Peer-to-Peer Web Caching: Hype or Reality?

Yonggen Mao, Zhaoming Zhu, and Weisong Shi

Wayne State University
{*yoga,zhaoming,weisong*}*@wayne.edu*

Abstract

In this paper, we systematically examine the design space of peer-to-peer Web caching systems in three orthogonal dimensions: the caching algorithm, *the* document lookup algorithm, *and the* peer granularity. *Based on the observation that the traditional* URL-based *caching algorithm suffers considerably from the fact of cacheability decrease caused by the fast growing of dynamic and personalized Web content, we propose to use the* content-based *caching algorithm. In addition to compare two existing document lookup algorithms, we propose a simple and effective* geographic-based *document lookup algorithm. Four different peer granularities, i.e.,* host level, organization level, building level, *and* centralized , *are studied and evaluated using a seven-day Web trace collected at a medium-size education institution. Using a trace-driven simulation, we compared and evaluated all design choices in terms of two performance metrics:* hit ratio *and* latency reduction. *Finally, several implications derived from the analysis are also discussed.*

1. Introduction

Peer-to-peer networking has become a hot research topic recently [12, 18]. Peer-to-peer Web caching is thought as one of the potential applications that could be benefited from these underlying peer-to-peer substrates, and has been exploited by several projects [6, 17]. In [17], Xiao *et al.* proposed a browser-aware proxy server model and evaluated using BU-95 trace [3] collected from Boston University (1995) and NLANR-uc trace [5] (2000). In Squirrel [6], Iyer *et al.* presented a peer-to-peer Web caching system built on top of the Pastry [12], and evaluated using the traces of Microsoft Research Redmond campus [16] (1999) and Cambridge campus (2001) respectively. Although these two studies showed optimistic results for peer-to-peer Web caching, the study of Wolman *et al.* [16] indicated a relative pessimistic results using the traces from Microsoft Corporation (1999) and University of Washington's (1999).

The possible reasons for the controversial observation of above studies are: (1) those studies worked with different traces; (2) the peer granularity of these studies was different. Squirrel and Xiao *et al.*'s studies were at the *host level*, while Wolman *et al.*'s study was at the *organization level*. Furthermore, from the perspective of users, the *latency reduction* resulted from cooperative caching is more important than *hit ratio*, those previous efforts, however, did not quantitatively evaluate the latency improvement.

On the other hand, recent study [13] shows the fast growing of the dynamic and personalized Web content. This trend will reduce the cacheability of cooperative Web caching significantly under the conventional *URL-based* caching algorithm. Fortunately, recent study [7, 19] shows that dynamic objects have a large portion of repeatness based on their content digests. This repeatness provides an opportunity to improve the cacheability, and motivates us to propose a *content-based* caching algorithm for peer-to-peer Web caching.

In this paper, we intend to answer the following question: *Is peer-to-peer Web caching a hype or a reality?* We first systematically examine the design space of a peer-to-peer Web caching system in three orthogonal dimensions: the *caching algorithm*, the *document lookup algorithm*, and the *peer granularity*. Based on the observation that the traditional *URL-based* caching algorithm suffers considerably from the fact of cacheability decrease caused by the fast growing of dynamic and personalized Web content, we propose to use the *content-based* caching algorithm which exploits the fact of the large repeatness of Web objects even though their URLs are different. In addition to compare two existing document lookup algorithms, we propose a simple and effective *geographic-based* document lookup algorithm. Four different peer granularities, i.e., *host level*, *organization level*, *building level*, and *centralized* , are studied and evaluated using a seven-day Web trace collected from a medium-size university. Using a trace-driven simulation, we compared and evaluated all the design choices in terms of two performance metrics: *hit ratio* and *latency reduction*. The reasons that we collected the trace by ourselves instead of using existing public traces are: (1) most available traces are lack of the latency information which is one of performance metrics in our study; (2) the entire Web object is required to calculate the content digest, which is not available in any present trace.

The experimental results suggest that: (1) ideally, the *content-based* caching algorithm could improve the cacheability of Web objects substantially, increasing from 6.9% (*URL-based*) to 62.0% (*content-based*); (2) the document sharing among peers is very effective, ranging from 22.0% (*building level*) to 34.2% (*host level*); (3) the average user-perceived latency is reduced three to six times compared with the measured latency at all peer granularities using the hierarchical index-server algorithm [17]; (4) the proposed *geographic-based* document lookup algorithm has a comparable *hit ratio* and significant *latency reduction*.

Based on these observations, we derive several implications for peer-to-peer Web caching: (1) there is a need to deploy the *content-based* Web caching mechanism; (2) the *organization* or *building* level peer-to-peer Web caching using the hierarchical index-server is the most appropriate choice; (3) the *geographic-based* lookup algorithm should be exploited further to benefit from its superior *latency reduction* and easy implementation.

