Spark

zenika

ARCHITECTURE INFORMATIQUE

RDD

- Resilient
- Distributed
- Dataset

RDD

- Liste de partitions
- Fonction pour traiter chaque partition
- Noeuds les plus proches d'une partition
- RDDs parents

Transformations

1 ou plusieurs RDD → 1 RDD.

Evaluation lazy

- map, flatMap, filter, distinct
- groupBy, reduce, fold,
- substract, intersection, union, cartesian

Actions

1 RDD → Donnée brute

Declenche la soumission d'un Job.

- count, countByValue, min, max
- first, take, collect, foreach
- saveAsTextFile, saveAsObjectFile

PairRDD

Couples (clé,valeur)

- mapValue
- groupByKey, reduceByKey, sortByKey,
- join, leftOuterJoin, rightOuterJoin, cogroup
- partitionBy, coalesce
- saveAsSequenceFile

Caching

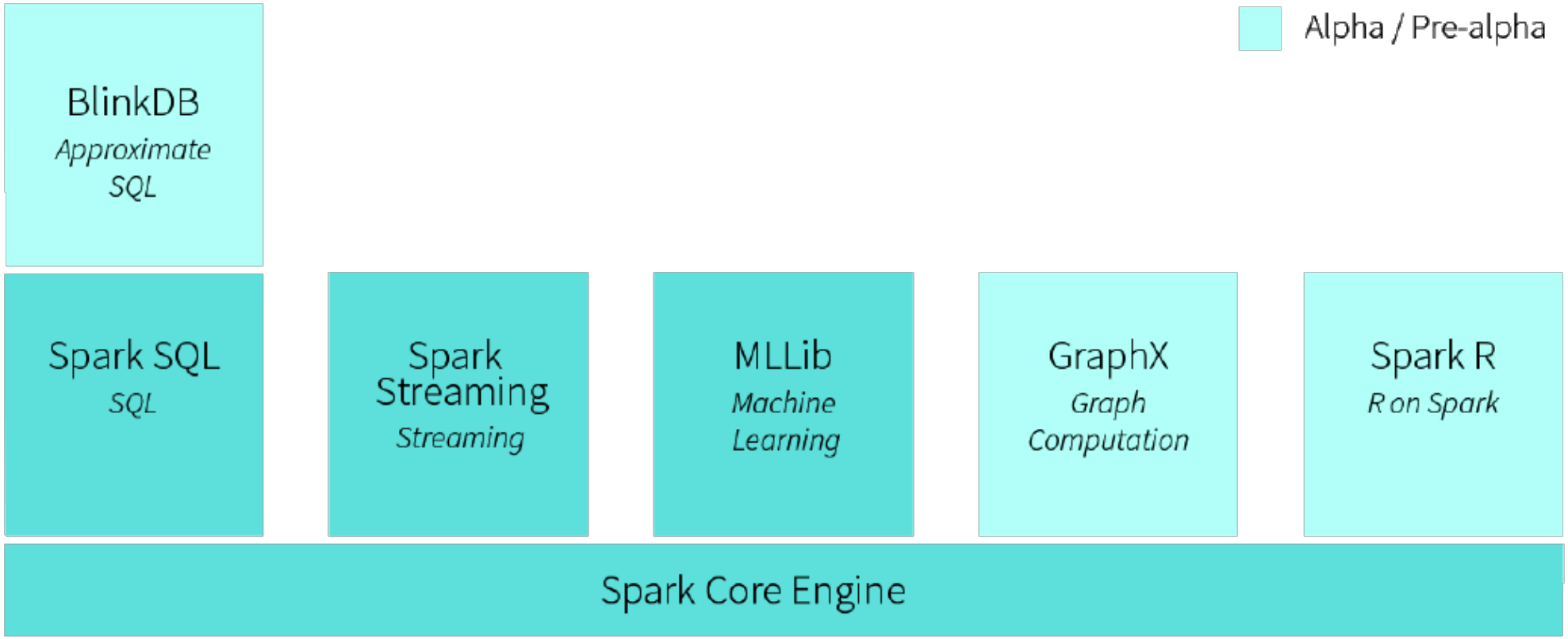
- cache, persist
- Memoire et/ou Disque local
- Sérialisation ou pas
- Off-heap TACHYON

