Advanced Data Management Technologies

Unit 17 — Executing MapReduce

J. Gamper

Free University of Bozen-Bolzano
Faculty of Computer Science
IDSE

Outline

Outline

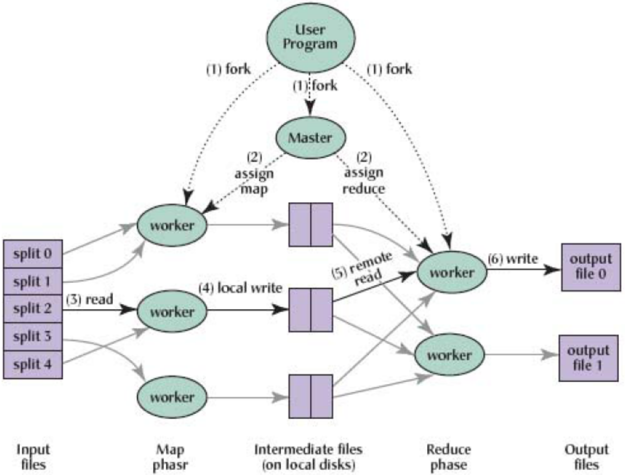
1. Task Scheduling in MapReduce

2. Error Handling

Some Terminology

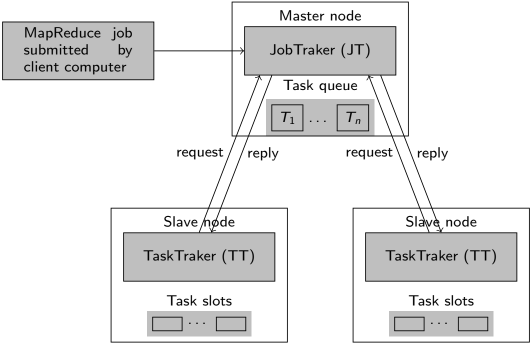
- A Job is a “full programm”, i.e., the execution of a Mapper and Reducer across a data set.
- A task is an exectuation of a Mapper or a Reducer on a slice of data.
- A task attempt is a particular instance of an attempt to execute a task on a machine.
  - A particular task will be attempted at least once, possibly more times if it crashes.
- Example
  - Running “word count” across 20 files is one job
  - 20 files to be mapped imply 20 map tasks + some reduce tasks.
  - At least 20 map task attempts will be performed (more if a machine crashes)

MR Architecture

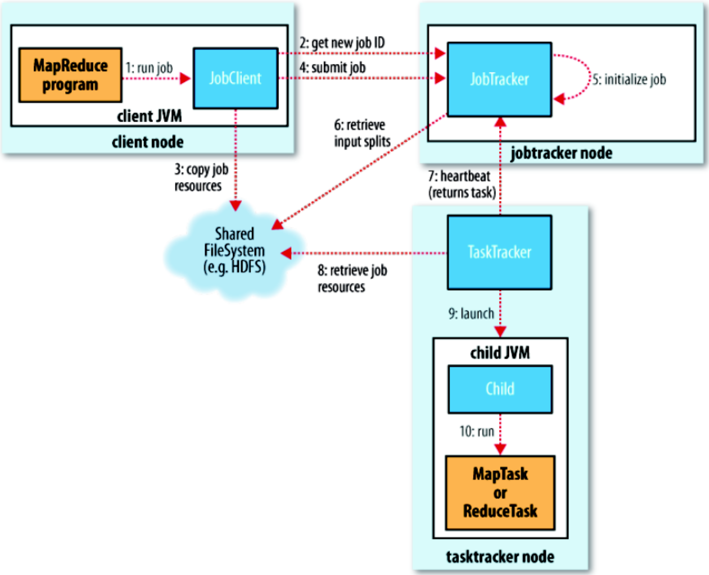


Program Execution – High Level View

- MapReduce adopts a master-slave architecture.
- The master node in MapReduce is referred to as Job Tracker (JT).
- Each slave node in MapReduce is referred to as Task Tracker (TT).
- MapReduce adopts a pull scheduling strategy rather than a push one,
  - i.e., JT does not push map and reduce tasks to TTs, but TTs pull them by making pertaining requests.



