

Processing Data with Map Reduce

Allahbaksh Mohammedali Asadullah

Infosys Labs, Infosys Technologies

2nd IndicThreads.com Conference On
Cloud Computing

3,4 JUNE 2011



PUNE, INDIA

Content

Map Function

Reduce Function

Why Hadoop

HDFS

Map Reduce – Hadoop

Some Questions



What is Map Function

Map is a classic primitive of Functional Programming.

Map means apply a function to a list of elements and **return** the modified **list**.

```
function List Map(Function func, List elements){  
    List newElement;  
    foreach element in elements{  
        newElement.put(apply(func, element))  
    }  
    return newElement
```



Example Map Function

```
function double increaseSalary(double salary){  
    return salary* (1+0.15);  
}
```

```
function double Map(Function increaseSalary, List<Employee>  
employees){  
    List<Employees> newList;  
    foreach employee in employees{  
        Employee tempEmployee = new (  
            newList.add(tempEmployee.income=increaseSalary(  
                tempEmployee.income)  
            )  
        }  
    }  
}
```



Fold or Reduce Function

Fold/Reduce reduces a list of values to one.

Fold means apply a function to a list of elements and **return** a resulting **element**

```
function Element Reduce(Function func, List elements){  
    Element earlierResult;  
    foreach element in elements{  
        func(element, earlierResult)  
    }    return earlierResult;  
}
```



Example Reduce Function

```
function double add(double number1, double number2){  
    return number1 + number2;  
}
```

```
function double Reduce (Function add, List<Employee> employees){  
    double totalAmount=0.0;  
    foreach employee in employees{  
        totalAmount =add(totalAmount,employee.income);  
    }  
    return totalAmount  
}
```



I know Map and Reduce, How do I use it

I will use some library or framework.



Why some framework?

Lazy to write boiler plate code

For modularity

Code reusability



What is best choice



Why Hadoop?





open source



Programming Language Support



C++



Who uses it

krugle

Linked 

facebook

Aol.

twitter 

Bai  du 百度



Adobe

YAHOO!



Strong Community



Image Courtesy <http://goo.gl/15Nu3>



Commercial Support



Hadoop



Hadoop HDFS



Hadoop Distributed File System

Large Distributed File System on
commodity hardware

4k nodes, Thousands of files, Petabytes of
data

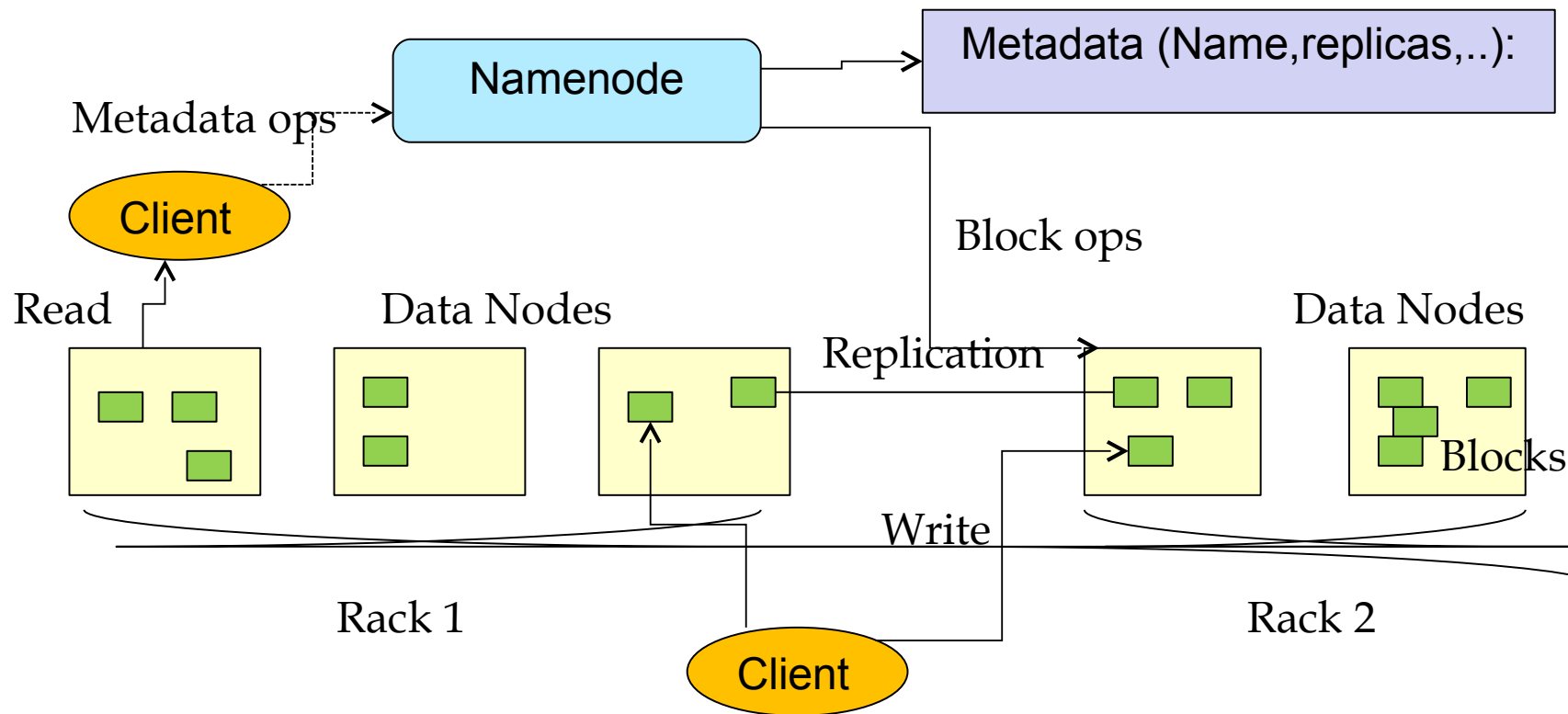
Files are replicated so that hard disk failure
can be handled easily

