ModelNet: Towards a DataCenter Emulation Environment

Kashi Venkatesh Vishwanath*, Diwaker Gupta†, Amin Vahdat‡ and Ken Yocum‡

*Microsoft Research, † Aster Data, ‡ University of California, San Diego

***Abstract*—ModelNet is a network emulator designed for repeatable, large-scale experimentation with real networked systems. This talk introduces the ideas behind ModelNet that have made it a successful experimental platform. Beyond these core concepts, the talk highlights the latest additions to our methodology to test the next generation of network protocols and applications. Many of these developments address the datacenter compute environment: high-capacity networks, sophisticated infrastructure software (storage and virtualization), and complex network load. While these efforts significantly extend ModelNet's capabilities, there remain a number of open challenges, including incorporating new performance objectives (energy) and multicore architectures.**

I. Introduction: ModelNet v00.99

ModelNet [1] gives the experimenter control, repeatability, and accuracy. The ModelNet emulator needs a set of dedicated machines to mimic realistic deployment conditions. A subset of these machines (the core) emulate a user-specified network topology, reproducing the latency, bandwidth, and loss rates of individual network hops. The remainder of machines (the edge nodes) run end-user applications and transport protocols. ModelNet allows unmodified operating systems, networking stacks, and applications to communicate across the emulated network. Unlike shared wide-area testbeds, the network and edge nodes are dedicated to a single user. This makes experiments reproducible; they are isolated from external traffic, routing changes, or CPU contention on the end nodes. This architecture has proven to be a powerful design point for experimenting with networked systems and has been used to develop various transport and application level protocols including: DHTs [2] and overlays, content distribution protocols [3], smart switch architectures [4], and to evaluate bandwidth estimation tools [5].

II. The Need to Evolve

ModelNet was designed when network overlays, multicast protocols and peer-to-peer systems ran on hundreds of machines across the wide area. Today, datacenters are the dominant computing platform with current estimates of 7000 data centers in the US alone [6]. Networked systems, including data processing architectures like MapReduce, run at multiple datacenters, using thousands of machines and petabytes of data. These datacenters internally use high-speed networking technologies and are connected to each other via multiple long/fat links. While traditional DHT systems have found new life within the networking [7] and object-based storage infrastructure [8], other infrastructure software must manage the swarms of virtual machines (and storage) that host client applications, providing the proverbial "compute cloud."

This environment challenges various aspects of the original ModelNet design. In a datacenter setting, how can we explore the impact of future networking technologies without buying the corresponding new hardware? Or consider the task of buying a new storage system: How can we test the system at scale (hundreds to thousands of clients) without the hardware and administrative overhead? Finally, datacenter networks carry an incredibly diverse, time varying workload; how will this impact other services? By appropriately expanding the roll of ModelNet's edge nodes, we can make it easier to answer these kinds of questions.

III. Time Dilation

The ability to use unmodified end-user applications is a key strength of the ModelNet architecture, but it requires emulating the network in real time. This fundamentally limits the fastest emulated link to the fastest physical link in the ModelNet cluster. In contrast, simulators [9] often manipulate time, trading hours of real time to simulate fast (or huge) networks for a few simulated minutes. But they must often run alternate, simulated versions of the system under test.

We developed *time dilation* [10] to resolve this seeming impasse. Time dilation provides the illusion to an operating system and its applications that time is passing at a rate different from physical time. We modify a virtual machine monitor to make time appear to move slower for the guest OS. For example, time dilation can convince a system that for every 10 seconds of wall clock time, only one second of time passes in the operating system's dilated time frame. Time dilation does not, however, change the arrival rate of events from I/O devices such as the network interface. Hence, from the guest OS's perspective, physical resources appear 10 times faster: in particular, data arriving from a network interface at a physical rate of 1-Gbps appears to the OS to be arriving at 10-Gbps. We refer to the ratio between the rate at which time passes in the physical world to the operating system's perception of time as the *time dilation factor*, or TDF.

By employing TDF values greater than one, time dilation enables the network to appear to have more capacity than is physically possible. Using this, for instance, we can explore the impact of a potential upgrade to 10-GigE equipment while still experimenting on 1-GigE technology. We can also evaluate the impact of networks with gigabits of bisection bandwidth or the high-speed/high-latency links between datacenters. For instance, while TCP is known to operate sub-optimally across such links, time dilation makes it feasible to evaluate new transport protocols designed for these high bandwidth-delay environments [11].

