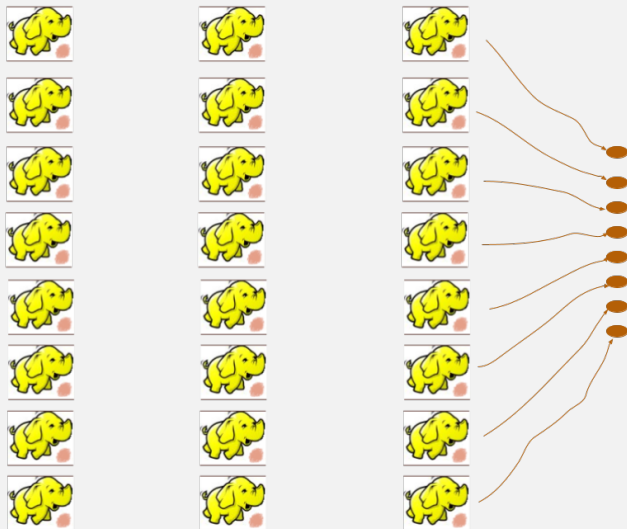


MapReduce on Apache Hadoop (using Java)

Amit Jain



Hadoop Map-Reduce Inputs and Outputs

- ▶ The Map/Reduce framework operates exclusively on $\langle \text{key}, \text{value} \rangle$ pairs, that is, the framework views the input to the job as a set of $\langle \text{key}, \text{value} \rangle$ pairs and produces a set of $\langle \text{key}, \text{value} \rangle$ pairs as the output of the job, but the key and value may be of different types.
- ▶ The key and value classes have to be serializable by the framework and hence need to implement the `Writable` interface. Additionally, the key classes have to implement the `WritableComparable` interface to facilitate sorting by the framework.
- ▶ The user needs to implement a `Mapper` class as well as a `Reducer` class. Optionally, the user can also write a `Combiner` class.

(input) $\langle k1, v1 \rangle \rightarrow \text{map} \rightarrow \langle k2, v2 \rangle \rightarrow \text{combine} \rightarrow \langle k2, v2 \rangle$
 $\rightarrow \text{reduce} \rightarrow \langle k3, v3 \rangle$ (output)

MapReduce API

```
public class MyMapper extends
    Mapper<KEYIN, VALUEIN, KEYOUT, VALUEOUT> { ... }

protected void map(KEYIN key,
                    VALUEIN value,
                    Mapper.Context context)
    throws IOException,
           InterruptedException

public class MyReducer extends
    Reducer<KEYIN, VALUEIN, KEYOUT, VALUEOUT> { ... }

protected void reduce(KEYIN key,
                      Iterable<VALUEIN> values,
                      Reducer.Context context)
    throws IOException,
           InterruptedException
```

WordCount example with Hadoop API

Problem: To count the number of occurrences of each word in a large collection of documents.

```
/**
 * Counts the words in each line.
 * For each line of input, break the line into words
 * and emit them as (word, 1).
 */
public static class TokenizerMapper
    extends Mapper<Object, Text, Text, IntWritable>{

    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    public void map(Object key, Text value, Context context)
        throws IOException, InterruptedException {
        StringTokenizer itr =
            new StringTokenizer(value.toString());
        while (itr.hasMoreTokens()) {
            word.set(itr.nextToken());
            context.write(word, one);
        }
    }
}
```

See full code here: [WordCount.java](#)

WordCount Example with Hadoop API (contd.)

```
/**
 * A reducer class that just emits the sum of the input
 * values.
 */
public static class IntSumReducer
    extends Reducer<Text,IntWritable,Text,IntWritable> {
    private IntWritable result = new IntWritable();

    public void reduce(Text key, Iterable<IntWritable> values,
                       Context context)
        throws IOException, InterruptedException
    {
        int sum = 0;
        for (IntWritable val : values) {
            sum += val.get();
        }
        result.set(sum);
        context.write(key, result);
    }
}
```

