SLURM: Advanced Usage

Rod Schultz
email: rod.schultz@bull.com

Architect of an Open World

Outline

- Dynamic Allocation (Growing | Shrinking)
- Generic Resources
- Reservations

Dynamic Allocation (Shrink)

scontrol has the ability to shrink a job's size.

Use a either

- **`scontrol update JobId=# NumNodes=#`**
- **`scontrol update JobId=# NodeList=<names>".`**

These command generates a script (slurm_job_#_resize.sh) to be executed in order to reset SLURM environment variables for proper execution of subsequent job steps.

See **FAQ 24** for more details.

Shrink Allocation Example

Sample batch script: shrink.sh

```
#!/bin/bash
srun -N4 --tasks-per-node=2 /app/slurm/rbs/local/bin/rbsrun
echo "Resize allocation"
scontrol update jobid=$SLURM_JOB_ID NumNodes=1
RESIZENAME="_resize.sh"
echo "Environment Variables to shrink allocation"
cat slurm_job_$SLURM_JOB_ID$RESIZENAME
. ./slurm_job_$SLURM_JOB_ID$RESIZENAME
echo Now with only one node
scontrol show jobid=$SLURM_JOB_ID
srun -N1 -n1 --ntasks-per-node=1 /app/slurm/rbs/local/bin/rbsrun

> sbatch -N4 --tasks-per-node=2 shrink.sh
Submitted batch job 87
```

Shrink Allocation Example (1)

Step 0 Results (rbsrun is test program displaying allocation)

```
srun -N4 --tasks-per-node=2 /app/slurm/rbs/local/bin/rbsrun
SLURM_Job=87,Step=0,Task=4 Node=trek2  mask=0x1 CGroupCPUset=0,2
SLURM_Job=87,Step=0,Task=5 Node=trek2  mask=0x4 CGroupCPUset=0,2
SLURM_Job=87,Step=0,Task=3 Node=trek1  mask=0x4 Cpus_allowed list: 2
SLURM_Job=87,Step=0,Task=0 Node=trek0  mask=0x1 CGroupCPUset=0,2
SLURM_Job=87,Step=0,Task=2 Node=trek1  mask=0x1 Cpus_allowed list: 0
SLURM_Job=87,Step=0,Task=1 Node=trek0  mask=0x4 CGroupCPUset=0,2
SLURM_Job=87,Step=0,Task=6 Node=trek3  mask=0x101 Cpus_allowed list: 0,8
SLURM_Job=87,Step=0,Task=7 Node=trek3  mask=0x202 Cpus_allowed list: 1,9
```

Shrink allocation

```
scontrol update jobid=$SLURM_JOB_ID NumNodes=1
To reset SLURM environment variables, execute
  For bash or sh shells:  . ./slurm_job_87_resize.sh
  For csh shells:         source ./slurm_job_87_resize.csh
```

Contents of slurm_job_87_resize.sh

```
cat slurm_job_$SLURM_JOB_ID$RESIZENAME
export SLURM_NODELIST="trek0"
export SLURM_JOB_NODELIST="trek0"
export SLURM_NNODES=1
export SLURM_JOB_NUM_NODES=1
export SLURM_JOB_CPUS_PER_NODE="2"
unset SLURM_TASKS_PER_NODE
```

Shrink Allocation Example (2)

Shrink Job

```
scontrol update jobid=$SLURM_JOB_ID NumNodes=1
scontrol show jobid=$SLURM_JOB_ID
JobId=87 Name=shrink.sh
   UserId=slurm(200) GroupId=slurm(200)
   Priority=4294901754 Account=slurm QOS=normal
   JobState=RUNNING Reason=None Dependency=(null)
   Requeue=1 Restarts=0 BatchFlag=1 ExitCode=0:0
   RunTime=00:01:02 TimeLimit=UNLIMITED TimeMin=N/A
   SubmitTime=2011-07-18T07:44:49 EligibleTime=2011-07-18T07:44:49
   ResizeTime=2011-07-18T07:44:50
   StartTime=2011-07-18T07:44:49 EndTime=Unknown
   PreemptTime=NO_VAL SuspendTime=None SecsPreSuspend=0
   Partition=all AllocNode:Sid=sulu:2043
   ReqNodeList=(null) ExcNodeList=(null)
   NodeList=trek0
   BatchHost=trek0
   NumNodes=1 NumCPUs=2 CPUs/Task=1 ReqS:C:T=*:*:*
   MinCPUsNode=2 MinMemoryNode=0 MinTmpDiskNode=0
   Features=(null) Gres=(null) Reservation=(null)
   Shared=OK Contiguous=0 Licenses=(null) Network=(null)
   Command=/app/slurm/rbs/_Scripts/shrink.sh
   WorkDir=/app/slurm/rbs/_Scripts
   Switches=0 Wait-for-Switch=0 (seconds)
```

