Distributed Systems

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

Paul Krzyzanowski
TA: Long Zhao
Rutgers University
Fall 2017
Hadoop Map-Reduce
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

MapReduce

Cluster,
1. Google
2. Apache Hadoop

Multicore CPU,
Phoenix @ stanford

GPU,
Mars@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 process 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

- IntWritable
- LongWritable
- BooleanWritable
- ByteWritable
- Text
- BytesWritable
- MD5Hash
- ObjectWritable
- GenericWritable
- FloatWritable
- DoubleWritable
- String
- Text
- NullWritable
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);
        }
    }
}

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
Reports and Charts
Perf-Log Format

**Event Level**

```
2011-08-15 20:13:36,936 TIMESTAMP=131346416936 | CLIENT_IP=10.50.9.1 COMPANY=PMPerf USER=718066811 DBPOOL=dbPool11 LEVEL=EVENT
EVENT_ID=d50525def0d4e1f0b2ada1f86c9e9ea7 EVENT_NAME=GM_GOALS_TAB RENDER_TIME=47 SERVER_TIME=160645949 TOTAL_TIME=323 JS_NUM=55 CSS_NUM=12
```

**Request Level**

```
2011-08-15 20:13:36,774 CLIENT_IP=10.50.9.1 COMPANY=PMPerf USER=718066811 DBPOOL=dbPool11 LEVEL=REQ EVENT_ID=d50525def0d4e1f0b2ada1f86c9e9ea7
EVENT_NAME=GM_GOALS_TAB_REQ_ID=15161 URL=/sf/goals SERVER_TIME=160 PARAMETER_NAME=[sqlite_company=PMPerf&_s_crb=SRqBp15VYDFlEoRPOuc9dsNWJ1%3d]
CALLSTACK={"name":"/sf/goals","invocations":1,"totalNanos":640487},{"name":"GlobalSysConfigHome.create","invocations":1,"totalNanos":77300},
{name="UserHome.findUsersForManager","invocations":1,"totalNanos":46206513,"children":
[{"name":"PreferenceHome.getPreferenceBean","invocations":1,"totalNanos":45734672,"children":
[{"name":"UserHome.findUsersForManager","invocations":1,"totalNanos":359645}]
]}},
{name="SysConfigHome.create","invocations":2,"totalNanos":100321},
{name="ObjectiveHome.canUserAccessObjTemplate","invocations":2,"totalNanos":846692},
{name="ObjectiveHome.getPermittedActiveTemplates","invocations":1,"totalNanos":568277},
{name="UserHome.findUserMtrixManager","invocations":1,"totalNanos":712994,"children":
[{"name":"UserHome.findUserName","invocations":1,"totalNanos":355801}],
{name="UserHome.findMatrixReport","invocations":1,"totalNanos":1168255,"children":
[{"name":"UserHome.findMatrixReport","invocations":1,"totalNanos":808920,"children":
[{"name":"UserHome.findUserMtrixManager","invocations":1,"totalNanos":785091}]
}]},
{name="GlobalSysConfigHome.getConfigGlobalDowntimeNotificationParameters","invocations":1,"totalNanos":872239},
{name="ObjectiveHome.getUserDefaultTemplateId","invocations":1,"totalNanos":331770},
{name="UserHome.findUserById","invocations":5,"totalNanos":27756607,"children":
[{"name":"UserHome.create","invocations":2,"totalNanos":150006},{"name":"SysConfigHome.getSysConfigList","invocations":1,"totalNanos":380324}],
{name="ObjectiveHome.getObjListByPlan","invocations":1,"totalNanos":5167794,"children":
[{"name":"UserHome.findUserById","invocations":1,"totalNanos":387348,"children":
[{"name":"UserHome.create","invocations":1,"totalNanos":83692}]
}]
},
{name="SuccessionHome.create","invocations":2,"totalNanos":1374992},
{name="ServiceCommandAndProcessorHome.execute","invocations":4,"totalNanos":16875294},
{name="LearningActivityOpesHome.create","invocations":1,"totalNanos":675411},{"name":"SysConfigHome.getSysConfig","invocations":33,"totalNanos":11624952},
{name="PreferenceHome.getPreferenceBeans","invocations":1,"totalNanos":775122},
{name="UserHome.findUserRelationships","invocations":7,"totalNanos":256792},
{name="PreferenceHome.create","invocations":2,"totalNanos":127257}]
```
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.

<table>
<thead>
<tr>
<th>Map</th>
<th>Shuffle(auto)</th>
<th>Reduce</th>
</tr>
</thead>
<tbody>
<tr>
<td>event PLT_LOGIN request record… request record… event PM_HOME request record… event PM_OPENFORM request record… request record… event CDP_LOGOUT request record… request record… request record…</td>
<td>(11/12, PLT_LOGIN) (11/12, PM_HOME) (11/12, PLT_LOGIN) (11/12, PM_LOGOUT) . . (11/13, CDP_LOGIN) (11/13, CDP_LOGIN) . .</td>
<td>(11/12, [PLT_LOGIN, PM_HOME, PLT_LOGIN, PLT_LOGOUT…]) (11/13, [CDP_LOGIN, CDP_LOGIN…]) . . .</td>
</tr>
</tbody>
</table>
Map-Reduce with HBase

• Here is an example:

```java
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);
    }
}
```
The end