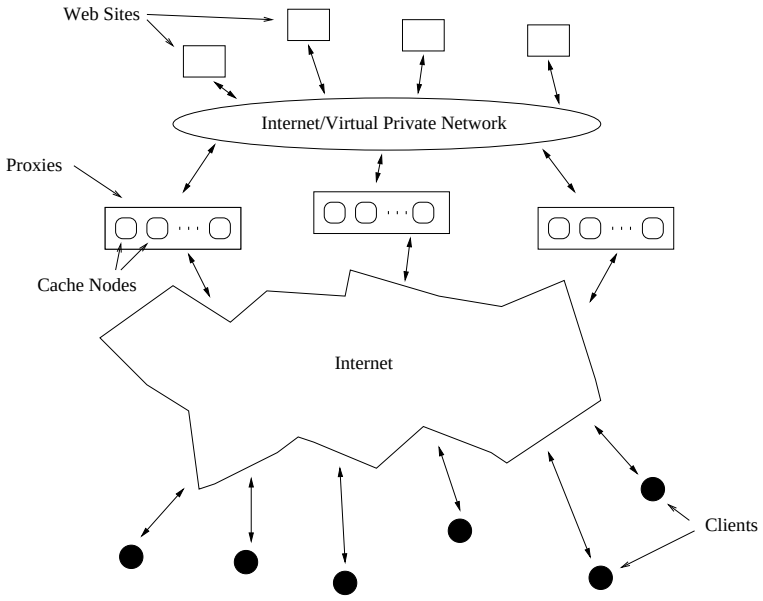

Figure 1: Target System.

2 Reverse Proxy Cache Architectures

As shown in Figure 1 the target system consist of Web sites and Proxy sites, connected through the Internet or a (virtual) private network. Each Proxy site, in turn, has a set of cache nodes and a load balancer that directs object requests to cache nodes according to the IP address of the associated Web site. Each cache node has a two-level storage system (RAM/disk), and is directly connected both to the Internet, and, if necessary, to the virtual private network connecting the cache nodes to the Web server nodes.

A cache node can be either assigned to a unique Web site (exclusive assignment) or be shared among multiple (as an extreme all) Web sites (shared assignment). With exclusive assignment, the cache node maintains copies of objects from a single Web site. Instead, with shared assignment, it maintains copies of objects from multiple Web Sites.

The assignment of a cache node to a Web site can be either static or dynamic, depending on workload conditions. Similarly, in case of shared assignment, the portion of the cache node RAM destined for objects of a specific Web site is established either statically or dynamically. In this section, we qualitatively discuss the benefits and drawbacks of the different options.

Exclusive vs Shared Cache Node Assignment. Sharing of cache nodes among multiple Web sites allows balancing the load of object requests, so that surges in traffic for requests to one or a few of the Web sites can be supported. Further, having multiple cache nodes handle requests for the same object can be expected to reduce the response time for the "hottest" objects, because the load on these hot objects can be handled by multiple nodes. On the other hand, sharing of cache nodes implies assigning only a portion of the cache node RAM to each Web site, with increase in the likelihood that the assigned portion can not maintain copies of

all the cacheable objects of that Web site. In this case, disk access might be required, with consequent increase in the response time, and reduction in the throughput relative to hits in RAM.

Static vs Dynamic RAM Partitioning. Cache node sharing among multiple Web sites requires partitioning of the cache node RAM to assign a specific portion to each Web site. Static RAM partitioning allows controlling the RAM hit ratio for each Web site on the basis of the object relative popularities and of the amount of objects that can be maintained in the specific RAM portion. On the other hand, dynamic RAM partitioning (i.e. with a variable RAM portion assigned to each Web site depending on the object access pattern), allows maximizing the global RAM hit ratio in case classical LFU (or LRU) is adopted as the object replacement policy.

Static vs Dynamic Cache Node Assignment. Static assignment of a cache node to a Web site is easy to handle but does not allow support of time-varying workloads or surges in traffic across the Web sites, since the statically assigned nodes may become overloaded. Dynamic assignment of additional nodes can handle a surge in traffic to one or a few Web sites. However, it gives rise to cache node "warm-up" periods, leading to high loads at the Web sites during the warm-up period. Specifically, upon a new cache node assignment to a Web site, the miss ratio for requests related to objects of that Web site might be extremely high since the new cache node maintains no cached object related that Web site.

3 Analytical Model

To ease the presentation we assume that the traffic of HTTP requests related to the k-th Web site, namely WS_k, is equally distributed among all Proxy sites. Generalizing the analysis to the case of non-uniform traffic distribution is straightforward. Specifically, it only requires solving the model we propose using a set of distinct values, one for each Proxy site, for the request traffic related to WS_k.

We denote as λ_k the the arrival rate of HTTP requests related to WS_k, and we suppose requests arrive according to a Poisson process. Although a log-normal distribution is typically considered as a better model for the arrival rate [3], the exponential assumption is reasonable since the request distribution can be approximated as a Markovian arrival rate [21] or as a Markovian Modulated-Poisson process [21, 22], and each state if this stochastic process is actually characterized by request arrival distributed according to a Poisson process. In addition, it has been shown that the exponential distribution better matches real workloads, as compared to the log-normal distribution, during the busiest periods [9, 19]. We assume that object replacement within the RAM of any cache-node/Web-site is made according to the LFU policy. Also, we assume that cache node disk has unbounded capacity, but limited throughput.

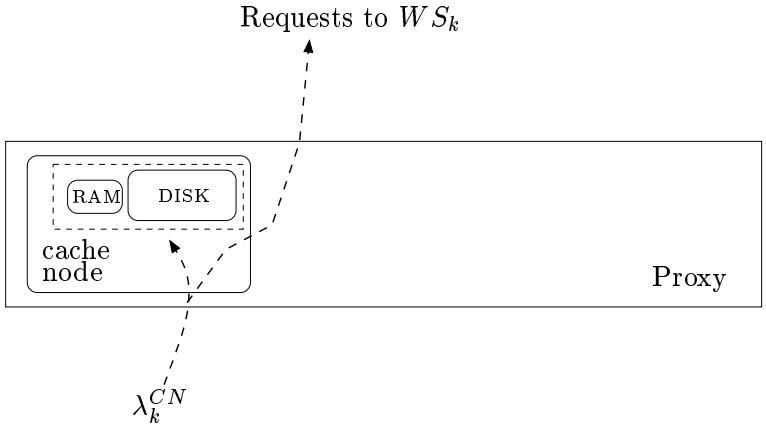

Figure 2: Split of Requests Between Cache Node and Web Site.

