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

Yonggen Mao, Zhaoming Zhu, and Weisong Shi

Department of Computer Science
Wayne State University
{*ygmao,zhaoming,weisong*}*@wayne.edu*

Abstract

Peer-to-peer Web caching has attracted a great attention from the research community recently, who believes it as a potential killer application for peer-to-peer networking. However, the observed results from several previous efforts are not consistent, even controversial.

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 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 *digest-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*. The experimental results suggest that: (1) ideally, the *digest-based* caching algorithm could improve the cacheability of Web objects substantially, from 6.9% (*URL-based*) to 62.0% (*digest-based*); (2) the document sharing among peers is very effective, 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-based (*home1* in our jargon) document lookup algorithm; (4) the proposed *geographic-based* document lookup algorithm has comparable *hit ratio* and significant *latency reduction*. Finally, several implications derived from these observations are also listed.

Key Words: Peer-to-Peer Systems, Web Caching, Performance Evaluation

1 Introduction

Peer-to-peer networking has become a hot research topic recently [13, 16, 21, 25]. 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 [9, 24]. In [24], Xiao *et al.* proposed a browser-aware proxy server model and evaluated using BU-95 trace [6] collected from Boston University (1995) and NLANR-uc trace [8] (2000). In Squirrel [9], Iyer *et al.* presented a peer-to-peer Web caching system built on top of the PASTRY [16], and evaluated using the traces of Microsoft Research Redmond

campus [23] (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.* [23] 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 *host level*, while Wolman *et al.*'s study was at the *organization level*. Furthermore, from the perspective of users, the *latency reduction* resulted from those cooperative caching is more important than *hit ratio*. But those studies did not quantitatively evaluate the latency improvement.

On the other hand, recent studies [3, 17] show the fast growing of the dynamic and personalized Web content. This trend will reduce the cacheability of cooperative Web caching significantly under *URL-based* caching algorithm. Fortunately, recent study [10, 26] shows that the dynamic objects have a large portion of repeatness based on their content digest. This repeatness provides an opportunity to improve the cacheability, and motivates us to propose a *digest-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 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 a *digest-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 in order to calculate the content digest, which is not available in any present trace.

The experimental results suggest that: (1) ideally, the *digest-based* caching algorithm could improve the cacheability of Web objects substantially, increasing from 6.9% (*URL-based*) to 62.0% (*digest-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 *home1* routing algorithm; (4) the proposed *geographic-based* document lookup algorithm has 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 *digest-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; (4) dynamic `type2` (*DynGen*) and `type7` (*ZeroTTL*) are most promising to benefit from the *digest-based* caching algorithm.

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 *digest-based* caching algorithm. To our knowledge, this work is the first performance evaluation using real Web trace with content digest; (3) comprehensive evaluating the performance of Web caching in terms of two performance metrics; (4) proposing a *geographic-based* document lookup algorithm.

Algorithm	Description
Home1	A hierarchical index server is used to maintain web content in peer's cache [24]
Home2	A decentralized index (using hash value) is used to locate web content in each peer [9]
Geo	Only hosts located in the same subnet are considered

Table 1: Explain of document lookup algorithms

The rest of the paper is organized as follows. Section 2 examines the design space of peer-to-peer Web caching systems. Section 3 describes the trace data collection, and classification of multiple dynamic content. A comprehensive comparison of different algorithms in terms of two performance metrics is reported in Section 4. Several implications derived from the analysis are listed in Section 5. Related work and conclusion remarks are listed in Section 6 and Section 7 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 [13, 16, 21] 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.

