Model-Based Load Balancing for High Performance Network Data Planes

Wei Zhang Abhigyan Sharma* Timothy Wood
The George Washington University **AT&T Labs Research*

Abstract

Network function virtualization (NFV) allows network functions to run as software on commodity servers. As they become more complex and grow in processing cost, replication is needed to ensure reliability and improve NF performance. However, balancing the load across multiple NFV servers can be challenging due to diverse service costs, server and flow heterogeneity, and dynamic workload conditions. In this poster, we propose NFVBalance, a resource-aware load balancer for network function service chains. NFVBalance models the CPU load on NFV servers in order to effectively guide its load balancing policies, while simultaneously achieving high performance with a DPDK-based design.

1 Introduction

Software-based data planes for NFV have seen a flurry of work recently. An important class of such data planes use a modular approach in which network functions (NFs) are implemented as modules that can be composed into service chains [1, 2]. Infrastructures that allow these modules to share CPU resources are particularly appealing since they support multi-tenancy or diverse service chains applied to different traffic classes.

Our work considers a cluster-wide deployment of such a modular data plane. Existing efforts on modular data planes focus on techniques to perform the workload assigned to an execution thread efficiently and securely. Our work focuses on the complementary problem of how to assign workloads to NFV servers and cores in the first place. These two efforts can deliver the promise of a high-performance and cost-effective software data plane.

To this end, our work seeks to address two questions. The first is a modeling question. Can we design a simple model that can predict the utilization of a CPU core as a function of the traffic assigned to it and a parameter that represents the per packet processing cost? Can this model work well even with multiple traffic classes each with different processing costs? Is this model robust to cross-core interference and different types of processing performed by NF modules?

Next, our work seeks to evaluate the usefulness of the model in load balancing traffic across CPU cores and across servers. In particular, how effective is the model in meeting diverse load balancing objectives, e.g., consolidate load on the least number of cores, minimize the maximum load on any core among a given number of cores, or ensure that higher priority traffic classes do not experience CPU load above a given threshold.

A model-based approach offers several advantages over a monitoring-based approach. Models for NFV resource utilization enable automation systems to predict behavior and proactively manage the system. In contrast, a more passive monitor-driven approach is likely to have out of date information potentially resulting in load oscillations. Since a model-based approach does not rely on continuous monitoring from processing cores, it also avoids measurement overhead at those cores.

Further, our work explores questions related to integrating a model-based approach with a stateful load balancer deployed in front of a pool of NFV servers. In particular, how the load balancer can efficiently track traffic statistics for each traffic class across multiple processing threads? How to translate the model developed above into a load balancing strategy that is amenable to an efficient implementation? Finally, do the performance gains of a model-based approach over existing strategies such as round-robin justify the additional engineering effort?

In the next section, we present our model and report on our experience in empirically validating the model using a DPDK-based prototype. We then describe how we envision this model being integrated with our high performance load balancer architecture.

2 Modeling Network Processing Costs

CPU load is an important metric to evaluate whether an NFV system is overloaded or is wasting resources. Many existing systems monitor the CPU usage and report them to a controller or load balancer. However, traffic in a high performance NFV environment can change rapidly. This can easily result in stale monitoring data. Rather than rely on feedback from data plane nodes, we are designing a model to predict CPU load, which is affected by the NF or chain processing cost per packet and the number of packets per second.

To train the model, we initially profile each network function to calculate the average processing cost per packet for each service chain. We then can estimate CPU usage as a product of the arrival rate of packets for each chain and its processing cost. Aggregating across all service chains on a specific core results in an estimate of the overall load.

Empirical Validation: We present a preliminary evaluation of our model's accuracy in predicting CPU loads. In the test, on one core, we run two service chains with

