



Moab End-User Training

CLI Module

Agenda

- Job Submission
- Job Dependencies
- Job Monitoring
- Job Management
- Job Scripting

Job Submission

- A Moab Job: A request for compute resources needed to perform computational work
- A Job Specifies:
 - What resources are needed
 - When the resources are needed
 - For how long are the resources needed
- The most common Moab command is `msub`

Job Submission

Common `msub` Options

- `-l` Resource List
 - `nodes`
 - `nodes=X:ppn=X`
 - `procs`
 - `walltime`

Job Submission

Common `msub` Options (Cont'd)

- `-q` Destination Queue (Class)
- `-A` Account
- `-N` NAME
- `-o` Output Path
- `-e` Error Path
- `-h` Job Hold

Job Submission

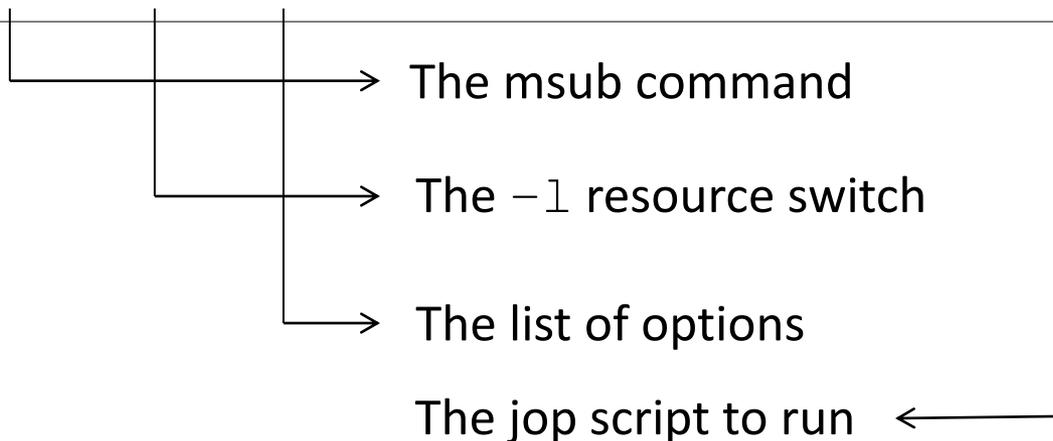
Common `msub` Options (Cont'd)

- `-m` Mail Options
- `-M` Mail List
- `-E` Environment Variables
- `-V` All Variables
- `-v` Variable List

msub -l

- Specifies resource requirements for your job
- Establishes limits to resources
- `-l` is followed by one or more option
- Options depended upon resource manager

```
$ msub -l nodes=32,walltime=3600 cmd.sh
```



nodes

- Identifies the number and type of nodes
- The node and properties of the nodes are separated by a colon (:)
- Request 12 nodes of any type:

```
$ msub -l nodes=12 cmd.sh
```

- Request 4 processors on one node:

```
$ msub -l nodes=1:ppn=4 cmd.sh
```

nodes=X : ppn=X

- Moab uses the concept of tasks to schedule workload
- Today's multiple-processor compute nodes can often support more than one task simultaneously
- Submit a job requesting 4 nodes with 2 processors per node:

```
$ msub -l nodes=4:ppn=2 cmd.sh
```

Moab interprets this as “On whatever nodes the job lands on, there needs to be 2 processors there to handle the tasks”

nodes=X : ppn=X

- Consider a cluster of 8 quad-core physical computers:

```
$ echo sleep 300 | msub -l nodes=4:ppn=2,walltime=200
Moab.1
$
```

```
$ mdiag -n
compute node summary
Name      State      Procs      Memory      Opsys
node01    Idle       4:4        0:0         -
node02    Idle       4:4        0:0         -
node03    Idle       4:4        0:0         -
node04    Idle       4:4        0:0         -
node05    Idle       4:4        0:0         -
node06    Idle       4:4        0:0         -
node07    Busy       0:4        0:0         -
node08    Busy       0:4        0:0         -
```

nodes=X : ppn=X

- To force Moab to distribute the job across all 4 nodes, use `nmatchpolicy=exactnode`:

```
$ msub -l nodes=4:ppn=2,nmatchpolicy=exactnode cmd.sh
```