2.1 Caching Algorithm

Two caching algorithms, the *URL-based* and the *digest-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 *digest-based* caching algorithm is inspired by our recent study [26], 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 *digest-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) as listed in Table 1. 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 [24]. When a host requests a Web document, it first checks its local cache. If the request misses, the

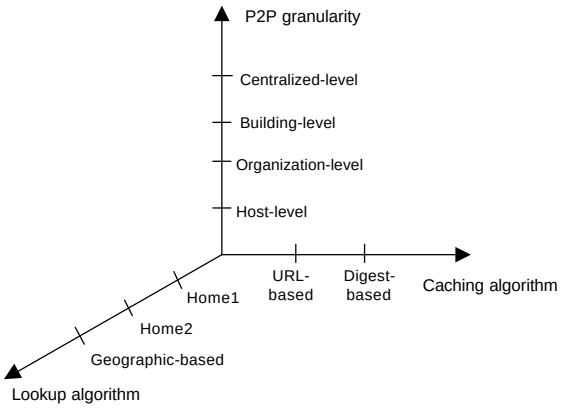
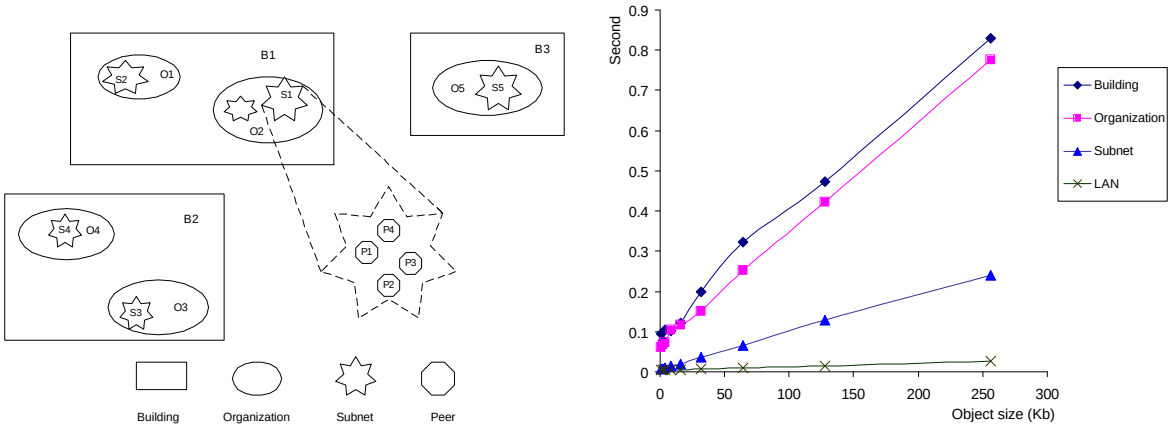


Figure 1: A three dimension design space of peer-to-peer Web caching.

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.



(a) Peer granularity (b) Four levels latency model

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

In the *home2* algorithm, each requested document is associated with a certain host as its logical home (based on its 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 [5]. 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.

Granularity	a	b
LAN	0.0001	0.0042
Subnet	0.0009	0.0066
Organization	0.0028	0.0671
Building	0.0029	0.0920

Table 2: The coefficients of the simple latency estimation models using linear regression.

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 [1] 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, the corresponding parameters a and b are listed in Table 2.

3 Trace Generation

By examining many existing traces, we find that they are either (1) lack of the user-perceived latency which is one of performance metrics in our study; or (2) absence of the entire Web object which is required to calculate the content digest. We decide to collect the trace on our own. We collected all inbound Web traffic and rebuilt the trace. The inbound Web traffic means those Web sessions originated by the inside-campus clients and served by outside-campus Web servers.

3.1 Trace Collection

Tcpdump [22] was used to collect TCP packets at the network entrance of a middle-size education institution, while all Web traffic through port 80 is sniffed. To extract the complete HTTP information, including both header and content, we have developed WebTACT, a **W**eb **T**raffic **A**nalysis and **C**haracterize **T**ool [26]. The output of WebTACT includes the hash digest values for requested Web content, and the user-perceived latency, which is measured as the difference between the capture time of last packet of

response and that of first packet of request. The TTL (time-to-live) value associated with each Web document is calculated using Squid's [19] implementation.

