Scheduling Jobs on Clusters

21 March 2007

MichiganEngineering

What is a Cluster?

A cluster is a logical set of computers under the management of a single job scheduler and on a local network.

The CAC (CAEN HPC Group) operates two clusters: `nyx` and `aon`.

- `nyx` is 400 nodes with 976 CPUs
- `aon` is 224 nodes with 448 CPUs.

We run two schedulers, one for each cluster.

What is a Job?

A job is a computer program and a description of its resource requirements.

Examples of resource requirements are:

- number of CPUs required
- amount of time required
- type of network required
- amount of memory required

What is Scheduling?

Scheduling is looking at a list of jobs (the queue) and deciding when and where each job can run.

This is meaningful only if there are more demands than resources, otherwise everything starts immediately (the simplest scheduling case).

Why do we bother?

Why do we make these hard decisions? Why not just let the first job submitted run first, the second job run second, etc.?

- Not everyone has access to all of the available resources
  - Node owners, node renters, the general public
- We don't want a few people to use all of the resources
  - Scheduling lets us limit the total number of resources per person (generally we only look at the number of nodes in use)
  - We limit the number of jobs eligible for scheduling
- We greatly affect priority by considering:
  - The amount of time a job has been in the queue (FIFO)
  - The quantity of resources recently consumed (Fairshare)
  - External factors (we weight people, groups, and queues)

Why do we bother?

Why do we make these hard decisions? Why not just let the first job submitted run first, the second job run second, etc.?

- Not everyone has access to all of the available resources
  - Node owners, node renters, the general public
- We don't want a few people to use all of the resources
  - Scheduling lets us limit the total number of resources per person (generally we only look at the number of nodes in use)
  - We limit the number of jobs eligible for scheduling
- We greatly affect priority by considering:
  - The amount of time a job has been in the queue (FIFO)
  - The quantity of resources recently consumed (Fairshare)
  - External factors (we weight people, groups, and queues)

Why do we bother?

Why do we make these hard decisions? Why not just let the first job submitted run first, the second job run second, etc.?

- Not everyone has access to all of the available resources
  - Node owners, node renters, the general public
- We don't want a few people to use all of the resources
  - Scheduling lets us limit the total number of resources per person (generally we only look at the number of nodes in use)
  - We limit the number of jobs eligible for scheduling
- We greatly affect priority by considering:
  - The amount of time a job has been in the queue (FIFO)
  - The quantity of resources recently consumed (Fairshare)
  - External factors (we weight people, groups, and queues)

Why do we bother?

Why do we make these hard decisions? Why not just let the first job submitted run first, the second job run second, etc.?

- Not everyone has access to all of the available resources
  - Node owners, node renters, the general public
- We don't want a few people to use all of the resources
  - Scheduling lets us limit the total number of resources per person (generally we only look at the number of nodes in use)
  - We limit the number of jobs eligible for scheduling
- We greatly affect priority by considering:
  - The amount of time a job has been in the queue (FIFO)
  - The quantity of resources recently consumed (Fairshare)
  - External factors (we weight people, groups, and queues)

Job States

These factors put each job into one of three states.

Running Jobs

- jobs currently running on some CPU(s)

Eligible Jobs

- jobs that could run if there were resources available and will probably run when one of the running jobs finishes

Ineligible jobs

- jobs owned by people/groups who have exceeded their maximum allowed resource consumption
- jobs owned by people/groups who have exceeded their maximum allowed number of eligible jobs

Job States

These factors put each job into one of three states.

Running Jobs

- jobs currently running on some CPU(s)

Eligible Jobs

- jobs that could run if there were resources available and will probably run when one of the running jobs finishes

Ineligible jobs

- jobs owned by people/groups who have exceeded their maximum allowed resource consumption
- jobs owned by people/groups who have exceeded their maximum allowed number of eligible jobs

Job States

These factors put each job into one of three states.