Step 1 (Shrinked) Job Results

```
srun -N1 -n1 --ntasks-per-node=1 /app/slurm/rbs/local/bin/rbsrun
SLURM_Job=87,Step=1,Task=0 Node=trek0  mask=0x1 CGroupCPUset=0,2
```

Dynamic Allocation (Expand)

scontrol has the ability to expand a job's size.
Some restrictions apply. See **FAQ 24** for more details.

Inside sbatch (salloc) do another sbatch (salloc), and 'expand' with the dependency option.

```
sbatch -N2 --dependency=expand:$SLURM_JOBID expander.sh
```

Inside the new sbatch (salloc), release the resources

```
scontrol update jobid=$SLURM_JOBID NumNodes=0
exit
```

In original job, pickup the resources

```
scontrol update jobid=$SLURM_JOB_ID NumNodes=ALL
```

This commands generates a script (slurm_job_#_resize.sh) to be executed in order to reset SLURM environment variables for proper execution of subsequent job steps.

```
. ./slurm_job_#_resize.sh
```

Expand Allocation Example

batch script: expand.sh

```
#!/bin/bash
srun -N $SLURM_JOB_NUM_NODES /app/slurm/rbs/local/bin/rbsrun
echo "Resize allocation"
sbatch -N2 --dependency=expand:$SLURM_JOBID expander.sh
Sleep 5 let the new resources be allocated.
Echo “Pick up new resources”
scontrol update jobid=$SLURM_JOB_ID NumNodes=ALL
RESIZENAME="_resize.sh"
echo "Environment Variables to expand allocation"
cat ./slurm_job_$SLURM_JOB_ID$RESIZENAME
. ./slurm_job_$SLURM_JOB_ID$RESIZENAME
echo Now with 3 nodes
srun -N $SLURM_JOB_NUM_NODES /app/slurm/rbs/local/bin/rbsrun
Exit 0
```

batch script to relinquish additional resources: expander.sh

```
#!/bin/bash
echo "Relenquish resources"
echo
scontrol update jobid=$SLURM_JOBID NumNodes=0
Exit 0
```

Expand Allocation Example (1)

```
> sbatch -N2 expand.sh
```

Step 0 Results (rbsrun is test program displaying allocation)

```
srun -N $SLURM_JOB_NUM_NODES /app/slurm/rbs/local/bin/rbsrun
SLURM_Job=17,Step=0,Task=0 Node=trek0  mask=0xff Cpus_allowed list: 0-7
```

Expand Job

```
echo "Resize allocation"
sbatch -N2 --dependency=expand:$SLURM_JOBID expander.sh
Sleep 5 let the new resources be allocated.
Echo "Pick up new resources"
scontrol update jobid=$SLURM_JOB_ID NumNodes=ALL
```

Contents of slurm_job_87_resize.sh

```
cat ./slurm_job_$SLURM_JOB_ID$RESIZENAME
Environment Variables to expand allocation

export SLURM_NODELIST="trek[0-2]"
export SLURM_JOB_NODELIST="trek[0-2]"
export SLURM_NNODES=3
export SLURM_JOB_NUM_NODES=3
export SLURM_JOB_CPUS_PER_NODE="8(x3)"
unset SLURM_TASKS_PER_NODE
```

Expand Allocation Example (2)

Step 1 Results

```
. ./slurm_job_$SLURM_JOB_ID$RESIZENAME
echo Now with 3 nodes
srun -N $SLURM_JOB_NUM_NODES /app/slurm/rbs/local/bin/rbsrun

SLURM_Job=17,Step=1,Task=2 Node=trek2  mask=0xff Cpus_allowed list: 0-7
SLURM_Job=17,Step=1,Task=0 Node=trek0  mask=0xff Cpus_allowed list: 0-7
SLURM_Job=17,Step=1,Task=1 Node=trek1  mask=0xff Cpus_allowed list: 0-7
```

Generic Resource (GRES)

Generic Resources (GRES) are resources associated with a specific node that can be allocated to jobs and steps. The most obvious example of GRES use would be GPUs. GRES are identified by a specific name and use an optional plugin to provide device-specific support.

