Hadoop

Leleux Laurent

2017 - 2018

Généralités

- Production massive de données
- 3V

- Système distribué
  - HDFS (Hadoop Distributed File System)
  - MapReduce

- Doug Cutting

Généralités

- 2002 Apache Nutch
- 2003 Google publie GFS
- 2004 Google publie MapReduce

- 2006 Hadoop sous-projet d’Apache Lucene
- 2008 Hadoop projet indépendant de la fondation Apache

Intérêt et usages

- Environnement distribué
- Données de grandes tailles

- Yahoo – 42,000 Noeuds (publicités ciblées…)
- Nokia – 1Tb par jour
- EDF
- US Xpress
- Etsy

Intérêt et usages

HDFS

- Notions de EXT2 – FAT
  - Bloc
  - Métadonnées nom –> blocs
  - Droits
  - Répertoires

HDFS

- Différences
  - Non lié au noyau
  - Portable
  - Application externe de montage
  - Distribué (pas de limite de taille)
  - Taille de blocs supérieurs (64Mo+)
  - Réplication des blocs

HDFS - Namenode

- Service central (Maître)
- Connaissance de l’état du FS
  - Métadonnées
  - Répertoires
  - Droits
  - …
- Connaissance des Datanodes

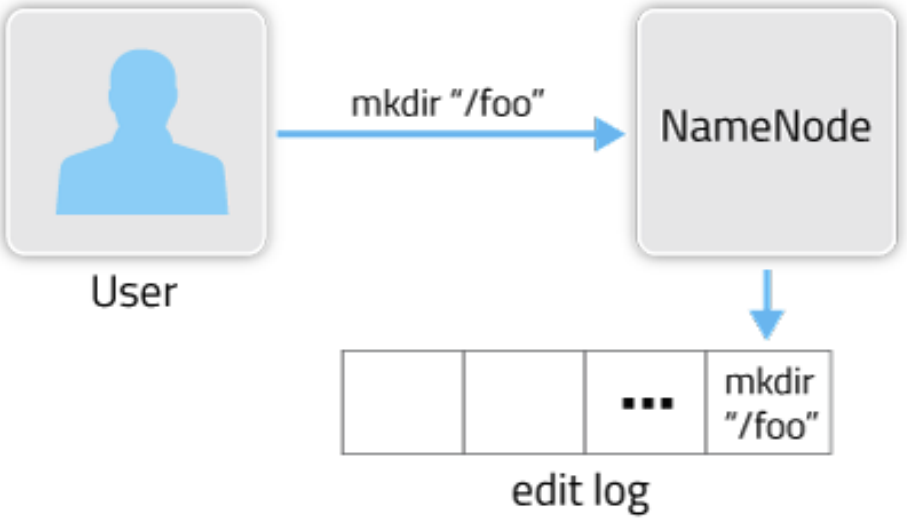
HDFS - Namenode

- Safe mode au démarrage
  - Chargement des position de blocs
  - Read-only
  - Long
  - edits_xxx / fsimage_xxx
- Tout en mémoire
- 150 bytes par fichier => gourmand

HDFS – Secondary Namenode

- SPOF – Single Point Of Failure
- Vérifie l’état du Namenode Principal
- edits_xxx / fsimage_xxx