One NameNode and many DataNode



Hadoop Distributed File System

HDFS ARCHITECTURE



NameNode

Meta-data in RAM

The entire metadata is in main memory.

Metadata consist of

- List of files
- List of Blocks for each file
- List of DataNodes for each block
- File attributes, e.g creation time
- Transaction Log

NameNode uses heartbeats to detect DataNode failure



Data Node

Data Node stores the data in file system

Stores meta-data of a block

Serves data and meta-data to Clients

Pipelining of Data i.e forwards data to other specified DataNodes

DataNodes send heartbeat to the NameNode every three sec.



HDFS Commands

Accessing HDFS

```
hadoop dfs -mkdir myDirectory
```

```
hadoop dfs -cat myFirstFile.txt
```

Web Interface

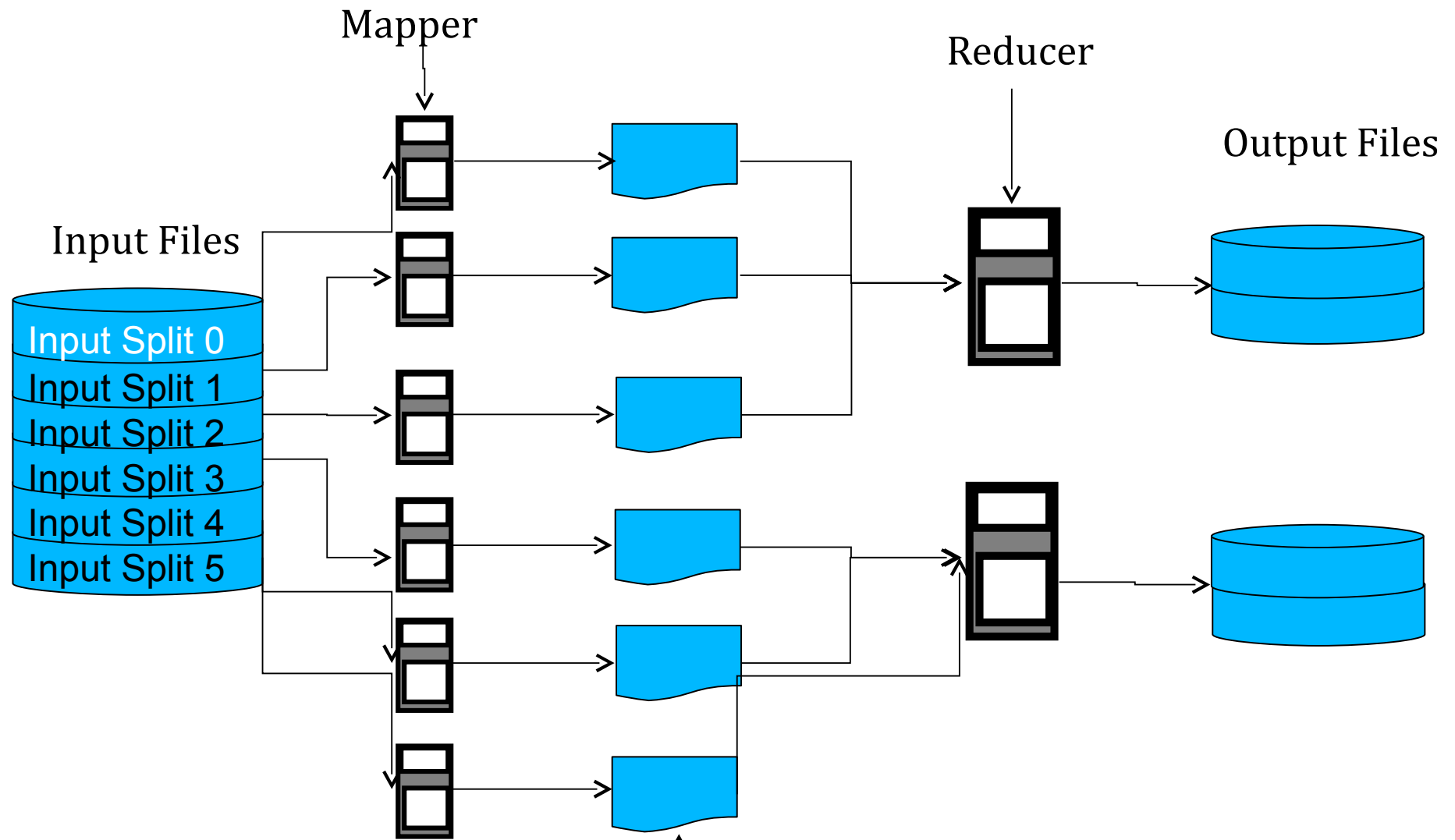
```
http://host:port/dfshealth.jsp
```



Hadoop MapReduce



Map Reduce Diagrammatically



Intermediate file is divided into R partitions, by partitioning function



Input Format

InputFormat describes the input specification to a MapReduce job. That is how the data is to be read from the File System .

Split up the input file into logical InputSplits, each of which is assigned to an Mapper

Provide the RecordReader implementation to be used to collect input record from logical InputSplit for processing by Mapper

RecordReader, typically, converts the byte-oriented view of the input, provided by the InputSplit, and presents a record-oriented view for the Mapper & Reducer tasks for processing. It thus assumes the responsibility of processing record boundaries and presenting the tasks with keys and values.



Creating a your Mapper

The mapper should implements `.mapred.Mapper`

Earlier version use to extend class `.mapreduce.Mapper` class