Our contributions of this study include: (1) systematically examining the design space of peer-to-peer Web caching; (2) validating the great potential of the *content-based* caching algorithm. To our knowledge, this work is the first performance evaluation using real Web trace with content digests; (3) comprehensive evaluating the performance of Web caching in terms of two performance metrics; (4) proposing a *geographic-based* document lookup algorithm.

The rest of the paper is organized as follows. Section 2 examines the design space of peer-to-peer Web caching systems. A comprehensive comparison of different algorithms in terms of two performance metrics is reported in Section 3. Several implications derived from the analysis are listed in Section 4. Related work and concluding remarks are listed in Section 5 and Section 6 respectively.

2. Design Space of Peer-to-Peer Web Caching

As illustrated in Figure 1, there are three orthogonal dimensions in designing a peer-to-peer Web caching system: the *caching algorithm*, the *lookup algorithm*, and the *peer granularity*. Note that, the notion of peer, or peer cache, in this paper is quite flexible. Unlike traditional P2P network [12, 15] where the notion of peer refers to a physical end host, each peer cache is defined as the one which performs the caching function on behalf of host(s) inside its scope and cooperates with other counterparts at the same level. For example, an end host itself is a *host level* peer cache. It performs the caching function for itself and cooperates with other *host level* peer caches. Napster, Gnutella, and KaZaA follow this concept. *Organization*/*building level* peer caches perform the caching function for hosts inside their scope and cooperate with other *organization*/*building level* peer caches. Centralized cache performs the caching function for all hosts behind it and does not have any same level peer cache to cooperate with.

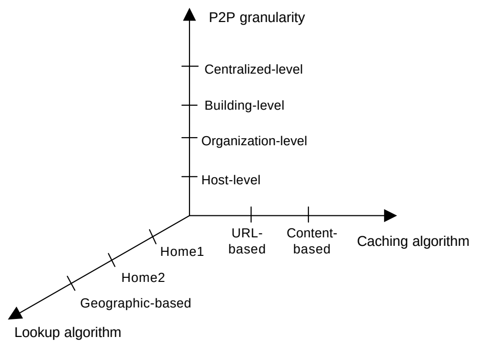


Figure 1. A three dimension design space of peer-to-peer Web caching.

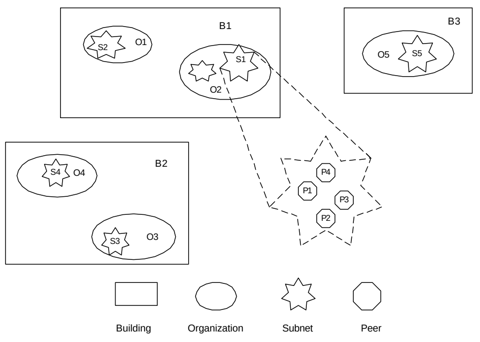
2.1. Caching Algorithm

Two caching algorithms, the *URL-based* and the *content-based*, are evaluated in this paper. The *URL-based* caching algorithm is based on the URL of static Web object and its related freshness time, and has been widely used in Web caching. The *content-based* caching algorithm is inspired by our recent study [19], where we found that the static Web content only occupied 10.2% of total Web requests, and there were 59.1 % Web requests, which are repeated based on their hash-based digests, are traditionally perceived uncacheable. This implies that these uncacheable Web content could be cached if certain protocol could be designed based on the digest value. The basic idea of the *content-based* caching algorithm is the content digest is exchanged apriori to decide whether or not the requested object should be fetched.

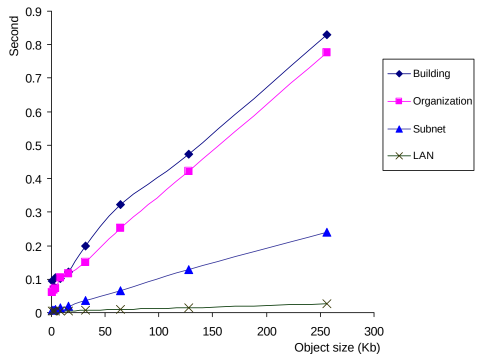
2.2. Document Lookup Algorithm

As in any P2P networking system, the document lookup algorithm is the core of the whole design. Three lookup algorithms are evaluated in this paper, namely *home1*, *home2* and *geographic-based* (*Geo* in short). The basic idea of the *home1* algorithm is that a high level index server maintains an index file of all Web objects stored in hosts within its peer scope. This protocol is used in Xiao *et al.*'s work [17]. When a host requests a Web document, it first checks its local cache. If the request misses, the host will send the request to the index server to search the index file. If the request hits at the host *i*, the index server will inform the host *i* to send the Web object to the request host. If the request misses, the request host will go to the original server directly.

