Aalto University
School of Science

Big Data Processing - The MapReduce Programming Model

Hong-Linh Truong
Department of Computer Science
linh.truong@aalto.fi, *https://rdsea.github.io*

MapReduce Programming Model

MapReduce programming model

- **MapReduce is a programming model from Google**
  - Various implementations/frameworks support MapReduce
    - Apache Hadoop (**https://hadoop.apache.org**)
    -
- **Support batch data processing for very large datasets**
- **Suitable for batch jobs in big data, e.g.,**
  - Web search, document processing, ecommerce information
  - Extract, transform, data wrangling, and data cleansing tasks

Common needs

- **Thinking if we have data that can be represented as record=(key,value)**
  - e.g., key=”aalto”, value=”1000” (1000 likes in linkedin)
  - potentially millions of records, with millions of keys
- **Operations**
  - data analytics like summarization/aggregation/filtering
    - *count, min, max, average, etc.*
  - joining data from big data set
  - collecting data and shuffling the data to the right tasks

Map & Reduce

- **Map: map data into (key, value)**
  - Receives <key,value>
  - Outputs <key,value> - new set of <key,value>
- **Reduce: compute results from the same key**
  - Receives <key, Iterable[value]>
  - Outputs <key,value>

Example of a real data

Look at the network monitoring data

```
PROVINCECODE,DEVICEID,IFINDEX,FRAME,SLOT,PORT,ONUINDEX,ONUID,TIME,SPEEDIN,SPEEDOUT
XXN,10XXXXXX023,26XXXXXX8,1,2,7,39,100XXXXXX2310207039,01/08/2019 00:04:07,148163,49018
XXN,10XXXXXX023,26XXXXXX8,1,2,7,38,100XXXXXX2310207038,01/08/2019 00:04:07,1658,1362
XXN,10XXXXXX023,26XXXXXX8,1,2,7,9,100XXXXXX2310207009,01/08/2019 00:04:07,6693,5185
```

Sample: https://github.com/rdsea/bigdataplatforms/tree/master/data/onudata

Understand the MapReduce programming model

ONUID SPEEDIN

Key	Value
…	879656
…	65252
…	…
….	103831
…	1045755
…	

Big files (e.g. in HDFS)

Map

Key	Value
7039	879656
7038	65252
…	

Key	Value
7039	103831
7038	1045755

Parallelizing data

Key	Value
7038	65252
7038	1045755
…	…
7039	103831
7039	879656
…	…

Sort/Shuffle

Reduce

Key	Value
7038	…
7039	…
…	…
…	…
…	…
…	

Group/Aggregation

Key ideas of MapReduce

- **A kind of divide-and-conquer paradigm**
- **Data can be divided by "Map" operators**
  - data processing tasks extract "intermediate results"
- **Intermediate results can be aggregated through "Reduce" operators**
  - data processing tasks produce a result from "intermediate results"
- **We can glue "Map" and "Reduce" operators into a multi-stage data flow model**
- **Other possible operators:**
  - `Combiner`: performs "Reduce" at local nodes
  - `Partitioner`: decides key/value for Reduce

Key ideas of MapReduce

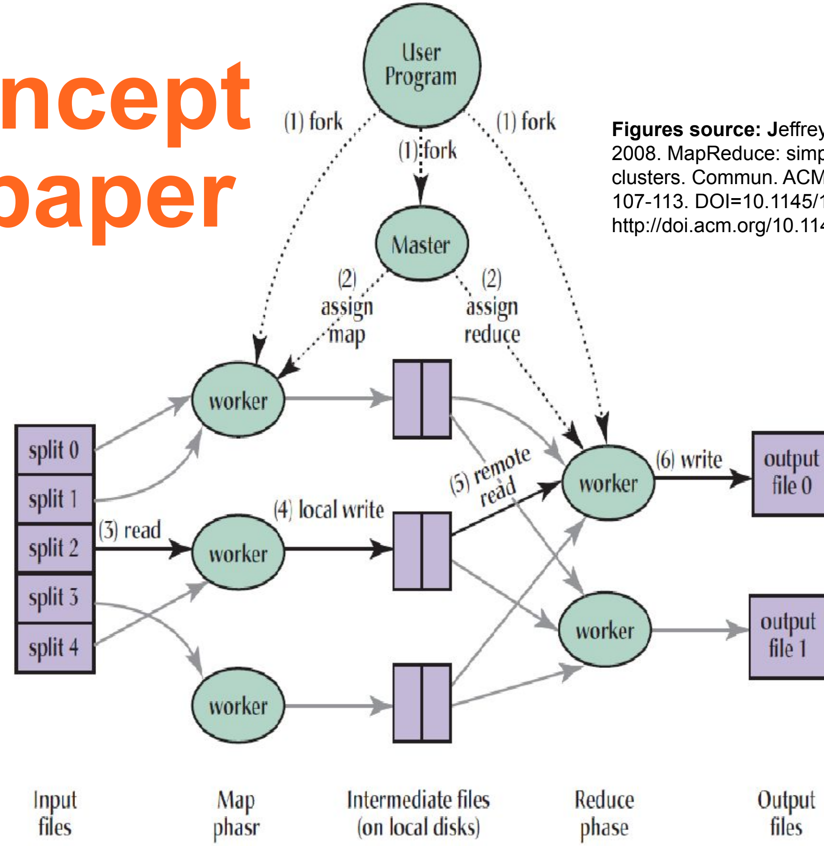
- **Key points for the developers**
  - should write only the main "logic": Map and Reduce operators
- **The runtime framework will**
  - handle data movement and input/output management for Map/Reduce tasks
  - parallelizing tasks in multiple nodes

