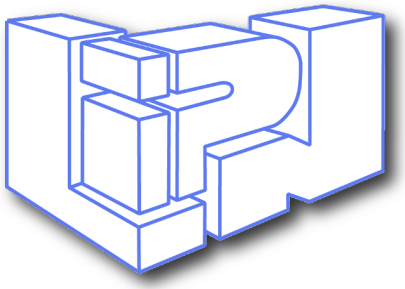
Apprentissage massivement distribué dans un environnement Big Data

Tugdual Sarazin

tugdual.sarazin@altic.org

Thésard CIFRE (janvier 2013)

- Décisionnel & Business Intelligence
- Intégration de données
- Offre BigData
- Principalement des solutions Open Source

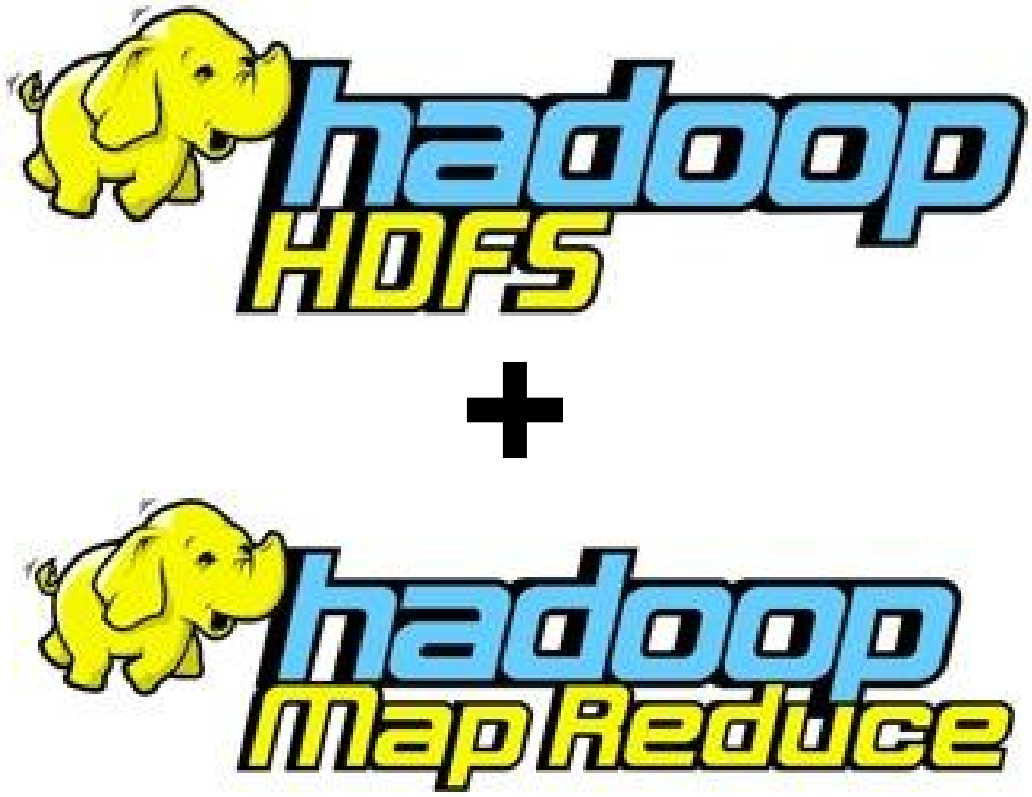


- Laboratoire d'Informatique de Paris-Nord
- Équipe A3 : Apprentissage Artificiel et Applications
- Encadrants: M. Lebbah, H. Azzag

