SmartSwitch: Blurring the Line Between Network Infrastructure & Cloud Applications

Wei Zhang* and Timothy Wood
The George Washington University
**Beihang University*

K.K. Ramakrishnan
Rutgers University

Jinho Hwang
IBM T. J. Watson Research Center

Abstract

A revolution is beginning in communication networks with the adoption of network function virtualization, which allows network services to be run on common off-the-shelf hardware—even in virtual machines—to increase flexibility and lower cost. An exciting prospect for cloud users is that these software-based network services can be merged with compute and storage resources to flexibly integrate all of the cloud's resources.

We are developing an *application aware* networking platform that can perform not only basic packet switching, but also typical functions left to compute platforms such as load balancing based on application-level state, localized data caching, and even arbitrary computation. Our prototype "memcached-aware smart switch" reduces request latency by half and increases throughput by eight fold compared to Twitter's TwemProxy. We also describe how a Hadoop-aware switch could automatically cache data blocks near worker nodes, or perform some computation directly on the data stream. This approach enables a new breed of application designs that blur the line between the cloud's network and its servers.

1 Introduction

Virtualization has revolutionized how data center servers are managed by allowing greater flexibility, easier deployment, and improved resource multiplexing. A similar change is beginning within communication networks through Network Function Virtualization (NFV) [1, 2]. This approach moves common network functionality such as switches, routers, and firewalls out of single-purpose "specialized" hardware devices and into common off-the-shelf (COTS) servers. NFV promises to both lower the cost of networking equipment and increase the flexibility of deployment and management.

For cloud platforms, the advent of NFV opens the possibility of customer controlled cloud networks. We all recognize that granting customers access to physical network devices can be unsafe and unscalable. But, if network functions can be run within virtual machines, suddenly offering "network functions as a service" becomes economical for cloud providers. However, we believe the most interesting impacts of NFV will come when cloud customers begin to intermix their applications and the functionality of the networking infrastructure [3].

Traditionally, network tasks such as routing and switching have been performed in hardware devices that are carefully tuned to redirect packets based on IP headers or 5-tuple flow information. In many cases, this information is not sufficient, since request routing may depend on the nature of the request being made, the load on the end host servers, etc. As a result, application-level load balancers and proxies are often used to redirect flows based on this type of higher level information (e.g., the general purpose HAProxy or the memcached specific TwemProxy). Unfortunately, these applications lack the high speed packet processing capabilities of network hardware, potentially causing the load balancer to become the bottleneck. NFV promises to change this by allowing software-based network services to be customized for application needs, yet still run at line speeds.

In this paper we explore how the bounds between applications and network infrastructure can be blurred by new technologies that allow user-space applications (running natively or in virtual machines), to perform complex packet processing and forwarding at rates of 10Gbps and beyond [4, 5]. Towards this goal we have developed a prototype "memcached-aware switch[1]," that performs both *application level redirection* to memcached servers and *local caching* to allow immediate responses for "hot" data. Unlike existing proxies such as Twitter's TwemProxy, our MemSwitch incurs minimal overhead on request latency, and supports a throughput at least eight

[1] Memcached is used as an example in this paper, but our system is general enough to handle different network functions in the backend.

times that of TwemProxy. We believe that this is the first step towards an integrated platform that supports flexible switching, data caching, and arbitrary computation.

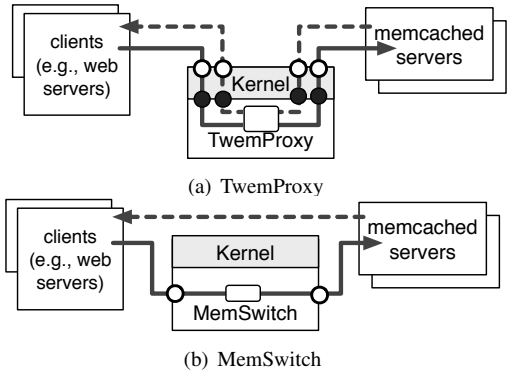
2 Application Integrated Networks

Having a data center network infrastructure that includes computation and storage capability will offer an opportunity for both cloud providers and users to reconsider how applications and network services are deployed and interact. To enable this, we are building the *SmartSwitch* platform. SmartSwitch is built on commodity servers with high-speed NICs, and will provide a framework to simplify the development of services that merge computation and storage into the network. SmartSwitch could be used to let a future router perform complex processing on each packet, or a remote desktop platform could be tightly integrated with the networking components to reduce latency. Our application driven networking framework can help users merge networking and processing components to build this type of complex functionality. There are three broad areas we will consider: 1) switching/load balancing, 2) storing data within the network, and 3) performing computation in the network.