Partititioning

- A la source
- Par clé + hashage

Demo

Ecosystème Spark



Spark SQL

- Ex-Shark

SchemaRDD

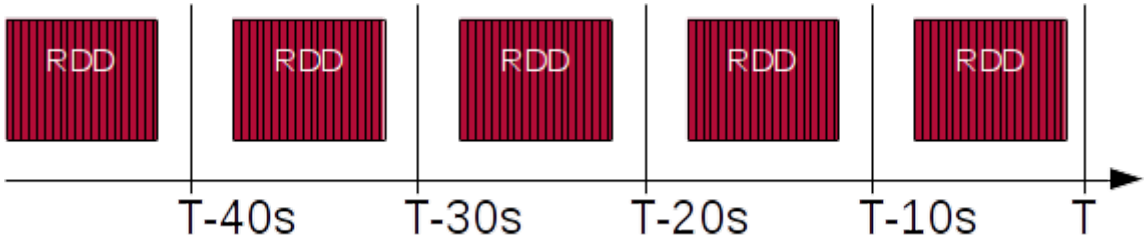
- Table: Lignes × Colonnes
- Description des colonne: nom, type
  - Manuel
  - Détection (JSON, Parquet, DB)

SQL

- select ... from ... join ... where ... group by ... order by
- Rule based optimizer: Catalyst
- Compatible HiveQL?

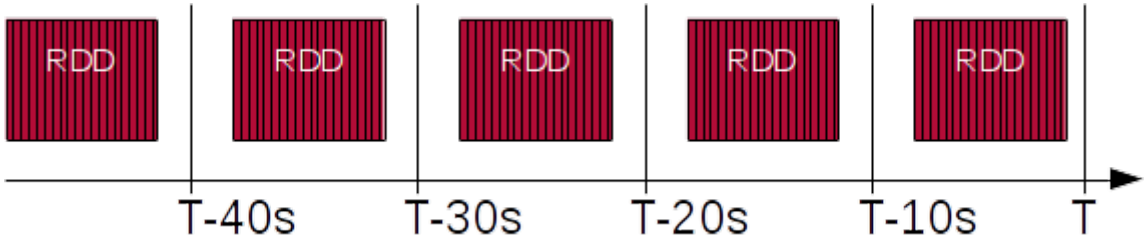
Spark Streaming

- Micro-batch
- DStream
  - Discrete Stream
  - Suite de RDD, 1 toutes les N secondes
  - Même API que Spark Core: transformations, actions



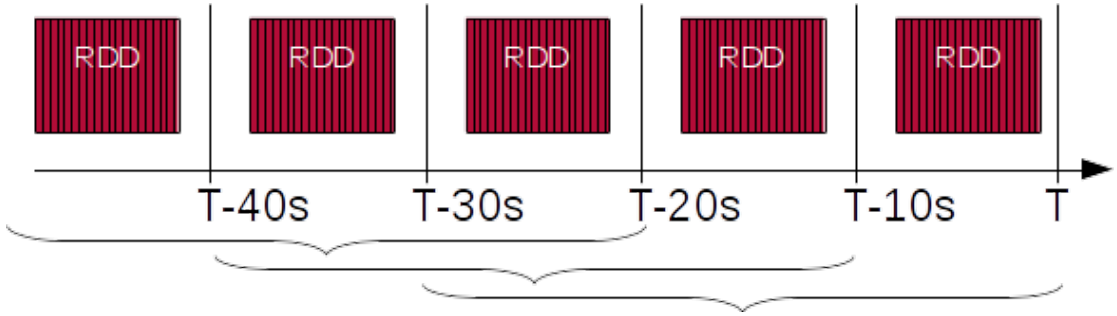
Stream Source

- Fichiers: local, HDFS
- "Broker": Kafka, ZeroMQ, Akka
- Autre: Twitter, Socket, Flume