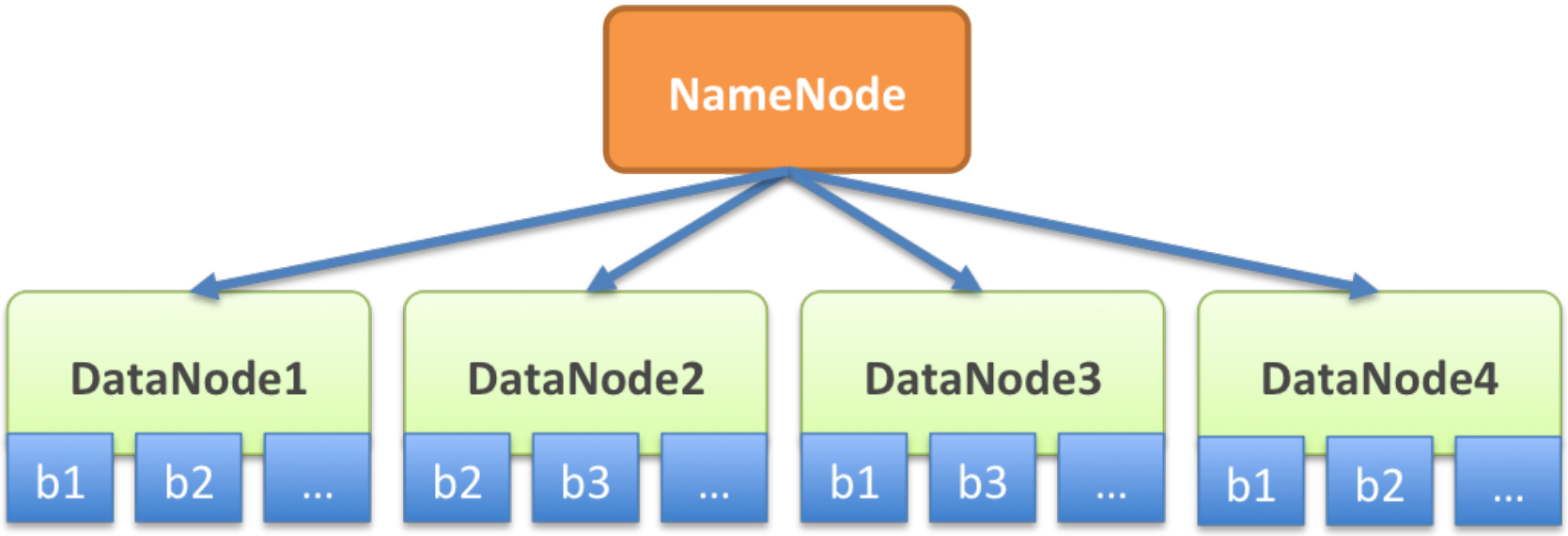
Plan

- Présentation de Hadoop
- Spark une alternative à Hadoop MapReduce
- Implémentation de k-means sur Spark
- Le monde de Spark

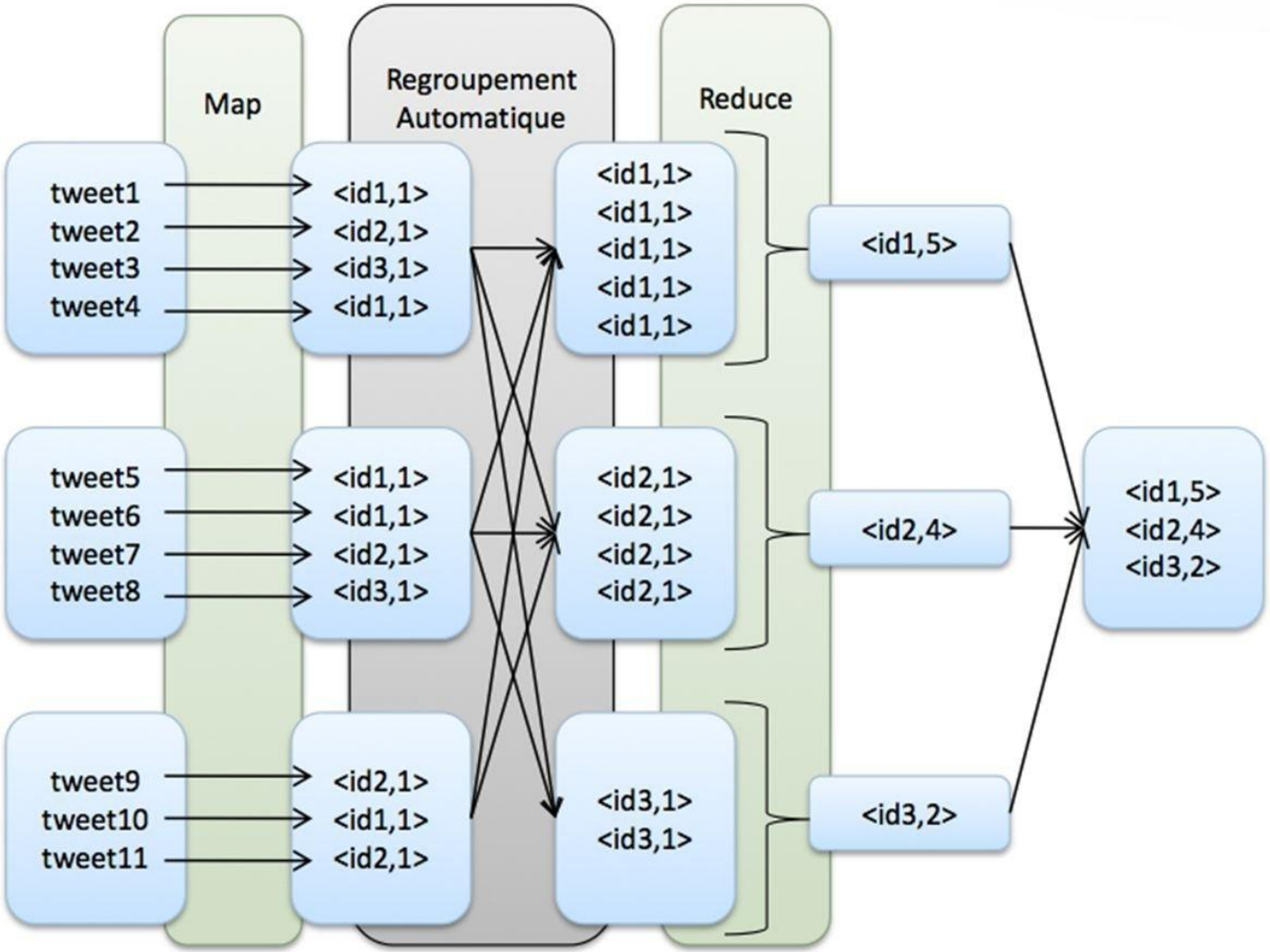
Qu'est-ce que Hadoop?



