How to Use the MapReduce API

February 17, 2015 | BY James Casaletto (/blog/author/james-casaletto/)

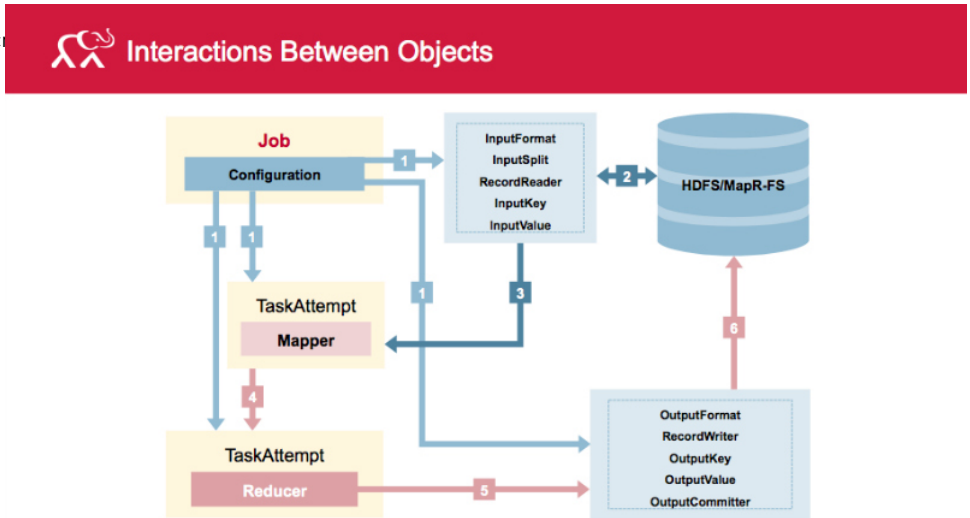
Hadoop MapReduce is a framework that simplifies the process of writing big data applications running in parallel on large clusters of commodity hardware.

The MapReduce framework consists of a single master ResourceManager, one slave NodeManager per cluster-node, and one MRAppMaster per application (see the YARN Architecture Guide). Each MapR software release supports and ships with a specific version of Hadoop (/products/apache-hadoop). For example, MapR 3.0.1 shipped with Hadoop 0.20.2, while MapR 4.0.1 uses Hadoop 2.4 including YARN.

There are two API packages to choose when developing MapReduce applications: org.apache.hadoop.mapred and org.apache.hadoop.mapreduce. In this blog post, we'll briefly review the MapReduce API, examine the Mapper input processing and Reducer output processing data flow, as well as explore the Mapper, Reducer, and Job class API.

Framework Functionality

A MapReduce job usually splits the input data set into independent chunks which are processed by the map tasks in a completely parallel manner. The framework sorts the outputs of the maps, which are then input to the reduce tasks. Typically, both the input and the output of the job are stored in a file system. The framework takes care of scheduling and monitoring tasks, then re-executes the failed tasks.

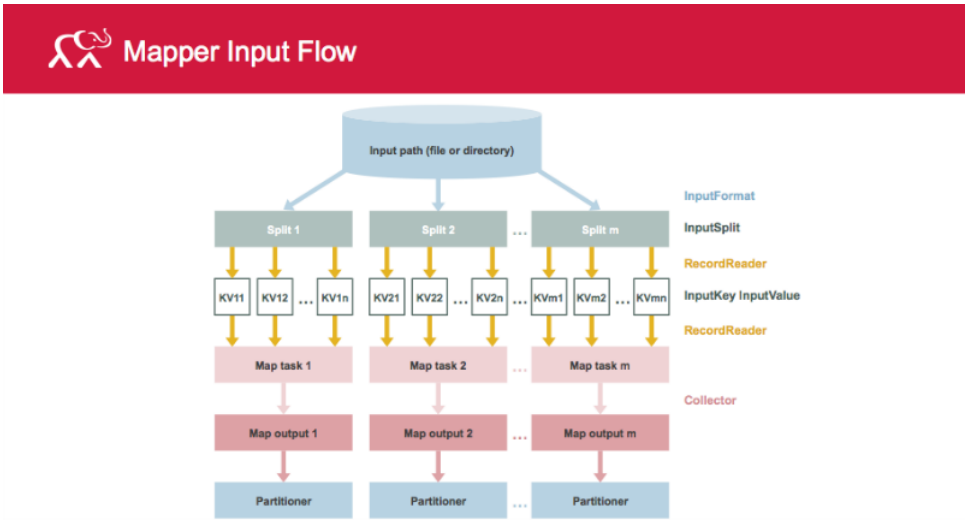


Typically, the compute nodes and the storage nodes are the same—that is, the MapReduce framework and the Hadoop Distributed File System are running on the same set of nodes. This configuration allows the framework to effectively schedule tasks on the nodes where data is already present, resulting in very high aggregate bandwidth across the cluster.