For more infromation see the html page: **gres.html**

Generic Resource Configuration

SLURM supports no generic resourses in the default configuration. One must explicitly specify which resources are to be managed in the **slurm.conf** configuration file. The configuration parameters of interest are:

- **GresTypes** a comma delimited list of generic resources to be managed (e.g. GresTypes=gpu,nic). This name may be that of an optional plugin providing additional control over the resources.
- **Gres** the specific generic resource and their count associated with each node (e.g. NodeName=linux[0-999] Gres=gpu:8,nic:2) specified on all nodes and SLURM will track the assignment of each specific resource on each node. Otherwise SLURM will only track a count of allocated resources rather than the state of each individual device file.

Generic Resource Configuration (1)

Each **compute node** with generic resources must also contain a **gres.conf** file describing which resources are available on the node, their count, associated device files and CPUs which should be used with those resources. The configuration parameters available are:

The configuration parameters available are:

- **Name** name of a generic resource (must match GresTypes values in slurm.conf).
- **Count** Number of resources of this type available on this node.
- **CPUs** Specify the CPU index numbers for the specific CPUs which can use this resources.
- **File** Fully qualified pathname of the device files associated with a resource. The name can include a numberic range suffix to be interpretted by SLURM (e.g. File=/dev/nvidia[0-3]). This field is generally required if enforcement of generic resource allocations is to be supported

Generic Resource Configuration (2)

gres.conf example

```
# Configure support for our four GPUs
Name=gpu File=/dev/nvidia0 CPUs=0,1
Name=gpu File=/dev/nvidia1 CPUs=0,1
Name=gpu File=/dev/nvidia2 CPUs=2,3
Name=gpu File=/dev/nvidia3 CPUs=2,3
```

Generic Resource Running Jobs

The –gres option has been added to salloc, sbatch, and srun.

--gres=<list>
The format of each entry on the list is "name[:count[*cpu]]". The name is that of the consumable resource. The count is the number of those resources with a default value of 1. The specified resources will be allocated to the job on each node allocated unless "*cpu" is appended, in which case the resources will be allocated on a per cpu basis.

Examples --gres=gpus:2*cpu

Generic Resource Running Jobs (1)

Job steps can be allocated generic resources from those allocated to the job using the --gres option with the srun command as described above. By default, a job step will be allocated none of the generic resources allocated to the job, but must explicitly request desired generic resources. The job step can be allocated specific generic resources and those resources will not be available to other job steps.

A simple example is shown below.

```
#!/bin/bash
#
# gres_test.bash
# Submit as follows:
# sbatch --gres=gpu:4 -n4 -N1-1 gres_test.bash
#
srun --gres=gpu:2 -n2 --exclusive show_device.sh &
srun --gres=gpu:1 -n1 --exclusive show_device.sh &
srun --gres=gpu:1 -n1 --exclusive show_device.sh &
wait
```

Reservations

A resource reservation identifies the resources in that reservation and a time period during which the reservation is available. The resources which can be reserved include nodes and/or licenses. Note that resource reservations are not compatible with SLURM's gang scheduler plugin since the termination time of running jobs is not possible to accurately predict.

Reservation Example (1)

One common mode of operation for a reservation would be to reserve an entire computer at a particular time for a system down time. The example below shows the creation of a full-system reservation at 16:00 hours on 6 February and lasting for 120 minutes. The "maint" flag is used to identify the reservation for accounting purposes as system maintenance.

```
$ scontrol create reservation starttime=2009-02-06T16:00:00 \
   duration=120 user=root flags=maint,ignore_jobs nodes=ALL
Reservation created: root_3

$ scontrol show reservation
ReservationName=root_3 StartTime=2009-02-06T16:00:00
   EndTime=2009-02-06T18:00:00 Duration=120
   Nodes=ALL NodeCnt=20
   Features=(null) PartitionName=(null)
   Flags=MAINT,SPEC_NODES,IGNORE_JOBS Licenses=(null)
   Users=root Accounts=(null)
```

Reservation Example (2)