2.1 High Performance Networking in VMs

Recent advances in network interface cards (NICs) and technologies like Intel's Data Plane Development Kit (DPDK) library allow end-host applications to receive data directly from the NIC, eliminating overheads inherent in traditional interrupt driven OS-level packet processing [6]. We have been building the NetVM platform [5] to extend these high-speed networking capabilities to applications running inside groups of related virtual machines. NetVM allows high throughput network applications to read and modify entire packets at near line speed, while taking advantage of the flexibility and customization of low cost commodity servers. Our preliminary implementation of NetVM is capable of reaching at least 10Gbps throughput using zero-copy message delivery for efficient inter-VM communication. With the newer systems and processors becoming available, we expect the system to be able to achieve 40Gbps on the state-of-the art systems.

Although NetVM's original goal was to exploit COTS hardware for network functions, our experience building it has shown us the potential benefits from incorporating compute and storage capabilities into network devices running as virtual machines on COTS hardware. This allows computational and storage functionality to be more flexibly positioned wherever it is needed in the network data stream.



(a) TwemProxy

(b) MemSwitch

Figure 1: (a) TwemProxy requires eight packet copies: 4 DMAs (light circles) and 4 kernel-user space copies (dark circles), to redirect a request and reply. (b) MemSwitch needs only two DMA copies since data is moved directly from the NIC to userspace and the server can transmit back to the client without requiring a redirection.

2.2 Application Directed Networking

The basic building blocks of a network are the routers and switches that direct packets towards their eventual destination. These devices typically make decisions based only on layer 2 (switches) or layer 3 (routers) information. The growing popularity of Software Defined Networks (SDNs) means that these devices may have their forwarding rules set by an intelligent controller that can apply higher level software policies to the network's control plane. However, the SDN controller cannot make decisions based on data plane information. As a result, it is still necessary to send packets through user applications such as load balancers or proxies if requests must be routed based on their content.

For example, a web application may use a cluster of memcached servers to cache the results of database queries. Since the number of memcached servers may change dynamically, it is impractical to have the web servers know the mapping of keys to cache servers. A common solution is to employ a proxy such as TwemProxy or moxi, user space applications that act as a front end to a memcached cluster and redirect requests after determining which key is being requested. As shown in Figure 1(a), this can result in as many as eight packet copies to service a request since packet data must be copied from kernel to user space for each send and receive at the proxy.

The layer of indirection provided by the proxy can thus increase the latency of each request, and the proxy itself can even become a bottleneck if it must service many different clients. Instead, our SmartSwitch uses a software-based switch that can be made aware of the memcached protocol to perform redirection, as shown in Figure 1(b). The "MemSwitch" is able to interpret the application data inside incoming requests and redirect them to the appropriate server. Further, since MemSwitch is simply

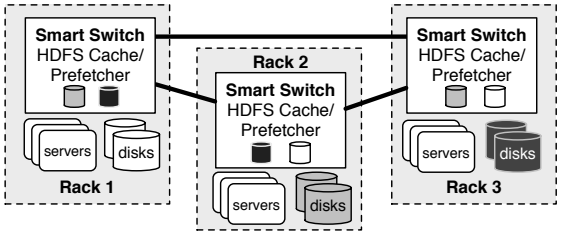


Figure 2: Hadoop-aware top of rack switches could cache or prefetch data blocks, making them available to local worker nodes running iterative or repeated jobs.

redirecting packets (as opposed to establishing a connection to the client), the response can be sent directly back to the client, reducing the load on the proxy.