From the viewpoint of Web caching, generally Web content could be categorized as uncacheable Web object or cacheable Web object. The cacheable Web content refers to those infrequently changed Web objects (also known as static Web content). The uncacheable Web content could be further subcategorized into seven uncacheable types, depends on the following rules:

- **Type 1** - `NonGet`: If the HTTP method, appeared in the HTTP request header, is not a `GET` method, then the corresponding HTTP object would be classified as `NonGet` subtype;
- **Type 2** - `DynGen`: If the method is `GET`, and the request URL contains keywords (like "cgi", "asp", "=", and "?", ...etc.), which implies the HTTP response object is probably generated dynamically, then that object would be classified as `DynGen` subtype;
- **Type 3** - `Pragma`: In the cases that HTTP request/response header part contains "Pragma: no cache" control information header, this object could be considered as `Pragma` subtype;
- **Type 4** - `CacheCtl`: In the case that HTTP request/response "Cache Control" header contains information indicating this is a dynamic, uncacheable HTTP object, this HTTP object is classified as `CacheCtl` subtype;
- **Type 5** - `Personalized`: If the HTTP request header contains `Cookie` or Authorization-related headers,or the HTTP response contains `Set-Cookie` header, the corresponding HTTP content is defined as `personalized` subtype;
- **Type 6** - `AbnormalStatus`: If the return status code, from server, does not belong to 2XX or 3XX, we think the response object is not a cacheable response and treat it as of `AbnormalStatus` subtype;
- **Type 7** - `ZeroTTL`: Except above six subtypes, we are also interested in the HTTP objects whose TTL (time-to-live) value equals to zero. This sort of objects is classified as `ZeroTTL` subtype.

3.2 Host Traffic Clustering

To cluster the inbound traffic at different peer granularities, we obtained the network topology information from Computing& Information Technology (C&IT) division of the education institution . Based on this, we could identify the relationship between any two internal IP addresses (two clients), and calculate the simulation latency using the latency estimation model as described above. Note that if two users use the same machine, we have to consider them as one peer in *host-level* caching.

4 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 granulairties. We collected seven-day period (Aug 25, 2003 — Aug 31, 2003) Web traffic.

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.