nodes=X : ppn=X

- Can be seen as this:

```
$ mdiag -n
compute node summary
Name      State      Procs      Memory      Opsys
node01    Idle       4:4        0:0         -
node02    Idle       4:4        0:0         -
node03    Idle       4:4        0:0         -
node04    Idle       4:4        0:0         -
node05    Busy       2:4        0:0         -
node06    Busy       2:4        0:0         -
node07    Busy       2:4        0:0         -
node08    Busy       2:4        0:0         -
```

nodes=X : ppn=X

- The bottom line: If you don't care how many nodes are used for your job, use `nodes=8` and let Moab distribute accordingly

procs

- The number of total processors to be allocated to a job
- Can come from one or more nodes (Depending on system configuration)
- Use only 1 `procs` declaration per `msub` job submission

```
$ msub -l procs=8 cmd.sh
```

walltime

- The wall-clock time defines the maximum amount of time your job will run on the cluster
- Moab will force the running code to terminate at the end of the walltime setting
- The value for walltime is DD:HH:MM:SS

```
$ msub -l procs=8,walltime=5:00 cmd.sh
```

- Since Moab is a scheduler, time is crucial

```
$ echo sleep 200 | msub -l host=node00,walltime=300
```

walltime

- Using checkjob to see how much time is left on a job's walltime:

```
$ checkjob Moab.3
job Moab.3
State: Running
...
Required HostList:
[node00:1]
...
Reservation 'Moab.3' (-00:00:05 -> 00:04:55)
Duration: 00:05:00)
```

walltime

- Assuming there is space available, you can adjust walltime using the “Request Adjust Walltime Duration” (reqawduration) switch to `mjobctl`:

```
$ mjobctl -m reqawduration-=10:00 <JOBID>
```

- Time is converted to seconds, regardless of how it is input
- Adding walltime must be configured by Admin
- Can only adjust your own jobs

-q Destination Queue/Class

- Queues allow the system administrator to define resource allocation for jobs
- E.g., can limit the number and/or types of CPUs users can submit jobs to
- Queues are defined in moab.cfg:

```
CLASSCFG [lowprocs]          MAX.PROC=2
```

- When submitting jobs to this queue, only 2 processors will be made available:

```
$ msub -q lowprocs cmd.sh
```

-q Destination Queue/Class

- Submit a job requesting more than 2 procs

```
$ echo sleep 300 | msub -q lowprocs -l procs=4
```

```
$ showq
active jobs-----
JOBID USERNAME STATE PROCS REMAINING STARTTIME
0 active jobs 0 of 100 processors in use by local jobs
(0.0%)
0 of 25 nodes active (0.00%)
```

```
eligible jobs-----
JOBID USERNAME STATE PROCS WCLIMIT QUEUETIME
0 eligible jobs
```

```
blocked jobs-----
JOBID USERNAME STATE PROCS WCLIMIT QUEUETIME
Moab.4 root BatchHold 4 99:23:59:59 Fri Jul 22
```

-A Account

- The `-A` switch allows jobs to be submitted to a named account
- Accounts can be used to further define limits on resources
- Accounts are defined in `moab.cfg`:

```
ACCOUNTCFG[projectX]    MAXJOB=2
```

Limits the max number of jobs to 2 for projectX

```
$ msub -A projectX cmd.sh
```

-A Account

- Submit more than allowed:

```
$ echo sleep 300 | msub -A projectX
```

- View the results:

```
$ showq
active jobs-----
JOBID USERNAME STATE PROCS REMAINING STARTTIME
Moab.5 user1 Running 1 99:23:59:56 Fri Jul 22
Moab.6 user1 Running 1 99:23:59:57 Fri Jul 22
2 active jobs 2 of 100 processors in use by local jobs (2.0%)
1 of 25 nodes active (4.00%)
eligible jobs-----
JOBID USERNAME STATE PROCS WCLIMIT QUEUETIME
0 eligible jobs
blocked jobs-----
JOBID USERNAME STATE PROCS WCLIMIT QUEUETIME
Moab.7 user1 Idle 1 99:23:59:56 Fri Jul 22
Moab.8 user1 Idle 1 99:23:59:56 Fri Jul 22
```

-N NAME

- The -N switch allows you to give a name to a job when submitting

```
$ msub -N myjob cmd.sh
```

- Can use the name with commands such as checkjob:

```
$ checkjob myjob
job Moab.11
AName: myjob
State: Running
...
```

-o Output Path

- Defines the path and name to be used for the standard output stream of a batch job
- The named file ends up on the compute node's file system
- The output file is relative to the where the job script is being run on the compute node

-o Output Path

- Example pingtest.sh:

```
#!/bin/bash
#PBS -l nodes=TFE1,walltime=300 -o TFE1/stdout.txt
ping -c 3 localhost
```

