WADN: Web Application Delivery Network

Jungsook Kim[1], *SungJae Jo*[1], *Junehwa Song*[1], *Minyeol Lim*[2], *Hyngwoo Park*[2]

1: Network Computing Labaratory
KAIST, Guseong-Dong, Yuseong-Gu, Daejeon, 305-701, Korea
{sun, stanleylab, junesong}@nclab.kaist.ac.kr

2: KISTI, Eoun-Dong 52, Yuseong-Gu, Daejeon, 305-806, Korea
mylim@hpcnet.ne.kr, hwpark@kisti.re.kr

Abstract

The high dynamicity of e-Business environments causes the merchants to confront the sudden increase in loads. In this paper, we propose WADN (Web Application Delivery Network), a new service supporting dynamic scalability for Web application servers. Using WADN, system capacity is adjusted to workload by expanding additional servers dynamically. Also, WADN help minimize the total number of currently used Web application servers while stably serving customers. Thus, merchants can reduce the total cost of ownership.

1. Introduction

With rapid expansion of the Internet technology, e-Business becomes an important part of business activity. The notable characteristics distinguishing e-Business from traditional one is as follows. First, business requests tend to be temporally skewed. People often share interests to common business events in similar periods of time, which is remarkable on the highly accessible Internet. Yet, it is difficult to predict such burst requests. Second, dynamic contents are taking a significant portion of e-Business contents. Merchants realize several advanced services, e.g., personalized information presentation and database searching, via dynamic contents flexibly handling customer's demand.

The e-Business characteristics incur a serious problem of overloading servers frequently and abruptly. In such an overloaded situation, both customers and merchants suffer from bad performance. Once incoming requests exceed server capacity, response time and connection error rate explosively increase and this terrible situation affects all customers. The delayed response time and high connection error rate can impose a serious damage even on a major merchant site. A recent report from Zona Research [1] showed that with response time of less than 7 seconds, e-commerce sites find 7% abandonment rate, whereas this rate sharply increases to 70% with response time greater than 12 seconds.

To efficiently address the overloading problem, we propose a new service called "Web Application Delivery Network" (WADN). WADN enables the system capacity to be adaptive by expanding the system dynamically. That is, the numbers of servers comprising the system are adjusted to workload dynamically and automatically. In order to participate in WADN, each server should deploy WADN core. Then, it turns into a WADN node. The WADN node continuously monitors its load and senses the overload. When the WADN node forecasts to be overloaded in the near future, it will join to a new WADN node (after we call it a target WADN node) to the system in runtime. In Join Process, WADN node automatically delivers and deploys Web applications needed for serving requests. When the total system load is reduced to the normal level, the system releases the target WADN nodes.

The WADN provides two advantages. First, the WADN enables merchants to reduce their total cost of ownership while they stably serve customers even during the peak request periods. Merchants rent some WADN nodes during rush time and pay for them on time-based. Second, the WADN efficiently uses the Internet resources. WADN uses idle WADN nodes in order to distribute the load of busy system.

Server clustering [2] and proxy cache server [3, 4] are the most commonly used approach to increase the server capacity and partly address the problem of overloading server easily. Using Server clustering, the merchants should guess at amount of resources enough to serve peak request rate. Based on the estimation, the merchants invest big money in system resources, which may be underutilized most of time. Thus, this approach is inefficient and the estimation is likely to be inaccurate. Proxy cache intercepts and serves customer requests in an intermediary. Thus, it makes the server capacity seem increased. The approach is good at handling static content. However, for dynamic contents, it is still immature.

The paper is structured as follows. In section 2, we introduce a new scalable service, called WADN. In section 3, we describe a WADN component in detail. We conclude in section 4 with a discussion of WADN

2. Architecture

In this section, we introduce WADN and the architecture of the WADN node. Figure 1 shows the WADN. In the WADN, WADN nodes are flexibly coupled with others and forms dynamically expansible virtual merchant system. Each WADN node can belong to several virtual merchant systems.