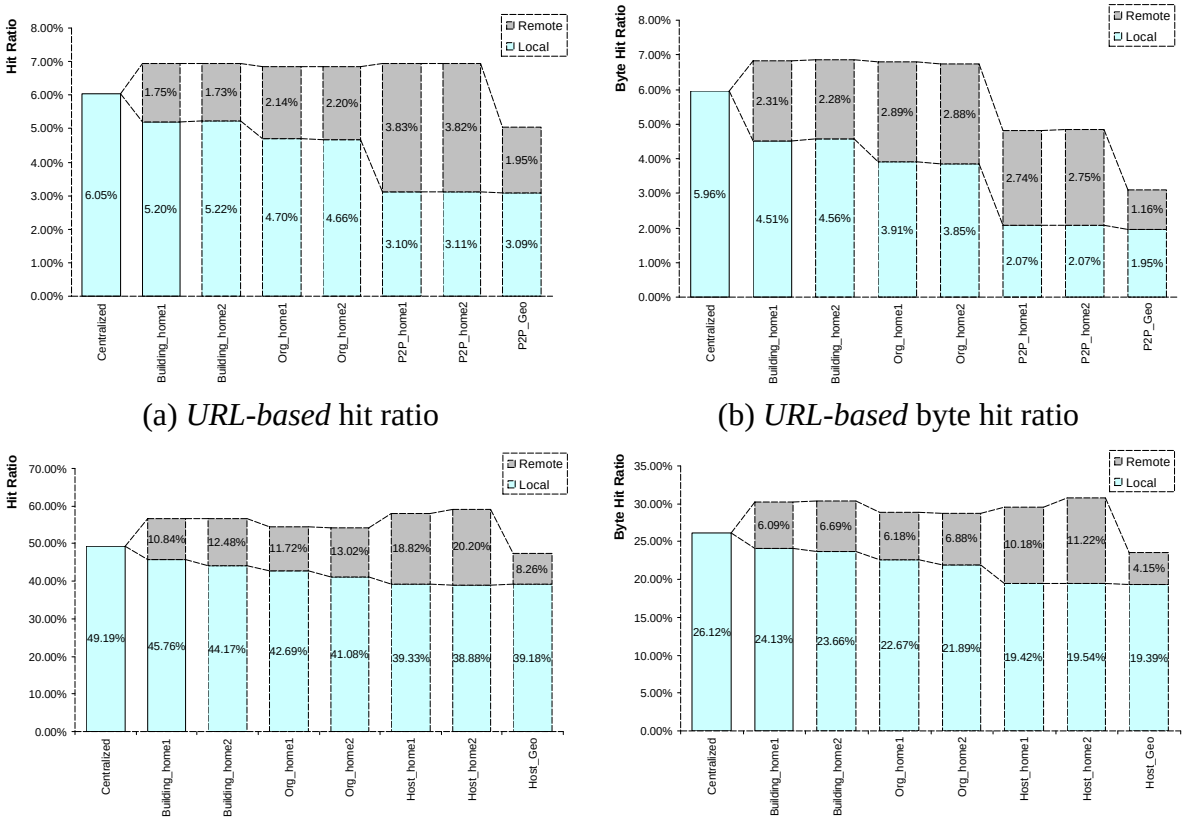
4.1 Performance Metrics

Although most previous studies chose performance metrics like *hit ratio* and *byte hit ratio* to evaluate Web caching, from the perspective of clients, the user-perceived latency is the most crucial. In this study, we focus not only on *hit ratio* and *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.

4.2 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 *URL-based* and *digest-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 *digest-based* caching algorithm, we only simulate the uncacheable Web content, while this algorithm works for the static Web content as well. Thus, the total *hit ratio* or *byte hit ratio* of the *digest-based* caching algorithm is the sum of that from *digest-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 *digest-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 *digest-based* caching algorithms due to the sum of peers cache size increasing. In general, *home1* has a big hit ratio than *home2*. The reason is *home2* algorithm will store the web object at request host and also request home, this will cause the disk space redundancy to decrease the hit ratio. We will discuss the *hit ratio* and *byte hit ratio* based on caching algorithms, document lookup algorithms and peer granularities separately.



(a) *URL-based* hit ratio (b) *URL-based* byte hit ratio

(c) *Digest-based* hit ratio (d) *Digest-based* byte hit ratio

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

4.2.1 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 *URL-based* algorithm has the value from 5.04% to 6.94%, while *digest-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 *URL-based* caching algorithm is that there are 49.6% requests whose TTL's values are zero, probably caused by the cache busting [11] technique, and therefore those requests are uncachable for the traditional caching algorithm. These results indicate that the *digest-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 *dynamic-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 *hit ratio* using *digest-based* caching algorithms. The possible reasons are: (1) the cache busting [11] 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.

	URL-based		Digest-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 3: The peer share gain of two caching algorithms at three peer granularities.

4.2.2 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 difference of *hit ratio* for *home1*, *home2* lookup algorithm, as shown in Figure 3, is caused by two possible reasons: (1) the space limitation of cache size; (2) the disk redundancy of *home2* algorithm. Compare with *home1*, *home2* will store the web objects at two locations, one is at the request host, one is at the request logic home. 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 *Geo* algorithm. Although the *Geo* algorithm only has two third of the *hit ratio* compared with the *home1* and the *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.

4.2.3 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 *centralized level* to *host level*. The increment of *hit ratio* is caused by the cache capacity increasing with the changing of peer granularity. Figure 3 shows that the *hit ratio* and *byte hit ratio* increases with peer granularity changing from *centralized level* to *host level*. But there are some exceptions for *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 *organization level* and *host level* is 20MB and 2MB respectively, and there exist some files that are too large to be cached. For the exception of the *digest-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 *organization level* cache is 7,700MB, which is less than the capacity sum of *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*.