In the *home2* algorithm, each requested document is associated with a certain host as its logical home (based on its



(a) Peer granularity



(b) Four levels latency model

Figure 2. Peer granularity and the simple latency estimation model.

hash value of URL or digest). When a host requests a document, it will check its local cache first. If missing happens, it will use P2P routing algorithm to forward the request to the corresponding home. The home will send the requested document back to the client if the request is hit. If missing happens again, the home will send the request to the original server and forward the Web object to the request host.

The *Geo* algorithm comes from our intuition, that people will have similar Web-browsing interests at same geographic location. Currently, only hosts located in the same subnet are considered geographically closed and contacted to query for the missing document. It can be easily implemented using IP level multicast (if available). Otherwise, application-level multicast can be used here too [2]. When a local cache missing happens, the client first multicast its request within the subnet. If the request hits at a host's local cache, that host will send the Web object back. If there is no reply, the request host will send the request to the original server.

2.3. Peer Granularity

Figure 2 shows four possible peer granularities for a medium-size institution, which has tens of building, multiple logical organizations, and thousands of computers. Each building could have more than one organization. In each organization, there also possibly exist multiple subnets. The P2P model could be applied at any of those levels. In our simulation, we implement *host*, *organization*, *building* level peer-to-peer Web caching, and a *centralized* Web caching.

2.4. A Simple Latency Estimation Model

To estimate the possible *latency reduction* of different design options, we use a simple latency model to compute the latency between any two hosts. According to Figure 2 (a), there are four possible host-to-host latency models: hosts within the same LAN (in the reach of the same switch); hosts not in the same LAN but within the same subnet; hosts between different organizations but within the same building and hosts between different buildings. To measure these latencies, we ran a client program to fetch different Web objects, with size ranging from 1KB to 256KB, against an Apache Web server inside campus. We calculated the latency between the request and the last-byte of response as in Figure 2(b). Since these latency values include not only the delay of network, but also the overhead of the application, we call this *application level latency*. Using the minimum square linear programming approach, we found all latencies follow a linear model, i.e., $f(x) = ax + b$, where x is a variable of the file size in KByte, $f(x)$ represents the latency in second, details of the parameter a and b are available in the technical report version [8].

3. Analysis Results

We adopt the trace-driven approach to examine the different design choices of peer-to-peer Web caching, implementing two caching algorithms, and three document lookup methods at four peer granularities. We collected seven-day period (Aug 25, 2003 — Aug 31, 2003) Web traffic from a middle-size education institution via the WebTACT tool developed at Wayne State University [19].

Totally, there are 8,889 unique hosts observed from the trace based on their IP addresses. These hosts belong to 110 subnets, disperse in 77 organizations that are located in 60 buildings. In order to emulate the behavior of real deployed Web caches, we set the size limit for the caches at centralized, building, organization and host levels to 1GB, 300MB, 100MB and 10MB, respectively. We also limit the maximum size of cacheable objects to 20% of the corresponding cache capacity. Although in the real life the cache size could be set much bigger, we are interested in the relative relationship (relative size ratio) among caches at different levels. A least-recent used (LRU) replacement algorithm is used in our simulation. We also exploit the cacheability of seven different dynamic content types in the technical report version of this paper [8].

3.1. Performance Metrics

Although most previous studies chose performance metrics like the *hit ratio* and the *byte hit ratio* to evaluate Web caching, from the perspective of clients, the user-perceived latency is also crucial. In this study, we focus not only on the *hit ratio* and the *byte hit ratio*, but also on the *latency reduction*, which is the improvement of the estimated latency compared with the measured latency. In addition, we also introduce a notion of *peer sharing gain* to indicate the resource share degree between those peers. The *peer sharing gain* is defined as the ratio of the number of remote hits and the number of total hits. Regarding to the *latency reduction*, it could be improved (positive) or deteriorated (negative). We use $L_{improve}$ and $L_{deteriorate}$ to depict these two cases respectively.