HDFS - Checkpointing



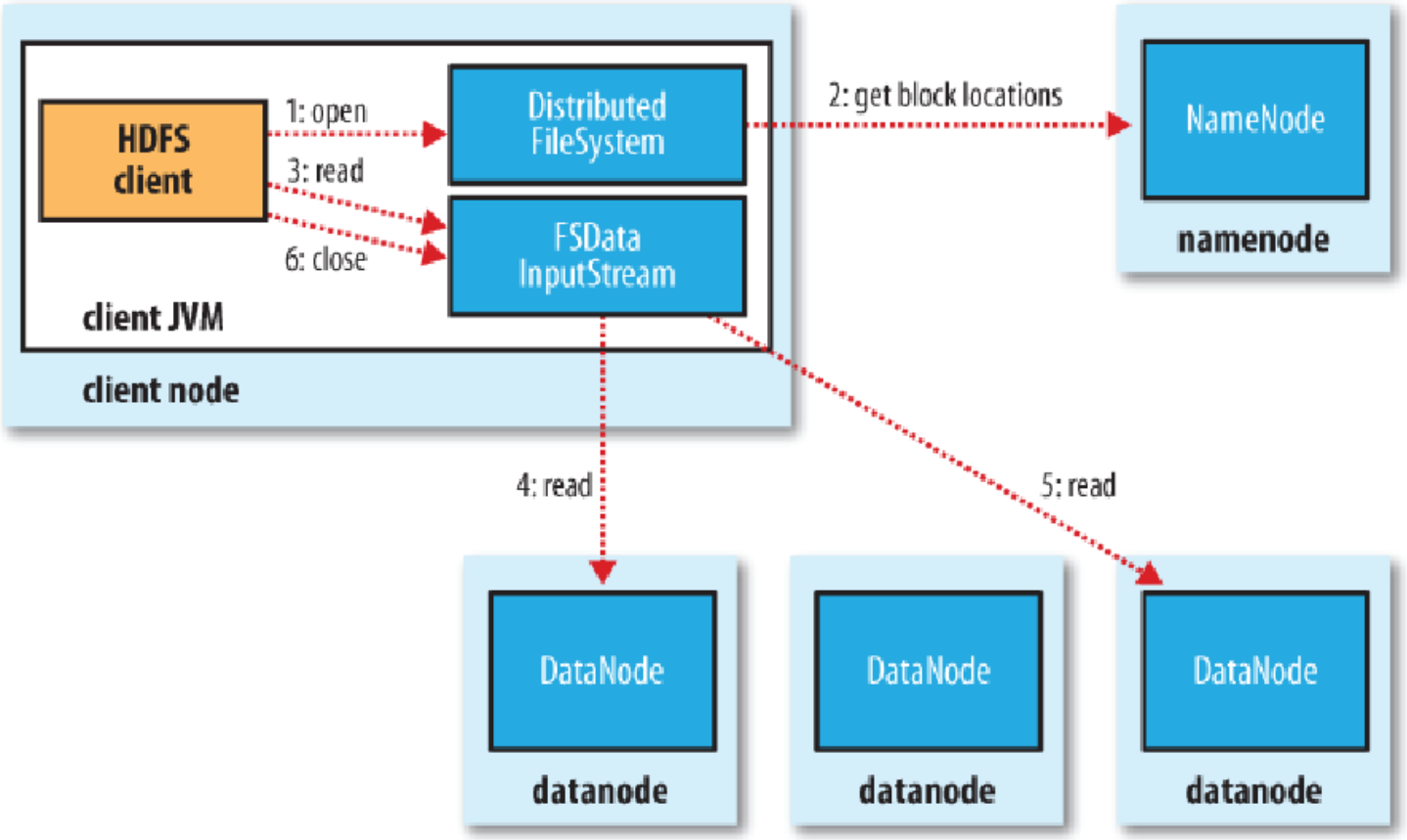
HDFS - Checkpointing

HDFS - Datanode

- Workers
- Contient les blocs
- Réplication (3)

