BIU Slurm System

Slurm is a Linux open-source resource manager and job scheduler designed to utilize a fair share of the computing resources.
What Are Slurm Partitions?
Partitions in Slurm are sets of compute nodes grouped for specific types of jobs. Each partition has its own limits such as maximum job runtime, hardware (CPUs, GPUs), and access permissions. Partitions are similar to queues in other workload managers.
Programs (jobs) run via Slurm are sent to one or more of the physical servers (Nodes). Slurm is software that helps defining and executing these jobs, as well as managing users, permissions, and resource allocation. It helps track and display job details as well.
Users must select a partition that matches the resources their job needs. If not specified, the default partition will be used.

1. BIU Cluster partitions Summary

Cluster General Partitions Summary

Partition Name	Purpose	Nodes	Max Time	Access (Accounts)	MaxJobs Per User	GPUs /CPUs Allowed PU	Node Mem
generic	General GPU jobs	dsigpu[01–05] dsicsgpu[02-10]	4h	All users	4 jobs	2 GPUs	128G/ 192G
H200-4h	GPU jobs on H200	hpc8h200-01	4h	All users	4 jobs	2 GPUs	2T
H200-12h	GPU jobs on H200	hpc8h200-01	12h	All users	4 jobs	2 GPUs	2T
L4-4h	GPU jobs on L4	hpc8l4-01-01	4h	All users	4 jobs	2 GPUs	512G
L4-12h	GPU jobs on L4	hpc8l4-01-01	12h	All users	4 jobs	2 GPUs	512G
A100-4h	GPU jobs on A100	hpc2a100-01	4h	All users	4 jobs	2 GPUs	512G
cpu1T-24h	CPU jobs	hpccpu01	24h	All Users	8 jobs	48 CPUs	1T
cpu192G-48h	CPU jobs	dml[02-21]	48h	All Users	16 jobs	48 CPUs	192G

Cluster Private Partitions Summary

Partition Name	Purpose	Nodes	Max Time	Access (Accounts)	MaxJobs Per User	GPUs /CPUs Allowed PU	Node Mem
p_kugler	GPU jobs on A100 Kugler group	hpc2a100-01	Unlimited	Users of Prof. Kugler’s group	Unlimited	Unlimited	512G
p_amsterdamer	CPU jobs Amsterdamer group	dml[02-21]	Unlimited	Users of Prof. Amsterdamer’s group	Unlimited	Unlimited	192G
p_glickman	GPU jobs Glickman group	dsicsgpu02–dsicsgpu10	Unlimited	Users of Prof. Glickman’s group	Unlimited	Unlimited	192G

2. Job Submission Examples

Note: All job submission scripts must include lines that begin with #SBATCH. These are special directives that Slurm uses to allocate resources and control job behaviour.

How to create a job script file and submit it:

Basic CPU Job

```
#!/bin/bash
#SBATCH --job-name=cpu_test
#SBATCH --output=cpu_test_%j.out
#SBATCH --error=cpu_test_%j.err
#SBATCH --partition=<Partition-name>
#SBATCH --cpus-per-task=4
 #SBATCH --mem=<size>[M G]
< user commands>
< example: python script.py>
```

Important Note: By default, each job gets 1 CPU. Use **--cpus-per-task** to request more.
By default, each job gets 16G. Use --**mem** to request more.

Email notification: You can declare (enable) email notifications in your job Slurm script file:

```
#SBATCH --mail-user=your.email@example.com
#SBATCH --mail-type=[ALL, BEGIB,END,FAIL]
```

Basic GPU Job

```
#!/bin/bash
#SBATCH --job-name=gpu_test
#SBATCH --output=gpu_test_%j.out
#SBATCH --error=gpu_test_%j.err
#SBATCH --partition=<partition name>
#SBATCH --gres=gpu:1
 #SBATCH --mem=<size>[M G]
<user commands>
<example: python gpu_script.py>
```

Important Notes:
If you don't specify **--gres=gpu:N**, default GPU count is 0.
If you don't specify --**mem=<size>[M G]**, default is 16G
in case the partition includes several types of GPU's and you want a specific GPU, use:

```
#SBATCH --gres=gpu:<type>:<count>
```

Docker Job
If you are using docker:
1. Run the container in the foreground (**without using -d**).
2. Docker container name should be **slurm-$SLURM_JOB_ID** for traceability
Use this in your batch script:
DockerName=slurm-job-$SLURM_JOB_ID
docker run --name "$**DockerName**" --rm my-image:latest python script.py

Conda Job

```
#!/bin/bash
#SBATCH --job-name=conda_job
#SBATCH --output=conda_job_%j.out
#SBATCH --error=conda-job_%j.err
#SBATCH --partition=<partition name>
 #SBATCH --mem=<size>[M G]
source ~/miniconda3/etc/profile.d/conda.sh
conda activate myenv   # Or source your environment
python script.py
```

