

SLURM (Simple Linux Utility for Resource Management) is a powerful and open-source workload manager and job scheduler designed for high-performance computing (HPC) clusters. It efficiently manages resources and schedules jobs on compute nodes. Here's a detailed explanation of SLURM's architecture, its components, commands, and examples:

SLURM Architecture

SLURM architecture consists of three primary components:

1. Control Machine (Controller)

- **Role:** This machine runs the SLURM Control Daemon (slurmctld) and acts as the master node of the SLURM cluster.
- **Responsibilities:**
 - Job scheduling and dispatching.
 - Managing cluster state and resource allocation.
 - Communicating with compute nodes.
- **Command:**
 - To check the status of slurmctld:

```
bash
```

```
CopyEdit
```

```
systemctl status slurmctld
```

- To start the SLURM controller:

```
bash
```

```
CopyEdit
```

```
systemctl start slurmctld
```

2. Compute Nodes

- **Role:** These are the worker nodes where the actual computation takes place. Each compute node runs the SLURM Node Daemon (slurmd).
- **Responsibilities:**
 - Execute jobs assigned by the SLURM controller.
 - Communicate with the SLURM controller about resource status.
- **Command:**
 - To check the status of slurmd:

```
bash
```

```
CopyEdit
```

systemctl status slurmd

- To start the compute node daemon:

bash

CopyEdit

systemctl start slurmd

3. SLURM Database Daemon (Optional)

- **Role:** The SLURM Database Daemon (slurmdbd) is used for storing accounting information about jobs, users, and resource usage.
- **Responsibilities:**
 - Maintain a historical log of jobs.
 - Enable advanced reporting features.
- **Command:**
 - To check the status of slurmdbd:

bash

CopyEdit

systemctl status slurmdbd

- To start the SLURM database daemon:

bash

CopyEdit

systemctl start slurmdbd

SLURM Components

1. **Nodes:**
 - Represent the physical or virtual machines in the cluster.
 - Configured in the SLURM configuration file (/etc/slurm/slurm.conf).
2. **Partitions:**
 - Logical grouping of nodes into a queue-like structure for job submission.
 - Define the set of resources available for different users or purposes.
3. **Jobs:**
 - The computational tasks submitted by users to the SLURM scheduler.
4. **Schedulers:**

- Allocate resources based on policies and job priorities.

SLURM Workflow Diagram

1. **User** submits jobs to the SLURM controller using commands like sbatch or srun.
2. **SLURM Controller** (slurmctld) schedules the job on the appropriate **Compute Nodes** based on resource availability and scheduling policies.
3. **Compute Nodes** execute the job and return results to the user.
4. **SLURM Database** (slurmdbd) records job details (if configured).

SLURM Commands and Examples

1. Submit a Job:

- Command:

bash

CopyEdit

```
sbatch job_script.sh
```

- Example:

bash

CopyEdit

```
sbatch --time=00:10:00 --ntasks=4 my_job_script.sh
```

2. View Job Queue:

- Command:

bash

CopyEdit

```
squeue
```

- Output Example:

sql

CopyEdit

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	NODELIST(REASON)
-------	-----------	------	------	----	------	-------	------------------

12345	compute	test	user	R	0:05	1	node01
-------	---------	------	------	---	------	---	--------

3. Check Node Status:

- Command:

bash

CopyEdit

sinfo

- Output Example:

css

CopyEdit

PARTITION AVAIL TIMELIMIT NODES STATE NODELIST

compute* up infinite 10 idle node[01-10]

4. Cancel a Job:

- Command:

bash

CopyEdit

scancel JOBID

- Example:

bash

CopyEdit

scancel 12345

5. Check SLURM Logs:

- Command:

bash

CopyEdit

tail -f /var/log/slurm/slurmctld.log

Sample SLURM Configuration

A typical SLURM configuration file (/etc/slurm/slurm.conf) includes:

bash

CopyEdit

ClusterName=my_cluster

SlurmctldHost=controller

NodeName=node[01-10] CPUs=8 RealMemory=32000 State=UNKNOWN

PartitionName=compute Nodes=node[01-10] Default=YES MaxTime=INFINITE State=UP

Would you like me to create a diagram to visualize SLURM architecture?

4o

You said:

how to create a simple hpc job and submitted using slurm ?with command and example and job code ?

ChatGPT said:

ChatGPT

Creating and submitting a simple HPC job using SLURM involves writing a job script and submitting it with the sbatch command. Here's how you can do it:

Steps to Create and Submit an HPC Job Using SLURM

1. Write a Job Script

A SLURM job script is a shell script containing SLURM directives and the commands to execute your workload.

- **Example Job Script:**

bash

CopyEdit

```
#!/bin/bash
```

```
#SBATCH --job-name=my_hpc_job    # Job name
```

```
#SBATCH --output=my_hpc_job.out  # Standard output file
```

```
#SBATCH --error=my_hpc_job.err   # Standard error file
```

```
#SBATCH --ntasks=4              # Number of tasks (processes)
```

```
#SBATCH --time=00:10:00         # Time limit (hh:mm:ss)
```

```
#SBATCH --partition=compute     # Partition name
```

```
# Print the hostname of the node where the job runs
```

```
echo "Running on node: $(hostname)"
```

```
# Load any necessary modules
```

```
module load python/3.8
```

Run the job (example: a Python script)

```
python3 my_script.py
```

2. Create the Workload Code

For this example, let's assume your workload is a Python script (my_script.py).

- **Example Python Script** (my_script.py):

```
python
```

```
CopyEdit
```

```
# my_script.py
```

```
import time
```

```
from multiprocessing import Pool
```

```
def compute_task(x):
```

```
    return f"Task {x} completed on worker."
```

```
if __name__ == "__main__":
```

```
    print("Starting tasks...")
```

```
    with Pool(4) as pool:
```

```
        results = pool.map(compute_task, range(4))
```

```
    for result in results:
```

```
        print(result)
```

```
    print("All tasks completed.")
```

3. Submit the Job

- Save the job script as submit_job.sh and the Python script as my_script.py.
- Submit the job to SLURM:

```
bash
```

```
CopyEdit
```

```
sbatch submit_job.sh
```

4. Monitor the Job

- Check the status of the job queue:

bash

CopyEdit

squeue

Example Output:

sql

CopyEdit

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	NODELIST(REASON)
12345	compute	my_hpc_job	user	R	0:01	1	node01

- Check the output or error files:
 - my_hpc_job.out: Contains the standard output of the job.
 - my_hpc_job.err: Contains any errors from the job.

5. Cancel the Job (if needed)

- If the job needs to be stopped:

bash

CopyEdit

scancel JOBID

6. Example Output

Assume the Python script outputs the following and it is captured in my_hpc_job.out:

csharp

CopyEdit

Running on node: node01

Starting tasks...

Task 0 completed on worker.

Task 1 completed on worker.

Task 2 completed on worker.

Task 3 completed on worker.

All tasks completed.

Key SLURM Commands

1. **Submit a Job:** sbatch submit_job.sh
2. **Check Job Queue:** squeue
3. **View Node Status:** sinfo
4. **Cancel a Job:** scancel JOBID

Would you like additional examples, such as MPI-based jobs or GPU workloads?