Smart Proxying for Microservices

Middleware 2019 Doctoral Symposium

Ratnadeep Bhattacharya
ratnadeepb@gwu.edu
Geroge Washington University
Washington, District of Columbia

Abstract

Proxies provide the networking infrastructure to the microservices architecture which has become ubiquitous in the cloud today. However, these proxies either cannot operate at line rate or are unsuitable for generic deployments. Furthermore, state-of-the-art autoscaling algorithms are still unable to account for quality of service the applications need to provide. In my dissertation I plan to explore a split proxy architecture that would be able to operate at line rate and provide autoscaling services that take into account QoS. Furthermore, such disaggregated L4/L7 processing provides a unique opportunity to embed DDoS mitigation like security features within the proxy framework.

CCS Concepts • **General and reference** → **Performance**; • **Networks** → **Cloud computing**; • **Security and privacy** → Denial-of-service attacks.

Keywords microservices, cooperative load balancing, split design, automatic scaling, DDoS mitigation

ACM Reference Format:
Ratnadeep Bhattacharya. 2019. Smart Proxying for Microservices: Middleware 2019 Doctoral Symposium. In *Middleware '19: 20th International Middleware Conference Doctoral Symposium (Middleware '19), December 9–13, 2019, Davis, CA, USA.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3366624.3368164

1 Introduction

With micro-services becoming more and more ubiquitous, proxy services have become key to providing networking in cloud computing environments. Microservices are meant to operate and scale as standalone units while still being an integral part of the overall application. While proxies and load balancers have long been used for traditional web services, microservices are radically more dynamic in nature and require a completely different solution. Load balancers, in case of microservices, need to provide extremely low latency and be able to reliably route traffic in an ever-changing environment.

In order to provide reliable connections between microservices that also respect strict latency and throughput requirements, these proxy services have to make improvements in areas such as speed of operations, efficiency of load balancing and autonomous scalability. I evaluated Envoy [3], a popular general purpose proxy, which is flexible and supports L7 load balancing, and found it could not operate at line rates of 10Gbps NICs. Recently open-sourced high performance stateless load balancers like Microsoft's Ananta [5], Google's Maglev [8], GitHub's GLB Director [6] and Facebook's Katran [7] operate at or close to the line rate but do not provide HTTP (L7) level services. As a result there is a need for a platform that provides the flexibility of Layer 7 routing with the efficiency of Layer 4 routing.

In my thesis I will explore challenges related to network and container management for microservices. The thesis will be composed of three projects that build on one another. In the first phase, I will investigate the systems challenges related to building efficient TCP and HTTP layer load balancers that provide both the performance and flexibility needed for microservices. Next I will research how a smart control plane can provide the intelligent load balancing and auto scaling capabilities needed to ensure that both the microservices and the networking infrastructure are able to efficiently meet their incoming workloads. Finally I will study the security challenges of running a microservices framework and show how the load balancing infrastructure can assist with authentication, DDoS prevention, and intrusion detection.

2 Problem

Currently, a popular micro-services deployment model is as pods that contain both the service instance and a proxy. Typically, these proxy services, such as Envoy, Docker Overlay Networking, WeaveNet, etc., have varied capabilities and generally provide both TCP and HTTP proxy services within the same worker threads. Unfortunately, their designs limit their scalability and can incur high resource consumption. Alternatively, higher performance TCP-only load balancers

ACM ISBN 978-1-4503-7039-4/19/12.
https://doi.org/10.1145/3366624.3368164

can be deployed such as Maglev or GLB, but they lack insight into L7 data and are not well integrated with the microservice management framework. Thus current approaches lack the combination of performance, flexibility, and automation that are desired for microservice management.

Essentially, a TCP proxy can load balance arbitrary requests since it is working at the session level. However, the major drawback of TCP load balancing is that application level information cannot be incorporated into the load balancing decision. Most application protocols today are multiplexed; an HTTP proxy can separate multiple multiplexed connections and open required number of sessions to backends - multiplexed or otherwise. A TCP proxy will not be able to make this distinction and the load balancing will not be as effective. Further, a TCP proxy can only perform routing based on IP, so fine grained rules such as redirection based on an HTTP URL cannot be used. On the other hand though, a TCP proxy can process a much higher number of incoming packets than a typical HTTP proxy.

To demonstrate the performance limits of existing HTTP proxies, I have evaluated the maximum throughput and resource consumption of the popular Envoy proxy. Our results show that the Envoy proxy is both unable to operate at line rates or scale linearly as a function of processor cores allocated.

