Job Input

InputFormat describes the input-specification for a MapReduce job.

The MapReduce framework relies on the `InputFormat` of the job to:

1. Validate the input-specification of the job.
2. Split-up the input file(s) into logical `InputSplit` instances, each of which is then assigned to an individual `Mapper`.
3. Provide the `RecordReader` implementation used to glean input records from the logical `InputSplit` for processing by the `Mapper`.

The default behavior of file-based `InputFormat` implementations, typically sub-classes of FileInputFormat, is to split the input into *logical* `InputSplit` instances based on the total size, in bytes, of the input files. However, the `FileSystem` blocksize of the input files is treated as an upper bound for input splits. A lower bound on the split size can be set via `mapred.min.split.size`.

Clearly, logical splits based on input-size is insufficient for many applications since record boundaries must be respected. In such cases, the application should implement a `RecordReader`, who is responsible for respecting record-boundaries and presents a record-oriented view of the logical `InputSplit` to the individual task.

TextInputFormat is the default `InputFormat`.

If `TextInputFormat` is the `InputFormat` for a given job, the framework detects input-files with the *.gz* extensions and automatically decompresses them using the appropriate `CompressionCodec`. However, it must be noted that compressed files with the above extensions cannot be *split* and each compressed file is processed in its entirety by a single mapper.

InputSplit

InputSplit represents the data to be processed by an individual `Mapper`.

Typically `InputSplit` presents a byte-oriented view of the input, and it is the responsibility of `RecordReader` to process and present a record-oriented view.

FileSplit is the default `InputSplit`. It sets `map.input.file` to the path of the input file for the logical split.

RecordReader

RecordReader reads `<key, value>` pairs from an `InputSplit`.

Typically the `RecordReader` converts the byte-oriented view of the input, provided by the `InputSplit`, and presents a record-oriented to the `Mapper` implementations for processing. `RecordReader`thus assumes the responsibility of processing record boundaries and presents the tasks with keys and values.

Job Output

OutputFormat describes the output-specification for a MapReduce job.

The MapReduce framework relies on the `OutputFormat` of the job to:

1. Validate the output-specification of the job; for example, check that the output directory doesn't already exist.
2. Provide the `RecordWriter` implementation used to write the output files of the job. Output files are stored in a `FileSystem`.

`TextOutputFormat` is the default `OutputFormat`.

OutputCommitter

OutputCommitter describes the commit of task output for a MapReduce job.

The MapReduce framework relies on the `OutputCommitter` of the job to:

1. Setup the job during initialization. For example, create the temporary output directory for the job during the initialization of the job. Job setup is done by a separate task when the job is in PREP state and after initializing tasks. Once the setup task completes, the job will be moved to RUNNING state.
2. Cleanup the job after the job completion. For example, remove the temporary output directory after the job completion. Job cleanup is done by a separate task at the end of the job. Job is declared SUCCEDED/FAILED/KILLED after the cleanup task completes.
3. Setup the task temporary output. Task setup is done as part of the same task, during task initialization.
4. Check whether a task needs a commit. This is to avoid the commit procedure if a task does not need commit.
5. Commit of the task output. Once task is done, the task will commit it's output if required.
6. Discard the task commit. If the task has been failed/killed, the output will be cleaned-up. If task could not cleanup (in exception block), a separate task will be launched with same attempt-id to do the cleanup.

`FileOutputCommitter` is the default `OutputCommitter`. Job setup/cleanup tasks occupy map or reduce slots, whichever is free on the TaskTracker. And JobCleanup task, TaskCleanup tasks and JobSetup task have the highest priority, and in that order.