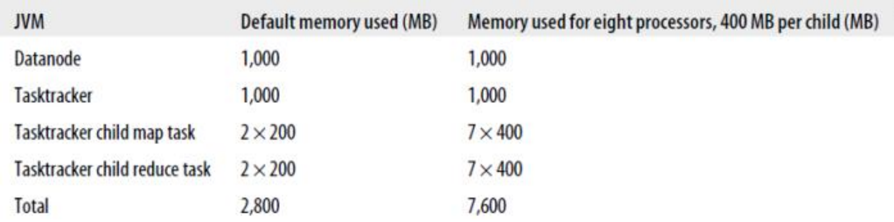
Hadoop Learning - MapReduce Part

Sunday, March 30, 2014 8:50

1. Map Reduce Process
   a. Mapper: **<K1, V1> ==> list<K2, V2>**
   b. Reducer: **<K2, List<V2>> ==> <K3, V3>**

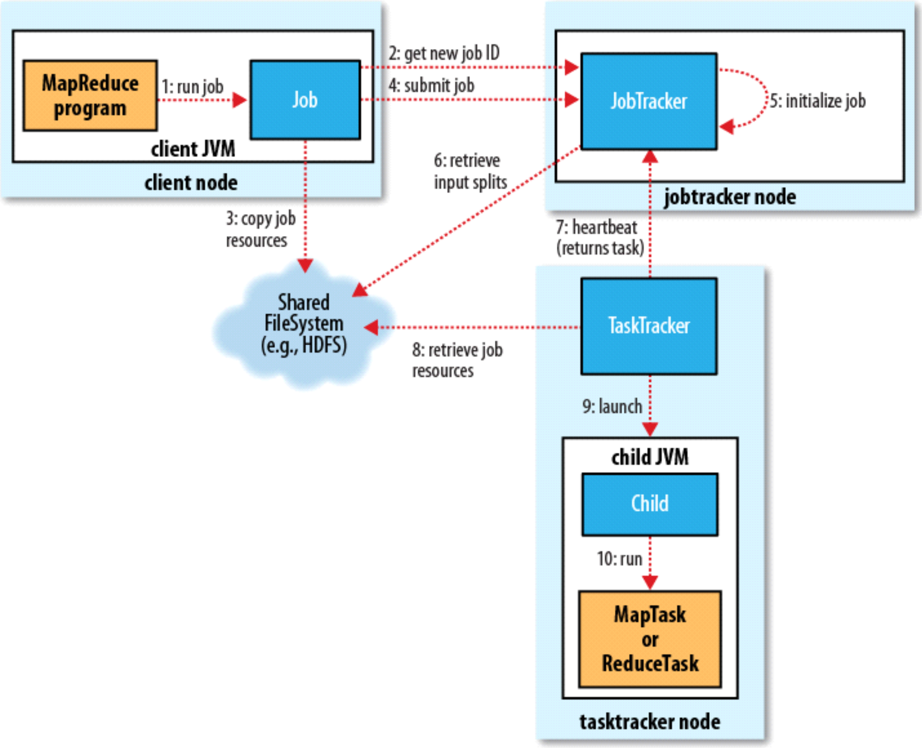
2. **Configuration - Important properties**
   a. mapred.job.tracker - The Server location of Job tracker
   b. mapred.local.dir - The storage path for datanode's mapper tasks intermediate data.
   c. mapred.system.dir - The shared storage path for datanodes' reducer tasks
   d. mapred.tasktracker.map.tasks.maximum - The maximum number of mapper tasks
   e. mapred.tasktracker.reduce.tasks.maximum - The maximum number of reducer tasks

3. MapReduce Memory Structure
   a. Master Node (Namenode)
      i. **Each of the namenode, secondary namenode, and jobtracker daemons uses 1,000 MB by default, for a total of 3,000 MB.**
      ii. We can increase the namenode's memory without changing the memory allocated to other Hadoop daemons by setting **HADOOP_NAMENODE_OPTS** in **hadoop-env.sh.**
      iii. Likewise, we can increase secondary namenode's memory by setting **HADOOP_SECONDARYNAMENODE_OPTS** variable also in **hadoop-env.sh**
   b. Worker Node (Datanode)
      i. By default, Hadoop allocates **1,000MB (1 GB)** of memory to each daemon it runs. This is controlled by the **HADOOP_HEAPSIZE** setting in **hadoop-env.sh**.
      ii. In addition, the task tracker launches separate child JVMs to run map and reduce tasks in. The memory given to each child JVM running a task can be changed by setting the **mapred.child.java.opts** property. The default setting is **-Xmx200m**, which gives each task 200 MB of memory.