MapReduce concept in the original paper

Figures source: Jeffrey Dean and Sanjay Ghemawat. 2008. MapReduce: simplified data processing on large clusters. Commun. ACM 51, 1 (January 2008), 107-113. DOI=10.1145/1327452.1327492 http://doi.acm.org/10.1145/1327452.1327492

```
map(String key, String value):
   // key: document name
   // value: document contents
   for each word w in value:
      EmitIntermediate(w, "1");

reduce(String key, Iterator values):
   // key: a word
   // values: a list of counts
   int result = 0;
   for each v in values:
      result += ParseInt(v);
   Emit(AsString(result));
```



Key point: parallelize workers to process a lot of input files and produce a lot of output files

Tasks and their dependencies

- **A task (Map or Reduce) is stateless**
  - executed as an individual process
- **Acyclic graph of tasks as a workflow**
  - can be executed using a batch job scheduler
  - files as the exchange medium among tasks

Hadoop MapReduce

- **Hadoop supports the MapReduce programming model**
  - Use cluster nodes for data processing tasks
  - Access data in HDFS files and partitions in different nodes
  - Hadoop runtime automatically creates parallel tasks
  - YARN is used to run jobs of MapReduce applications
- **Data management (HDFS) and data processing (MapReduce) are aligned nicely**
  - Run in the same nodes ⇒ data locality optimization

Map/Reduce tasks and data/node partitions

- **A Map task can handle a data partition in the same node**
  - e.g., a Map task handles a HDFS data block ⇒ local data optimization: no data movement - local processing
  - Results from a Map task are intermediate ⇒ to where a task will store them?
  - *what if a Map task fails?*
- **Reduce Task**
  - to deal with data produced from different Map tasks ⇒ *where to run the Reduce tasks?*

Examples - Map

Input

Output

```java
public static class SpeedInMapper
     extends Mapper<Object, Text, LongWritable , AverageWritable>{
  private LongWritable id =new LongWritable();
  private AverageWritable averagecount = new AverageWritable();
  public void map(Object key, Text value, Context output)
       throws IOException, InterruptedException {

       String valueString = value.toString();
       String[] record = valueString.split(",");
       id.set(Long.parseLong(record[7]));
       averagecount.setAverage(Float.parseFloat(record[9]));
       averagecount.setCount(1);
       output.write(id,averagecount);
  }
}
```

Parse the data to get ONUID and SPEEDIN

Map (ONUID,(SPEEDIN, count))

Example - Reduce

Input

Output

```
public static class SpeedInAverageReducer
        extends Reducer<LongWritable,AverageWritable,LongWritable,FloatWritable> {
    private FloatWritable new_result = new FloatWritable();


    public void reduce(LongWritable key, Iterable<AverageWritable> values,
                       Context context
                       ) throws IOException, InterruptedException {
      float avg = 0;
      int count = 0;
      for (AverageWritable val : values) {
          float current_avg =val.getAverage();
          int current_count =val.getCount();
          avg = avg + (current_avg*current_count);
          count = count + current_count;
      }

     new_result.set(avg/count);
     context.write(key, new_result);
    }
}
```

