BEST PRACTICES FOR VIRTUOZZO CONTAINERS FOR LINUX

Using Virtual Swap to Maximize Container Performance

Q1 2013

Configuration	#1 PVC 4.6 @ Kernel 2.6.18 (UBC)	#2 PVC 4.7 @ Kernel 2.6.32 (VSwap)
Container		
CPU	VCPU @ 2.67 GHz	vCPU @ 2.67 GHz
RAM	256 MB	256 MB
OS	Linux 2.6.18 028stab077.1 #1 SMP Mon Nov 1 19:26:08 MSK 2010 Red Hat 5.6 Tikanga x86_64	Linux 2.6.32 042stab036.6 #1 SMP Tue Sep 13 19:37:36 MSD 2011 Red Hat 6.1 Santiago x86_64
Host		
CPU	24 SMP @ 2.67 GHz	24 SMP @ 2.67 GHz
RAM	48 GB	48 GB
OS	Linux 2.6.18 028stab077.1 #1 SMP Mon Nov 1 19:26:08 MSK 2010 Red Hat 5.4 FFinal x86_64	Linux 2.6.32 042stab036.6 #1 SMP Tue Sep 13 19:37:36 MSD 2011 Red Hat 5.0.0 1066 x86_64

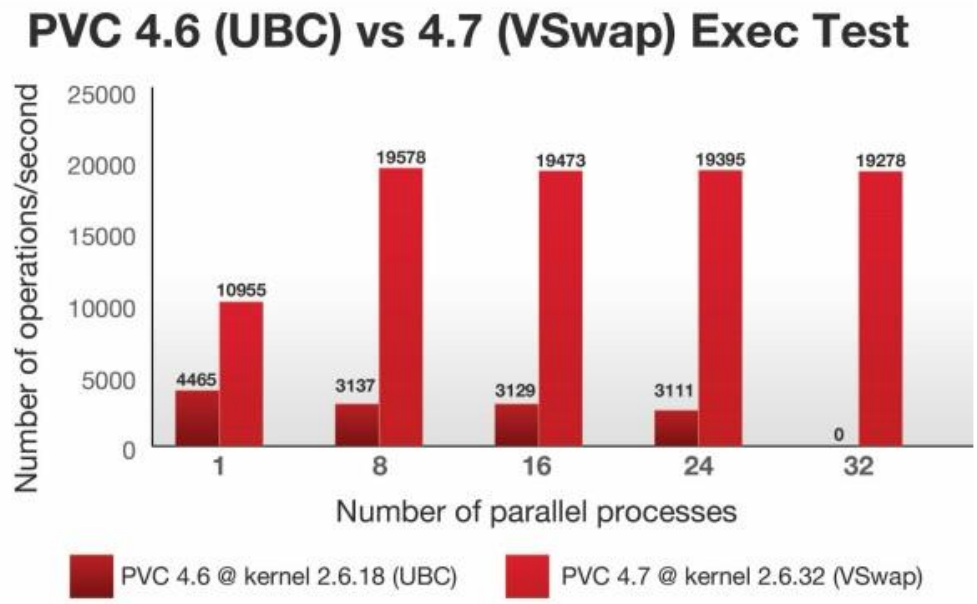

Figure 1. Differences between UBC and VSwap in times required to run the exec test.