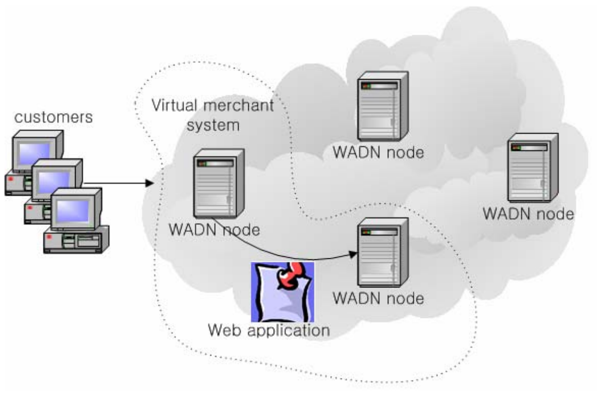

Figure 1. Web Application Delivery Network

The WADN node is composed of a Web server, an application server and WADN core. The overall architecture of the WADN node is illustrated in Figure 2. The components of WADN core are Request Processor, System Manager, Delivery Manager, Packer & UnPacker, Source Converter and Deliverer. We explain a rough role of the components and operations between them. Following describes the steps that WADN node takes for dynamic Web application delivery:

- A merchant system receives an initial request.
- If the request requires dynamic content generation, front-end Web server relays it to Request Processor of WADN core. Based on its load, Request Processor decides whether its own application server process it or not.
- When the load is under the threshold, Request process relays it to its own application server. Otherwise, Request Processor orders that Delivery Manager add a new WADN node.
- Then, Delivery Manager transmits and deploys a Web application needed for serving customer requests using Converter, Packer, and Deliverer.
- After the delivery, the target WADN node is ready to serve and the virtual merchant system capacity is finally increased.

The WADN core is located in between a Web server and an application server. This approach enables the WADN system to take dynamic expansion without any changes to the web server and the application server. A server can easily become a WADN node by changing the connector module between the Web server and the application server.

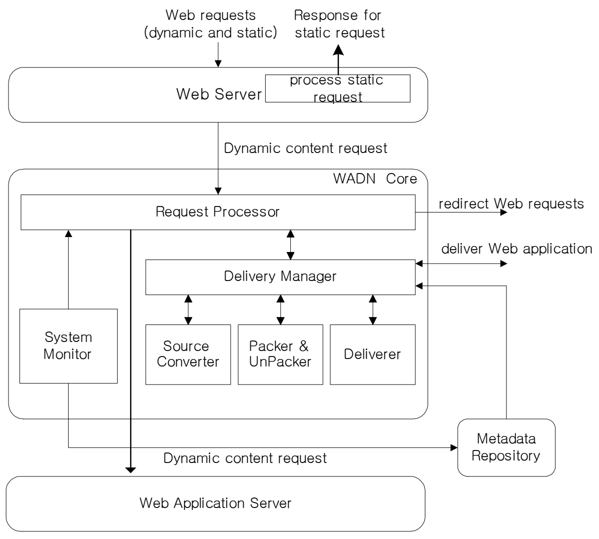

Figure 2. Architecture of the WADN node

3. Design

In this section, we discuss each component of the WADN node. Our goal in designing the components was to high performance and automatic application delivery and deployment.

3.1 Request Processor

Based on the delivery policy and monitored load information, the Request Processor decides whether it adds a new WADN node on the virtual merchant system or not. If the system needs another WADN node, Request Processor notifies this decision to Delivery Manager. Then, Delivery Manager selects the target WADN node.

For easy session management, we propose that the Request Processor performs request delivery decision based on each incoming user sessions rather than on an individual request. Session based control is important because many web-based services are transactional in nature, and consist of many requests to the web service. Under overload situation, session-based control schemes allow existing user sessions to continue, while new sessions are redirected to the target WADN node. Our approach is effective and reasonable because most sessions last less than 1000sec and have less than 10 requests except agents or robots.

