Chapter 13

Looking Ahead

13.1 Introduction

To conclude this work, we first ask what have we learned over the preceding chapters. We summarize the key contributions below. We then consider the next steps — how the major systems in this thesis can be extended in the future, and what is needed to make prefetching commonplace in the Web. Finally, we will offer some thoughts about the future of the broader task of improving the performance of content and service delivery.

13.1.1 User action prediction

In the first part of the dissertation, we focused on the task of predicting the next action that a user will take. *We introduced a simple approach called IPAM for predicting the UNIX shell commands issued by a user, and realized a predictive accuracy of approximately 40% over two independent datasets.* We also specified the characteristics of an idealized prediction algorithm.

Later we considered specifically the challenges involved in predicting Web requests from history. We presented many domain-specific choice-points, as well as the details of various Markov-like prediction models. We implemented generalized prediction codes to model the various algorithms, and compared their performance on a consistent set of Web request traces (realizing predictive accuracies from 12-50% or better, depending on the dataset). *The identification, implementation, and evaluation of Markov-based predictions based on the history of actions taken by one or more users* is one of our key contributions. The best performing approaches included predictions from multiple

contexts (as in PPM – prediction by partial match).

In addition to prediction methods using history-based techniques, we also considered the complementary idea of content-based prediction. First *we experimentally validated widespread assumptions of textual similarity in nearby pages and descriptions on the Web*, and then used those assumptions to implement a content-based prediction approach and tested it on a full-content Web usage log. Another key contribution is *the evaluation of the use of content-based prediction methods* in which we found a content-based approach to be 29% better than random link selection for prediction, and 40% better than not prefetching in a system with an infinite cache.

13.1.2 Evaluation of Web cache prefetching techniques

While the initial chapters dealt with the concerns of designing and evaluating prediction techniques in isolation, later chapters concentrated on evaluating those techniques within the environment of a prefetching system. We first introduced the topics of concern to evaluators, and considered specifically the task and common techniques for proxy cache evaluation.

In order to estimate client-side response times, we implemented and *validated a new Web caching and network simulator.* In contrast to testing predictive accuracy as we did earlier, the use of the simulator allowed us to evaluate prediction mechanisms embedded within a prefetching environment. Using NCS, *we demonstrated the potential performance improvements resulting from the use of multiple prediction sources for prefetching.* We showed some scenarios in which a combination of prediction sources can provide better performance than either alone, and found that client prefetching based on Web server predictions can significantly reduce typical Web response times, without excessive bandwidth overhead. We also presented some analysis of when history-based prefetching will not be particularly effective. In general, NCS provides a useful framework to test other network and caching configurations with real-world usage traces.

Since simulation is not helpful to evaluate proxy cache implementation, *we proposed a novel architecture called Simultaneous Proxy Evaluation that can optionally utilize live traffic and real data to evaluate black-box Web proxies.* Unlike existing evaluation

methods, *SPE is able to objectively evaluate prefetching proxies* because it provides access to real content and live data for speculative retrieval. *We then implemented the SPE architecture, discussed various issues and their ramifications to our system, and demonstrated the use of SPE in validation tests and experiments using SPE to evaluate five publicly available proxy caches.*

Finally, *we identified the need to change HTTP because of fairness and safety concerns with prefetching using the current standard.*

13.2 The Next Steps

The past chapters have dealt with the mechanisms to predict user actions on the Web for prefetching, and to evaluate proposed algorithms and implemented prefetching systems.

In fact, multiple approaches to predict Web activity were proposed, implemented, and evaluated. Multiple approaches to evaluate Web prefetching algorithms were proposed and implemented. We showed that higher predictive accuracy and lower client-side response times can be accomplished by combining the results of separate prediction systems.

However, there are some aspects that were not able to be addressed by this dissertation, and so are directions for future work. Such issues include:

- Congestion modeling and packet loss in NCS. The simulator currently provides for the equivalent of infinite queues, which are unrealistic. However, modeling the network at this level would likely also require modeling routers and additional TCP protocol detail, rendering it much slower.

