Altair PBS Professional™ Support on NVIDIA DGX Systems

Scott Suchyta

Developed to meet the demands of AI and analytics, NVIDIA® DGX™ Systems are built on the revolutionary NVIDIA Volta™ GPU platform. Combined with innovative GPU-optimized software and simplified management tools, in addition to the industry-leading PBS Professional workload manager and job scheduler, these fully integrated solutions deliver groundbreaking performance and results.

NVIDIA DGX Systems are designed to give data scientists the powerful tools they need for AI exploration — from the desk to the data center to the cloud. With NVIDIA DGX Systems, data scientists can experiment faster, train larger models, and arrive at insights — all starting on day one. NVIDIA DGX Systems are:

- Powered by the NVIDIA DGX software stack
  - Integrated suite of optimized deep learning software
  - Simplified workload management
- Designed to boost productivity
  - Save hundreds of thousands of dollars in engineering effort
  - Avoid months of the lost productivity spent on IT
- Proven to accelerate return on investment (ROI) for AI
  - Get started in one day instead of weeks or months
  - Accelerate deep learning training by 140X

Introduction to PBS Professional

PBS Professional software optimizes job scheduling and workload management in high-performance computing (HPC) and artificial intelligence (AI) environments — clusters, clouds, and supercomputers — improving system efficiency and team productivity.

PBS Professional is dual-licensed — open source and commercial — providing the private sector and the public sector with the same capabilities, scalability, and robustness that is expected in resource management and job scheduling. Private-sector organizations tend to select the commercially licensed software for quality assurance, support, and software maintenance because businesses like having insurance that critical issues affecting them are addressed in a timely manner. Public-sector organizations (e.g., universities and research labs) that have staff with good knowledge and skills working with open-source software gravitate toward using open source.

PBS Professional Architecture & Components

PBS Professional consists of two major component types: user-level commands and system daemons/services. A brief description of each is given here to help you understand how the pieces fit together.

Commands – PBS Professional supplies both command-line programs that are POSIX 1003.2d-conforming and a graphical interface. These are used to submit, monitor, modify, and delete jobs. These client commands can be installed on any system type supported

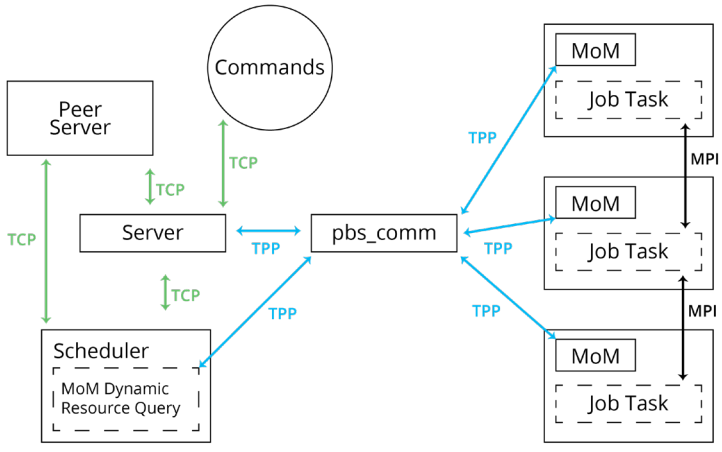
by PBS Professional and do not require local presence of any of the other PBS Professional components. There are three command classifications — user commands, which any authorized user can use; operator commands; and manager (or administrator) commands. Operator and manager commands require special privilege (like root access) to use.

Server – The server daemon/service, pbs_server, is the central focus of PBS Professional. All commands and other daemons/services communicate with the server via an Internet Protocol (IP) network. The server's main function is to provide basic batch services such as receiving/creating a batch job, modifying the job, and passing the job to the execution node. Normally, there is one server managing given set of resources.

Execution Node (MOM) – The execution node daemon/service, pbs_mom, is responsible for instantiating and monitoring the job process(es) — creating a new session for each job and gathering information about that job's resource usage. The pbs_mom service is informally called MOM, as it is a mother of all processes for that job. If the job requires multiple execution nodes, then Mother Superior is the MOM on the head or first host of a multi-host job. Mother Superior controls the job, communicates with the server, and controls and consolidates resource usage information.

Scheduler – The job scheduler daemon/service, pbs_sched, implements the site's scheduling policy, controlling when each job is run and on which resource. The scheduler may communicate with the various MOMs to query the state of system resources and with the server for availability of jobs to execute.

Communication Daemon – The communication daemon/service handles communication between the other PBS daemons. The PBS server, scheduler, and MOM daemons communicate with each other using TPP through the communication daemon, pbs_comm, except for scheduler-server and server-server communication, which uses TCP. The server, scheduler, and MOMs are communication endpoints, connected by one or more pbs_comm daemons. The following figure illustrates communication within a PBS complex using TPP.



Communication daemons are connected to each other. If there are multiple pbs_comms and two endpoints on different pbs_comms transmit data, communication between endpoints goes from the first endpoint to the endpoint's configured pbs_comm daemon, then to the pbs_comm configured for the receiving endpoint, and finally to the receiving endpoint.

Obtaining PBS Professional

As mentioned earlier, PBS Professional is dual-licensed and each version provides the same capabilities, scalability, and robustness. So, you have two options for obtaining PBS Professional:

1. PBS Professional open-source software: Go to **www.pbspro.org** or **www.github.com/PBSPro**.
2. PBS Professional commercial software: Contact **pbssales@altair.com** or request a free software trial from the Altair PBS Works™ website at **secure.altair.com/onlinestore**.

