Optimizing a Virtualized Data Center

David Erickson, Brandon Heller,
Shuang Yang, Jonathan Chu,
Jonathan Ellithorpe, Nick McKeown,
Guru Parulkar, Mendel Rosenblum
Stanford University

Scott Whyte, Stephen Stuart
Google

ABSTRACT

Many data centers extensively use virtual machines (VMs), which provide the flexibility to move workload among physical servers. VMs can be placed to maximize application performance, power efficiency, or even fault tolerance. However, VMs are typically repositioned without considering network topology, congestion, or traffic routes.

In this demo, we show a system, Virtue, which enables the comparison of different algorithms for VM placement and network routing at the scale of an entire data center. Our goal is to understand how placement and routing affect overall application performance by varying the types and mix of workloads, network topologies, and compute resources; these parameters will be available for demo attendees to explore.

Categories and Subject Descriptors: C.2.3 [Computer-Communication Networks]: Network Operations

General Terms: Experimentation, Measurement, Performance

Keywords: Data center network, OpenFlow, Virtualization, Virtue

1. INTRODUCTION

Over the past decade, virtual machines have grown in popularity as application containers, since they improve server utilization and support fast provisioning for scale-out services. Increasingly, VMs are hosted in "clouds", large data centers that host thousands of customer VMs. Amazon EC2, the leading cloud provider, has approximately 40,000 servers [2], launches 80,000 VMs each day, and has launched 23 million VMs since its inception [7].

Equipment and energy costs provide a strong motivation for cloud owners to maximize operational efficiency. One way to improve operational efficiency is through workload placement algorithms, which map VMs onto physical machines (PMs). A placement algorithm might squeeze VMs onto as few servers as possible, then power down the unneeded servers or sell access to them on a spot market. Alternately, it might spread the VMs as evenly as possible, to maximize headroom for new VMs or avoid bandwidth bottlenecks. The placement algorithm could even maximize the performance of individual services, by co-locating their VMs or considering policy requests from customers.

Currently published workload placement algorithms enforce CPU, RAM, and NIC sharing policies [8, 9]; notably absent from this list is the network fabric interconnecting PMs in the infrastructure. Given the obvious operational impact of the network, it is natural to ask the question: *Can a virtualized data center be more efficient if the operator has visibility into network traffic, as well as control over packet routing?* Answering this question demands a number of prerequisites: representative workloads, a way to run those workloads at data center scale, an algorithm to place VMs and find routes, and a way to evaluate each algorithm's optimization metric. Perhaps most importantly, it requires a way to measure and control network traffic.

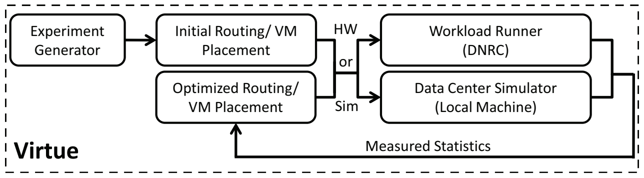
Experiment Generator
Initial Routing/ VM Placement
Optimized Routing/ VM Placement
HW
or
Sim
Workload Runner (DNRC)
Data Center Simulator (Local Machine)
Measured Statistics
Virtue

Figure 1: Experimenter's workflow using Virtue

The purpose of our demo is to show such a system, named *Virtue*. Virtue enables a researcher to run synthetic data center workloads on both a simulator and a real data center. To provide flexibility in VM placement, Virtue offers an interface to XenServer. To provide network traffic flexibility, Virtue leverages OpenFlow [5]. On this platform, we can investigate the scalability tradeoffs (state and computation) between coarse-grained traffic aggregation and fine-grained, low-level traffic routing. We can try approaches to VM placement that ignore the network, staged approaches that optimize for placement first then do routing, or even a joint optimization with a mixed integer linear program.

Our hope is that this investment in infrastructure and software will enable realistic comparisons of workloads and placement algorithms, resulting in both an understanding of the potential benefits of joint placement, as well as practical algorithms to achieve them.

2. SYSTEM

Virtue supports experiments targeted to either hardware or simulation, at the scale of an entire data center; its components are shown in Figure 1. The demo will show the entire Virtue system running live on a remote data center. Demo attendees can choose a workload and a placement algorithm, then explore the effects on VM placement, network communication, and service-level performance.

SIGCOMM'11, August 15–19, 2011, Toronto, Ontario, Canada.
ACM 978-1-4503-0797-0/11/08.

2.1 Generating an Experiment

There are many workloads, such as Facebook or Amazon, which we would love to test, but for which we will never have equivalent hardware or software. However researchers will likely have access to some hardware, and the ability to simulate other environments. These observations motivated the creation of a software abstraction layer for rapidly developing synthetic workloads that can run without modification on both real hardware and in a software simulator. The abstraction layer gives applications an API to send and receive network traffic, consume CPU cycles, access timers, and perform other utility methods. For example, an experimenter could create a 3-tier web workload by creating a simple application for each tier plus a client request application. The experimenter could then easily vary the number of VMs per tier, the distribution of request/response sizes between tiers, fan-out and fan-in of network messages, and network bisection bandwidth. More advanced experiments could introduce clusters of applications using varying numbers of VMs, similar to what might be seen in a cloud provider such as Amazon.