3.2 System Monitor

The system Monitor checks the load status of the system resources occupied by Web services on a regular interval; CPU, I/O, memory, and network resource, and etc. The detected load information is used in Request Processor to make a decision over the delivery and is saved in a metadata repository.

We propose a flexible load monitoring services that can be extended to support new load metrics, as well as different policies to collect such metrics. In a Web server, the load is directly proportion to the number of connections. However, the load of the Web application server is changed according to a kind of Web applications and services. A System Monitor implements a strategy for monitoring loads on given resources. The interface for reporting loads to the Request Processor and meta-data repository remains unchanged for each System monitoring strategy. Strategizing load monitoring makes it possible to use a Request Processor that is not specific to a particular type of load, such as CPU load or I/O load. Thus, a Request Processor need not be specialized for a given type of load. This design simplifies deployment of all kinds of Web application since one Request Processor can process many different types of load.

3.3. Delivery Manager

The Delivery Manager makes decisions for a target WADN node selection and application delivery. According to selection strategy, the Delivery Manager selects the target WADN node which might be a lot and scattered on the world with different loads. The selection strategy is composed of load status, location, and Web application existence information stored in meta-data repository. The selected target WADN node finally joined to the virtual merchant system only after getting a valid Web application.

3.4 Source Converter

The Source Converter enables Web applications to be portable. In our design and implementation, we suppose the Web applications are Java Servlet and Java Server Pages. Thus the Web application code is platform independent if servers have Java Virtual machine. However, Web applications usually access external resources such as a database and a file server, and they may be different according to servers. The Source Converter modifies the portion of server dependent codes so that the delivered Web application can be executable on the target WADN node.

3.5 Packer & UnPacker

Packer and UnPacker respectively marshals and unmarshals a deliverable Web application. Usually, the Web application is composed of many components such as, images, html page generating codes and etc. Packer gathers the components related to the deliverable Web applications and packs them. And then for efficient delivery, the Packer compresses them. The Packer has two alternatives in design; source trace mechanism and previous deployment mechanism. For effective and easy delivery, we choose the second method in our implementation. We pack and deploy all components related to Web application at the origin WADN node in advance. This significantly reduces searching overhead performed in every delivery.

UnPacker does reverse process of the Packer. UnPacker unpacks and deploys delivered packages in the WADN node.

3.6 Deliverer

Deliverer transmits the packed Web application to the target WADN node. When the Delivery Manager decides to deliver the Web application to the target WADN node, the Delivery Manager calls the Deliverer. After the delivery, the target WADN node is ready to serve and the virtual merchant system capacity is finally increased.

4. Conclusion

Burst requests for dynamic contents have been resulting in heavy overload on Web Application servers. In this paper, we have proposed the WADN supporting the dynamic scalability with dynamic server adaptation. Joining to WADN, all merchant systems can be robust systems separated from burst request increase. Thus, every merchant in WADN can always provide stable services.

We plan to continue our work in several directions. One is fully implementing the WADN system. We are also investigating other advanced services which can be realized by using the WADN.

5. Reference

[1] www.zonaresearch.com

[2] V.S. Pai, M. Aron, G. Banga, M. Svendsen, P. Druschel, W. Zwaenepoel, and E. Nahum, “Locality-Aware Request Distribution in Custer-based Network Servers”, Proceedings of the Eighth Symposium on Architectural Support for Programming Languages and Operating Systems, pp. 205-216, October 1998.

[3] Pei Cao and Sandy Irani, “Cost-Aware WWW Proxy Caching Algorithms”, in Proceedings of the 1997 USENIX Symposium on Internet Technology and Systems, pp. 193-206, Dec 1997

[4] Martin Arlitt, Ludmilla Cherkasova, John Dilley, Rich Friedrich, and Tai Jin, “Evaluating Content Management Techniques for Web Proxy Caches”, in Second Workshop on Internet Server Performance, in conjunction with ACM SIGMETRICS 99, Atlanta, GA, May 1st, 1999