WordCount Example with Hadoop API (contd.)

```
public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "word count");
    job.setNumReduceTasks(8); // we can control it

    job.setJarByClass(WordCount.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.class);

    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);

    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));

    System.exit(job.waitForCompletion(true) ? 0 : 1);
}
```

Case Analysis Example

See full code here: [CaseAnalysis.java](#)

```
public class CaseAnalysis {
    public static class Map extends Mapper<LongWritable, Text, Text,
        IntWritable> {
        private final static IntWritable one = new IntWritable(1);
        private final static IntWritable zero = new IntWritable(0);
        private Text word = new Text();

        public void map(LongWritable key, Text value,
            Context context)
            throws IOException, InterruptedException
        {
            String line = value.toString();

            for (int i = 0; i < line.length(); i++) {
                if (Character.isLowerCase(line.charAt(i))) {
                    word.set(String.valueOf(line.charAt(i)).toUpperCase());
                    context.write(word, zero);
                } else if (Character.isUpperCase(line.charAt(i))) {
                    word.set(String.valueOf(line.charAt(i)));
                    context.write(word, one);
                } else {
                    word.set("other");
                    context.write(word, one);
                }
            }
        }
    }
}
```

Case Analysis Example (contd.)

```
public static class Reduce
    extends Reducer<Text, IntWritable, Text, Text> {
    private Text result = new Text();

    public void reduce(Text key, Iterable<IntWritable> values,
        Context context) throws IOException, InterruptedException
    {
        long total = 0;
        int upper = 0;

        for (IntWritable val: values) {
            upper += val.get();
            total++;
        }
        result.set(String.format("%16d %16d %16d %16.2f", total, upper,
            (total - upper), ((double) upper / total)));
        context.write(key, result);
    }
}
```


Case Analysis Example (contd.)

```
public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    String[] otherArgs = new GenericOptionsParser(conf, args).
        getRemainingArgs();

    if (otherArgs.length != 2) {
        System.err.println("Usage: hadoop jar caseanalysis.jar <in>
        <out>");
        System.exit(2);
    }

    Job job = Job.getInstance(conf, "case analysis");
    job.setJarByClass(CaseAnalysis.class);
    job.setMapperClass(Map.class);
    job.setReducerClass(Reduce.class);

    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(Text.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);

    FileInputFormat.addInputPath(job, new Path(otherArgs[0]));
    FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));

    System.exit(job.waitForCompletion(true) ? 0 : 1);
}
```

Inverted Index Example

Given an input text, an inverted index program uses MapReduce to produce an index of all the words in the text. For each word, the index has a list of all the files where the word appears. See full code here: [InvertedIndex.java](#)

```
public static class InvertedIndexMapper extends
    Mapper<LongWritable, Text, Text, Text>
{
    private final static Text word = new Text();
    private final static Text location = new Text();

    public void map(LongWritable key, Text val, Context context)
        throws IOException, InterruptedException
    {
        FileSplit fileSplit = (FileSplit) context.getInputSplit();
        String fileName = fileSplit.getPath().getName();
        location.set(fileName);

        String line = val.toString();
        StringTokenizer itr = new StringTokenizer(line.toLowerCase());
        while (itr.hasMoreTokens()) {
            word.set(itr.nextToken());
            context.write(word, location);
        }
    }
}
```

Inverted Index Example (contd.)

The reduce method is shown below.

```
public static class InvertedIndexReducer extends
    Reducer<Text, Text, Text, Text>
{
    public void reduce(Text key, Iterable<Text> values, Context
context)
    throws IOException, InterruptedException
    {
        boolean first = true;
        StringBuilder toReturn = new StringBuilder();
        while (values.hasNext()) {
            if (!first)
                toReturn.append(", ");
            first = false;
            toReturn.append(values.next().toString());
        }
        context.write(key, new Text(toReturn.toString()));
    }
}
```