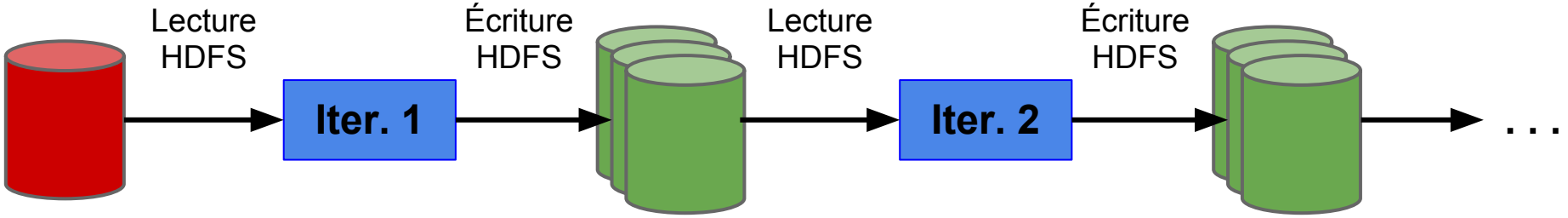
HaDoop File System



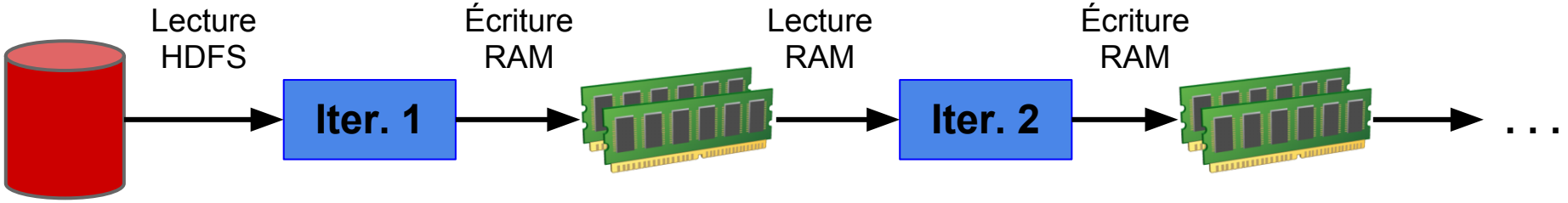
MapReduce

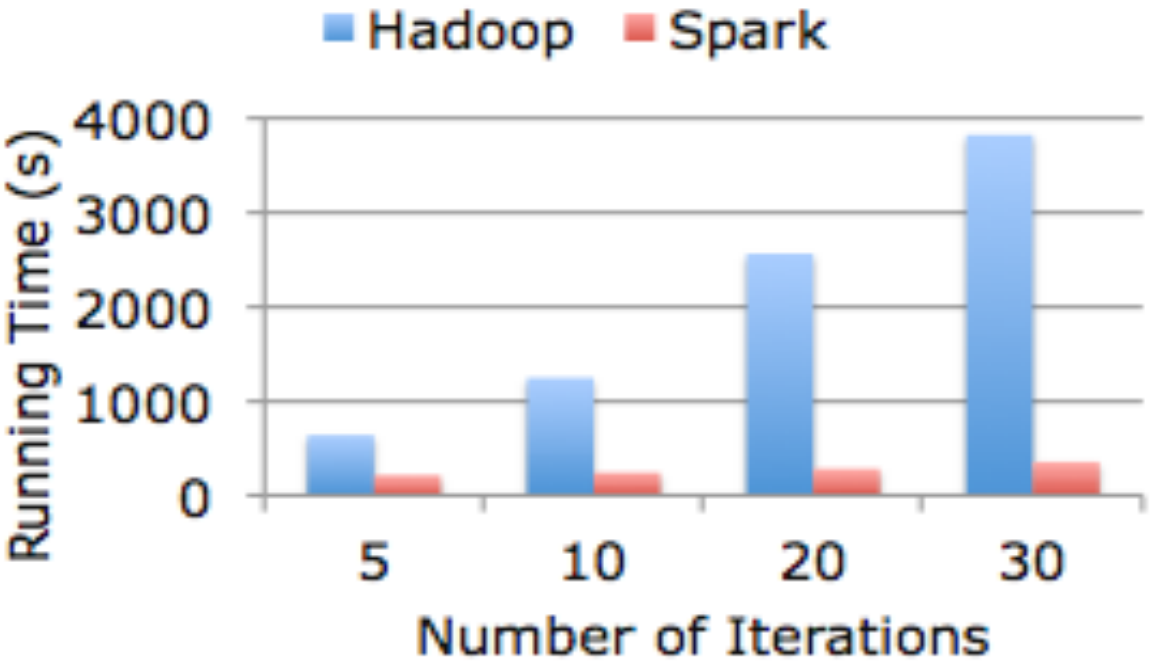

Map
Regroupement Automatique
Reduce
tweet1
tweet2
tweet3
tweet4
<id1,1>
<id2,1>
<id3,1>
<id1,1>
tweet5
tweet6
tweet7
tweet8
<id1,1>
<id1,1>
<id2,1>
<id3,1>
tweet9
tweet10
tweet11
<id2,1>
<id1,1>
<id2,1>
<id1,1>
<id1,1>
<id1,1>
<id1,1>
<id1,1>
<id2,1>
<id2,1>
<id2,1>
<id2,1>
<id3,1>
<id3,1>
<id1,5>
<id2,4>
<id3,2>
<id1,5>
<id2,4>
<id3,2>

Machine learning avec Hadoop MapReduce



Machine learning avec Spark

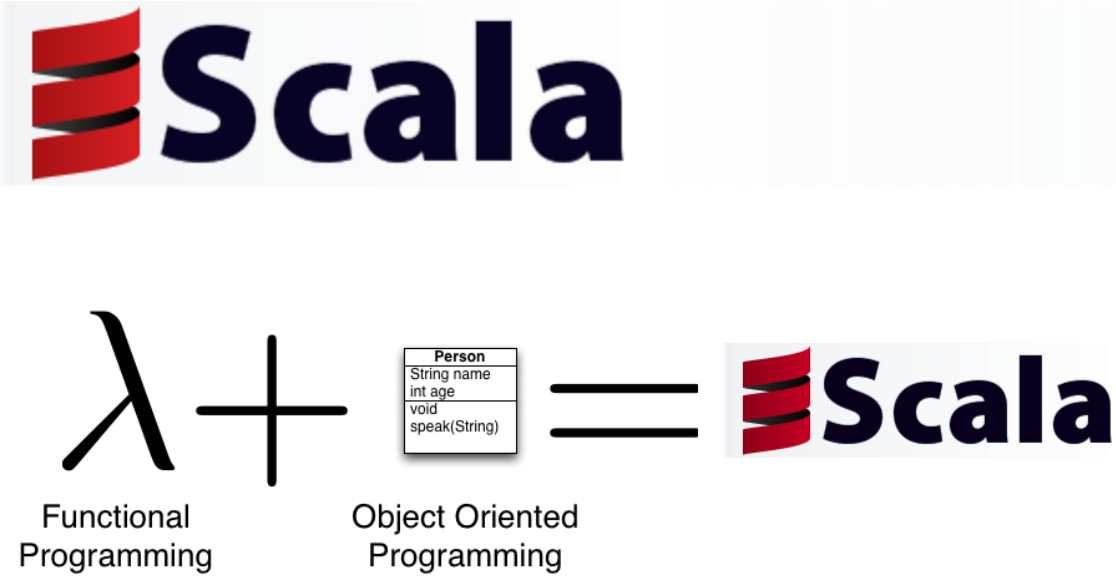




Spark : les bases

Objectif : travailler sur des collections distribuées comme vous le feriez en local

- Resilient Distributed Datasets (RDDs)
  - Collection d'objets immuables et partitionnés
  - Stockage en colonne
  - Persistance contrôlable (mise en cache en RAM)
- MapReduce like
- Transformations (ex: map, filter, groupBy, join)
  - Évaluation paresseuse
- Actions (ex: count, collect, save)