2.2 Routing and VM Placement

After defining the workload specifics of an experiment, the "algorithm" stage must define the physical locations on which to run virtual machines, as well as the physical path of flows between virtual machines. Here, a range of placement algorithms are possible, from naive (random placement) to intelligent (constraint satisfaction), for a range of optimization metrics. We have initially created mixed integer programming model algorithms that will optimize for performance (spread workload and network traffic as evenly as possible) or energy (collapse workload onto as few machines and network links as possible). Due to the complexity of the problem, we expect to also create and evaluate other approximation algorithms.

2.3 Hardware

To run a workload on hardware, Virtue requires physical servers running the Xen hypervisor and switches supporting the OpenFlow protocol. The demo will run on the Data center Network Research Cluster (DNRC), a 200-node cluster developed in a collaboration with Google for research use. The switches in the DNRC are wired together to provide three distinct topologies, two different fat-trees and a multi-rooted tree, while the nodes are virtualized by XenServer.

To run an experiment, the *workload runner* coordinates with the XenServer pool master to start the VMs, then passes the routing table to Beacon, the OpenFlow controller managing all aspects of the DNRC network. While the workload is running, Virtue records utilizations for the CPU, RAM, NIC, network links, and network traffic matrix, plus application-level measurements such as responses per second and completion time. After the workload completes, the experiment description and measurement data are passed back to the Optimized Routing/VM Placement stage, which can modify the virtual machine placement and network routes. The modified experiment can then be run to look for changes in any metrics of interest.

2.4 Simulation

To explore other network topologies, hardware configurations, and scales, we created a software data center simulator. The simulator is a simple discrete event-driven simulation based on the DESMO-J framework. It tracks all the resources available via the application abstraction layer, namely CPU cycles, network traffic at flow granularity, and scheduled timers. Although the simulator provides the same measurements and workflow as the hardware path, we don't expect both platforms to provide identical results. Still, the simulator allows us to explore and characterize unavailable hardware configurations within a reasonable error margin.

3. RELATED WORK

Several authors have analyzed data center workload placement optimization, focusing on shared resources such as the CPU, RAM, and NIC. Wood et al. explore hotspot detection and mitigation in a virtualized data center [9]. A system by Ruth et al. migrates VMs in clusters to achieve higher performance [8]. Commercial products are also available from VMware and Citrix; our work expands on all of these techniques by also monitoring and controlling the network.

Prior work on data center networks focuses on routing algorithms that spread traffic over all available paths, often via the introduction of scalable topologies [3, 6]. These papers generally assume traffic follows a path picked by a routing protocol or by oblivious load balancing. Hedera [1] uses OpenFlow (alongside ECMP) to detect and re-route long-lived, high data-rate flows, to achieve higher bi-section bandwidth than possible with ECMP alone. We make no assumptions about the network topology or hardware, other than OpenFlow support. Another example in this vein, ElasticTree, routes traffic through a minimum of switches and links to save energy by turning off unneeded switches [4].

Acknowledgments

David Erickson is funded by a Microsoft Research PhD fellowship.

References

[1] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat. Hedera: Dynamic flow scheduling for data center networks. In *NSDI*, San Jose, CA, April 2010. USENIX.

[2] R. Bias. Amazon's EC2 generating 220M+ annually. `http://cloudscaling.com/blog/cloud-computing/amazons-ec2-generating-220m-annually`, Oct 2009.

[3] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta. VL2: a scalable and flexible data center network. In *SIGCOMM*, pages 51–62, New York, NY, 2009. ACM.

[4] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, and N. McKeown. ElasticTree: Saving Energy in Data Center Networks. In *NSDI*, San Jose, CA, April 2010. USENIX.

[5] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. OpenFlow: enabling innovation in campus networks. *ACMCCR*, 38(2):69–74, April 2008.

[6] R. Niranjan Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, and A. Vahdat. PortLand: a scalable fault-tolerant layer 2 data center network fabric. 39(4):39–50, 2009.

[7] G. Rosen. Presentation at CloudConnect. `http://www.jackofallclouds.com/2010/04/presentation-at-cloudconnect/`, April 2010.

[8] P. Ruth, J. Rhee, D. Xu, R. Kennell, and S. Goasguen. Autonomic live adaptation of virtual computational environments in a multi-domain infrastructure. In *ICAC*, pages 5–14, Washington, DC, 2006. IEEE.

[9] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif. Black-box and Gray-box Strategies for Virtual Machine Migration. In *NSDI*, Cambridge, MA, April 2007. USENIX.