Programm Execution – Details



Task Execution/1

- Every TT sends a heartbeat message periodically to JT
  - tells that TT is alive,
  - but contains also requests for a map or a reduce task to run,
  - or simply the return of a task.
- When a new task is requested, the JT chooses a job and selects a task from that job.
- Map Task Scheduling
  - JT satisfies requests for map tasks via attempting to schedule mappers in the vicinity of their input splits, i.e., locality is considered.
- Reduce Task Scheduling
  - However, JT simply assigns the next yet-to-run reduce task to a requesting TT regardless of TTs network location and its implied effect on the reducers shuffle time, i.e., locality is not considered.

Task Execution/2

- MapReduce programs are contained in a Java “jar” file + an XML file containing serialized program configuration options.
- Running a MapReduce job places these files into the HDFS and notifies TaskTrackers where to retrieve the relevant program code.
- Task execution consists of the following steps for a TT:
  - Copy the JAR-file from the HDFS to the local file system.
  - Similar, configuration data are copied from the distributed cache.
  - The actual task is run in a new JVM to avoid that bugs in user-defined map and reduce functions affect the tasktracker.

Data Distribution

- All data is accessible via a distributed filesystem with replication, such as HDFS or GFS
- Files in GFS (and similar in HDFS) are
  - divided into chunks (default 64MB) and
  - stored with replications (typically 3 replicas on different nodes)
- Data transfer is handled by the distributed file system

Locality

- Since all mappers are equivalent, the master tries to do the work on nodes that store a replica of the data
  - Reading from local disk is much faster than reading from a remote server
- MR uses locality hints from GFS/HDFS and assigns map tasks as follows:
  - Try to assign a task to a machine with a local copy of the input data;
  - or, less preferable, to a machine for which a copy of the data is stored on a server on the same network switch;
  - or, assign to any available worker.

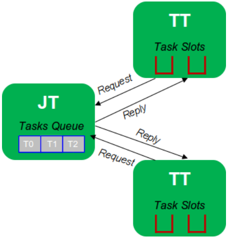
Job Scheduling

- MapReduce in Hadoop comes with a choice of schedulers
- The default is the FIFO scheduler which schedules jobs in order of submission.
- There is also a multi-user scheduler, called the fair scheduler, which aims to give every user a fair share of the cluster capacity over time.

Outline

Fault Tolerance in Hadoop

- MR can successfully complete jobs, even when they are executed on large clusters, where the probability of failures increases.
- The primary way for MR to achieve fault tolerance is through restarting tasks.

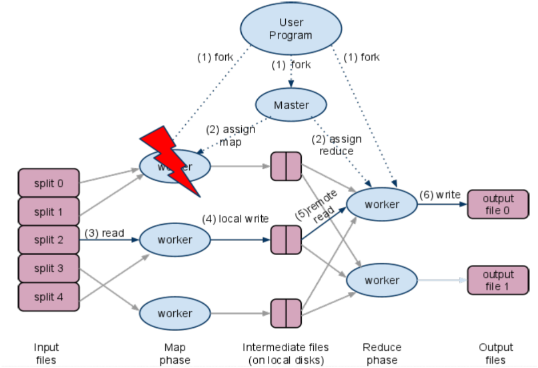


- If a TT fails to communicate with the JT for a period of time (by default, 1 minute in Hadoop), the JT assumes that the TT has crashed:
  - Job is still in the map phase: JT asks another TT to re-execute all mappers that previously ran at the failed TT.
  - Job is in the reduce phase: JT asks another TT to re-execute all reducers that were in progress on the failed TT.

