Summary

Performance Guarantees for Web Applications

Users are increasingly demanding for responsive Web applications. A survey from 2006 revealed that 62% Web users were willing to wait only 6 seconds or less for a single page load before they leave the web site. A more recent research (2009) indicated that this performance expectation has become more demanding as 83% Web users expected a Web page to load in 3 seconds or less. In addition, this research also found that 79% of online shoppers who visited an underperforming Web site were likely not to buy from that site. Obviously, performance guarantees for Web applications are business-critical.

One important performance metric is the response time of a Web application. The response time can be split into three parts: client-side latency, network delay, and server-side latency. Recently, Web applications started employing client-side codes, such as JavaScript, to enrich application features. Client-side latency refers to the time used to execute client-side code. The research community has made efforts to address several client-side performance issues, such as JavaScript runtime behavior investigations for improving the representativeness of performance benchmark suites, remote monitoring for client-side performance diagnosing, and JavaScript performance optimizations by trace-based just-in-time compiler. In the ICT industry, the browser war among various vendors also targets JavaScript performance improvement for a major part. Client-side latency mainly depends on two factors: application client-side code and specific mechanisms built in each Web browser. From the perspective of Web hosting providers, these two factors are beyond their controllable scope.

Network delay refers to the transmission time of a request's response from the server to the client over a network such as the Internet. Various techniques, such as edge computing, data caching, and data replication have been proposed to reduce these delays. Commercial products such as Akamai CDN and Amazon CloudFront are also available for guaranteeing the best possible access performance. These

academic and industrial efforts work together to significantly reduce the network delay incurred by Web applications, and have been quite successful.

Though optimizing client-side latency and network delay is important, we cannot guarantee performance of a Web application unless its server-side latency is also under control. For instance, previous experiments showed that service-side latency could account for nearly 50% of the end-to-end latency of a Web application. As Web applications continue to become more complex, we can only expect server-side latency to increase. Server-side latency refers to the residence time of an incoming request waiting for its response at the server. For instance, a typical Web application consists of a business-logic tier and a data tier. The business-logic tier may be deployed on an application server while the data tier is often deployed on a database server. The server-side latency includes the time of executing application code at the application server and the time of fetching data from the database server.

Guaranteeing server-side Web application performance is made difficult by the fact that Web application workloads are fluctuating and highly unpredictable. The unpredictability and fluctuations introduce two important demands for the hosting system. First, a Web-application architecture must be ready to accommodate arbitrary levels of load. Second, it must be capable of adjusting its own capacity to support fluctuating Web traffic.

On the one hand, given that Web traffic is unpredictable, one cannot predict the maximum workload a Web application will receive in advance. Meanwhile, Web application providers aim at attracting as many users as possible for potentially growing business benefits. Therefore, a Web application needs to be scalable. A scalable Web application is capable of handling arbitrary levels of traffic by adding resources while sustaining reasonable performance. However, constructing a scalable Web application is nontrivial in that it requires careful partitioning of both business-logic tier and data tier.

On the other hand, the fluctuations of Web application workload make it impossible to plan "proper" fixed hosting resource capacity at minimal resource cost. Cost-sensitive Web application providers, for example small and medium-sized providers, expect a cost-effective manner to host their applications. By applying the utility computing model to Web-application hosting and varying the number of resources Web applications use according to the current load, application providers can expect reducing their costs.

Utility computing provides a model of packaging computing resources as a metered service. Since year 2000, IT providers have been making efforts to develop products and services to implement utility computing model in computer clusters and data centers. Recently, cloud computing started applying utility computing by provisioning resources in a pay-as-you-go manner. In clouds, resources such as computation, storage, and network are rented as services and charged by usage. The utility computing model facilitates dynamic resource provisioning for Web applications to handle varying resource demands. However, efficient dynamic resource provisioning faces challenges from both Web applications and hosting environments. This brings us to the central research question of this thesis: how to guarantee the server-side performance for Web applications in a cost-effective manner.

This thesis uses the server-side average response time as the Web-application performance measurement. Other performance metrics, such as percentiles of the response time, are also useful for performance guarantees. We believe that our techniques can be extended to support such metrics. The issue of guaranteeing server-side performance can be translated into sustaining reasonable average response time for Web applications under fluctuating traffic. A reasonable response time is defined to be under the maximum response time within which an application should finish processing an incoming request. Web application providers usually set this maximum response time in their Service Level Objectives (SLOs).

Besides choosing performance metrics, this thesis uses the number of utilized machines as the cost measurement. A utilized machine can be either a dedicated physical machine in a cluster or a virtual machine in a cloud. The number of machines can be further translated into the monetary cost if given the charging price.

This thesis mainly involves two aspects of research efforts to address our central research question: i) constructing a scalable Web application architecture; and ii) designing dynamic resource provisioning systems.

Constructing a scalable Web application can be done in two main ways: scale-up and scale-out. Scale-up means adding more capacity, such as CPU speed and memory size, to individual application servers and database servers. In contrast, scale-out means adding more servers to the two tiers. Scale-out outperforms scale-up when the performance/cost ratio is concerned for Web applications. Scale-up also has a hard limit by the scale of the hardware while scale-out allows to continuously add resources. Therefore, we construct a scalable Web application architecture by using scale-out techniques in this thesis.

Adding more servers to the business-logic tier of a Web application can improve the performance by alleviating the workload addressed to each individual server at that tier. However, adding more servers to the data tier cannot always improve the performance of that particular tier under arbitrary levels of load. Partial database replication, careful data partition and placement, allow improved scalability of the data tier through adding more resources. However, the coarse partition granularity limits the scalability extent of current scaling techniques. In this thesis, we show the potential scalability of Web applications resulted from finer data-partition granularity.

Though a scalable Web-application architecture provides promising mechanisms for guaranteeing the performance of Web applications, Web applications still face the issue of fluctuating traffic. Over-provisioning Web applications according to the peak workload can result in inefficient resource usage while under-provisioning creates a risk of violating SLO. The most straightforward technology used to guarantee performance for Web applications under fluctuating traffic is dynamic resource provisioning. This technology consists of adding extra resources to a Web application when its response time is close to violating its SLO, and removing underutilized resources from a Web application with retaining its SLO.

Unfortunately complex Web applications and heterogeneous hosting environments challenge current dynamic resource provisioning techniques. On the one hand, current Web applications are not designed as a monolithic 3-tier application. For instance, the Web application used to generate Web pages of Amazon.com consists of hundreds of services. It is hard to figure out the performance bottleneck within such applications that consist of multiple interacting services. It is even harder to handle this issue by dynamically and efficiently provisioning resources. On the other hand, heterogeneous physical machines and virtual machines in data centers and clouds result in performance heterogeneity of virtual hosting resources. This feature also limits the applicability of current resource provisioning techniques that assume the existence of homogeneous underlying resources.