Universidade Estadual de Feira de Santana

Tutorial de uso do SLURM

Feira de Santana - BA

Março, 2016

1 Introdução

Tutorial criado por Victor Oliveira, voluntário do Laboratório de Computação de Alto Desempenho, com o objetivo de orientar aos usuários do cluster do Lacad como se usa o SLURM.
O conteúdo deste tutorial foi criado para fins de pesquisa e pode ser usado livremente desde que citada a fonte.
O LaCAD não se responsabiliza pelo uso dessas informações.

1.1 O que é o Slurm

o Slurm (Simple Linux Utility for Resource Management) é um sistema gerenciador de submissão de tarefas para pequenos e grandes clusters Linux. Possui diversas vantagens, pois não requer modificação do kernel, é open-source, tolerante a falhas e escalável. O processo do Slurm consiste em: alocar exclusivamente ou não exclusivamente recursos para os usuários por uma certa quantia de tempo, depois um framework é responsável por executar e monitorar a tarefa que o usuário submeteu, e por fim ele administra a fila de tarefas pendentes.

1.2 Siglas e Termos

Nodes Refere-se aos computadores ligados ao cluster.

Partition Grupo de nós, separado em um conjunto lógico. São como filas de *jobs*, onde há restrições de tamanho, tempo, permissão, entre outros.

Jobs Alocação de recurso destinada ao usuário por um período, neste tutorial um Job será referido como tarefa.

Job steps Os passos que serão executados em um *job*.

1.3 Comandos

sacct - Usado para reportar informações sobre *jobs* em atividade ou completos.
salloc - Usado para alocar recursos para um *job*. Normalmente é usado para alocar o recurso e surgir um *shell* de comandos. Nesse shell é usado o comando srun para iniciar a tarefa.
sattach - Usado para anexar saídas ou entradas para um *job* em execução.
sbatch - Usado para submeter um *job* para executar depois. Esse script deve conter comandos srun para executar tarefas em paralelo.
sbcast - Usado para transferir um arquivo de um disco local para para o disco de um nó que foi alocado para *jobs*. Pode ser usado para aperfeiçoar o desempenho em relação ao uso de arquivos compartilhados, ou melhorar o uso de nós sem disco.
scancel - Usado para cancelar um *job* pendente ou em execução.
scontrol - Ferramenta administrativa para ver ou modificar o estado do SLURM. Alguns comandos só são executados como root.
sinfo - Exibe o informações sobre partições ou nós que são gerenciados pelo SLURM.
smap - Semelhante ao sinfo, mas exibe informações de *jobs*, partições ou nós gráficamente.

squeue - Informa o estado de um *job* ou *passos de um trabalho*. Há uma variedade de opções para exibir. Por padrão, exibe os *trabalhos* por ordem de prioridade.
srun - Usado para submeter um *trabalho* para execução. Há varias opções para especificar o uso de recursos, como: número de nós, número de processdores, quais nós usar, caracteristicas do nó(memória, espaço em disco), entre outros. Um *trabalho* pode conter uma série de *passos de trabalho* que executam sequencialmente ou paralelamente em nós independentes ou compartilhados dentro de uma alocação.
strigger - Usado para enviar, receber ou ver eventos como nós caídos ou *trabalhos* perto de esgotar seu tempo.
sview - Interface gráfica para o usuário receber e atualizar informações de estado dos *trabalhos*, partições ou nós.

2 Comandos informativos no SLURM.

Usando alguns dos comandos anteriores, o usuário pode obter diversas informações sobre o sistema que o SLURM está hospedado. Como exemplo de comandos, temos o scontrol e o sinfo.

2.1 scontrol

Usar o comando scontrol no Umbu retornará:

```
comando: scontrol show partitions
PartitionName=DEBUG
    AllowGroups=ALL AllowAccounts=ALL AllowQos=ALL
    AllocNodes=umbu Default=NO QoS=N/A
    DefaultTime=NONE DisableRootJobs=NO ExclusiveUser=NO GraceTime=0 Hidden=NO
    MaxNodes=UNLIMITED MaxTime=UNLIMITED MinNodes=1 LLN=NO
    MaxCPUsPerNode=UNLIMITED
    Nodes=(null)
    Priority=1 RootOnly=NO ReqResv=NO Shared=NO PreemptMode=OFF
    State=UP TotalCPUs=0 TotalNodes=0 SelectTypeParameters=N/A

PartitionName=CLUSTER
    AllowGroups=ALL AllowAccounts=ALL AllowQos=ALL
    AllocNodes=umbu Default=YES QoS=N/A
    DefaultTime=NONE DisableRootJobs=NO ExclusiveUser=NO GraceTime=0 Hidden=NO
    MaxNodes=UNLIMITED MaxTime=UNLIMITED MinNodes=1 LLN=NO
    MaxCPUsPerNode=UNLIMITED
    Nodes=umbu-1-[1-16]
    Priority=1 RootOnly=NO ReqResv=NO Shared=NO PreemptMode=OFF
    State=UP TotalCPUs=64 TotalNodes=16 SelectTypeParameters=N/A
```

O que indica informações sobre as partições disponíveis no Umbu, que são as partições DEBUG e CLUSTER. Com ele podemos visualizar informações sobre quantidade de nós ou permissões.

