LimitingTaskSlotUsage

Limiting Task Slot Usage

There are many reasons why one wants to limit the number of running tasks.

Job is consuming all task slots

The most common reason is because a given job is consuming all of the available task slots, preventing other jobs from running. The easiest and best solution is to switch from the default FIFO scheduler to another scheduler, such as the FairShareScheduler or the CapacityScheduler. Both solve this problem in slightly different ways. Depending upon need, one may be a better fit than the other.

Job has taken too many reduce slots that are still waiting for maps to finish

There is a job tunable called mapred.reduce.slowstart.completed.maps that sets the percentage of maps that must be completed before firing off reduce tasks. By default, this is set to 5% (0.05) which for most shared clusters is likely too low. Recommended values are closer to 80% or higher (0.80). Note that for jobs that have a significant amount of intermediate data, setting this value higher will cause reduce slots to take more time fetching that data before performing work.

Job is referencing an external, limited resource (such as a database)

In Hadoop terms, we call this a 'side-effect'.

One of the general assumptions of the framework is that there are not any side-effects. All tasks are expected to be restartable and a side-effect typically goes against the grain of this rule.

If a task absolutely must break the rules, there are a few things one can do:

- Disable SpeculativeExecution .
- Deploy ZooKeeper and use it as a persistent lock to keep track of how many tasks are running concurrently
- Use a scheduler with a maximum task-per-queue feature and submit the job to that queue, such as FairShareScheduler or CapacityScheduler

Job consumes too much RAM/disk IO/etc on a given node

The CapacityScheduler in 0.21 has a feature whereby one may use RAM-per-task to limit how many slots a given task takes. By careful use of this feature, one may limit how many concurrent tasks on a given node a job may take.

Increasing the Number of Slots Used

There are both job and server-level tunables that impact how many tasks are run concurrently.

Increase the amount of tasks per node

There are two server tunables that determine how many tasks a given TaskTracker will run on a node:

- mapred.tasktracker.map.tasks.maximum sets the map slot usage
- mapred.tasktracker.reduce.tasks.maximum sets the reduce slot usage

These must be set in the mapred-site.xml file on the TaskTracker. After making the change, the TaskTracker must be restarted to see it. One should see the values increase (or decrease) on the JobTracker main page. Note that this is **not** set by your job.

Increase the amount of map tasks

Typically, the amount of maps per job is determined by Hadoop based upon the InputFormat and the block size in place. Using mapred.min.split.size and mapred.max.split.size settings, one can provide hints to the system that it should use a size that is different than the block size to determine what the min and max input size should be.

Increase the amount of reduce tasks

Currently, the number of reduces is determined by the job. mapred.reduce.tasks should be set by the job to the appropriate number of reduces. When using Pig, use the PARALLEL keyword.