BIGDATA SI SCALABILITATE

Hadoop MapReduce

Recapitulare Hadoop

- Miezul hadoop are 2 componente
  - Hadoop Distributed File System (HDFS)
  - Hadoop MapReduce
- Cele 2 componente sunt independente
- Aduce procesarea in locul unde datele traiesc

Recapitulare Hadoop: HDFS

- Sistem distribuit de fisiere
- Sistem cu redundanta si disponibilitate ridicata
- NameNode si DataNode
- Rack awareness

MapReduce

- Este un model de programare (framework) pentru a procesa seturi mari de date
- Utilizatorul specifica doua functii
  - map: $Chei_1$ X $Valori_1$ ➔ $(Chei_2$ X $Valori_2)^*$
  - reduce: $Chei_2$ X $Valori_2^*$ ➔ $(Chei_2$ X $Valori_2)^*$
- Programele sunt automat paralelizate si rulate pe clustere mari de masini non enterprise (commodity hardware)

MapReduce: WordCount

Vrem sa aflam care sunt cuvintele care apar intr-un set de documente si de cate ori apar ele.

MapReduce: WordCount Map

- Intrare
  - Cheie: Fisierul din care sunt informatiile
  - Valoare: O linie din fisier
- Iesire
  - Cheie: Un cuvant din linia de input
  - Valoare: 1

MapReduce: WordCount Map

- Intrare: (fisier1.txt, “Ana nu mai are mere”)
- Iesire: [(Ana,1), (nu,1), (mai,1), (are,1), (mere,1)]

- Intrare: (fisier2.txt, “Ana are portocale”)
- Iesire: [(Ana,1), (are,1), (portocale,1)]

MapReduce: WordCount Reduce

- Intrare
  - Cheie: Cuvantul extras din fisier
  - Valoare: O lista care contine doar valori de 1
- Iesire
  - Cheie: Cuvantul extras din fisier
  - Valoare: Numarul de valori de 1

MapReduce: WordCount Reduce

- Intrare: (Ana, [1, 1])
- Iesire: (Ana, 2)

- Intrare: (nu, [1])
- Iesire: (nu, 1)

- Intrare: (are, [1, 1])
- Iesire: (are, 2)

etc

MapReduce: WordCount Reduce

Ana 2

nu 1

mai 1

are 2

mere 1

portocale 1

MapReduce: Numar cuvinte unice

- Folosim iesirea de la WordCount ca intrare
- Iesirea de la functia map va avea o singura cheie
- Intrarea functiei reduce va fi o singura cheie si lista de cuvinte unice
- Iesirea functiei reduce va fi numarul de cuvinte din lista

Numar cuvinte unice: Map

Ana 2	→	(1, Ana)
nu 1	→	(1, nu)
mai 1	→	(1, mai)
are 2	→	(1, are)
mere 1	→	(1, mere)
Portocale 1	→	(1, portocale)

Numar cuvinte unice: Reduce

(1, [Ana, nu, mai, are, mere, portocale])

➔

(unice, 6)

Alte exemple, exercitii

Lista de loguri cu urmatorul format: url username

prot://one.does.not/simply/pass ionel

prot://this.course popeasca

Aflati pentru fiecare utilizator lista de url-uri accesate.

Alte exemple, exercitii

Lista de loguri cu urmatorul format: url username

prot://one.does.not/simply/pass ionel

prot://this.course popeasca

Aflati pentru fiecare utilizator lista de url-uri accesate.

Solutie:

- map(cheie, valoare)->(valoare,cheie)
- reduce este functia identitate

Alte exemple, exercitii

Sistem de recomandari pentru videoclipuri.

Avem o lista de loguri de forma: user videoclip

ionel 2bc3f

popeasca 3ydhy

Stim si lista de prieteni ai fiecarui utilizator, zona geografica etc.

Discutie!

Tipuri de noduri

- JobTracker – nod unic in cluster
- TaskTracker – pe fiecare masina unde se executa codul

