Scheduling Modes

Overview

Scheduling refers to the policies and algorithms used by Tractor to decide the order in which tasks are assigned to blades, and more generally to the enforcement of rules governing priorities, permissions, and capability requirements.|

Dispatching Tiers

Each job on the queue is associated with a dispatching tier. Tiers provide the highest level job dispatching control, jobs in high-valued tiers are always considered before those in low-valued tiers. For example, jobs in a "rush" tier can be handled before a "batch" tier. Within each tier jobs are sorted by the tier's Scheduling Mode (below).

New jobs are placed into the "default" tier when they are submitted, unless explicitly set at spool time (e.g. "tractor-spool --tier=batch ..."). Jobs can be moved to a different tier manually in the Dashboard. For example, during a crisis it may be useful to move several important jobs to a higher tier while leaving their job priorities unchanged to preserve sorting relative to each other. Each tier can also be **paused**individually in the Dashboard, so dispatching across entire groups of jobs can be temporarily suspended or enabled -- for example, general dispatching can be paused while a series of administrative jobs are allowed to proceed. The Dashboard also provides some controls for filtering and sorting jobs based on tiers.

Tier membership is stored on each job as a simple tier name attribute. The *tier definitions* themselves are located in tractor.config with other policy settings.

Tier names, and the number of tiers defined, are arbitrary and can be changed as desired. Tractor will ensure that a tier named "default" always exists, even if one is not defined explicitly. Other tier names and definitions are not required. If a job refers to a tier name that does not exist then it is handled according to the "default" tier settings.

NOTE: **Each tier can have a different scheduling policy,** as described below. If a tier does not specify a policy, then the "JobSchedulingMode" fallback value is used.

```
 "JobSchedulingMode": "P+ATCL+RR",

 "DispatchTiers": {
    "admin":   {"priority": 100.0, "scheduling": "P+FIFO"},
    "rush":    {"priority":  75.0},
    "preview": {"priority":  60.0, "scheduling": "P+CHKPT"},
    "default": {"priority":  50.0},
    "batch":   {"priority":  25.0}
 },
```

Scheduling Modes

Tractor supports several job queue sorting modes that lie at the heart of the main task assignment scheduling system. Most studios find that *P+FIFO* is a good match to a mostly-batch style of jobs, while *P+ATCL+RR* is good for a mostly-interactive mix. The *P+ATCL* scheme is often a reasonable middle ground. In all of these modes, **simple job priority** is the primary factor determining job processing order. Be sure to review the *Dispatching Tiers* section above as well for prioritizing entire classes of jobs.

Within each tier jobs are sorted by the tier's Scheduling Mode. Modes that start with "P+" always sort jobs first by numerical Priority value, highest to lowest. Jobs with the same priority are considered in the order produced by the selected policy mode:

- **P+FIFO** -- This is the default scheme for assigning available blades to jobs. It always picks the job with the highest priority value (largest numerical value), and if there are several jobs with the same priority then it selects the one that was submitted first. All runnable tasks in that job will be assigned as matching blades become available. If more blades are available than runnable tasks, or an available blade does not match the requirements of the top job, then the next job in the sorted queue is checked.
- **P+RR** -- After the primary priority sort, a simple "Round-Robin" scheme is used to select among jobs of equal priority. Within a tier, jobs with the same priority are visited in a circular fashion as blades become available. Several jobs with many ready tasks can be running concurrently. This mode tends to favor jobs with long-running tasks since those jobs may still have existing running tasks when their next "turn" comes back around; thus they will tend to "collect" blades if the job pool is small. As a result, this mode is rarely used in production, consider *P+ATCL+RR* instead.
- **P+ATCL** -- Active Task Count Leveling, this mode prefers to assign available blades to jobs with the fewest active tasks, attempting to equalize the number of active tasks in each job. When there are more ready tasks in the queue than available blade slots, then older jobs will be allowed to finish before newer jobs start. Given that it maintains roughly equal active task counts, jobs with short-running tasks will finish sooner than jobs having the same number of long-running tasks. Generally tends to favor the oldest jobs.
- **P+ATCL+RR** -- This mode is similar to *P+ATCL* with a difference in how it handles more waiting tasks than blades. It adds a round-robin component (within a group of equal priority jobs), tending to balance active task count *over time* across *all* jobs in the priority block. Generally favors jobs that have been waiting longest for blades.

  In both ATCL schemes, older jobs using many blades on the farm will get fewer and fewer blades as more jobs are spooled into the system. This is usually the desired behavior: studios choose it so that newly spooled jobs get some rendering feedback "soon" rather than having to wait for all prior jobs to finish first. The disadvantage is, of course, that the total number of blades assigned to older jobs diminishes as new jobs are spooled.

  As an example, consider the case where there are 100 waiting jobs at the same priority, and only 25 blades in the farm. The *P+ATCL* scheme will eventually converge on assigning 1 blade to each of the oldest 25 jobs, and the remaining 75 newer jobs will be idle until the "window" of 25 blades steps across them, as the oldest jobs from the currently active set finishes. The "P+ATCL+RR" mode will instead cause blades that finish a task to be assigned more broadly across *all* of those waiting jobs. So all 100 jobs will get some of the blades periodically. Both schemes favor assignments to the jobs with fewest already-running tasks. When *that* criteria is the same, then P+ATCL favors the oldest jobs, whereas P+ATCL+RR favors jobs that have been waiting longest for a new blade assignment.

- **P+CHKPT** -- A special policy enabling re-execution of tasks that participate in checkpoint - yield - resume processing across all of the jobs in the tier. This is generally a FIFO ordering but each job is suspended once it has reached a checkpoint on all active tasks. Processing resumes again with the top job when all jobs in an equal priority group have reached that point.

Limit Sharing and Resource Allocations

See also the Tractor Limits discussion.