3.2. Hit Ratio

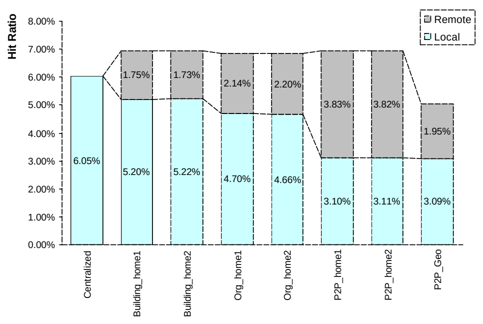
In terms of the *hit ratio*, including both request *hit ratio* and *byte hit ratio*, we examine the different design choices. Figure 3 shows that the *hit ratio* and *byte hit ratio* of the *URL-based* and the *content-based* caching algorithms at four peer granularities respectively. Each item in the X axis represents a combination of peer granularity and document lookup algorithm. For example, host-geo means the *geographic-based* document lookup algorithm is applied at *host level*. The Y axis of Figure 3 (a) and (c) indicates the *hit ratio* in percentage. The Y axis of Figure 3 (b) and (d) shows the *byte hit ratio* in percentage. In Figure 3, each bar consists of two parts, the *local hit* (lower part) and *remote hit* (upper part). The *local hit* refers to the hit happened at the default cache (for example, at *host level* the default cache is the local cache of host itself), and the *remote hit* refers to the hit happened at the requested document's home cache (*home1* and *home2*), or neighbor with in the same subnet (*Geo*). For the centralized cache, the remote hit is zero.

Note that, for the *content-based* caching algorithm, we only simulate the uncacheable (dynamic) Web content, while this algorithm works for the static Web content as well. Thus, the total *hit ratio* or *byte hit ratio* of the *content-based* caching algorithm is the sum of that from *content-based* and that from *URL-based* correspondingly. The *URL-based* caching algorithm, as illustrated in Figure 3 (a) and (b) has the lower cache hits in terms of the *hit ratio* and *byte hit ratio* compared with the *content-based* caching algorithm as showed in Figure 3 (c) and (d). From those figures, we observe that the local hit ratio decreases from *centralized level* to *host level* caused by the total cache size decreasing at each level; and the remote hit ratio increases respectively for both *URL-based* and *content-based* caching algorithms due to the sum of peers cache size increasing. In general, *home1* has a higher hit ratio than *home2*. The reason is the *home2* algorithm will store the requested web object at both the requesting host and the home of the requested object, which causes the disk space redundancy to decrease the hit ratio. We will discuss the *hit ratio* and the *byte hit ratio* based on caching algorithms, document lookup algorithms and peer granularities separately.

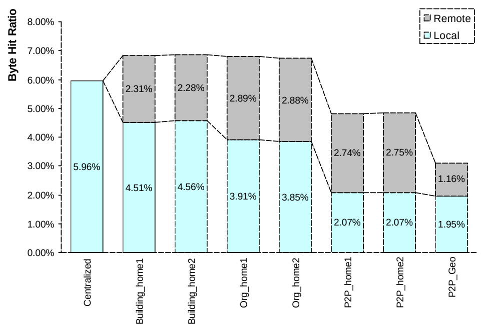
3.2.1. Caching Algorithm In our study, we are interested in which caching algorithm could achieve higher *hit ratio* and *byte hit ratio*. From Figure 3(a) and (c), we find that the *hit ratio* of the *URL-based* algorithm has the value from 5.04% to 6.94%, while the *content-based* algorithm gains an order of magnitude additional *hit ratio*, ranging from 47.44% to 59.08%, depending on different P2P granularities. The reason for the lower hit ratio of the *URL-based* caching algorithm is that there are 49.6% requests whose TTL's values are zero, probably caused by the cache busting technique [7], and therefore those requests are uncacheable for the traditional caching algorithm. These results indicate that the *content-based* algorithm has the great potential to increase the *hit ratio*. Figure 3(b) and (d) report that the *byte hit ratio* of the *URL-based* caching algorithm is from 3.11% to 6.84%, while the *content-based* algorithm gains additional *byte hit ratio* from 23.55% to 30.22%, depending on different peer granularities. Surprisingly, it can be seen from the figure that the *byte hit ratio* does not gain as much as the *hit ratio* using the *content-based* caching algorithm. The possible reasons are: (1) the cache busting technique tends to apply on small object, like advertised `gif` or `jpeg` images; (2) some very small HTTP response heads (for example, `404` for "document not found" in HTTP protocol) happen a lot of times, and they have the same digest.

3.2.2. Document Lookup Algorithm Logically, the *hit ratio* resulting from a document lookup algorithm is determined by the scope of lookup, independent of the specific document lookup algorithms. The *hit ratio* difference between *home1* and *home2*, as shown in Figure 3, is caused by two possible reasons: (1) the space limitation of cache size; (2) the disk redundancy of the *home2* algorithm. Compare with *home1*, *home2* stores each requested web object at two locations, one is at the requesting host, and the other is at the logic home of the requested object. This will cause some disk space redundancy to decrease the hit ratio. However, it can be seen from the Figure that the *hit ratio* of *geographic-based* algorithm is lower than that of two other algorithms. This is caused by the limited host number in each subnet searched by the *Geo* algorithm. Although the *Geo* algorithm only has two third of the *hit ratio* compared with *home1* and *home2*, we still think it as a very promising document lookup algorithm because it uses only one percent of host population on average compared to the *home1* and the *home2* algorithms. As such, the *Geo* algorithm can scale very well.

