L7LB: High Performance Layer-7 Load Balancing on Heterogeneous Programmable Platforms

Xiaoyi Shi, Yifan Li, Chengjun Jia, Xiaohe Hu, Jun Li
Tsinghua University, Beijing, China
{shixiaoy20, liyifan18, jcj18}@mails.tsinghua.edu.cn, {hxhe, junl}@tsinghua.edu.cn

***Abstract*—Layer-7 load balancing is an essential pillar in modern enterprise infrastructure. It is inefficient to scale software layer-7 load balancing which requires hundreds of servers to meet the large-scale service requirements of 1Tbps throughput and 1M concurrent requests. This paper presents L7LB with a novel fast path and slow path co-design architecture running on a heterogeneous programmable server-switch. L7LB is efficient by offloading most packets' forwarding onto the Tbps bandwidth switch chip, with few CPU cores processing application connections. The preliminary prototype demonstrates the layer-7 load balancing functionality and shows that L7LB can meet the large-scale service requirements.**

***Index Terms*—layer-7, load balancing, server-switch**

I. Introduction

Layer-7 load balancing distributes traffic by inspecting application data, *e.g.*, HTTP host or HTTP path. Therefore, modern service providers adopt layer-7 load balancing to enhance the infrastructural abilities of traffic management, application performance analysis, and network security.

It is challenging to realize scalable and efficient layer-7 load balancing. Nowadays, software layer-7 load balancing solutions on commodity servers are flexible to customize functions and open to develop operation tools. However, due to the 10Gbps NIC and the 10K concurrent request processing capability of a single commodity server, to scale software layer-7 load balancing, hundreds of servers are required to meet the large scale service requirements of 1Tbps throughput and 1M concurrent requests.

In this paper, we present, *L7LB*, a high performance, *i.e.*, scalable and efficient, layer-7 load balancing design by leveraging recent advances on programmable switch chip [1] and combining the advantages of the switch chip high bandwidth and the CPU fast computation. The heterogeneous programmable platform used by L7LB is the commodity *server-switch* [2] with a 3.2Tbps switch chip and an Intel Xeon processor. Recent research works [3], [4] design millions of concurrent requests layer-4 load balancing with the programmable switch chip through connection table compression. However, these designs can not be directly adopted to L7LB due to the complex application processing, *i.e.*, L7LB establishing and processing HTTP connections with both clients and back-end servers. The solution and novelty of L7LB design lie in two parts:

- packet processing decomposition, L7LB is decoupled into *fast path*, which is I/O intensive and runs on the switch chip for packet parsing, matching, rewriting and encapsulation, and *slow path*, which is computation intensive and running on the CPU for layer-7 connection establishing and application information parsing and matching,
- fast path and slow path co-design, the established connection metadata in the slow path is passed to and stored in the fast path connection table to enable that the slow path only processes the first several handshake packets and the fast path modifies and forwards the most data packets.

Therefore, benefiting from the fast path and slow path design, L7LB achieves line rate processing on the server-switch, realizing, *scalability*, the order of magnitude of Tbps throughput and Million concurrent requests and *efficiency*, one server-switch instead of hundreds of servers. Our preliminary L7LB prototype demonstrates the layer-7 load balancing functionality and shows that most packets are handled by the programmable switch chip.

This work is supported by the Science and Technology Project of State Grid Corporation of China under Grant (No. SGHAXT00WWJS2200033), National Natural Science Foundation (No. 61872212), and Industry-university-research Innovation Fund for Chinese Universities (No. 2021FNA04002).

II. Design and Implementation

The basic functionality of L7LB is mapping and balancing the pair (SIP:SPORT, VIP:VPORT) of the client source address and the service external virtual address to one of the back-end destination addresses DIP:DPORT according to layer-7 application information. Clients establish layer-7 connections with L7LB and L7LB establishes connections with back-end servers. Then, the requests and responses are forwarded by L7LB.

The system architecture is shown in Fig. 1. L7LB runs on the heterogeneous programmable server-switch. The first few packets of a connection are sent to the slow path to calculate the back-end server, and then the subsequent packets are forwarded by the fast path. The fast path is realized using P4 and runs on the switch chip. The modules of the fast path include packet parsing, packet matching, packet rewriting, packet encapsulation and compressed connection table. The slow path is realized by Go and runs on the CPU. The modules of the slow path include packet caching, connection selecting and establishing, table updating, and original connection table.