Running Jobs

- jobs currently running on some CPU(s)

Eligible Jobs

- jobs that could run if there were resources available and will probably run when one of the running jobs finishes

Ineligible jobs

- jobs owned by people/groups who have exceeded their maximum allowed resource consumption
- jobs owned by people/groups who have exceeded their maximum allowed number of eligible jobs

Job States

These factors put each job into one of three states.

Running Jobs

- jobs currently running on some CPU(s)

Eligible Jobs

- jobs that could run if there were resources available and will probably run when one of the running jobs finishes

Ineligible jobs

- jobs owned by people/groups who have exceeded their maximum allowed resource consumption
- jobs owned by people/groups who have exceeded their maximum allowed number of eligible jobs

An Example: The Cluster

Our example cluster:

- 8 nodes
- 2 of them are owned by the Rolling Stones
- 2 of them are owned by the Beatles
- 4 of them are available for anyone to use

Things we're ignoring in this example:

- Fairshare
- Other weights
- Maximum jobs running
- Maximum jobs eligible

An Example: The Cluster

Our example cluster:

- 8 nodes
- 2 of them are owned by the Rolling Stones
- 2 of them are owned by the Beatles
- 4 of them are available for anyone to use

Things we're ignoring in this example:

- Fairshare
- Other weights
- Maximum jobs running
- Maximum jobs eligible

An Example: The Queue

ID	User	Queue	CPUs	Req.Time	Queue Time
1006	mick	stones	2	2:00:00	15:10
1007	keith	stones	1	2:00:00	15:20
1008	mick	stones	2	2:00:00	15:30
1009	keith	stones	1	2:00:00	15:40
1010	john	beatles	1	6:00:00	15:10
1011	ringo	beatles	2	2:00:00	15:20
1012	ringo	beatles	2	2:00:00	15:30
1013	paul	beatles	1	3:00:00	15:30
1014	pbuck	general	3	3:00:00	15:10
1015	fffreddy	general	2	1:00:00	15:10
1016	pdoherty	general	1	6:00:00	15:10
1017	bsumner	general	4	2:00:00	15:10
1018	bsumner	general	4	2:00:00	15:10



An Example: One Possible Outcome

stones	node1	1006	1006	1007	1007	1008	1008				
	node2	1006	1006	1009	1009	1008	1008				
beatles	node3	1010	1010	1010	1010	1010	1010	1011	1011	1012	1012
	node4	1013	1013	1013				1011	1011	1012	1012
general	node5	1014	1014	1014	1015			1017	1017	1018	1018
	node6	1014	1014	1014	1015			1017	1017	1018	1018
	node7	1014	1014	1014				1017	1017	1018	1018
	node8	1016	1016	1016	1016	1016	1016	1017	1017	1018	1018
		1	2	3	4	5	6	7	8	9	10

ID	User	Queue	nodes	Req.Time	Queue Time
1006	mick	stones	2	2:00:00	15:10
1007	keith	stones	1	2:00:00	15:20
1008	mick	stones	2	2:00:00	15:30
1009	keith	stones	1	2:00:00	15:40
1010	john	beatles	1	6:00:00	15:10
1011	ringo	beatles	2	2:00:00	15:20
1012	ringo	beatles	2	2:00:00	15:30
1013	paul	beatles	1	3:00:00	15:30
1014	pbuck	general	3	3:00:00	15:10
1015	fffreddy	general	2	1:00:00	15:10
1016	pdoherty	general	1	6:00:00	15:10
1017	bsumner	general	4	2:00:00	15:10
1018	bsumner	general	4	2:00:00	15:10

Real Life: aon

aon's scheduler is configured like this:

