Performance evaluation of peer-to-peer Web caching systems

Weisong Shi *, Yonggen Mao

Mobile and Internet Systems Group, Department of Computer Science, Wayne State University, Detroit, MI 48202, USA

Received 25 February 2005; received in revised form 13 August 2005; accepted 18 August 2005
Available online 27 September 2005

Abstract

Peer-to-peer Web caching has attracted a great attention from the research community recently, and is one of the potential peer-to-peer applications. In this paper, we systematically examine the three orthogonal dimensions to design a peer-to-peer Web caching system, including the caching algorithm, the document lookup algorithm, and the peer granularity. In addition to the traditional URL-based caching algorithm, we also evaluate the content-based caching algorithm for both dynamic and static Web content. Four different document lookup algorithms are compared and evaluated in the context of four different peer granularities, i.e., host level, organization level, building level, and centralized. A detailed simulation, using the traces collected at a medium size education institution, is conducted for the purpose of performance evaluation. Finally, several implications derived from this analysis are also discussed.

© 2005 Elsevier Inc. All rights reserved.

Keywords: Peer-to-peer systems; Web caching; Analysis model; Latency reduction; Hit ratio; Performance evaluation

1. Introduction

Peer-to-peer networking has become a hot research topic recently (Androustelis-Theotokis and Spinellis, 2004; Ratnasamy et al., 2001; Rowstron and Druschel, 2001; Stoica et al., 2001; Zhao et al., 2001). 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 (Iyer et al., 2002; Xiao et al., 2002). Xiao et al. (2002) proposed a browser-aware proxy server model and evaluated using BU-95 trace (Cunha et al., 1995) collected from Boston University (1995) and NLANR-uc trace (2000) (IRCache Project, 1995). In Squirrel, Iyer et al. (2002) presented a peer-to-peer Web caching system built on top of the PASTRY (Rowstron and Druschel, 2001), and evaluated using the traces of Microsoft Research Redmond campus (1999) (Wolman et al., 1999b) and Cambridge campus (2001) respectively. Although these two studies showed optimistic results for peer-to-peer Web caching, the study of Wolman et al. (1999b) 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 the hit ratio, but those studies did not quantitatively evaluate the effect of latency improvement.

On the other hand, recent studies (Barford et al., 1999; Shi et al., 2003a) show 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 studies (Kelly and Mogul, 2002; Zhu et al., 2003) show that the dynamic objects have a large portion of repeatness based on their content digest. This repeatability provides an opportunity to improve the cacheability, and motivates us to propose a content-based caching algorithm for peer-to-peer Web caching.

* Corresponding author. Tel.: +1 313 577 3186; fax: +1 313 577 6868. E-mail address: weisong@wayne.edu (W. Shi).
In this paper, we intend to evaluate the performance of peer-to-peer Web caching systems and to examine the future research directions of peer-to-peer Web caching. We first develop an analysis model to calculate the speedup of the peer-to-peer caching system based on Dykes and Robbins’s work (2002). Then, we 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 a content-based caching algorithm which exploits the fact of the large repetativeness of Web objects even though their URLs are different. In addition to comparing three 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 educational institution. Using the trace-driven simulation, we compared and evaluated all the design choices in terms of three performance metrics: hit ratio, latency reduction, and speedup. 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 content-based caching algorithm could improve the cacheability of Web objects substantially, increasing from 7.0% (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 homel (Xiao et al., 2002) and the Tuxedo algorithm (Shi et al., 2003b); (4) the Tuxedo algorithm at the host level has the highest speedup comparing with all combinations; (5) 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 content-based Web caching mechanism to improve the performance of content delivery on the Internet; (2) the organization or building level peer-to-peer Web caching using the Tuxedo document lookup algorithm 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) among the seven types of dynamic Web content (see definitions in Section 3.1), dynamic type2 (dynamically generated) and type7 (zero value of time-to-live) are the most promising to benefit from the content-based caching algorithm.

