Subject: Re: CUDA support inside containers
Posted by abufrejoval on Wed, 16 Nov 2016 02:37:23 GMT
View Forum Message <> Reply to Message

Yes, the module is loaded and working on the hardware node: I'm testing in parallel doing strace via the CUDA device info utililty outside and inside to see where things start going wrong.

One of the first things the Nvidia runtime is doing is to check for the presence of the nvidia* loadable modules scanning /proc/modules and that file is always empty inside containers as far as Internet searches tell me. It then goes and tries to load a matching nvidia0 module which neither exists nor would make sense inside containers, I guess.

I couldn't actually find any information on how the GPU code is represented to a user process and perhaps there are good reasons details are somewhat scarce as process isolation and security are somewhat lower on the priority list for CUDA. I also wonder how process scheduling is done for GPU side processes and how that is coordinated with CPU side processes. I don't wonder too much, though, because I can only imagine that being ugly.

It generally seems to work, as I've run quite a few CUDA code samples in parallel without any evident problems: Resource contention could become an issue, given that GPU VRAM tends to be far more limited and then lack virtual memory management. Could be useful to manage CUDA memory limits on containers eventually.

Somehow I doubt the GPU uses paged virtual memory for its own code but I see DMA access from the GPU to normal 'CPU' memory mention in documents and that reminds me of some of the earliest classical security exploits on mainframes, where virtual channel programs would happily deliver you the clear text password file, virtual memory and OS security successfully hid from you.

I guess GPU memory would eventually be MMAPed to CPU address space and at least provide process isolation for CPU access via PTEs. Hopefully the GPU code then doesn't start copying data across process boundaries of distinct CUDA users, neither for CPU side memory nor for GPU side memory. Could be interesting for people trying to attack usage of CUDA for Blockchain type cryptography.

Long term I'm just hoping AMD style HSA will become the norm and GPU code will run in normally managed virtual HBM2 memory.

Short term I guess I need to find some way to either disable the CUDA sanity check or fake a /proc/modules file which makes the CUDA runtime happy enough.