Moab

How do NODEAVAILABILITYPOLICY and NODEALLOCATIONPOLICY work?

Here is a tutorial showing how some common settings of the following Moab parameters work:

JOBNODEMATCHPOLICY EXACTNODE

JOBNODEMATCHPOLICY AUTO

NODEAVAILABILITYPOLICY DEDICATED:PROC

NODEAVAILABILITYPOLICY COMBINED:PROC

NODEALLOCATIONPOLICY CPULOAD

For all the examples here, imagine that we have 2 nodes (node00, node01) each with 4 procs.For these examples, CGROUPS was enabled and Moab 9.1.1 and Torque 6.1.1.1 were used.

Effect of JOBNODEMATCHPOLICY.

Let’s start with these settings:

JOBNODEMATCHPOLICY EXACTNODE // This requires the # of nodes used to be EXACT.

NODEAVAILABILITYPOLICY DEDICATED:PROC

NODEALLOCATIONPOLICY LASTAVAILABLE

Assume our system is idle and we execute the following command:

echo sleep 100|msub -l nodes=2:ppn=2,walltime=200

“mdiag -n” shows:

```
----------------------------------------------------

Name State Procs Memory Opsys

node00 Running 2:4 32202:32202 linux
```

```
node01 Running 2:4 32202:32202 linux

----- --- 4:8 64404:64404 -----

Total Nodes: 2 (Active: 2 Idle: 0 Down: 0)

---------------------------------------------------
```

Here Moab chose two nodes as requested.

If we change JOBNODEMATCHPOLICY to AUTO (or let it default) then run the same

experiment on our idle system, “mdiag -n” shows:

```
---------------------------------------------------

Name State Procs Memory Opsys

node00 Idle 4:4 32202:32202 linux

node01 Busy 0:4 32202:32202 linux

----- --- 4:8 64404:64404 -----

Total Nodes: 2 (Active: 1 Idle: 1 Down: 0)
```

Here Moab packs all 4 procs on the same node. Which node is picked depends on NODEALLOCATIONPOLICY. Which is set to LASTAVAILABLE.

Effects of NODEAVAILABILITYPOLICY

NODEAVAILABILITYPOLICY helps Moab decide if a node is BUSY or not.

New jobs cannot be scheduled on a node which is BUSY.

Let’s start with these settings:

```
JOBNODEMATCHPOLICY AUTO

NODEAVAILABILITYPOLICY DEDICATED:PROC

NODEALLOCATIONPOLICY LASTAVAILABLE
```

NODEAVAILABILITYPOLICY DEDICATED:PROC tells Moab

to calculate the number of available procs on a node as follows:

AvailableProcs = Configured – Dedicated.

Assuming Moab has not scheduled any jobs, running “mdiag -n” shows:

```
---------------------------------------------------------------------

Name State Procs Memory Opsys

node00 Idle 4:4 32202:32202 linux

node01 Idle 4:4 32202:32202 linux

----- --- 8:8 64404:64404 -----

Total Nodes: 2 (Active: 0 Idle: 2 Down: 0)

–––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––
```

We see here that there are 8 idle procs out of 8 configured. Since we are using DEDICATED:PROC it does not matter what the cpu loading is on node00 and node01. If fact, one could log into node00 and node01 and manually start processes that utilize 100% of each proc and Moab would still report 8 idle procs. Availability of procs is not affected by cpu utilization with NODEAVAILABILITYPOLICY DEDICATED:PROC.

Again, assume our system is idle. Let’s start a job which which needs 1 node and 1 proc by executing “echo sleep 100|msub -l nodes=1:ppn=1,walltime=200”.

Now “mdiag -n” shows:

```
---------------------------------------------------------------------

Name State Procs Memory Opsys

node00 Idle 4:4 32202:32202 linux

node01 Running 3:4 32202:32202 linux

----- --- --- 7:8 64404:64404 -----

 Total Nodes: 2 (Active: 1 Idle: 1 Down: 0)
```

node01 was choosen because the default NODEALLOCATIONPOLICY is LASTAVAILABLE. If the policy had been FIRSTAVAILABLE, node00 would have been chosen. Now imagine that the job we just started used 100% of all 4 procs on node01. Moab would still report 3 idle procs on node01.

If we rapidly execute the same command 7 more times, “mdiag -n” shows:

```
 -----------------------------------------------------------

Name State Procs Memory Opsys

node00 Busy 0:4 32202:32202 linux

node01 Busy 0:4 32202:32202 linux

----- --- 0:8 64404:64404 -----

 Total Nodes: 2 (Active: 2 Idle: 0 Down: 0)
```

All 8 jobs are running and no procs are available. If we execute the same command once more, the job will be eligible, but not running because there are no procs available. Because these jobs are all doing nothing but a “sleep”, they are hardly loading the procs. On the system which I am using to test these commands the system is 98% idle. But, since we are using DEDICATED:PROC we cannot take advantage of those available cpu cycles. This would not be good in a timeshare system; however, in many HPC environments, guaranteeing exclusive access to a resource is often more important than sharing a resource.