Our contributions of this study include: (1) developing an analysis model to evaluate peer-to-peer Web caching systems; (2) systematically examining the design space of peer-to-peer Web caching; (3) 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 digest; (4) comprehensive evaluating the performance of Web caching in terms of three performance metrics; (5) 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, and describes the algorithms used in the evaluation. Section 3 describes the trace data collection, and classification of multiple dynamic content. A comprehensive performance evaluation and comparison of different algorithms in terms of three 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 Sections 6 and 7, respectively.

2. Design space of peer-to-peer Web caching

As illustrated in Fig. 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 (Ratnasamy et al., 2001; Rowstron and Druschel, 2001; Stoica et al., 2001) 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 (2005), Gnutella (2000), and KaZaA (2005) 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 algorithms

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 a static Web object and its related freshness time, and has been widely used in Web caching. The content-based caching algorithm is inspired by recent studies (Kelly and Mogul, 2002; Zhu et al., 2003). In Zhu et al. (2003), we found that the static Web content occupies only 10.2% of total Web requests, and 59.1% Web requests, which are repeatedly accessed (based on their content 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 to exchange a content digest to decide whether or not a real content communication should happen.

2.2. Document lookup algorithms

As in any P2P networking system, the document lookup algorithm is the core of the whole design. Four lookup algorithms are evaluated in this paper, namely home1, home2, geographic-based (Geo in short) and Tuxedo 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 scope. This protocol is used in Xiao et al.’s work (2002). 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 again in the index server, the request host will go to the original server directly.

In the home2 algorithm (Iyer et al., 2002), 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 a miss occurs, it will employ the 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 a miss occurs again, the home will send the request to the original server and then forward the received Web object to the request host. In addition to using an average latency estimation, we also implement the home2 algorithm at the host level based on the FreePastry (FreePastry, 2003) package to evaluate the effect of P2P routing overhead.

The Tuxedo algorithm was proposed by Shi et al. (2003b). The novelty of this approach includes twofold: maintaining a specific neighbor set for each Web server and using a hierarchical cache digest approach to store multiple transcoded versions of the same content. Each Tuxedo node maintains a server table and a neighbor table as shown in Fig. 2. The server table keeps two pieces of information: (1) an integer number which indicates how many peers are closer than the origin Web server in the neighbor table; and (2) the original server information (latency/bandwidth). The neighbor table contains a hierarchical cache digest information for each peer and is ordered by the latency. In this paper, we are more interested in the benefit of performance improvement of the algorithm.

The Geo algorithm comes from our intuition that people will have similar Web-browsing interests at the same geographic location. Currently, only the 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, an application-level multicast can be used here too (Castro et al., 2003b). When a local cache missing happens, the client first multicasts 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

Fig. 3 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 the performance evaluation, we implement three decentralized peer-to-peer Web caching systems at different levels, including host, organization, and building level. For comparison purpose, a centralized Web caching is also implemented. On the Internet, the peer granularity can be larger than what we studied here, such as across institutions, however, previous result (Wolman et al., 1999a) shows that sharing across big organizations is limit. Therefore, we will consider the peer-to-peer sharing within a big organization only in this paper.

| Table 1 |
| Description of four document lookup algorithms |
| Algorithms | Description |
| home1 | A hierarchical index server is used to maintain Web content in peer’s cache (Xiao et al., 2002) |
| home2 | A decentralized index (using hash value) is used to locate Web content in each peer (Iyer et al., 2002) |
| Geo | Only hosts located in the same subnet are considered. |
| Tuxedo | An adaptive neighborhood mechanism is maintained for different Web server and a hierarchical cache digest is used to locate Web content (Shi et al., 2003b) |
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 Fig. 3 (a), there are four possible host-to-host latency models: hosts within the same LAN (in the reach of the same switch); hosts in different 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 1 to 256KB, against an Apache Web server (Apache HTTP Server Project, 1999) inside campus. We calculated the latency between the request and the last-byte of response as in Fig. 3 (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 kilobyte, \( f(x) \) represents the latency in second, the corresponding parameters \( a \) and \( b \) are listed in Table 2.

