Efficient Web Content Delivery Using Proxy Caching Techniques

Daniel Zeng, Fei-Yue Wang, *Fellow, IEEE*, and Mingkuan Liu

***Abstract*—Web caching technology has been widely used to improve the performance of the Web infrastructure and reduce user-perceived network latencies. Proxy caching is a major Web caching technique that attempts to serve user Web requests from one or a network of proxies located between the end user and Web servers hosting the original copies of the requested objects. This paper surveys the main technical aspects of proxy caching and discusses recent developments in proxy caching research including caching the "uncacheable" and multimedia streaming objects, and various adaptive and integrated caching approaches.**

***Index Terms*—Caching performance evaluation, dynamic content caching, proxy caching, Web caching.**

I. Introduction

THE AMOUNT of traffic over the Internet has experienced tremendous growth in recent years largely due to the wide adoption of the World Wide Web technologies and the resulting explosion of Web-based content development and dissemination [1]–[3]. The Internet bandwidth capacity expansion, on the other hand, is lagging behind, making the Web a major performance bottleneck. The gap between the Web infrastructure capacity and demand will continue to exist, if not expand, as information search and business transactions are being increasingly conducted over the Web. Another compounding factor is related to the recent developments in the Web technologies such as Web services, which will potentially bring in new classes of distributed applications in large numbers that will communicate among one another over the Internet, consuming network bandwidth [4].

Web caching is an established approach to meet the important Web capacity challenge and address related issues such as user-perceived network latencies. Broadly speaking, caching can be defined as serving user Web requests from places other than the Web servers that publish the original copies of the requested objects [5]. Recent years have seen significant growth in the Web caching literature and a large number of commercial offerings from both established network vendors and startup companies that exclusively focus on caching-related hardware and software solutions (see http://www.web-caching.com/ for a partial list of Web caching products). In effect, caching is ubiquitous in today's computing environment. All of the major Internet backbone providers and Internet service providers (ISPs) now implement Web caching as part of their infrastructure, often transparent to end users and service subscribers [5]. Many medium-to-large enterprises are using a variety of caching products and services to improve the network performance and reduce networking connection costs. Many end-user programs, including Web browsers, also maintain their local caches to reduce user-perceived network latencies.

According to the location of caches, Web caching systems can be classified into three types: browser caches, proxy caches, and surrogate caches. Browser caches are located within user browser programs. Surrogate caches are typically located near the Web servers and are owned and operated by the Web content providers [6]. Proxy caches are located between end-user client sites and original Web servers, typically closer to the clients than to the servers. Proxy caches are typically configured and operated by ISPs and enterprises operating internal networks that are connected to the Internet. This paper mainly focuses on proxy caching for the following four reasons. First, a dominant portion of the current caching literature is directly related to various technical aspects of proxy caches. Although surveys on Web caching technology exist in the literature (e.g., [5], [7]–[9]), new developments in proxy caching and its extended applications in areas such as caching "uncacheable" Web objects (e.g., [4], [10]) and differentiated services (e.g., [11]) are of important practical significance, calling for a new updated survey. Second, from the point of view of system deployment, proxy caching does not require major changes in the networking environment and can achieve the economy of scale because multiple users are served. In addition, proxy caching does not rely on any major changes (e.g., with respect to protocols) to original Web servers and, in most cases, does not require much end-user configuration efforts. As a result, proxy caching can be implemented in a relatively transparent manner, making it easy to adopt and upgrade in practical settings. Third, there are several new streams of proxy caching research that have emerged in recent years. For instance, analytical models have been developed to characterize various decisions that proxy caching systems have to make (e.g., [12], [13]). Such models are capable of predicting the performance of caching systems under different operating environments and establishing the applicability of some

Manuscript received July 9, 2002; revised November 14, 2003. This work was supported in part by the Outstanding Young Scientist Research Program (Grant 60125310), the Key Project on Networked Systems (Grant 60334020), from the National Natural Science Foundation; a 973 Project (Grant 2002CB312200), from the Ministry of Science and Technology; a Shandong 863 Project (Grant 030335), from Shandong Provincial Government; and a grant for open research projects (ORP-0303) from the Key Lab of Complex Systems and Intelligence Science, Chinese Academy of Sciences, China. This paper was recommended by Associate Editor J. A. Miller.

D. Zeng is with the Management Information Systems Department, The University of Arizona, Tucson, AZ 85721 USA (e-mail: zeng@bpa.arizona.edu).

F.-Y. Wang is with the Key Laboratory of Complex Systems and Intelligent Science (LCSIS), Institute of Automation, the Chinese Academy of Sciences, Beijing 100080, China, and also with the Program for Advanced Research in Complex Systems (PARCS), Systems and Industrial Engineering Department, The University of Arizona, Tucson, AZ 85721 USA (e-mail: feiyue@sie.arizona.edu).

M. Liu is with the Electrical and Computer Engineering Department, The University of Arizona, Tucson, AZ 85721 USA (e-mail: mingkuan@email.arizona.edu).

Digital Object Identifier 10.1109/TSMCC.2004.829261

of the widely used caching decisions or policies. Another interesting trend is to develop integrated and adaptive caching approaches that treat several aspects of caching, which have traditionally been studied in isolation, in an integrated manner (e.g., [14], [15]). One of the main purposes of this paper is to summarize the key results and point out future directions in these new research areas. Fourth, the distinction between browser caches, proxy caches, and surrogate caches is somewhat artificial. From a system implementation and deployment standpoint, such distinction is useful and easy to make. However, from a research perspective, these caches share many similar technical challenges and many techniques developed for one cache type are directly applicable to others. For instance, most of the caching techniques developed for single proxy caches (see Section III) can be readily applied to browser caches. As such, although in this paper we frame all of the issues in the context of proxy caching, a majority of them are also applicable to other types of caches.

The rest of the paper is structured as follows. Section II discusses the intended impacts of Web caching and related performance measures. Section III focuses on proxy caching approaches that use a single cache to serve multiple users. This section summarizes research results on all classical single cache topics including replacement, consistency, and prefetching, with the emphasis on new developments such as using analytical modeling as an evaluation methodology and applying data mining and machine learning techniques to predict user future Web-access patterns. In Section IV, we present research issues related to the development of cooperative proxy caches where a network of caches works together to serve user Web requests. Our main focus is on the design of intercache cooperation and routing protocols and the resulting performance implications. Section V provides an overview of several emerging topics in proxy caching. We believe that these topics are of significant practical relevance and the related technical solutions can potentially become a core component of the next-generation caching technology. We conclude the paper in Section VI by summarizing the paper and pointing out future research directions.

II. Benefits of Web Caching and Performance Measures

The potential benefits of Web caching are multifold. From the end user's standpoint, caching can significantly reduce the user perceived network latency and improve their Web experience. From the perspective of the Internet infrastructure, caching can reduce the amount of the Web traffic. As a result, the overall performance of the Internet can be improved and network congestion minimized. In addition, for enterprises that pay the ISPs for wide-area network bandwidth based on the amount of network traffic, reduced traffic means lowered costs. From the point of view of the Web server, caching can significantly reduce the server loads and improve the system responsiveness. In addition, because copies of Web objects are maintained throughout the network, the user can access them even when the original servers hosting them are down. Caching servers have been used in recent years to host value-added services such as security, content filtering, and advertisement that are not directly related to caching [16].

Before presenting various performance measures that have been developed to evaluate the effectiveness of caching systems, we briefly describe how proxy caches work. Conceptually, proxy caches can be viewed as a middleware connecting end-user Web programs and Web servers through Web protocols. Such caches function as servers to client programs and as clients to Web servers. When a user requests a Web object, the proxy cache processes the request first. If the object has a valid cached copy stored in the cache, this copy is immediately returned to the user and the request is not forwarded to the Web server hosting the requested object. If a cached copy is not located in the cache, then we say that a cache *miss* occurs. In a single proxy cache configuration, when a miss occurs, the original Web server is contacted and the returned Web object is forwarded to the client program. In a configuration that consists of a network of proxy caches, before connecting to the original Web server, the proxy cache follows certain established protocols to route the request to other participating caches in an attempt to locate a cached copy.

