Guidelines for Using DGX GPU Cluster (with Sample Script)

Welcome to the IIT Kharagpur GPU Cluster! Please adhere to the guidelines below to ensure efficient system utilisation.

Accessing the CCDS DGX GPU Cluster: Use the following SSH command to connect:

```
ssh <username>@gpucluster.iitkgp.ac.in
```

The GPU cluster comprises five NVIDIA DGX H100 systems, each equipped with eight H100 Tensor Core GPUs and built-in high-speed NVMe storage, ensuring efficient AI training and inference. Three dedicated servers, each powered by an INTEL(R) XEON(R) GOLD 6530, host the virtual machines for the Master Node, Login Nodes, Slurm, OpenLDAP, and Grafana services. The servers are installed with Proxmox Virtual Environment to provide a High Availability (HA) system.

To further enhance data throughput and scalability, the cluster has been recently upgraded with a dedicated 1.1 PiB Parallel File System (PFS) storage infrastructure. This storage system is integrated using NVIDIA Quantum InfiniBand switching technology with 400 Gbps bandwidth, providing high-speed, low-latency communication between compute and storage layers.

This architecture significantly improves performance for large datasets, distributed AI workloads, and multi-node training environments. This setup is housed at the Paramshakti data centre and delivers a robust and scalable environment for advanced AI research and enterprise applications. Find more details at GPU Cluster.

Job Submission Guidelines:

1. **Partition and Node mapping:**
    - The cluster consists of five DGX nodes. Four nodes (dgx1–dgx4) are grouped under the **`dgx_all`** partition, while dgx5 is available under the **`dgx_ccds`** partition.
    - Each DGX node is equipped with 8 NVIDIA H100 GPUs.
    - Users are assigned access to either the **`dgx_all`** or **`dgx_ccds`** partition based on their allocation.
    - The **`dgx_all`** partition supports both single-node and multi-node job execution across dgx1–dgx4. Users requiring multiple nodes should specify the desired number of nodes using **`--nodes`** and request GPUs per node using **`--gres=gpu:<count>`** in their job script.
    - ***Select the appropriate partition when submitting your job***.
2. **QoS Restrictions:**
    - Each partition supports four distinct QoS (Quality of Service) policies: gpu2, gpu4, gpu6, and gpu8.

- Each QoS has defined constraints:

QoS Name	Max Jobs/User	Max CPUs	Min GPUs	Max GPUs	Max Wall Time
gpu2	3	56	1	2	72 hours (3 days)
gpu4	1	56	3	4	48 hours (2 days)
gpu6	1	112	5	6	24 hours (1 day)
gpu8	1	112	7	8	12 hours

- Maximum running jobs are three per user account.
- Jobs not meeting these requirements will remain pending with the reason displayed in the squeue command.

3. **Job Constraints:**
   - Jobs may be submitted on one or more nodes depending on the workload and resource availability. Specify the required number of nodes using **`--nodes=<N>`**. Refer to the GPU resource usage policy.
   - Jobs must be submitted using the SBATCH command from the login node (#SBATCH <job_script.sh>).
   - The number of tasks per node **(`--ntasks-per-node`**) and GPUs per node **(`--gres=gpu:<count>`**) must match the requested resources and the selected QoS.
   - Users must specify the correct QoS specifications and partition in their job script.

Data Management:

1. **Temporary Storage:**
   - Each user is allocated **1.50 TB** of storage in /home/<username>.
   - To utilise multi-node, store all datasets and job-related files in /home/<username>.
   - Each node's RAID directory (27TB) is mounted on the master node as **/dgx[1-5]**, allowing users to access their allocated RAID storage (~1TB) via cd /dgx1/<username>/ from the master node.
   - On the compute nodes (dgx[1-5]), the same directory is accessible as **/dgx<dgx no>/<username>**, ensuring seamless access across the cluster.
2. **Backup Policy:**
   - Transfer your data to your local system upon job completion to avoid auto-deletion.
   - Users are responsible for maintaining backups of their data. The cluster administration is not liable for any data loss.

3. **Auto-Deletion Policy:**
    - Data in /home/<username> is automatically deleted after **30 days** of inactive state.
    - Regular backups are essential to prevent data loss.

Resource Usage Policy:

1. The DGX GPU Cluster is intended for GPU-intensive AI, deep learning, and HPC workloads that require the computational capabilities of NVIDIA H100 GPUs.
2. Users whose workloads require less than 16 GB of GPU memory per GPU or are primarily CPU-intensive are strongly encouraged to use the Param Shakti HPC Cluster instead of the DGX GPU Cluster.
3. Judicious use of DGX resources helps minimise queue waiting times and ensures that GPU-intensive workloads receive timely access to the cluster.
4. Users are requested to select the most appropriate computing resource based on their application requirements to maximise overall cluster utilisation.