MATLAB job

```
#!/bin/bash
#SBATCH --job-name=matlab_job
#SBATCH --output=matlab_job_%j.out
#SBATCH --error=matlab-job_%j.err
#SBATCH --partition=<partition name>
 #SBATCH --mem=<size>[M G]
#SBATCH --cpus-per-task=48
<user commands>
matlab -batch "run(mymatlab.m ')"
<user commands>
```

When running MATLAB jobs with **parallel workers**, and submitting the job to Slurm with:
Important Note: When Running MATLAB with **parallel workers** and submitting a MATLAB job with Slurm using:
#SBATCH --cpus-per-task=48 **% <--** 48 is just an example

To ensure MATLAB uses **all the CPU cores allocated by Slurm**, add the following to your MATLAB job script (e.g. **mymatlab.m**):

```
num_workers = str2double(getenv('SLURM_CPUS_PER_TASK'));
c = parcluster('local');
c.NumWorkers = num_workers;    % or  exact number of cores you want to use (e.g. 48)
saveProfile(c);
```

3. How to submit the job

1. **Connect to a slurm login Server:**

   Log in to one of the available Slurm login servers using SSH:
   ssh slurm-login1.lnx.biu.ac.il
   or
   ssh slurm-login2.lnx.biu.ac.il
   or
   ssh slurm-login3.lnx.biu.ac.il

2. Submit Your Job**: sbatch myJob.sh**

3. Monitor your job using **squeue**
   After submission, Slurm will generate output and error files in your current working directory with the following format:

   <JobName>_<JobID>.out
   <JobName>_<JobID>.err
   Make sure you check output and error files for progress

---

4. Job Time Limits and Requeue Policy

Jobs submitted with **sbatch** will be automatically suspended and **requeue**d by the system when they reach the time limit of the partition they were submitted to (see the Max time in Part 1)

Checkpoints:
To benefit from requeuing, implement **checkpointing** in your application:
- Save progress periodically.
- On restart, resume from the latest checkpoint.

Without checkpointing, the required job will start from the beginning.

---

5. Interactive Jobs

```
srun --partition=<partition-name>  <command>
srun --partition=<partition-name> --pty bash
srun --partition=<partition-name> --gres=gpu:1 --pty bash
```

Important Notes:
- When using **srun**, your job will run interactively and remain active until it is manually stopped or reaches the **TimeLimit** of the partition.
- Interactive jobs launched this way will **not** be automatically requeued.
- Use **srun** only for quick tests, debugging, or development. For longer jobs or those requiring resilience to timeouts, use sbatch with checkpointing

6. Private Partitions and Accounts

Some partitions are private and limited to specific accounts and have **higher priority** in job scheduling: p_kugler, p_amsterdamer, p_glickman.

Example: p_kugler is accessible only to users in the Prof Kugler's research group.
If you belong to one of the private groups and want to run a job on one of the private partitions, you should use the relevant account for this partition, by adding to your job script the following line:

```
#SBATCH --account=<Account-name>
```

For Kuglers group use: ug_kugler as Account-name
For Amsterdamer group use: ug_amsterdamer as Account name
For Glikman group use: ug_cs_dsi as Account-name

To see which Slurm accounts you belong to use the command:

```
sacctmgr list associations where user=$USER format=User,Account
```

---

7. How to start a Jupyter Notebook session on the slurm cluster:

1. Connect to one of the slurm-login nodes and run the following command:

```
srun --partition=<partition> --gres=gpu:1 jupyter-notebook --no-browser --ip=0.0.0.0 &
```

- Note the **node name** and **port number** assigned by Slurm.
- Copy and save the URL that Jupyter prints (you'll need it later)

output example:

```
http://nodename:8888/tree?token=e34d73f70fa48254500e0556663db7b859985448f9d9829d
http://127.0.0.1:8888/tree?token=e34d73f70fa48254500e0556663db7b859985448f9d9829d
```

In this example **node name** and **port number,** assigned by Slurm, are: nodename and 8888

2. Set up port forwarding from your local desktop (PC):

On your local PC, run:

```
ssh -L <port-num>:localhost:<port-num> -J <username>@slurm-login <username>@<node>
```

- Replace <port-num> with the actual port number from step 1.
- Replace <username> with your username.
- Replace <node> with the node assigned by Slurm (from step 1).

3. Access the notebook:

- Paste the copied Jupyter URL (from step 1, second line) into your **local web browser**. You should now see the notebook interface running on the remote Slurm node.

Important Notes:

- It is recommended to perform **debugging and testing** in a partition with **less powerful CPU/GPU nodes**, to avoid occupying high-demand resources during development.
- Once your code is tested and ready, you can submit the job to your desired partition using **sbatch** command.
- Jobs will stop when: you close the notebook, time limit is reached or you exit the session.
- For production jobs, use sbatch and sbatch scripts (not notebooks).

8. Useful Slurm Commands

Submit a job	**sbatch job.sh**
Check your jobs	**squeue -u $USER**
Cancel a job	**scancel <job_id>**
View available partitions	**sinfo**

Version 1.4, 5 Aug 2025