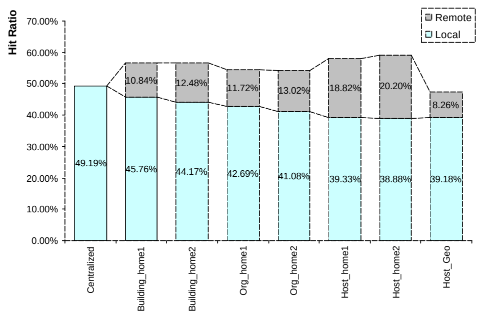
3.2.3. Peer Granularity In this paper, we are interested in which peer granularity level the peer-to-peer Web caching should be deployed. An analytic result indicates that the *hit ratio* should increase with the peer granularity changing from



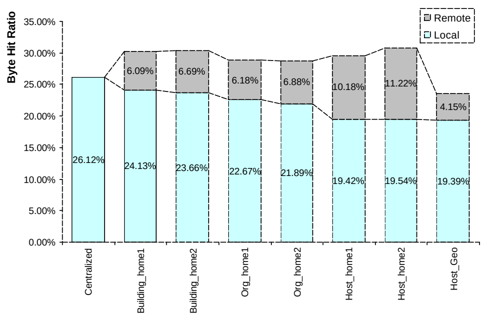
(a) *URL-based* hit ratio



(b) *URL-based* byte hit ratio



(c) *Content-based* hit ratio



(d) *Content-based* byte hit ratio

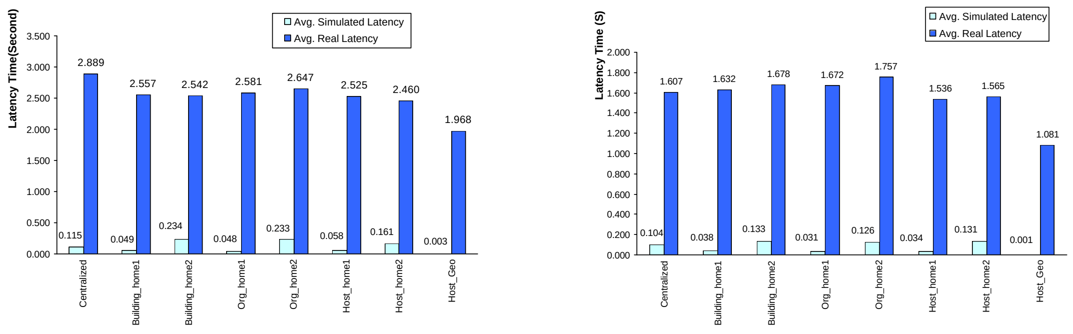
Figure 3. The hit ratio and byte hit ratio of different algorithms.

	URL-based		Content-based	
Granularity	Peer share gain	Peer share gain (byte hit)	Peer share gain	Peer share gain (byte hit)
Building-home1	25.2%	33.9%	19.1%	20.1%
Building-home2	24.9%	33.4%	22.0%	22.0%
Org-home1	31.2%	42.5%	21.5%	21.4%
Org-home2	32.1%	42.4%	24.1%	23.9%
Host-home1	57.0%	55.1%	32.4%	34.4%
Host-home2	55.2%	57.1%	34.2%	36.5%
Host-Geo	38.6%	37.2%	17.4%	17.6%

Table 1. The peer share gain of two caching algorithms at three peer granularities.

the *centralized level* to the *host level*. The increment of the *hit ratio* is caused by the cache capacity increasing with the changing of peer granularity. Figure 3 shows that the *hit ratio* and the *byte hit ratio* increase with peer granularity changing from the *centralized level* to the *host level*, but there are some exceptions for the *URL-based* algorithm at the *organization level* P2P caching. The possible culprits are: (1) the total cacheable Web objects number is small, and their total bytes are less than the sum of cooperative cache capacity; (2) the object size limitation at the *organization level* and the *host level* is 20MB and 2MB respectively, and there exist some files that are too large to be cached. For the exception of the *content-based* lookup algorithm at the *organization level* whose *hit ratio* and *byte hit ratio* are less than those of *building level* cache, the possible reason is that the capacity sum of the *organization level* cache is 7,700MB, which is less than the capacity sum of the *building level* cache, 18,000MB. Another exception is the relative low remote hit ratio in host-Geo scenario, which is caused by the limitation of its neighbor population (limited by the size of subnet). Despite this, it still achieves a very impressive *hit ratio*.