Extend `.mapred.MapReduceBase` class which provides default implementation of `close` and `configure` method.

The Main method is `map (WritableComparable key, Writable value, OutputCollector<K2,V2> output, Reporter reporter)`

One instance of your Mapper is initialized per task. Exists in separate process from all other instances of Mapper - no data sharing. So static variables will be different for different map task.

Writable-- Hadoop defines a interface called Writable which is Serializable. Examples `IntWritable`, `LongWritable`, `Text` etc.

`WritableComparables` can be compared to each other, typically via `Comparators`. Any type which is to be used as a key in the Hadoop Map-Reduce framework should implement this interface.

`InverseMapper` swaps the key and value



Combiner

Combiners are used to optimize/minimize the number of key value pairs that will be shuffled across the network between mappers and reducers.

Combiner are sort of mini reducer that will be applied potentially several times still during the map phase before to send the new set of key/value pairs to the reducer(s).

Combiners should be used when the function you want to apply is both commutative and associative.

Example: WordCount and Mean value computation

Reference <http://goo.gl/iU5kR>



Partitioner

Partitioner controls the partitioning of the keys of the intermediate map-outputs.

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.

Some Partitioner are BinaryPartitioner, HashPartitioner, KeyFieldBasedPartitioner, TotalOrderPartitioner



Creating a your Reducer

The mapper should implements `.mapred.Reducer`

Earlier version use to extend class `.mapreduce.Reduces` class

Extend `.mapred.MapReduceBase` class which provides default implementation of `close` and `configure` method.