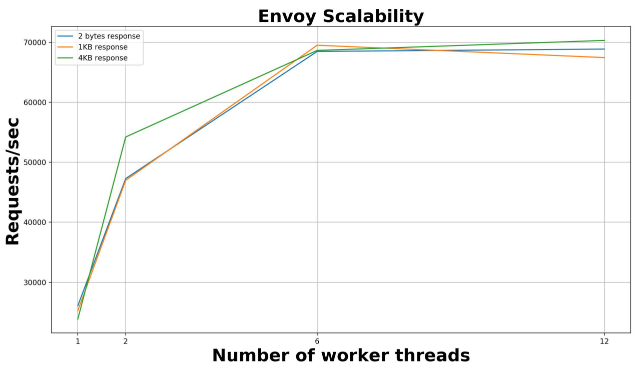

Figure 1. Fig: Envoy Response

3 Proposed Approach

TCP Layer: This layer would be deployed at the edge of the network, handling incoming connections directly. At this point, the idea is to simply load balance TCP connections and map packets to existing connections, but other functions can be evaluated in the future.

The benefit of the split TCP layer is that it can typically handle a much higher load than application layer components. This means that screening of DDoS and similar malicious connection attempts can be incorporated at this layer.

HTTP Layer: Unlike the TCP layer, the HTTP (or any generic L7 protocol) layer understands both multiplexed and kept-alive connections, enabling load balancing to happen based on the number of "true" connections.

Another benefit of this is that this protocol level understanding at the L7 layer can be used to influence load balancing decisions at the L4 layer through the control plane.

Control Plane: This component would generally be responsible for synchronizing connection maps at the TCP layer, for updating load information gathered from the HTTP layer to the TCP layer, for making intelligent scaling decisions and as a security manager.

In this project, I want to explore the potential of multi-level load balancing algorithms. In this case, I want to evaluate a feedback-based power-of-random-two [9,10] algorithm operating at the TCP layer. At a later stage, I might also look at machine learning based algorithms and do a comparison.

Autoscaling is one of the most critical requirements in the microservices environment today. The microservices themselves have been configured to be able to scale in or out as an individual unit. Under most circumstances, load balancers cope with this either by attaching to each service instance as a sidecar or by not providing services beyond the TCP layer. In our work, I wish to explore the potential of machine learning algorithms to provide automatic predictive scaling decisions that satisfy pre-defined SLAs (Service Level Agreements).

A split level design provides the opportunity to protect against attacks like SYN flood with traffic monitoring and scrubbing within the proxy framework with feedback from the HTTP layer, if required.

4 Conclusion

I expect to usher line rate network processing to microservices network management. I also anticipate to introduce a novel approach to load balancing and to be a herald of smart solutions for scalability and security into the proxy framework.

5 Acknowledgement

This work is being done in collaboration with Dr. Jinho Hwang at IBM Research and under the advisement of Dr. Timothy Wood at GWU. It is supported in part by NSF grant CNS-1763548.

6 References

[1] DPDK. https://www.dpdk.org/
[2] Introduction to Modern Network Load Balancing and Proxying. https://blog.envoyproxy.io/introduction-to-modern-network-load-balancing-and-proxying-a57f6ff80236
[3] Envoy Proxy. https://envoyproxy.io
[4] HAProxy. https://www.haproxy.com/

[5] P. Patel, D. Bansal, L. Yuan, A. Murthy, A. Greenberg, D. Maltz, R. Kern, H. Kumar, M. Zikos, H. Wu, C. Kim, N. Karri. Ananta: Cloud Scale Load Balancing. In SIGCOMM'13.

[6] GitHub GLB Director. https://github.com/github/glb-director

[7] Facebook Katran. https://engineering.fb.com/open-source/open-sourcing-katran-a-scalable-network-load-balancer/

[8] D. E. Eisenbud, C. Yi, C. Contavalli, C. Smith, R. Kononov, E. Mann-Hielscher, A. Cilingiroglu, B. Cheyney, W. Shang and J. D. Hosein. Maglev: A Fast and Reliable Software Network Load Balancer. 2016.

[9] M. Mitzenmacher A. W. Richa, R. Sitaraman. The Power of Two Random Choices: A Survey of Techniques and Results. Handbook of Randomized Computing.

[10] Test Driving "Power of Two Random Choices" Load Balancing. https://www.haproxy.com/blog/power-of-two-load-balancing/