Improving Performance of Cluster-based Secure Application Servers with User-level Communication *

Jin-Ha Kim, Gyu Sang Choi and Chita R. Das
Department of Computer Science and Engineering, The Pennsylvania State University
{jikim,gchoi,das}@cse.psu.edu

1 Introduction

State-of-art cluster-based data centers consisting of three tiers (web server, application server and database server) are being used to host complex Web services such as e-commerce applications. The application server handles dynamic and sensitive Web contents that need protection from eavesdropping, tampering and forgery. Although the Secure Socket Layer (SSL) is the most popular protocol to provide a secure channel between a client and an application server, its high overhead degrades the server performance considerably and thus, affects the server scalability. Therefore, improving the performance of SSL-enabled application servers is critical for designing scalable and high performance data centers.

In this paper, we examine the impact of SSL offering in cluster-based application servers and propose a backend forwarding scheme, called *SSL_with_bf*, that employs a low-overhead user-level communication mechanism like VIA to achieve good load balance among server nodes. We compare three application server models: Round Robin (RR), *SSL_with_session* and *SSL_with_bf* through simulation. The experimental results with 16-node cluster configurations show that while session reuse of *SSL_with_session* is critical to improve the performance of application servers, the proposed backend forwarding scheme can further enhance the performance due to better load balancing. The *SSL_with_bf* scheme can minimize average latency by about 30% and improve throughput across a variety of workloads.

2 Application Server Models

In this paper, we study three different distributor models (*Round Robin*, *SSL_with_session* and *SSL_with_bf (backend_forwarding)*) to analyze the performance of application servers.

- **Round Robin:** In this model, a distributor does not consider the SSL session of clients, and the requests are distributed by using the *RR* policy. Thus, requests from the same client are distributed to several nodes, and SSL setup is required for each connection.

- **SSL_with_session:** The distributor of *SSL_with_session* maintains the client information to forward subsequent requests from the same client to the same application server. The advantage of the *SSL_with_session* is that it avoids the unnecessary authentication and negotiation phase when a client tries to reconnect to the server. The disadvantage of this model is that it may cause load imbalance among the servers.

- **SSL_with_bf (backend_forwarding):** The *SSL_with_bf* is aimed at mitigating the limitation of *SSL_with_session* by using a backend forwarding mechanism to achieve load balance. The load is calculated by the number of open connections and the CPU utilization of a server. Whenever a highly loaded server receives a request from a client, it forwards the request along with a negotiated session key to a lightly loaded server. The server, which receives the request from another node, generates and encrypts the dynamic contents using the forwarded session key. Finally, it returns the reply to the initial node, which sends the response back to the client.

Figure 1 depicts the *SSL_with_bf* model. Application servers in the cluster are interconnected through a SAN. Steps 2 and 3 in the Figure show the backend forwarding scheme between the cluster nodes.

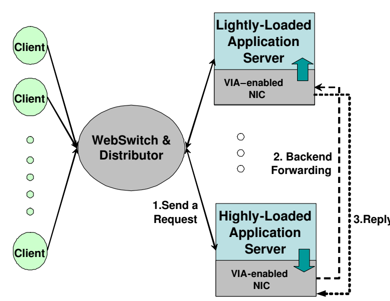


Figure 1. An Application Server Layer Model with the Backend Forwarding Scheme

3 System model

We have developed a comprehensive simulation testbed to evaluate the application server design alternatives. The simulator written in CSIM, consists of two major components; a communication module, an integrated a Web server and application server module. The VIA communication module [3] communicates with the NIC device driver for intra-cluster communication.

*This research was supported in part by NSF grants CCR-9900701, CCR-0098149, EIA-0202007 and CCR-0208734.

