Capacity Scheduler Guide

Table of contents

1 Purpose....................2
2 Features....................2
3 Picking a task to run....................2
4 Installation....................3
5 Configuration....................3
5.1 Using the Capacity Scheduler....................3
5.2 Setting up queues....................3
5.3 Configuring properties for queues....................3
5.4 Memory management....................5
5.5 Job Initialization Parameters....................6
5.6 Reviewing the configuration of the Capacity Scheduler....................7

1. Purpose

This document describes the Capacity Scheduler, a pluggable MapReduce scheduler for Hadoop which provides a way to share large clusters.

2. Features

The Capacity Scheduler supports the following features:

- Support for multiple queues, where a job is submitted to a queue.
- Queues are allocated a fraction of the capacity of the grid in the sense that a certain capacity of resources will be at their disposal. All jobs submitted to a queue will have access to the capacity allocated to the queue.
- Free resources can be allocated to any queue beyond it's capacity. When there is demand for these resources from queues running below capacity at a future point in time, as tasks scheduled on these resources complete, they will be assigned to jobs on queues running below the capacity.
- Queues optionally support job priorities (disabled by default).
- Within a queue, jobs with higher priority will have access to the queue's resources before jobs with lower priority. However, once a job is running, it will not be preempted for a higher priority job, though new tasks from the higher priority job will be preferentially scheduled.
- In order to prevent one or more users from monopolizing its resources, each queue enforces a limit on the percentage of resources allocated to a user at any given time, if there is competition for them.
- Support for memory-intensive jobs, wherein a job can optionally specify higher memory-requirements than the default, and the tasks of the job will only be run on TaskTrackers that have enough memory to spare.

3. Picking a task to run

Note that many of these steps can be, and will be, enhanced over time to provide better algorithms.

Whenever a TaskTracker is free, the Capacity Scheduler picks a queue which has most free space (whose ratio of # of running slots to capacity is the lowest).

Once a queue is selected, the Scheduler picks a job in the queue. Jobs are sorted based on when they're submitted and their priorities (if the queue supports priorities). Jobs are considered in order, and a job is selected if its user is within the user-quota for the queue, i.e., the user is not already using queue resources above his/her limit. The Scheduler also makes sure that there is enough free memory in the TaskTracker to tun the job's task, in case the job has special memory requirements.

Once a job is selected, the Scheduler picks a task to run. This logic to pick a task remains unchanged from earlier versions.

4. Installation

The Capacity Scheduler is available as a JAR file in the Hadoop tarball under the *contrib/capacity-scheduler* directory. The name of the JAR file would be on the lines of hadoop-*-capacity-scheduler.jar.

You can also build the Scheduler from source by executing *ant package*, in which case it would be available under *build/contrib/capacity-scheduler*.

To run the Capacity Scheduler in your Hadoop installation, you need to put it on the *CLASSPATH*. The easiest way is to copy the `hadoop-*-capacity-scheduler.jar` from to `HADOOP_HOME/lib`. Alternatively, you can modify *HADOOP_CLASSPATH* to include this jar, in `conf/hadoop-env.sh`.

5. Configuration

5.1. Using the Capacity Scheduler

To make the Hadoop framework use the Capacity Scheduler, set up the following property in the site configuration:

Property	Value
mapred.jobtracker.taskScheduler	org.apache.hadoop.mapred.CapacityTaskScheduler

5.2. Setting up queues

You can define multiple queues to which users can submit jobs with the Capacity Scheduler. To define multiple queues, you should edit the site configuration for Hadoop and modify the *mapred.queue.names* property.

You can also configure ACLs for controlling which users or groups have access to the queues.

For more details, refer to Cluster Setup documentation.

5.3. Configuring properties for queues

The Capacity Scheduler can be configured with several properties for each queue that control the behavior of the Scheduler. This configuration is in the *conf/capacity-scheduler.xml*. By default, the configuration is set up for one queue, named *default*.

To specify a property for a queue that is defined in the site configuration, you should use the property name as *mapred.capacity-scheduler.queue.<queue-name>.<property-name>*.

For example, to define the property *capacity* for queue named *research*, you should specify the property name as *mapred.capacity-scheduler.queue.research.capacity*.

The properties defined for queues and their descriptions are listed in the table below:

Name	Description
mapred.capacity-scheduler.queue.<queue-name>	Percentage of the number of slots in the cluster that are made to be available for jobs in this queue. The sum of capacities for all queues should be less than or equal 100.
mapred.capacity-scheduler.queue.<queue-name>	If true, priorities of jobs will be taken into account in scheduling decisions.
mapred.capacity-scheduler.queue.<queue-name>	Each queue enforces a limit on the percentage of resources allocated to a user at any given time, if there is competition for them. This user limit can vary between a minimum and maximum value. The former depends on the number of users who have submitted jobs, and the latter is set to this property value. For example, suppose the value of this property is 25. If two users have submitted jobs to a queue, no single user can use more than 50% of the queue resources. If a third user submits a job, no single user can use more than 33% of the queue resources. With 4 or more users, no user can use more than 25% of the queue's resources. A value of 100 implies no user limits are imposed.
mapred.capacity-scheduler.queue.<queue-name>	maximum-capacity defines a limit beyond which a queue cannot use the capacity of the cluster.This provides a means to limit how much excess capacity a queue can use. By default, there is no limit. The maximum-capacity of a queue can only be greater than or equal to its minimum capacity. Default value of -1 implies a queue can use complete capacity of the cluster. This property could be to curtail certain jobs

	which are long running in nature from occupying more than a certain percentage of the cluster, which in the absence of pre-emption, could lead to capacity guarantees of other queues being affected. One important thing to note is that maximum-capacity is a percentage , so based on the cluster's capacity it would change. So if large no of nodes or racks get added to the cluster , maximum Capacity in absolute terms would increase accordingly.

