Integrating the Data Store into the Stateless Network Function Hosts

Anurag Dubey(student), Murad Kablan (student), Eric Keller
University of Colorado
(anurag.dubey, murad.kablan, eric.keller)@colorado.edu,

1 Introduction

StatelessNF System [2] proposes, a new architecture for network functions virtualization, where it decouples the existing design of network functions (*e.g.,* firewalls, NAT, load balancers) into a stateless processing component along with a data store layer. In breaking the tight coupling, StatelessNF allows network functions to achieve greater elasticity and failure resiliency.

As shown in Figure 1, StatelessNF system considers the data store as a remote node. Thus, an added latency is naturally introduced: reading from remote memory versus local will always be slower. To achieve acceptable performance, StatelessNF leverages advances in low-latency systems such as RAMCloud [4] where read and write operations are less than 100us. However, in applications where state needs to be updated for every network packets (*e.g.,* traffic counters, timers) such delays can greatly affect network applications.

In this poster, we propose exploring an integration of the data store nodes and the network function nodes, as illustrated in Figure 2 or more generally, a rethinking of the concept that the data store is separate from and independent of the nodes accessing the data store. This has two main advantages that must be considered, along with the potential for negative impacts such as reduced fault tolerance.

First, similar to a cache within each node, it can reduce latency and increase bandwidth to access data. Unlike a cache, the data store nodes' functionality is to replicate data for resilience, but not provide consistency across all accessing nodes. That is, with a caching architecture, each node accesses data and caches it locally. This, in turn, requires mechanism to maintain cache coherency. With an integrated data store, access to data goes to the nodes actually storing that data (which may be replicated among a few nodes, and coherency needs to be maintained between that small subset of nodes). This subtle difference makes this more scalable.

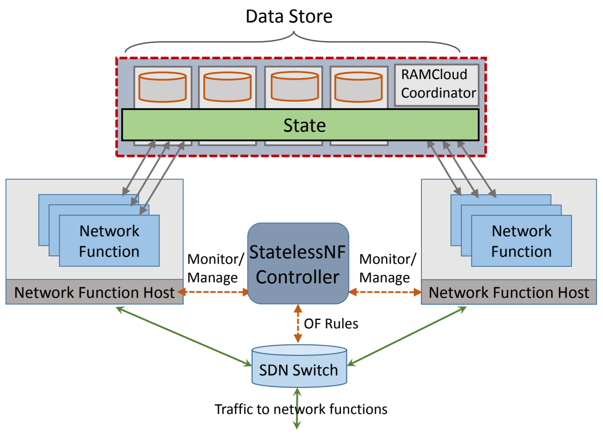

Figure 1: StatelessNF System Architecture with Centralized Data Store

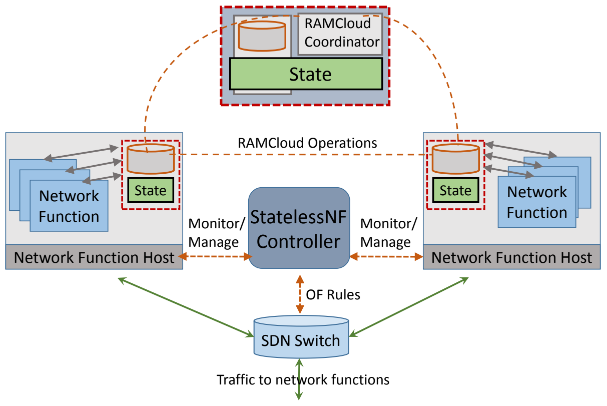

Figure 2: StatelessNF System Architecture with Distributed Data Store

Second, if we do not use replication for fault tolerance and have a 1-1 , this effectively reproduces the architectures which used migration of data from within the network functions to other instances [1, 3, 5]. It does so, however, with a general data store, moving the burden from every network function implementation to a common data store (which, of course, would require the data store to include the ability to control data placement).

However, such integration will not be that simple as multiple questions will arise to identify what and where to place data store nodes. Such questions are:

- Should there be a data store instance on every server that hosts a network function?
- Can we efficiently replicate data in a common infrastructure?
- What is the performance benefit of such an approach and what is the penalty in terms of failure resilience in such an approach?
- How do we place data to ensure optimize access across multiple instances?

Our work will be focusing on exploring efforts to answer such questions without compromising the consistency of the state.

2 Initial Prototype and Future Work

We have built an initial prototype where we emulated the entire StatelessNF system and data center infrastructure by packaging them as software into a single box (server). While this system may not show the overall performance advantage of the proposed integrated data store nodes, it does show its functionalities and performance for local read and write operations.

As future work, we intend to fully implement and explore designs for further improving performance and answering the above mentioned questions. This will be the actual complete deployment of the StatelessNF with distributed data nodes infrastructure. It involves deploying our software on multiple large servers that are connected with 10Gbit switches.

References

[1] A. Gember-Jacobson, R. Viswanathan, C. Prakash, R. Grandl, J. Khalid, S. Das, and A. Akella. OpenNF: Enabling Innovation in Network Function Control. In *Proc. SIGCOMM*, 2014.

[2] M. Kablan, A. Alsudais, F. Le, and E. Keller. Stateless Network Functions: Breaking the Tight Coupling of State and Processing. In *Proc. NSDI*, 2017.

[3] E. Keller, J. Rexford, and J. Van Der Merwe. Seamless BGP Migration with Router Grafting. In *Proc. NSDI*, 2010.

[4] D. Ongaro, S. M. Rumble, R. Stutsman, J. Ousterhout, and M. Rosenblum. Fast Crash Recovery in RAMCloud. In *Proc.*, 2011.

[5] S. Rajagopalan, D. Williams, H. Jamjoom, and A. Warfield. Split/Merge: System Support for Elastic Execution in Virtual Middleboxes. In *Proc. of USENIX NSDI*, Lombard, IL, Apr. 2013.