Simple way to determine the average as "Reduce" operator

Reduce (ONUID,AVG)

Driver: connect Map and Reduce operators

```
public static void main(String[] args) throws Exception {
  Configuration conf = new Configuration();
  Job job = Job.getInstance(conf, "simpleonuaverage");
  job.setJarByClass(SimpleAverage.class);
  job.setMapperClass(SpeedInMapper.class);
  job.setCombinerClass(SpeedInAverageCombiner.class);
  job.setReducerClass(SpeedInAverageReducer.class);
  job.setMapOutputKeyClass(LongWritable.class);
  job.setMapOutputValueClass(AverageWritable.class);
  job.setOutputKeyClass(LongWritable.class);
  job.setOutputValueClass(FloatWritable.class);
  FileInputFormat.addInputPath(job, new Path(args[0]));
  FileOutputFormat.setOutputPath(job, new Path(args[1]));
  System.exit(job.waitForCompletion(true) ? 0 : 1);
}
```

Combiner

Example with Python using MRJob

```python
class ONUSpeedinAverage(MRJob):
    def mapper(self, _, entry):
        provincecode,deviceid,ifindex,frame,slot,port,onuindex,onuid,timestamp,speedin,speedout= entry.split(",")
        #average speed is speedin with count = 1
        yield (onuid, (float(speedin),1))

   ## recalculate the new speedin average through an array of speedin average values
    def _recalculate_avg(self, onuid, speedin_avg_values):
        current_speedin_total = 0
        new_avg_count = 0
        for speedin_avg, avg_count  in speedin_avg_values:
            current_speedin_total = current_speedin_total +(speedin_avg*avg_count)
            new_avg_count = new_avg_count + avg_count
        new_speedin_avg = current_speedin_total/new_avg_count
        return (onuid, (new_speedin_avg, new_avg_count))

    def combiner(self, onuid, speedin_avg_values):
        yield self._recalculate_avg(onuid, speedin_avg_values)

    def reducer(self, onuid,speedin_avg_values):
        onuid, (speedin_avg, avg_count) = self._recalculate_avg(onuid,speedin_avg_values)
        yield (onuid, speedin_avg)

if __name__ == '__main__':
    ONUSpeedinAverage.run()
```

Note: see code examples in our GIT

Scheduling and monitoring

- **A MapReduce program runs ⇒ MapReduce Job**
  - includes many tasks (Map and Reduce processes + others)
- **JobTracker**
  - monitors the whole job (all tasks of a MapReduce program)
- **TaskTracker**
  - performs a task of the MapReduce applications
  - informs JobTracker about the state of the tasks

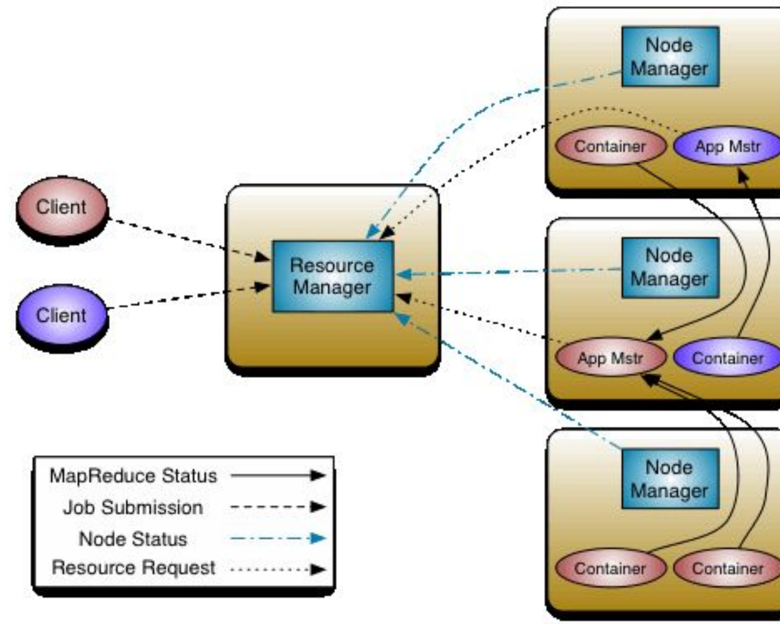


Figure source:
http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html

Monitoring MapReduce Jobs

Logged in as: dr.who

Application application_1570429323498_0008

- Cluster
  - About
  - Nodes
  - Node Labels
  - Applications
    - NEW
    - NEW_SAVING
    - SUBMITTED
    - ACCEPTED
    - RUNNING
    - FINISHED
    - FAILED
    - KILLED
  - Scheduler
- Tools