The Main method is `reduce(WritableComparable key, Iterator values, OutputCollector output, Reporter reporter)`

Keys & values sent to one partition all goes to the same reduce task

`Iterator.next()` always returns the same object, different data

`HashPartioner` partition it based on Hash function written

`IdentityReducer` is default implementation of the Reducer



Output Format

OutputFormat is similar to InputFormat

Different type of output formats are

TextOutputFormat

SequenceFileOutputFormat

NullOutputFormat



Mechanics of whole process

Configure the Input and Output

Configure the Mapper and Reducer

Specify other parameters like number Map job, number of reduce job etc.

Submit the job to client

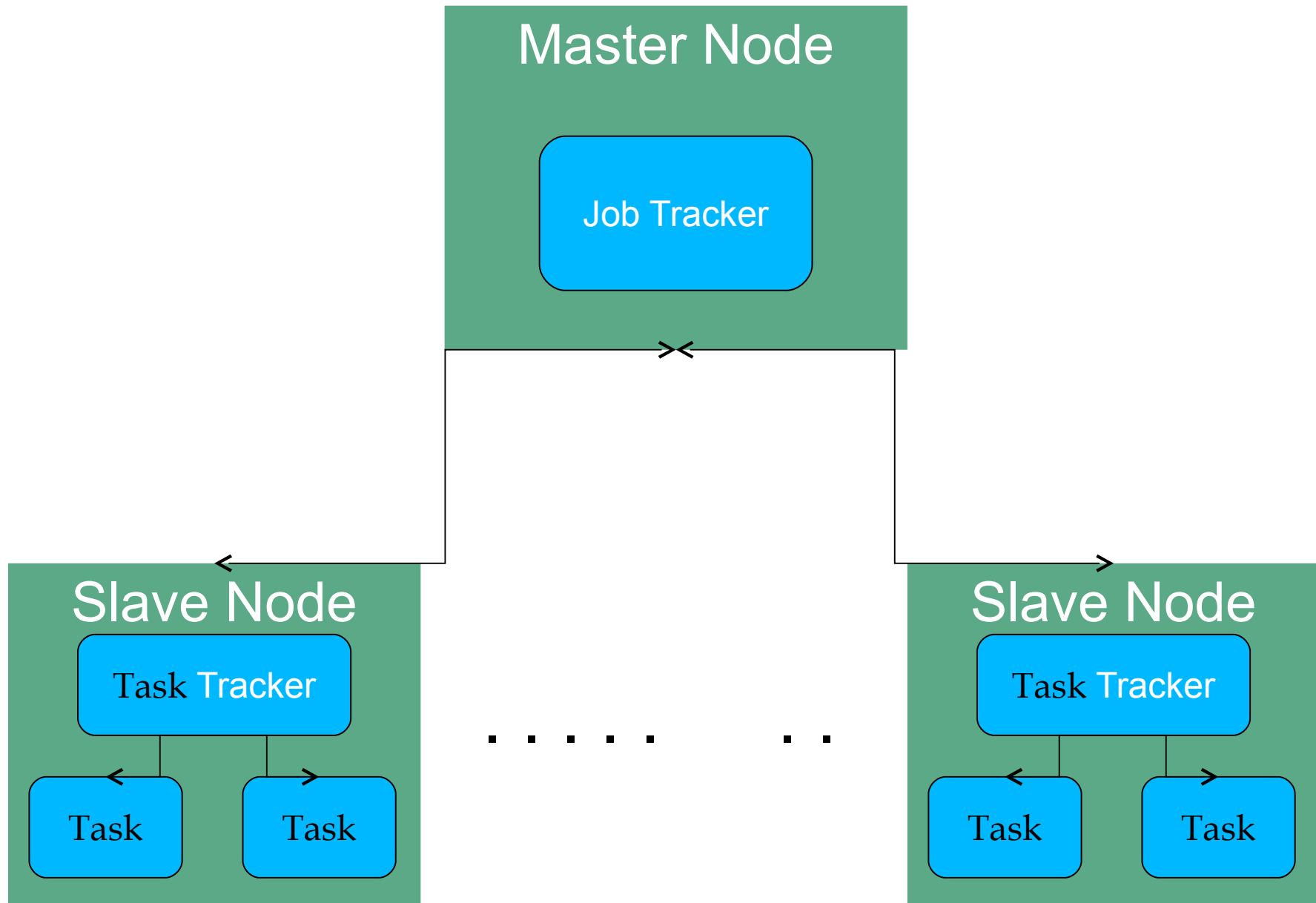


Example

```
JobConf conf = new JobConf(WordCount.class);  
conf.setJobName("wordcount");  
conf.setMapperClass(Map.class);  
conf.setCombinerClass(Reduce.class);  
conf.setReducerClass(Reduce.class);  
conf.setOutputKeyClass(Text.class);  
conf.setOutputValueClass(IntWritable.class);  
conf.setInputFormat(TextInputFormat.class);  
conf.setOutputFormat(TextOutputFormat.class);  
FileInputFormat.setInputPaths(conf, new  
    Path(args[0]));FileOutputFormat.setOutputPath(conf, new Path(args[1]));  
JobClient.runJob(conf); //JobClient.submit
```