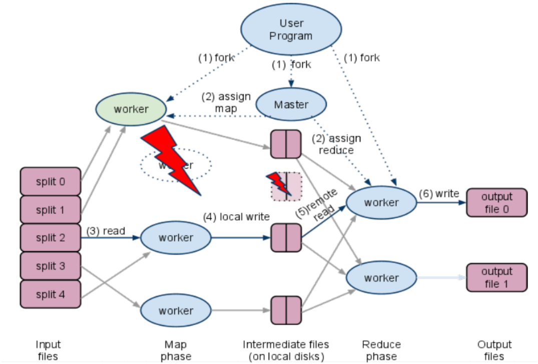
Task Failure

- Task in a local node fails
  - The TT marks the task attempt as failed and notifies the JT.
  - The JT re-schedules the task if possible on another node.
  - The slot in the local node is freed up for another task.
- Hanging tasks in a local node
  - If the TT gets no progress update for a while, the task is marked as failed.
  - The JVM will be killed after a timeout (normally 10 minutes).
  - The JT is notified about the failed task.
- Setting the timeout to zero
  - Disables the timeout
  - Long-running tasks are never marked as failed.
  - A hanging task will never free up its slot → cluster slowdown over time
  - Not recommended!

Recover from Task Failure by Re-execution/1



Recover from Task Failure by Re-execution/2

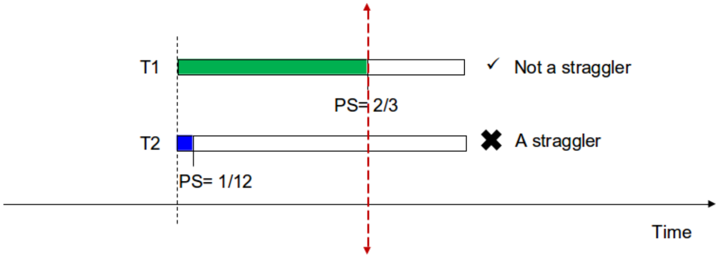


Speculative Execution

- A MR job is dominated by the slowest task.
- MR attempts to locate slow tasks, called stragglers.
- If a straggler is discovered, a redundant (speculative) task is run that will optimistically commit before the corresponding straggler.
- Whichever copy (among the two copies) of a task commits first, it becomes the definitive copy, and the other copy is killed by the JT.
- This process is known as speculative execution.
- Only one copy of a straggler is allowed to be speculated.

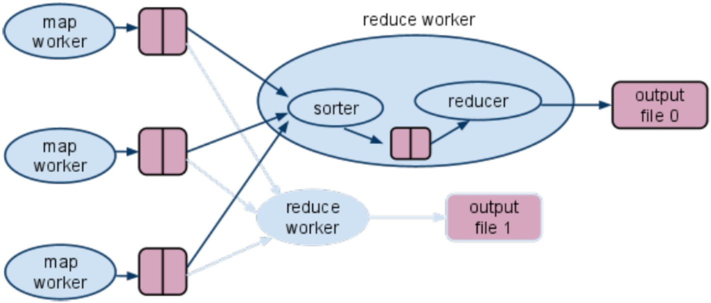
How does Hadoop Locate Stragglers?

- Hadoop monitors each task progress using a progress score between 0 and 1
- If a task's progress score is less than $(average - 0.2)$, and the task has run for at least 1 minute, it is marked as a straggler



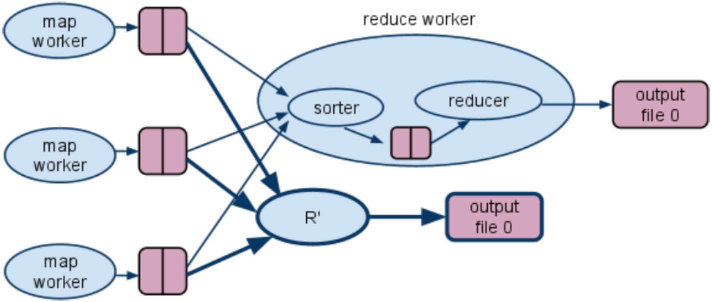
Dealing with Reduce Stragglers/1

- Stragglers in the reduce phase are particularly expensive:
  - Reducer retrieves data remotely from many servers
  - Sorting is expensive on local resources
  - Reducing usually can not start until Mapping is done
- Re-execution due to machine failures could double the runtime.