In the caching literature, two main performance measures–*cache hit rate* and *cache byte hit rate*–have been used to evaluate Web caching systems [5], [9]. Cache hit rate is calculated as the number of the user-requested Web objects that are answered by the cache divided by the total number of the requested objects. Cache byte hit rate is defined as the number of bytes served from cached content divided by the total number of bytes served. Achieving the highest hit rate and byte hit rate under given resource constraints is one of the primary goals of Web caching.

Several additional measures have also been used by caching researchers to evaluate the impact of Web caching on the end-user Web experience and network performance, although they are not as widely accepted as hit rate and byte hit rate and their definitions can be somewhat vague. *User-perceived network latency* measures, from an end user's perspective, are the actual delay between the time a Web request is issued and the time the Web object is returned to the Web browser for rendering [17]. Statistics regarding *network utilization*, such as average and maximum bandwidth consumption rates, can help determine the impact of caching on networking resources. In particular, researchers often resort to the following two measures [18]: the percentage saving of the amount of network traffic because of the use of locally cached contents, and the amount of network traffic that can be attributed to maintaining and using caches. The latter measure provides an indication of the overhead associated with a caching approach.

III. Single Proxy Cache—Replacement, Consistency, and Prefetching

In this section, we focus on Web caching systems that are based on a single proxy cache. We discuss three key operational decisions that the proxy has to make. The first decision, called *replacement*, is concerned with how to utilize the limited cache storage capacity to achieve the best caching performance. Although the cost of disk storage has dropped dramatically in recent years, storage capacity is still a limiting factor, especially for proxy caches that may serve a large number of users. The second decision, called *consistency*, is motivated to deal with a

fundamental dilemma in caching: How to ensure that the cached Web objects are "fresh" while the original copies may change over time? The third decision, called *prefetching*, attempts to further improve the performance of proxy caching systems by proactively fetching certain Web objects that have not yet been accessed by the users but may be requested in the future with high probability.

A. Replacement

There are two major issues concerning storage management in a proxy cache. First, the proxy needs to decide which Web objects should be stored in the cache. Second, it needs to decide which Web objects should be evicted from the cache in case storage is needed for new incoming objects.

Most caching systems use simple heuristics to resolve the first issue, often referred to as the admission control problem. For instance, many proxies try to store a copy of new cacheable objects as they are accessed by the user. Prefetching techniques (see Section III-C) may also generate lists of Web objects that need to be fetched and stored in the cache. In traditional approaches, the second issue concerning the eviction of objects is treated separately from the first issue. In this section, we survey these traditional approaches and then discuss the basic ideas behind several new replacement policies. Some of these new policies make admission control and eviction decisions in an integrated manner.

Traditional replacement approaches or policies can be divided into three groups [2]. The first group includes the classic replacement policies originally developed in the context of computer memory and disk caching system such as least recently used (LRU) [19], [20] and least frequently used (LFU) [21]. Whenever storage space is needed, LRU evicts the object that was used least recently and LFU evicts the object that was used least frequently. The second group makes eviction decisions based on other attributes of the object such as size or the retrieval speed. For instance, one such policy evicts the largest object [22] and another evicts the object that can be retrieved fastest from its original Web server [23]. The third group of replacement policies tries to make eviction decisions by considering a number of relevant factors simultaneously using a cost function [24], [25]. One representative policy in this group considers size, retrieval speed, and time last used based on the greedy-dual-size algorithm [26]. In this approach, a Web object is given an initial value based on its size when it enters the cache or is accessed by a user. Then, its value decreases gradually over time unless it is accessed again. The object with the least value is then evicted if necessary.

Caching researchers typically use two kinds of simulation methods to evaluate and compare the performance of various replacement policies. The first kind uses event-based simulation where various characteristics of Web objects (such as size), network speed and connection delays, and user Web requests are artificially simulated based on some commonly accepted statistical models using a limited number of parameters (e.g., Zipf distributions are frequently used to generate Web object sizes [27]). The second kind of simulation is called trace-based simulation and uses the actual logs from Web servers or proxy servers, providing realistic Web environments for testing purposes. A predominant performance measure used in these studies is hit rate.

More recent research in replacement starts to use analytical methods to augment simulation-based evaluation. For instance, in [12], a precise analytical model is developed to evaluate the LRU policy and its variations. The user Web access behavior is modeled using a system of differential equations which explicitly take into consideration the ages of documents (the time elapsed since these documents were last accessed). The exact expressions for the hit rate and expected network latency can then be derived. The benefits of this kind of analytical work are twofold. First, it provides a theoretical framework to evaluate known policies and can generate technical insights as to why certain policies are effective in practice and outperform others. Second, it can lead to the development of new policies based on the deepened understanding of the structure of the caching problems and the interactions among various components of the caching system.

We conclude this section by summarizing several new trends in research aimed at developing effective replacement policies. First, recent studies start to focus on more realistic performance measures such as user-perceived latencies as opposed to simple criteria such as hit rate and byte hit rate which are easy to define and determine but do not directly measure the impact of caching either on end users or the network performance. Researchers are also examining performance measures combining hit rate and byte hit rate when measures such as user perceived latencies are difficult to obtain [28]. Second, many refinements to traditional replacement approaches have been proposed and evaluated, taking into account the specific characteristics of Web caching. For instance, LRU has been extended to be adjustable according to the size of Web objects [2]. Third, recent work emphasizes the importance of admission control. Various effective admission control policies have been developed and their performance studied along with replacement policies. [2], [12]. For instance, in one approach, before a new object is cached, its expected contribution to the hit rate is compared with that of one or more cached objects that will need to be evicted to make storage space available for the new object [2]. This new object will be cached (and some other objects evicted) only if its expected contribution outweighs the loss as a result of necessary evictions.

B. Caching Consistency

If the Web objects cached in the proxy cache are never updated on their original Web servers, these cached objects will always be fresh and the user will never be given outdated contents. However, in real-world applications, most of the Web servers update their published contents dynamically. Some Web objects or servers may even become unavailable. As a result, the cached objects may be stale or invalid. Caching consistency (also referred to as coherency) research is aimed at developing various protocols between proxies and Web servers and related operational polices to address such staleness problems. This section surveys major types of caching consistency techniques. Before presenting the basic ideas behind these techniques, we first discuss the caching performance measures relevant to consistency and the technical constraints under which caching consistency techniques operate.

Two measures of staleness–worse-case staleness and average staleness–are commonly used to evaluate the effectiveness of a consistency approach [29]. An upper bound on worst-case staleness ensures that objects returned by the cache are never stale by

more than the given bound. Average staleness is defined as the product of two terms: the percentage of Web objects returned by the cache that are stale, and the average amount of time for which the returned object has been stale relative to the update that occurred on the server.

Another important performance measure is the network bandwidth overhead associated with consistency approaches. Such overload includes the bandwidth consumed by consistency-related communications between the proxy and the Web servers and by re-downloading the cached objects. The main technical objective of all consistency techniques is to achieve the desired staleness bounds with the minimal network bandwidth overhead.

Caching consistency poses several unique technical challenges. Unlike caching replacement, where the related decisions are made by the proxy based on local information, consistency inherently involves both the proxy and the Web servers. Thus, when designing a consistency approach, one needs to consider issues such as what can be expected of Web servers in terms of protocol support and the amount and quality of information available on the Web objects. In addition, scalability issues play a significant role as the proxy may need to maintain state information for a large number of Web servers and a Web server may need to maintain state information for a typically even larger number of clients including proxies.