3.1 Evaluation of the Cache Node Hit/Miss Ratio

In this section we evaluate the cache node hit ratio at the level of both the cache node RAM and the cache node disk. An innovative contribution in our analysis consists of extending existing analytical results in the context of the evaluation of steady-state properties of Web caching systems with unbounded cache size [26], in order to include the effects of capacity misses. Actually there already exists a result that takes into account capacity misses [3], however it applies only to the case of non-updateable objects. We will consider updateable objects in the analysis.

To ease the presentation we assume that all cacheable objects of WS_k have the same size, we note however that extending the analysis to the case of different sizes is relatively straightforward. We denote as n_k the total number of cacheable objects associated with WS_k, and with C_k the cache node RAM capacity associated with cacheable objects of that same Web site. The relative popularity of cacheable objects of WS_k follow a Zipf-like distribution with parameter α_k [3]. We denote as $p_{k,j}$ the relative popularity associated with the j-th object of WS_k. According to the Zipf-like distribution, the value of $p_{k,j}$ decreases vs j. Also, the j-th cacheable object of WS_k has update rate equal to $\mu_{k,j}$, with exponential distribution of the length of the time interval between updates. Finally, λ_k^{CN} denotes the request arrival rate, associated with WS_k, seen by any single cache node of the Proxy site.

As show in Figure 2, each request in the flow λ_k^{CN} is either served through an object cached within the cache node RAM/disk, or is forwarded to WS_k. Given that the cache node disk capacity is assumed as unbounded, at steady state all the n_k (cacheable) objects associated with WS_k are actually maintained into the cache node disk (some of them are also maintained into the cache node RAM). Therefore, (cacheable) object misses within the cache node RAM/disk can occur only in case of object staleness. As a consequence, the miss ratio MR_k within the cache node RAM/disk for requests associated with WS_k can be computed as

$$MR_k = \sum_{i=1}^{n_k} p_{k,i} \frac{\mu_{k,i}}{\lambda_k^{CN} p_{k,i} + \mu_{k,i}} \tag{1}$$

$1 - MR_k$ is the fraction of the requests that are served through non-stale objects maintained within the RAM/disk of each cache node. These requests are split, in turn, depending on whether the corresponding object is found within the RAM or not. Given the LFU replacement policy, the cache node RAM maintains the most requested objects of WS_k (i.e. the most popular), therefore the cache node RAM hit ratio RHR_k related to requests associated with objects of WS_k can be expressed as

$$RHR_k = (1 - MR_k) \sum_{i=1}^{min(C_k, n_k)} p_{k,i} \tag{2}$$

The min operator for the upper limit in the previous sum captures the fact that, in case $C_k > n_k$, all the cacheable objects of WS_k are actually maintained into the cache node RAM. Trivially, the cache node disk hit ratio DHR_k related to requests associated with objects of WS_k can be expressed as

$$DHR_k = (1 - MR_k)(1 - RHR_k) = (1 - MR_k) \sum_{i=min(C_k, n_k)+1}^{n_k} p_{k,i} \tag{3}$$

In some sense the modeled scenario can be see as a two-level hierarchical caching system in which the second level, i.e. the disk, has unbounded capacity, while the first level, i.e. the RAM, has limited capacity. The disk maintains replicas of objects maintained in the RAM, and object staleness into the RAM implies object staleness into the disk. Therefore only those requests associated with non stale objects and with RAM capacity misses go to the disk.

Note that our analysis differs from those already proposed for two-level caching systems [10, 22]. Specifically, those analyses are based on the assumption of unbounded cache size at both the first and the second level, therefore, miss at the first level can be due only to object staleness. Also, the caching systems considered are different. Specifically, a second level cache manages object misses from multiple first level caches, instead, in the system we consider, a second level cache, namely the cache node disk, serves capacity misses of a single first level cache, namely the cache node RAM.

As a last point, the value of both C_k and λ_k^{CN} depend on the selected Reverse Proxy Cache organization among those discussed in Section 2 (for example they depend on the amount of Web sites assigned to a specific cache node and also on how the cache node RAM is partitioned among these Web sites). We shall report a complete analysis of each organization in the following section.

3.2 Exclusive Cache Node Assignment

In the exclusive cache node assignment organization, each cache node serves requests for a single Web site. Indicating with NP the total number of Proxies, with NCN_k the number of cache nodes within a Proxy that are assigned to WS_k and with C^{tot} the total cache node RAM capacity, in terms on number of objects, we get

$$C_k = C^{tot} \tag{4}$$

and

$$\lambda_k^{CN} = \frac{1}{NCN_k}\frac{\lambda_k}{NP} \tag{5}$$

Expression 5 simply states that, to get the request traffic λ_k^{CN}, we have to divide λ_k for the total number of Proxy sites NP due to the homogeneity assumption of load distribution among the Proxy sites, and then we have to split the obtained traffic value among the number of cache nodes NCN_k assigned to WS_k.

Denoting with:

- $E[ram\_hit]$ the expected CPU time for serving a request (supposing the object is already in the cache node RAM);
- $E[disk\_request]$ the expected CPU time for a disk/RAM object transfer request at the cache node;
- $E[http]$ the cache node CPU time for an HTTP session to download an object from the Web site;
- $E[disk]$ the expected time for handling an object transfer from/to the cache node disk;

we get the following expressions for the utilization factors of the cache node CPU and the cache node disk

$$\rho_{CPU} = \lambda_k^{CN}(E[ram\_hit] + DHR_k E[disk\_request] + MR_k E[http]) \tag{6}$$

$$\rho_{disk} = \lambda_k^{CN}(DHR_k + MR_k)E[disk] \tag{7}$$

Note that in expression (7) the multiplier factor for $E[disk]$ contains both DHR_k and MR_k since cache node disk access occurs in case of RAM capacity miss and also in case of staleness miss, with consequent download of the object from WS_k.

Denoting with:

- C_{WS}^k the RAM capacity (in terms of objects) of WS_k;
- $E[WS\_http]$ the CPU time at WS_k for an HTTP session to upload an object to a Proxy;
- $E[WS\_disk\_request]$ the expected CPU time for a disk/RAM object transfer request at WS_k;
- $E[WS\_disk]$ the expected time for handling a disk/RAM object transfer at WS_k;

we get the following expressions for the utilization factors of the CPU and the disk of WS_k

$$\rho_{WS\_CPU} = \lambda_k MR_k(E[WS\_http] + \sum_{i=min(C_{WS}^k, n_k)+1}^{n_k} p_{k,i} E[WS\_disk\_request]) \tag{8}$$

$$\rho_{WS\_disk} = \lambda_k MR_k \sum_{i=min(C^k_{WS},n_k)+1}^{n_k} p_{k,i} E[disk] \quad (9)$$

where the term $\sum_{min(C^k_{WS},n_k)+1}^{n_k} p_{k,i}$ in both previous expressions indicates the probability that a requested object is not in the RAM of WS_k due to a capacity miss (recall that due to the LFU replacement policy, only the most popular objects are maintained into the Web site RAM).