Minimally, applications specify the input/output locations and supply map and reduce functions via implementations of appropriate interfaces and/or abstract classes. These, and other job parameters, comprise the job configuration.

The Hadoop job client then submits the job (jar/executable, etc.) and configuration to the ResourceManager which then assumes the responsibility of distributing the software/configuration to the slaves, scheduling tasks and monitoring them, and providing status and diagnostic information to the job-client.

The Mapper and Reducer



In a running Hadoop job, applications typically implement the Mapper and Reducer interfaces to provide the map (individual tasks transforming input records into intermediate records) and reduce methods to reduce a set of intermediate values which share a key to a smaller set of values.

- The driver class instantiates a Job object and its configuration, and then uses set() methods to define each element of the Configuration object.
- The Mapper reads data from the Hadoop file system to do its processing.
- The Mapper leverages the InputFormat, InputSplit, RecordReader, InputKey, and InputValue types to process its input.

- The reducer writes data to the Hadoop file system after it is done processing, using the outputformat, recordwriter, outputkey, outputvalue, and outputcommittertypes to process its output.
- The number of maps produced typically reflects the total number of blocks of the input files.

Mapper Input

The InputFormat class performs the following tasks:

- Validates the input files and directories that exist for the job. (You'll see an IOException on invalid input.)
- Splits the input files into InputSplits.
- Instantiates the RecordReader to be used for parsing out records from each input split.

InputSplit is a logical representation of a subset of the data to be processed by an individual Mapper. The size of a split is given by the following formula: Max(minimumSize, min(maximumSize, blockSize)). The maximum split size is configurable (mapred.max.split.size) up to a hard bound of the block size. The minimum split size is also configurable (via mapred.min.split.size).

InputSplit is a logical representation of a part of a file, and that split may not (most likely does not) contain an integral number record. In other words, the last record of an input split may be incomplete, as may be the first record of an input split. Processing whole records is the responsibility of the RecordReader.

Input files are split using an implementation of the InputFormat class. Key-value pairs from the input splits are generated for each split using the RecordReader class. The InputKey and InputValue objects are subclasses of the Writable class.

All the key-value pairs from a given input split are sent to the same Mapper. The map() method is called once for each key-value pair, and all the results for each mapper are collected and sent to the partitioner, which breaks out results based on hashed key values. These results are stored in the file system local to the Mapper task where they await pickup by the Hadoop framework running on the Reducer nodes to perform the shuffle and sort phase.

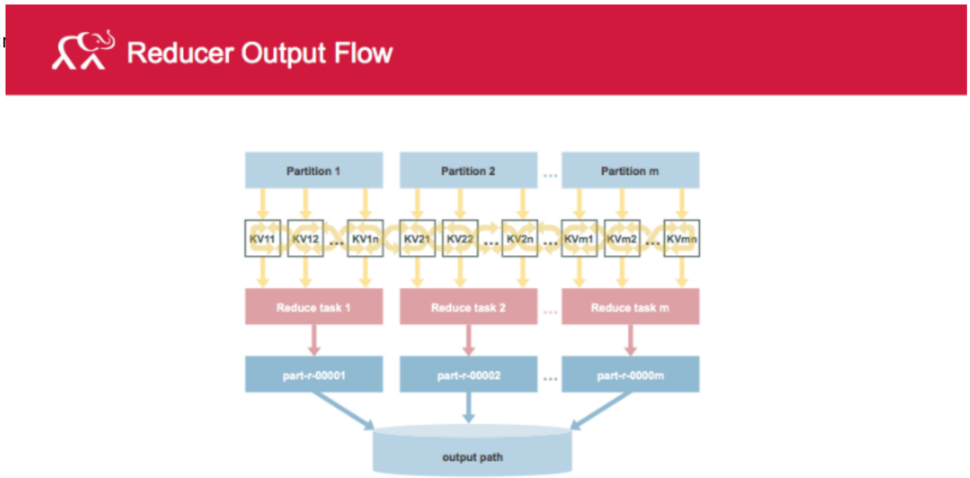
The Record Reader

The record reader breaks up the data in an input split into key-value pairs and presents them to the Mapper assigned to that split.