Caching consistency approaches can be grouped into three types: client-driven, server-driven, and hybrid. In a client-driven approach, the proxy is responsible for verifying the validity of its cached objects. In server-driven and hybrid approaches, the Web servers also actively participate in the validation process by notifying proxies updates. Current real-world caching applications predominantly use client-driven consistency methods due to their simplicity [5]. However, recent research has shown that other two approaches have the potential to significantly reduce the network bandwidth overhead while maintaining the similar staleness guarantees [18]. We project that these approaches involving servers will become more popular in practice. In effect, this is already the case for surrogate proxies.

In the remainder of this subsection, we present the basic ideas behind representative protocols for each of the above three consistency approach types. Note that due to the large number of consistency approaches that have been developed, it is not our intention to provide a comprehensive survey. Rather, we aim to illustrate the key tradeoffs and the basic engineering ideas related to maintaining caching consistency.

We first discuss client-driven approaches. All of these approaches make use of the time-to-live (TTL) information. The HTTP protocol allows the Web server to specify the `expires` or `max-age` header fields for all cacheable objects. The proxy can then use TTL information to decide whether a cached object needs to be validated with the Web server. Such validations can be done either reactively or proactively [29]. In a reactive approach, a validation message is sent to the server only when the cached object is requested by the user and the associated TTL indicates that it is stale. In a proactive approach, validation messages are sent periodically to the Web servers for objects that have expired regardless whether they are being requested by the user. The basic tradeoffs between these two approaches are clear: The reactive approach can guarantee the strong consistency (assuming that the TTL provided by the server is valid) but at the cost of increased user-perceived latency, network traffic, and server load. The periodic proactive approach may reduce user-perceived latency. However, it can result in significant network and server overhead.

In cases where the Web server does not supply (accurate) TTL information, the adaptive TTL approach is shown to be useful [10]. This approach estimates an object's TTL based on the observations of its lifetime in both the proxy and the server. In its simplest form, an object's TTL is set to equal a fixed percent of the document's current age, defined as the difference between the current time and the last modified time of the object. This estimation reflects the intuition that an object that remains unchanged for a relatively long period of time is unlikely to be changed quickly.

A useful technique in implementing caching consistency protocols is piggybacking. Whenever the proxy needs to communicate with a server (not necessarily for consistency reasons), it piggybacks a list of cached objects from that server that need to be validated. The server handles the primary request first and then indicates which cached objects are now expired. The piggybacking technique can reduce the network bandwidth overhead and, to a lesser degree, Web server load. This technique applies to all consistency methods, not limited to client-driven ones.

We now turn our attention to server-driven consistency approaches. The basic version is called the Callback protocol [10], [30]. In this approach, the Web server keeps track of which proxies are caching which objects. Whenever an object needs to be modified, the server notifies the corresponding proxies by sending them invalidation messages. Obviously, the Callback approach can be improved by using piggybacking approaches to reduce network bandwidth consumption. Server-driven approaches have several advantages. They help reduce the messaging overhead since consistency-related messages are sent only when objects are actually updated on the server. They also help maintain strong consistency since the server has immediate knowledge of which Web objects are updated. However, server-driven approaches make important assumptions about the server behavior and functionality, which in real-world applications, are often not met. In these approaches, the server needs to dynamically keep track of proxies on a per-object basis. It may also cause communication "bursts" when objects are being updated. Before the consensus has been reached among the Web content providers and server developers regarding servers' responsibilities with respect to caching consistency, it seems that the applicability of server-driven (and hybrid) approaches is limited with the exception of surrogate caching where servers and caches are closely tied together and designed in an integrated manner.

The last type of consistency approach, the hybrid approach, requires close collaboration between servers and proxies through the use of *leases* [18], [29], [31]. Two types of leases have been proposed and studied in the literature. The first type, object lease, is a promise by the Web server to a proxy that the proxy will receive an invalidation message if the leased object is modified during the time period specified in the lease. The second type, volume lease, is referred to the leases involving a collection of Web objects, called a volume, instead of individual Web objects.

Object leases can be used to maintain strong consistency of the cached Web objects, especially in implementations where the Web server commits the updates to an object only after all of the proxies with valid leases acknowledging the invalidation message or their leases expire. In this case, strong consistency is achieved without significant overhead for two reasons. First, the server only needs to maintain a relatively short list of proxies that explicitly express an interest in the object. Second, the volume protocol provides an important fault tolerance feature through a "time-out" mechanism. Consider, for example, that the server sends out an invalidation message to a proxy which is not reachable because of network problems. The server will then automatically assume that the proxy is no longer interested in the object (after its lease expires) and commit the change without further delay. Volume leases may further reduce the server and communication overhead by aggregating the proxy lists and invalidation messages. In certain protocols, object and volume leases are used together to provide guarantees for both average staleness and worse-case staleness measures [32].

Two decisions that have important performance implications need to be made when applying a lease-based approach. The first one is concerned with the determination of the lease duration. The second one, applicable to volume leases, is concerned with the granularity problem: How to group objects into volumes? The basic tradeoffs are clear [32], [33]: Longer leases may reduce messaging overhead but increase the length of the proxy lists and may lead to extended delay in object updates; larger volumes reduce the proxy lists but may lead to unnecessary invalidations to the Web objects in the volume. How to make effective decisions, however, remains open. Empirical studies indicate that the specific characteristics of Web sites and applications need to be carefully considered when making these decisions [5].

Recent work has also explored the use of multicasting and hierarchical networks to further reduce the communication overhead associated with various lease-based consistency approaches [29]. Although promising results have been shown, this line of research cannot be immediately applied because of lack of support for the needed protocols in the present networking environment.

C. Prefetching

Caching objects that have already been accessed by the user is of limited use in an environment where the user frequently needs to explore new information. Prefetching is a Web caching technique that caches Web objects in anticipation of the user's future needs [34]–[37]. A central issue in prefetching is how to predict user future Web-access patterns. Depending on where the information used for prediction is collected, prefetching methods can be divided into three groups: proxy based, server based, and hybrid where proxies and servers jointly decide what should be prefetched. In this subsection, we first focus on learning mechanisms that can predict user future-access patterns regardless of the source of the data used for prediction. We then discuss how such predictions can be used by a caching system and summarize the related operational issues.

Many factors influence user Web usage. To achieve high predictive performance, a prediction approach needs in-depth knowledge about the user's information needs as well as the contents of potentially relevant Web pages. In the context of caching, however, developing such a high-performance approach which necessarily entails costly information collection and computational overhead is typically unnecessary. Instead, caching-related prediction approaches typically aim to provide *reasonable* Web usage prediction based on a limited set of information that can be easily acquired. One commonly used information source is past Web-access information either at the individual user level or at the aggregated level based on user groups. Such historical access data are typically recorded at various points of Web-based systems including browsers, proxy caches, and Web servers. Another common information source that can be easily tapped into is the hyperlink structure of given Web pages. We call the approaches that mainly make use of historical access data *history-based* approaches. The approaches that primarily use Web page structural information are referred to as *structure-based* approaches.

The simplest structure-based approach is the one that prefetches all of the embedded links of the Web page that is being currently accessed by the user. More sophisticated approaches take into consideration factors such as the type of the Web object referred to by the embedded hyperlink and the past latency statistics of the Web server hosting the object [38].

The simplest history-based approach is the "Top-10" algorithm [37]. In this approach, the number of past access requests is maintained for each Web object residing on the Web server or the proxy. A list of the most popular (i.e., with the largest number of requests) documents is then periodically compiled and used as predictions for future Web accesses. The Top-10 algorithm is based on highly aggregated information and is essentially *stateless* in that the predication is made regardless of the user or which Web pages he or she is currently visiting. A large number of *state-conscious* prediction methods have been developed that explicitly consider the user's current browsing activities when making predictions [39], [40]. Most of these methods fall under the general umbrella of Markov-based methods. Simply put, a Markov model of Web-access patterns has three components: a set of all possible Web objects that may be accessed by the user, a set of all possible states characterizing the user's current browsing activities that are believed to be predictive of future accesses, and a transition probability matrix recording the probabilities of accessing a given Web object when the user is in a particular state. The main differentiator for these Markov-based methods is the way in which the state is defined. The first-order models typically define the state as the Web object most recently accessed by the user, whereas the second-order models use the most recent (ordered) pair of Web objects. Higher-order Markov approaches relying on longer user access paths have also been explored in various forms [39], [41]. The key algorithmic aspect of these Markov-based methods is how to construct the transition probability matrix efficiently both in computing time and memory space using historical access logs.