Dynamic type		NonGet	DynGen	Pragma	CacheCtl	Personalized	AbnormalStatus	ZeroTTL
Hit ratio	Centralized	0.17%	11.94%	0.33%	1.36%	0.06%	4.93%	24.66%
	Building	0.21%	12.48%	0.35%	1.65%	0.07%	5.11%	30.63%
	Organization	0.20%	12.28%	0.33%	1.57%	0.06%	5.05%	28.94%
	Host	0.23%	13.44%	0.54%	1.65%	0.07%	4.56%	31.42%
	Host-Geo	0.21%	12.12%	0.37%	1.39%	0.06%	3.96%	23.09%
Byte hit ratio	Centralized	0.11%	4.35%	1.83%	0.78%	0.22%	0.61%	16.35%
	Building	0.12%	4.91%	1.92%	0.91%	0.23%	0.62%	19.52%
	Organization	0.11%	4.73%	1.87%	0.86%	0.23%	0.63%	18.47%
	Host	0.11%	5.06%	1.94%	0.92%	0.10%	0.58%	19.17%
	Host-Geo	0.11%	4.23%	1.70%	0.76%	0.10%	0.54%	14.49%

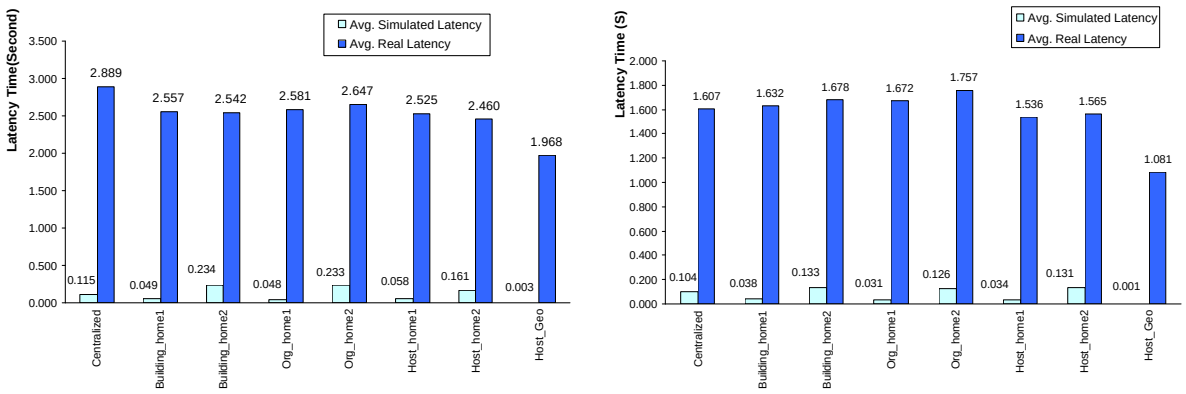
Table 4: The dynamic types hit ratio and byte hit ratio of four peer granularities.

4.2.4 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 3 shows the *peer share gain* in terms of both request hit and byte hit based on different peer granularities and two caching algorithms. From Table 3, we can find that *host level* caching has the highest *peer share gain* for both *hit ratio* and *byte hit ratio*, for two caching algorithms, *URL-based* and *digest-based*. Table 3 also shows that *building* and *organization level* have around 20% sharing gain for *hit ratio*, and 20% for *byte hit ratio* when applying *digest-based* caching algorithm. This observation implies 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 *digest-based* caching algorithm actually reduce the peer share gain in terms of both number of requests and number of bytes. The reason is the *digest-based* caching algorithm will caused more cache replacement due to the cache size limit.

4.2.5 Dynamic Type

Although the *digest-based* algorithm has the great potential for content reusing, we are more interested in where this benefit comes from. Therefore, we analyze the *hit ratio* for dynamic types in detail and exploit which type of dynamic content could be efficiently cached or has higher *hit ratio* in our simulation. Table 4 shows the *hit ratio* for seven dynamic types at different peer granularities. Table 4 indicates that there is no significant difference of *hit ratio* and *byte hit ratio* for the four level peer granularities. `ZeroTLL` contributes about 50% *hit ratio* of total digest hit. `DynGen` contributes about 15% *hit ratio* of total digest hit. These results imply that for *digest-based* caching algorithm we need to focus on `DynGen` and `ZeroTTL` types. We also evaluate the corresponding *latency reduction* for these two types, and we found that the latency improvement for `DynGen` is 93%, for `ZeroTTL` is about 84%. We think these improvements are acceptable.



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

Figure 4: The average latency of simulation and measurement.