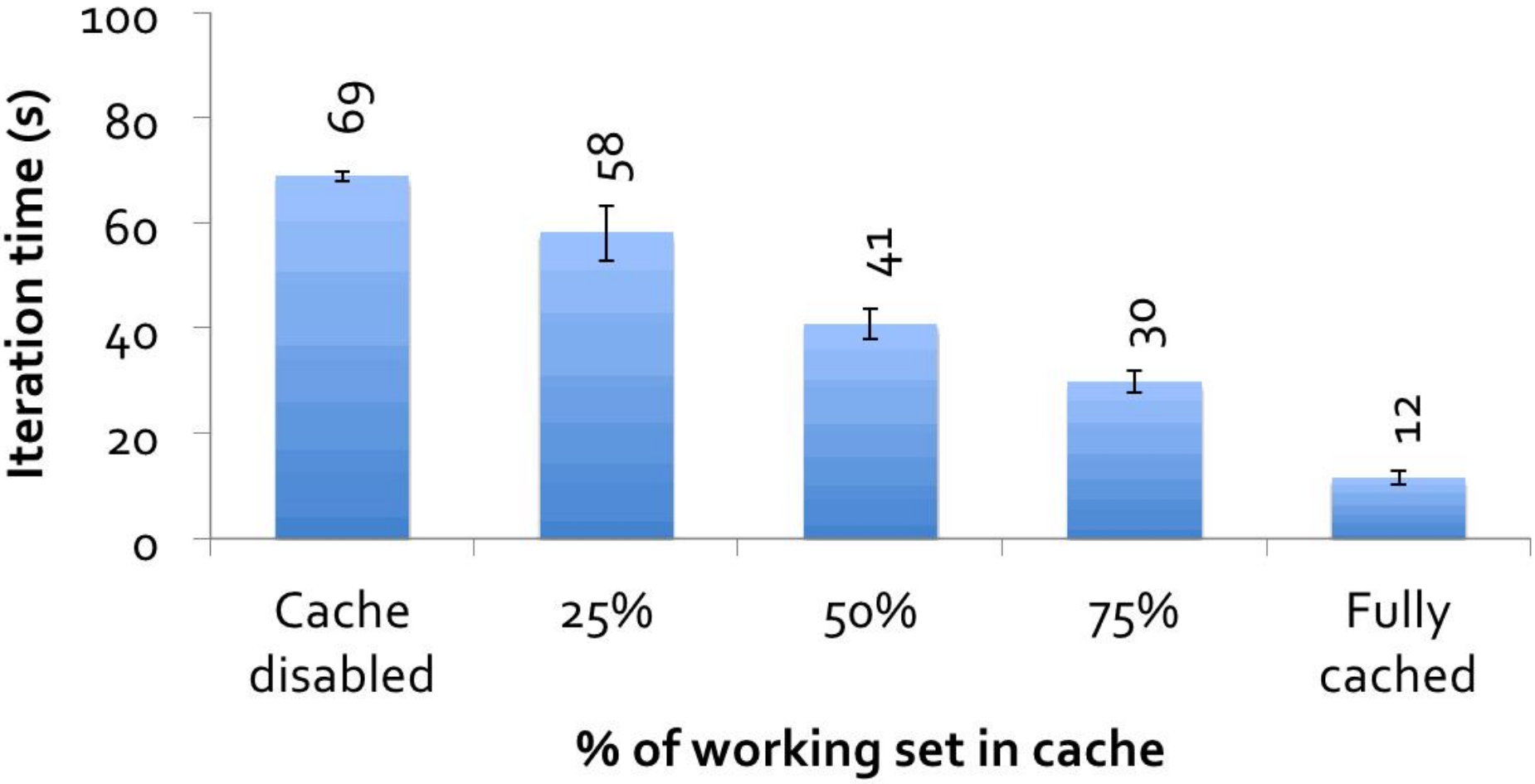
- Bytecode Java (exécutable sur la JVM)
- Syntaxe concise
- Mode **interactif** : console Scala