Application Overview

User:	mybdp
Name:	cse4640-nytaxicount
Application Type:	SPARK
Application Tags:	
Application Priority:	0 (Higher Integer value indicates higher priority)
YarnApplicationState:	FINISHED
Queue:	default
FinalStatus Reported by AM:	SUCCEEDED
Started:	Fri Oct 25 19:22:08 +0000 2019
Elapsed:	3mins, 6sec
Tracking URL:	History
Log Aggregation Status:	DISABLED
Application Timeout (Remaining Time):	Unlimited
Diagnostics:	
Unmanaged Application:	false
Application Node Label expression:	<Not set>
AM container Node Label expression:	<DEFAULT_PARTITION>

Application Metrics

Total Resource Preempted:	<memory:0, vCores:0>
Total Number of Non-AM Containers Preempted:	0
Total Number of AM Containers Preempted:	0
Resource Preempted from Current Attempt:	<memory:0, vCores:0>
Number of Non-AM Containers Preempted from Current Attempt:	0
Aggregate Resource Allocation:	5039065 MB-seconds, 973 vcore-seconds
Aggregate Preempted Resource Allocation:	0 MB-seconds, 0 vcore-seconds

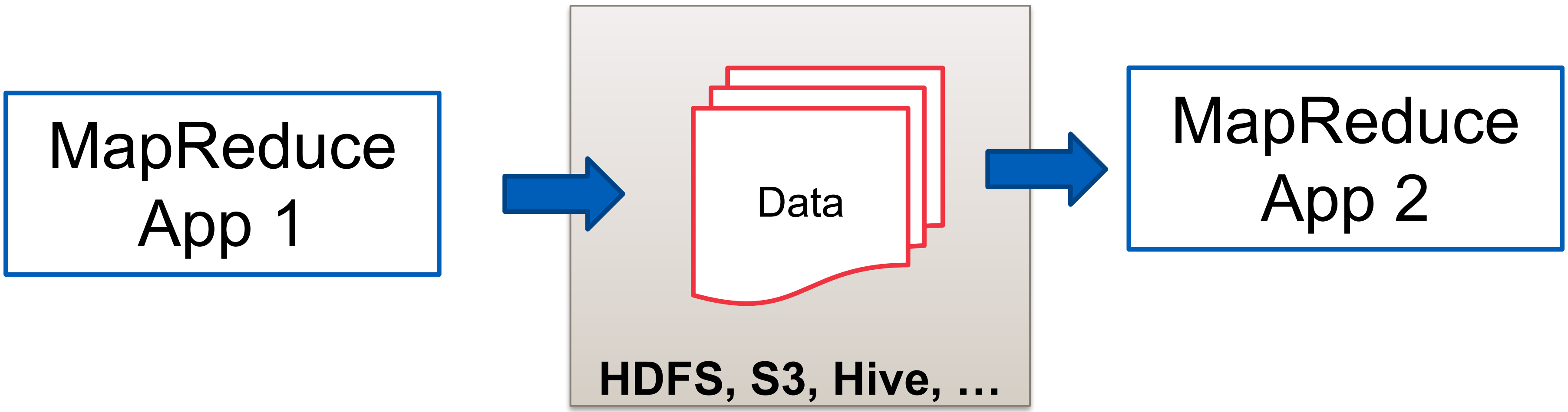
Show 20 entries Search:

Attempt ID	Started	Node	Logs	Nodes blacklisted by the app	Nodes blacklisted by the system
appattempt_1570429323498_0008_000001	Fri Oct 25 22:22:08 +0300 2019	http://cluster-bdp-w-3.c.bigmultidatstore.internal:8042	Logs	0	0

Showing 1 to 1 of 1 entries First Previous 1 Next Last

Connecting MapReduce applications

Build complex MapReduce pipelines



Using big data storage/database as data exchange

We can use workflows to coordinate different MapReduce apps

Problems with MapReduce

- **Strict Map & Reduce tasks connection ⇒ limitation**
- **Need more flexible in processing big data workloads**
  - batch data flows and streaming data flows
- **Programming diversity support**
  - software engineering productivity

Thanks!

Hong-Linh Truong
Department of Computer Science

rdsea.github.io