Modeling as usual the cache-node/Web-Site CPU with an M/G/1/PS queue [9] (this matches our assumption of exponential distribution for the HTTP request arrival process), and the cache node disk with an M/M/1 queue [14], and denoting with Δ the delay for object transfer between the Web site and a Proxy site ([1]), we can express the expected latency time T of a request, evaluated at the cache node level, as ([2])

$$\begin{aligned} T &= \frac{E[ram\_hit]}{1-\rho_{CPU}} + DHR_k\left(\frac{E[disk\_request]}{1-\rho_{CPU}} + \frac{E[disk]}{1-\rho_{disk}}\right) + \\ &\quad + MR_k\left(\frac{E[WS\_http]}{1-\rho_{WS\_CPU}} + \sum_{i=min(C^k_{WS},n_k)+1}^{n_k} p_{k,i}\frac{E[WS\_disk]}{1-\rho_{WS\_disk}} + \Delta\right) \end{aligned} \quad (10)$$

Expression (10) can be evaluated by computing MR_k, RHR_k and DHR_k (as expressed by (1), (2) and (3)) on the basis of the constraints in expressions (4) and (5), and then computing the utilization factors expressed in (6)-(9).

3.3 Shared Cache Node Assignment with Static RAM Partitioning

In the shared cache node assignment organization with cache node RAM statically partitioned, each cache node serves requests for multiple Web sites. Also, the cache node RAM is split into equal portions, each one assigned to objects of a specific Web site. Exploiting notation already introduced in Section 3.2, and indicating with N the total number of Web sites hosted by the cache node and, again, with NCN_k the total number of cache nodes within a Proxy assigned to WS_k, we get

$$C_k = \frac{C^{tot}}{N} \quad (11)$$

and

$$\lambda_k^{CN} = \frac{1}{NCN_k}\frac{\lambda_k}{NP} \quad (12)$$

[1] As pointed out in Section 2, the presence of a (virtual) private network between Web sites and Proxies allows approximating the object upload latency to a Proxy with a constant value.

[2] Considering for the CPU model an M/G/1/PS queue with non-minimal amount of concurrency in the PS discipline, i.e. a relatively large maximum amount of requests that can be handled concurrently, the CPU response time can be approximated with the formula related to the M/M/1 queue.

Similarly to expression (5) related to the case of exclusive cache node assignment, λ_k^{CN} is computed by dividing λ_k for the total number of cache nodes assigned to WS_k among all the Proxies. This is due to homogeneous split of the workload among the proxies and also among all the cache nodes assigned to WS_k within each Proxy site.

Using the same notation as in Section 3.2 for expected CPU times and disk access cost at the cache node, we get the following expressions for the cache node CPU and the cache node disk utilization factors

$$\rho_{CPU} = \sum_{k=1}^{N} \lambda_k^{CN}(E[ram\_hit] + DHR_k E[disk\_request] + MR_k E[http]) \tag{13}$$

$$\rho_{disk} = \sum_{k=1}^{N} \lambda_k^{CN}(DHR_k + MR_k)E[disk] \tag{14}$$

The previous expressions point out that, in case of shared node assignment, the cache node CPU and disk utilization factors are computed by considering request traffic for multiple Web sites, weighted by the RAM/disk hit/miss ratios.

The expressions for the utilization factors of the CPU and the disk of WS_k remain identical to those in (8)-(9). The same is true for the latency time as in expression (10). Therefore, solving this model requires the same steps as those listed at the end of Section 3.2, with the difference that the constraints to be used in order to compute MR_k, RHR_k and DHR_k are those in expressions (11) and (12).

3.4 Shared Cache Node Assignment with Dynamic RAM Partitioning

In the shared cache node assignment organization with cache node RAM dynamically partitioned, each cache node serves requests for multiple Web sites, however the cache node RAM capacity is not split into equal portions among the Web sites. Specifically, the objects maintained in the cache node RAM (and therefore the amount of RAM capacity assigned to each Web site) are determined dynamically on the basis of the LFU policy considering the spectrum of access frequencies, related to the whole set of cacheable objects of the Web sites, seen by any single cache node.

In other words, we can construct an ordering among those objects based on their access frequencies seen by the cache node (we recall that in this configuration the access frequency at each cache node for the j-th object of WS_k is computed as $\frac{\lambda_k}{NCN_k \ NP} p_{k,j}$) and we can associate with the j-th object of WS_k an index , namely $I_{k,j}$, indicating the position of that object in the ordering. The j-th object of WS_k is maintained into the cache node RAM if $I_{k,j} \leq C^{tot}$, therefore the amount of cache node RAM capacity C_k assigned to WS_k can be computed as:

$$C_k = \sum_{\forall I_{k,j} \leq C^{tot}} 1 \tag{15}$$

For estimating the value of λ_k^{CN}, the utilization factors and the latency time, we get the same expressions as the ones related to the case of static RAM partitioning in Section 3.3. Therefore

solving the model for dynamic RAM partitioning requires the same steps as those for static RAM partitioning, with the only exception that the constraint in expression (11) must be replaced with the constraint in expression (15).

3.5 Transient Behavior

As already discussed in Section 2, independently of the partitioning policy adopted for the cache node RAM, the two organizations based on shared cache node assignment are characterized by better load balance among the cache nodes within each Proxy, with consequent better balanced utilization of the CPUs and disks among all the cache nodes. This is not the case for the exclusive cache node assignment organization, where strongly unbalanced utilization of distinct cache nodes might arise in case distinct Web sites are associated with very different request rates.

As pointed out, to improve the caching system performance, especially in case of exclusive cache node assignment, a cache node might be dynamically switched between Web sites. Specifically, it might be de-assigned from a lightly loaded Web site and re-assigned to a Web site whose load tends to become heavy. However, upon the assignment of a cache node to WS_k, the cache node itself maintains no cached object related to WS_k. As a consequence, requests directed to that cache node must be forwarded to WS_k in order to download (for caching) the corresponding objects. In other words, we might get a traffic peak on WS_k in the interval between the instant of the cache node assignment and the instant in which the cache node reaches a steady state for what concerns cached objects of WS_k.