Records won't necessarily start and end on whole split boundaries (or block boundaries for that matter). It is the job of the RecordReader to ensure that it discards incomplete records at the start of a split and "reads ahead" into the next split to get an entire record that is broken across split boundaries.

LineRecordReader treats keys as offset in file and value as lines up to the newline character. SequenceFileRecordReader reads data as specified by the header of a sequence file. The mapred package includes an implementation for KeyValueLineRecordReader (allows user to define key-value separator).

Reducer Output Data Processing



The Hadoop framework is responsible for taking the partitions from each of the Mappers, shuffling and sorting the results, and presenting a sorted set of key-value pairs as single partitions to each Reducer.

The key (or a subset of the key) is used to derive the partition, typically by a hash function. The total number of partitions is the same as the number of reduce tasks for the job.

Each reducer takes its partition as input, processes one iterable list of key-value pairs at a time, and produces an output file called "part-r-0000x" (where *x* denotes the number of the reducer).

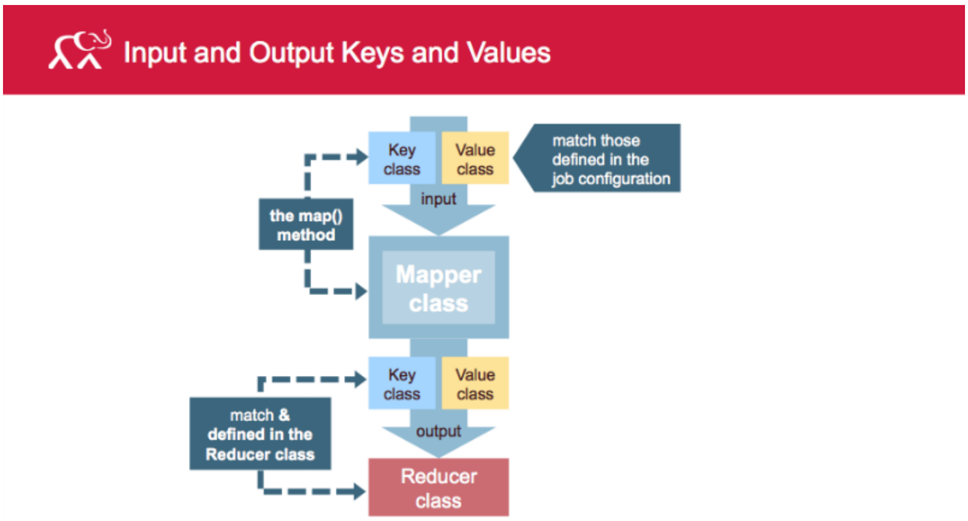
The output directory for this file is specified in the Job configuration and must reside on the HDFS/Mapr-FS file system. The directory cannot both exist and already contain reduce task output files, or the OutputFormat class will generate an IOException. The RecordWriter (encapsulated in the OutputCommitter object) is responsible for writing the Reducer output to the file system.

The OutputCommitter is responsible for:

- Initializing the job at job start (e.g., creating temp directories) in setupJob().
- Cleaning up the job on job completion (e.g., removing temp directories) in cleanupJob().
- Setting up the task temporary output (in setupTask()).
- Checking whether a task needs a commit (in needsTaskCommit()).
- Committing the task output (in commitTask()).
- Discarding the task commit (abortTask()).

Mapper, Reducer, and Job class API

The Mapper class is parameterized with the key and value types for input and output.

