Distributed Systems
11r. Map-Reduce Programming on AWS/EMR (Part II)

What is Map-Reduce
- Origin from Google
- A simple programming model
- Functional model
- For large-scale data processing
  - Exploits large set of commodity computers
  - Executes process in distributed manner
  - Offers high availability

Motivation
- Lots of demands for very large scale data processing
- A certain common themes for these demands
  - Lots of machines needed (scaling)
  - Two basic operations on the input
    - Map
    - Reduce

Applications
- String Match, such as Grep
- Reverse index
- Count URL access frequency
- Log analysis
- Lots of examples in data mining

Map-Reduce Implementations
- Cluster
  1. Google
  2. Apache Hadoop
- Multicore CPU
- Stanford
- GPU
- HKUST
Map-Reduce in Hadoop

- Map-Reduce Job:
  - It works by breaking the processing into two phases: the map phase and the reduce phase.
  - Each phase has key-value pairs as input and output, the types of which may be chosen by the programmer.

- Mapper
  - A Mapper usually processes data in single lines. Ignore the useless lines and collect useful information from data into <Key, Value> pairs.

- Reducer
  - Receive the <Key, <Value1, Value2, ...>> pairs from Mappers. Tabulate statistics data and write the results into <Key, Value> pairs.

---

Map-Reduce: Architecture

---

Map-Reduce: API Model

---

Map-Reduce: API Model v2.0

---

Map-Reduce: Data Flow

---

Serialization in Hadoop
Example: WordCount

WordCount is a simple application that counts the number of occurrences of each word in a given input set.

```java
public static class Map extends MapReduceBase implements Mapper<LongWritable, Text, Text, IntWritable> {
    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();
    public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable> output, Reporter reporter) throws IOException {
        String line = value.toString();
        StringTokenizer tokenizer = new StringTokenizer(line);
        while (tokenizer.hasMoreTokens()) {
            word.set(tokenizer.nextToken);
            output.collect(word, one);
        }
    }
}
```

```java
public static class Reduce extends MapReduceBase implements Reducer<Text, IntWritable, Text, IntWritable> {
    public void reduce(Text key, Iterator<IntWritable> values, OutputCollector<Text, IntWritable> output, Reporter reporter) throws IOException {
        int sum = 0;
        while (values.hasNext()) {
            sum += values.next().get();
        }
        output.collect(key, new IntWritable(sum));
    }
}
```

Map-Reduce Job Configuration

- Before running a MapReduce job, the following fields should be set:
  - Mapper Class
    - The mapper class written by yourself to be run.
  - Reducer Class
    - The reducer class written by yourself to be run.
  - Input Format & Output Format
    - Define the format of all input and outputs. A large number of formats are supported in Hadoop Library.
  - OutputKeyClass & OutputValueClass
    - The data type class of the outputs that Mappers send to Reducers.

Example: WordCount

- Code to run the job

```java
public class WordCount {
    public static void main(String[] args) throws Exception {
        JobConf conf = new JobConf(WordCount.class);
        conf.setJobName("wordcount");
        conf.setOutputKeyClass(Text.class);
        conf.setOutputValueClass(IntWritable.class);
        conf.setMapperClass(Map.class);
        conf.setReducerClass(Reduce.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);
    }
}
```

Perf-Log: A Simple Map-Reduce Case

Architecture of Perf-Log Project
Perf-Log Format

• Event Level

• Request Level

Example Using Map-Reduce

• Here we use a MapReduce job to calculate the most used event everyday. All the event records are collected in Map and the most used events are counted in Reduce.

Map
Shuffle(auto)
Reduce

Map-Reduce with HBase

• Here is an example:

```
static class HBaseMapper<K, V> extends MapReduceBase
  implements Mapper<LongWritable, Text, K, V> {
  private HTable table;

  @Override
  public void configure(JobConf jc) {
    super.configure(jc);
    try {
      this.table = new HTable(HBaseConfiguration.create(), ”table_name”);
    } catch (IOException e) {
      throw new RuntimeException(”Failed HTable construction”, e);
    }
  }

  @Override
  public void close() throws IOException {
    super.close();
    table.close();
  }

  public void map(LongWritable key, Text value, OutputCollector<K, V> output, Reporter reporter)
  throws IOException {
    Put p = new Put();
    // Set your own put.
    table.put(p);
  }
}
```