JVM	Default memory used (MB)	Memory used for eight processors, 400 MB per child (MB)
Datanode	1,000	1,000
Tasktracker	1,000	1,000
Tasktracker child map task	2 × 200	7 × 400
Tasktracker child reduce task	2 × 200	7 × 400
Total	2,800	7,600

4. Anatomy of MapReduce Works



5. Failures in MapReduce
   a. Task level failure
      i. User code failed. Just failed.
      ii. Task JVM crash. Just failed
      iii. Timeout, which can be set in **mapred.task.timout**

iv. After several times' failure. Job tracker will not assign it again. (Set in **mapred.map.max.attempt**s and **mapred.reduce.max.attempts**)

v. It may be possible to use the result even having some failures (Set in the **mapred.max.map.failures.percent** and **mapred.max.reduce.failures.percent**)

b. Tasktracker level failure

i. Heart beat back to Job tracker (Set in **mapred.tasktracker.expiry.interval**)

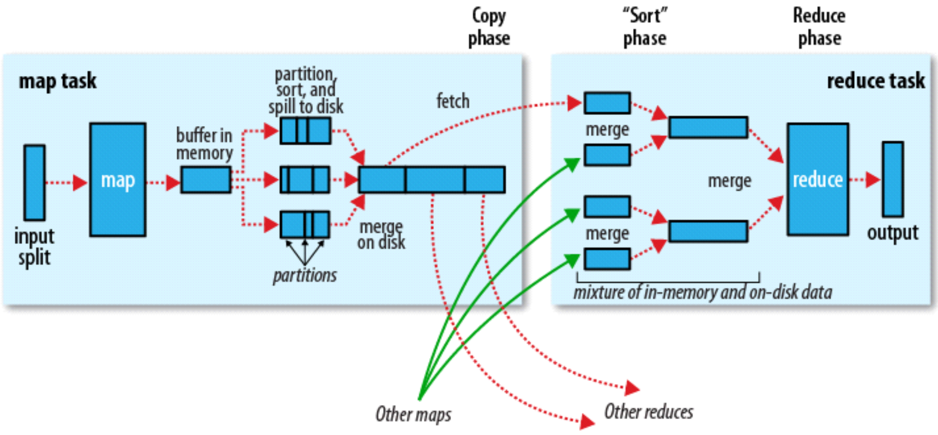
ii. If more than several tasks from the same job fail on a particular tasktracker (set by **mapred.max.tracker.failures**), the jobtracker records this as a fault.

iii. A tasktracker is blacklisted if the number of faults is over some minimum threshold (Set by **mapred.max.tracker.blacklists**, which default is four)

c. JobTracker level failure

i. Most serious failure

6. Shuffle and Sort



a. Mapper Side

i. **Each map task has a circular memory buffer that it writes the output to**. The buffer is 100 MB by default, a size that can be tuned by changing the **io.sort.mb** property.

ii. **When the contents of the buffer reaches a certain threshold size** (io.sort.spill.percent, which has the default 0.80, or 80%), **a background thread will start to spill the contents to disk.**

iii. Spills are written in round-robin fashion to the directories specified by the mapred.local.dir property, in a job-specific subdirectory.

iv. **Before it writes to disk, the thread first divides the data into partitions corresponding to the reducers that they will ultimately be sent to.** Within each partition, the background thread performs an in-memory sort by key.

v. Each time the memory buffer reaches the spill threshold, a new spill file is created, so after the map task has written its l ast output record, there could be several spill files.

