Extending SLURM with support for rCUDA

Why extending SLURM to support rCUDA?

Previous to this extension, SLURM did not support the virtual GPUs provided by rCUDA or any other remote GPU virtualization framework. That is, SLURM was only able to deal with real GPUs. Therefore, when a job included within its computing requirements one or more GPUs per node, SLURM tried to assign that job nodes owning the requested amount of GPUs, thus neglecting the benefits of remote GPU virtualization. By extending SLURM with support for rCUDA, it is possible to make it aware of the fact that now the assignment of GPUs is no longer constrained to the processes running in the same node they are located, but applications are able to make use of GPUs independently of the exact nodes where applications on one side and GPUs on the other are placed.

What does provide the integration of rCUDA with SLURM?

The new extension of SLURM allows:

- Scheduling virtual GPUs by SLURM in a user transparent way
- Scheduling virtual GPUs either as exclusive or shared resources
- Virtual GPUs and standard physical GPUs can be used at the same time

What does scheduling virtual GPUs either as exclusive or shared resources mean?

rCUDA allows to share remote GPUs among several jobs. Therefore, SLURM inherits that feature and can schedule, according to the cluster administration directions, the use of GPUs as in the traditional way, that is, a GPU is exclusively assigned to a process, or in a shared way, where a given GPU is granted to several processes. The amount of processes that actually share a given GPU currently depends on their memory requirements and will also depend in the near future on the computing availability of the GPU, which will be dynamically monitored.

Cluster throughput is increased with rCUDA+SLURM

The combination of rCUDA+SLURM in your cluster increases its productivity, as shown in the example below, where five different jobs are submitted for execution and must be scheduled by SLURM. The resources requested by each of the jobs are displayed.

SLURM without rCUDA

In the scheduling example above we can see that job number one cannot be scheduled because no node in the cluster satisfies the requirements of this job. After that error, the rest of jobs are scheduled and executed. Notice that job five cannot be immediately executed after submission despite of having the required resources available.

SLURM with rCUDA

When SLURM+rCUDA is in use, GPUs are logically decoupled from nodes and therefore any GPU can be assign to any job, independently of their location. In this way overall cluster throughput is increased. In the example above all the jobs are executed faster.

About rCUDA

In the context of HPC and datacenter clusters, the rCUDA framework grants CUDA-accelerated applications being executed in a server transparent access to GPUs installed in other server of the cluster. In this way, applications are not aware of being accessing an external device, as the rCUDA remote GPU virtualization framework hides all the details, while maintaining application performance.