Another mode of operation would be to reserve specific nodes for an indefinite period in order to study problems on those nodes. This could also be accomplished using a SLURM partition specifically for this purpose, but that would fail to capture the maintenance nature of their use.

```
$ scontrol create reservation user=root starttime=now \
   duration=infinite flags=maint nodes=sun000
Reservation created: root_5

$ scontrol show res
ReservationName=root_5 StartTime=2009-02-04T16:22:57
   EndTime=2009-02-04T16:21:57 Duration=4294967295
   Nodes=sun000 NodeCnt=1
   Features=(null) PartitionName=(null)
   Flags=MAINT,SPEC_NODES Licenses=(null)
   Users=root Accounts=(null)
```

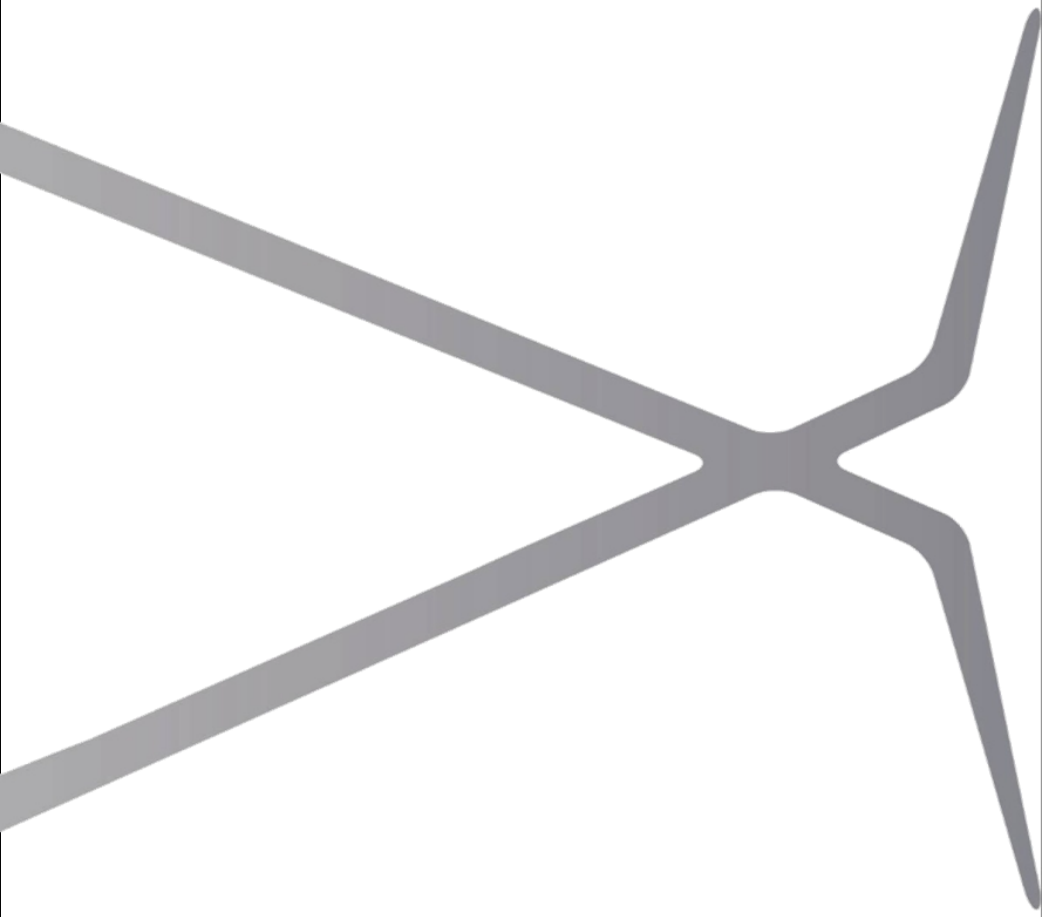
Reservation Example (3)

Our final example is to reserve ten nodes in the default SLURM partition starting at noon and with a duration of 60 minutes occurring daily. The reservation will be available only to users alan and brenda.

```
$ scontrol create reservation user=alan,brenda starttime=noon duration=60
flags=daily nodecnt=10
Reservation created: alan_6

$ scontrol show res
ReservationName=alan_6 StartTime=2009-02-05T12:00:00
   EndTime=2009-02-05T13:00:00 Duration=60
   Nodes=sun[000-003,007,010-013,017] NodeCnt=10
   Features=(null) PartitionName=pdebug
   Flags=DAILY Licenses=(null)
   Users=alan,brenda Accounts=(null)
```

Note that specific nodes to be associated with the reservation are made immediately after creation of the reservation. This permits users to stage files to the nodes in preparation for use during the reservation.



bullx

instruments for innovation