BuddyWeb: A P2P-based Collaborative Web Caching System (Position Paper)

XiaoYu Wang[1], WeeSiong Ng[2], BengChin Ooi[2], Kian-Lee Tan[2], AoYing Zhou[1]

[1] Department of Computer Science, Fudan University
{xiaoyuwang,ayzhou}@fudan.edu.cn
[2] Department of Computer Science, National University of Singapore
{ngws,tankl,ooibc}@comp.nus.edu.sg

1 Introduction

The peer-to-peer (P2P) computing model has been increasingly deployed for a wide variety of applications, including data mining, replica placement, resource trading, data management and file sharing (see [1]). In this paper, we look at yet another application - that of collaborative web caching. Unlike existing web caching techniques that are typically managed at the proxies, we look at how to exploit local caches of nodes (or rather PCs) within an enterprise network.

To illustrate, consider the campus network at the Fudan University, where there are thousands of PCs, each with a web browser installed. Here, the network is not only protected from the "outside world" by firewall, any incoming and outgoing requests must go through a central proxy. In addition, there is a quota policy to restrict the amount of bandwidth each member of the university community can utilize, and every bit of external data transferred will be charged! Since there is no cache sharing among different users in current web browser architecture, even if the requested information is available in a node within the campus network, the request may be sent out, incurring both long response time and cost. By sharing the local caches, we can expect to save cost as well as reduce the response time.

To this end, we have designed the BuddyWeb, a P2P-based collaborative web caching system. In BuddyWeb, all the local cache of participating nodes are sharable, and nodes within the enterprise network will be searched first before remote external accesses are invoked. BuddyWeb is unique in several way. First, the peer network can dynamically reconfigure itself based on the similarity of its interests. In other words, over time, we expect to see communities of different interests formed. Second, we propose a novel routing strategy, which based on the idea of similarity of peers' contents. Query will be routed from a peer to its neighbor that has the highest similarity value. Third, BuddyWeb adopts a self-adaptable hopping strategy in which the TTL will self adjust to achieve the objectives of maximizing the positive search results and minimizing the bandwidth usage. Finally, BuddyWeb combines the power of mobile agents into P2P systems to perform operations at peers' sites. This facilitates content-based searching easily.

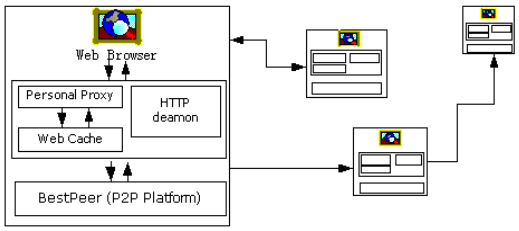

Fig. 1. BuddyWeb Architecture

We are implementing BuddyWeb on top of BestPeer[2], a agent-based P2P system. The BestPeer network consists of two types of entities: a large number of computers (nodes) and a relatively fewer number of *Location Independent Global Names Lookup* (LIGLO) servers. Each participant node must register with a LIGLO and can communicate or share resources with other nodes (i.e. peers) in the network. A LIGLO server, which has a fixed IP address, provides two main functions. It generates a BestPeer Global Identity (BPID) for a node so that nodes with varying IP addresses can be"recognized" as a single unique entity. It also maintains peer's current status, such as whether it is online or offline. Agents are extensively used in searching and to perform tasks at remote peer site.

2 BuddyWeb Node

Figure 1 depicts the architecture of an autonomous peer in BuddyWeb. The web browser serves as the front-end interface to user. We are using Microsoft Internet Explorer as the browser interface. Users will not notice any difference between a BuddyWeb-enabled browser and the original browser. In a BuddyWeb-enabled browser, there is a personal proxy that works with the local cache, and a HTTP deamon to support HTTP requests. The cache in collaboration with the BestPeer platform, is responsible for sharing cache data with other peers in the BuddyWeb network. The low level communication between peers are managed by the BestPeer platform.

Whenever the web browser submits a URL query, the local proxy will receive and rewrite the query into the input format of the BestPeer platform. The query will then be passed to BestPeer platform. Based on the requirement, BestPeer generates a mobile agent and dispatches it to the BuddyWeb network to search for matching documents. Upon receiving a match, BestPeer passes the information, i.e., document location, back to the personal proxy. In this way, the personal proxy will issue HTTP request directly to the peer that has the documents. The peer, upon receiving the HTTP request, will process it by the HTTP deamon, and sends the requested documents to the requester.

3 Similarity-based Reconfiguration

Dynamic reconfiguration is facilitated so that a peer is always directly connected to peers that provide the best service. BuddyWeb uses LIGLOs to facilitate the dynamic reconfiguration. Each registered peer is responsible for sending to the LIGLO its IP address and extra surfing information. Such surfing information will reasonably reflect the peer's interests.

Getting the peer's interested tendency could be deployed by use of its browsed pages. The information must be both representative and brief. Different policies might be employed from IR field. One simple way is to send some useful meta-data (say, <TITLE> </TITLE>) in the browsed pages to the associated LIGLO. Alternatively, user can provide feedback in the form of highlighting some keywords in the browsed pages.

Peers interest tendencies will be represented as word lists in LIGLO. Dynamic reconfiguration could be facilitated on the basis of those word lists, which we call *Peer-Tendency*. Keeping reconfiguration in mind, we could see all the words in *Peer-tendencies* as a word bag, which could be used to construct a vector space. Each *Peer-Tendency* will be transformed to a corresponding vector in such a vector space according to some weighting schemes. Note that each LIGLO only holds information of the peers registering to it. Thus, a negotiation must be held among all the LIGLOs, and thereby determines which LIGLO receive the *Peer-Tendencies*. An alternative approach to this problem is to use hash method, which could avoid vector computation in a single server. However, in our context, there are relatively fewer LIGLO severs in the BuddyWeb network. Besides, the *Peer-Tendencies* are "light-weight" files. Therefore, we decide to implement the vector computation with the former approach.