3.2.4. Peer Share Gain The motivation of peer-to-peer Web caching is to share Web objects among a group of clients. We define a notion of *peer sharing gain* to indicate the resource share degree between those peers. The *peer sharing gain* is defined as the ratio of the number of remote hits and the number of total hits. Table 1 shows the *peer share gain* in terms of both request hit and byte hit based on different peer granularities and two caching algo-



(a) *URL-based* (b) *Content-based*

Figure 4. The average latency of simulation and measurement.

rithms. From Table 1, we can see that *host level* caching has the highest *peer share gain* in terms of both *hit ratio* and *byte hit ratio*, for two caching algorithms, *URL-based* and *content-based*. Table 1 also shows that *building* and *organization* levels have around 20% sharing gain for the *hit ratio*, and 20% for the *byte hit ratio* when applying the *content-based* caching algorithm. This observation implies that peer-to-peer Web caching can efficiently share Web objects in terms of both *hit ratio* and *byte hit ratio* at different peer granularities. Note that, the *content-based* caching algorithm actually reduces the peer share gain in terms of both the number of requests and the number of bytes. The reason is the *content-based* caching algorithm will cause more cache replacement due to the cache size limit in our simulation.

3.3. Latency Reduction

Now we are in the position to examine the corresponding *latency reduction* resulting from peer-to-peer Web caching. We use the user-perceived last-byte latency as a performance metric to examine all possible caching design space. Figure 4 illustrates the average latency obtained from simulation and measurement for different caching algorithms at all possible peer granularities. In the Figure, the average simulation latency is the average of all simulated latencies (calculated from the simple latency model) resulted from hit requests during the simulation. On the other hand, the average measurement latency is the average of all measured latencies (calculated from the trace data) corresponding to *these* hit requests. The average simulation latency reduces from three to four times for the *URL-based* caching algorithm and from four to eight times for the *content-based* caching algorithm compared to the average measurement latency. The host-home1 scenario has the best *latency reduction* except for *Geo* which has a less *hit ratio*, but host-home1 is difficult to implement in the real situation due to the concern of scalability.

Table 2 shows the percentage of latency improvement represented by $L_{improve}$ and the latency deterioration represented by $L_{deteriorate}$ for all hit requests in different peer-to-peer Web caching design options. $L_{improve}$ represents the number of hit requests where the simulation latency is less than the measurement latency. $L_{deteriorate}$ represents the difference between total number of hits and $L_{improve}$. From the Table, we observe that although the average simulation latency is reduced greatly compared to the average measurement latency, the $L_{improve}$ is almost equal to or less than $L_{deteriorate}$ except for the host-Geo scenario. This abnormality indicates that we need examine $L_{improve}$ and $L_{deteriorate}$ in more detail. The cumulative distribution functions (CDFs) of $L_{improve}$ and $L_{deteriorate}$ show that the $L_{improve}$ is uniformly distributed, while in the $L_{deteriorate}$ is likely exponentially distributed, which explain the significant improvement shown in Figure 4.[1] Furthermore, the CDF figures also show that in the *building level* applying the *URL-based* caching algorithm using the *home1* lookup algorithm, the average improved time of $L_{improve}$ is 1.412 seconds while the average deteriorated time of $L_{deteriorate}$ is 0.058 second. This indicates a fact that if a hit request latency is improved, it will improve greatly. On the other hand, if a hit request latency is deteriorated, it will deteriorate very limited. Next, we will discuss the *latency reduction* in terms of document lookup algorithms, P2P granularities and caching algorithms.

3.3.1. Documents Lookup Algorithm The *Geo* has the most significant *latency reduction* in both average simulation latency and $L_{improve}$ (98.7%). The reasons are: (1) it does not need routing to locate the Web document; (2) the latency within a LAN or subnet is minimal. From Figure 4, *home1* is observed to be superior to *home2* in terms of both average simulation latency and $L_{improve}$. This can be explained logically as *home2* needs more routing steps to locate Web objects' home as compared to *home1*.

1 Due to space limit, we did not show the figures here, but they are available in technical report version [8].

Caching algorithm	URL-based		Content-based	
Granularity	$L_{improve}$	$L_{deteriorate}$	$L_{improve}$	$L_{deteriorate}$
Centralized	373,754 (57.5%)	276,463 (42.5%)	2,972,846 (56.3%)	2,311,386 (43.7%)
Building-home1	626,884 (84.0%)	119,081 (16.0%)	5,102,580 (83.9%)	976,728 (16.1%)
Building-home2	250,873 (33.6%)	496,014 (66.4%)	3,084,401 (50.7%)	3,000,871 (49.3%)
Org-home1	622,764 (84.8%)	111,570 (15.2%)	5,176,172 (88.6%)	667,359 (11.4%)
Org-home2	251,161 (34.1%)	485,337 (65.9%)	2,993,966 (51.5%)	2,817,516 (48.5%)
Host-home1	577,786 (77.6%)	167,176 (22.4%)	5,278,443 (84.5%)	967,573 (15.5%)
Host-home2	434,263 (58.4%)	309,775 (41.6%)	3,195,313 (50.4%)	3,150,430 (49.6%)
Host-Geo	533,895 (98.7%)	7,065 (1.3%)	5,034,302(98.8%)	62,223 (1.2%)

Table 2. The latency improvement and deterioration of two caching algorithms at four peer granularities.

