Get What You Pay For: Providing Performance Isolation in Virtualized Execution Environments*

Hannes Payer† Harald Röck† Christoph M. Kirsch

firstname.lastname@cs.uni-salzburg.at
Department of Computer Sciences
University of Salzburg, Austria

1. BACKGROUND

Virtualization allows multiple systems encapsulated in so-called domains to share completely isolated from each other a single physical machine. Several companies are already taking advantage of virtualization technology in order to sell a certain amount of CPU speed and I/O capacity in terms of latency and throughput on demand to their customers. Providers of such a service face the challenge of isolating the performance of domains from each other. Independent of the load generated by the domains running on the system each domain has to get what its customer is paying for, not more and not less.

Our study is based on the open-source virtual machine monitor Xen [1]. In the system architecture of Xen, the hypervisor is the lowest layer. On top of the hypervisor run several domains, which encapsulate complete operating system instances. The main tasks of the hypervisor are domain scheduling, I/O handling, and managing memory. Some of the domains running on the hypervisor are special domains which belong to the trusted code base, e.g. a driver domain where device drivers run. The driver domain is similar to a network router; domains performing I/O send their requests from their so-called frontend driver through an event channel to the backend driver, located in the driver domain. The backend driver multiplexes all requests onto the actual I/O device.

Several studies in the past dealt with the performance isolation problem in Xen. In [4] the authors evaluated the effects of different schedulers in Xen on the I/O performance of domains running several applications with different I/O requirements. I/O operations are typically not considered by the scheduler. However, indirectly the CPU scheduler influences the I/O performance of the applications running within a domain. The authors presented several system modification which, in many cases, provide better I/O isolation than the baseline system. Xen patched with these modifications is our baseline system.

*Supported by the EU ArtistDesign Network of Excellence on Embedded Systems Design and the Austrian Science Fund No. P18913-N15.
†Authors are students.

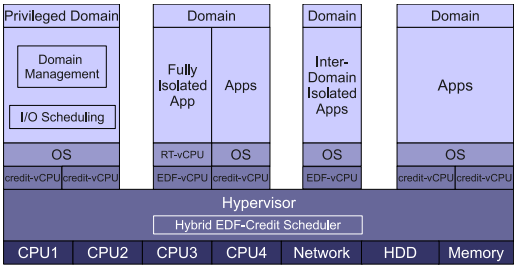

Figure 1: Virtualization architecture

A domain that performs I/O generates CPU load in the driver domain. This can have an effect on I/O performance of other domains, since all domains rely on the driver domain to perform I/O. In [3] the authors proposed a mechanism which accounts for load generated in the driver domain by other domains. They showed that for several workloads it is possible to isolate domains from each other in terms of network throughput, but they did not consider latency.

In the following we discuss our proposal which effects different components on different layers in the Xen system architecture as depicted in Figure 1. Our goal is to provide for each domain the performance it requires: CPU speed, I/O throughput, and I/O latency. Moreover, we propose to use CPU isolation techniques that offer strict temporal isolation for applications running within a domain resulting in reduced observable system jitter. The outcome of our study should also be applicable to other virtual machine monitors that are based on a similar system architecture like Xen.

2. PROPOSED SOLUTION

Providing performance isolation in virtualized execution environments requires extensions across the entire system, including the hypervisor (CPU scheduling), guest kernels (CPU isolation) and driver domains (I/O scheduling).

2.1 CPU Scheduling

In the Xen hypervisor the unit of scheduling is called a virtual CPU (vCPU), which is an abstraction of a physical CPU. The current scheduler in Xen, called credit scheduler, is tuned for high throughput and good fairness among all ac-

tive vCPUs in the system. Unfortunately, it does not provide low latency or guaranteed CPU shares.

Therefore, we propose a hybrid EDF-credit scheduler that can schedule a vCPU using either the standard credit scheduler (credit-vCPU) or alternatively a CPU-local EDF-based scheduler (EDF-vCPU). The hybrid EDF-credit scheduler provides low latency and allows to specify a guaranteed CPU share for EDF-vCPUs while still achieving high throughput for credit-vCPUs [2]. We have implemented the hybrid EDF-credit scheduler in the Xen hypervisor, using a runqueue for each physical CPU. It applies work stealing and dynamic migration of vCPUs in order to balance the workload across all available physical CPU cores. Furthermore, we extended the Xen tools to modify the scheduling parameters of a vCPU and to switch the vCPU from a credit-vCPU to an EDF-vCPU and back. Additionally, we implemented a true global EDF scheduler with a single run queue to evaluate the trade-off between global EDF scheduling accuracy and scheduling overhead.