- Submit the job:

```
$ msub pingtest.sh
```

- View the output:

```
$ cat /home/user1/TFE1/stdout.txt
PING TFE1 (127.0.0.1) 56(84) bytes of data.
64 bytes from TFE1 (127.0.0.1): icmp_seq=1 ttl=64 time=1ms
64 bytes from TFE1 (127.0.0.1): icmp_seq=2 ttl=64 time=1ms
64 bytes from TFE1 (127.0.0.1): icmp_seq=3 ttl=64 time=1ms
--- mgmtnode ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1s
```

-e Error Path

- Similar to the output path, the -e can define the standard error stream of a batch job
- The named file ends up on the compute node's file system
- The output file is relative to the where the job script is being run on the compute node

-m Mail Options

- The -m switch sends email to a predetermined user upon certain conditions:
 - abort
 - begin
 - end
- To send notification of all events:

```
$ msub -m abe cmd.sh
```

-M Mail List

- This switch overrides the default mailto setting of for email notifications

```
$ msub -M fred@r1i0n0,barney@r1i0n1,wilma@r1i0n2
```

-E Environment Variables

- The `-E msub` switch only works with SLURM and TORQUE/PBS resources managers
- When jobs run on compute resources, they behave according to the defined environmental shell of that compute node
- Environment variables can be sent with job submission:

MOAB_ACCOUNT

MOAB_DEPEND

MOAB_JOBNAME

MOAB_NODELIST

MOAB_QOS

MOAB_BATCH

MOAB_GROUP

MOAB_MACHINE

MOAB_PARTITION

MOAB_TASKMAP

MAOB_CLASS

MOAB_JOBID

MOAB_NODECOUNT

MOAB_PROCCOUNT

MOAB_USER

-E Environment Variables

- Edit pingtest.sh:

```
#!/bin/bash
#PBS -l nodes=TFE1,walltime=300 -E
echo $MOAB_USER
ping -c 3 localhost
```

- Submit the job:

```
$ msub pingtest.sh
```

- View the output:

```
$ cat /home/user1/STDIN.o1
user1
PING TFE1 (127.0.0.1) 56(84) bytes of data.
64 bytes from TFE1 (127.0.0.1): icmp_seq=1 ttl=64 time=1ms
64 bytes from TFE1 (127.0.0.1): icmp_seq=2 ttl=64 time=1ms
64 bytes from TFE1 (127.0.0.1): icmp_seq=3 ttl=64 time=1ms
--- mgmtnode ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1s
```

-V All Variables

- The -V switch declares that all environment variables in the `msub` environment be exported to the batch job
- Once declared, they can be used with the `env` command

-v Variable List

- The `-v` switch allows the user to define one or more variables to be exported into the job environment
- Does not export all environment variables like the `-V` switch

-h Hold

- The `-h` switch allows the user to submit a job to Moab and immediately place the job on hold
- Allows users to “Stack” jobs in the job queue until their data are ready to submit

```
$ msub -h -l walltime=30:00 cmd1.sh
Moab.1
$ msub -h -l walltime=30:00 cmd2.sh
Moab.2
$ msub -h -l walltime=30:00 cmd3.sh
Moab.3
```

-h Hold

- Show the hold:

```
$ showq
...
blocked jobs-----
JOBID      USERNAME      STATE      PROCS      WCLIMIT      QUEUETIME
Moab.1     student       UserHold   1 00:30:00 Mon Dec 12 12:24:00
Moab.2     student       UserHold   1 00:30:00 Mon Dec 12 12:24:30
Moab.3     student       UserHold   1 00:30:00 Mon Dec 12 12:24:45
```

- Un-hold (release) by using mjobctl -u:

```
$ mjobctl -u Moab.1
$ mjobctl -u Moab.2
$ mjobctl -u Moab.3
```

Job Dependencies

- A job's completion or failure can be used to step through job workflow
- Job dependencies are active by default
- Dependent jobs are only supported through a resource manager
- Syntax is specific to the resource manager

Job Dependencies

■ Job Submission Example:

```
$ msub myjob1.sh  
Moab.1
```

```
$ msub -W x=depend:afterok:Moab.1 myjob2.sh  
Moab.2
```

```
$ checkjob Moab.2
```

```
...
```

```
NOTE: job cannot run (dependency Moab.1  
jobsuccessfulcomplete not met)
```