3.3.2. Peer Granularity The *centralized level* caching has a comparable *latency reduction* compared with *building level*, *organization level* and *host level*. But it will suffer scaling problem in a real implementation with a large client population. The *building level* has very similar results in terms of average latency and $L_{improve}$ compared with *organization level* cache. Although *host level* caches have similar *latency reduction* compared with *building level* and *organization level* respectively, they have a better $L_{improve}$(e.g., 84.5% for *home1* and 50.4% for *home2*) than *building level* (e.g., 83.9% for *home1* and 50.7% for *home2*) using the *content-based* caching algorithm.

3.3.3. Caching Algorithm Figure 4(a) and (b) also show that the *content-based* caching algorithm has a better *latency reduction* than that of the *URL-based* caching algorithm. The possible reasons are: (1) in our simulation, we assume that a client, which sends a request, knows the digest of the request Web content in prior. This is impractical in the real implementation. Thus, a possible latency overhead for the *content-based* caching algorithm is expected in a real deployment; (2) dynamic Web objects usually tend to have longer measured latencies caused by dynamic generation [19]. When these objects are cached and hit, these dynamic generating latency will be reduced.

4. Implications

Based on the analysis results in the last section, several implications could be derived as follows:

- **Need protocol support for deploying the *content-based* Web caching mechanism**: The results of the experiment show that the *content-based* caching algorithm improves the *hit ratio* and latency tremendously.
- **Tradeoff between *latency reduction* and scalability**: The results show that the *home1* document lookup algorithm has a good *latency reduction*, but it has a scalability problem. Although the *home2* algorithm is exempted from the scalability problem, it has a less *latency reduction* than the *home1* counterpart. Therefore, we argue that peer-to-peer Web caching at *organization level* or *building level* using the *home1* lookup algorithm is a good choice. Our recent work on Tuxedo is an alternative to solve this problem [14].
- **Exploiting the *geographic-based* lookup algorithm**: The results show that the *geographic-based* lookup algorithm has an acceptable *hit ratio* and a significant *latency reduction*.

5. Related Work and Discussions

Peer-to-peer Web caching (also known as cooperative Web caching) has been extensively studied in recent years [6, 17, 16, 14, 1]. To the best of our knowledge, our effort is the first try to systematically examine the design space of peer-to-peer Web caching in three dimensions, and quantitatively evaluate their performance in terms of two performance metrics: *hit ratio* and *latency reduction*.

Cooperative caching was first proposed by Dahlin *et al.* [4] in the context of memory caching sharing in file system. However, we focus on Web content sharing, and evaluate different peer granularities, caching algorithms, and document lookup algorithms in this paper.

The pioneer work in cooperative caching was conducted by Wolman *et al.* in 1999 [16]. This is the closest work to our analysis. There are three differences exist. First, the peer grains examined in our paper is wider than their work. Second, the qualitative latency improvement analysis in [16] was done by an analytical model, while we perform a quantitatively study. Finally, a new *content-based* caching algorithm is proposed in this paper.

Recently, Iyer *et al.* proposed Squirrel [6], a peer-to-peer Web caching system built on the Pastry [12]. Xiao *et al.* studied the reliability and scalability of a browser-aware proxy server. We implemented both of their algorithms in this paper for comparison purposes. In addition to hit ratio and cooperative hit ratio, this paper compares the likely *latency re-*

duction as well. Furthermore, the traces used in our analysis is more up-to-date than the traces used in [6, 17].

The *content-based* caching algorithm proposed in this paper is motivated by the fact that there exist a large amount of content repeatness in Web traffic. This phenomenon was observed in our recent traffic analysis [19] and [7]. The recent proposed value-based Web caching (VBWC) by Rhea *et al.* [11] shares the similar idea as ours, but we come out this idea independently.

Peer lookup algorithms are a very hot research topic in recent years, *i.e.*, Pastry [12]. In this paper, the average latency of the *home2* protocol is based on Pastry. Due to the similarity of these protocols (less than $O(log(n))$ hops), we argue that our analysis can be easily extended to other algorithms. The simple *geographic-based* lookup algorithm proposed in this paper produces a reasonable performance. Theoretically, we believe that our work will definitely benefit from several recent work on geographically-aware clustering technologies, such as landmark based binning algorithm [10], global network positioning (GNP) service [9].