![Fig. 2. Two tables maintained at each Tuxedo cache node: server table and neighbor table.](image)

![Fig. 3. (a) Peer granularity and (b) the simple latency estimation model.](image)

### Table 2

<table>
<thead>
<tr>
<th>granularity</th>
<th>( a )</th>
<th>( b )</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAN</td>
<td>0.00012</td>
<td>0.0042</td>
</tr>
<tr>
<td>Subnet</td>
<td>0.00009</td>
<td>0.0066</td>
</tr>
<tr>
<td>Organization</td>
<td>0.0028</td>
<td>0.0671</td>
</tr>
<tr>
<td>Building</td>
<td>0.0029</td>
<td>0.0920</td>
</tr>
</tbody>
</table>
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 (tcpdump, 2001) is 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 Web Traffic Analysis and Characterize Tool (Zhu et al., 2003). 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 (squid, 1998) 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, depending 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 AbnormalStatus subtype;
- **Type 7**—ZeroTTL: Except above six subtypes, we are also interested in the HTTP objects whose TTL (time-to-live) values equal 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 the host-level caching.

4. Performance evaluation and analysis

We adopt the trace-driven approach to examine the different design choices of peer-to-peer Web caching, implementing two caching algorithms, and four document lookup methods at four peer granularities. The trace used in the performance evaluation was collected at a medium size education institution during a seven-day period (August 25–31, 2003). Only the inbound traffic is examined in the analysis.

Totally, there are 10,481 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 cache sizes of real deployment could be set much bigger, we are interested in the relative relationship (relative size ratio) among caches at different levels. The least-recent used (LRU) replacement algorithm is used in our simulation.

4.1. 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 the most 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 to the absolute latency reduction, we also use speedup, a relative comparison, to evaluate different Web caching systems.
In our previous work (Mao et al., 2004), the P2P routing overhead of the home2 algorithm is estimated based on average latency (calculated as the average number of hops times the average latency per application layer hop). In this paper, we also implement a variant of the home2 algorithm by using the FreePastry (FreePastry, 2003) package at the host level. Our results in Section 4.3 show that the FreePastry introduces more latency than our estimation in (Mao et al., 2004). Moreover, 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 \( L_{\text{improve}} \) and \( L_{\text{deteriorate}} \) to depict these two cases, respectively.

4.2. Hit ratio

In terms of the hit ratio, including both request hit ratio and the byte hit ratio, we examine different design choices. Fig. 4 shows that the hit ratio and the 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 Fig. 4(a) and (c) indicates the hit ratio in percentage. The Y-axis of Fig. 4(b) and (d) shows the byte hit ratio in percentage. In Fig. 4, 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 (the home1 and the home2), or neighbor within the same subnet (the Geo). For the centralized cache, the remote hit is zero.

Note that, for the content-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 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 Fig. 4(a) and (b), has the lower cache hits in terms of the hit ratio and the byte hit ratio compared with the content-based caching algorithm as shown in Fig. 4(c) and (d). From those figures, we observe that the local hit ratio decreases from the centralized level to the host level caused by the total cache size.

![Fig. 4. Comparison of hit ratios and byte hit ratios of different algorithms: (a) URL-based hit ratio, (b) URL-based byte hit ratio, (c) Content-based hit ratio and (d) Content-based byte hit ratio.](image-url)
decreasing at each level, and the remote hit ratio increases respectively for both the URL-based and the content-based caching algorithms due to the sum of peers cache size increasing. In general, the home1 has a big hit ratio than the home2, and the Tuxedo has less hit ratio than both of them. The reason is the home2 algorithm will store the Web objects at both the requesting host and the home of the requested object, which causes the disk space redundancy to decrease the hit ratio. The lower hit ratio of the Tuxedo is due to the low document lookup efficiency and the adaptive neighbor set algorithm, as explained in Section 4.2.2. Next, we will discuss the hit ratio and byte hit ratio based on caching algorithms, document lookup algorithms and peer granularities separately.