Spark Kmeans

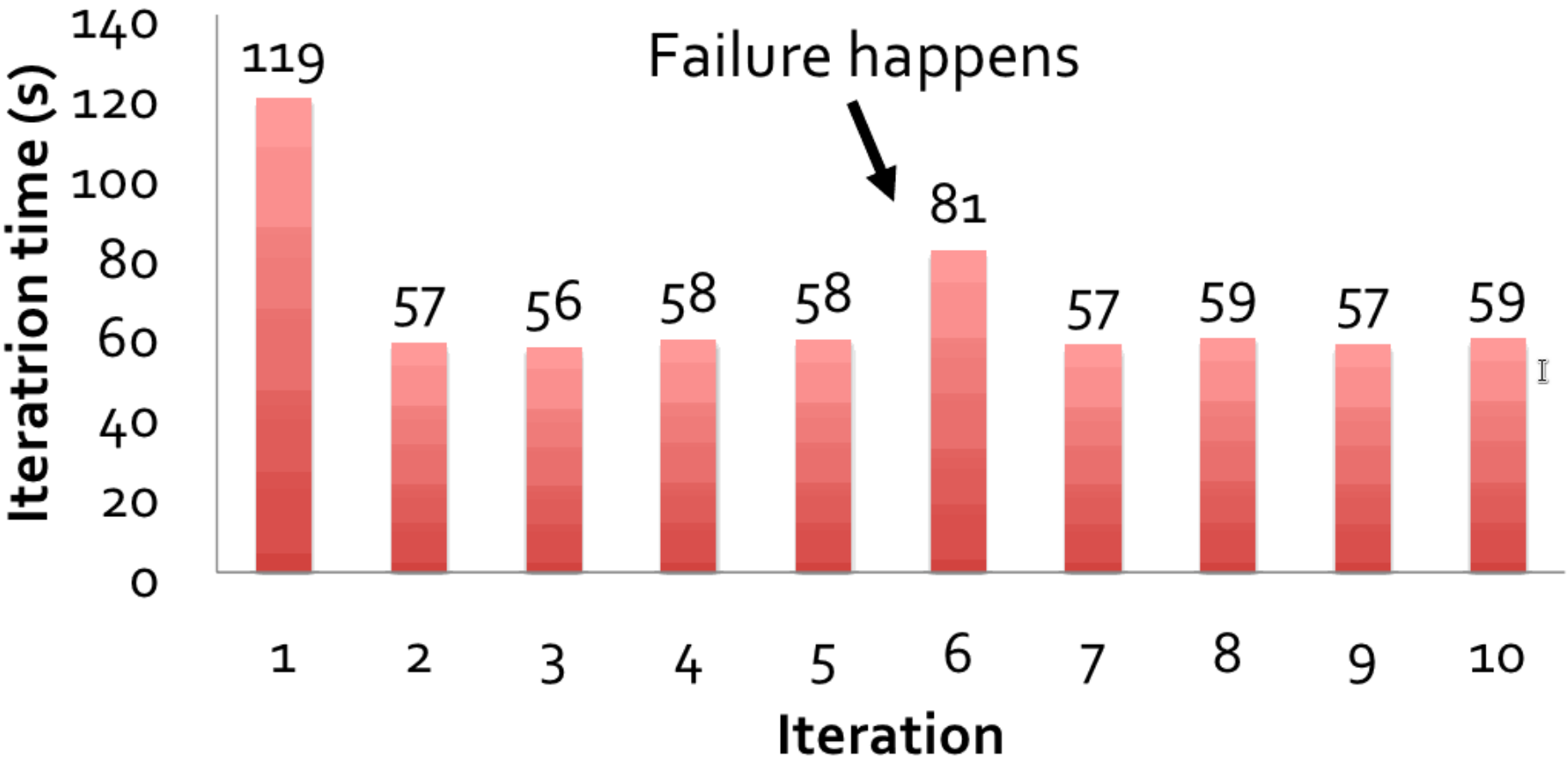
```
for (i <- 1 until 10) {

  // Recherche du centroid le plus proche de chaque point
  val closest = data.map(p =>
    (closestCentroid(p, centroids), (p, 1))
  )



  // Somme des points affectés à chaque cluster
  val pointStats = closest.reduceByKeyLocally{
    case ((p1, sum1), (p2, sum2)) => (p1 + p2, sum1 + sum2)
  }


  // Recalcule les centroids
  pointStats.foreach{case(id, value) =>
