Job Priority

Numerical Priority

Jobs have a numerical priority that is assigned to them when they are submitted, and which can be changed by the user or the wrangler later. The priority ranges from 1 to 9999, with 1 being the highest and 9999 the lowest.

Priority can be limited for non-privileged users

Administrators may not want to allow users to assign themselves the highest possible priority, reserving that privilege for render wranglers or other special users. The highest priority a non-administrator can assign to a job is controlled by the supervisor_highest_user_priority supervisor configuration parameter.

Clustering and Job Priority

Qube! makes use of a concept it calls "clusters". The clusters in a Qube! farm are laid out in a tree-like structure, much like a directory tree where files are kept on disk. The classic example of this is the show/sequence/shot hierarchy.

Clusters are designated in the same way as a directory tree; the root is written as '*/*', and clusters in the root are written as /A, /B, and /C/D (/C is in the root, and D is "above" /C). Clusters are organized like a directory tree, but instead of branching left to right, it's convenient to imagine them branching upwards, away from the root. In the same way that folders in a directory tree contain files and other folders, clusters can contain other clusters. Clusters can also contain Workers and jobs. This is the key to Qube!'s priority scheme.

- Workers belong to 1 and only 1 cluster.
- Jobs are submitted into a cluster; you can change a job's cluster after it's submitted, but it will always be in 1 and only 1 cluster at a time.

The cluster membership of both a Worker and a job is considered when determining the overall priority of a job to run on a particular Worker. The basic rules are:

- A Worker will run the lowest-priority job that is in its own cluster before it runs the highest-priority job from any other cluster.
- If there are no jobs in its own cluster waiting to run, a Worker will then run jobs from other cluster in decreasing priority.

So jobs from other clusters have lower priority than jobs from the same cluster. But there is even a precedence ordering for all the other clusters; jobs in clusters nearer a Worker's cluster are considered before jobs from clusters farther away. It "costs" a job priority to run in clusters other than its own, and the cost increases as the distance between the job's cluster and the Worker's cluster increases.

Climbing "down" the tree (toward the root) usually has no cost. However, a job submitted to /A/B/C has higher priority in /A/B/C, compared to /A/B, but that's just because the former case is in its own cluster. It's when a job starts climbing up the tree that it starts to lose priority.

- A job has the top cluster priority in its exact cluster path.
- It has 2nd cluster priority at all cluster nodes under its branch (towards the root).
- So, a job submitted to /A/B/C/D/E has the highest priority in /A/B/C/D/E.
- It has 2nd priority in /A/B/C/D, since a job specifically submitted to /A/B/C/D has top priority in there.

When a job cannot find any hosts by descending down its tree branch towards the root and has to start "climbing" the tree to find hosts, it loses priority. The more level it has to climb up, the lower its priority.

- If jobA is submitted to /A, jobB to /B, then they both will have the same priority in /C.
- If jobA is submitted to /, jobB to /B, then still both of them have the same priority in /C.
- If jobA is submitted to /A, jobB to /B, jobA will have higher priority than jobB in /A/C because jobA only has to climb up 1 level, whereas jobB has to climb up 2 levels (over to / then up to /A/C).

Another example that shows how jobs in different clusters are prioritized on the same Worker in /A/B/C:

Job Cluster	**Calculated cluster priority order for a Worker in /A/B/C**	**Cluster traversal count**
job cluster = /A/B/C	1	no traversal
job cluster = /A/B/C/D	2	1 traversal down
job cluster = /A/B/C/D/E	2	2 traversals down

job cluster = /A/B	3	1 traversal up
job cluster = /A/B/F	3	1 traversal down, 1 up
job cluster = /A	4	2 traversals up
job cluster = /	5	3 traversals up
job cluster = /G	5	1 down to /, 3 up into /A/B/C

All jobs are first sorted by the cluster priority order for the Worker, and then sorted by the job's numerical priority value within that priority order.

- the lowest priority job with cluster priority order 1 will have a higher effective priority than the highest priority job with cluster priority order 2
- the lowest priority job with cluster priority order 2 will have a higher effective priority than the highest priority job with cluster priority order 3
- and so on…

See Also

How to use clustering for workers

How clustering affects priority and worker selection for jobs

worker_cluster

supervisor_default_cluster

client_cluster

supervisor_highest_user_priority