

Re: [PATCH] Add I/O hypercalls for i386 paravirt

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2007-08/msg09455.html>

- *From:* Avi Kivity <avi@xxxxxxxxxxxxx>
 - *Date:* Wed, 22 Aug 2007 11:37:49 +0300
-

Zachary Amsden wrote:

Avi Kivity wrote:

Zachary Amsden wrote:

In general, I/O in a virtual guest is subject to performance problems. The I/O can not be completed physically, but must be virtualized. This means trapping and decoding port I/O instructions from the guest OS. Not only is the trap for a #GP heavyweight, both in the processor and the hypervisor (which usually has a complex #GP path), but this forces the hypervisor to decode the individual instruction which has faulted. Worse, even with hardware assist such as VT, the exit reason alone is not sufficient to determine the true nature of the faulting instruction, requiring a complex and costly instruction decode and simulation.

This patch provides hypercalls for the i386 port I/O instructions, which vastly helps guests which use native-style drivers. For certain VMI workloads, this provides a performance boost of up to 30%. We expect KVM and lguest to be able to achieve similar gains on I/O intensive workloads.

Won't these workloads be better off using paravirtualized drivers? i.e., do the native drivers with paravirt I/O instructions get anywhere

Re: [PATCH] Add I/O hypercalls for i386 paravirt

near the performance of paravirt drivers?

Yes, in general, this is true (better off with paravirt drivers). However, we have "paravirt" drivers which run in both fully-paravirtualized and fully traditionally virtualized environments. As a result, they use native port I/O operations to interact with virtual hardware.

Suffering from terminology overdose here: "fully traditionally virtualized, fully-paravirtuallized, para-fullyvirtualized".

Since this is only for newer kernels, won't updating the driver to use a hypercall be more efficient? Or is this for existing out-of-tree drivers?

Since not all hypervisors have paravirtualized driver infrastructures and guest O/S support yet, these hypercalls can be advantages to a wide range of scenarios. Using I/O hypercalls as such gives exactly the same performance as paravirt drivers for us, by eliminating the costly decode path, and the simplicity of using the same driver code makes this a huge win in code complexity.

Ah, seems the answer to the last question is yes.

--

error compiling committee.c: too many arguments to function

-

To unsubscribe from this list: send the line "unsubscribe linux-kernel" in the body of a message to majordomo@xxxxxxxxxxxxxxxxxxx

More majordomo info at <http://vger.kernel.org/majordomo-info.html>

Please read the FAQ at <http://www.tux.org/lkml/>