We now evaluate the peak traffic on WS_k due to requests occurring during the cache node warm-up period. We denote as $X_{k,j}(M)$ the conditional probability that no request for the j-th object of WS_k occurs at the newly assigned cache node, given that M requests related to objects of WS_k have been issued to that cache node since the new assignment. This quantity can be evaluated as

$$X_{k,j}(M) = (1 - p_{k,j})^M \tag{16}$$

Therefore, the cache node miss ratio (related to WS_k) due to warm-up at the $M+1$-th request arrival, namely $MRWU_k(M+1)$, can be evaluated as

$$MRWU_k = \sum_{i=1}^{n_k} p_{k,i} X_{k,i}(M) = \sum_{i=1}^{n_k} p_{k,i}(1 - p_{k,i})^M \tag{17}$$

Actually, to derive expression (17) we have implicitly assumed that cache node misses due to object staleness have a negligible impact during the cache node warm-up period. This assumption is likely to be true in practice since dynamic assignment of a cache node to WS_k takes place in case of high request arrival rate that is likely to produce a very short warm-up period, during which few objects are likely to be updated at the Web site.

The frequency of requests associated with WS_k and directed to the newly assigned cache node is λ_k^{CN} as expressed by (5), with the parameter NCN_k taking into account the newly assigned

cache node. Therefore, M can be expressed as a function of the time interval δt since the cache node assignment as

$$M = \lambda_k^{CN} \delta t \tag{18}$$

We can now evaluate the request traffic to WS_k (due to cache miss at the newly assigned cache node) at any instant of the warm-up period, which we denote as λ_k^{WU}, as

$$\lambda_k^{WU} = \lambda_k^{CN} MRWU_k = \lambda_k^{CN} \sum_{i=1}^{n_k} p_{k,i}(1 - p_{k,i})^{\lambda_k^{CN} \delta t} \tag{19}$$

4 Quantitative Comparison

By the previous analysis we argue that the performance of the different architectural alternatives depends mainly on the cache node RAM miss ratio and on the workload assigned to each cache node. Therefore, given a number of Web sites to serve, performance optimizations can be achieved by keeping the RAM miss ratio low, while simultaneously avoiding load bottlenecks. Note that the RAM miss ratio depends on the RAM capacity and on object relative request rates, which, in turn, depend on the request arrival rate for each Web site and on the distribution of the object popularity; meanwhile, the workload on each each cache node depends on the number of Web sites assigned to that cache node and on their request arrival rate. In this section, we shall compare architectural alternatives that analyze the performance behavior of design alternatives that attempt to keep the RAM miss ratio low and/or the cache node utilization within bounds.

To keep low the RAM miss ratio, we should try to have most of the (very) popular objects of each Web site into the RAM. This could be achieved by assigning few Web sites to each cache node in order to allow a reasonable size RAM partition to be assigned to each site. On the other hand, avoidance of load bottlenecks can be obtained by allowing all the Web sites to share all the cache nodes. However, this type of sharing does not favor RAM hit given that a reduced percentage of cacheable objects of each Web site can be maintained in the cache node RAM. In other words, there is a clear tradeoff between advantages due to RAM hit and those due to balanced request load distribution. Such a tradeoff can be optimized through intermediate architectural configurations where a group of Web site share a group of cache nodes. These configurations, as well as extreme configurations, will be the object of this quantitative study.

We consider a reverse proxy cache architecture consisting of 10 Proxy sites and 10 cache nodes per Proxy site. The cache node RAM has capacity of 1 GB that, assuming 8 KB as the average size of cacheable objects [6], allows maintaining about 130000 cacheable objects. Other system parameters, chosen on the basis of estimates reported in [23], related to standard hardware/software systems, are listed in Table 1.

We will study the performance provided by different configurations of the Proxy architecture while varying the number of Web sites. Specifically, we will consider the case of 50 (CASE A) and 100 (CASE B) Web sites hosted by the Reverse Proxy Cache architecture. In both cases, each Web

Table 1: System Parameters.

$E[ram\_hit]$	0.5 msec.
$E[disk\_request]$	0.05 msec.
$E[http]$	1 msec.
$E[disk]$	10 msec.
$E[WS\_http]$	1 msec.
$E[WS\_disk\_request]$	0.05 msec.
$E[WS\_disk]$	10 msec.
Δ	100 msec.

site maintains 15000 cacheable objects and has a RAM able to maintain all the 15000 cacheable objects. Additional Web site related system parameters, always selected on the basis of the results in [23], are listed in Table 1.

For the parameter α characterizing the Zipf-like distribution for the object popularity, several values have been identified in the literature. For example, we have an estimated α of 1.37 for the 1998 World Cup Web site [2], 0.77 from DEC traces, 0.78 from University of Pisa traces, 0.83 from FuNet traces, 0.69 from UCB traces, 0.73 from Questnet traces and 0.64 from NLAR traces [3]. In our study we will assume different values of α ranging between 0.6 and 1.4 so as to cover an interval containing all the values identified above.

Cacheable objects of each Web site are considered to have update rates ranging between 1/15 min. and zero (passing through 1/30 min., 1/1 hour, 1/12 hours and 1/24 hours), with update rate decreasing with decrease in the object popularity.

Finally, according to [7], we assume that the 20% of the requests directed to a specific Web site are related to non-cacheable objects, that need to be obtained from the Web site. These requests produce on the CPU same overhead as that one for dowloading a cacheable object. Instead they do not produce disk overhead since non cacheable objects are not retained by the cache node memory system.

4.1 CASE A

As mentioned above, CASE A refers to 50 Web sites. We consider for this case 3 different configurations. In each configuration, two cache nodes of each Proxy site host 10 Web sites, namely $WS_0, \ldots, WS_9$, each with different request arrival rates. Specifically, the total request arrival rate that we denote as $10 \times \bar{\lambda}$, is distributed among the 10 Web sites according to the distribution in Table 2. Specifically, there are six Web sites with light relative load, three Web sites with slightly higher relative load and one Web site with significantly higher relative load.

The 3 configurations are as follows:

Configuration 1. $WS_0 - WS_4$ are assigned to the first node of the couple of cache nodes, while $WS_5 - WS_9$ are assigned to the second one. In other words, each cache node hosts the same number of Web sites. Given that the entire set of cacheable objects of each Web site requires

Table 2: Load Distribution Among the 10 Web Sites.

WS_0	WS_1	WS_2	WS_3	WS_4	WS_5	WS_6	WS_7	WS_8	WS_9
1/24	1/24	1/24	2/24	2/24	1/24	1/24	1/24	2/24	12/24

120 MB, this configuration allows all the objects of the 5 Web sites hosted by each node to be maintained into the cache node RAM, thus favoring RAM hit (i.e. RAM misses can be due only to object staleness). On the other hand, the request load is unbalanced among the cache nodes.

Configuration 2. $WS_0 - WS_8$ are assigned to the first node of the couple of cache nodes, while WS_9 is assigned to the second one (exclusive cache node assignment). In this case we get balanced load (each node handle 12/24 of the whole request traffic), but a reduced amount of cacheable object per Web site is retained into the RAM of the first cache node, thus not favoring RAM hit on this cache node. On the other hand, we favor RAM hit on the second cache node for the heavily loaded Web site, namely WS_9.

