Introduction To Hadoop

Kenneth Heafield

Google Inc

January 14, 2008

Outline

Mapper

$< \text{“wikipedia.org”}, \text{“The Free”} > \rightarrow < \text{“The”}, 1 >, < \text{“Free”}, 1 >$

```
public void map(WritableComparable key,
    Writable value, OutputCollector output,
    Reporter reporter) throws IOException {
  String line = ((Text)value).toString();
  StringTokenizer itr = new StringTokenizer(line);
  Text word = new Text();
  while (itr.hasMoreTokens()) {
    word.set(itr.nextToken());
    output.collect(word, new IntWritable(1));
  }
}
```

Mapper

$<$ "wikipedia.org", "The Free" $> \rightarrow\ <$ "The", 1 $>, <$ "Free", 1 $>$

```
public void map(WritableComparable key,
    Writable value, OutputCollector output,
    Reporter reporter) throws IOException {
  String line = ((Text)value).toString();
  StringTokenizer itr = new StringTokenizer(line);
  Text word = new Text();
  while (itr.hasMoreTokens()) {
    word.set(itr.nextToken());
    output.collect(word, new IntWritable(1));
  }
}
```

Mapper

$<$ "wikipedia.org", "The Free" $> \rightarrow <$ "The", $1 >, <$ "Free", $1 >$

```
public void map(WritableComparable key,
    Writable value, OutputCollector output,
    Reporter reporter) throws IOException {
  String line = ((Text)value).toString();
  StringTokenizer itr = new StringTokenizer(line);
  Text word = new Text();
  while (itr.hasMoreTokens()) {
    word.set(itr.nextToken());
    output.collect(word, new IntWritable(1));
  }
}
```

Mapper

$< \text{“wikipedia.org”}, \text{“The Free”} > \rightarrow < \text{“The”}, 1 >, < \text{“Free”}, 1 >$

```
public void map(WritableComparable key,
    Writable value, OutputCollector output,
    Reporter reporter) throws IOException {
  String line = ((Text)value).toString();
  StringTokenizer itr = new StringTokenizer(line);
  Text word = new Text();
  while (itr.hasMoreTokens()) {
    word.set(itr.nextToken());
    output.collect(word, new IntWritable(1));
  }
}
```

Mapper

$< \text{“wikipedia.org”}, \text{“The Free”} > \rightarrow < \text{“The”}, 1 >, < \text{“Free”}, 1 >$

```
public void map(WritableComparable key,
    Writable value, OutputCollector output,
    Reporter reporter) throws IOException {
  String line = ((Text)value).toString();
  StringTokenizer itr = new StringTokenizer(line);
  Text word = new Text();
  while (itr.hasMoreTokens()) {
    word.set(itr.nextToken());
    output.collect(word, new IntWritable(1));
  }
}
```

Reducer

$< \text{“The”}, 1 >, < \text{“The”}, 1 > \rightarrow < \text{“The”}, 2 >$

```
public void reduce(WritableComparable key,
                   Iterator values,
                   OutputCollector output,
                   Reporter reporter)
                   throws IOException {
  int sum = 0;
  while (values.hasNext()) {
    sum += ((IntWritable) values.next()).get();
  }
  output.collect(key, new IntWritable(sum));
}
```

Reducer

$< \text{“The”}, 1 >, < \text{“The”}, 1 > \rightarrow\ < \text{“The”}, 2 >$

```
public void reduce(WritableComparable key,
                   Iterator values,
                   OutputCollector output,
                   Reporter reporter)
                   throws IOException {
  int sum = 0;
  while (values.hasNext()) {
    sum += ((IntWritable) values.next()).get();
  }
  output.collect(key, new IntWritable(sum));
}
```

Reducer

$< \text{“The”}, 1 >, < \text{“The”}, 1 > \rightarrow\ < \text{“The”}, 2 >$

```
public void reduce(WritableComparable key,
                   Iterator values,
                   OutputCollector output,
                   Reporter reporter)
                   throws IOException {
  int sum = 0;
  while (values.hasNext()) {
    sum += ((IntWritable) values.next()).get();
  }
  output.collect(key, new IntWritable(sum));
}
```

Reducer

$<$ "The", $1 >, <$ "The", $1 > \rightarrow\ <$ "The", $2 >$

```
public void reduce(WritableComparable key,
                   Iterator values,
                   OutputCollector output,
                   Reporter reporter)
                   throws IOException {
  int sum = 0;
  while (values.hasNext()) {
    sum += ((IntWritable) values.next()).get();
  }
  output.collect(key, new IntWritable(sum));
}
```

Main

```
public static void main(String[] args)
    throws IOException {
  JobConf conf = new JobConf(WordCount.class);
  conf.setJobName("wordcount");
  conf.setMapperClass(MapClass.class);
  conf.setCombinerClass(ReduceClass.class);
  conf.setReducerClass(ReduceClass.class);
  conf.setNumMapTasks(new Integer(40));
  conf.setNumReduceTasks(new Integer(30));
  conf.setInputPath(new Path("/shared/wikipedia_small"));
  conf.setOutputPath(new Path("/user/kheafield/word_count"));
  conf.setOutputKeyClass(Text.class);
  conf.setOutputValueClass(IntWritable.class);
  JobClient.runJob(conf);
}
```