Let’s start over with an idle system and let’s change NODEAVAILABILITYPOLICY to COMBINED:

```
 NODEAVAILABILITYPOLICY COMBINED:PROC
```

Now Moab will calculate available procs on a node based on Utilized and Dedicated:

AvailableProcs = Configured – max(Dedicated, Utilized)

See the Moab documentation for a definition of Utilized and how it relates to “Load Average”. Utilized is an approximation for sum of the the proc utilizations on a node. For example if the individual proc utilizations on a node with 4 procs were (0, 75%, 50%, 25%), then the equivalent in “utilized procs” would be (0 + 75 + 50 + 25)/100 = 1.5procs. Ignoring I/O, runQ, and many other issues, this corresponds roughly to “Load Average” on unix systems. “Load Average” is shown by the “top” command, but its definition is more complicated than presented here and varies between different flavors of unix.

Using NODEAVAILABILITYPOLICY COMBINED:PROC, here is what “mdiag -n” shows on my system when it is “idle”:

```
-----------------------------------------------------

Name State Procs Memory Opsys

node00 Idle 3:4 32202:32202 linux

node01 Idle 3:4 32202:32202 linux

----- --- 6:8 64404:64404 -----

Total Nodes: 2 (Active: 0 Idle: 2 Down: 0)
```

We have “lost” two procs! There are so many background daemons and other activities running on my system that Moab shows that here are only 6 procs available. On this test system, moab, torque and the compute nodes are all on a desktop computer with 8 cores, so there is a lot of background activity.

But, this exercise shows CLEARLY that with NODEAVAILABILITYPOLICY COMBINED:PROC, Utilized can be greater than 0 without even running one job with Moab.

On our “idle” system, let’s execute the same command:

```
echo sleep 100|msub -l nodes=1:ppn=1,walltime=200
```

Here is what “mdiag -n” shows:

```
----------------------------------------------------------

Name State Procs Memory Opsys

node00 Idle 3:4 32202:32202 linux

node01 Running 3:4 32202:32202 linux

----- --- 6:8 64404:64404 -----

Total Nodes: 2 (Active: 1 Idle: 1 Down: 0)

-----------------------------------------------------------
```

Hmmm...we still have 6 procs available. Actually right after executing the job, “mdiag -n” only showed 5 available, but after about 30 seconds it showed 6. It is very easy to understand DEDICATED resources, but since COMBINED is based also on Load Average the situation is more dynamic and depends on OS features. If fact, imagine that this job had launched several cpu intensive processes that fully utilized 3 procs. If NODEAVAILABILITYPOLICY were DEDICATED:PROC “mdiag -n” would show 7 idle procs whereas with COMBINED:PROC, idle procs would be about 4 (depending on Load Average),

If we quickly execute the same sleep job 7 more times “mdiag -n” shows:

Name State Procs Memory Opsys

node00 Busy 0:4 32202:32202 linux

node01 Busy 0:4 32202:32202 linux

----- --- 0:8 64404:64404 -----

Total Nodes: 2 (Active: 2 Idle: 0 Down: 0)

All 8 jobs run as expected; however, notice that both nodes are marked “Busy”.And if we execute the command one more time, the job is eligible but does not run because there are no more procs available, eventhough the system is 90% idle.

Moab will not allow more than 1 job to be run on a proc, no matter what the Load Average is.

What about NODEAVAILABILITYPOLICY UTILIZED:PROC? Currently UTILIZED:PROC and COMBINED:PROC function the same. However, in the future UTILIZED:PROC may be changed such that

AvailableProcs = Configured – Utilized

That would allow more then 1 job per proc. It is recommended that UTILIZED:PROC not be used since its definition may change in the future.

Effects of NODEALLOCATIONPOLICY

Let’s start with these settings:

JOBNODEMATCHPOLICY EXACTNODE

NODEAVAILABILITYPOLICY COMBINED:PROC

NODEALLOCATIONPOLICY CPULOAD

While NODEAVAILABILITYPOLICY helps Moab decide which nodes are BUSY,

NODEALLOCATIONPOLICY helps Moab decide on which nodes a job will run.

Let’s examine what happens when we select COMBINED:PROC and CPULOAD (see above). CPULOAD will cause Moab to select nodes that have the lowest cpu utilization (Load Average).

Starting with an idle system, “mdiag -n” shows:

```
---------------------------------------------------

Name State Procs Memory Opsys

node00 Idle 3:4 32202:32202 linux

node01 Idle 4:4 32202:32202 linux

----- --- 7:8 64404:64404 -----

Total Nodes: 2 (Active: 0 Idle: 2 Down: 0)
```

Just as in the previous experiment, COMBINED:PROC shows idle procs as less than the number configured. In fact as I watch the screen it varies between 6 and 7 depending on daemons, my typing, etc.

Let’s execute this command repeatedly.

```
echo sleep 100|msub -l nodes=1:ppn=1,walltime=200
```

The result will be very similar to the previous experiment except the nodes will be kept balanced with respect to cpu utilization. For example, job1 → node00, job2 → node01, job3 → node00, …

Unique solution ID: #1202
Author: William Groves
Last update: 2018-03-05 21:59