Configuration 3. All the ten Web sites $WS_0 - WS_9$ are assigned to both the cache nodes. In this case we get again balanced load, at the expense of RAM hit ratio on both the cache nodes due to the larger number of Web sites hosted by each cache node.

For all the three configurations we consider the case of both dynamic and static RAM partitioning among the Web sites hosted by the same cache node. In the case of static partitioning, equal portions of the RAM capacity are assigned to the Web site hosted by a cache node.

The results are reported in Figures 3-5 (the "average request arrival rate per WS" on the x axis represents $\bar{\lambda}$). Each reported value is the worst case among the pair of cache nodes.

From the plots, the CPU is the bottleneck for Configuration 1, while the disk is often the bottleneck for the other two configurations, especially for small values of α. For configuration 1, the CPU is the bottleneck because of load imbalance between the pair of nodes, while in the other configurations, better load balance is achieved. The throughput at which the disk saturation point occurs in Configurations 2 and 3 increases with increasing values of α . This is because, in the Zipf-like distribution, increasing α means higher skew of the object access pattern towards the most popular objects, leading to higher hits in the RAM for higher values of α.

The latency at the proxy server is always under 10 milliseconds, except when system (CPU/disk) saturation occurs. Also, the latency curves exhibit a minimum for intermediate values of the average request arrival rate. This phenomenon is due to the effect of object updates. Specifically, for low throughput, there is a higher likelihood of requests for stale objects than that for intermediate workloads.

With respect to Configuration 1 (see Figure 3), we recall that cache node disk load is due exclusively to downloads of updated objects upon a staleness miss; this is because the cacheable

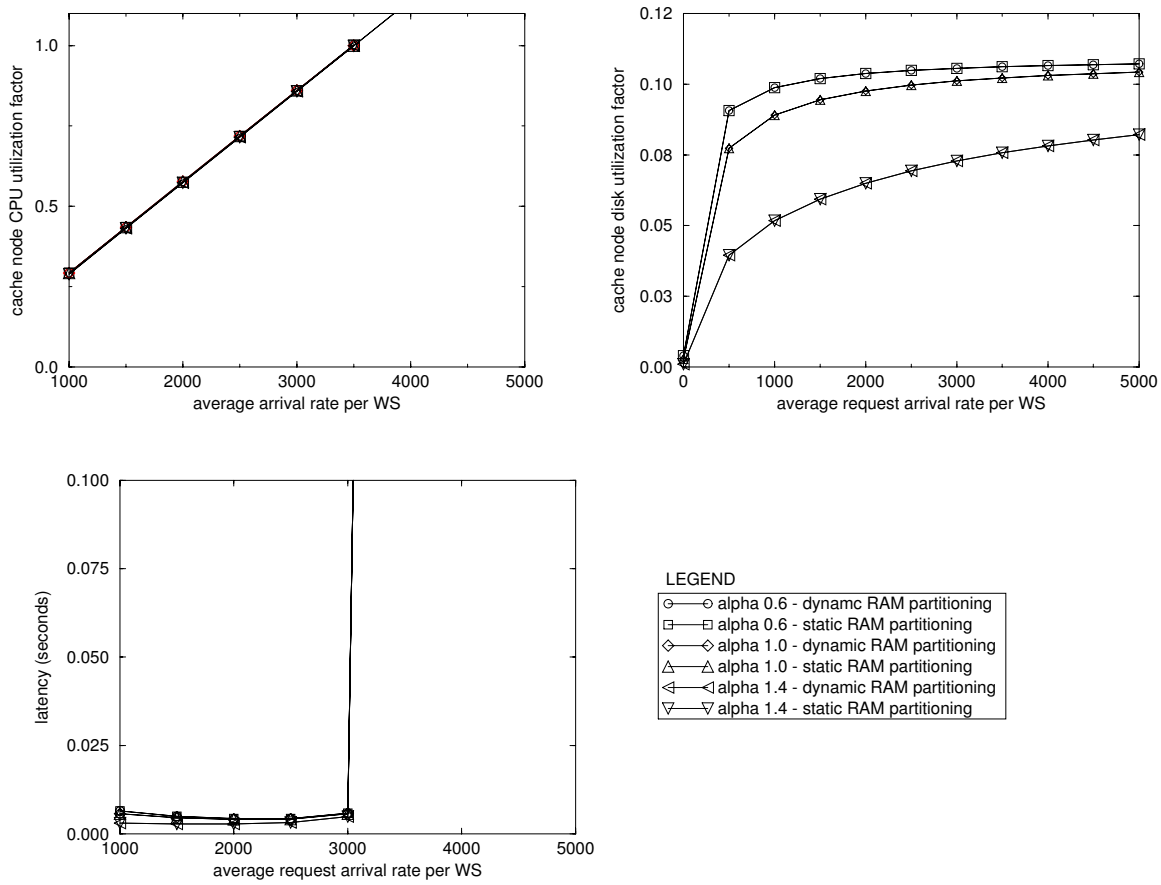

Figure 3: Configuration 1.

objects of the Web sites assigned to the cache nodes can all be maintained in the cache node RAM. From the plots we get the disk utilization factor increases quickly and then tends to be stable. This is because, at low throughput, requests are often to updated documents, which also need to be written to disk; at higher rates, most requests of cacheable objects are hits in the RAM cache, and the disk access rate stabilizes.

Overall, as expected, Configuration 1, keeps low the RAM miss ratio, thus suffering from no overload on the cache node disk, at the expense of earlier CPU saturation due to unbalanced load among the cache nodes. Configuration 2, characterized by better load balance among the cache nodes, suffers from disk overload especially for low values of the the Zpif-like distribution parameter α (i.e. in case of limited skew in the access pattern to the cacheable objects of a Web site). Such a phenomenon is less evident for Configuration 3, due to the balanced split of the requests for objects of the same Web site among the two cache nodes.