Job Tracker & Task Tracker



Job Launch Process

JobClient determines proper division of input into InputSplits

Sends job data to master JobTracker server.

Saves the jar and JobConf (serialized to XML) in shared location and posts the job into a queue.



Job Launch Process Contd..

TaskTrackers running on slave nodes periodically query JobTracker for work.

Get the job jar from the Master node to the data node.

Launch the main class in separate JVM queue.

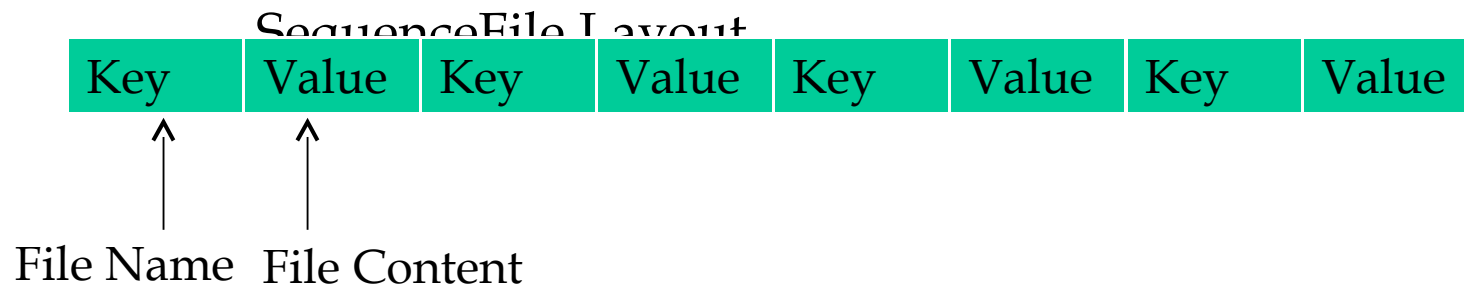
`TaskTracker.Child.main()`



Small File Problem

What should I do if I have lots of small files?

One word answer is SequenceFile.



Tar to SequenceFile <http://goo.gl/mKGC7>

 Consolidator <http://goo.gl/EVvi7>

Problem of Large File

What if I have single big file of 20Gb?

One word answer is There is no problems with large files



SQL Data

What is way to access SQL data?

One word answer is DBInputFormat.

DBInputFormat provides a simple method of scanning entire tables from a database, as well as the means to read from arbitrary SQL queries performed against the database.

DBInputFormat provides a simple method of scanning entire tables from a database, as well as the means to read from arbitrary SQL queries performed against the database.

Database Access with Hadoop <http://goo.gl/CNOBc>



```
JobConf conf = new JobConf(getConf(), MyDriver.class);

conf.setInputFormat(DBInputFormat.class);

DBConfiguration.configureDB(conf, "com.mysql.jdbc.Driver",
"jdbc:mysql://localhost:port/dbName");

String [] fields = { "employee_id", "name" };

DBInputFormat.setInput(conf, MyRow.class, "employees", null /* conditions */, "employee_id",
fields);
```



```

public class MyRow implements Writable, DBWritable {

    private int employeeNumber;

    private String employeeName;

    public void write(DataOutput out) throws IOException {

        out.writeInt(employeeNumber);           out.writeChars(employeeName);

    }

    public void readFields(DataInput in) throws IOException {

        employeeNumber= in.readInt();           employeeName = in.readUTF();

    }

    public void write(PreparedStatement statement) throws SQLException {

        statement.setInt(1, employeeNumber);  statement.setString(2, employeeName);

    }

    public void readFields(ResultSet resultSet) throws SQLException {

        employeeNumber = resultSet.getInt(1); employeeName = resultSet.getString (2);

    }
}

```



Question & Answer



Thanks You