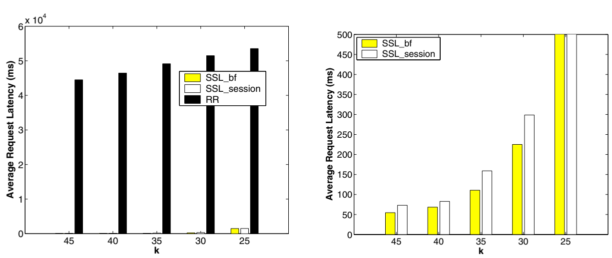
Table 1. Distribution Models Used in Simulation

Parameter	Description	Value
Distribution of objects (body)	Lognormal distribution	$P(x) = \frac{1}{x\sigma\sqrt{2\pi}} \times e^{\frac{-(\ln x - \mu)^2}{2\sigma^2}} \mu = 1.16, \sigma = 0.9, x \leq 10$
Distribution of objects (tail)	Pareto distribution	$P(x) = \alpha k^{\alpha} / ((1 - (\frac{k}{p})^{\alpha}) \times x^{\alpha+1}) x > k, k = 10, p = 2000, \alpha = 1.1$
Incoming request interval	Pareto distribution	$P(x) = \alpha k^{\alpha} / x^{\alpha+1}, \alpha = 1.1$

Table 1 shows the distribution models used for generating the Web files and incoming request intervals. Since many studies have supported that the characteristics of Web server follow the long tail distribution[4], we use the Pareto distribution to model the Web file size and the incoming request intervals. However, because the Pareto distribution itself does not fit well to model small size of files, the Lognormal distribution model is also used [1]. The inter-arrival time of the requests is also modeled by Pareto distribution [2], where k in the inter-arrival time distribution represents the minimum arrival interval of the clients. In our simulation, k is varied to reflect the load of the Web cluster.

4 Performance Analysis

The latency and throughput results of the three models (*RR, SSL_with_session* and *SSL_with_bf*) in a 16-node data center are plotted in Figures 2 and 3. The results are measured as k value decreases from 45 to 25, where k is the parameter of the Pareto distribution for the incoming request interval, $P(x) = \alpha k^{\alpha} / x^{\alpha+1}$. When k decreases, the request interval becomes short, and consequently the load on the servers increases.



(a) Latency (b)Latency of two models

Figure 2. Average Latency of a 16-node Application Server

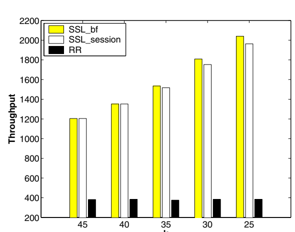


Figure 3. Average Throughput of a 16-node Application Server

Figure 2 (a) shows the average request latency with the *RR, SSL_with_session* and *SSL_with_bf* models. Since the latency of the *RR* distributor is too high compared to the two other models, we separate the *SSL_with_session* and *SSL_with_bf* results in Figure 2 (b). From Figure 2, we can observe that *RR* shows the worst performance since subsequent requests from a client are not likely to be forwarded to the same server, which caches the previous session information of the client. It means that CPU cycles are wasted to re-authenticate and negotiate keys between a client and a server.

In Figure 2 (b), the proposed *SSL_with_bf* scheme shows lower latency than the *SSL_with_session* by about 30%. The latency results of the *SSL_with_session* and *SSL_with_bf* are truncated at k = 25 due to servers' saturation. Figure 3 plots the throughput results of the three models. The *SSL_with_bf* model also yields better throughput compared to *SSL_with_session* as the load increases.

5 Conclusions

In this paper, we have investigated the performance implications of SSL protocol for providing secure service in a cluster-based application server, and have proposed a backend forwarding scheme for improving server performance through better load balance. The proposed *SSL_with_bf* scheme exploits the underlying user-level communication for minimizing the intra-cluster communication overhead.

All results in this paper indicate that the proposed backend forwarding scheme is a viable mechanism for improving the performance of application servers in cluster-based data centers.

References

[1] M. Andreolini, E. Casalicchio, M. Colajanni, and M. Mambelli. A Cluster-Based Web System Providing Differentiated and Guaranteed Services. *Cluster Computing*, 7(1):7–19, 2004.

[2] E. Casalicchio and M. Colajanni. A Client-Aware Dispatching Algorithm for Web Clusters Providing Multiple Services. In *Proceedings of 10th International World Wide Web Conference*, pages 535–544, May 2001.

[3] Compaq, Intel and Microsoft. Specification for the Virtual Interface Architecture. Available from `http://www.viarch.org`, 1997.

[4] M. Crovella and P. Barford. Self-Similarity in the World Wide Web Traffic-Evidence and Possible Cause. In *Proceedings of ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems*, pages 160–169, 1996.