Idle Job or Task

A common question of users of a scheduling system is *why is this job (or task) not running?* There are many reasons why a job may not run: network or hardware failure, misconfiguration of the system, or simply normal scheduling rules preventing the job's execution.

To understand how and when a job becomes active, it may be helpful to trace the lifetime of a job's task through the system.

The job must have been submitted.

First must you have pressed some in-app button or issued a command that submits a job. The job must first be processed and loaded by the engine.

To verify the job has been received by the engine, check that it is visible in the dashboard. If you cannot see the job:

- make sure no job filter or client side search box is preventing the job from being displayed,
- check that the dashboard is pointing to the same engine as the entity spooling the job, and
- check the engine log file that a "job Loaded" message had appeared when the job was spooled. Adjacent error messages in the log file could indicate that there was a syntax error in the job file.

You can also look at recently spooled jobs on the command line. e.g. "tq jobs mine --sort jid"

The job must be able to run.

Verify that the job is *not*:

- paused,
- delayed, nor
- waiting of other jobs to finish, as specified in its afterJids list.

The job must have ready tasks.

Verify that the job actually has ready tasks. When a job has been submitted, there will be at least one ready task that can run (unless the job author has been able to weave a job with circular dependencies!). Ready tasks will be leaf nodes on the job graph, though not necessarily all leaf nodes will be ready. A job's ready tasks can be listed on the command line with "tq tasks state=ready and jid=<jid>".

The remaining tasks at submission time will be blocked; a task will become ready once all of its child tasks have been successfully completed (or have been manually skipped). In the case of serial subtasks, a task may have to also wait for its "older" sibling tasks to finish before it can become ready.

Blades must be making requests for work.

Verify that blades are actually requesting work (and for the correct engine:port). Participating blades will be visible in the blades tab of the dashboard. If there are none, ensure that blades are contacting the correct engine and that the blade and engine hosts have the proper ports open in their firewall. The default port for the engine is 80 (TCP), and for blades is 9005 (TCP). (Note that 9005 (UDP) must be open on the blade and engine in order for progress messages to be reported to the engine; however, this will not affect whether or not a task starts on a blade.)

You can verify that a blade is requesting work by running a tracer in the dashboard. This will report which jobs have been considered for dispatching. If the job in question is displayed in the trace, the reason why it has not picked up will be displayed. If the job in question is not displayed, it may be because a task of a higher priority job was chosen, or because of other scheduling rules that would preclude the job from running on the blade.

Also consider that blades may not request work if they incurr a number of task failures within some interval. Such blades are marked with error accrual hiatus in the Note column of the blade list. The RecentErrorThrottle setting in the blade's profile manages the parameters of this hiatus calculation.

Scheduling rules must permit the job to be considered.

There are several scheduling rules that determine whether or not a task runs on a given blade:

- the requesting blade's service keys must match the task's service key expression,
- the requesting blade's metrics must satisfy the task's metric requirements,
- the task must not be restricted by limits. The limits in question could be represented by the limit tags of the job, task, or the task's current command, and
- the task must not be restricted by sharing limits. The limits in question would be those associated with the job's project.

Even if a task might satisfy those requirements, there may be a task of a higher priority job that also satisfies those requirements, and is thus picked up by the job.

Also note that every task assignment and completion affects limit counters, which will affect which tasks will be eligible for the next blade's request.