As a last point, one drawback of Configuration 2 is the need for cache node re-assignment when significant changes in the request arrival rate for one or more Web sites occur. To provide insight into this issue, we report in Figure 6 plots related to the workload experienced by a Web site during

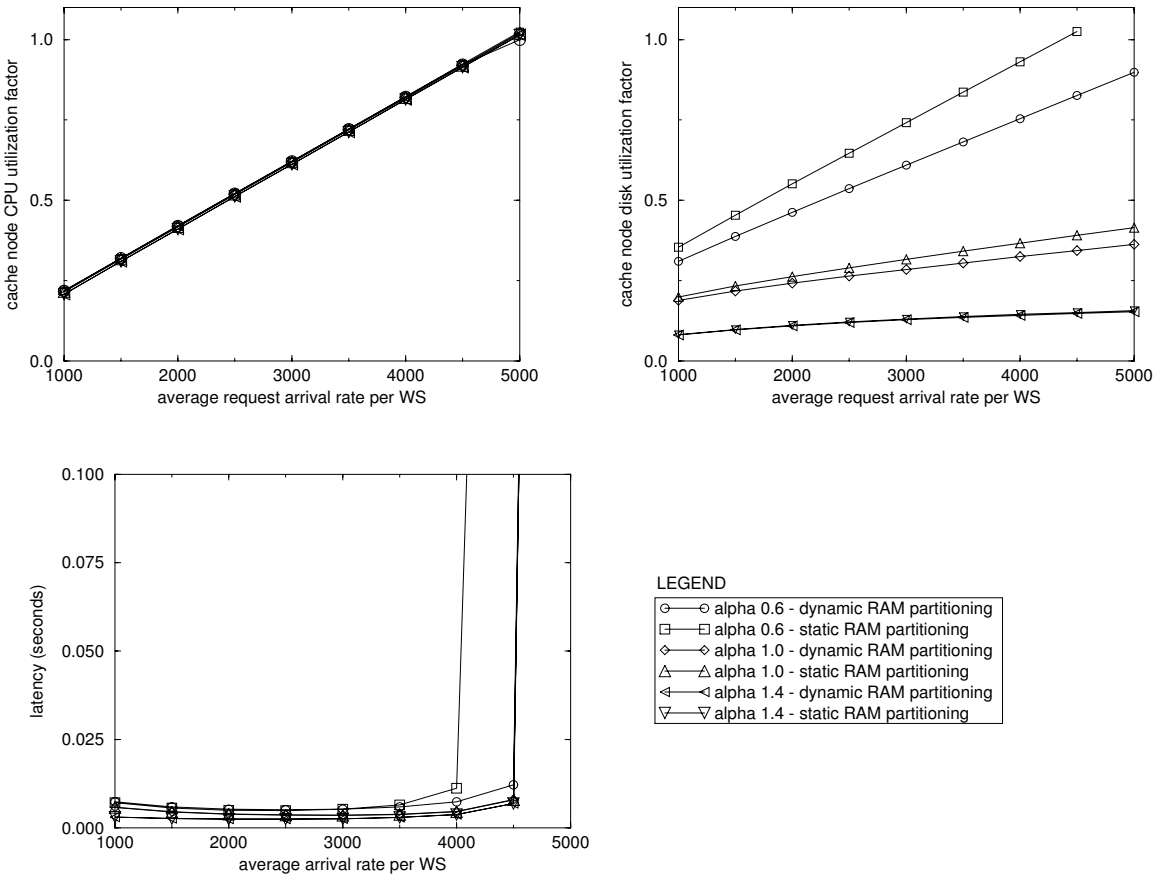

Figure 4: Configuration 2.

the cache node warm-up period for three different values of the Zipf-like distribution parameter α. These plots have been obtained for the case of access rate to the Web site of 10000 requests per second considering that one of the two cache nodes in the pair is dynamically reassigned to that Web site. These plots indicate that surges in traffic to the home Web site occur during the warm-up period of the newly introduced cache, and these surges persist for a significant period. Often, the home Web site is not configured to handle such a surge, and the length of the warm up period would be longer, with disruption in the service. Thus, configuration 2 is unacceptable, without a solution to the cache warm-up problem. Possible solutions include pushing the cache content from the existing data on the warm cache node, having the new cache fetch data on demand from the first cache, or adding a hierarchy of caches.

4.2 CASE B

CASE B refers to 100 Web sites. For this case we also consider three different configurations, similar in nature to those previously discussed. The difference is that, this time, each pair of cache

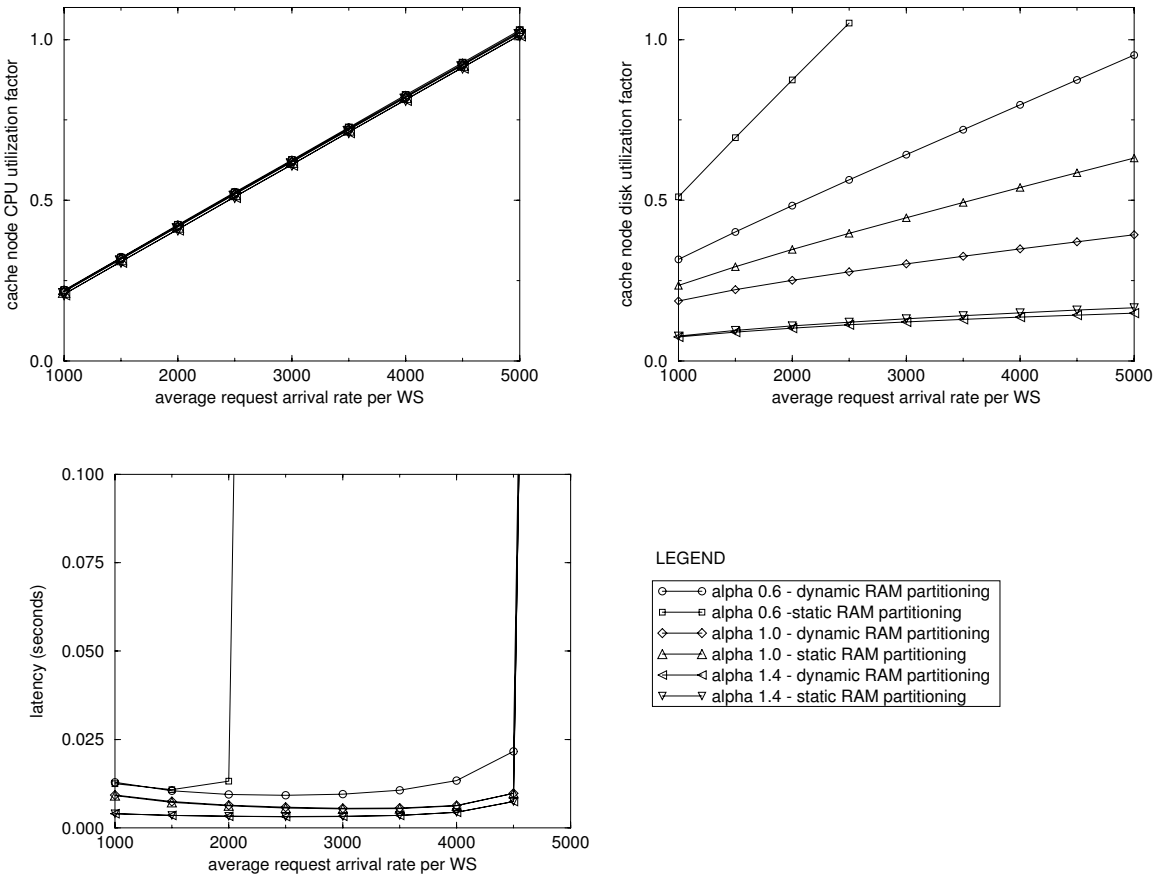

Figure 5: Configuration 3.

nodes of each Proxy site host 20 Web sites, $(WS_0, \ldots, WS_{19})$, with different relative request arrival rates according to that shown in Table 3, and the three configurations are as follows:

Configuration 1. $WS_0 - WS_4$ and $WS_{10} - WS_{14}$ are assigned to the first node of the couple of cache nodes, while $WS_5 - WS_9$ and $WS_{15} - WS_{19}$ are assigned to the second one. Here we again favor RAM hit against load balancing.