       centroids(id) = value._1 / value._2
  }
}
```

```
closest = {
 1 => ([3.145, 2.410], 1)
 0 => ([1.841, 3.846], 1)
 2 => ([1.321, 2.078], 1)
 1 => ([3.186, 2.608], 1)
 ... }


pointStats = {
 0 => ([72.646, 137.915], 44)
 1 => ([363.649, 257.316], 105)
 2 => ([78.741, 91.841], 51)
}

centroids = [
 [1.651, 3.134]
 [3.463, 2.450]
 [1.543, 1.800]
]
```

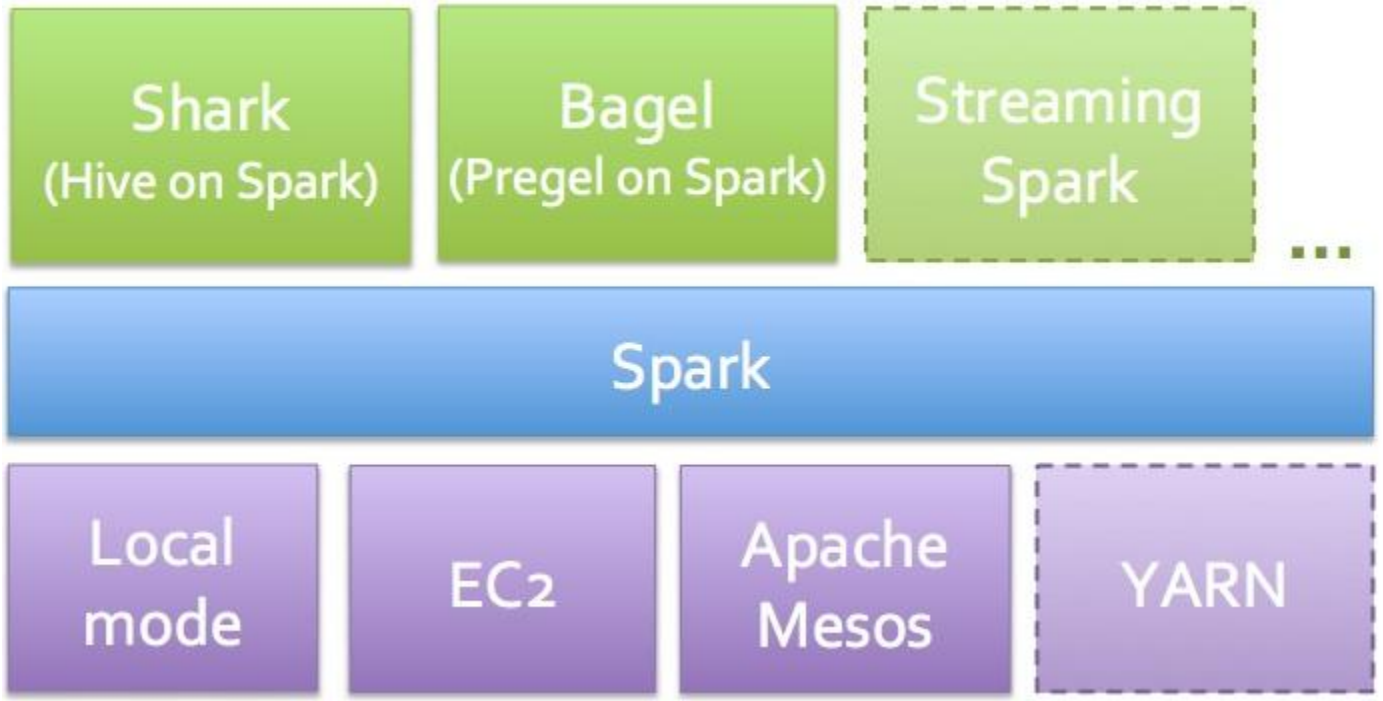
Et si il n'y a pas assez de mémoire?