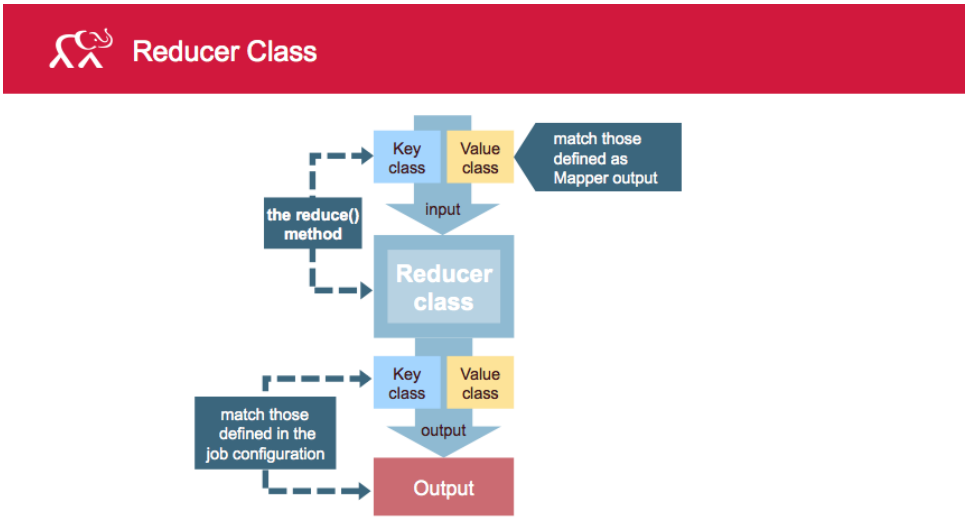


Here are a few rules regarding input and output keys and values for the Mapper class:

- The input key class and input value class in the Mapper class must match those defined in the job configuration.
- The input key class and input value class in the map() method must match those defined in the Mapper class.
- The output key class and output value class in the Mapper must match the input key class and input value class defined in the Reducer class.

The primary method used in the Mapper class is the map() method. The cleanup() method is called when the mapper task finishes (i.e., when all the records in the split have been processed on a mapper). It is the logical counterpart to the setup () method, which is called when the task starts (i.e., before the map() method is called).

The Reducer class is parameterized with the key and value types for input and output.



Here are a few rules regarding input and output keys and values for the Reducer class:

- The input key class and input value class in the Reducer must match the output key class and output value class defined in the Mapper class.
- The output key class and output value class in the Reducer must match those defined in the job configuration. The behavior of the cleanup(), run(), and setup() methods are identical to those described for the Mapper class.

Now that you have a basic understanding of the MapReduce API, including framework functionality, the Mapper and Reducer, Mapper input, the record reader, reducer output data processing, and the Mapper, Reducer and Job class API, I suggest that you dive into some additional training. Be sure and check out our free Hadoop On-Demand Training (/services/mapr-academy/big-data-hadoop-online-training) for full-length courses on a range of Hadoop technologies.

Do you have any comments on this MapReduce API blog post? Add your thoughts in the comments section below.

Search

Categories

All (/blog/)
Apache Drill (/blog/apache-drill/)
Apache Hadoop (/blog/apache-hadoop/)
Machine Learning (/blog/machine-learning/)
NoSQL (/blog/nosql/)
Apache Spark (/blog/apache-spark/)

Partners (/blog/partners/)	Cloud Computing (/blog/cloud-computing/)
Apache Hive (/blog/apache-hive/)	MapReduce (/blog/mapreduce/)
MapR Platform (/blog/mapr-platform/)	Open Source Software (/blog/open-source-software/)
Apache Mesos (/blog/apache-mesos/)	Use Cases (/blog/use-cases/)
Whiteboard Walkthrough Videos (/blog/whiteboard-walkthrough-videos/)	Streaming (/blog/streaming/)
Enterprise Data Hub (/blog/enterprise-data-hub/)	Apache Myriad (/blog/apache-myriad/)

Why MapR?
(/why-mapr/)

Customers (/customers/)

Solutions (/solutions/)

Products (/products/)

Services (/services/)

Training (/training/)

Company

(/company/)

Press (/company/press-releases/) | News (/company/news/)

Leadership (/company/leadership/)

Investors (/company/investors/)

Partners (/partners/)

Careers (/company/careers/)

Awards (/company/awards/)

Contact Us

(/company/contact-mapr/)

Contact Sales

(mailto:sales@mapr.com)

United States: +1 408-914-2390

(tel:4089142390)

Outside the US: +1 855-NOW-MAPR

(tel:8556696277)

Legal

(/legal/)

(https://www.linkedin.com/company/mapr-technologies)

(https://www.facebook.com/maprtech/) (https://twitter.com/mapr)

(https://www.youtube.com/user/maprtech) (/company/contact-mapr/)