SAMPLE BATCH SCRIPT

```
#!/bin/bash
#SBATCH --job-name=job_name                          # Job name
#SBATCH --output=/home/<username>/output_%j.txt # Standard output
#SBATCH --error=/home/<username>/error_%j.txt      # Standard error
#SBATCH --partition=dgx_all       # Partition assigned to your account
#SBATCH --qos=gpu2                # QoS (must match the number of gpu)
#SBATCH --nodes=1                 # Number of nodes (increase for multi-node jobs)
#SBATCH --ntasks-per-node=2       # Number of tasks per node (max 8)
#SBATCH --gres=gpu:2              # GPUs per node (max 8)
#SBATCH --time=24:00:00           # Adjust wall time according to the selected QoS

# Navigate to the working directory (path for the data stored in any dgx[1-5] node, for eg, dgx1)
cd /dgx1/<username>/<path>

# Load necessary modules (optional)
module load cuda/12.4
module load python/3.10
# Activate your Python environment
source /home/<username>/miniconda3/bin/activate <env>

# Execute the training script
python train_model.py
```

Key Notes:

1. Replace <username> with your actual username.
2. Ensure all paths (/home/<username> and /dgx<dgx_no>/<username>) are correctly set.
3. Select the appropriate partition and QoS according to your assigned partition and GPU requirement.
4. For multi-node jobs, increase the value of --nodes to the required number of nodes. The value specified in --gres=gpu:<count> represents the number of GPUs per node.
5. Users must adhere to QoS policies when submitting jobs to avoid scheduling delays or rejections.
6. Several packages are available as modules in the cluster, which can be accessed using the command ‘module avail’ and loaded using the command ‘*module load <package>*’. If additional packages are required, they can be installed in the user's home directory.

For further assistance or inquiries, please contact the system administrator through the Institute intercom: **84604** or email at **shaktisupport@iitkgp.ac.in**.

COMMON ERRORS IN THE SCRIPT

Example 1: Mismatch between --gres=gpu and --qos (Incorrect GPU Count)

```
#!/bin/bash
#SBATCH --job-name=job1
#SBATCH --output=/home/<username>/output_%j.txt
#SBATCH --error=/home/<username>/error_%j.txt
#SBATCH --partition=dgx_all
#SBATCH --qos=gpu2                  # QoS allows max 2 GPUs
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=2
#SBATCH --gres=gpu:4                # Requesting 4 GPUs, but gpu2 allows max 2
#SBATCH --time=24:00:00

cd /dgx1/<username>/<path>
source /home/<username>/miniconda3/bin/activate <env>
python train_model.py
```

Error Explanation

- The gpu2 QoS only allows up to 2 GPUs, but the script requests 4 GPUs (--gres=gpu:4).
- Slurm will reject this job due to exceeding the allowed GPU count.
- Fix: Change --gres=gpu:4 to --gres=gpu:2.

Example 2: Wall Time Exceeds QoS Limit

```
#!/bin/bash
#SBATCH --job-name=job2
#SBATCH --output=/home/<username>/output_%j.txt
#SBATCH --error=/home/<username>/error_%j.txt
#SBATCH --partition=dgx_all
#SBATCH --qos=gpu4   # gpu4 has max wall time of 48 hours
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=2
#SBATCH --gres=gpu:3
#SBATCH --time=72:00:00  # Exceeding max wall time of 48 hours for gpu4

cd /dgx1/<username>/<path>
source /home/<username>/miniconda3/bin/activate <env>
python train_model.py
```

Error Explanation

- The gpu4 QoS has a maximum wall time of 48 hours, but the script requests 72 hours (--time=72:00:00).
- The job will be rejected or truncated to the max allowed wall time.
- Fix: Reduce --time to 48:00:00 or select an appropriate QoS.

Example 3: Insufficient GPUs for Selected QoS

```
#!/bin/bash
#SBATCH --job-name=job4
#SBATCH --output=/home/<username>/output_%j.txt
#SBATCH --error=/home/<username>/error_%j.txt
#SBATCH --partition=dgx_all
#SBATCH --qos=gpu6   # gpu6 requires min 5 GPUs
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=2
#SBATCH --gres=gpu:2  # Only 4 (2x2) GPUs requested, but gpu6 requires at least 5
#SBATCH --time=24:00:00

cd /home/<username>/<path>
source /home/<username>/miniconda3/bin/activate <env>
python train_model.py
```

Error Explanation

- The job requests 2 GPUs per node on 2 nodes, resulting in a total of 4 GPUs.
- However, the selected gpu6 QoS requires at least 5 GPUs per node (or the minimum GPU requirement defined by the QoS).
- Fix: Increase --gres=gpu to satisfy the QoS requirement or select a lower QoS (e.g., gpu4).

Example 4: Submitting a multi-node job from a Compute Node instead of the Home Directory

Incorrect job submission:

sbatch /dgx1/<username>/train_script.sh

Error Explanation

- Jobs should be submitted from the login node, not from a compute node.
- Submission from dgx1 (or any dgx node) will fail because Slurm requires multi-node job submission from the login node.

Fix: Run the job submission from the home directory:

cd ~
sbatch train_script.sh

==========xxx==========