Another class of history-based methods based on association rule learning have been developed in parallel with Markov-based approaches [42]–[44]. The basic intuition is that if a group of Web objects has been frequently visited together in the past,

it is very likely that they will be visited again as a group in future accesses. The association rule learning algorithms provide an efficient computational mechanism to identify such highly correlated groups from historical data. These algorithms also allow the caching system designer to specify the desired degree of correlation or association to tradeoff between the coverage and reliability of the learned rules.

A challenging technical issue relevant to both association rule-based and Markov approaches is how to define operationally a transaction—a sequence of user Web accesses that are closely related to each other in the context of a concrete user task [5]. If these transactions are defined in such a way that unrelated Web accesses are grouped together, false associations or links will then be introduced into the Markov model building or association rule learning. Conversely, if these transactions are constructed such that related accesses are separated into different transactions, valuable association or linkage information will be lost. In both cases, the overall predictive power of these history-based approaches may be reduced significantly. Another challenging issue with history-based approaches is that they are relatively slow to respond to changing Web behavior and can be completely useless when confronted with a document not requested before [45], [46]. Recent research has explored the possibility of combining structure-based and history-based approaches to deal with these challenges with limited success [39], [47].

We now turn our attention to caching-specific operational issues, assuming that good-quality predictions on user future-access patterns are available. From the system perspective, the basic tradeoff in prefetching is between potential reduction in user-perceived Web latency and increase in network traffic and server loads as a result of fetching Web objects that may never be accessed by the user [46], [48]. This tradeoff governs the design of prefetching operations which have two key elements: *when* to fetch and *which* Web objects to fetch. In terms of the timing of fetching activities (after predictions are completed), the following two main approaches have been developed [45]: The online approach fetches Web objects during pauses while the user reads the displayed material on the computer screen; the offline approach fetches Web objects during the off-peak periods or after the user becomes idle for a certain period of time. As for deciding which Web objects to be fetched, a number of heuristic filtering mechanisms have been developed in both commercial products and caching research [39], [45]. For instance, some mechanisms do not fetch Web objects that can be quickly retrieved even if they are very likely to be visited in the future. Other mechanisms apply a threshold to filter out Web objects with relatively low access probability.

Recent studies start to examine the interaction between prefetching decisions and other types of caching decisions. For instance, storage allocation has been studied in the prefetching context. Several analytical models have been formulated to achieve the maximum (expected) hit rate, the maximum byte hit rate, and the minimum user-perceived network latency [17]. These models also lead to a precise understanding of the widely-used Top-10 policy and its applicability and effectiveness. In another example, researchers have proposed a model treating prefetching aggressiveness, replacement, and network bandwidth overhead in an integrated framework [49].

IV. Cache Networks: Cooperation and Routing

The previous section focuses on key operational decisions that govern the behavior of a single proxy cache. The benefit of a single cache is limited by its inherent computational resource constraints (e.g., storage and CPU cycles) and the size of the user/client population it serves. Cooperative proxy caching (i.e., using a distributed network of proxy caches to satisfy user Web requests), has been developed to address these scalability issues associated with single caches [50]–[52].

Cache networks bring about several advantages. First, through sharing caches among a large number of users, more efficient utilization of caching resources can be realized when compared with a single cache approach. In addition, the potential savings in network bandwidth can also be significantly higher than those in the single cache case. Second, caching networks provide a natural solution to applications that involve serving a large, geographically dispersed user population in support of their diverse Web requests, since multiple caches can be strategically located between the users and original Web servers. Third, cache networks help improve the overall performance of the caching system by balancing loads between proxies. Furthermore, they improve the network fault tolerance and robustness by removing the single point of failure. Large organizations and ISPs, including those operating the Internet backbones, have been the main adopters of cache networks [5].

From a design perspective, each participating proxy in the cache network is a full, independently-run proxy cache. As a result, all of the single cache operational issues discussed in Section III need to be resolved. Often these issues need to be re-examined in the cooperative caching context. We use two examples to illustrate the need for such re-examinations. The first example is concerned with the choice of replacement strategies in a hierarchically organized cache network [53]. Since the workload characteristics differ across the levels of the caching hierarchy due to the filtering effects at lower-level caches, it is optimal to adopt different replacement policies depending on which level a cache is situated in the entire hierarchy. The second example involves subtle difficulties of maintaining caching consistency in a cooperative caching environment [54]. With a single cache approach, the user may receive a stale object but future revisits will always guarantee the same or fresher copy of the object. In cooperative caching, however, repeated access to the object may return versions older than the one previously received.

In addition to these single cache issues, coordination among participating caches has to be carefully designed. This section mainly focuses on two groups of approaches dealing with these coordination issues. The first group includes *organizational* approaches, which assign a fixed role to each participating proxy. Whenever a miss occurs at a proxy, a fixed sequence of participating caches will be contacted regardless of which Web object is being requested. The second group includes *hashing-based* approaches that do not rely on fixed roles. Instead, they directly map the URL of the requested object to the participating proxies that may have a cached copy of the object when a miss occurs.

Before discussing these two groups of approaches in detail, we briefly review two performance issues unique to cooperative caching approaches [13], [55]. First, in the single cache

case, a cache miss can be easily determined and does not add much delay when it occurs. In cooperative caching, however, determining whether a requested object can be served by one of the participating caches can be time- and resource-consuming. Sometimes, it is more efficient to retrieve the object from the original Web server than to locate it from the cache network. Second, the communication overhead and the resulting network bandwidth consumption relating to inter-cache coordination and synchronization can be significant. Such costs have to be factored into consideration when selecting a cooperative caching approach.

A. Organizational Approaches

Organizational approaches link participating proxy caches together statically. These links essentially establish routes for Web requests to be forwarded among proxies until a cached copy is found. Depending on the topology of the cache network, the organizational approaches can be further divided into three categories: *hierarchical*, *distributed*, and *hybrid* caching.

In *hierarchical caching*, proxy caches are organized as a tree [52]. If a cache miss occurs at a proxy, it will forward the Web request to its parent proxy. This forwarding process continues until either a cached copy of the requested object is found or none of the proxies along the forwarding path including the root cache has a copy. In the latter case, the original Web server will be contacted. The retrieved object then travels in the opposite direction to reach the user, leaving a copy at all intermediate caches.

Hierarchical caching has been heavily used by the ISPs, partly because access to the Internet is provided through similarly structured hierarchical layers [56]. It is relatively simple to maintain and does not incur much coordination overhead. The main problems associated with hierarchical caching are as follows. (a) Each cache level introduces additional delays. (b) The degree of sharing of cached objects across proxy caches is low. (c) Multiple copies of the same Web object are maintained at different levels of the hierarchy. (d) Caches close to the root may easily become processing bottlenecks and may require huge storage space.

In *distributed caching*, proxy caches operate at the same level of the network, typically close to the clients. In the simplest case, topologically, these caches are organized as a fully connected graph. Whenever a miss occurs, the proxy cache forwards the request to sibling caches for possible matches. In this approach, there is no need to keep multiple copies of the Web objects at various network levels. Also, most of the coordination-related traffic flows through local networks, reducing congestions at higher network levels. In a relatively small environment with high local network bandwidth, distributed caching can be very effective [13]. The main problem with distributed caching is that it does not scale. In a large environment, distribute caching suffers several deficiencies, such as high coordination overhead and long connection time. Various approaches have been developed to improve the performance of distributed caching [57]–[60]. The key idea behind these approaches is to efficiently identify the set of sibling caches that are likely to hold the requested object through metadata which summarizes which Web objects are cached in each sibling proxy cache. Such metadata are typically duplicated at each participating cache and updated frequently. (There is significant overlap between these metadata-based approaches and hashing-based methods to be discussed in the next subsection).