Stateful Stream

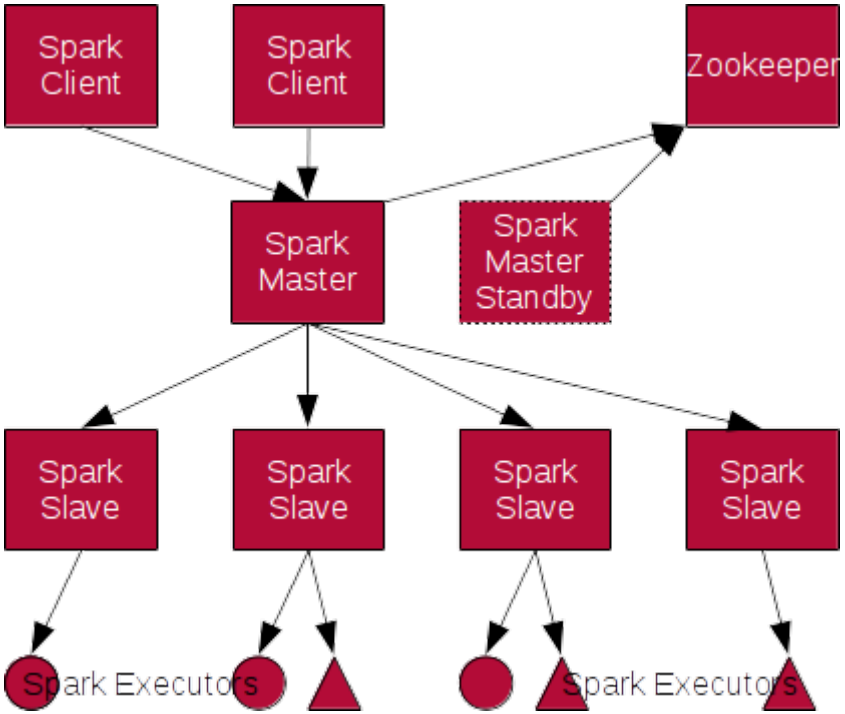
- Window:
  - Détection de fraude, de tendance...



- UpdateStateByKey:
  - Maintenir un état
  - 10 articles/utilisateur
- Stateful => checkpoint