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 Fig. 4(a) and (c), we find that the hit ratio of the URL-based algorithm has the value from 5.04% to 7.03%, 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 there are 49.6% requests whose TTL’s values are zero, probably caused by the cache busting (Krishnamurthy and Rexford, 2001) technique, and therefore those requests are uncachable for the traditional caching algorithm. These results indicate that the content-based algorithm has the great potential to increase the hit ratio. Fig. 4(b) and (d) report that the byte hit ratio of the URL-based caching algorithm is from 3.12% to 6.80%, while 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 hit ratio using content-based caching algorithms. The possible reasons are: (1) the cache busting (Krishnamurthy and Rexford, 2001) technique tends to apply to 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.

4.2.2. Document lookup algorithm

Logically, the hit ratio resulting from a document lookup algorithm is determined by the scope of lookup and the efficiency of the document lookup algorithm, independent of the specific document lookup algorithm. The difference of hit ratio for the home1, the home2 lookup algorithm, as shown in Fig. 4, is caused by two possible reasons: (1) the space limitation of cache size, and (2) the disk redundancy of the home2 algorithm. Comparing with the home1, the home2 will store the Web objects 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. On the other hand, the low hit ratio of the Tuxedo (especially for the remote hit ratio) is not caused by the space limitation, instead is the result of the specific algorithm of Tuxedo. The Tuxedo algorithm uses cache digests to predict the Web content among its peers. The efficiency of prediction depends on the size of bloom filter (for cache digests) and the frequency of digest updating. Experiment shows that 30 min updating interval will achieve 90% lookup efficiency, while 1 h updating interval will get only 87% using the same trace. Another factor to decrease the Tuxedo hit ratio is the adaptive neighbor set algorithm which intends to access the origin Web server if the cache peers is slower than the origin Web server; however, we find that the loss of the hit ratio will be compensated by the benefit of latency reduction or speedup. Interestingly, it can be seen from the figures that the hit ratio of the geographic-based algorithm is lower than that of three 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 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 with the home1 and the home2 algorithms. Therefore, 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 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. Fig. 4 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 20 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 7700MB, 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.

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 the host level caching has the highest peer share gain for both hit ratio and byte hit ratio, for two caching algorithms, URL-based and content-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 content-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 content-based caching algorithm actually reduce the peer share gain when the cache size limit. It is worth noting that the peer share gains of the FreePastry implementation always have the highest values in Table 3, which agrees with the guarantee of document lookup of structured peer-to-peer systems; however, as we will see in Section 4.3, most of the hit achieved by the FreePastry implementation actually deteriorate the user-perceived latency.

4.2.5. Effect of dynamic type

Although the content-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 and byte hit ratio of different dynamic types at four 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 content-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 find that the latency improvement for DynGen is 93% and for ZeroTTL is about 84%. We think these improvements are acceptable.
4.3. Latency reduction and speedup

Now we are in the position to examine the corresponding latency reduction and speedup 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. In this section, instead of examining the latency as a performance metric to examine all possible caching algorithms, we evaluate the URL-based algorithm only, because the overhead of the content-based depends on specific implementation details. We first evaluate the speedup of the different algorithms at three peer granularities, i.e., host level, organization level, and building level. The speedup is defined as the ratio of \( T_{\text{without}} \) and \( T_{\text{with}} \), where the \( T_{\text{without}} \) is the average response time of a client to retrieve a Web object from its local cache \( L \) without peer cache cooperation and the \( T_{\text{with}} \) is the average response time of a client to retrieve a Web object from its local cache \( L \) and with the cooperation of peer cache \( R \) cooperation.

Fig. 5 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 host-Tuxedo scenario has the best latency reduction except for Geo which has a less hit ratio. The FreePastry has very low latency reduction comparing with others. Our speedup analysis also shows it increases the user perceived latency in average. This indicates the home2 algorithm will not achieve good latency reduction as well.

Table 5 shows the percentage of latency improvement represented by \( L_{\text{improve}} \) and the latency deterioration represented by \( L_{\text{deteriorate}} \) for all hit requests in different peer-to-peer Web caching design options. \( L_{\text{improve}} \) represents the number of hit requests where the simulation latency is less than the measurement latency. \( L_{\text{deteriorate}} \) represents the difference between total number of hits and \( L_{\text{improve}} \). From the table, we observe that the Tuxedo, the Geo and the home1 have a higher \( L_{\text{improve}} \) ratio, but the Tuxedo has the highest \( L_{\text{improve}} \) number. The speedup analysis also support this conclusion.