4.3 Latency Reduction

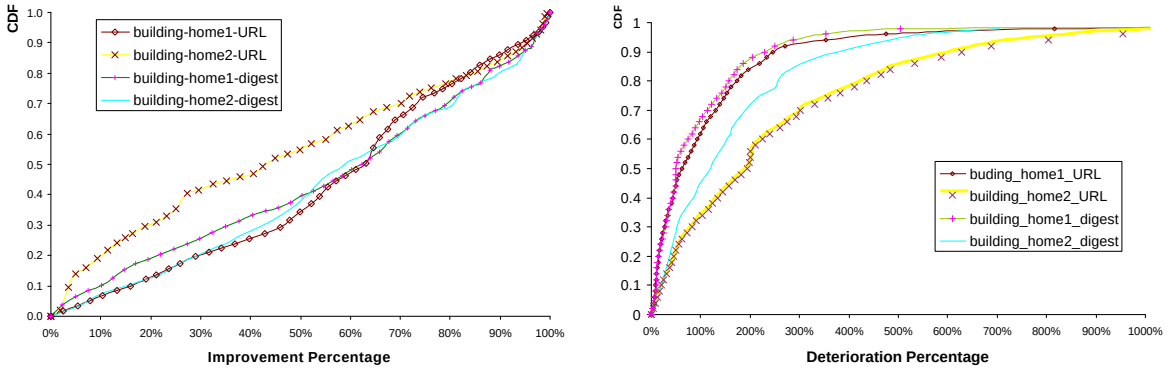
Now we are in the position to examine the corresponding *latency reduction* resulting from peer-to-peer Web caching. We use the user-perceived last-byte latency as a performance metric to examine all possible caching design space. Figure 4 illustrates the average latency obtained from simulation and 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 *digest-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 5 shows the 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 host-Geo scenario. This abnormality indicates that we need examine $L_{improve}$ and $L_{deteriorate}$ in more detail. Figure 5 shows the cumulative distribution function (CDF) of $L_{improve}$ and $L_{deteriorate}$ in *building level*.[1] This figure could be used to explain the significant improvement shown in Figure 4, that the average latency reduces significantly while the difference of $L_{improve}$ and $L_{deteriorate}$ is very low. For example, in Figure 5(a) the $L_{improve}$ is uniformly distributed, while in the Figure 5(b) $L_{deteriorate}$ is likely exponentially distributed. Furthermore, in *building level* applying the *URL-based* caching algorithm using the *home1* lookup algorithm, the average improved time of $L_{improve}$ is 1.412 second 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.

[1]We also examined the CDFs of $L_{improve}$ and $L_{deteriorate}$ in *host level*, *organization level* and *centralized level*, and got the same patterns.

Caching algorithm	URL-based		Digest-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 5: The latency improvement and deterioration of two caching algorithms at four peer granularities.



(a) The CDF of latency improvement (b) The CDF of latency deterioration

Figure 5: The latency improvement and deterioration of different algorithms in *building level* peer-to-peer Web caching.

Next, we will discuss the *latency reduction* in terms of document lookup algorithms, P2P granularities and caching algorithms.

4.3.1 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, the *home1* is observed to be superior to the *home2* in both average simulation latency and $L_{improve}$. This can be explained logically as *home2* needs more routing steps to locate the Web object's home as compared to *home1*.

4.3.2 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*) under the *digest-based* caching algorithm.

4.3.3 Caching Algorithm

Figure 4 (a) and (b) also show that the *digest-based* caching algorithm has a better *latency reduction* than the *URL-based* caching algorithm. The 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, if a *digest-based* caching algorithm was implemented, a possible latency overhead for the *digest-based* caching algorithm is expected; (2) the average size of dynamic Web objects is smaller than static Web objects. The gain from the *digest-based* algorithm is mostly come from the reuse of dynamic Web content, as such, the advertised gif or jpeg images; (3) the dynamic Web objects tend to have longer measured latencies caused by dynamic generation. When these objects are hit during the simulation, these dynamic generated latency will be reduced.