- People in the glotzer queue can use up to 190 CPUs
- People in the kieffer queue can use up to 155 CPUs
- People in the falk queue can use up to 86 CPUs
- People in the larson queue can use up to 37 CPUs
- Jobs in the short queue can use as many CPUs as available, but get scheduled at a low priority and cannot exceed 24 hours
- Fairshare and queue time order the queue
- Fairshare considers the last two weeks worth of jobs
- Jobs cannot run between aon's two locations (EECS and Chrysler)
- Nodes with high-speed networking (2Gbps) are the last to be used, unless requested

Real Life: aon

aon's scheduler is configured like this:

- People in the glotzer queue can use up to 190 CPUs
- People in the kieffer queue can use up to 155 CPUs
- People in the falk queue can use up to 86 CPUs
- People in the larson queue can use up to 37 CPUs
- Jobs in the short queue can use as many CPUs as available, but get scheduled at a low priority and cannot exceed 24 hours
- Fairshare and queue time order the queue
- Fairshare considers the last two weeks worth of jobs
- Jobs cannot run between aon's two locations (EECS and Chrysler)
- Nodes with high-speed networking (2Gbps) are the last to be used, unless requested

MichiganEngineering

Real Life: aon

aon's scheduler is configured like this:

- People in the glotzer queue can use up to 190 CPUs
- People in the kieffer queue can use up to 155 CPUs
- People in the falk queue can use up to 86 CPUs
- People in the larson queue can use up to 37 CPUs
- Jobs in the short queue can use as many CPUs as available, but get scheduled at a low priority and cannot exceed 24 hours
- Fairshare and queue time order the queue
- Fairshare considers the last two weeks worth of jobs
- Jobs cannot run between aon's two locations (EECS and Chrysler)
- Nodes with high-speed networking (2Gbps) are the last to be used, unless requested

Real Life: aon

`aon`'s scheduler is configured like this:

- People in the `glotzer` queue can use up to 190 CPUs
- People in the `kieffer` queue can use up to 155 CPUs
- People in the `falk` queue can use up to 86 CPUs
- People in the `larson` queue can use up to 37 CPUs
- Jobs in the `short` queue can use as many CPUs as available, but get scheduled at a low priority and cannot exceed 24 hours
- Fairshare and queue time order the queue
- Fairshare considers the last two weeks worth of jobs
- Jobs cannot run between `aon`'s two locations (EECS and Chrysler)
- Nodes with high-speed networking (2Gbps) are the last to be used, unless requested

Real Life: nyx

nyx's scheduler has:

- 13 queues
- 3 of the queues are split between two different networks
- 2 of the queues have "short" sub-queues
- 16 standing reservations (for purchased nodes)
- 3 user reservations (for rented nodes)
- a 10% weight for Fairshare in job priority calculations
- software license management for Stata, Comsol, Nastran, and parallel Nastran

Real Life: nyx

nyx's scheduler has:

- 13 queues
- 3 of the queues are split between two different networks
- 2 of the queues have "short" sub-queues
- 16 standing reservations (for purchased nodes)
- 3 user reservations (for rented nodes)
- a 10% weight for Fairshare in job priority calculations
- software license management for Stata, Comsol, Nastran, and parallel Nastran

Real Life: nyx

nyx's scheduler has:

- 13 queues
- 3 of the queues are split between two different networks
- 2 of the queues have "short" sub-queues
- 16 standing reservations (for purchased nodes)
- 3 user reservations (for rented nodes)
- a 10% weight for Fairshare in job priority calculations
- software license management for Stata, Comsol, Nastran, and parallel Nastran

Real Life: nyx

nyx's scheduler has:

- 13 queues
- 3 of the queues are split between two different networks
- 2 of the queues have "short" sub-queues
- 16 standing reservations (for purchased nodes)
- 3 user reservations (for rented nodes)
- a 10% weight for Fairshare in job priority calculations
- software license management for Stata, Comsol, Nastran, and parallel Nastran

Questions

- Questions
- Comments
- Suggestions



Questions

- Questions
- Comments
- Suggestions



Questions

- Questions
- Comments
- Suggestions

