3rd Workshop on I/O Virtualization (WIOV '11)

Portland, OR
June 14, 2011

I/O Virtualization Architectures

Summarized by Sejin Park (baksejin@postech.ac.kr)

SplitX: Split Guest/Hypervisor Execution on Multi-Core

Alex Landau, IBM Research Haifa; Muli Ben-Yehuda, Technion Israel Institute of Technology and IBM Research Haifa; Abel Gordon, IBM Research Haifa

Machine virtualization lies at the foundation of many datacenters and cloud computing, but its use is often limited due to unacceptable performance overhead. Muli Ben-Yehuda and his co-authors argue that this overhead is inherent in Popek and Goldberg's "trap and emulate" model for machine virtualization. In that model, when a guest operating system executes a privileged instruction, the instruction traps, causing the core to exit from the guest context and switch to the hypervisor context. The hypervisor then emulates the trapping instruction and switches back to the guest OS. The overhead of machine virtualization comes from these exits.

To achieve the holy grail of zero-overhead machine virtualization, the authors propose the SplitX architecture, where the guest and the hypervisor each runs on a dedicated set of cores. Exits are replaced by inter-core messages. When the guest executes a privileged instruction, the guest's core sends a message to the hypervisor's core, which handles the exit and sends a message back. Such an architecture replaces the costs of an exit completely, replacing them with the cost of fast inter-core communication, which is an order of magnitude smaller. It also enables removing some or all of an exit's synchronous cost, since the hypervisor can handle certain types of exits while the guest continues running. Muli presented an analysis of a networking workload which incurs a 35% slowdown with current methods. With SplitX, the same workload incurs a slowdown of less than 1%. SplitX requires some hardware support for running unmodified operating systems, but they are implementing SplitX functionality on current hardware.

Flash Memory Performance on a Highly Scalable IOV System

Peter Kirkpatrick, Adel Alsaadi, Purnachandar Mididuddi, Prakash Chauhan, Afshin Daghi, Daniel Kim, Sang Kim, K.R. Kishore, Paritosh Kulkarni, Michael Lyons, Kiran Malwankar, Hemanth Ravi, Swaminathan Saikumar, Mani Subramanian, Marimuthu Thangaraj, Arvind Vasudev, Vinay Venkataraghavan, Carl Yang, and Wilson Yung, Aprius, Inc.

No presentation was made for this paper.

VAMOS: Virtualization Aware Middleware

Abel Gordon, IBM Research Haifa; Muli Ben-Yehuda, Technion Israel Institute of Technology and IBM Research Haifa; Dennis Filimonov and Maor Dahan, Technion Israel Institute of Technology

Abel Gordon said that virtualization overhead is still a problem, due to the switching between the guest and the hypervisor. Because previous approaches to deal with this performance problem focused on the interaction between the hypervisor and the guest OS, there are still potential optimization issues in the application layer. These can be assigned to virtualization-aware middleware such as databases, Web servers, or application servers, thereby reducing virtualization overhead. The architecture he showed was simple to understand. In contrast to the traditional architecture, some I/O module—say, module C—in the middleware moved down to the hypervisor. Thus module C directly interacts with the middleware in the guest OS. By using the hypervisor-level middleware module, the virtualization performance is improved without any modification to the operating system, and because the author moved the module, the code can be reused. In the evaluation, they achieved about 5%–30% improvement. They have plans to apply this technique to other middleware and are also considering rethinking the middleware from scratch.

During Q&A, Abel Gordon explained that Oracle has a similar architecture. In Oracle, there is some kind of JVM and Java application that runs directly on the hypervisor without the OS. Actually what they did, interfacing between the OS and the middleware, created some kind of small OS. Someone asked whether the authors tried multiple guest scenarios. Abel said just the single desktop was considered.

Invited Talk

Data Center Challenges: Building Networks for Agility

David A. Maltz, Senior Researcher, Microsoft

Summarized by Henrique Rodrigues (hsr@dcc.ufmg.br)

David started his presentation giving a general view of the network usage in a datacenter. Using Bing as an example, he described how network-intensive applications such as data mining and search indexing algorithms, which are focused on improving user experience, can saturate the core of a datacenter network. In this scenario, if any device at the highest level of the network topology goes down, the result would be massive network congestion.

David argued that increasing network capacity does not solve the problem, because demand is constantly growing. To support his point of view, he used a network utilization graph that showed the demand growth on the same period that the network received some capacity upgrades. Having noticed this problem, his research group started to think about a different solution to solve the capacity issues.

David commented that, most of the time, datacenter resource utilization is between 10% and 30% of total capacity. To increase the return on investment (ROI), the datacenter needs to be agile, able to quickly expand and contract the pool of active resources dynamically, following user demands. However, today's datacenter networks are built using a tree-based topology and VLANs to isolate different layer-2 network domains, which restricts their ability to assign any service to any server. Each service is constrained to a single L2 domain. This network architecture is also the cause of poor performance isolation and high oversubscription ratios. Finally, traffic measurement results show not only that traffic patterns on datacenters are very different from Internet traffic patterns but that the datacenters' traffic matrices are highly volatile.

David presented VL2, whose main principles are randomized routing to deal with datacenter traffic matrix volatility; decoupling server names from locations to allow the assignment of any service to any server; use of existing technologies to make the solution deployable on today's devices; and use of end-hosts, which are programmable and have lots of useful resources. The two key actions performed by a VL2 network are the encapsulation of complete routing information on each packet, performed by each end host using the information stored on a centralized directory system, and the random traffic spreading over multiple paths using valiant load balancing and ECMP. David explained how the solution works on a given Clos topology and how VL2 is able to provide full-bisection bandwidth and high resilience in case of link failures. Evaluation results showed that VL2 achieves high throughput and good performance isolation between different services.