Configuration 2. $WS_0 - WS_8$ and $WS_{10} - WS_{18}$ are assigned to the first node of the couple of cache nodes, while WS_9 and WS_{19} are assigned to the second one (exclusive cache node assignment). Here again we get balanced load with reduced RAM hit on the first of the two cache nodes.

Configuration 3. All the twenty Web sites $WS_0 - WS_{19}$ are assigned to both the cache nodes. Here again we get balanced load, at the expense of RAM hit ratio on both the two cache nodes.

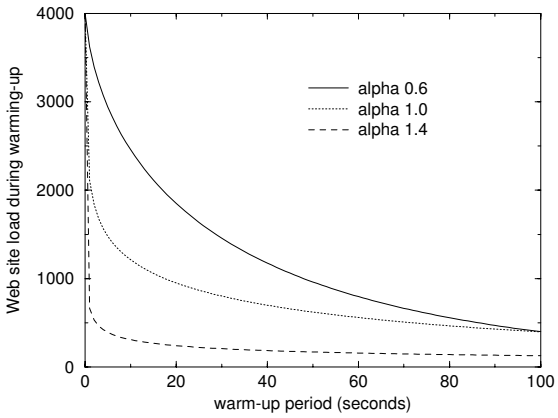

Figure 6: Web Site Load (Requests per Second Due to Cache Node Miss) vs Warm-up Period.

Table 3: Load Distribution Among the 20 Web Sites.

WS_0	WS_1	WS_2	WS_3	WS_4	WS_5	WS_6	WS_7	WS_8	WS_9
1/48	1/48	1/48	2/48	2/48	1/48	1/48	1/48	2/48	12/48
WS_{10}	WS_{11}	WS_{12}	WS_{13}	WS_{14}	WS_{15}	WS_{16}	WS_{17}	WS_{18}	WS_{19}
1/48	1/48	1/48	2/48	2/48	1/48	1/48	1/48	2/48	12/48

This time we have fixed the parameter α at 1.0. From the plots in Figure 7 we observe that, even though Configuration 1 still exhibits the higher CPU utilization factor, it provides better latency as compared to the other configurations. This is because, when each cache node serves more than 10 Web sites (which occurs for both configurations 2 and 3), disk overload becomes a more critical factor. However, additional disks could be added per node to mitigate this effect.

Actually, Configuration 3 performs worse than Configuration 2 for dynamic RAM partitioning. This is because in Configuration 3 two Web sites with request rate values roughly an order of magnitude higher than the other Web sites are assigned to both the cache nodes in the pair. As a consequence, most of the objects from both these Web sites are retained into the cache node RAM at the expense of reducing the amount of retained objects from the other Web sites. Instead, in the case of static RAM partitioning, Configuration 3 performs a little better than Configuration 2 since, even if the requests for the two most accessed Web Sites generate a large load, they do not push out the cache data for the other sites sharing that node.

5 Conclusions

In this paper we analyzed trade-offs between different design options for reverse proxy caches, and provided an analytical model for quantitatively comparing these designs. Specifically, we examined designs where reverse proxy caches are shared to support several Web sites. The alternatives include

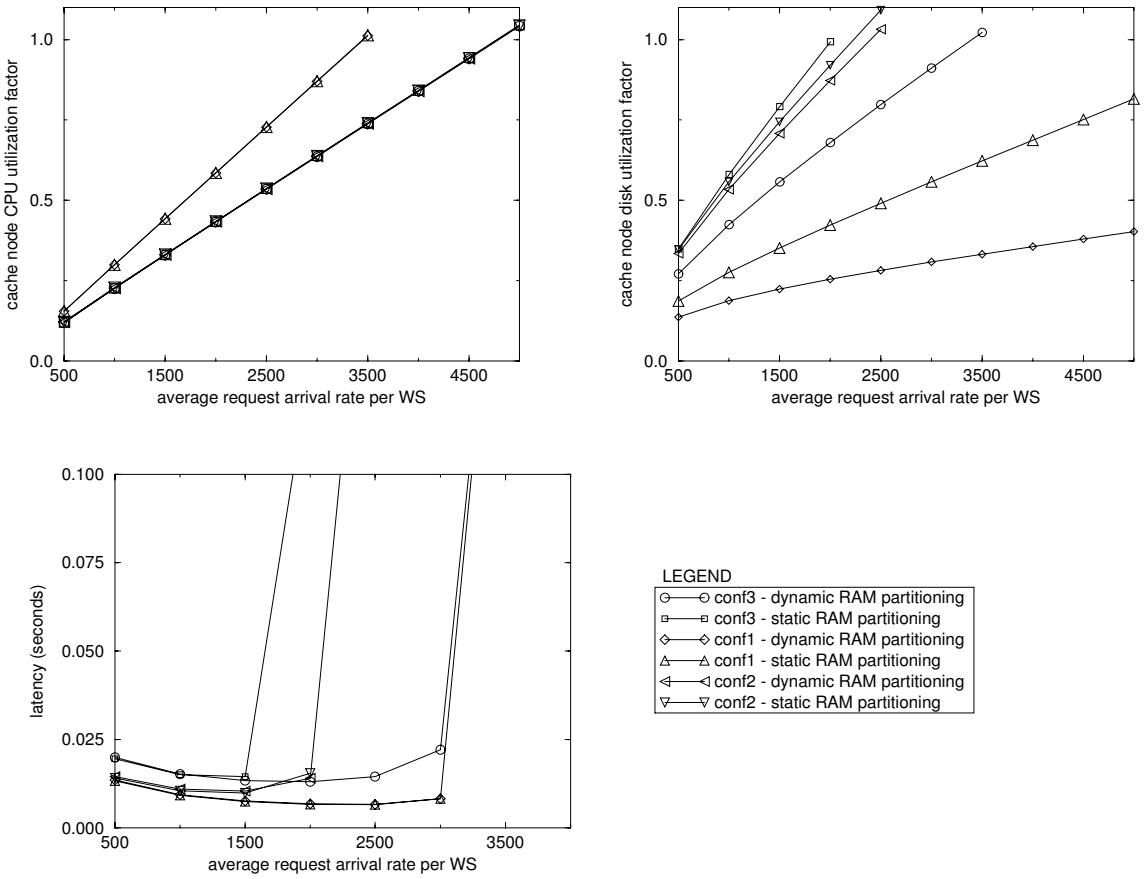

Figure 7: Results for the 100 Web Sites Case.

partitioning the Web sites among the proxy nodes or sharing a subset of the proxy nodes among multiple Web sites. Partitioning schemes include the balancing load of the proxy nodes by suitable assignment of the Web sites to nodes, or random assignment of Web sites to proxy nodes. We also examined static or dynamic partitioning of the RAM in each proxy node among the Web sites. Finally, we examined the impact of transient behavior when a new node is introduced to handle a surge in traffic to a Web site.