2.2 sinfo

```
comando: sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
DEBUG        up   infinite      0    n/a
CLUSTER*     up   infinite     11  down* umbu-1-[1,7-16]
CLUSTER*     up   infinite      5   idle umbu-1-[2-6]
```

Já no comando sinfo as informações são menos detalhadas, porém mais fácil de visualizar. A aba partition indica a partição(caso esteja marcada com um asterisco, significa que a partição é default), avail indica se a partição está funcionando ou não, timelimit indica o limite de tempo(que nesse caso não foi configurado), nodes indica a quandidade de nós, state indica o estado, no exemplo há uma partição inativa e outra sem responder, nodelist indica quais são os nós participantes.

2.3 squeue

Já o comando squeue retorna a fila de tarefas submetidas ao SLURM, como por exemplo:

```
JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
2         debug myscript   victor PD       0:00      1 (Resources)
```

Onde o JOBID representa o ID da tarefa submetida, Partition é a partição onde está sendo ou será executada, NAME é o nome do script submetido, USER é o nome do usuário que submeteu, ST é o estado da tarefa(R - Rodando, PD - Pendente), TIME o tempo de execução(no exemplo, como a tarefa está pendente, o tempo de execução é zero), NODES são quantos nós foram destinados. NODELIST é a lista de nós usados na aplicação que está rodando, caso ela não esteja rodando ele exibe a razão. Neste exemplo, está escrito Resources porque a tarefa está esperando por recursos para a execução, caso estivesse escrito Priority, a tarefa estaria em espera por conta de outra com mais prioridade.

3 Submetendo scripts no SLURM

Para executar algum algoritmo no SLURM da maneira mais simples, o comando srun pode ser usado da seguinte maneira:
$ srun nome_do_arquivo
ele executará da maneira padrão no sistema, ou seja, dedicará apenas um nó à execução, não gerará o arquivo de saída e o usuário deverá manter o terminal aberto até o fim da execução onde a saída será escrita.
Para submeter um trabalho ao SLURM, é necessário criar um script em batch, segundo o exemplo a seguir(teste.sh):

```
#!/bin/sh
#SBATCH --time=1
/bin/hostname
```

Onde no exemplo, o comando hostname será executado com uma restrição de no máximo 1 minuto. Após a criação do script, o seguinte comando deve ser executado:

```
$ sbatch teste.sh
```

A saída da execução é salva em um arquivo, na pasta onde foi realizada a submissão, no formato: slurm-<id>.out, onde <id> é o número do id da *tarefa*. Os seguintes parâmetros também podem ser adicionados ao script.
–time(-t)- tempo máximo de execução("minutos","minutos:segundos","horas:minutos:segundos", "dias-horas").
–cpus-per-task(-c)- número de núcleos por processo(Para multithread).
–nodes(-N)- número mínimo de servidores.
–ntasks(-n)- número total de processos.
–output(-o)- especifica o arquivo de saída.
–job-name(-J)- especifica um nome para a tarefa.

3.1 Tarefas sequenciais

3.1.1 Tarefas com um nó e um núcleo

O exemplo a seguir ilustra um script para tarefas sequenciais, ou seja, com apenas um nó e um núcleo.

```
#!/bin/bash
#SBATCH -J nome_da_tarefa
#SBATCH --partition nome_da_particao
#SBATCH --nodes 1
#SBATCH --ntasks 1
#SBATCH --cpus-per-task 1
```

3.1.2 Tarefas multithread

Apenas um nó, dois processos e 4 núcleos.

```
#SBATCH -J nome_da_tarefa
#SBATCH --partition nome_da_particao
#SBATCH --nodes 1
#SBATCH --ntasks 2
#SBATCH --cpus-per-task 4
```

3.2 Tarefas Paralelas(MPI)

Agora, o número de nós deve ser modificado.

```
#SBATCH -J nome_da_tarefa
#SBATCH --partition nome_da_particao
#SBATCH --nodes 4
#SBATCH --ntasks 32
#SBATCH --cpus-per-task 1
```

3.2.1 Aplicações MPI Multithread

Para aplicações MPI multithread, além de ter mais de um nó, é necessário mais de um núcleo por tarefa(cpus-per-task).

```
#SBATCH -J nome_da_tarefa
#SBATCH --partition nome_da_particao
#SBATCH --nodes 8
#SBATCH --ntasks 32
#SBATCH --cpus-per-task 2
```

Referências

[1] Documentação do SLURM. Disponível em http://slurm.schedmd.com/documentation.html, acesso em Março de 2016.