Job Dependency Syntax

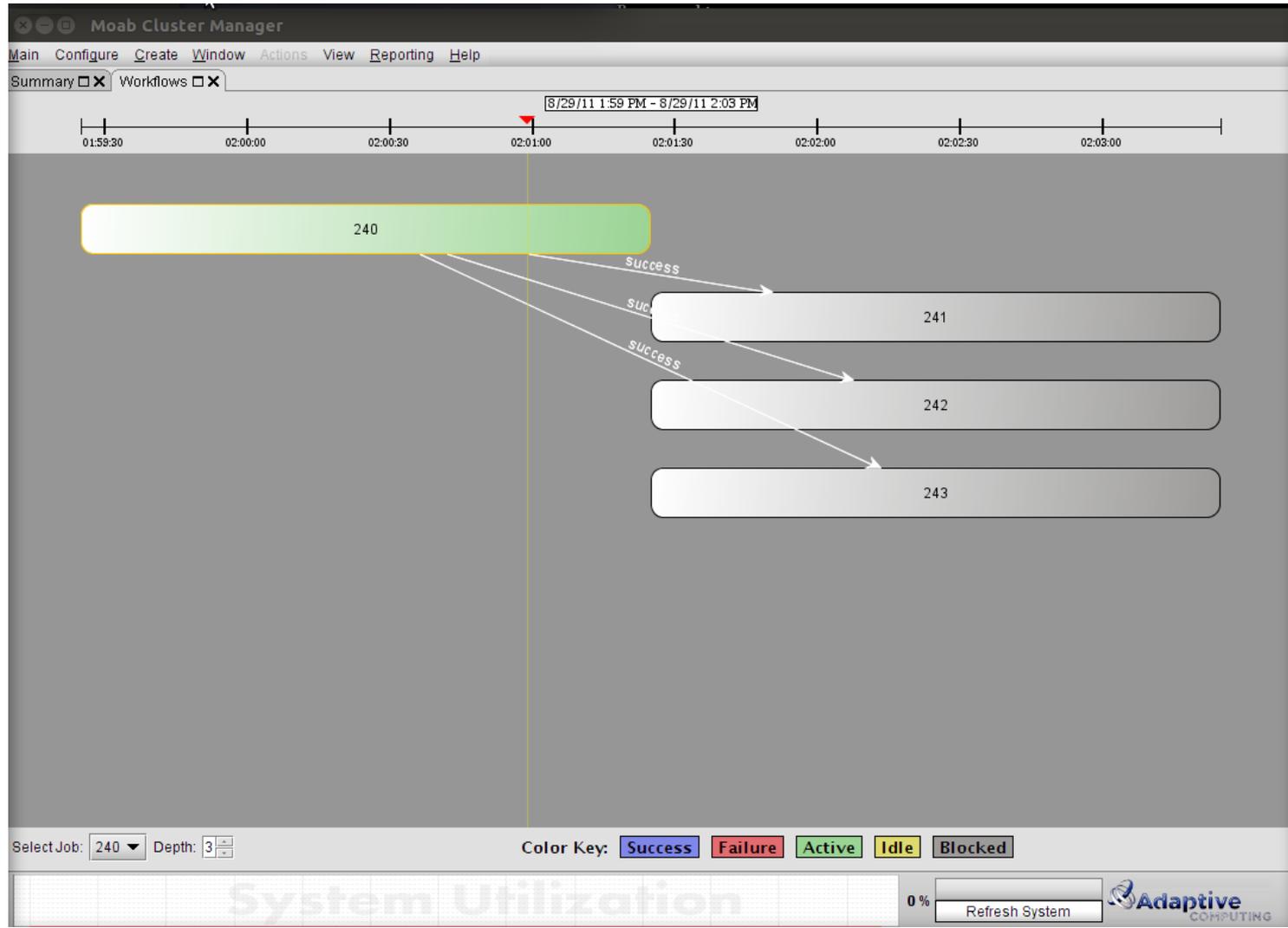
Dependency	Format	Description
after	after:<job>[:<job>]...	Job may start at any time after specified jobs have started execution.
afterany	afterany:<job>[:<job>]...	Job may start at any time after all specified jobs have completed regardless of completion status.
afterok	afterok:<job>[:<job>]...	Job may be start at any time after all specified jobs have successfully completed.
afternotok	afternotok:<job>[:<job>]...	Job may start at any time after all specified jobs have completed unsuccessfully.