Fault Tolerance

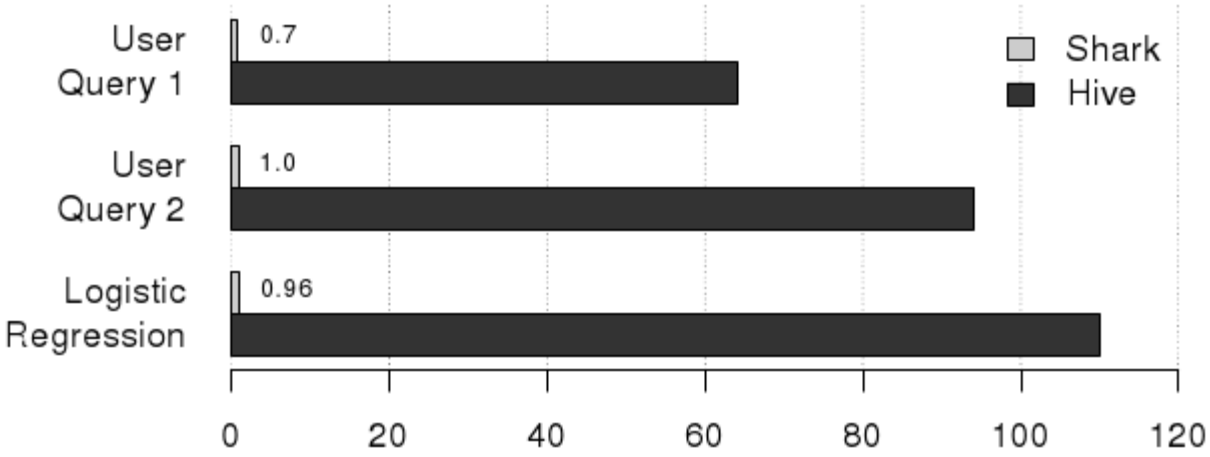


Le monde de Spark



Shark : Hive pour Spark

- HiveQL ~= SQL like
- Compatible avec Hive data (HDFS, HBase) et metastore
- 100x plus rapide que Hive



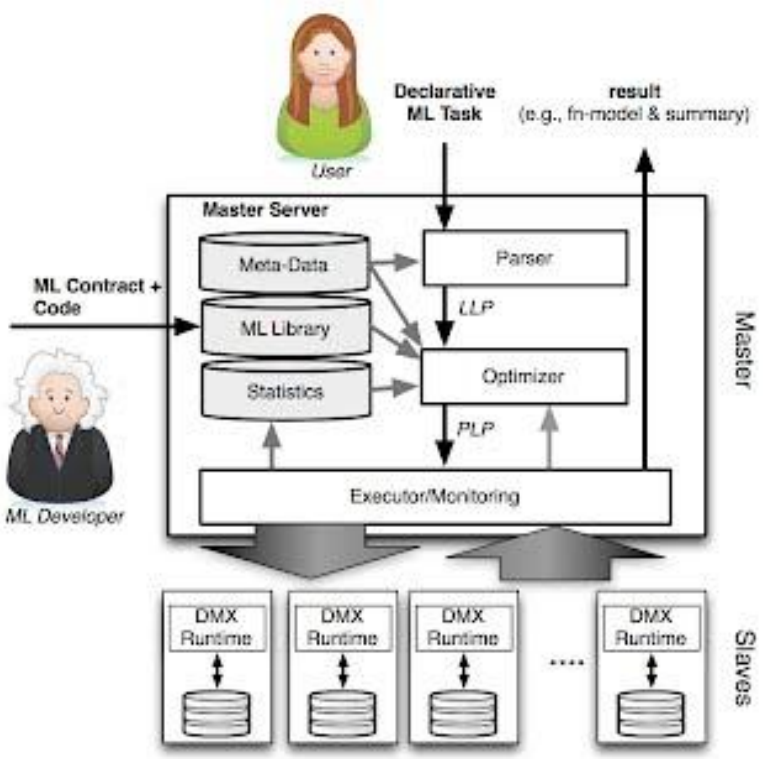
Shark + Spark

- SQL pour la sélection
- Scala pour le traitement

```
val youngUsers = sql2rdd("SELECT * FROM users WHERE age < 20")

println(youngUsers.count)

val featureMatrix = youngUsers.mapRows(extractFeatures(_))
kmeans(featureMatrix)
```