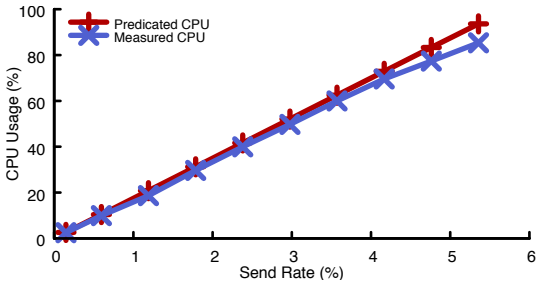

Figure 1: NFV CPU Model Accuracy

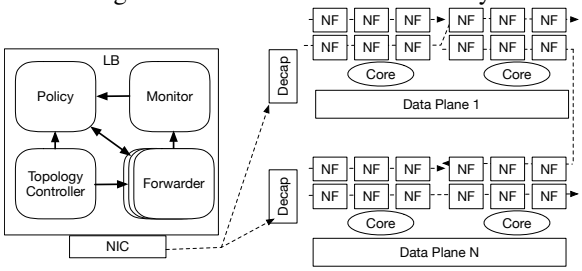

Figure 2: NFVBalance Architecture

processing cost 300 ns (high) and 100 ns (low) respectively. The arrival rate of high versus low cost service chain is about two times. We send 64 byte packets to the load balancer by using MoonGen. Figure 1 shows that even with our simple model, the aggregate predicted CPU usage closely tracks the measured CPU usage from the two service chains.

3 Load Balancing System Design

We are applying the above model-based approach in designing *NFVBalance*. The key components of NFVBalance are shown in figure 2. The critical path for packet processing goes through the Forwarder components, which can be replicated across multiple threads for scalability. The Forwarder must efficiently redirect incoming packets to the an NFV server running the appropriate service chain. To avoid contention between threads, each Forwarder maintains its own statistics about the flows it processes. This data is then periodically aggregated by the Monitoring component, which tracks statistics on a per-traffic class basis. The Topology Controller tracks which service chains are active on which servers and can start and stop additional replicas. Information from these components is fed to the Policy component, which guides balancing decisions made by the Forwarders.

Policy Component: Our architecture separates the data path (through the Forwarders) from the monitoring and control paths. This allows load balancing policies to be periodically updated and pushed to the Forwarders. Such dynamic load balancing is necessary to deal with server heterogeneity and dynamism in NFV workloads, e.g., short-lived vs. long-lived flows, and skewed inter-arrival times of flows. However, the load balancing policy should not be complex, especially in networks with a high arrival rate of flows. So we need a way to dynamically distribute flows in a very light manner, which can take into account flow and server heterogeneity.

	Server1 CPU (%)	**Server2 CPU (%)**
NFVBalance	30.40	34.42
RoundRobin	26.04	53.29

Table 1: CPU usage of NFVBalance vs Round Robin

Our load balancing policy uses our model to dynamically adjust weights for a weighted round-robin policy. For each core, we calculate an aggregate processing cost AC as a product of processing cost for each chain and the number of packets of each chain over the total number of packets on that core. The weight of a core is determined based on the objective of load balancing, e.g, to minimize the load on the most utilized core, we use $1/AC$ to get an updated weight for that core.

Preliminary Result: Our experiment compares NFVBalance to round-robin. Our testbed consists of two data plane nodes – server 1 and server 2. On each server, we use one core to run two service chains. The processing cost of each service chain is different on each server (chain 0, server 1: 100ns; chain 0, server 2: 200ns; chain 1, server 1: 100ns; chain 1, server 2: 300ns). Table 1 shows that round robin results in a CPU usage of server 2 that is twice that of server 1, while NFVBalance achieves a much more even load distribution.

Conclusions and Future Work. To our knowledge, this is the first work to explore a model-based load balancing approach for software-based data planes. Our initial results show that a model-based approach is accurate in predicting utilization for service chains with varying traffic demands and processing costs. In preliminary experiments, our NFVBalance prototype distributes load more evenly than static policies such as round-robin. There are several avenues of future work. The first one is a more robust validation of our model for more types of NFs and across multiple server cores. We will also look to evaluate load balancing performance for other objectives and compare against monitor-based approaches. Finally, we will seek to address systems challenges in scaling the load balancer itself to multiple servers.

References

[1] Bo Han, Vijay Gopalakrishnan, Lusheng Ji, and Seungjoon Lee. Network functions virtualization: Challenges and opportunities for innovations, 2015.

[2] Aurojit Panda, Sangjin Han, Keon Jang, Melvin Walls, Sylvia Ratnasamy, and Scott Shenker. Netbricks: Taking the v out of NFV. In *OSDI*. USENIX Association, 2016.