vi. Before the task is finished, the spill files are merged into a single partitioned and sorted output file. The configuration property **io.sort.factor** controls the maximum number of streams to merge at once; the default is 10.

vii. The output file's partitions are made available to the reducers over HTTP. The maximum number of worker threads used to serve the file partitions is controlled by the **tasktracker.http.threads** property; **this setting is per tasktracker, not per map task slot.** The default of 40 may need to be increased for large clusters running large jobs.

viii. By default, the output is not compressed, but it is easy to enable this by setting **mapred.compress.map.output to true**. The compression library to use is specified by **mapred.map.output.compression.codec**.

b. Reducer side

i. The map output file is sitting on the local disk of the machine that ran the map task

ii. The map tasks may finish at different times, so the reduce task starts copying their outputs as soon as each completes.

iii. The reduce task has a small number of copier threads so that it can fetch map outputs in parallel. The default is five thread s, but this number can be changed by setting the **mapred.reduce.parallel.copies** property.

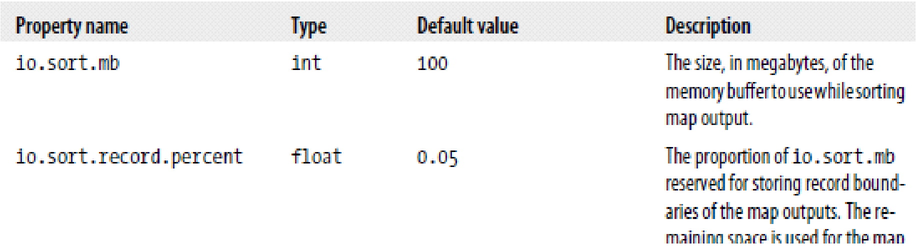
iv. **As map tasks complete successfully, they notify their parent tasktracker of the status update, which in turn notifies the jobtracker**, which is how reducers know which machines to fetch map output from.

v. The map outputs are copied to the reduce task JVM's memory if they are small enough (the buffer's size is controlled by **mapred.job.shuffle.input.buffer.percent**, which specifies the proportion of the heap to use for this purpose); otherwise, they are copied to disk.

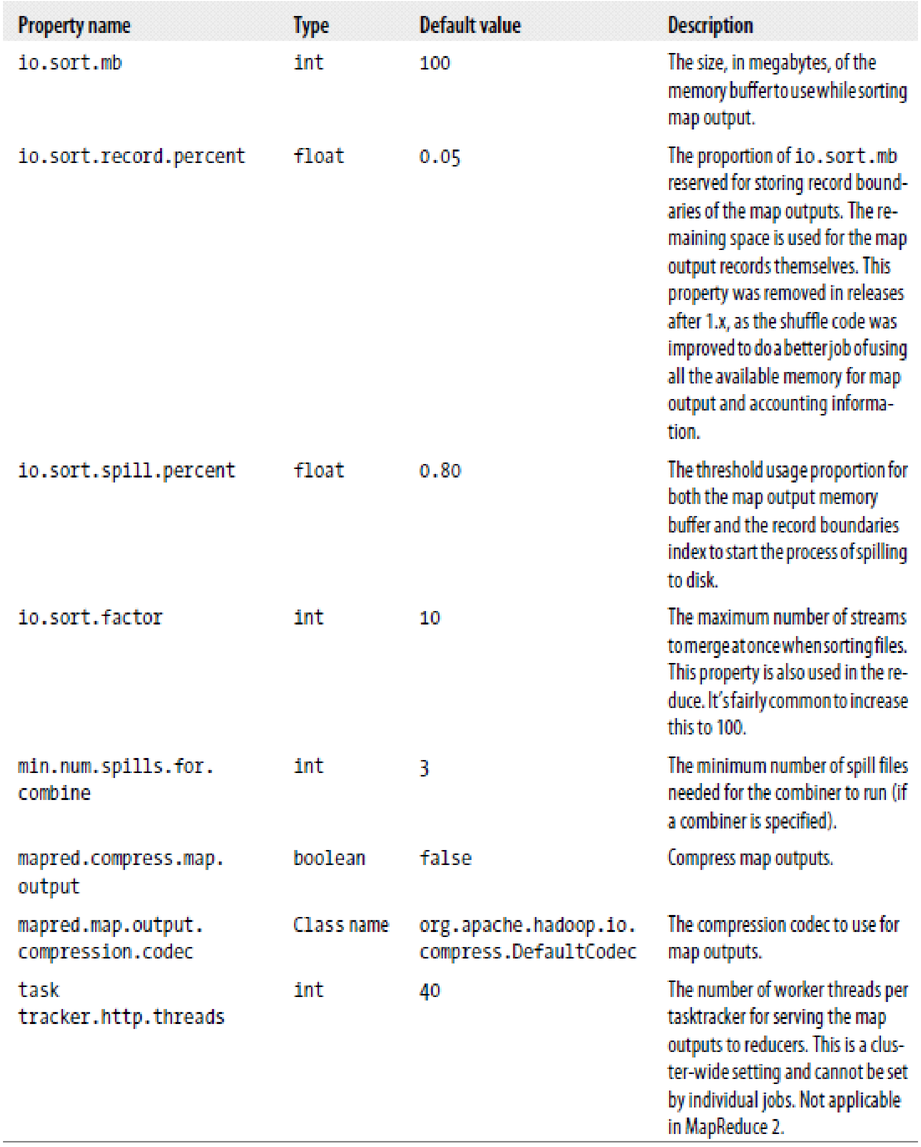
vi. When the in-memory buffer reaches a threshold size (controlled by **mapred.job.shuffle.merge.percent**) or reaches a threshold number of map outputs (**mapred.inmem.merge.threshold**), it is merged and spilled to disk.