Table 6 shows the speedup of different document lookup algorithms at three peer granularities. The speedup analysis needs to consider the lookup overhead and efficiency of each document lookup algorithm. The home1 algorithm has no lookup overhead, but the lookup efficiency is imperfect. In paper (Dykes and Rabbins, 2002), they suggest using nighttime update, and the lookup efficiency is 0.91.

---

Table 5

<table>
<thead>
<tr>
<th>Granularities</th>
<th>( L_{\text{improve}} )</th>
<th>( L_{\text{deteriorate}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Centralized</td>
<td>380,233 (55.1%)</td>
<td>309,608 (44.9%)</td>
</tr>
<tr>
<td>Building-home1</td>
<td>610,090 (82.7%)</td>
<td>127,454 (17.3%)</td>
</tr>
<tr>
<td>Building-home2</td>
<td>219,136 (29.7%)</td>
<td>519,464 (70.3%)</td>
</tr>
<tr>
<td>Building-Tuxedo</td>
<td>603,355 (93.2%)</td>
<td>44,120 (6.8%)</td>
</tr>
<tr>
<td>Org-home1</td>
<td>622,764 (84.8%)</td>
<td>111,570 (15.2%)</td>
</tr>
<tr>
<td>Org-home2</td>
<td>251,161 (34.1%)</td>
<td>485,337 (65.9%)</td>
</tr>
<tr>
<td>Org-Tuxedo</td>
<td>609,839 (98.1%)</td>
<td>11,956 (1.9%)</td>
</tr>
<tr>
<td>Host-home1</td>
<td>577,786 (77.6%)</td>
<td>167,176 (22.4%)</td>
</tr>
<tr>
<td>Host-home2</td>
<td>434,263 (58.4%)</td>
<td>309,775 (41.6%)</td>
</tr>
<tr>
<td>Host-Geo</td>
<td>533,895 (98.7%)</td>
<td>7065 (1.3%)</td>
</tr>
<tr>
<td>Host-Tuxedo</td>
<td>613,946 (98.3%)</td>
<td>10,341 (1.7%)</td>
</tr>
<tr>
<td>FreePastry</td>
<td>329,333 (44.6%)</td>
<td>408,530 (55.4%)</td>
</tr>
</tbody>
</table>

Table 6

<table>
<thead>
<tr>
<th>Granularities</th>
<th>home1</th>
<th>home2</th>
<th>Tuxedo</th>
<th>Geo</th>
<th>FreePastry</th>
</tr>
</thead>
<tbody>
<tr>
<td>Building level</td>
<td>1.09</td>
<td>1.01</td>
<td>1.10</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Organization level</td>
<td>1.15</td>
<td>1.11</td>
<td>1.09</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Host level</td>
<td>1.37</td>
<td>1.10</td>
<td>1.38</td>
<td>1.23</td>
<td>0.74</td>
</tr>
</tbody>
</table>
The home2 algorithm has a perfect lookup efficiency, but has a considerable lookup overhead. In the home2 implementation, the average routing hops is used to simulate the lookup overhead of P2P routing. Comparing with the home2 implementation, the lookup overhead of FreePastry is much larger, which indicates that the FreePastry does not take the proximity into consideration efficiently, and always route requests to a peer far away from the requesting host. This is a typical phenomenon of most structured distributed hash tables (DHTs) that there is a mismatch between physical network and logic space caused by distributed hash tables. The speedup of FreePastry is 0.74, which means the Pastry based P2P Web caching indeed increases the user-perceived latency, although with a high hit ratio, as listed in Table 3. Therefore, we will not consider either home2 or FreePastry in later analysis. The lookup efficiency of the Tuxedo is perfect with 6KB cache digest for each peer at the host level. The update interval of cache digest is 30 min and the experiment shows that it could achieve 90% lookup efficiency. From Table 6, we observe that the host level implementation has the higher speedup than that of the implementations at the building and organization levels. The home1 and Tuxedo algorithms are faster than other algorithms at all peer granularities.