Demo

Cluster



Cluster manager

Standalone

YARN

Mesos

Demo

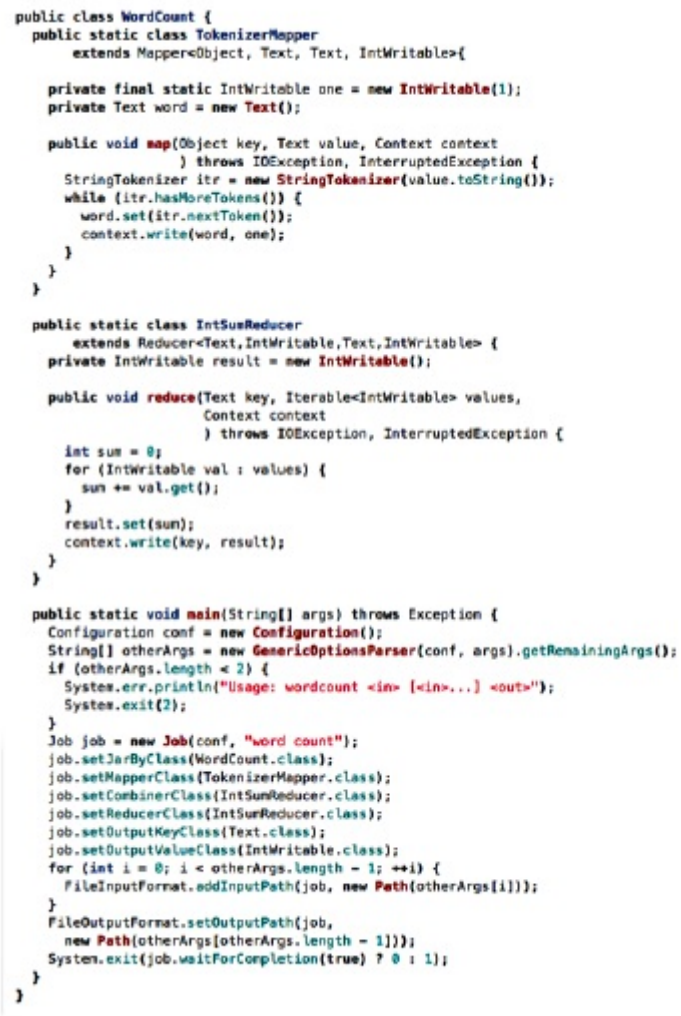
Hadoop

- Mêmes cas d'utilisation
- Intégration dans l'écosystème

Plus ...

Les slides qui suivent ne sont pas objectifs

Plus simple



```java
public class WordCount {
  public static class TokenizerMapper
       extends Mapper<Object, Text, Text, IntWritable>{

    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    public void map(Object key, Text value, Context context
                    ) throws IOException, InterruptedException {
      StringTokenizer itr = new StringTokenizer(value.toString());
      while (itr.hasMoreTokens()) {
        word.set(itr.nextToken());
        context.write(word, one);
      }
    }
  }

  public static class IntSumReducer
       extends Reducer<Text,IntWritable,Text,IntWritable> {
    private IntWritable result = new IntWritable();

    public void reduce(Text key, Iterable<IntWritable> values,
                       Context context
                       ) throws IOException, InterruptedException {
      int sum = 0;
      for (IntWritable val : values) {
        sum += val.get();
      }
      result.set(sum);
      context.write(key, result);
    }
  }

  public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    String[] otherArgs = new GenericOptionsParser(conf, args).getRemainingArgs();
    if (otherArgs.length < 2) {
      System.err.println("Usage: wordcount <in> [<in>...] <out>");
      System.exit(2);
    }
    Job job = new Job(conf, "word count");
    job.setJarByClass(WordCount.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    for (int i = 0; i < otherArgs.length - 1; ++i) {
      FileInputFormat.addInputPath(job, new Path(otherArgs[i]));
    }
    FileOutputFormat.setOutputPath(job,
      new Path(otherArgs[otherArgs.length - 1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