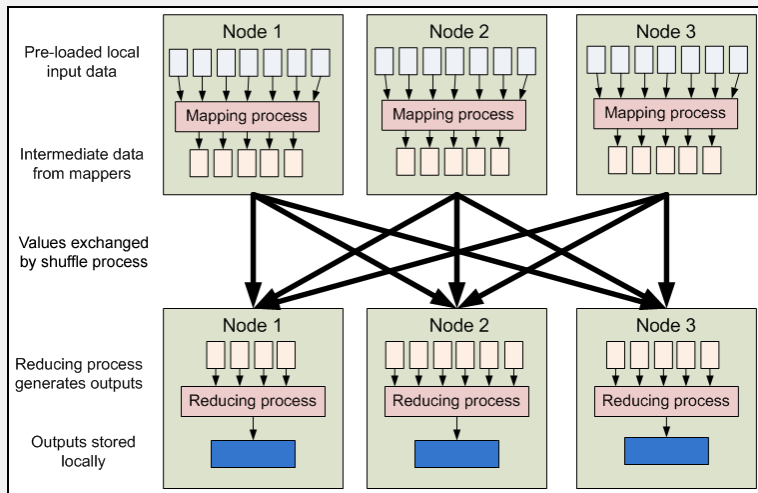
Inverted Index Example (contd)

```
public static void main(String[] args) throws IOException
{
    Configuration conf = new Configuration();
    String[] otherArgs = new GenericOptionsParser(conf,
                                                    args).getRemainingArgs();

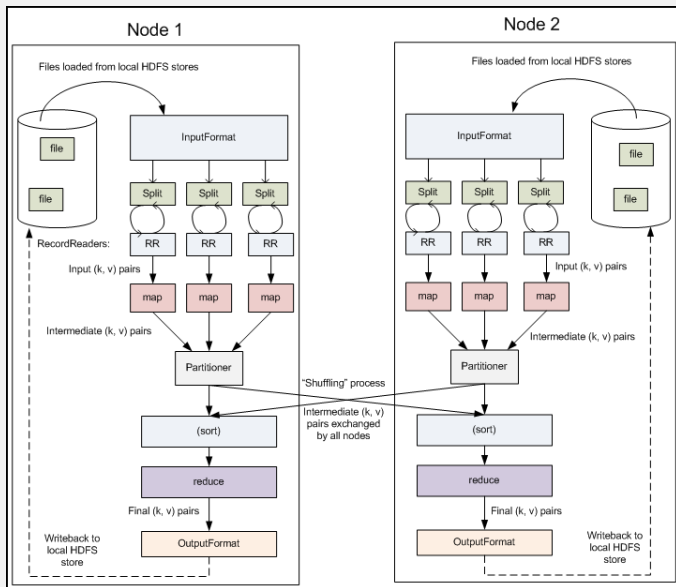
    if (args.length < 2) {
        System.out
            println("Usage: InvertedIndex <input path> <output path>");
        System.exit(1);
    }
    Job job = Job.getInstance(conf, "InvertedIndex");
    job.setJarByClass(InvertedIndex.class);
    job.setMapperClass(InvertedIndexMapper.class);
    job.setReducerClass(InvertedIndexReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(Text.class);

    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
}
```

MapReduce: High-Level Data Flow



MapReduce: Detailed Data Flow



Top-N Example

- ▶ Given a list of movies with the number of views, find the top 10 movies by number of views (assume that the number of views is unique)
- ▶ See example code: [hadoop/top-n-movies-v1](#)
- ▶ Illustrates the **setup/cleanup technique** where we can take only one action for a map and reduce.
- ▶ What if we have multiple movies with the same number of views? See the example below for a sample solution.
- ▶ See example code: [hadoop/top-n-movies-v2](#)

- ▶ Modify WordCount to not set number of reducers or change it to something different. Generate the jar file again and test to see if it does what you expected.
- ▶ Modify Case Analysis to skip the other non characters. Generate the jar file and run it to test it.

References

- ▶ Overall documentation for Hadoop 3.3.6:
<https://hadoop.apache.org/docs/r3.3.6/>
- ▶ MapReduce API (Hadoop 3.3.6):
<https://hadoop.apache.org/docs/stable/api>
- ▶ MapReduce Tutorial (for Java)