Torque

Why will my job not start when there is no other job on the compute resource but CPU usage on that node is high?

Issue: Why will my job not start when there is no other job on the compute resource but CPU usage on that node is high?

Affected Versions: All

Symptom: Some sites have workload that forks off processes during the lifecycle of the batch job. Although TORQUE does a great job following most of these processes, it is possible for an application to do a double fork, and for pbs_mom to lose track of that process. If the application does not clean up these processes, then it is up to the administrator to either go into the compute node and kill those rogue processes, or for the administrator to implement an automated solution.

Common log entries surrounding this case are:

Message[8] 3 nodes unavailable to start reserved job after 1 seconds
(job 621346 has exceeded wallclock limit on node compute001 - check job)
Message[9] 2 nodes unavailable to start reserved job after 2 seconds
(job 633415 has exceeded wallclock limit on node compute001 - check job)
Message[10] 2 nodes unavailable to start reserved job after 1 seconds
(reserved node compute001 is in state 'Running' - check node)

nodes unavailable to start reserved job after 1 seconds (reserved node node1 is in state 'Idle' - check node)

WARNING: node 'compute001' has excessive load (state: 'Busy' load: 27.160)
WARNING: node 'compute001' has more processors utilized than dedicated (6 > 4)

WARNING: node 'compute001' has more processors utilized than dedicated (8 > 4)

Additionaly, you can look at the output of checknode, pbsnodes or the loadavg on the compute node to see how much of a compute resource is available.

root@server:/proc$ cat loadavg
0.35 0.35 0.33 1/286 10444

/proc/loadavg
The first three fields in this file are load average figures, giving the number of jobs in the run queue (state R) or waiting for disk I/O (state D), averaged over 1, 5, and 15 minutes. They are the same as the load average numbers given by uptime(1) and other programs. The fourth field consists of two numbers separated by a slash (/). The first of these is the number of currently executing kernel scheduling entities

(c) 2024 Shawn Hoopes <shoopes@adaptivecomputing.com> | 2024-05-04 20:58
URL: /phpmyfaq/index.php?action=artikel&cat=2&id=30&artlang=en

(processes, threads); this will be less than or equal to the number of CPUs. The value after the slash is the number of kernel scheduling entities that currently exist on the system. The fifth field is the PID of the process that was most recently created on the system.

Note: Until the rogue process is removed, Moab will not permit a job to start due to insufficient resources available.

Solution:

First:

There are a few solutions. Some sites run extra monitoring applications that run in tandem with their batch compute jobs. In these cases, the extra load is expected. If a site is sure that the system is not being used by rogue processes, you can set the following in your moab.cfg

NODEAVAILABILITYPOLICY DEDICATED:PROCS

http://docs.adaptivecomputing.com/8-1-1/enterprise/MWM/help.htm#topics/moabWorkloadManager/topics/appendices/a.aparameters.html#nodeavailabilitypolicy

*Moab will ignore resource utilization information in locating available processors for jobs

Secondly:

You can use the Node Health Check provided by LBNL.

Alternatively, you can use a more agressive option, such as Reaver provided by Troy Baer.

Unique solution ID: #1029
Author: Jason Booth
Last update: 2015-12-23 04:13

(c) 2024 Shawn Hoopes <shoopes@adaptivecomputing.com> | 2024-05-04 20:58
URL: /phpmyfaq/index.php?action=artikel&cat=2&id=30&artlang=en