Contribution ID: **61** Type: **not specified**

Fast Checkpoint Restore for GPUs

Monday, 20 September 2021 09:10 (40 minutes)

We recently announced our work to support Checkpoint and Restore with AMD GPUs. This was first time a device plugin is introduced and that deals with one of the most complex devices on the system i.e. GPU. We made several changes to CRIU, introduced new plugin hooks and reported some issues with CRIU.

https://github.com/RadeonOpenCompute/criu/tree/amd-criu-dev-staging/plugins/amdgpu#readme

While there were several new challenges that we faced to enable this work, we were finally able to support real tensorflow/pytorch work loads across multi-gpu nodes using criu and were also able to migrate the containers running gpu bound worklaods. We have another proposed talk where we'll talk about the bigger picture but in this talk, we'd like to specifically talk about our journey where we started with a small 64KB buffer object in GPU VRAM to Gigabytes of single VRAM buffer objects across GPUs. We started with /PROC/PID/MEM interface initially and then switched to a faster direct approach that only worked with large PCIE BAR GPUs but that was still slow. For instance, to copy 16GB of VRAM, it used to take ~15 mins with the direct approach on large bars and more than 45 mins with small bars. We then switched to using system DMA engines built into most AMD GPus and this resulted in very significant improvements. We can checkpoint the same amount of data within 10 seconds now. For this we initially modified libdrm but the maintainers didn't agree to change an private API to expose GEM handles to the userspace so we finally ended up make a kernel change and exporting the buffer objects in VRAM as DMABUF objects and then import in our plugin using libdrm.

We would also like to to talk about how we further optimized it using multithreading and also our experience with compression using criu-image-streamer to save time and space further. We also encountered limitation of google protobuf in handling large vram buffer objects.

Overall this was a very significant feature addition that made our work usable from a POC to handle real world huge machine learning and training workloads.

Thank you
Rajneesh

I agree to abide by the anti-harassment policy

I agree

Primary authors: Mr BHARDWAJ, Rajneesh (AMD); Mr KUEHLING, Felix (AMD); Mr YAT SIN, David (Mr)

Presenters: Mr BHARDWAJ, Rajneesh (AMD); Mr KUEHLING, Felix (AMD); Mr YAT SIN, David (Mr)

Session Classification: Containers and Checkpoint/Restore MC

Track Classification: Containers and Checkpoint/Restore MC