6. Summary

In this paper, we have systematically examined the design space of peer-to-peer Web caching, in terms of three design dimensions: *the caching algorithm*, *the lookup algorithm*, and *the peer granularity*. Our study shows that the *content-based* caching algorithm could greatly improve the Web objects cacheability; peer-to-peer Web cache at different granularities can share Web documents efficiently, ranging from 22.0% (at *building level*) to 34.2% (at *host level*); the simulated latency could be reduced three to six times compared with the measured latency; and the *geographic-based* document lookup algorithm has comparable *hit ratio* and a significant *latency reduction*. Based on these observations, we argue that the organization/building level peer-to-peer Web caching using a hierarchical index-server (*home1*) is the most appropriate choice. Our trace is available for research purpose at `http://mist.cs.wayne.edu`.

References

[1] C. Canali, W. Cardellini, M. Colajanni, R. Lancellotti, and P. S. Yu. Cooperative architectures and algorithms for discovery and transcoding of multi-version content. *Proc. of the 8th International Workshop on Web Caching and Content Distribution (WCW'03)*, Sept. 2003.

[2] M. Castro and et al. An evaluation of scalable application-level multicast built using peer-to-peer overlays. *Proc. of IEEE Conference on Computer Communications (INFOCOM'03)*, Mar. 2003.

[3] C. Cunha, A. Bestavros, and M. E. Crovella. Characteristics of WWW client-based traces. Tech. Rep. BU-CS-95-010, Computer Science Department, Boston University, July 1995.

[4] M. Dahlin, R. Wang, T. Anderson, and D. Patterson. Cooperative caching: Using remote client memory to improve file system performance. *Proc. of the First USENIX Symposium on Operating Systems Design and Implementation*, Nov. 1994.

[5] IRCache Project. A distributed testbed for national information provisioning, `http://www.ircache.net/Cache/`.

[6] S. Iyer, A. Rowstron, and P. Druschel. SQUIRREL: A decentralized, peer-to-peer web cache. *Proceedings of the 12th ACM Symposium on Principles of Distributed Computing (PODC 2002)*, July 2002.

[7] T. Kelly and J. Mogul. Aliasing on the world wide web: Prevalence and performance implications. *Proc. of the 11th International World Wide Web Conference (2002)*, May 2002.

[8] Y. Mao, Z. Zhu, and W. Shi. Peer-to-peer web cache: Hype or reality? Tech. Rep. MIST-TR-2003-004, Department of Computer Science, Wayne State University, May 2003.

[9] T. S. Ng and H. Zhang. Predicting internet network distance with coordinate-based approaches. *Proc. of IEEE Conference on Computer Communications (INFOCOM'02)*, June 2002.

[10] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. Topologically-aware overlay construction and server selection. *Proc. of IEEE Conference on Computer Communications (INFOCOM'02)*, June 2002.

[11] S. Rhea, P. Eaton, D. Geels, H. Weatherspoon, B. Zhao, and J. Kubiatowicz. Pond: The oceanstore prototype. *Proc. of the 2nd USENIX Conf. On File and Storage Technologies*, pp. 1-14, Apr. 2003.

[12] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large scale peer-to-peer systems. *IFIP/ACM Middleware 2001*, 2001.

[13] W. Shi, E. Collins, and V. Karamcheti. Modeling object characteristics of dynamic web content. *Journal of Parallel and Distributed Computing)* 63(10):963–980, Oct. 2003.

[14] W. Shi, K. Shah, Y. Mao, and V. Chaudhary. Tuxedo: A peer-to-peer caching system. *Proceedings of PDPTA 2003*, June 2003.

[15] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. *ACM SIGCOMM'2001*, 2001.

[16] A. Wolman, G. M. Voelker, N. Sharma, N. Cardwell, A. Karlin, and H. M. Levy. On the scale and performance of cooperative web proxy caching. *Proc. of 17th ACM Symposium on Operating Systems Principles (SOSP)*, pp. 16-31, Dec. 1999.

[17] L. Xiao, X. Zhang, and Z. Xu. On reliable and scalable peer-to-peer web document sharing. *Proceedings of 2002 International Parallel and Distributed Processing Symposium*, Apr. 2002.

[18] B. Zhao, J. Kubiatowicz, and A. Joseph. Tapestry:an infrastructure for fault-tolerant wide-area location and routing. Tech. Rep. UCB/CSD-01-1141, Computer Science Division, UC Berkeley, Apr. 2001.

[19] Z. Zhu, Y. Mao, and W. Shi. Workload characterization of uncacheable http content. Tech. Rep. MIST-TR-2003-003, Department of Computer Science, Wayne State University, May 2003.