Job Dependency Syntax

Dependency	Format	Description
before	before:<job>[:<job>]...	Job may start at any time before specified jobs have started execution.
beforeany	beforeany:<job>[:<job>]...	Job may start at any time before all specified jobs have completed regardless of completion status.
beforeok	beforeok:<job>[:<job>]...	Job may start at any time before all specified jobs have successfully completed.
beforenotok	beforenotok:<job>[:<job>]...	Job may start at any time before any specified jobs have completed unsuccessfully.
on	on:<count>	Job may start after <count> dependencies on other jobs have been satisfied.

Job Dependencies

- Moab Cluster Manager View:



Job Monitoring

- checkjob
- showhist
- shoq
- showbf
- showstart

checkjob

- Displays detailed job state information
- Used to view diagnostic output for a specific job
- Individual users can run `checkjob` on their own jobs
- Using the `-v` switch get verbose output
- Using the `-v -v` switch get very verbose output

showhist

- `showhist` is an executable perl script that queries the history of jobs submitted in the past
- Run by itself, will show all jobs submitted
- Run against a specific job ID, shows information about that specific job

showq

- Shows information about active, eligible, blocked, and/or recently completed jobs
- Shows the actual job ordering of the Moab scheduler
- Can be used to see if the scheduler is running, stopped, or paused
- Can be run by any user
- `--loglevel=0-9` shows more details

showq

- Sample output:

```
$ showq
active jobs-----
JOBID          USERNAME      STATE  PROCS   REMAINING      STARTTIME
Moab.5         student      Running    1    99:23:59:56    Fri Jul 22
Moab.6         student      Running    1    99:23:59:56    Fri Jul 22
2 active jobs 2 of 100 processors in use by local jobs (2.0%)
    1 of 25 nodes active (4.00%)
```

showbf

- Shows how many processors are immediately available for use on the cluster
- Can be run by any user

```
$ showbf
Partition      Tasks      Nodes      Duration StartOffset      StartDate
-----
ALL            20         5           INFINITY 00:00:00          07:00:48_07/27
torque         4          1           INFINITY 00:00:00          07:00:48_07/27
nativerm      16         4           INFINITY 00:00:00          07:00:48_07/27
```

showstart

- Displays the estimated start time of a job based on a number of analysis types:
 - Historical usage
 - Earliest available reserveable resources
 - Priority based backlog analysis
- If a job is running, `showstart` displays the time the job started
- If a job has a reservation, `showstart` displays the start time of the reservation
- Can be run by any user

showstart

■ Example output:

```
$ echo sleep 300 | msub -l nodes=16,walltime=300  
Moab.10
```

```
$ showstart -e all Moab.10  
job Moab.10 requires 16 procs for 00:05:00
```

```
Estimated Rsv based start in 00:04:09 on Wed Jul 27 07:25  
Estimated Rsv based completion in 00:04:09 on Wed Jul 27:07:30  
Estimated Priority based start in 00:04:09 on Wed Jul 27 07:25  
Estimated Priority based completion in 00:04:09 on Wed Jul 27:07:30  
Estimated Historical based start in 00:04:09 on Wed Jul 27 07:25  
Estimated Historical based completion in 00:04:09 on Wed Jul 27:07:30
```

```
Best Partition: nativerm
```

Job Management

- canceljob
- mjobctl

canceljob

- Once a job has been submitted to the queue, it can be canceled any time by `canceljob`
- Users can only cancel their own jobs
- Moab Administrators can cancel any jobs
- Can cancel based on jobID or job name

```
$ echo sleep 300 | msub -l nodes=16,walltime=300
Moab.11
$ canceljob Moab.11
job 'Moab.11' cancelled
```

mjobctl

- Controls various aspects of job
- Can be used to:
 - submit
 - cancel
 - execute
 - checkpoint

Job Scripting

- A job script allows you to send a sequence of commands all at once
- A job script is a simple text file, and can contain:
 - Shell script environment (hashpling)
 - Environment variables
 - Comments
 - Job executables
- `msub` statements begin with `#PBS`

Job Scripting

```
#!/bin/bash
# This script is called test.sh
#### Moab environment
#PBS -l nodes=TFE1
#PBS -l walltime=300
#PBS -N pingjob

#### Shell commands
ping -c 3 localhost
cd /home/user1
touch output
date > output
echo "The JOBID of this job is $PBS_JOBID" >> output
echo "Job submitted by $PBS_O_LOGNAME" >> output
echo "The exit status of this job is $? >> output
echo "Done" >> output
```

```
$ musb test.sh
```

Demo

Questions



Adaptive

COMPUTING