In *hybrid caching*, caches cooperate with one another at all levels of the network [13], [51]. The Web object can be fetched either from parent or sibling caches if a miss occurs. The key issue with a hybrid approach is to decide where to retrieve the object, either from one of the caches or the original Web server, to minimize latency. For instance, one technique limits the cooperation between sibling caches to reduce the network latency and caching-related communication overhead [13].

B. Hashing-Based Approaches

In contrast to organizational approaches, hashing-based approaches do not enforce static, role-based routing for unfulfilled Web requests. Instead, hashing-based approaches try to map directly from the Web request to one or a small set of cooperating caches that may keep a cached copy of the requested object. Two methods have been used to implement this mapping idea: directory based and hash function.

Using the *directory-based* method, the location of cached objects is maintained explicitly by a directory service running on a separate directory server [57]. Whenever the cached contents of a participating proxy change, the proxy will send a notification to the directory server. When a miss occurs, the proxy first queries the directory service, which responds with either the location of one or several proxies that keep copies of the requested object, or a global miss message. In the latter case, the Web request will be immediately rerouted to the original Web server. The directory-based method does not incur much communication overhead and enables loosely coupled cache networks in which individual proxies can be added and removed easily without coordinating with other participating proxies. A major disadvantage of this method is that the directory server may become a single point of failure and a performance bottleneck.

The *hash function-based* method uses a hash function shared by all clients and proxies to directly map the requested URL to one or several participating proxy caches [58], [61]. When a client or a proxy needs to locate a copy of the requested Web object, it applies this shared hash function to the requested URL and then contacts the proxies identified by the returned hash value. Similar to the directory-based method, the hash function-based method does not incur much communication overhead and utilizes cache space efficiently because no multiple copies of the Web objects need to be maintained. The main disadvantage of this method is the need for all clients and proxies to use the same global hash function. The coordination overhead is nontrivial when this global function needs to be updated because of the changes in the cache network.

V. Recent Developments in Proxy Caching

The past decade has witnessed the rapid growth of the Web caching literature and the emergence and wide adoption of related technology and products in commercial settings. Despite its relative maturity, Web caching remains an active field of study and technology development because of the ever-present need for improving Web performance, and the dynamic nature of the Web infrastructure and Web-based applications.

This section surveys proxy caching research in three emerging areas: a) caching various types of "uncacheable" Web objects, b) adaptive cooperative Web caches, and c) quality-of-service (QoS)-aware differentiated caching services. Although research in these areas is still in its initial stage of development and the effectiveness of some of the related approaches has yet to be established, we argue that these areas are of significant practical relevance and representative of the ongoing and emerging Web caching research that may lead to the next-generation Web caching technology.

A. Caching the "Uncacheable"

One major factor limiting the usefulness of Web caching is that a large portion (up to 40%) of Web content is "uncacheable" [62]. Existing caching approaches typically treat static Web objects such as static HTML and picture files as cacheable and ignore nonstatic objects that contain material dynamically generated by Web servers or multimedia streaming objects. Given the maturity of Web technology, it is not unreasonable to assume that the portion of such uncacheable objects will increase. Another compounding factor is related to the recent emergence of Web services which have the potential to become a significant part of the Web. It is clear that in order to stay relevant and achieve further performance improvements, Web caching has to explore ways to cache these traditionally uncacheable objects.

This subsection focuses on caching the following types of nonstatic objects: a) Web objects that contain a mixture of static and dynamic contents, b) streaming objects, and c) Web transactions. We lay out the key technical issues and present several solution concepts that have been proposed in the literature. We end this subsection by briefly discussing techniques that have shown promise in implementing these solution concepts.

1) Caching Mixed Objects: Most existing caching approaches treat each Web object as an atomic structure. Any object that is not fully static (usually inferred by the file extension in the URL) is viewed as uncacheable. However, a large portion of dynamic Web objects have both static and dynamic parts. In fact, in many cases, the static part dominates the dynamic part in size. Several approaches have been developed to take advantage of this observation (e.g., [63], [64]). The basic idea is to separate dynamic and static parts either based on server-provided cues or templates. The static part is then cached on the proxy. When the user requests the mixed object with the required parameters, the proxy queries the original server with the given parameters, retrieves only the dynamically generated portion, and then merges the returned dynamic part with the locally stored static part before sending the page to the client for rendering.

2) Caching Streaming Data: The emergence of streaming multimedia applications on the Internet has posed new challenges to caching. Theoretically, streaming objects can be viewed by proxy caches as regular static objects. However, since these objects are typically very large, this simple approach does not work well in practice due to the cache space constraint. Another complicating factor is that multimedia object playback typically requires relatively high network bandwidth. As such, besides reducing access latency and network congestion, caching is charged with an additional technical objective to improve the quality of content delivery.

Various caching decisions have been re-examined to accommodate the unique characteristics of streaming objects [65]–[68]. For instance, several customized replacement policies have been developed to take into consideration the large size of streaming data. In another example, the large, aggregated storage capacity from cache networks is used to store streaming objects. Researchers have also explored the specific characteristics of multimedia data to facilitate caching. A prominent example is the prefix caching technique [69]. Using this technique, the proxy only caches the initial frames of multimedia streams and use work-ahead smoothing techniques to ensure the high-quality playback. This technique can significantly reduce user-perceived latency without much cache space overhead.

3) Caching Transactions: Web-based transactions encompass not only content delivery but also other more complex Web server/client interactions driven by various contingent business rules. Caching has the potential to improve the performance of such transactions which, in turn, may result in significant business value.

In most existing web caching systems, proxies are simple Web object repositories that do not provide any transaction processing capabilities. To deal with the demanding requirements of Web transaction caching, we posit that some transaction processing capabilities at the proxy level will be beneficial [70]. Many important design issues have to be tackled when developing a Web transaction caching system. Depending on the specifics of a transaction that is being cached, certain portions of the transaction can be pushed to proxies close to the user for quick responsiveness, while others still need to be processed on the Web server. Maintaining data consistency and integrity is particularly important for transaction caching. Existing Web caching systems deal with "read-only" operations, while Web transaction caching systems may need to perform both "read" and "write" operations. Research from real-time database management (DBMS) and concurrent transaction processing may provide important insights to deal with these issues.

4) Implementation Issues: There exist several standards to support caching of Web objects that have both static and dynamic elements (e.g., HPP [64]). The Web content provider needs to follow these standards to publish mixed Web objects to make them cache-friendly. Another useful and versatile implementation technique is *active caches* [10]. In this approach, the Web server supplies cache applets which are attached to Web documents. The proxy is required to execute these cache applets upon a cache hit to prepare the document to be returned to the client for rendering without explicitly accessing the original Web server. It has been shown that the active cache technique can be effectively used to turn many types of uncacheable contents cacheable at the expense of increased CPU loads on proxies.

B. Adaptive Cooperative Web Caches

Web-access patterns can change rapidly over time. For instance, the "hot spot" phenomenon is not uncommon in real-world applications: Certain Web contents are in high demand during a short period of time and then after a while, user interests decline. Such a dynamic and rapidly changing nature of the

Web suggests the usefulness of highly adaptive caching architectures.

In an example of such adaptive architectures [71], [72], the Web caching function is provided by multiple, distributed caches which dynamically join and leave cache groups based on the Web-access patterns. More specifically, each individual proxy is implemented as a software agent. The Web caching service is provided through a distributed collection of such proxy agents that act autonomously but also collaboratively to serve user needs. These proxy agents form dynamic "virtual" teams based on content demand. A proxy agent can join or leave a particular cache group depending on the user-access patterns and the updating rates of the Web sources covered by this agent. Proxy agents can also clone themselves in case of heavy user traffic, remove themselves in case of idleness to release resources, migrate on the network to "follow" user requests, and exchange capabilities among themselves to better serve changing needs.