For a virtualized OS running in a domain, a vCPU appears as a regular physical CPU core on which applications are scheduled by the OS scheduler. Therefore, adapting the hypervisor scheduler is not sufficient to provide the requested performance for an end-user application since the OS kernel introduces additional scheduling jitter. Hence, we plan to apply CPU isolation extensions (CPUISOL) to the Linux kernel in order to reduce the jitter and scheduling latencies introduced by the Linux kernel scheduler. CPUISOL isolates a CPU from the Linux scheduler, interrupts, and workqueues, while regular user-applications can still run on the isolated CPU. If the isolated CPU is an EDF-vCPU, we call it an RT-vCPU since the application running on it is temporally isolated from all other applications running inside the domain (intra-domain isolation) as well as from all other domains (inter-domain isolation). Such full temporal isolation weakens to just inter-domain temporal isolation for domains that only use EDF-vCPUs but not the CPU isolation feature.

2.2 I/O Scheduling

Domains not only require temporally deterministic execution of their program code but also temporally deterministic handling of their I/O requests. In order to support given latency and throughput requirements of an I/O request, the driver domain applies I/O scheduling among all domains currently connected to a device.

In the driver domain we experimented with a hierarchical token bucket (HTB) traffic shaping algorithm, which regulates the packet flow from guest domains to the network device and evaluated its performance. Depending on the HTB parameters and the network backend driver configuration (delayed copy and always copy mode) one can trade-off network traffic throughput and driver domain CPU utilization versus network latency jitter. Figure 2 and Figure 3 depict the results of our first experiments. We see the performance of the always copy versus the delayed copy mode

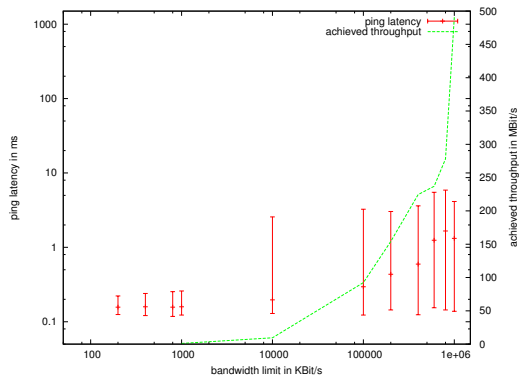

Figure 2: Latency and throughput in always copy mode

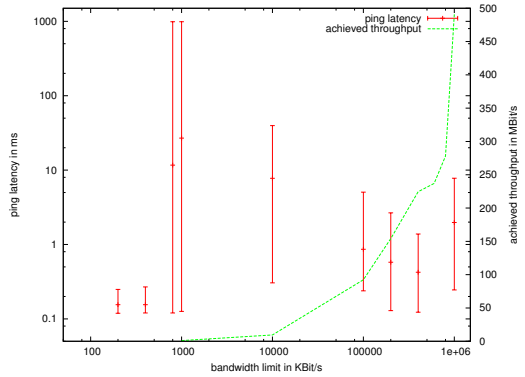

Figure 3: Latency and throughput in delayed copy mode

configuration under different HTB configurations. The latter one allows higher throughput but has high latency outliers due to its delaying policy. We aim for a solution that provides both high throughput and low latency.

Moreover, we experienced performance degradations on multi-processor platforms when interrupts of the network driver and of the network event channels (frontend and backend driver) were scheduled on the same core. An IRQ balancer which is aware of the domains bandwidth specifications could distribute the interrupts intelligently over the available CPUs which would optimize overall system performance and guarantee performance.

3. REFERENCES

[1] BARHAM, P., DRAGOVIC, B., FRASER, K., HAND, S., HARRIS, T., HO, A., NEUGEBAUER, R., PRATT, I., AND WARFIELD, A. Xen and the art of virtualization. In *Proc. SOSP* (2003).

[2] CRACIUNAS, S., KIRSCH, C., PAYER, H., RÖCK, H., AND SOKOLOVA, A. Programmable temporal isolation through variable-bandwidth servers. In *Proc. SIES* (2009).

[3] GUPTA, D., CHERKASOVA, L., GARDNER, R., AND VAHDAT, A. Enforcing performance isolation across virtual machines in Xen. In *Proc. Middleware* (2006).

[4] ONGARO, D., COX, A., AND RIXNER, S. Scheduling I/O in virtual machine monitors. In *Proc. VEE* (2008).