```scala
val f = sc.textFile(inputPath)
val w = f.flatMap(l => l.split(" ")).map(word => (word, 1)).cache()
w.reduceByKey(_ + _).saveAsText(outputPath)
```

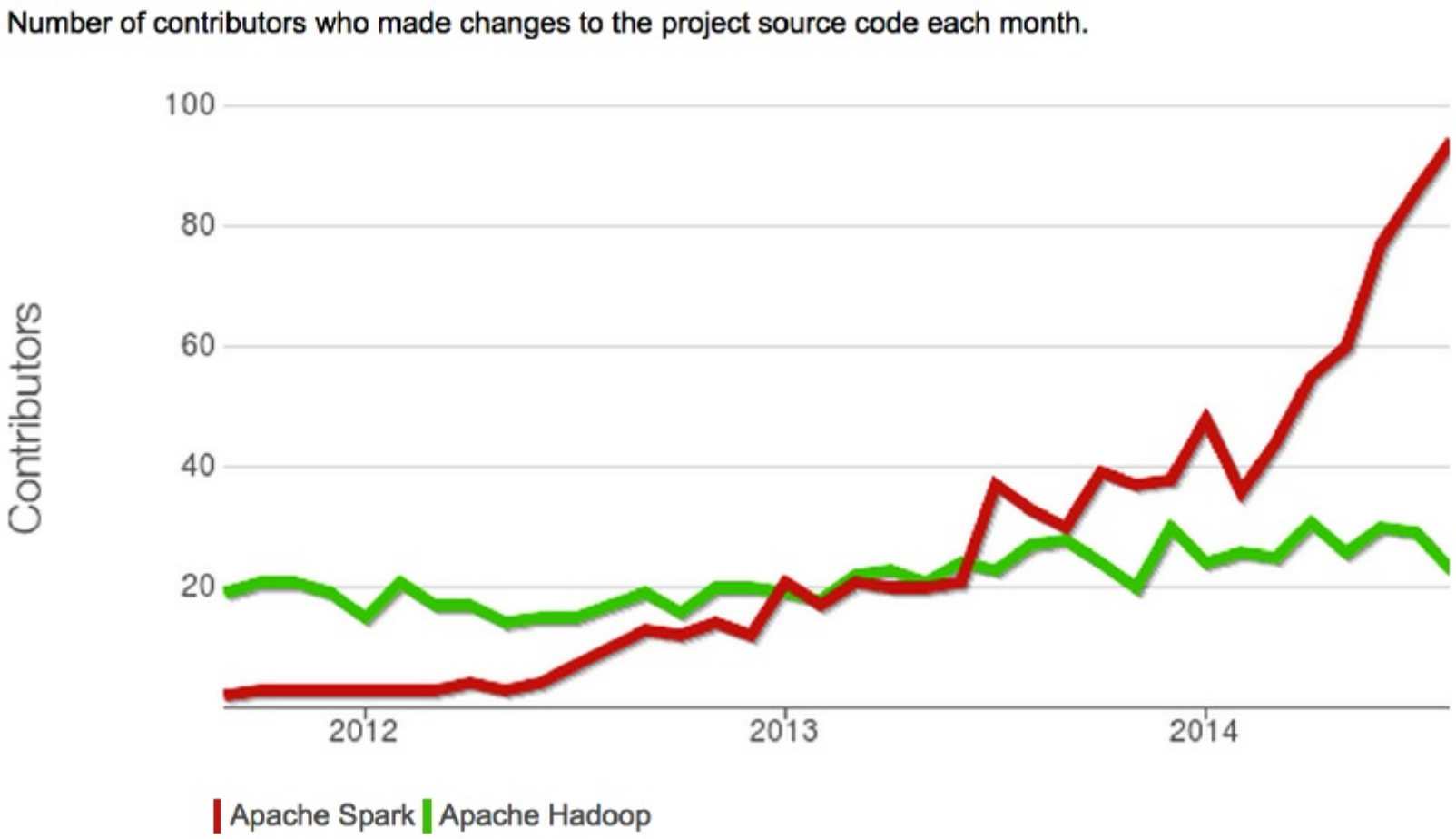
- API style collection habituelle
- Spark Shell
- Spark local

Plus rapide

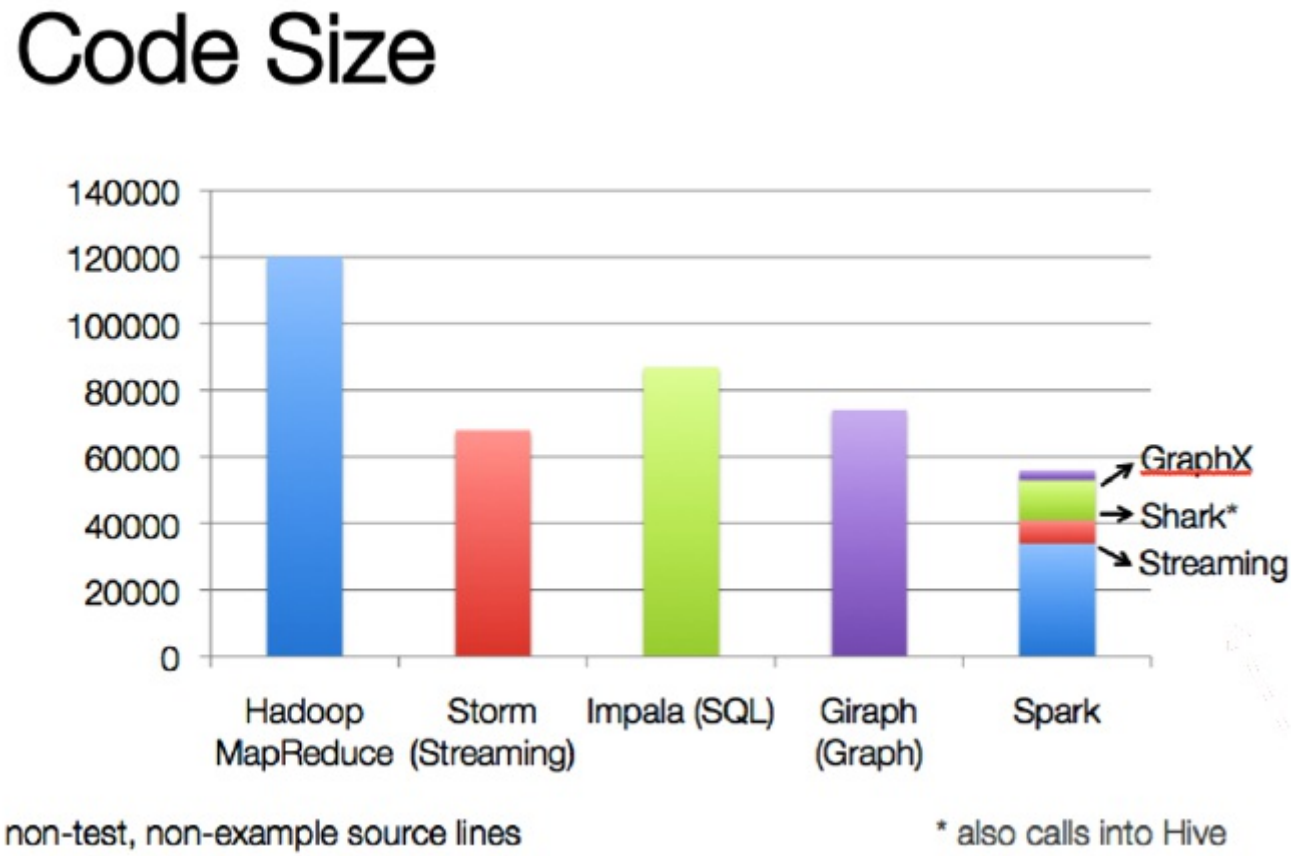
Spark officially sets a new record in large-scale sorting

	Hadoop MR Record	**Spark Record**	**Spark 1 PB**
Data Size	102.5 TB	100 TB	1000 TB
Elapsed Time	72 mins	23 mins	234 mins