- Proxy evaluation utilizing SPE on a live request stream. While designed to be able to handle live user requests, the experiments using SPE presented in this dissertation used artificial and captured traces. We look forward to the opportunity to use SPE to conclusively evaluate multiple proxies on a live request stream.

- Evaluation of caching and prefetching implemented in browsers. While caching and prefetching in browsers is considered within NCS, the SPE architecture is not

directly suited to the evaluation of client-side systems. One potential alternative is to break out whatever system is being added to a browser (such as the caching and prefetching) and place it into a separate proxy that can then be tested using SPE.

- The cooperation of proxies, whether in a mesh, a hierarchy or as part of a content delivery network. Such systems of proxy caches may be able to provide significant benefits, but would not, for example, be amenable to evaluation using the SPE architecture since SPE would be unable to wrap itself around a truly distributed CDN. However, it may be possible to locate the Multiplier with the load generator, and the Collector at the origin server. If the CDN can be configured to access the Collector instead of the origin server, then SPE could indeed be applied to the constrained task of evaluating response times for requests for a particular server.

- Evaluation of larger scenarios in NCS. Presently NCS builds all of its data structures in memory, limiting possible simulation sizes to the amount of memory supported by the OS to be assigned to an application. While likely to be considerably slower, a disk-based simulator would have the ability to simulate longer and larger scenarios. Larger scenarios might enable simulation of CDN-style sets of proxy caches or test theories of what would happen "if everyone were to prefetch" with some level of aggressiveness.

- The use of a SPE implementation in which clients are not explicitly configured to access the Multiplier. Many proxy caches are deployed to intercept Web requests transparently and thus serve responses without the knowledge of the client. With additional equipment and small changes, SPE could be implemented in this fashion as well, and to be able to evaluate proxies so configured.

There are two reasons why the multi-source prefetching explored in this dissertation cannot be put on the Web today. The first is that some requests on the Web have side effects, as discussed in Chapter 12. The second is that prediction hints (sent, in general from server to client) need to be transported under some standardized mechanism. As demonstrated in this thesis and elsewhere, a Web server or

proxy that has seen many requests for an object will have a better model of what is likely to be requested next. Therefore, in a client-based prefetching system, the predictions need to be sent to the client in some form. A number of researchers have proposed the use of hints of some kind for caching, invalidation, or prefetching [Pad95, PM96, Mog96, CKR98, KW97, KW98b, Duc99]. While the hints themselves could be delivered through HTTP using a particular unique URL, the client still needs to know the identity of the appropriate URL that would contain hints of actions likely to follow a particular object retrieval. One possibility is to calculate the hints URL by the use of a predetermined URL transformation function. Such an approach might take the original URL and add a prefix or suffix, or generate an opaque identifier such as by an MD5 checksum. However, those approaches limit capability with objects that may already be served using such URLs that would result from the transformation. Another, more reasonable and commonly proposed approach is to include the hints or a pointer to them, in the headers of an HTTP response. While not presently standardized, making such a standard does not pose any significant technological hurdles. Both of these difficulties would be obviated with appropriate modifications to HTTP.

13.3 The Future of Content Delivery

In order to provide useful services to massive numbers of clients, the popular network services of today and tomorrow require large-scale yet economical solutions. For aspiring companies, it sometimes does more harm than good to be mentioned on a very high traffic site (e.g., the Slashdot effect [Adl99]). As a result, even smaller-scale services need the ability to scale to large numbers of clients. Instead of the obvious (but expensive and often complex) approach of placing Web servers or caches in distributed hosting centers, content distribution networks are typically the recommended solution.