From the analysis in Section 4, we can extrapolate the following conclusions:

Architectural alternatives (such as Configuration 1), that try to minimize the miss ratio through the assignment of the minimal number of Web sites to each cache node, have no problem with the disk, but, in case of unbalanced load on the nodes, or surges in traffic, their CPUs can quickly saturate. These solutions work well when the load is well balanced among all the nodes and if the majority of the cacheable objects are in the cache node RAM their performance is independent of the Zipf-like distribution parameter α.

Architectural alternatives (such as Configuration 2), that try to optimize the load balance among the nodes by assigning each Web site to just one node per proxy server and maintaining

roughly the same aggregate load per proxy server, have good steady state performance; however, this can lead to larger RAM miss rates especially when the Zipf-like distribution parameter α is below a threshold, and consequently their disk access frequency becomes high and the disk can quickly saturate. Another drawback of this alternative that it is unable to handle surges in Web site traffic, because the non-shared, static assignment of Web sites to proxy cache nodes leads to overload of that proxy node. Furthermore, if an additional node is dynamically assigned to a Web site detected to have a surge in traffic, the cache warm-up of the new node can overload the home Web site.

Cache node sharing alternatives (like Configuration 3), that try to balance the load of the node assigning each WS to more nodes, are a good compromise between the two previous kind of alternatives. However, even in this case, because a larger number of Web sites now share the same set of nodes than the other alternatives with partitioning, there is a higher RAM cache miss rate and the disk often becomes the bottleneck. For this reason a number of disks, proportional to the number of hosted Web sites, should be used in each cache node.

As this summary indicates, there is no clear "winner" among these approaches. The third alternative, with sharing across multiple proxy nodes, is best when hot documents dominate. The second option, with dynamic assignment of additional proxy nodes is good for both high and moderate skew, but only if the cache warm-up problem is solved. Further work includes study of hierarchical caching or other alternatives to reduce the traffic to the home Web server during cache warm up.

References

[1] http://www.akamai.com

[2] M. Arlitt, and T. Jin, "A Workload Characterization Study of the 1998 Wold Cup Web Site", IEEE Network, May/June 2000, pp.30-37.

[3] L. Breslau, P. Cao, L. Fan, G. Phillipps, and S. Shenker, "Web Caching and Zipf-like Distributions: Evidence and Implications", Proc. of IEEE INFOCOM, 1999.

[4] J. Challenger, A. Iyengar, and P. Dantzig, "A Scalable System for Consistently Caching Dynamic Web Data", INFOCOM 1999.

[5] J. Challenger, A. Iyengar, and D. Dias, "High-Performance Web Site Design Techniques", IEEE Internet Computing, Vol.4, No. 2, March/April 2000.

[6] J. Challeger, A. Iyengar, P. Dantzig, D. Dias, and N. Mills, "Engineering Higly Accessed Web Sites for Performance", Web Engineering 2001.

[7] P. Dantzig, Manager High Volume Web Serving of IBM T. J. Watson Research Center, personal comunication.

[8] http://www.digisle.com

[9] Y. Fujita, M.Murata, and H. Miyahara, "Analysis of Web Server Performance Toward Modeling and Performance Evaluation of Web Systems", Proc. of IEEE SICON, 1998.

[10] S. Gadde, J. Chase, and M. Rabinovich, "Web Caching and Content Distribution: A View From the Interior", Computer Networks and ISDN Systems, Feb. 2001.

[11] http://www.inktomi.com

[12] K. Johnson, J. Carr, M. Day, and M. Kaashoek, "The Measured Performance of Content Distribution Networks", International Web Caching and Content delivery Workshop, May 2000.

[13] D. Karger, E. Lehman, T. Leighton, M. Levin, D. Lewin, and R. Panigrahy, "Consistent Hashing and Random Trees: Distributed Caching Protocols for Relieving Hot Spots on the World Wide Web", Proc. of ACM STOC, 1997.

[14] L. Kleinrock, "Queuing Systems", Volume I: Theory, John Wiley & Sons, 1975.

[15] E. Levy-Abegnoli, A. Iyengar, J. Song, and D. Dias, "Design and Performance of a Web Server Accelerator", Proc. of IEEE INFOCOM, 1999.

[16] http://www.netcache.com

[17] http://www.squid-cache.org

[18] http://www.novell.com

[19] F. Quaglia, B. Ciciani, and M. Colajanni, "An Analitical Comparison of Cooperation Protocols for Web Proxy Servers", Proc. of IEEE MASCOTS, 1999.

[20] http://www.faqs.org/rfcs/rfc3040.html

[21] A. Riska, M. Squillante, S. Yu, Z. Liu, and L. Zhen, "Matrix-Anlytic Analysis of a MAP/PH/1 Queue Fitted to Web Server Data", International Conference on Matrix Analytic Methods in Stochastic Models, July 2002.

[22] P. Rodriguez, C. Spanner, and E. W. Biersack, "Analysis of Web Caching Architectures: Hierarchical and Distributed Caching", IEEE/ACM Transactions on Networking, Vol. 9, No. 4, Aug. 2001, pp. 404-418.

[23] D. Rosu, A. Iyengar, and D. Dias, "Web Proxy Accelerator", Cluster Computing (Baltzer) Vol. 4, No. 4, October 2001.

[24] L. San-qi, and C. Hwang, "On the Convergence of Traffic Measurement and Queueing Analysis: A Statistical-Matching and Queueing (SMAQ) Tool, IEEE/ACM Transactions on Networking, Vol. 5, No. 4, Feb. 1997, pp. 95-110.

[25] J. Song, E. Levy-Abegnoli, A. Iyengar and D. Dias, "Architecture of a Web Server Accelerator", Computer Networks, Vol. 38 , No. 1, January 2002.

[26] A. Wolman, G. Voelker, N. Sharma, N. Cardwell, A. Karlin, and H. Levy, " On the Scale and Performance of Cooperative Web Proxy Cahing", 17th ACM Symposium on Operating Systems Principles, December 1999.