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

4.3.1. Documents lookup algorithm

Although the Geo has a significant latency reduction and a good speedup, the performance will be impaired by the multicast overhead of the network in the real implementation. From Fig. 5, we can see that home1 and Tuxedo algorithms have very high latency reduction and a large percentage (Limprove) as well. The speedup of the Tuxedo algorithm and the home1 algorithm at host level is 1.38 and 1.37, respectively. This shows Tuxedo and home1 could efficiently reduce the user perceived latency.

4.3.2. Peer granularity

The centralized level caching has a comparable latency reduction comparing 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, speedup, and Limprove compared with organization level cache. Although host level caches have higher speedup (e.g., 1.38 for the Tuxedo and 1.37 for the home1) than building level speedup (e.g., 1.10 for the Tuxedo and 1.09 for the home1), we prefer to the building level or the organization level implementation for ease management and less overhead of updating discovery information.

5. 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 tremendously. For example, in Fig. 4, the hit ratio increased from 6.9% for the URL-based caching algorithm to 62.0% for the content-based caching algorithm at building level granularity. However, in our current simulation, we assume that a client, which sends the 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 content-based caching algorithm, an efficient mechanism is required to obtain the content digest. DTD (Kelly and Mogul, 2002) and VBWC (Rhea et al., 2003) are two recent efforts to exploit this.

- Tradeoff between latency reduction and scalability: The latency reduction is a key performance metric in our study. Fig. 5 shows that the home1 and the Tuxedo document lookup algorithms are always outperforms the 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 will increase the user-perceived latency, because of the extra overhead of P2P routing. Therefore, we argue that peer-to-peer Web caching at organization or building levels using the Tuxedo lookup algorithm is a good choice. The experiment of FreePastry shows that the mismatch between the logical space exploited by Pastry and the physical space of peers does hurt the performance, although the proximity has been considered in Pastry’s design (Castro et al., 2003a). Proximity in peer-to-peer routing is still an active and open problem (Gummadi et al., 2003). Intuitively, we think if some geographically-aware clustering technologies (Krishnamurthy and Rexford, 2001; Ratnasamy et al., 2002) 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 (Krishnamurthy and Rexford, 2001), or landmark based binning algorithm (Ratnasamy et al., 2002), 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
6. Related work and discussions

Peer-to-peer Web caching (also known as cooperative Web caching) has been extensively studied in recent years (Iyer et al., 2002; Xiao et al., 2002; Wolman et al., 1999b; Shi et al., 2003b; Stading et al., 2002). 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 three performance metrics: hit ratio, latency reduction, and speedup.

Cooperative caching was first proposed by Dahlin et al. in the context of memory caching sharing in file system (Dahlin et al., 1994), 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 (Wolman et al., 1999b). 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 focuses on the organization level only. Second, the qualitative latency improvement analysis in (Wolman et al., 1999b) was done by an analytical model, while we perform a quantitatively study. Finally, a new content-based caching algorithm is proposed in this paper, distinguishing our work from their URL-based analysis.

Recently, Iyer et al. proposed Squirrel (Iyer et al., 2002), a peer-to-peer Web caching system built on the PASTRY peer-to-peer routing substrate (Rowstron and Druschel, 2001). 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 (Iyer et al., 2002; Xiao et al., 2002). Our recent work on Tuxedo (Shi et al., 2003b) proposed another object lookup protocol, called adaptive neighbor table, which compliments to this work. Backslash (Stading et al., 2002) 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.

Dykes and Robbin examined the benefits of cooperative proxy caching (Dykes and Rabbins, 2002); however, our work differs from theirs in threefold: (1) cooperative scope. We look at the peer-to-peer Web caching within an autonomous system, while their work apply for wide area caching; (2) design space. Their work looks at both hierarchical and mesh, while our work focuses on peer-to-peer (mesh) only; but to the best of knowledge, we are the first effort to systematically evaluate the design space of peer-to-peer Web caching systems in three dimensions; and (3) caching algorithm. Only the URL-based caching algorithm is evaluated in their work, we focus on both the URL-based and the content-based algorithms. The latter has a great potential for the future of Web caching. To our knowledge, we are the first to evaluate the performance of the content-based P2P caching using the real traces with content digests.