5 Implications

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

- **Need protocol support for deploying the *digest-based* Web caching mechanism**: The results of the experiment show that the *digest-based* caching algorithm improves the *hit ratio* tremendously. For example, in Figure 3, the *hit ratio* increased from 6.9% for the *URL-based* caching algorithm to 62.0% for the *digest-based* caching algorithm at *building level* granularity. However, in our current 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 situation. Thus, to exploit the benefit of the *digest-based* caching algorithm, an efficient mechanism is required to obtain the content digest. DTD [10] and VBWC [15] are two recent effort to exploit this.

- **Tradeoff between *latency reduction* and scalability**: The *latency reduction* is a key performance metric in our study. Figure 4 shows that the *home1* document lookup algorithm is always superior than *home2* algorithm. In the real implementation, the *home1* algorithm needs a centralized index server, which is a big obstacle of scalability for a large client population. Although the *home2* algorithm is exempted from the scalability problem, it has a less *latency reduction* than the *home1* counterpart, because of the extra overhead of P2P routing. Therefore, we argue that peer-to-peer Web caching at organization or *building level* using the *home1* lookup algorithm is a good choice. In the real deployment of the *home2* algorithm, the P2P routing is implemented at the application level which is not good for the *latency reduction*. Intuitively, we think if some geographically-aware clustering technologies [11, 14] were applied on the *home2* algorithm, the *latency reduction* could be improved, but further study is required.

- **Exploiting the *geographic-based* lookup algorithm**: We propose a simple and effective *geographic-based* document lookup algorithm. The results show that it has an acceptable *hit ratio* and a significant *latency reduction*. The reason of the relative lower *hit ratio* compared with the other two algorithms is due to the limited host population participating in the *geographic-based* cluster. We

believe that if some geographically-aware clustering technologies, such as network clustering [11], or landmark based binning algorithm [14], are integrated with the *geographic-based* lookup algorithm to increase the client population, the *hit ratio* will increase greatly, while the advantage of the *latency reduction* still remains good.

- **Potential of caching dynamic types**: We examine the *hit ratio* of dynamic content based on dynamic types. The purpose of exploiting dynamic types is to examine the cacheability of different dynamic types according to our classification technique. In our simulation, we assume that all those seven types of dynamic content could be cached based on their content digest. Intriguingly, experiment results in Table 4 show that only `DynGen`, `AbnormalStatus` and `ZeroTTL` dynamic content have an acceptable *hit ratio*. `AbnormalStatus` represents abnormal response status, definitely not suitable for caching. (`DynGen`) and (`ZeroTTL`) contribute about 75% of the *hit ratio* in the *digest-based* caching algorithm. Caching those two dynamic types will significantly improve the Web caching performance and save tremendous network bandwidth.

- **Which peer granularity is the best?**: From the perspective of *peer share gain*, the experiments suggest the lower the peer granularity the larger the *peer share gain*, which advocates the *host level* peer-to-peer Web caching. But from the perspective of the *latency reduction*, which is the most crucial consideration for clients, the *organization level* or *building level* peer-to-peer Web caching using the *home1* algorithm is the best one. If there are some improvements at lookup algorithm to reduce the latency caused by P2P routing, the *host level* P2P Web caching using the *home2* algorithm will be a good choice too.

6 Related Work and Discussions

Peer-to-peer Web caching (also known as cooperative Web caching) has been extensively studied in recent years [9, 24, 23, 18, 20]. The work presented in this paper is inspired by the controversial observations drawn from these earlier efforts. 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.* in the context of memory caching sharing in file system [7], which examined and compared four different cooperative caching algorithms using a trace-driven simulation. 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, using the traces from University of Washington and Microsoft Research Redmond [23]. 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, which focus on the *organization level* only. Second, the qualitative latency improvement analysis in [23] was done by an analytical model, while we perform a quantitatively study. Finally, a new *digest-based* caching algorithm is proposed in this paper, distinguishing our work from their URL-based analysis.