- Dialoguent entre-eux (réplication)
- Se connectent au Namenode au démarrage

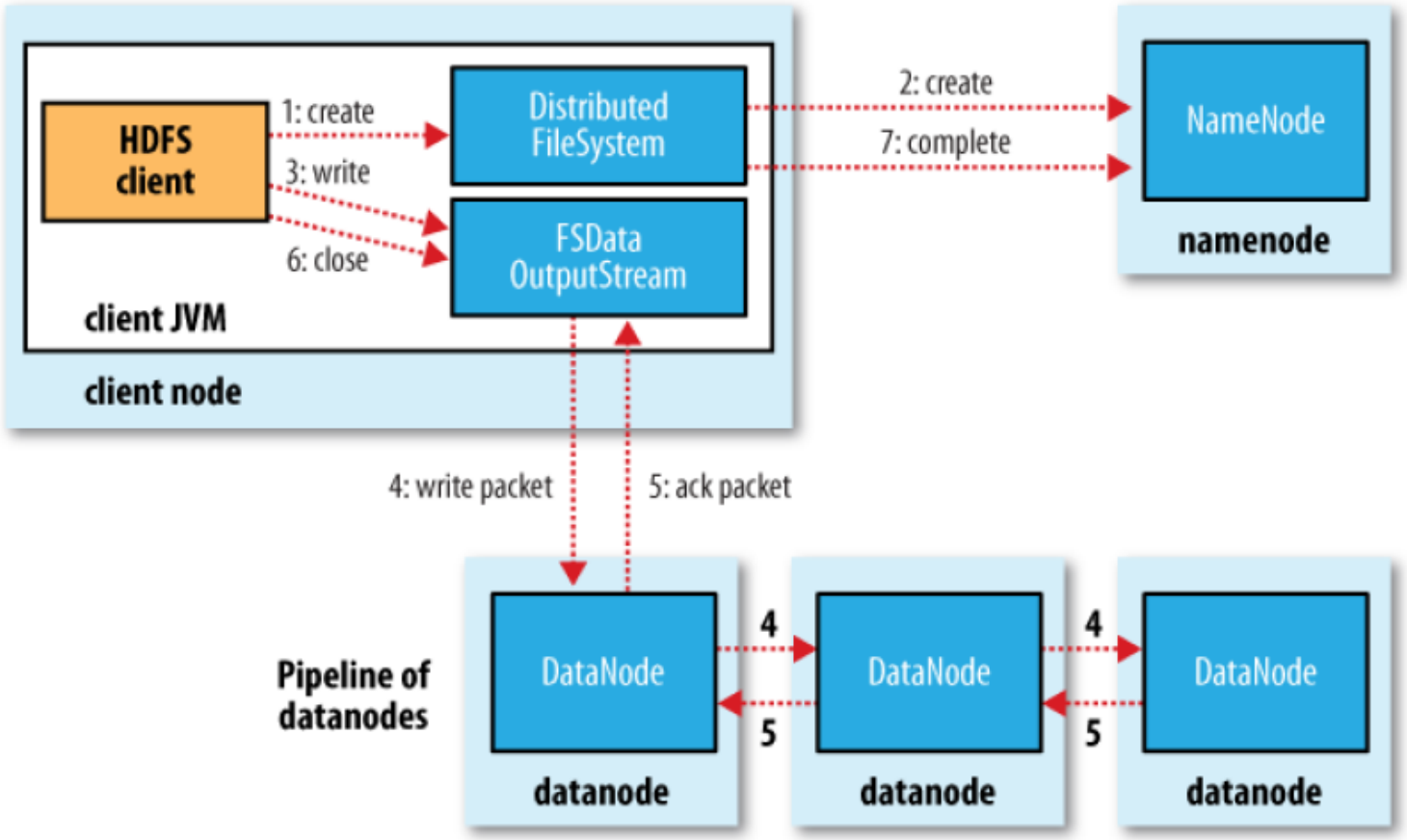
HDFS - Lecture



HDFS - Lecture

```
FileSystem fileSystem = FileSystem.get(conf);
Path path = new Path("/path/to/file.ext");
if (!fileSystem.exists(path)) {
   System.out.println("File does not exists");
   return;
}
FSDataInputStream in = fileSystem.open(path);
int numBytes = 0;
while ((numBytes = in.read(b)) > 0) {
   System.out.prinln((char)numBytes));// code to manipulate
the data which is read}
   in.close();
   out.close();
   fileSystem.close();
}
```

HDFS - Ecriture



HDFS - Ecriture

```
FileSystem fileSystem = FileSystem.get(conf);
// Check if the file already exists
Path path = new Path("/path/to/file.ext");
if (fileSystem.exists(path)) {
    System.out.println("File " + dest + " already exists");
    return;
}
// Create a new file and write data to it.
FSDataOutputStream out = fileSystem.create(path);
InputStream in = new BufferedInputStream(new FileInputStream(new File(source)));

byte[] b = new byte[1024];
int numBytes = 0;
while ((numBytes = in.read(b)) > 0) {
    out.write(b, 0, numBytes);
}
// Close all the file descripters
in.close();
out.close();
fileSystem.close();
```

MapReduce

- Modèle de programmation
- Framework d’implémentation

- Parallélisation et distribution, récupération, couche réseau…

MapReduce

- Job
  - données d’entrée
  - programme MapReduce
  - paramètres d’exécution

- Divisé en tâches par Hadoop
  - Map
  - Reduce

MapReduce - Map

- map(clé, valeur) → List(clé, valeur)
- Input

  Hello World Bye World
  Hello Hadoop Goodbye Hadoop

- Output

  < Hello, 1>
  < World, 1>
  < Bye, 1>
  < World, 1>

  < Hello, 1>
  < Hadoop, 1>
  < Goodbye, 1>
  < Hadoop, 1>

MapReduce - 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);
        }
    }
}
```

MapReduce - Combine

- Combine(List(clé, valeur)) → List(clé, valeur)
- Input

< Hello, 1>
< World, 1>
< Bye, 1>
< World, 1>

< Hello, 1>
< Hadoop, 1>
< Goodbye, 1>
< Hadoop, 1>

- Output

< Bye, 1>
< Hello, 1>
< World, 2>

< Goodbye, 1>
< Hadoop, 2>
< Hello, 1>

MapReduce - Reduce

- reduce(clé, List(valeur)) → List(valeur)
- Input

  < Bye, 1>
  < Hello, 1>
  < World, 2>

  < Goodbye, 1>
  < Hadoop, 2>
  < Hello, 1>

- Output

  < Bye, 1>
  < Goodbye, 1>
  < Hadoop, 2>
  < Hello, 2>
  < World, 2>

MapReduce - 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));
    }
}
```

MapReduce - Hadoop

- Split
- Map
  - proximité de la donnée
  - stockage sur le nœud
- Reduce
  - sur un seul noeud

Cloudera

- 2008
- Apache Hadoop-based software, support and services, and training to business customers.
- CDH