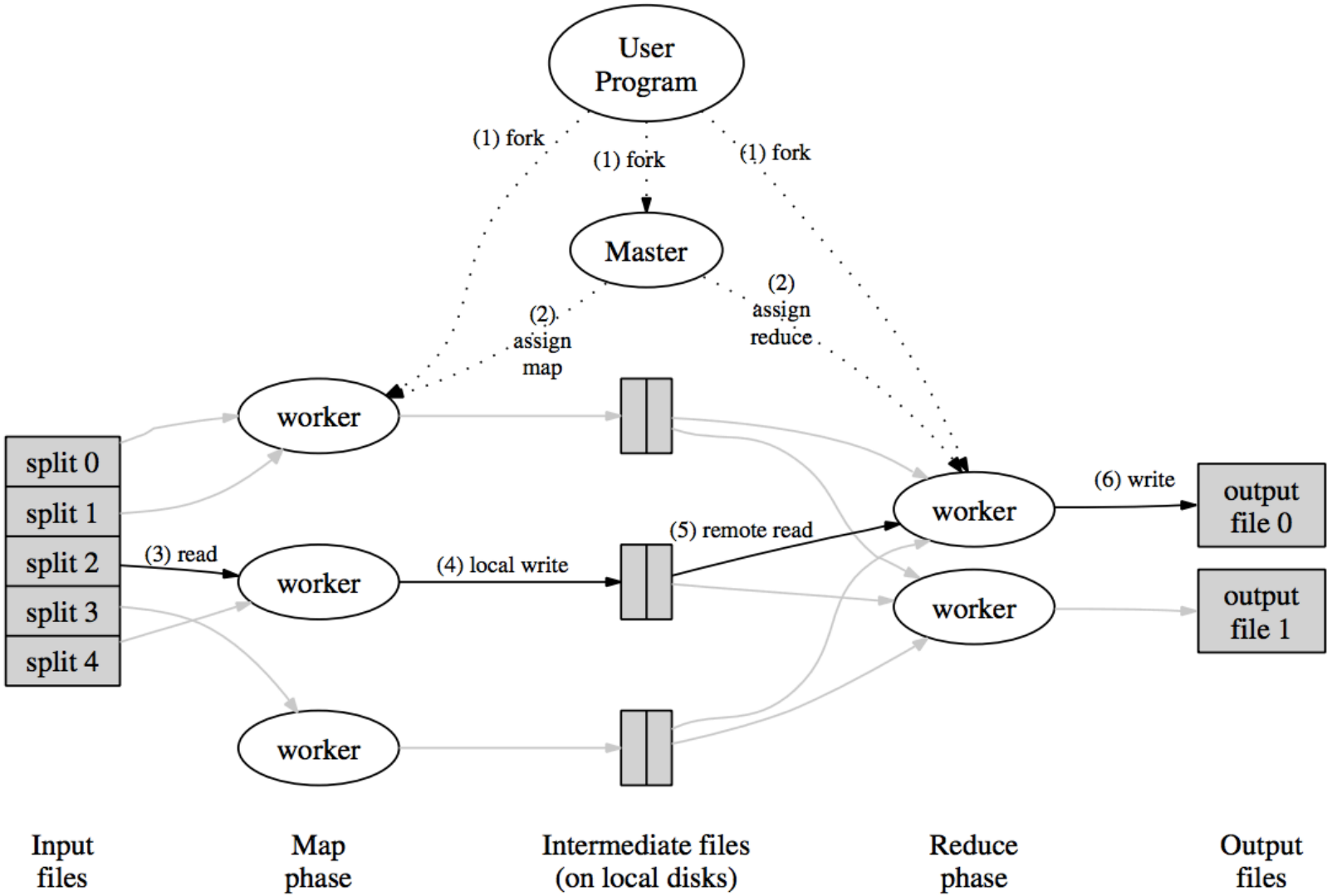
Tipuri de noduri: JobTracker

- La fel ca si NameNode, este este doar un coordonator.
- Decide comunicand cu NameNode-ul masina pe care se executa codul. Alege una aproape de masina pe care sunt stocate datele.
- Monitorizeaza si raporteaza progresul.
- Reporneste task-urile care s-au terminat cu eroare
- etc

Tipuri de noduri: TaskTracker

- Coordoneaza task-urile de pe masina unde ruleaza
- Ruleaza functiile map si reduce
- Monitorizeaza si raporteaza progresul pentru fiecare functie care este executata
- Poate rula mai multe task-uri in paralel. In general sunt rulate intre 10 si 100 de taskuri in functie de capacitatile masinii gazda
- etc

MapReduce Workflow



Mapper

- Se ruleaza functia map pe datele de intrare
- Numarul de task-uri de tipul map depinde de cantitatea datelor dar poate fi controlat.

Reducer

- Faza de reducere este formata din 3 componente:
  - Shuffle: Framework-ul colecteaza partile relevante din iesirea Mapper-ului
  - Sort: Framework-ul grupeaza intrarea pentru reduce dupa chei. Shuffle si sort in general sunt executate simultan
  - Reduce: Functie reduce este apelata

Partitioner

- Partitioneaza spatiul chilor din iesirea functiei map
- Numarul total de partitii este acelasi cu numarul task-uri de tipul reduce
- In general este o functie hash. HashPartitioner default.
- Pentru control mai mare, functia de partitionare se poate suprascrie.

Java SDK

- Ofera un set de interfete pe care aplicatiile trebuie sa le implementeze pentru a rula job-uri.
- Clase utilitare si metode de configurare.
- import org.apache.hadoop.mapred.*;
- class Map extends MapReduceBase implements Mapper<Cheie1, Valoare1, Cheie2, Valoare2>

Exemplu de map

```
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);
                }
        }
}
```

Exemplu de reduce

```
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));
        }
}
```

Exemplu de main

```
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.setCombinerClass(Reduce.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);
}
```

Compilare si impachetare

$ mkdir wordcount_classes

$ javac -classpath ${HADOOP_HOME}/hadoop-${HADOOP_VERSION}-core.jar -d wordcount_classes Sursa.java

$ jar -cvf /usr/joe/wordcount.jar -C wordcount_classes/ .

Rulare

bin/hadoop jar /cale/catre/pachet.jar namespace.Clasa /cale/catre/input /cale/catre/output

Exemplu complet

http://hadoop.apache.org/docs/stable/mapred_tutorial.html

Link-uri utile

- Google map reduce paper: http://static.googleusercontent.com/external_content/untrusted_dlcp/research.google.com/en//archive/mapreduce-osdi04.pdf
- MapReduce tutorial: http://hadoop.apache.org/docs/stable/mapred_tutorial.html