In essence, SmartSwitch allows any application-level information (either within packet bodies, or gathered from other monitors) to control network redirection and load balancing in a far more flexible and dynamic way than traditional switches and routers.

2.3 Storage Within the Network

In a traditional Storage Area Network (SAN), the latency and throughput of hard disks were orders of magnitude slower than the network itself. However, it is now feasible to have 1TB or more RAM on a server, and PCI-based flash memory technologies are also providing large amounts of storage with very high speed access. As the capacity of DRAM and the throughput of Non-volatile memory (NVM) increases, it will become more economical to place large amounts of storage inside the network for caching, or to allow storage servers to be integrated with the network to increase their performance.

An example use case that we believe could benefit from dynamic in-network storage and caching is Hadoop. Hadoop attempts to place computational tasks close to where data is stored, but that is not always possible, particularly when a single data set is being used by multiple jobs. Since data stored in HDFS is typically read-only, it could be easily cached (or prefetched) within the network itself through a Hadoop-aware SmartSwitch, as shown in Figure 2. This could allow data requests to be transparently serviced from a cache closer to the point of computation, e.g., a top-of-rack SmartSwitch.

A data center that uses dynamic in-network caching will require the ability to map requests to the nearest cache entry. Tracking the locations of all the cache entries can be a challenge, given the number of data "chunks" and the number of potential cache locations in a large data center. The recent developments in content-centric networks, where the forwarding of requests and responses are based on *names* rather than the *location* [7] can be helpful. For example, a request would include the data name (each relevant data item would be expected to have a unique name). The routing/forwarding plane in the network switches would be populated with the information needed to enable forwarding the request towards all the cache locations. The forwarding rules specified for the switching subsystem would direct the request to the nearest cache location. The critical aspect here is that the name of the data item needs to be uniquely assigned, but the requester does not need to know the dynamic location of the current caches storing the named data item. We envisage that such name based forwarding capabilities will become increasingly common as the data center infrastructure scales and there is increasing use of dynamic placement of data, such as with in-network caches.

Many organizations now run large applications spread across public or private clouds. SmartSwitch provides a platform that could be used to move storage from network endpoints to software-based middle-boxes, allowing for pervasive caching and high performance data access.

2.4 Network based Computation

A software-based networking infrastructure means that any type of application can be pushed from running on end-hosts to running within the network path. We believe this could be particularly effective for the growing field of "live stream processing" applications. These systems move large volumes of data through processing pipelines for purposes such as video processing, website click-stream analysis, etc.

Especially when the data being produced must traverse multiple local or wide-area networks, it is desirable to intelligently filter the incoming data and only process or store the relevant information. Since it allows any type of computation to be flexibly deployed into the network, SmartSwitch's application integrated networking could provide high speed processing and filtering capabilities to these types of services. In contrast, performing such filtering in hardware incurs a high cost and results in yet another single-purpose device that cannot be flexibly updated in response to changing application needs.

Ultimately, our goal is to provide a platform where the distinction between end-host servers and network middle-boxes is no longer important. Processing pipelines (be it for data stream processing or even traditional multi-tier web applications), could be flexibly deployed and redirected across a cloud data center.

3 MemSwitch Prototype

We have developed a prototype MemSwitch capable of interpreting and redirecting memcached requests. Our prototype builds upon the DPDK libraries that allow userspace network IO and runs on an HP ProLiant DL160 G6 server with a dual Intel X5650 (2.66 Ghz) CPUs, 16GB of RAM, and an Intel 82599EB 10-Gigabit

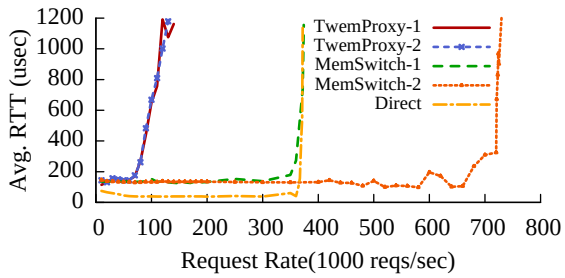


Figure 3: MemSwitch can support a much higher throughput than TwemProxy, and provides similar performance as connecting directly to the server, yet can offer advanced features like load balancing and caching.