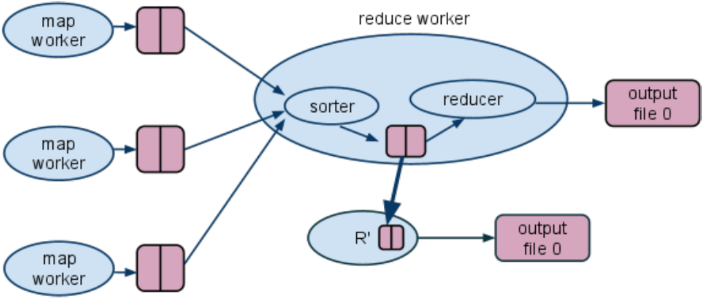
Dealing with Reduce Stragglers/2

- Technique 1: Create a backup instance as early and as necessary as possible.



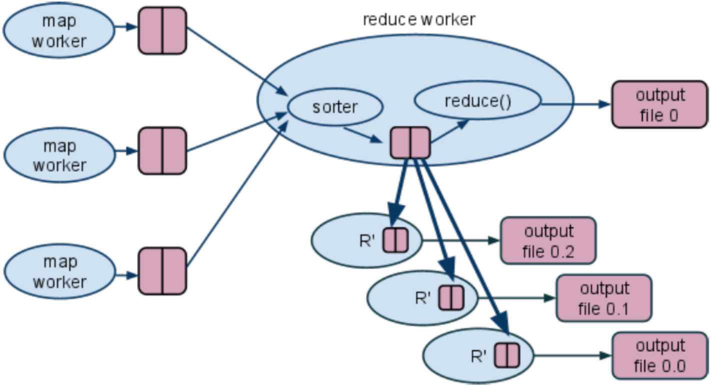
Dealing with Reduce Stragglers/3

- Technique 2: Retrieving map output and sorting are expensive, but we can transport the sorted input to the backup reducer.



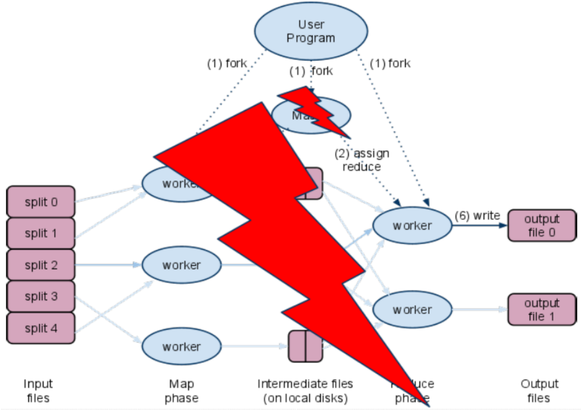
Dealing with Reduce Stragglers/4

- Technique 3: Divide a reduce task into smaller ones to take advantage of more parallelism.



Master as Single Point of Failure

- Most serious failure mode: the JT fails, and hence all running jobs fail.
- Hadoop has no mechanism for dealing with JT failure.
- After restart, all jobs that were running at the time of failure need to be resubmitted.



Summary

- MR programm execution is based on a master-slave architecture
- The master node runs the JobTracker (JT), the slave nodes run TaskTracker (TT)
- Pull scheduling strategy, i.e., TTs pull tasks from JT.
- TT send heartbeat message to JT
- Locality principle in assigning map tasks is applied.
- FIFO and Fair scheduler are available.
- Error handling mainly through restarting tasks
- Start speculative tasks to deal with stragglers
- Reduce stragglers are more expensive.
- Master failure is most serious failure – single point of failure. Restart!