Command interface

Master to worker

1. StartTask

messageType:command

messageID: start

task: startingTask

1. killTask

messageType:command

messageID: KILL

taskID

3. shutDown

messageType: command

messageID: SHUTDOWN

1. assignId

mesageType: command

messageID:ASSIGNID

workerID

Worker to Master

1. Start response

messageType: response

messageID: START

taskId

Result : SUCCESS|FAIL(SUCCESS:0,FAIL:-1)

CAUSE: STRING

1. killTaskRsp

messageType: response

messiageID: killTaskRsp

int taskID

Result:

Cause:

1. killJobRsp

messageType: response

messageID: killJobRsp

int jobId

Result

Cause

1. heartBeatInd

messageType: Indication

MsgId: HEARTBEATIND

TaskTrackerStatus: taskStatusReport

2. Task interface

Task class{

Jobid,

Jobconf

taskId

taskType: mapper or reducer

}

mapTask and reduceTask will implement this interface. The mapTask and reduceTask will send to taskTracker in the task action message.

3. TaskTrackerStatus

workerId;

   ArrayList<TaskStatus]]>(taskReports);

   maxMapTasks;

   maxReduceTasks

**4. Job Input**

[InputFormat](http://hadoop.apache.org/docs/r1.2.1/api/org/apache/hadoop/mapred/InputFormat.html) 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](http://hadoop.apache.org/docs/r1.2.1/api/org/apache/hadoop/mapred/FileInputFormat.html), 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](http://hadoop.apache.org/docs/r1.2.1/api/org/apache/hadoop/mapred/TextInputFormat.html) 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.

***5. InputSplit***

[InputSplit](http://hadoop.apache.org/docs/r1.2.1/api/org/apache/hadoop/mapred/InputSplit.html) 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](http://hadoop.apache.org/docs/r1.2.1/api/org/apache/hadoop/mapred/FileSplit.html) is the default InputSplit. It sets map.input.file to the path of the input file for the logical split.

***6. RecordReader***

[RecordReader](http://hadoop.apache.org/docs/r1.2.1/api/org/apache/hadoop/mapred/RecordReader.html) 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.