vii. As the copies accumulate on disk, a background thread merges them into larger, sorted files.

viii. When all the map outputs have been copied, the reduce task moves into the sort phase (which should properly be called the merge phase)



Property name	Type	Default value	Description
io.sort.mb	int	100	The size, in megabytes, of the memory buffer to use while sorting map output.
io.sort.record.percent	float	0.05	The proportion of io.sort.mb reserved for storing record boundaries of the map outputs. The remaining space is used for the map



Property name	Type	Default value	Description
io.sort.mb	int	100	The size, in megabytes, of the memory buffer to use while sorting map output.
io.sort.record.percent	float	0.05	The proportion of io.sort.mb reserved for storing record boundaries of the map outputs. The remaining space is used for the map output records themselves. This property was removed in releases after 1.x, as the shuffle code was improved to do a better job of using all the available memory for map output and accounting information.
io.sort.spill.percent	float	0.80	The threshold usage proportion for both the map output memory buffer and the record boundaries index to start the process of spilling to disk.
io.sort.factor	int	10	The maximum number of streams to merge at once when sorting files. This property is also used in the reduce. It's fairly common to increase this to 100.
min.num.spills.for.combine	int	3	The minimum number of spill files needed for the combiner to run (if a combiner is specified).
mapred.compress.map.output	boolean	false	Compress map outputs.
mapred.map.output.compression.codec	Class name	org.apache.hadoop.io.compress.DefaultCodec	The compression codec to use for map outputs.
task tracker.http.threads	int	40	The number of worker threads per tasktracker for serving the map outputs to reducers. This is a cluster-wide setting and cannot be set by individual jobs. Not applicable in MapReduce 2.

7. Job Scheduling
    a. Early versions of Hadoop had a very simple approach to scheduling users' jobs: they ran in order of submission, using a FIFO scheduler.
    b. Later on, the ability to set a job's priority was added, via the mapred.job.priority property or the setJobPriority() method on JobClient (both of which take one of the values VERY_HIGH, HIGH, NORMAL, LOW, or VERY_LOW).
    c. **MapReduce in Hadoop comes with a choice of schedulers. The default in MapReduce 1 is the original FIFO queue -based scheduler.**
    d. User can choose the scheduler by setting in **mapred.jobtracker.taskScheduler**
    e. Scheduler Type:
        i. Fair Scheduler: The Fair Scheduler aims to give every user a fair share of the cluster capacity over time.
        ii. Capacity Scheduler
        iii. User-defined Scheduler