MLBase (dev)

- Système de machine learning basé sur Spark
- Sélection automatique de l'algorithme
- Optimisation des paramètres
- Utilisation interactive



BlinkDB: Queries with Bounded Errors and Bounded Response Times on Very Large Data

```
SELECT avg(sessionTime)
FROM Table
WHERE city='San Francisco'
WITHIN 2 SECONDS
```

Queries with Time Bounds

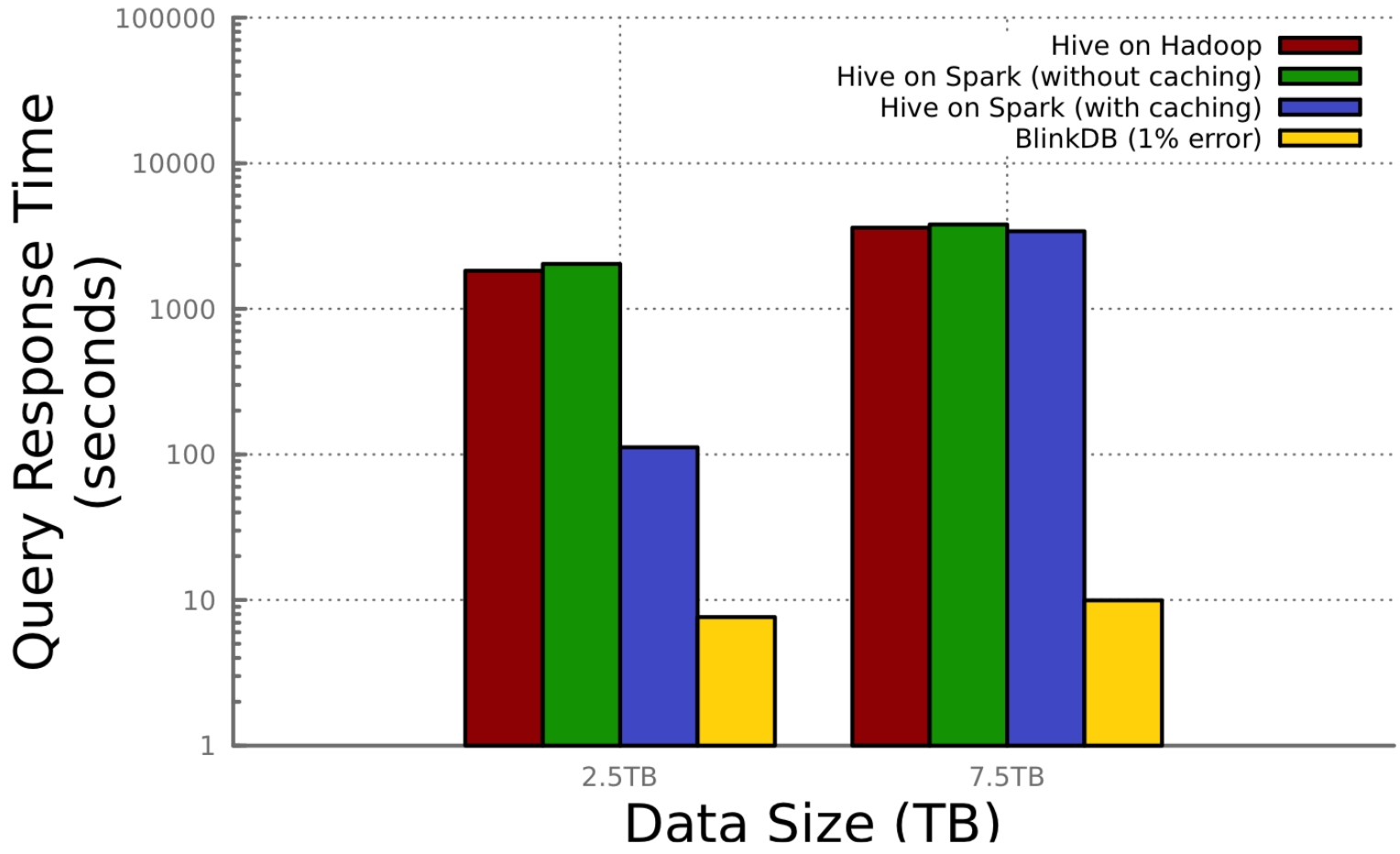
```
SELECT avg(sessionTime)
FROM Table
WHERE city='San Francisco'
ERROR 0.1 CONFIDENCE 95.0%
```

Queries with Error Bounds

amplab
UC Berkeley

BlinkDB

MIT CSAIL



Merci

Questions?