5.4. Memory management

The Capacity Scheduler supports scheduling of tasks on a `TaskTracker`(TT) based on a job's memory requirements and the availability of RAM and Virtual Memory (VMEM) on the TT node. See the MapReduce Tutorial for details on how the TT monitors memory usage.

Currently the memory based scheduling is only supported in Linux platform.

Memory-based scheduling works as follows:

1. The absence of any one or more of three config parameters or -1 being set as value of any of the parameters, `mapred.tasktracker.vmem.reserved`, `mapred.task.default.maxvmem`, or `mapred.task.limit.maxvmem`, disables memory-based scheduling, just as it disables memory monitoring for a TT. These config parameters are described in the MapReduce Tutorial. The value of `mapred.tasktracker.vmem.reserved` is obtained from the TT via its heartbeat.
2. If all the three mandatory parameters are set, the Scheduler enables VMEM-based scheduling. First, the Scheduler computes the free VMEM on the TT. This is the difference between the available VMEM on the TT (the node's total VMEM minus the offset, both of which are sent by the TT on each heartbeat)and the sum of VMs already allocated to running tasks (i.e., sum of the VMEM task-limits). Next, the Scheduler looks at the VMEM requirements for the job that's first in line to run. If the job's VMEM requirements are less than the available VMEM on the node, the job's task can be scheduled. If not, the Scheduler ensures that the TT does not get a task to run (provided the job has tasks to run). This way, the Scheduler ensures that jobs with high memory requirements are not starved, as eventually, the TT will have enough VMEM available. If the high-mem job does not have any task to run, the Scheduler moves on to the next job.
3. In addition to VMEM, the Capacity Scheduler can also consider RAM on the TT node. RAM is considered the same way as VMEM. TTs report the total RAM available on their node, and an offset. If both are set, the Scheduler computes the available RAM on the node. Next, the Scheduler figures out the RAM requirements of the job, if any. As with VMEM, users can optionally specify a RAM limit for their job

`mapred.task.maxpmem`, described in the MapReduce Tutorial). The Scheduler also maintains a limit for this value (`mapred.capacity-scheduler.task.default-pmem-percentage-in-vmem`, described below). All these three values must be set for the Scheduler to schedule tasks based on RAM constraints.

4. The Scheduler ensures that jobs cannot ask for RAM or VMEM higher than configured limits. If this happens, the job is failed when it is submitted.

As described above, the additional scheduler-based config parameters are as follows:

Name	Description
mapred.capacity-scheduler.task.default-pmem-pe	A percentage of the default VMEM limit for jobs (`mapred.task.default.maxvmem`). This is the default RAM task-limit associated with a task. Unless overridden by a job's setting, this number defines the RAM task-limit.
mapred.capacity-scheduler.task.limit.maxpmem	Configuration which provides an upper limit to maximum physical memory which can be specified by a job. If a job requires more physical memory than what is specified in this limit then the same is rejected.

5.5. Job Initialization Parameters

Capacity scheduler lazily initializes the jobs before they are scheduled, for reducing the memory footprint on jobtracker. Following are the parameters, by which you can control the laziness of the job initialization. The following parameters can be configured in capacity-scheduler.xml

Name	Description
mapred.capacity-scheduler.queue.<queue-name>	Maximum number of jobs which are allowed to be pre-initialized for a particular user in the queue. Once a job is scheduled, i.e. it starts running, then that job is not considered while scheduler computes the maximum job a user is allowed to initialize.
mapred.capacity-scheduler.init-poll-interval	Amount of time in miliseconds which is used to poll the scheduler job queue to look for jobs to be initialized.
mapred.capacity-scheduler.init-worker-threads	Number of worker threads which would be used by Initialization poller to initialize jobs in a set of queue. If number mentioned in property is equal

	to number of job queues then a thread is assigned jobs from one queue. If the number configured is lesser than number of queues, then a thread can get jobs from more than one queue which it initializes in a round robin fashion. If the number configured is greater than number of queues, then number of threads spawned would be equal to number of job queues.

5.6. Reviewing the configuration of the Capacity Scheduler

Once the installation and configuration is completed, you can review it after starting the MapReduce cluster from the admin UI.

- Start the MapReduce cluster as usual.
- Open the JobTracker web UI.
- The queues you have configured should be listed under the *Scheduling Information* section of the page.
- The properties for the queues should be visible in the *Scheduling Information* column against each queue.