Main

```
public static void main(String[] args)
    throws IOException {
  JobConf conf = new JobConf(WordCount.class);
  conf.setJobName("wordcount");
  conf.setMapperClass(MapClass.class);
  conf.setCombinerClass(ReduceClass.class);
  conf.setReducerClass(ReduceClass.class);
  conf.setNumMapTasks(new Integer(40));
  conf.setNumReduceTasks(new Integer(30));
  conf.setInputPath(new Path("/shared/wikipedia_small"));
  conf.setOutputPath(new Path("/user/kheafield/word_count"));
  conf.setOutputKeyClass(Text.class);
  conf.setOutputValueClass(IntWritable.class);
  JobClient.runJob(conf);
}
```

Main

```
public static void main(String[] args)
    throws IOException {
  JobConf conf = new JobConf(WordCount.class);
  conf.setJobName("wordcount");
  conf.setMapperClass(MapClass.class);
  conf.setCombinerClass(ReduceClass.class);
  conf.setReducerClass(ReduceClass.class);
  conf.setNumMapTasks(new Integer(40));
  conf.setNumReduceTasks(new Integer(30));
  conf.setInputPath(new Path("/shared/wikipedia_small"));
  conf.setOutputPath(new Path("/user/kheafield/word_count"));
  conf.setOutputKeyClass(Text.class);
  conf.setOutputValueClass(IntWritable.class);
  JobClient.runJob(conf);
}
```

Main

```
public static void main(String[] args)
    throws IOException {
  JobConf conf = new JobConf(WordCount.class);
  conf.setJobName("wordcount");
  conf.setMapperClass(MapClass.class);
  conf.setCombinerClass(ReduceClass.class);
  conf.setReducerClass(ReduceClass.class);
  conf.setNumMapTasks(new Integer(40));
  conf.setNumReduceTasks(new Integer(30));
  conf.setInputPath(new Path("/shared/wikipedia_small"));
  conf.setOutputPath(new Path("/user/kheafield/word_count"));
  conf.setOutputKeyClass(Text.class);
  conf.setOutputValueClass(IntWritable.class);
  JobClient.runJob(conf);
}
```

Types

Purpose

Simple serialization for keys, values, and other data

Interface Writable

- Read and write binary format
- Convert to `String` for text formats
- `WritableComparable` adds sorting order for keys

Example Implementations

- `ArrayWritable` is only `Writable`
- `BooleanWritable`
- `IntWritable` sorts in increasing order
- `Text` holds a `String`

A Writable

```
public class IntPairWritable implements Writable {
  public int first;
  public int second;
  public void write(DataOutput out) throws IOException {
    out.writeInt(first);
    out.writeInt(second);
  }
  public void readFields(DataInput in) throws IOException {
    first = in.readInt();
    second = in.readInt();
  }
  public int hashCode() { return first + second; }
  public String toString() {
    return Integer.toString(first) + "," +
           Integer.toString(second);
  }
```

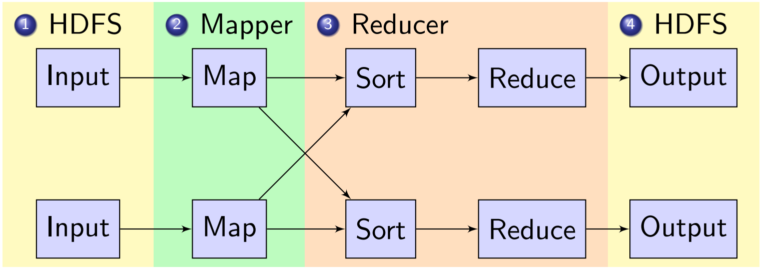
WritableComparable Method

```
public int compareTo(Object other) {
  IntPairWritable o = (IntPairWritable)other;
  if (first < o.first) return -1;
  if (first > o.first) return 1;
  if (second < o.second) return -1;
  if (second > o.second) return 1;
  return 0;
}
```

Data Flow

Default Flow

1. Mappers read from HDFS
2. Map output is partitioned by key and sent to Reducers
3. Reducers sort input by key
4. Reduce output is written to HDFS



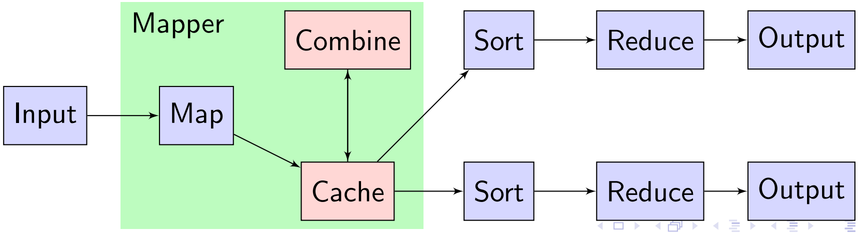
Combiners

Concept

- Add counts at Mapper before sending to Reducer.
- Word count is 6 minutes with combiners and 14 without.

Implementation

- Mapper caches output and periodically calls Combiner
- Input to Combine may be from Map or Combine
- Combiner uses interface as Reducer



Exercises

Recommended: Word Count

Get word count running.

Bigrams

Count bigrams and unigrams efficiently.

Capitalization

With what probability is a word capitalized?

Indexer

In what documents does each word appear? Where in the documents?

Instructions

1. Login to the cluster successfully (and set your password).
2. Get Eclipse installed, so you can build Java code.
3. Install the Hadoop plugin for Eclipse so you can deploy jobs to the cluster.
4. Set up your Eclipse workspace from a template that we provide.
5. Run the word counter example over the Wikipedia data set.