Configuring PBS Professional with GPUs

There are two options for configuring PBS Professional with GPUs and DGX systems. The first is to use what is called "exclusive mode access" and the second allows each GPU to be scheduled independently of the others.

The first step is to plan how you want to use the DGX system. There are two types of configurations:

1. The easiest configuration is to assume that a user gets exclusive access to the entire node. In this case the user gets the entire DGX system, i.e., access to all GPUs and CPU cores. No other users can access the resources while the first user is accessing them.

2. The second way is to make the GPUs a consumable resource. The user will then ask for the number of GPUs they need, ranging from 1 to 8 for the DGX-1 and 1 to 16 for the DGX-2.

Simple GPU Scheduling with Exclusive Node Access

If you're not interested in allowing simultaneous multiple jobs per compute node, you many not necessarily need to make PBS Professional aware of the GPUs in the system, and the configuration can be greatly simplified.

One way of scheduling a job to be exclusive on DGX systems is to create a custom Boolean resource and associate the custom resource with each DGX system. For example, as root on the PBS server, create custom Boolean resource p100.

```
qmgr –c "create resource p100 type=Boolean,flag=h"
```

Then associate the custom Boolean resource with each DGX system.

```
qmgr –c "set node ${dgx_hostname} resources_available.p100 = true"
```

To run a job requesting a p100 system, the user will request the p100 Boolean resource and place=excl via qsub.

```
$ qsub –I select=1:p100=true –I place=excl –– /usr/bin/nvidia–smi –L
```

This approach can be advantageous if you are concerned that sharing resources could result in performance issues on the node or that node resources could become overloaded. For example, in the case of a DGX-1,

if you think multiple users might overwhelm the 8TB NFS read cache, then you might want to consider using exclusive mode. Or, if you are concerned that users may use all the physical memory, causing page swapping with a corresponding reduction in performance, then exclusive mode may be useful.

Scheduling Resources at the Per-GPU Level

A second option for using PBS Professional is to treat the GPUs like a consumable resource (first-class resource) and allow users to request them in integer units (e.g., 1, 2, 3, etc.). PBS Professional can automatically detect the GPUs on the DGX and isolate the GPU(s) for the job. A very quick overview is below.

IMPORTANT: The pbs_cgroup hook restricts memory usage on shared nodes by default so that a user doesn't cause swapping with other user or system processes. On Ubuntu systems this is configurable via the file /etc/default/grub.

```
GRUB_CMDLINE_LINUX="cgroup_enable=memory swapaccount=1"
```

Otherwise, disable the memsw subsystem in the pbs_cgroup.json configuration file before importing (step 2) the updated configuration file.

As root on the PBS server:

1. Export the pbs_cgroups configuration file.

```
qmgr –c "export hook pbs_cgroups application/x–config default" > pbs_cgroups.json
```

2. Update the pbs_cgroups.json configuration file to enable the device's subsystem and add the nvidia-uvm parameter. Below is an example of the update (note the commas).

```
"devices" : {
    "enabled" : true,
    "exclude_hosts" : [],
    "exclude_vntypes" : [],
    "allow" : [
        "c 195:* m",
        "c 136:* rwm",
        ["infiniband/rdma_cm","rwm"],
        ["fuse","rwm"],
        ["net/tun","rwm"],
        ["tty","rwm"],
        ["ptmx","rwm"],
        ["console","rwm"],
        ["null","rwm"],
        ["zero","rwm"],
        ["full","rwm"],
        ["random","rwm"],
        ["urandom","rwm"],
        ["cpu/0/cpuid","rwm","*"],
        ["nvidia–modeset", "rwm"],
        ["nvidia–uvm", "rwm"],
        ["nvidia–uvm–tools", "rwm"],
        ["nvidiactl", "rwm"]        ]
},
```

3. Import the pbs_cgroup configuration file.

```
qmgr –c "import hook pbs_cgroups application/x–config default pbs_cgroups.json"
```

4. Enable the pbs_cgroup hook.

```
qmgr –c "set hook pbs_cgroups enabled = true"
```

5. Update the resources: parameter of the PBS scheduler configuration file to include ngpus.

```
resources: "ncpus, mem, arch, host, vnode, aoe, eoe, ngpus"
```

6. Update the flag of ngpus resource.

```
qmgr –c "set resource ngpus flag=nh"
```

Finally, as root on the DGX system(s), HUP or restart the pbs_mom daemon. This action will force pbs_mom to resend the hardware inventory with the PBS Professional server.

```
pkill –HUP pbs_mom
```

or

```
systemctl stop pbs ; systemctl start pbs
```

To run a job requesting GPU resources, the user will request the ngpus resource via qsub. For example, to run a job requiring two GPUs on a single DGX system the following qsub command can be used.

```
$ qsub –l select=1:ngpus=2 –– /usr/bin/nvidia–smi –L
```

Reference

For additional information about managing and using Altair PBS Professional, please consult **the online PBS Professional documentation** (Install Guide, Administrator Guide, and User Guide).

PBS Professional open-source sites can interact with our user community at **community.pbspro.org**.

PBS Professional commercial customers can send email to **pbssupport@altair.com**.