IV. DieCast and Swing

Services within datacenters may use hundreds or thousands of machines. Emulating how these work at scale is critical but running an experiment with thousands of nodes is challenging both economically and technically. DieCast [12], leverages time dilation to emulate large-scale services on a small set of physical nodes trading time for physical resources. Time dilation makes all devices appear faster, including the disk and the CPU; DieCast exploits this to scale down the physical hardware requirements for the experiment. By modifying the virtual machine monitor's CPU scheduler, and integrating a disk model into the VMM, DieCast allows a user to emulate many physical machines on a single node. DieCast has been used to test and validate commercial scalable storage systems. Panasas builds PanFS, an object-based cluster filesystem. To meet customer requirements, they need to make sure the system can perform well under a given client's access pattern. Using DieCast, we were able to increase the number of simultaneous clients used to stress-test their system by an order of magnitude, multiplexing a 1000-machine experiment on 100 physical machines.

Unlike live testbeds, such as Planetlab, a ModelNet emulation will only see the traffic from the users applications. In order to understand the effects of cross traffic we developed Swing [13]. Swing is a closed-loop, network responsive traffic generator that accurately captures the packet interactions of a range of applications using a simple structural model. Starting from observed traffic at a single point in the network, Swing automatically extracts distributions for user, application, and network behavior. It then generates live packet traffic corresponding to the underlying models in the ModelNet environment. By modeling fine-grained user, network and application behavior Swing can reproduce burstiness in traffic across a range of timescales. Swing also provides users with a set of intuitive knobs that can be tuned to project traffic to alternate scenarios. For instance, it allows the user to change assumptions about network conditions, application mix and application characteristics, for instance the response size of objects, to generate new traffic.

V. Future work

With these facilities, ModelNet provides an emulation environment for accurately testing the core network, future networking scenarios and protocols, and sophisticated large-scale distributed services. However a number of challenges remain. ModelNet provides a simplistic view of the emulated network with limited packet processing and shortest path routing via static routes. This limits the ability to experiment with new routing architectures and protocols [14], [15]. Additionally, models may be required for exploring energy-aware networking, and for emulating various failure modes of both end systems and network components. Our hope is that this extra functionality can be realised by taking advantage of emerging multicore architectures to limit the impact on emulation scalability.

ModelNet is currently available for FreeBSD and Linux. Efforts are underway for a new distribution that includes a regression suite to ensure correct installation along with a preliminary assist for debugging misconfigurations. More information may be found online at: http://modelnet.sysnet.ucsd.edu.

References

[1] A. Vahdat, K. Yocum, K. Walsh, P. Mahadevan, D. Kostic, J. Chase, and D. Becker, "Scalability and Accuracy in a Large-Scale Network Emulator," in *Proceedings of the Fifth Symposium on Operating System Design and Implementation (OSDI)*, December 2002.

[2] S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz, "Handling Churn in a DHT," in *Proceedings of USENIX Technical Conference*, June 2004.

[3] D. Kostic, A. Rodriguez, J. Albrecht, and A. Vahdat, "Bullet: High Bandwidth Data Dissemination Using an Overlay Mesh," in *19th ACM Symposium on Operating Systems Principles*, October 2003.

[4] K. Yocum, J. Chase, and A. Vahdat, "Anypoint Communication Protocol," in *Position Summary at HotOS-VIII*, May 2001.

[5] K. V. Vishwanath and A. Vahdat, "Evaluating Distributed Systems: Does Background Traffic Matter?" in *Proceedings of the 2008 Usenix Annual Technical Conference*, 2008.

[6] L. Siegele, "The Evolution of Data Centres," *The Economist*, vol. 389, no. 8603, October 2008.

[7] C. Kim, M. Caesar, and J. Rexford, "Floodless in SEATTLE: A Scalable Ethernet Architecture for Large Enterprises," in *ACM Sigcomm*, 2008.

[8] G. DeCandia, D. HAstorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels, "Dynamo: Amazon's Highly Available Key-Value Store," in *Proceedings of the Twenty First ACM Symposium on Operating Systems Principles*, 2007.

[9] The Network Simulator ns-2, "http://www.isi.edu/nsnam/ns."

[10] D. Gupta, K. Yocum, M. McNett, A. C. Snoeren, G. M. Voelker, and A. Vahdat, "To Infinity and Beyond: Time-Warped Network Emulation," in *Proceedings of the 3rd ACM/USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, San Jose, CA, May 2006.

[11] D. Katabi, M. Handley, and C. Rohrs, "Internet Congestion Control for Future High Bandwidth-Delay Product Environments," in *Proceedings of the ACM Conference on Communications Architectures and Data Communication (SIGCOMM)*, August 2002.

[12] D. Gupta, K. V. Vishwanath, and A. Vahdat, "DieCast: Testing Distributed Systems with an Accurate Scale Model," in *Proceedings of the 5th ACM/USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2008.

[13] K. V. Vishwanath and A. Vahdat, "Realistic and Responsive Network Traffic Generation," in *ACM Sigcomm*, 2006.

[14] M. Al-Fares, A. Loukissas, and A. Vahdat, "A Scalable, Commodity Data Center Network Architecture," in *ACM Sigcomm*, 2008.

[15] A. Greenberg, 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 *ACM Sigcomm*, 2009.