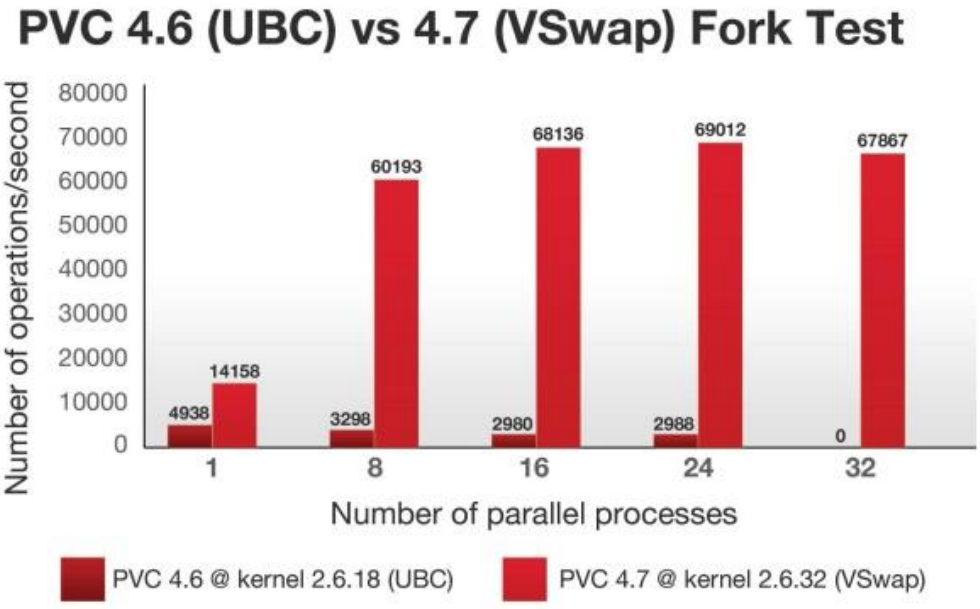

Figure 2. Differences between UBC and VSwap in times required to run the fork test.

Best Practices

VSwap makes the memory management of containers much easier. Instead of having to set a large number of primary and secondary UBC parameters, you just use the **physpages** and **swappages** parameters to set limits for the amount of memory and swap space available to each container.

Best Practice: Do not overcommit memory. That is, the sum of the **physpages** values for all containers should not exceed the memory available on the physical server, and the sum of the **swappages** values should not be greater than 25% of the hostnode's physical memory. As an illustration of this point, consider the following example:

Physical server = 32 GB RAM
Number of containers = 64
Physpages per container =~ 512 MB RAM
Swappages per container = ~128 MB (25% of 512 MB)

In this case, the physical memory of all containers is exactly the same as hostnode's memory, which will prevent an OOM situation from terminating an application or process inside the container. As for the **swappages** value, it does create a 25% chance that an OOM situation will terminate one or more processes in at least one container, and a .39% chance (25% divided by 64) that an OOM situation will terminate one or more processes in a specific container. However, since it's unlikely that all containers will simultaneously request the maximum memory at the same time, you may decide that this small risk is worth taking, since it will improve density on the server. If you do decide to overcommit, it's important to monitor containers' usage of memory closely and adjust it if you see that see certain containers are driving overall memory usage too high. It's also important to not overcommit too much of the hostnode's memory, as the greater the overcommitment, the greater the probability that the kernel will terminate applications.

Note that VSwap will automatically virtualize the **/proc/meminfo** output inside each container, so that the maximum allowable memory for that container is visible to the applications within the container. That way, applications will know how much memory is available to them and will not try to use more than the allotted amount. In contrast, before VSwap was available, the **/proc/meminfo** value was manually assigned and could be completely different from the actual available memory. In such a case, when the customer would try to use this additional memory, the application would fail.

As for **vm_overcommit**, the best practice is simply to use the default value of 1 so you don't have to worry about your application being terminated if it requests more memory than is available for the container.

Table 3 provides an overview of the differences between using UBC and VSwap with regard to memory available to the container.

Table 3: Memory Available to Containers with UBC vs. Swap

Memory Management	Hostnode	Container (what is seen)
UBC without meminfo virtualization	16 GB RAM 8 GB Swap	16 GB RAM 8 GB Swap
UBC with meminfo virtualization	16 GB RAM 8 GB Swap	configurable up to 16 GB RAM* configurable up to 8 GB Swap*
vSwap without overcommit (recommended configuration)	16 GB RAM 8 GB Swap	configurable up to 16 GB RAM configurable up to 8 GB Swap
vSwap with overcommit	16 GB RAM 8 GB Swap	configurable up to 16 GB RAM** configurable up to 8 GB Swap**

* But only the amount specified in PRIVVMPAGES would be actually usable.
** But only the amount specified in PHYSPAGES and SWAPPAGES would be actually usable.

One general suggestion: it's not a good idea to make container owners aware of resources that they can't use (e.g., there's no point in letting them know that the server has 128 GB of RAM when their container is only allowed to use only 16 GB).

Conclusion

VSwap, which is available with Paralllels Cloud Server 6.0, offers significant ease of use, performance, and stability improvements over legacy memory management schemes.

Instead of having to set numerous parameters, you only have to set two – physpages and swappages. In addition, VSwap lets you run memory-intensive applications inside containers without having to first tweak their configuration, and you can run parallel processes in far less time than you could with UBC without running out of memory. For all these reasons, VSwap is a much better approach than UBC for managing memory inside containers.