C. Differentiated Services

Efficient QoS provisioning mechanisms have been developed at different levels of the network infrastructure to respond to the heterogeneity of network applications and clients in latency requirement, bandwidth consumption, among others.

Caching, as part of the network infrastructure, also needs to consider such QoS requirements. For instance, when making replacement decisions, most proxies only consider object size, access frequency, and retrieval latency, among others. They do not differentiate objects based on their original Web servers. However, in a QoS-aware application, for instance, such location information may play a central role because the QoS provision may provide preferential status to certain servers in the form of guaranteed hit rate.

Recent research has started to develop QoS-aware caching approaches. For instance, a weighted replacement policy has been proposed which provides differential QoS [73]. In more recent work, researchers have applied adaptive control to design differentiated caching services (e.g., [11]). The main advantage of an adaptive control-based approach is that control parameters can be automatically adjusted to achieve the desired quality differentiation without any manual tuning.

VI. Summary and Future Research Directions

This paper presents a survey of major technical approaches related to the design and operation of both single Web proxy caches and proxy cache networks. It also discusses several emerging areas of proxy caching that are of practical importance. We conclude this paper by summarizing a number of potentially fruitful areas of study for future Web caching research. These areas are grouped into three themes: integrated modeling, evaluation, and nontechnical issues.

1) Integrated Modeling: There is a clear trend in Web caching research to develop integrated models and operational guidance that combine two or more traditionally separate caching areas. Examples include the integration of prefetching and replacement [17], consistency and replacement [14], [74], and replacement and admission control [2]. We believe that research considering all of the major caching-related decisions in an integrated manner will provide important insights and concrete guidelines to Web cache design and operation.

Recent caching research has also gone beyond traditional caching topics to explore ways to further improve the Web performance. For instance, some caching systems employ intelligent sensing mechanisms to monitor network traffic to decide when to prefetch or perform consistency-related activities [35], [71]. Researchers have also started to explore how to best coordinate proxy caches and server-side solutions, such as surrogate proxies and content distribution internetworking, to improve the overall Web performance [5].

2) Evaluation: Evaluating a Web caching system in a comprehensive and objective manner is difficult. Complex tradeoffs often exist between different and often conflicting objectives. Future research is needed to investigate such complexities in a rigorous framework.

From the viewpoint of evaluation methodology, two lines of approaches are worth mentioning. The first line concerns analytically-driven evaluation. Such formal evaluation, albeit directly useful in only very restrictive settings, can provide important insights into the structure of the problem and often point to effective operational policies (e.g., [12], [13], [17]). A related formal framework is competitive analysis [75]. Competitive analysis techniques have been successfully applied to analyze computer memory paging systems and have the potential to be used to analyze Web caching approaches.

The second line is mainly concerned with empirical evaluation of caching systems. Both event-based simulation and trace-based evaluation have been extensively used in the current Web caching research [76]. Nonetheless, much of the empirical evaluation process is ad-hoc and lacks cohesion. Individual researchers often use simulation models and traces that are not publicly available, making comparisons between different approaches unnecessarily hard and unreliable. A common, publicly accessible set of benchmark simulated and real traces is urgently called for. In particular, experience has shown that an "one-size-fits-all" approach that works uniformly well across applications is almost impossible. Therefore, traces organized according to different application-motivated workload characteristics will provide a useful research platform.

3) Nontechnical Issues: Research focusing on nontechnical aspects of caching has started to emerge. For instance, cost-based models have been developed that go beyond operational aspects of caching and explore caching capacity planning issues [12], [77]. We expect continued research activities in this important area of study. Research exploring issues related to protocol adoption and data ownership and security in the context of caching, may also yield fruitful results.

References

[1] M. Zari, H. Saiedian, and M. Naeem, "Understanding and reducing web delays," *IEEE Computer*, vol. 34, pp. 30–37, Dec. 2001.

[2] C. Aggarwal, J. Wolf, and P. Yu, "Caching on the World Wide Web," *IEEE Trans. Knowledge Data Eng.*, vol. 11, pp. 94–107, Jan./Feb. 1999.

[3] M. Liu, F. Wang, and D. Zeng, "Web caching: a way to improve web qos," *J. Comput. Sci. Technol.*, vol. 19, no. 2, pp. 113–127, Mar. 2004.

[4] T. Takase, Y. Nakamura, R. Neyama, and H. Eto, "A web services cache architecture based on xml canonicalization," in *Proc. 11th Int. World Wide Web Conf. (Poster Paper)*, Honolulu, HI, May 2002.

[5] M. Rabinovich and O. Spatscheck, *Web Caching and Replication*. Reading, MA: Addison Wesley, 2002.

[6] V. Cardellini, E. Casalicchio, M. Colajanni, and P. S. Yu, "The state of the art in locally distributed web-server systems," *ACM Comput. Surv.*, vol. 34, no. 2, pp. 263–311, June 2002.

[7] J. Wang, "A survey of web caching schemes for the internet," *ACM Comput. Commun. Rev.*, vol. 29, no. 5, pp. 36–46, Oct. 1999.

[8] G. Barish and K. Obraczka, "World wide web caching: trends and techniques," *IEEE Commun. Mag. Internet Technol. Series*, May 2000.

[9] B. D. Davison, "A web caching primer," *IEEE Internet Comput.*, vol. 5, pp. 38–45, July/Aug. 2001.

[10] P. Cao, J. Zhang, and K. Beach, "Active cache: caching dynamic contents on the web," in *Proc. IFIP Int. Conf. on Distributed Systems Platforms and Open Distributed Processing (Middleware)*, 1998, pp. 373–388.

[11] Y. Lu, A. Saxena, and T. F. Adbdelzaher, "Differentiated caching services: a control-theoretical approach," in *Proc. IEEE 21st Int. Conf. on Distributed Computing Systems*, 2001, pp. 615–622.

[12] V. S. Mookerjee and Y. Tan, "Analysis of a least recently used cache management policy for web browsers," *Oper. Res.*, vol. 50, no. 2, pp. 345–357, March–April 2002.

[13] P. Rodriguez, C. Spanner, and E. W. Biersack, "Web caching architectures: hierarchical and distributed caching," in *Proc. 4th Int. Caching Workshop*, 1999, pp. 37–48.

[14] B. Krishnamurthy and C. E. Wills, "Proxy cache coherency and replacement-toward a more complete picture," in *Proc. IEEE Int. Conf. Distributed Computer System*, 1999.

[15] G. Lai, M. Liu, F.-Y. Wang, and D. Zeng, "Web caching: architectures and performance evaluation survey," in *Proc. IEEE Int. Conf. Syst, Man, Cybern.*, vol. 5, 2001, pp. 3039–3044.

[16] C. Brooks, M. S. Mazer, S. Meeks, and J. Miller, "Application-specific proxy servers as http stream transducers," in *Proc. 4th World Wide Conf.*, 1995, pp. 539–548.

[17] D. Zeng, F.-Y. Wang, and S. Ram, "Storage allocation in prefetching techniques of web caches (extended abstract)," in *Proc. ACM Conf. on Electronic Commerce*. San Diego, CA, June 9–12, 2003.

[18] J. Yin, L. Alvisi, and M. Dahlin, "Engineering web cache consistency," *ACM Trans. Internet Technol.*, vol. 2, no. 3, pp. 224–259, Aug. 2002.

[19] E. O'Neil, P. O'Neil, and G. Weikum, "The LRU-K page replacement algorithm for database disk buffering," in *Proc. ACM SIGMOD Int. Conf. Management of Data*, New York, 1993, pp. 297–306.

[20] K. Cheng and Y. Kambayashi, "LRU-SP: a size-adjusted and popularity-aware LRU replacement algorithm for web caching," in *Proc. 24th Annu. Int. Computer Software and Applications Conf.*, 2000, pp. 48–53.

[21] J. Robinson and M. Devarkonda, "Data cache management using frequency-based replacement," *Perf. Eval. Rev.*, vol. 18, no. 1, pp. 134–142, May 1990.