NIC. In all experiments we restrict the proxy/switch to a single CPU core so that it will become saturated before our pair of backend memcached servers (with the same specs). While our current approach runs natively on a Linux host, porting it to our NetVM virtualization platform would be relatively simple and would incur minimal overhead [5].

Load Balancing: We configure the clients to send requests to a single memcached server IP which is shared by MemSwitch and all memcached servers. The packets initially reach the MemSwitch, which inspects the packet body to find the key being requested. As was shown in Figure 1(b), a flexible mapping can then be used to direct the request to one of the memcached servers by rewriting the packet's destination MAC address, but leaving the source IP of the original client. This is fairly easy process in in-path switches.

For *set* requests, the value being sent to the server may be spread across multiple packets, but only the first will contain the key. To handle this, the MemSwitch needs to keep a listing of on-going set requests, and ensure that all packets related to a single key are sent to the same memcached server. Our current implementation only supports *sets* for values less than the size of a packet.

Network-Caching To demonstrate the potential benefits of in-network storage, MemSwitch has the ability to cache hot data in memory, allowing it to directly respond to some client requests. If an incoming *get* request has a key that is found in a local hash table-based cache, then MemSwitch produces a reply packet and transmits the data directly back to the client. Our implementation is intentionally simple, and since our switch is currently single threaded, it avoids all locking. As a result, we consider our evaluation as a demonstration of the best potential performance for such a caching load balancer; a full implementation would certainly add overheads due to consistency concerns.

TCP vs UDP: MemSwitch's current implementation focuses on handling UDP *get* requests. TCP requests are more complicated to process because a connection handshake must be performed before the request is sent, yet MemSwitch will not know which server to direct the packets to until it can see the key being requested. We believe a more advanced switch could handle TCP requests by performing the initial TCP handshake itself, determining the requested key, and then either issuing a TCP RST to restart the connection or masquerading as the client to repeat the TCP handshake with the appropriate destination server. To evaluate its overhead, our current implementation *does* support TCP, but only when redirecting all requests to a single memcached server.

4 Evaluation

For baseline comparisons, we run (1) memcached directly connected to the client and (2) the TwemProxy developed by Twitter as a memcached load balancer [8], which is setup as shown in Figure 1(a). We use the *mcblaster* workload generator and compare throughput and response time. Unfortunately, TwemProxy only supports TCP, while MemSwitch only supports load-balancing under UDP; however, if there is only one backend server, we can use MemSwitch with either protocol.

Throughput: We first measure request round trip time as the *get* request rate is increased, and vary the number of backend servers from one to two. Figure 3 shows that connecting to memcached directly allows a request rate of about 350,000, while TwemProxy becomes overloaded when the rate is increased to only 90,000, regardless of whether there is one (the TwemProxy-1 line) or two (TwemProxy-2) memcached servers. TwemProxy has higher overhead because it needs to maintain the connections with both the client and servers, and because replies need to be returned through the proxy. In our testbed with two memcached servers, MemSwitch-2 becomes saturated at 720,000 requests per second, but since this is approximately double the rate of a single memcached server (MemSwitch-1 and Direct), we believe that the backend servers are becoming overloaded, and that MemSwitch could handle a much larger workload. Our current prototype with a single core can handle approximately 10 million requests (64-byte packets) per second. MemSwitch achieves this performance by using zero-copy between the application and the NIC, and because responses can go directly to the clients from memcached servers.

