Comandos Básico do Slum para Usuários

Um resumo dos principais comandos pode ser obtido em https://slurm.schedmd.com/pdfs/summary.pdf. Vejamos, a seguir, alguns desses comandos para as tarefas cotidianas.

Comandos Cotidianos

Aqui estão alguns exemplos de uso dos principais comandos do Slurm para gerenciar e monitorar o sistema de gerenciamento de trabalhos:

1. `sinfo`

O comando `sinfo` é usado para exibir o status dos recursos disponíveis no sistema, como nós e filas (partições).

- **Listar todas as filas (partições):**

```
sinfo -s
```

- **Listar todos os nós:**

```
sinfo -N
```

- **Listar informações detalhadas de todas as filas e nós:**

```
sinfo
```

- **Listar nós por estado (por exemplo, nós livres):**

```
sinfo -N -t idle
```

2. `squeue`

O comando `squeue` exibe o status dos trabalhos no sistema, incluindo trabalhos em execução, pendentes, etc.

- **Listar todos os trabalhos:**

```
squeue
```

- **Listar trabalhos de um usuário específico:**

```
squeue -u username
```

- **Listar trabalhos em uma fila específica:**

```
squeue -p partition_name
```

- **Listar detalhes de um trabalho específico:**

```
squeue -j job_id
```

3. `sbatch`

O comando `sbatch` é usado para submeter um script de trabalho ao Slurm.

- **Submeter um trabalho:**

```
sbatch job_script.sh
```

- **Submeter um trabalho com opções específicas (por exemplo, número de nós e tempo de execução):**

```
sbatch -N 2 -t 01:00:00 job_script.sh
```

4. `scancel`

O comando `scancel` é usado para cancelar trabalhos.

- **Cancelar um trabalho específico:**

```
scancel job_id
```

- **Cancelar todos os trabalhos de um usuário específico:**

```
scancel -u username
```

- **Cancelar todos os trabalhos em uma fila específica:**

```
scancel -p partition_name
```

5. `scontrol`

O comando `scontrol` é uma ferramenta administrativa para interagir com o Slurm e obter informações detalhadas sobre recursos e trabalhos.

- **Mostrar informações detalhadas sobre um nó específico:**

```
scontrol show node node_name
```

- **Mostrar informações detalhadas sobre um trabalho específico:**

```
scontrol show job job_id
```

- **Modificar parâmetros de um trabalho (por exemplo, aumentar o tempo de execução):**

```
scontrol update jobid=job_id TimeLimit=02:00:00
```

6. `srun`

O comando `srun` é usado para executar um trabalho interativo ou para iniciar uma tarefa em um ambiente de trabalho em lote.

- **Executar uma tarefa interativa:**

```
srun --pty /bin/bash
```

- **Executar um comando específico:**

```
srun hostname
```

- **Executar um trabalho com especificação de recursos (por exemplo, número de nós e CPUs por tarefa):**

```
srun -N 2 -n 4 ./meu_programa
```

7. `sacct`

O comando `sacct` exibe informações sobre os trabalhos já concluídos.

- **Listar trabalhos concluídos do usuário atual:**

```
sacct
```

- **Listar detalhes de um trabalho específico:**

```
sacct -j job_id
```

- **Listar trabalhos concluídos em um período específico:**

```
sacct --starttime=2023-06-01 --endtime=2023-06-30
```

8. `sstat`

O comando `sstat` exibe informações sobre o status dos trabalhos em execução.

- **Listar status de um trabalho específico:**

```
sstat -j job_id
```

- **Listar status com informações específicas (por exemplo, uso de CPU e memória):**

```
sstat -j job_id --format=JobID,MaxRSS,MaxVMSize,AveCPU
```

Esses comandos são fundamentais para gerenciar e monitorar trabalhos no Slurm, permitindo que os usuários submetam, monitorem e cancelem trabalhos de maneira eficiente.

Comandos na prática

Submissão de Jobs

Para submeter um job, o usuário cria um script de submissão de job. Aqui está um exemplo simples de um script chamado `job_script.sh`:

```
#!/bin/bash
#SBATCH --job-name=meu_job
#SBATCH --output=meu_job.out
#SBATCH --error=meu_job.err
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=4
#SBATCH --mem=8G
#SBATCH --time=01:00:00
#SBATCH --gres=gpu:1

echo "Job iniciado no nó: $(hostname)"
echo "Usando $(nproc) núcleos de CPU"
./meu_programa
sleep 60
echo "Job finalizado"
```

Neste script, o job será executado por 1 hora, usando 4 CPUs, 8 GB de memória e 1 GPU. O output e erros serão salvos em `meu_job.out` e `meu_job.err`, respectivamente.

Para submeter este job, o usuário utiliza o comando `sbatch`:

```
sbatch job_script.sh
```

Monitoramento de Jobs

- **Verificar o status dos jobs:**

  Para verificar o status dos jobs na fila, o usuário pode usar o comando `squeue`:

  ```
  squeue
  ```

  Isso exibirá uma lista de jobs, com informações sobre o estado de cada job.

- **Verificar detalhes de um job específico:**

  Para obter detalhes de um job específico, use `scontrol show job` seguido do ID do job:

```
scontrol show job <job_id>
```

- **Verificar o uso de recursos:**

  Para verificar o uso de recursos, como CPUs e memória, durante a execução do job, o comando `sacct` pode ser usado:

```
sacct -j <job_id> --
format=JobID,JobName,Partition,Account,AllocCPUs,State,ExitCode,MaxRSS,Elapsed
```

- **Cancelar um job:**

  Se o usuário precisar cancelar um job, o comando `scancel` pode ser utilizado:

```
scancel <job_id>
```

Exemplo Prático

Aqui está um exemplo completo de um usuário submetendo e monitorando um job:

1. **Criação do Script:**

   O usuário cria um arquivo chamado `meu_job_script.sh` com o seguinte conteúdo:

```
#!/bin/bash
#SBATCH --job-name=teste
#SBATCH --output=teste.out
#SBATCH --error=teste.err
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=2
#SBATCH --mem=4G
#SBATCH --time=00:30:00
#SBATCH --gres=gpu:1

echo "Iniciando job no nó: $(hostname)"
sleep 60
echo "Job finalizado"
```

2. **Submissão do Job:**

```
sbatch meu_job_script.sh
```

   Uma mensagem como `Submitted batch job 23` será retornada. Assumamos que `job_id = 23`

3. **Monitoramento do Job:**

- Verificar a fila:

```
squeue
```

- Obter detalhes do job:

```
scontrol show job 23
```

- Verificar o uso de recursos:

```
sacct -j 23 --
format=JobID,JobName,Partition,Account,AllocCPUs,State,ExitCode,MaxRSS,Elapsed
```

- Cancelar o job, se necessário:

```
scancel 23
```