A proper similarity function, e.g., cosine function, could be defined using VSM (Vector Space Model)[3]. To keep the most beneficial peers as directly connected neighbors, each peer should maintain those peers with which it has the highest similarity values. However, the computation of all pair-wise peers is a time-consuming task. A simple way to solve this problem is to distribute the similarity computation to every peer, i.e., the responsible LIGLO computes only all the vectors in VSM. After that, those vectors will be sent to every LIGLO server. The reconfiguration policy works as follows.

- In a certain period (say, every midnight), the LIGLOs will negotiate to compute the vectors of all the peers. After that, each LIGLO will hold the computed vectors of all the peers.
- When a peer is online, it will communicate with its registering LIGLO to send its current IP address and request all the other peers' vectors. Similarities with all the other peers will be computed by the peer locally. Those with the higher similarity will be ranked higher.
- The k peers with the highest value of similarity will be kept as the directly connected neighbors, where k is a system parameter that can be set by participant peers.

It is necessary to point out that effective clustering algorithms could be easily employed to group peers with common interests into clusters. This gives us a way to discover the peer communities appearing in the BuddyWeb network.

4 Similarity-based Routing and Self-adaptive Hopping

In BuddyWeb, the dynamic reconfiguration is facilitated by use of a similarity computation, which provides a reasonable measure of the relationships between peers. Considering the situation, in which a query agent is to be forwarded to the directly connected neighbors, the peer forwarding the query agent maintains the similarity values with its neighbors. Instead of forwarding an agent to all its direct neighbors, the peer could select the neighbor with the highest similarity value. Note that the query agent from the initial node will be propagated to all its directly neighbors and similarity-based routing policy will be adopted for further forwarding.

Another benefit from using similarity measure is that query agents could self-determine its spreading hops. To our knowledge, previous P2P systems have to pre-determine the number of hops of queries, such as TTL value. If the number of hops is set too low, the search process will be limited to a small scope. However, if it is set too high, the traffic over the peer network can become heavy. An ideal trade-off is hard to be achieved because the TTL value of queries is set one-size for all.

With agent technology, the query agent could "remember" its history information in the routing path. To more visually demonstrate our strategy of self-adaptive hop, we'd like to take distance as the measure of peers' relationships. Based on the distance of pair-wise peers, the longest distance between peers could be seen as the diameter of the peer network in a concept space. Note that the "diameter" here is just to measure the knowledge scope that all peers' interests cover.

The self-adaptive hopping strategy works as follows

- A peer initiates a query agent with a parameter s instead of TTL value, where s is pre-defined by the P2P network. The diameter of the peer network is denoted by D, which will be obtained in the process of facilitating the dynamic reconfiguration.
- When a query agent is forwarded to the directly connected neighbor by a peer, the agent "remembers" the distance value between the peer and its neighbor. The value is summed with the previous remembered distance values in the routing path.
- If the sum value exceeds the value of $s \cdot D$, the hopping will stop. Otherwise, the peer will forward the query agent to its neighbor further.

As such, the system need not have a fixed hop number for all the queries. With the parameter s given by the system, the number of hops of each query will be self-determined according to its searching scope in the concept space. The parameter s reflects the extent of scope the system likes its peers to search in

the concept space, and thereby enables our system to find a proper trade-off between network traffic and search completeness in a dynamic way.

5 Agent-based Self-adaptive Search

Applying mobile agents in P2P collaboration cache has several advantages: (a) it allows individual requester to filter the content according to what (s)he desires. (b) It facilitates extensibility - new algorithm or program can be used without affecting other parts of the system. (c) It optimizes network bandwidth utilization as only the necessary data is transmitted to the requester.

Considering the current application, the user interface allows many candidate results to be displayed. This demands that matched pages in the remote peers must be processed at the remote site and only abstracts of the pages be returned to the query peer. Thus, the search algorithm is encapsulated in the query agent and performs its operation in each destination peer. More importantly, this mechanism distributes the search operations over the peers in the BuddyWeb network. This is a reasonable approach as it exploits parallelism by enabling all peers to operate on their data simultaneously, thereby providing a potential improvement in the search performance.

Agent-based technology also enables search algorithm to self-adapt its parameters or even strategies according to the search results in the prior peers. In particular, if the given query results in few matching pages during the search process in the previous peer's cache, the algorithm could change its search policy in order to avoid final scanty results returned to the query peer. For instance, the algorithm might change some parameter of matching threshold or add some additional key words with the help of thesaurus embedded in the system. On the contrary, if the search ends up with abundant candidate pages from prior peer, it could increase some associated values of threshold parameters or even change the search strategy.

6 Conclusion

This position paper outlines some underlying ideas in our developing system, which provides the services of web caching and searching on the basis of P2P technology. The ideas mentioned in above sections will be more and more concrete in the future development of the system. We have begun implementing BuddyWeb, and expect to deploy it in the Fudan campus network very soon.

References

1. International Workshop on P2P Systems. *http://www.cs.rice.edu/Conferences/IPTPS02/*. 2002.
2. Wee Siong Ng Beng Chin Ooi and Kian-Lee Tan. Bestpeer: a self-configurable peer to peer system. In *submitted for publication*.
3. M.N. Murty A.K. Jain and P.J. Flynn. Data clustering: A review. In *ACM Computing Survey*, volume 31, Sept 1999.