Request Latency: We next measure the latency with a light load, a request rate of 100 get req/sec, with 32byte value size. Table 1 shows that TwemProxy has about 2.39 times the average latency of direct memcached. MemSwitch UDP and TCP are about 19% and 3% slower than direct memcached, respectively. We believe that the overhead of both protocols should be the same, and are investigating why MemSwitch is showing greater overhead for UDP packets. MemSwitch UDP with caching

	Min/Avg/Max RTT usec	Norm. Avg
Direct TCP	48 / 132 / 551	1
Direct UDP	39 / 175 / 527	1
TwemProxy TCP	107 / 315 / 716	2.39
MemSwitch TCP	53 / 136 / 522	1.03
MemSwitch UDP	41 / 208 / 365	1.19
w/ Cache UDP	4 / 19 / 39	0.11

Table 1: Using MemSwitch adds minimal overhead compared to directly connecting to memcached servers, and can significantly reduce latency for cached data.

reduces the latency significantly, allowing a request to complete in an average of 19 microseconds, compared to 175 microseconds in direct memcached or 315 microseconds with TwemProxy.

Impact of Data Size: In order to evaluate how value size affects the overhead between TwemProxy and our MemSwitch, we change the value size from 32 bytes to 11Kb and measure the round trip time. From Figure 4, we can see that the larger the value size is, the greater the overhead TwemProxy brings. When the value size is 11K, it goes up to 2151us. This is because the cost of copying packets containing the full value must be paid by TwemProxy when it forwards responses to the clients. Increasing the value size has minimal impact on MemSwitch since responses return directly to the clients from the memcached servers.

5 Related Work

Several recent projects seek to increase the network throughput of COTS servers by using GPUs [9], zero-copy I/O [4–6, 10], or massive parallelism [11]. The focus of these systems has been on allowing network functionality (primarily routers and switches), to run on commodity servers with a lower cost than network hardware. In this work we explore how these platforms allow not only COTS-based network functions, but also a rethinking of where computation, storage, and network redirection occur within cloud data centers.

Our current implementation focuses on memcached, the popular key-value store used by many large scale web applications. Facebook is reported to use more than 10,000 memcached servers, requiring an extensive load balancing and key-management infrastructure in front of it [12]. We believe that software-based network infrastructure could provide similar management functionality at much higher performance. Fully integrating the memcached server into a zero-copy I/O platform like DPDK [6], NetVM [5], or netmap [4] could allow the caching servers to operate at a full line rate of 10Gbps, performance that currently has only been demonstrated on special purpose hardware [13].

The Strata system [14] has demonstrated how an SDN controlled Flash storage system can provide a fast, dynamically load balanced, NFS storage array. This system uses PCIe Flash devices that can each saturate a 10Gbps network port. The SDN controller is used to present a single IP for the array, but allow requests to be routed to a lightly loaded dispatch server. In a large scale network, it may be desirable to extend this architecture so that storage is not only available at network endpoints, but throughout the network's infrastructure as needed. SmartSwitch seeks to provide a platform for building these types of "Software Defined Storage" systems within the network.

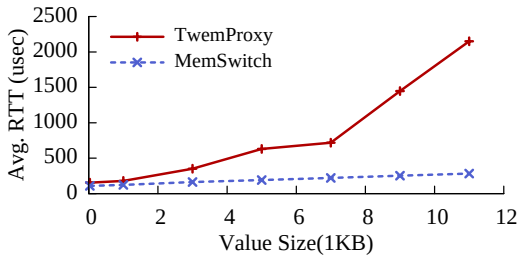


Figure 4: Forwarding requests through the TwemProxy adds significant overhead compared to our mem-cached aware switch. By caching data within the switch, the latency can be reduced even further.

6 Conclusions and Future Work

Advances in multi-core servers and high-speed network cards enable network services to be run inside commodity servers instead of dedicated hardware. Beyond simply allowing high performance software routers and switches, we believe this transition will present an opportunity to rethink where the boundary of network infrastructure and cloud applications should be. We have described how this could support application-aware load balancing, localized caching and network storage, or computational tasks that run directly on the network stream. To demonstrate the potential of these ideas, we have built MemSwitch, a memcached-aware load balancing switch. MemSwitch reduces request latency by 34% (or 94% with a switch-based cache) and can support a throughput at least eight times larger than TwemProxy.

In our ongoing work we are evaluating the performance of a memcached server that is fully integrated with a zero-copy network I/O stack. We are also exploring how other types of storage and computational functionality can be automatically placed throughout a network of software switches to provide performance and cost savings.