Is it possible to have more than one tenant on each physical server when the datacenter is using VL2? It is not possible, because the network performance isolation provided by the hypervisor is not as good as the performance isolation provided for CPU and memory. Is VL2 able to handle the incast problem? VL2 is not a solution and will try to keep the queuing at the edges of the network and as low queuing as possible inside the network. What is the overhead imposed on end hosts to encapsulate routing information on each packet? They didn't measure such overhead but it was very low, because only a few more instructions were added to the original code. Would you clarify how traffic flows between the datacenter network and the Internet? The logical view provided by a VL2 network is that all the devices of a datacenter (servers and external links) are connected to a single bus which provides full bisection bandwidth between any pair of devices.

Performance Management in IOV Systems

Summarized by Muli Ben-Yehuda (muli@cs.technion.ac.il)

Revisiting the Storage Stack in Virtualized NAS Environments

Dean Hildebrand, Anna Povzner, and Renu Tewari, IBM Almaden; Vasily Tarasov, Stony Brook University

Dean Hildebrand started this session with an illuminating presentation on the difficulties virtualized systems create for makers of NAS (Network Attached Storage) systems. Makers of NAS systems go to great lengths to optimize their storage controllers and the protocols used to access them (e.g., NFS) for the I/O profiles of common workloads. More and more of these workloads, however, are now running in virtual machines. The introduction of an additional layer between the workload running in the virtual machine and the storage controller dramatically changes the I/O profiles seen by the controller. Whereas a workload running on bare metal would generate a mixture of metadata (e.g., create) and data (e.g., read and write) I/O requests to the controller, for example, the same workload running in a virtual machine would generate only data (e.g., read and write) I/O requests, because all of its block operations appear to the controller as file operations: reads and writes to the virtual machine's image file.

The bulk of Dean's presentation included detailed information on the I/O profiles of the same workloads running on bare metal and in virtual machines and the differences between them in a mixture of operations, in I/O sizes, and in sequential vs. random characteristics. Both bare-metal and virtual machine experiments were conducted using NFSv3. His conclusions were that virtual machine image files being read and written over NFS make NFS do "unnatural things" and that NFS and server file systems, as well as NAS makers, will need to adapt to these new workloads.

Wenji Wu of Fermilab asked whether the slides will be available after the workshop. Slides are available at http://www.usenix.org/events/wiov11/tech/.

Nested QoS: Providing Flexible Performance in Shared IO Environment

Hui Wang and Peter Varman, Rice University

Hui Wang said this paper is unusual for the workshop, in that it is fairly theoretical. It presents a quality-of-service model for virtualized environments ("nested" environments—not to be confused with nested or recursive virtualization). The nested QoS model offers a spectrum of response time guarantees based on the burstiness of the workload. Since a disproportionate fraction of server capacity is used to handle a small tail of highly bursty requests, the hope is that by providing a range of different response times which depend on the burstiness of the workload, overall server capacity could be reduced.

The model works by dividing incoming requests into different traffic classes, also called traffic envelopes, with each request's response time guaranteed as long as traffic remains inside the corresponding envelope. The model was evaluated on traces of block level I/Os from different workloads and appears to work well, leading to a large potential reduction in server capacity without significant performance loss. The results were all based on simulation, which led Himanshu Raj of Microsoft to ask Hui whether she had any idea what the runtime cost of implementing nested QoS would be. Hui answered that the cost is mostly in classifying requests into the different envelopes and is expected to be "very small."

Gatekeeper: Supporting Bandwidth Guarantees for Multi-tenant Datacenter Networks

Henrique Rodrigues, Universidade Federal de Minas Gerais (UFMG); Jose Renato Santos and Yoshio Turner, Hewlett-Packard Laboratories (HP Labs); Paolo Soares, Universidade Federal de Minas Gerais (UFMG); Dorgival Guedes, Universidade Federal de Minas Gerais (UFMG) and International Computer Science Institute (ICSI)

Suppose you have a server that runs virtual machines belonging to multiple tenants, who do not necessarily trust or cooperate with each other. All of the tenants share the server's network bandwidth. How can you provide network performance isolation to the different tenants, so that one tenant will not be able to overload the network at everyone else's expense? Henrique Rodrigues explained why neither TCP or UDP solves this problem, and that using rate-limiting is not enough, since it limits the senders but not the receivers. He then presented Gatekeeper, which satisfies the four practical requirements for a traffic isolation mechanism: scalability, an intuitive service model so that tenants can specify their requirements and understand what they are receiving, robustness against untrusted tenants, and the ability to trade off flexibility vs. predictability and make use of idle bandwidth. Gatekeeper works by limiting the transmit and receive bandwidth of each virtual machine (VM) through distributed agents running at the hypervisor layer on each of the servers. The Gatekeeper prototype is implemented in Open vSwitch on Xen/Linux using the Linux traffic shaping mechanism (HTB) for rate limiting.

Himanshu Raj of Microsoft asked about the difference between Gatekeeper and Seawall. The primary difference is that Gatekeeper divides available bandwidth between different tenants (where each tenant has multiple VMs), whereas Seawall only allocates available bandwidth between different flows. Wenji Wu of Fermilab asked how one knows to set the limits to the minimum bandwidth required by each service. Henrique replied that it is the tenant's responsibility to specify the bandwidth requirements of the applications.

Panel/Wild Ideas Session

Panel: Challenges for Virtualized I/O in the Cloud

Participants: Muli Ben-Yehuda, Technion—Israel Institute of Technology and IBM Research—Haifa; Alan Cox, Rice University; Ada Gavrilovska, Georgia Institute of Technology; Satyam Vaghani, VMware; Parveen Patel, Microsoft

No report is available for this session.