The layer-7 load balancing functionality is mainly realized in the connection selecting and establishing module of the

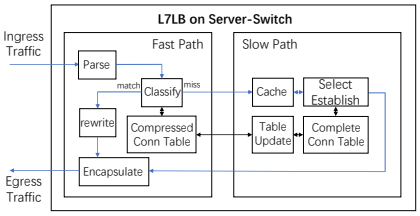

Fig. 1. L7LB system architecture

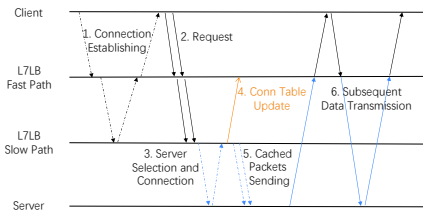

Fig. 2. L7LB packet processing sequence

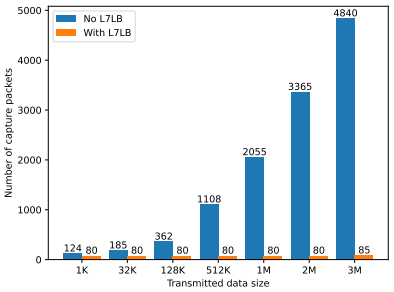

Fig. 3. The main namespace packets

slow path. In this module, the extracted application handshake information, *i.e.*, HTTP header metadata, as well as the access address information, *i.e.*, the pair (SIP:SPORT, VIP:VPORT), is classified with the configured string or regular expression rules to get the DIP:DPORT from the matched rule.

The key to achieving Tbps throughput and handling million concurrent requests is the compressed (to meet storage limitation) connection table in the fast path. With respect to each (SIP:SPORT, VIP:VPORT) access, the DIP:DPORT is calculated by the first few application handshake packets. Then, the key and value of (SIP:SPORT, VIP:VPORT) and DIP:DPORT is inserted to the connection table. Therefore, subsequent packets are directly matched and forwarded in the fast path on the switch chip, which has line rate processing capability for Tbps throughput and million connections.

Inserting only IP and Port pairs into the connection table and masquerading IP and Port pairs in L7LB fails the communication between the client and server, because the TCP state machine between the client and L7LB differs from that between L7LB and the server. The client and server can not directly recognize the packets without TCP state, *e.g.*, ACK number and SEQ number, masquerade. To this end, L7LB records the differences between the two TCP state machines and changes the ACK number and SEQ number on the fly.

Therefore, the sequence chart of L7LB packet processing is summarized and shown in Fig. 2. The client establishes a connection with L7LB slow path. When L7LB slow path receives requests from the client, the back-end server is classified and the connection between L7LB slow path and the server is established. Then, L7LB slow path updates the connection metadata to the connection table of L7LB fast path. The cached client messages are sent from L7LB slow path to the server. The subsequent requests and responses are handled among the client, L7LB fast path, and the server.

III. Experiment

The preliminary emulation experiment is run on a server with an Intel dual 52-core Xeon Gold 6230R Processor and 128GB memory. The programmable switch chip is emulated using a P4 software switch named bmv2. We start multiple ApacheBench instances as clients and multiple HTTP servers. The transmitted HTTP data is generated randomly. All the processes are isolated and run in different namespaces. L7LB distributes different client requests to different HTTP servers according to the pre-configured rules.

To demonstrate the scalability of L7LB, we conduct two kinds of emulation tests. One is the transparent transmission (No L7LB), and another is the L7LB in the middle transmission (With L7LB). We vary the transmitted data size and capture packets forwarded through the main namespace where the transparent transmission and L7LB slow path run. The number of captured packets is shown in Fig. 3. We can see that when running L7LB the number of packets passing through the main namespace is almost the same regardless of the sent packets size, which indicates that besides the first few handshake packets, the data packets between clients and servers are forwarded directly using L7LB fast path.

IV. Conclusion and Future Work

Layer-7 load balancing is becoming an vital pillar in modern enterprise infrastructure, distributing incoming accesses across the back-end servers based on application information. In this paper, we present a scalable and efficient layer-7 load balancing design on the heterogeneous programmable server-switch. Benefit from the fast path and slow path decomposition and the collaboration on connection table metadata updating and rewriting, most packets are handled by the line rate programmable switch chip.

Our future work includes exporting the preliminary L7LB prototype to the real server-switch platform and extending the L7LB functionality on application data modification and security protection.

References

[1] Pat Bosshart, Glen Gibb, Hun-Seok Kim, George Varghese, Nick McKeown, Martin Izzard, Fernando Mujica, and Mark Horowitz. "Forwarding metamorphosis: Fast programmable match-action processing in hardware for SDN." ACM SIGCOMM Computer Communication Review 43, no. 4, 99-110, 2013.

[2] Intel Server-Switch, shorturl.at/dxGQZ.

[3] Rui Miao, Hongyi Zeng, Changhoon Kim, Jeongkeun Lee, and Minlan Yu. "Silkroad: Making stateful layer-4 load balancing fast and cheap using switching asics." In Proceedings of the ACM SIGCOMM, pp. 15-28. 2017.

[4] Chaoliang Zeng, Layong Luo, Teng Zhang, Zilong Wang, Luyang Li, Wenchen Han, Nan Chen et al. "Tiara: A scalable and efficient hardware acceleration architecture for stateful layer-4 load balancing." In Proceedings of USENIX NSDI, pp. 1345-1358. 2022.