Acknowledgments

We thank the reviewers for their help improving this paper. This work was supported in part by NSF grant CNS-1253575, National Natural Science Foundation of China under Grant No. 61370059 and No. 61232009, and Beijing Natural Science Foundation under Grant No. 4122042.

References

[1] Frank Yue, "Network functions virtualization - everything old is new again," *http://www.f5.com/pdf/white-papers/service-provider-nfv-white-paper.pdf*, 2013.

[2] SDN and OpenFlow World Congress-Introductory White Paper, "Network functions virtualisation," *http://portal.etsi.org/NFV/NFV_White_Paper.pdf*, 2012.

[3] Wei Zhang, Jinho Hwang, Timothy Wood, K.K. Ramakrishnan, and Howie Huang, "Load balancing of heterogeneous workloads in memcached clusters," in *9th International Workshop on Feedback Computing (Feedback Computing 14)*, Philadelphia, PA, June 2014, USENIX Association.

[4] Luigi Rizzo, "netmap: A novel framework for fast packet I/O," in *Presented as part of the 2012 USENIX Annual Technical Conference*, Berkeley, CA, 2012, pp. 101–112, USENIX.

[5] Jinho Hwang, K.K. Ramakrishnan, and Timothy Wood, "NetVM: high performance and flexible networking using virtualization on commodity platforms," in *Symposium on Networked System Design and Implementation (NSDI)*, Apr. 2014.

[6] Intel Corporation, "Intel data plane development kit: Getting started guide," 2013.

[7] Van Jacobson, Diana K. Smetters, James D. Thornton, Michael F. Plass, Nicholas H. Briggs, and Rebecca L. Braynard, "Networking named content," in *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*, New York, NY, USA, 2009, CoNEXT '09, p. 112, ACM.

[8] "Twemproxy: A fast, light-weight proxy for memcached," Feb. 2012, https://blog.twitter.com/2012/twemproxy.

[9] Sangjin Han, Keon Jang, KyoungSoo Park, and Sue Moon, "PacketShader: a GPU-accelerated software router," in *Proceedings of the ACM SIGCOMM 2010 Conference*, New York, NY, USA, 2010, SIGCOMM '10, p. 195206, ACM.

[10] Guohan Lu, Chuanxiong Guo, Yulong Li, Zhiqiang Zhou, Tong Yuan, Haitao Wu, Yongqiang Xiong, Rui Gao, and Yongguang Zhang, "Serverswitch: A programmable and high performance platform for data center networks," in *Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation*, Berkeley, CA, USA, 2011, NSDI'11, pp. 2–2, USENIX Association.

[11] Mihai Dobrescu, Norbert Egi, Katerina Argyraki, Byung-Gon Chun, Kevin Fall, Gianluca Iannaccone, Allan Knies, Maziar Manesh, and Sylvia Ratnasamy, "RouteBricks: exploiting parallelism to scale software routers," in *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, New York, NY, USA, 2009, SOSP '09, p. 1528, ACM.

[12] Rajesh Nishtala, Hans Fugal, Steven Grimm, Marc Kwiatkowski, Herman Lee, Harry C. Li, Ryan McElroy, Mike Paleczny, Daniel Peek, and Paul Saab, "Scaling memcache at facebook," in *Proceedings of the 10th USENIX conference on Networked Systems Design and Implementation*, 2013, p. 385398.

[13] Michaela Blott, Kimon Karras, Ling Liu, Kees Vissers, Jeremia Br, and Zsolt Istvn, "Achieving 10Gbps line-rate key-value stores with FPGAs," in *Presented as part of the 5th USENIX Workshop on Hot Topics in Cloud Computing*, Berkeley, CA, 2013, USENIX.

[14] Brendan Cully, Jake Wires, Dutch Meyer, Kevin Jamieson, Keir Fraser, Tim Deegan, Daniel Stodden, Geoffre Lefebvre, Daniel Ferstay, and Andrew Warfield, "Strata: High-performance scalable storage on virtualized non-volatile memory," in *Proceedings of the 12th USENIX Conference on File and Storage Technologies*, Berkeley, CA, 2014, pp. 17–31, USENIX.