The content-based caching algorithm proposed in this paper is motivated by the fact that there exists a large amount of content repeatability in Web traffic, i.e., the content of two Web documents are the same even though their URLs are different. This phenomenon was observed in our recent traffic analysis (Zhu et al., 2003) and (Kelly and Mogul, 2002). The recent proposed value-based Web caching (VBWC) by Rhea et al. (2003) 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. (2002) focuses on reducing the repeatability of Web object on disk by using content digest, we are interested in peer-to-peer sharing of Web content.

Different from the content level Web sharing, at the semantic level, recently Zhuge et al. propose a semantic
link peer-to-peer (P2P) network specifies and manages semantic relationships between peers’ data schemas (Zhuge, 2003; Zhuge et al., 2005). These data schemas can be used as the semantic layer of a scalable Knowledge Grid, which considers not only nodes but also the XML structure in measuring the similarity between schemas to efficiently and accurately forward queries to relevant peers. Furthermore, it copes with semantic and structural heterogeneity and data inconsistency so that peers can exchange and translate heterogeneous information within a uniform view. We believe this work complements to our work very well by considering the high level semantics. Integrating the semantics of Web content in the Web caching is a promising direction and we plan to investigate it in our next step.

Peer lookup algorithm is a very hot research topic in recent years. Chord (Stoica et al., 2001), CAN (Ratnasamy et al., 2001), Pastry (Rowstron and Druschel, 2001) 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 (Krishnamurthy and Rexford, 2001), landmark based binning algorithm (Ratnasamy et al., 2002), and global network positioning (GNP) service (Ng and Zhang, 2002). 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 (Canali et al., 2003). Different from their effort, we examine the whole design space of the peer-to-peer Web caching in the paper.

7. Conclusions

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 cachability; 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 latency could be reduced three to six times compared with the measured latency; and the Tuxedo document lookup algorithm has a prominent hit ratio, a significant latency reduction, and speedup. Based on these observations, we argue that the organization/building level peer-to-peer Web caching using the Tuxedo algorithm is the most appropriate choice. Our trace is available for research purpose at http://mlst.cs.wayne.edu.

Acknowledgements

The authors would like to thank the anonymous reviewers for providing valuable comments to improve the presentation of this paper. The authors would also like to thank Zhaoming Zhu, for providing the Web trace, and his comments on the earlier version of this paper. This research was supported by the University Faculty Research Grant of Wayne State University.

References


Weisong Shi is an Assistant Professor of Computer Science at Wayne State University. He received his B.S. from Xidian University in 1995, and Ph.D. degree from the Chinese Academy of Sciences in 2000, both in Computer Engineering. His current research focuses on dynamic Web content delivery, trusted resource sharing in peer-to-peer systems, mobile computing, and wireless sensor networks. Dr. Shi has published more than 40 peer-reviewed journal and conference papers in these areas. He is the author of the book “Performance Optimization of Software Distributed Shared Memory Systems” (High Education Press, 2004). He has also served on technical program committees of several international conferences, including the chairs of poster track of WWW 2005 and WWW 2006. He is a recipient of Microsoft Fellowship in 1999, the President outstanding award of the Chinese Academy of Sciences in 2000, one of 100 outstanding Ph.D. dissertations (China) in 2002, “Faculty Research Award” of Wayne State University in 2004 and 2005, the “Best Paper Award” of ICWE’04 and IPDPS’05. He is a member of ACM, USENIX, and IEEE.

Yonggen Mao has completed his M.S. in computer science at Wayne State University on August 2004. His research focuses on Web caching and peer-to-peer systems. He received his B.E and M.E. degree from Huai University, Nanjing, China, in 1991 and 1994, respectively, both in Civil Engineering.