[22] S. Williams, M. Abrams, C. R. Standridge, G. Abdulla, and E. A. Fox, "Removal policies in network caches for World-Wide Web documents," in *Proc. ACM SIGCOMM Conf.*, CA, 1996.

[23] R. P. Wooster and M. Abrams, "Proxy caching that estimates page load delays," in *Proc. 6th World Wide Web Conf.*, Santa Clara, CA, Apr. 1997.

[24] S. Hosseini-Khayat, "On optimal replacement of nonuniform cache objects," *IEEE Trans. Comput.*, vol. 49, pp. 769–778, Aug. 2000.

[25] S. Jin and A. Bestavros, "Popularity-Aware Greedydual-Size Web Caching Algorithms," Comput. Sci. Dept., Boston Univ., Boston, MA, Tech. Rep. TR-99/09, 1999.

[26] P. Cao and S. Irani, "Cost-aware www proxy caching algorithms," in *Proc. USENIX Symp. Internet Technology and Systems*, Dec. 1997, pp. 193–206.

[27] L. Breslau, P. Cao, and L. Fan *et al.*, "Web caching and zipf-like distributions: evidence and implications," in *Proc. IEEE INFOCOM*, vol. 1, 1999, pp. 126–134.

[28] D. Zeng, F. Wang, and B. Fang, "Multiple-Queues: An Adaptive Document Replacement Policy in Web Caching," PARCS Lab, Univ. Arizona, Tucson, Tech. Rep. 03-0402, 2002.

[29] J. Yin, L. Alvisi, M. Dahlin, and C. Lin, "Volume leases for consistency in large-scale systems," *IEEE Trans. Knowl. Data Eng.*, vol. 11, pp. 563–576, July 1999.

[30] J. Howard, M. Kazar, S. Menees, D. Nichols, M. Satyanarayanan, R. Sidebotham, and M. West, "Scale and performance in a distributed file system," *ACM Trans. Comput. Syst.*, vol. 6, no. 1, pp. 58–81, Feb. 1988.

[31] C. Gray and D. Cheriton, "Leases: an efficient fault-tolerant mechanism for distributed file cache consistency," in *Proc. 12th ACM Symp. Operating Systems Principles*, 1989, pp. 202–210.

[32] H. Yu and L. Breslau, "A scalable web cache consistency architecture," in *Proc. ACM SIGCOMM*, 1999, pp. 163–174.

[33] V. Duvvuri, P. Shenoy, and R. Tewari, "Adaptive leases: a strong consistency mechanism for the world wide web," in *Proc. 19th Annu. Joint Conf. IEEE Computer and Communications Societies*, vol. 2, 2000, pp. 834–843.

[34] A. N. Eden, B. W. Joh, and T. Mudge, "Web latency reduction via client-side prefetching," in *Proc. IEEE Int. Symp. Performance Analysis of Systems and Software*, Apr. 2000, pp. 193–200.

[35] Z. Jiang and L. Kleinrock, "An adaptive network prefetch scheme," *IEEE J. Selected Areas Commun.*, vol. 6, pp. 358–368, Apr. 1998.

[36] T. M. Kroeger, D. D. Long, and J. C. Mogul, "Exploring the bounds of web latency reduction from caching and prefetching," in *Proc. USENIX, Symp. Internet Technology and Systems*, Dec. 1997, pp. 13–22.

[37] E. P. Markatos and C. E. Chronaki, "A top-10 approach to prefetching on the web," in *Proc. Internet Soc. Annu. INET Conf.*, 1998.

[38] I. Zukerman, D. W. Albrecht, and A. E. Nicholson, "Predicting users' requests on the www," in *Proc. 7th Int. Conf. on User Modeling*, 1999.

[39] M. Deshpande and G. Karypis, "Selective Markov models for predicting web-page accesses," in *Proc. SIAM Int. Conf. on Data Mining* , 2001.

[40] J. Dean and M. R. Henzinger, "Finding related pages in the World Wide Web," *Comput. Netw.*, vol. 31, no. 11–16, pp. 1467–1479, 1999.

[41] J. Pitkow and P. Pirolli, "Mining longest repeated subsequences to predict world wide web surfing," in *Proc. 2nd USENIX Symp. on Internet Technologies and Systems*, Boulder, CO, Oct. 11–14, 1999.

[42] J. Yang, W. Wang, R. Muntz, and J. Wang, "Dynamic Web Caching," Univ. California, Computer Science, Tech. Rep., Los Angeles, CA, Nov. 1998.

[43] X. Fang and O. Sheng, "Linkselector: a web mining approach to hyperlink selector for web portals," *ACM Trans. Internet Technol., Special Issue on Machine Learning for the Internet*, 2003.

[44] A. Pandey, J. Srivastava, and S. Shekhar, "Web proxy server with intelligent prefetcher for dynamic pages using association rules," Univ. Minnesota, Comput. Sci. Eng. Dept., Twin Cities, Tech. Rep. 01-004, 2001.

[45] R. P. Klemm, "Webcompanion: a friendly client-side web prefetching agent," *IEEE Trans. Knowl. Data Eng.*, vol. 11, pp. 577–594, July/Aug. 1999.

[46] T. Loon and V. Bharghavan, "Alleviating the latency and bandwidth problems in www browsing," in *Proc. Usenix Symp. on Internet Technologies and Systems*, Monterey, CA, 1997, pp. 219–230.

[47] S. Chakrabarti, *Mining the Web: Discovering Knowledge from Hypertext Data*. San Mateo, CA: Morgan Kaufmann, 2003.

[48] V. N. Padmanabhan and J. C. Mogul, "Using predictive prefetching to improve World-Wide Web latency?," in *Proc. ACM SIGCOMM Conf.*. Stanford, CA, 1996.

[49] Q. Yang and Z. Zhang, "Model based predictive prefetching," in *Proc. 12th Int. Workshop on Database and Expert Systems Applications*, 2001.

[50] H. Che, Z. Wang, and Y. Tung, "Analysis and design of hierarchical web caching systems," in *Proc. IEEE Conf. Computer Communications (INFOCOM)*, Anchorage, AK, Apr. 2001, pp. 1416–1424.

[51] M. Rabinovich, J. Chasse, and S. Gadde, "Not all hits are created equal: cooperative proxy caching over a wide-area network," *Comput. Netw. ISDN Syst. 30*, no. 22–23, pp. 2253–2259, Nov. 1998.

[52] A. Chankhunthod, P. B. Danzig, C. Neerdaels, M. F. Schwartz, and K. J. Worrell, "A hierarchical internet object cache," in *Proc. USENIX (the Advanced Computing Systems Association)*, 1996, pp. 153–164.

[53] M. Busari and C. Williamson, "Simulation evaluation of a heterogeneous web proxy caching hierarchy?," in *Proc. 9th Int. Symp. Modeling, Analysis and Simulation of Computer and Telecommunication System*, 2001, pp. 379–388.

[54] M. Makpangou, G. Pierre, C. Khoury, and N. Dorta, "Replicated directory service for weakly consistent replicated caches," in *Proc. 19th IEEE Int. Conf. on Distributed Computing Systems*, 1999, pp. 92–100.

[55] A. Wolman, G. M. Voelker, N. Sharma, N. Cardwell, A. R. Karlin, and H. M. Levy, "On the scale and performance of cooperative web proxy caching," in *Proc. ACM Symp. on Operating Systems Principles*, 1999, pp. 16–31.

[56] R. Tewari, M. Dahlin, H. Vin, and J. Kay, "Beyond Hierarchies: Design Considerations for Distributed Caching on the Internet," Dept. Comput. Sci., Univ. Texas at Austin, Austin, TX, Tech. Rep. TR98-04, Feb. 1998.

[57] L. Fan, P. Cao, J. Almeida, and A. Z. Broder, "Summary cache: a scalable wide-area Web cache sharing protocol," *IEEE/ACM Trans. Networking*, vol. 8, no. 3, pp. 281–293, 2000.