CDNs have become well-established parts of the network infrastructure. Some are presently independent companies, such as Speedera [Spe02] and Akamai [Aka02], which operates tens of thousands of systems located in thousands of ISP and university facilities. Others operate as part of a network or Web hosting provider, such as AT&T's

services [ATT02], or Digital Island [Cab02] which is presently part of Cable & Wireless's Exodus services. Finally, some use alternative communication methods to deliver content to caches, such as Cidera [Cid02] which uses satellite delivery. CDN services have become valuable mechanisms by which content providers can ensure better Web site performance for their customers.

In effect, CDNs allow customers to share the infrastructure with many other sites and leverage the low probability of a simultaneous surge in traffic to the shared sites. However, CDNs may not be tomorrow's solution, as dynamic content and other services gain in popularity. CDNs have already begun the process of change — they are no longer limited to static Web content, but now can serve various streaming media and some dynamic content (e.g., through the use of Edge Side Includes [OA01]). Increasing support for Sun's J2EE and Microsoft's .NET will augment the ability to provide dynamic Web services from CDN edge platforms (e.g., [VM02]). Thus, the generation of some dynamic content at the edge is just part of a progression, leading to not just caching at the edge, but computation at the edge.

However, while distributing content and some computation to the edge of the well-connected Internet can significantly improve user-perceived performance, it does not directly address the last-mile problem. The user's connection to the Internet is typically the bottleneck, generating the bulk of latencies and transmission delays. To improve this aspect of Web performance, caching, prefetching, and computation must move to the client. In some ways, that is already happening — browsers already cache content and connections, and typically implement JavaScript and a Java environment for applets, thus moving some computation to the client. However, with the cooperation of the major browser manufacturers, much more could be done with prefetching (as shown in this dissertation) and with client-side computation.

Caching and computation at the edge (wherever you define it) are important parts of the solution to improve performance of some applications. Others include:

- Caching and pre-loading of non-Web traffic. DNS is naturally cached by clients

and servers, and while we have mentioned its effects, it is just one of many non-Web protocols that benefit from caching. For example, streaming media such as audio or video can be and is distributed using specialized proxy caches.

- Caching of all network traffic. A few researchers and companies have proposed mechanisms for the caching and compression of network traffic at the packet or byte level [SW98, SW00, Exp02]. Such approaches have the potential to reduce the benefit of passive caches, but prefetching will still be valuable.

- Other techniques to reduce response times. Many other research areas affect user-perceived latencies on the Web, including routing, quality of service (QoS) efforts, and improvements in last-mile networking.

In summary, researchers and developers concerned with optimizing content and service delivery will need to consider a larger view — and not just optimize one portion of the delivery process.

13.4 Summary

Prefetching has been shown (both in this thesis and elsewhere) to be an effective mechanism to reduce client-side response times in the Web. Why is this? In Chapter 2, we mentioned a number of factors supporting the need for caching research. Prefetching can successfully combat the same sources of latencies (such as network congestion, speed-of-light delays, etc.) because it takes advantage of:

- User thinking time which can be overlapped with retrieval. The idle time between requests provide resources that can be speculatively exploited.

- Multiple sources that can provide content speculatively. By retrieving from multiple sources, we can increase the utilization of otherwise idle resources, making the system more efficient.

- Repeated activity by people or systems. Both people and systems repeat sequences of actions, and such patterns of activity can be recognized and exploited by capable systems for prefetching.

Given these arguments, why is Web prefetching not prevalent? While there may be many reasons, one important aspect is the lack of standard mechanisms to recognize non-demand HTTP requests (as we described in Chapter 12) and the transport of prediction hints (as described above). As a result, Web prefetching is more likely to be found within closed systems (e.g., a CDN) in which specialized headers or mechanisms can be deployed without concern for standards.

This final chapter has summarized the contributions of the previous chapters and has speculated on the future of content and service delivery. Given appropriate mechanisms for measurement such as NCS or SPE, prefetching systems can demonstrate significant contributions in the quest to improve user-perceived performance. While prefetching is just one tool to help improve performance, it is one that should not be ignored.