Questions Assignment 2

Benchmark questions

1. Look at your benchmark results. Are they consistent with your expectations, regarding the different virtualization platforms? Explain your answer. What are the main reasons for the differences between the platforms? Answer these questions for all benchmarks:

We expected the native platform to perform the best for all benchmarks, which turned out to be true, as they is are no redirections between the experiment host´s hardware (or the Parallels Pro “Hardware”) and the virtual machines, except for disk read benchmarks, which performed much better on KVM, which could be due to the experiment host not natively running on the hardware and instead being a VM of parallels itself.

1. CPU

We expected the QEMU VM without KVM to perform the worst, since it does not use Hardware acceleration methods like QEMU with KVM or Linux Containerization (if set up like that) with Docker do.

As most virtualization difficulties are caused by conventional IA-32 design, in which sensitive instructions do not always trap in rings bigger than zero, extended architecture for CPUs (1. generation extension), where the VMM in root mode and Guest OS in guest mode coexist in ring 0 and VMM handles permissions for which instructions guest is allowed to handle, has significantly better CPU performance, as there is less VMM intervention necessary.

This turned out to be true, with the exeption, that KVM has a bit lower value than docker, which could be due to the cost of hiding hardware details to the workload that can take advantage of it.

1. Memory

We expected the same to be valid for the memory benchmark. Since hardware acceleration with KVM (2. Generation extended hardware) introduced nested page tables and tagged TLBs (which improves performance for context switches), there is less VMM intervention required in QEMU with KVM and Docker with hardware acceleration. Also, there is no shadow page table overhead due to the nested page tables and a better scalability on multi-core CPUs.

This turned out to be not true, as the KVM performs clearly worse on the memory benchmark than the others.

QEMU performs the worst, as expected, because of the many traps due to the shadow page table mechanism.

1. Random disk access

We expected Docker with OS-assisted virtualization has no overhead in comparison to the native host, but KVM delivers significantly less IOPs, since every IO operation must go through QEMU, where I/O instructions usually trap (full virtualization).

This turned out to be wrong; QEMU and especially QEMU with KVM performed much better on both of the Disk read benchmarks than the native host and Docker, which could be due to the experiment host not natively running on the hardware and instead being a VM of parallels itself, which does not use kernel features.

1. Sequential disk access

We expected that there is no difference between the different hosts, since there is potential even for the native host to have much more IOPs.

Also, with Direct Assignment through Hardware assisted virtualization, guest VMs run the unmodified device drivers and there can be efficient I/O without VMM intervention.

This turned out to be wrong as described in the Random disk access answer.

1. Fork

We expected the highest value for the native machine, with a slightly lower value for KVM and Docker and the lowest value for QEMU. This turned out to be true, with the exeption, that KVM has a bit lower value than docker, which could be due to the cost of hiding hardware details to the workload that can take advantage of it.

f. Iperf uplink

We expected a worse performance for QEMU with and without KVM, since it uses a virtual network device, which turned out to be true.