[58] D. Karger, E. Lehman, T. Leighton, M. Levine, D. Lewin, and R. Panigrahy, "Consistent hashing and random trees: distributed caching protocols for relieving hot spots on the world wide web," in *ACM Symp. Theory of Computing*, May 1997, pp. 654–663.

[59] K. W. Ross, "Hash-routing for collections for shared web caches," *IEEE Network*, vol. 11, no. 6, pp. 37–44, Nov./Dec. 1997.

[60] K.-L. Wu and P. S. Yu, "Latency-sensitive hashing for collaborative web caching," *Comput. Netw.*, vol. 33, pp. 633–644, 2000.
[61] V. Valloppillil and K. W. Ross, Cache Array Routing Protocol v1.0, Feb. 1998.
[62] A. Feldmann, R. Caceres, F. Douglis, and M. Rabinovich, "Performance of web proxy caching in heterogeneous bandwidth environments," in *Proc. IEEE Infocom Conf.*, New York, Mar. 1999.
[63] B. C. Housel, G. Samaras, and D. B. Lindquist, "Webexpress: a client/intercept based system for optimizing web browsing in a wireless environment," *Mobile Networks Applicat.*, vol. 3, pp. 419–431, 1998.
[64] F. Douglis, A. Haro, and M. Rabinovich, "Hpp: html macro-preprocessing to support dynamic document caching," in *Proc. USENIX (the Advanced Computing Systems Association)*, 1997, pp. 83–94.
[65] Z. Miao and A. Ortega, "Proxy caching for efficient video services over the internet," in *Proc. 9th Int. Packet Video Workshop*, New York, Apr. 1999.
[66] S. Jin, A. Bestavros, and A. Iyengar, "Network-aware partial caching for internet streaming data," *ACM Multimedia Syst. J.*, 2003.
[67] S.-H. Chan and F. A. Tobagi, "Distributed servers architecture for networked video services," *IEEE/ACM Trans. Networking*, vol. 9, no. 2, pp. 125–136, 2001.
[68] S. Paknikar, M. Kankanhalli, K. R. Ramakrishnan, S. H. Srinivasan, and L. H. Ngoh, "A caching and streaming framework for multimedia," in *Proc. ACM Multimedia*, 2000.
[69] S. Sen, J. Rexford, and D. Towsley, "Proxy prefix caching for multimedia streams," in *Proc. IEEE INFOCOM*, New York, Apr. 1999.
[70] Y. Chen, L. Qiu, W. Chen, L. Nguyen, and R. H. Katz, "Efficient and adaptive web replication using content clustering," *IEEE J. Selected Areas in Commun., Special Issue on Internet and WWW Measurement, Mapping, and Modeling*, 2003.
[71] L. Zhang, S. Michel, K. Nguyen, A. Rosenstein, S. Floyd, and V. Jacobson, "Adaptive web caching: toward a new caching architecture," in *Proc. 3rd Int. Caching Workshop*, June 1998.
[72] F. Wang, M. Liu, and D. Zeng, "Isaac: Intelligent Strategies and Architectures for Adaptive Caching," PARCS Lab, Univ. Arizona, Tucson, Tech. Rep. 01-0402, 2002.
[73] T. P. Kelly, Y. M. Chan, S. Jamin, and J. K. MacKie-Mason, "Biased replacement policies for web caches: differential quality of service and aggregate user value," in *Proc. 4th Int. Web Caching Workshop*, San Diego, CA, Mar. 1999.
[74] B. Krishnamurthy and C. E. Wills, "Piggyback server invalidation for proxy cache coherency," in *Proc. 7th World Wide Web Conf.*, 1998, pp. 185–194.
[75] A. Borodin and R. El-Yaniv, *Online Computation and Competitive Analysis*. Cambridge, U.K.: Cambridge Univ. Press, 1998.
[76] B. D. Davison, "A survey of proxy cache evaluation techniques," in *Proc. 4th Int. Web Caching Workshop*, Mar. 1999.
[77] J. Chuang, K. Hosanagar, and R. Krishnan, "Pricing caching services with multiple levels of QoS," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, Tucson, AZ, Oct. 2001.

Daniel Zeng received the M.S. and Ph.D. degrees in industrial administration from Carnegie Mellon University, Pittsburgh, PA, and the B.S. degree in economics and operations research from the University of Science and Technology of China, Hefei.

Currently, he is an Assistant Professor and Honeywell Fellow in the Department of Management Information Systems at the University of Arizona, Tucson. He is also currently directing four National Science Foundation (NSF)-funded research projects as PI or co-PI. His research interests include software agents and their applications, distributed optimization, computational support for auctions and negotiations, intelligent information integration and caching, and recommender systems. He has co-edited two books and published about 50 peer-reviewed articles in management information systems and computer science journals, edited books, and conference proceedings.

Dr. Zeng is a member of INFORMS, AAAI, and ACM, and serves on the editorial board of *Journal of Database Management*.

Fei-Yue Wang (S'87–M'89–SM'94–F'03) received the B.S. degree in chemical engineering from Qingdao University of Science and Technology, Qingdao, China, in 1982, the M.S. degree in mechanics from Zhejiang University, Hangzhou, China, in 1984, and the Ph.D. degree in electrical, computer and systems engineering from the Rensselaer Polytechnic Institute, Troy, NY, in 1990.

Currently, he is the Director of the Program for Advanced Research in Complex Systems at the University of Arizona, Tucson, where he has been since 1990. He became a Full Professor of Systems and Industrial Engineering at the University of Arizona in 1999. In 1999, he found the Intelligent Control and Systems Engineering Center at the Institute of Automation, Chinese Academy of Sciences, Beijing, China, under the support of the Outstanding Oversea Chinese Talents Program. Since 2002, he has been the Director of the Key Laboratory of Complex Systems and Intelligence Science at the Chinese Academy of Sciences. His current research interests include modeling, analysis, and control mechanism of complex systems; agent-based control systems; intelligent control systems; real-time embedded systems, application–specific operating systems (ASOS); applications in intelligent transportation systems, intelligent vehicles and telematics, web caching and service caching, smart appliances and home systems, and network-based automation systems. He has published many books, book chapters, and papers in those areas since 1984 and has received more than $20 M USD and over ¥=50 M RMB from NSF, DOE, DOT, NNSF, CAS, Caterpillar, IBM, HP, AT&T, GM, BHP, RVSI, ABB, and Kelon. He received Caterpillar Research Invention Award with Dr. P.J.A. Lever in 1996 for his work in robotic excavation and the National Outstanding Young Scientist Research Award from the National Natural Science Foundation (NSF) of China in 2001, as well as various industrial awards for his applied research from major corporations. He was the Editor-in-Chief of the *International Journal of Intelligent Control and Systems* from 1995 to 2000, and currently is the Editor-in-Charge of Series in *Intelligent Control and Intelligent Automation,* Editor for the ITS Department of the *IEEE Intelligent Systems*, and an Associate Editor of the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION, IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, and several other international journals.

Dr. Wang is an elected member of IEEE SMC Board of Governors and IEEE ITSC AdCom, and the Secretary and Vice President of IEEE Intelligent Transportation Systems Council. He was the Program Chair of the 1998 IEEE International Symposium on Intelligent Control, the 2001 IEEE International Conference on Systems, Man, and Cybernetics, the General Chair of the 2003 IEEE International Conference on Intelligent Transportation Systems, and the Co-Program Chair of the 2004 IEEE International Symposium on Intelligent Vehicles, and will be the General Chair for the same conference in 2005. He was the Vice President and one of the major contributors of the American Zhu Kezhen Education Foundation.

Mingkuan Liu received the M.S. degree in electrical engineering from the Chinese Academy of Sciences, Beijing, China, in 2000, and the M.S. degree in industrial engineering from the University of Arizona, Tucson, in 2002. He is currently pursuing the Ph.D. degree in the Electrical and Computer Engineering Department at the University of Arizona.

His research interests include intelligent control, speech recognition, voice over IP, web caching, and service caching.