# Nodes	2100	206	190
# Cores	50400 physical	6592 virtualized	6080 virtualized
Cluster disk throughput	3150 GB/s (est.)	618 GB/s	570 GB/s
Sort Benchmark Daytona Rules	Yes	Yes	No
Network	dedicated data center, 10Gbps	virtualized (EC2) 10Gbps network	virtualized (EC2) 10Gbps network
Sort rate	**1.42 TB/min**	**4.27 TB/min**	**4.27 TB/min**
Sort rate/node	**0.67 GB/min**	**20.7 GB/min**	**22.5 GB/min**

Plus vivant



Plus léger

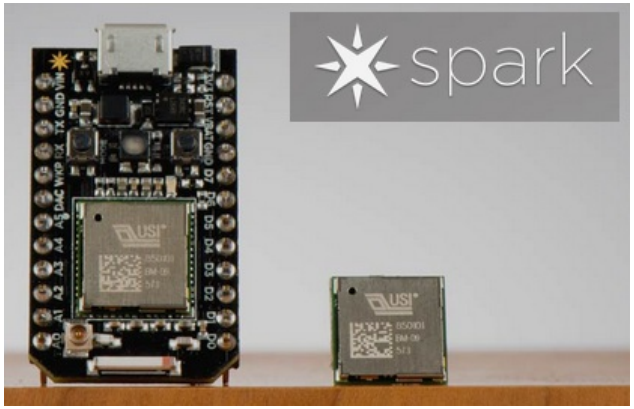


Moins Java

Moins répandu

Amazon, Autodesk, Baidu, eBay, Groupon, Kelkoo, NASA, Shazam, Yahoo...

Moins Google-friendly



Spark