Recently, Iyer *et al.* proposed Squirrel [9], a peer-to-peer Web caching system built on the PASTRY peer-to-peer routing substrate [16]. Xiao *et al.* studied the reliability and scalability of a browser-aware proxy server by using a centralized index server for multiple hosts. Our work was partially inspired by these two previous efforts, and 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 reduction*

as well. Furthermore, the traces used in our analysis was collected on March 2003, which is more up-to-date than the traces used in [9, 24]. Our recent work on Tuxedo [18] proposed another object lookup protocol, called adaptive neighbor table, which compliments to this work. Backslash [20] is a content distribution system based on peer-to-peer overlay and used for those who do not expect consistently heavy traffic (flash crowds) to their sites. Although it is also a peer-to-peer Web caching system, the goal is totally different from ours.

The *digest-based* caching algorithm proposed in this paper is motivated by the fact that there exist a large amount of content repeatness in Web traffic, i.e., the content of two Web documents are same even though their URLs are different. This phenomenon was observed in our recent traffic analysis [26] and [10]. The recent proposed value-based Web caching (VBWC) by Rhea *et al.* [15] shares the similar idea as ours, but we come out this idea independently. Moreover, the focus of VBWC is implementation details (block-level) in the last mile, while our work is examining the potential benefits by using digest as a general Web caching approach. Their implementation compliments to our effort. The work of Bahn *et al.* [2] focuses on reducing the repeatness of web object on disk by using content digest, we are interested in peer-to-peer sharing of Web content.

Peer lookup algorithm is a very hot research topic in recent years, Chord [21], CAN [13], Pastry [16] are three representatives. 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 in terms of hit ratio, and reduce the latency significantly. Theoretically, we believe that our work will definitely benefit from several recent work on geographically-aware clustering technologies, such as network clustering [11], and landmark based binning algorithm [14], global network positioning (GNP) service [12]. However, it is still an open problem to understand how much benefits can be obtained by employing these complicated algorithms. This will be our future work. Recently, Canali *et al.* evaluated the performance of two lookup algorithms, hierarchical and flat, in terms of transcoded version among cooperative caching [4]. Different from their effort, we examine the whole design space of the peer-to-peer Web caching in the paper.

7 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 *digest-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 the *home1* algorithm is the most appropriate choice. Our trace is available for research purpose at `http://mist.cs.wayne.edu`.

References

[1] Apache HTTP Server Project, `http://httpd.apache.org`.

[2] H. Bahn, H. Lee, S. H. Noh, S. L. Min, and K. Koh. Replica-aware caching for web proxies. *Computer Communications* 25(3):183–188, Feb 2002.

[3] P. Barford, A. Bestavros, A. Bradley, and M. E. Crovella. Changes in web client access patterns: Characteristics and caching implications. *World Wide Web, Special Issue on Characterization and Performance Evaluation* 2:15–28, 1999, `http://www.cs.bu.edu/techreports/1998-023-web-client-trace-changes-and-%implications.ps.Z`.

[4] 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.

[5] 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.

[6] 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, `http://www.cs.bu.edu/techreports/pdf/1995-010-www-client-traces.pdf`.

[7] 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.

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

[9] 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.

[10] 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.

[11] B. Krishnamurthy and J. Rexford. *Web Protocols and Practice: HTTP/1.1, Networking Protocols, Caching and Traffic Measurement*. Addison-Wesley, Inc, 2001.

[12] 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.

[13] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker. A scalable content addressable network. *Proc. of ACM SIGCOMM'01*, 2001.

[14] 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.

[15] 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.

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

[17] W. Shi, E. Collins, and V. Karamcheti. DYCE: A synthetic dynamic web content emulator. *Poster Proc. of 11th International World Wide Web Conference*, May 2002, `http://www.cs.wayne.eud/~weisong/papers/dyce.pdf`.

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

[19] Squid web cache, `http://www.squid-cache.com/`.

[20] T. Stading, P. Maniatis, and M. Baker. Peer-to-peer caching schemes to address flash crowds. *Proc. of the 1st International Workshop on Peer-to-Peer Systems (IPTPS'02)*, Feb. 2002.

[21] 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.

[22] Tcpdump, `http://www.tcpdump.org`.

[23] 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.

[24] 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.

[25] 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.

[26] Z. Zhu, Y. Mao, and W. Shi. Workload characterization of uncacheable web content — and its implications for caching. Tech. Rep. CS-MIST-TR-2003-003, Department of Computer Science, Wayne State University, May 2003.