Concepts and Models of Parallel and Data-centric Programming

MapReduce – Hadoop Ecosystem

Lecture, Summer 2020

Simon Schwitanski
Dr. Christian Terboven

i12 High Performance Computing
RWTH AACHEN UNIVERSITY

Outline

0. Organization
1. Foundations
2. Shared Memory
3. GPU Programming
4. Bulk-Synchronous Parallelism
5. Message Passing
6. Distributed Shared Memory
7. Parallel Algorithms
8. Parallel I/O
9. **MapReduce**
10. Apache Spark

a. MapReduce Programming Model
b. Parallelizing MapReduce
c. Hadoop Ecosystem
d. Hadoop Distributed File System
e. Yet Another Resource Negotiator
f. Comparison to Other Approaches
g. MapReduce Design Patterns

MapReduce Implementations

- MapReduce programming model generic, different implementations
- Two different implementations of MapReduce in Hadoop
  - “Classic” MapReduce (MapReduce 1) till including Hadoop 0.20
  - YARN (MapReduce 2) from Hadoop 2.0 onwards
- Further implementations
  - MapReduce-MPI (based on Message Passing Interface)
  - Disco (core written in Erlang, user code in Python)
  - Phoenix (shared memory implementation of MapReduce)
  - Bashreduce (MapReduce in a bash script)
  - …
- Covered here: Hadoop implementation

Word Count Example in Hadoop

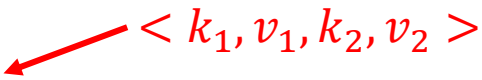
- Map function and Reduce function wrapped in separate classes
- Reminder:
  - $Map(k_1, v_1) \rightarrow list(k_2, v_2)$
  - $Reduce\big(k_2, list(v_2)\big) \rightarrow list(k_3, v_3)$
- Two base classes
  - $\mathrm{Mapper}< k_1, v_1, k_2, v_2 >$: Implementation of Map function
  - $\mathrm{Reducer}< k_2, v_2, k_3, v_3 >$: Implementation of Reduce function
- Application configuration set in Job object

Word Count Example in Hadoop (1)

$$Map(k_1, v_1) \rightarrow list(k_2, v_2)$$

Mapper for Word Count in Hadoop:

```
1  public class WordCountMapper
2          extends Mapper<Object, Text, Text, IntWritable> {
3      private final static IntWritable one = new IntWritable(1);
4      private Text word = new Text();
5
6      public void map(Object key, Text value, Context context)
7              throws IOException, InterruptedException {
8          StringTokenizer itr = new StringTokenizer(
9                                          value.toString());
10
11         while (itr.hasMoreTokens()) {
12             word.set(itr.nextToken());
13             context.write(word, one);
14         }
15     }
16 }
```



Word Count Example in Hadoop (2)

$$Reduce\big(k_2, list(v_2)\big) \rightarrow list(k_3, v_3)$$

Reducer for Word Count in Hadoop:

$< k_2, v_2, k_3, v_3 >$

```
1  public class WordCountReducer
2          extends Reducer<Text,IntWritable,Text,IntWritable> {
3      private IntWritable result = new IntWritable();
4
5      public void reduce(Text key, Iterable<IntWritable> values,
6                         Context context)
7             throws IOException, InterruptedException {
8          int sum = 0;
9          for (IntWritable val : values) {
10             sum += val.get();
11         }
12         result.set(sum);
13         context.write(key, result);
14     }
15 }
```

Sum Reduction

Emit(key, sum)

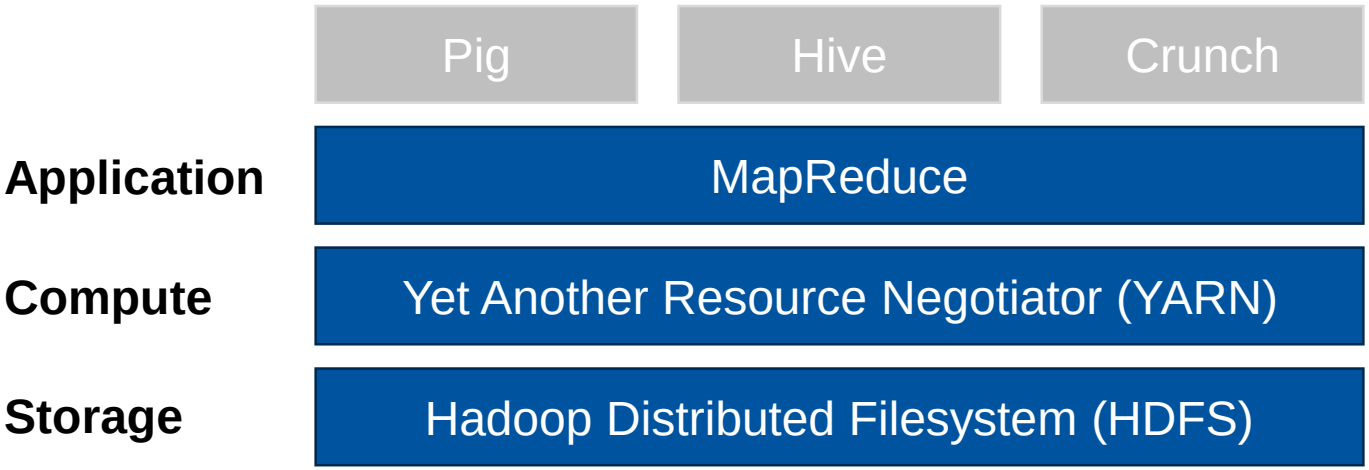
Word Count Example in Hadoop (3)

Job configuration and submission:

```
1  public class WordCount {
2    public static void main(String[] args) throws Exception {
3      Configuration conf = new Configuration();
4      Job job = Job.getInstance(conf, "word count");
5      job.setJarByClass(WordCount.class);
6
7      job.setMapperClass(WordCountMapper.class);
8      job.setReducerClass(WordCountReducer.class);
9
10     job.setOutputKeyClass(Text.class);
11     job.setOutputValueClass(IntWritable.class);
12
13     FileInputFormat.addInputPath(job, new Path(args[0]));
14     FileOutputFormat.setOutputPath(job, new Path(args[1]));
15
16     System.exit(job.waitForCompletion(true) ? 0 : 1);
17   }
18 }
```

Hadoop Ecosystem

- Hadoop: Framework for scalable distributed computing
- Consists of three main components (and extensions)



- Many extensions running on top of these components
  - Pig: High-level language abstraction for MapReduce
  - Hive: Data warehouse system on top of MapReduce, SQL like queries
  - Crunch: Define pipelines of user-defined functions based on MapReduce