Interactions avec les frameworks map-reduce

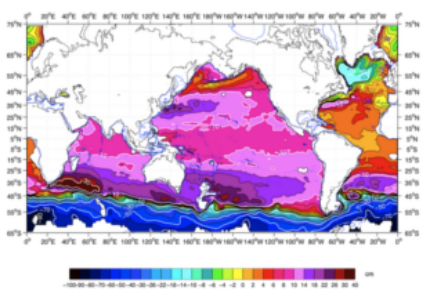
Tristan Le Toullec

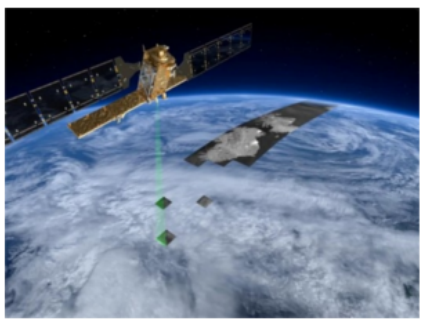
Laboratoire d'Océanographie Physique et Spatiale UMR6523
CNRS/Ifremer/IRD/UnivBrest

15 Décembre 2016

Outline

Tristan Le Toullec



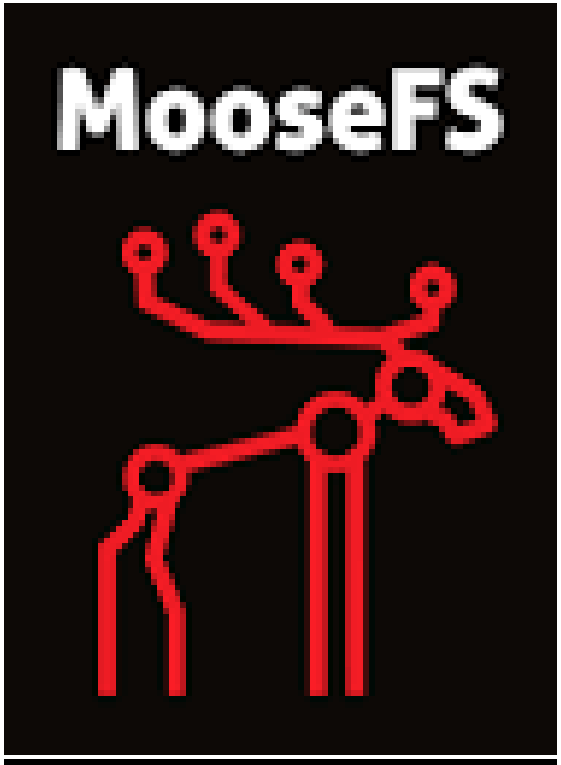


tristan.letoullec@cnrs.fr

Laboratoire d'**O**céanographie **P**hysique et **S**patiale (LOPS)

- Ingénieur Système
- Gestion des ressources de stockage et calcul
- Administrateur cloud OpenStack
- Formateur interne au CNRS

Cercloud



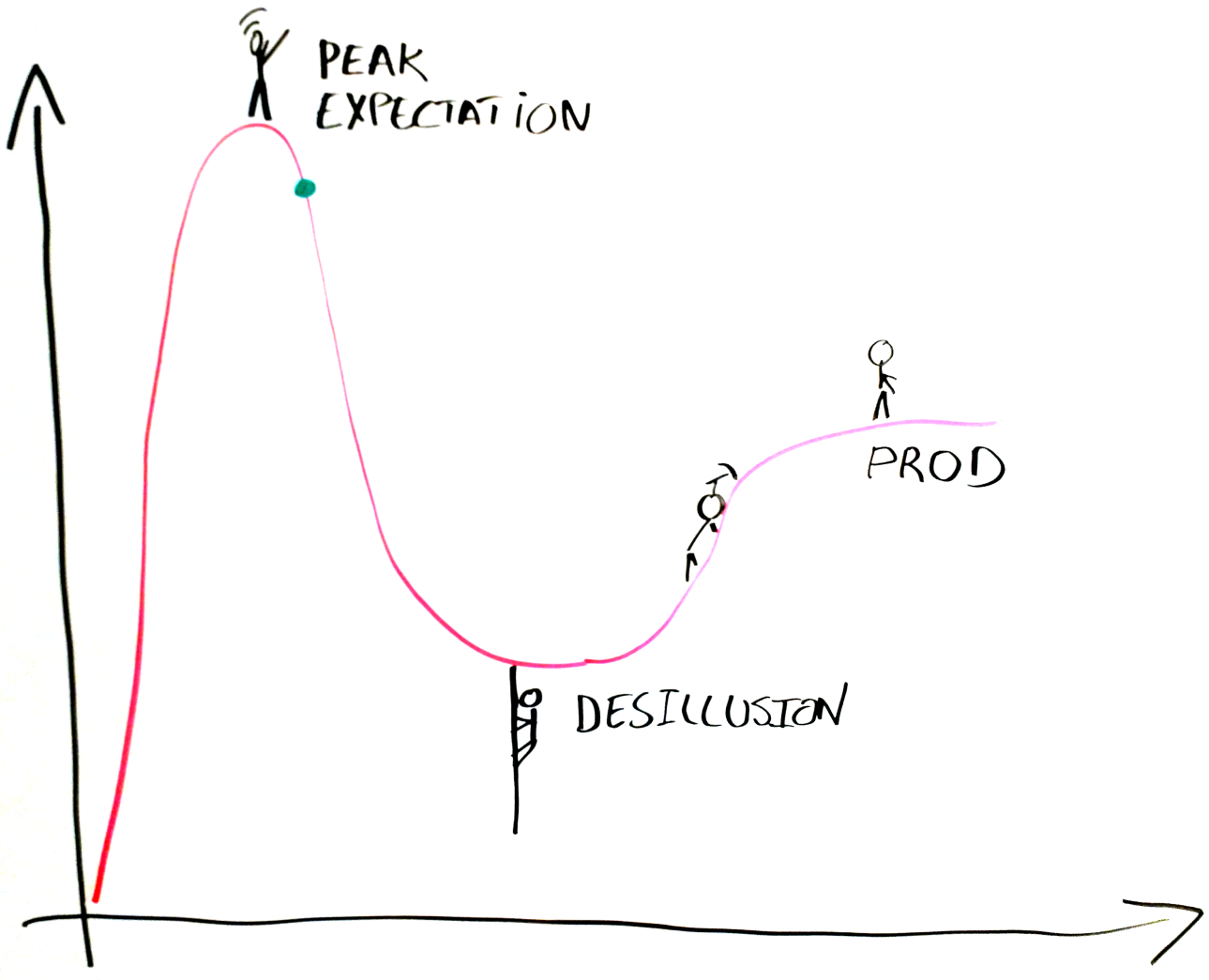
MooseFS

- 3,3 Po et 243 332 057 fichiers
- Scheduling pbs + Gogolist
- 102 noeuds
- Réseau Gigabit
- Metadata en RAM (73 Go)

Hadoop

- 12 noeuds
- 64 To
- Distribution HortonWork
- MapReduce, HBase, Spark

Hadoop on Hype Cycle



Pourquoi ?

L'augmentation des **volumes de données**, de **leurs sources**, les besoins recherche (corrélation, croisement, fouille de donnée). Une recherche d'instantanéité.

- Larges banques de données
- Données plus ou moins affinées
  - Différents format
  - Différentes qualités
- Besoin de monétisation...

Quels moyens ?

- HPC ? Couteux...
- SGBD ? Peu scalable...
- Grilles ? Latences et architectures complexes...

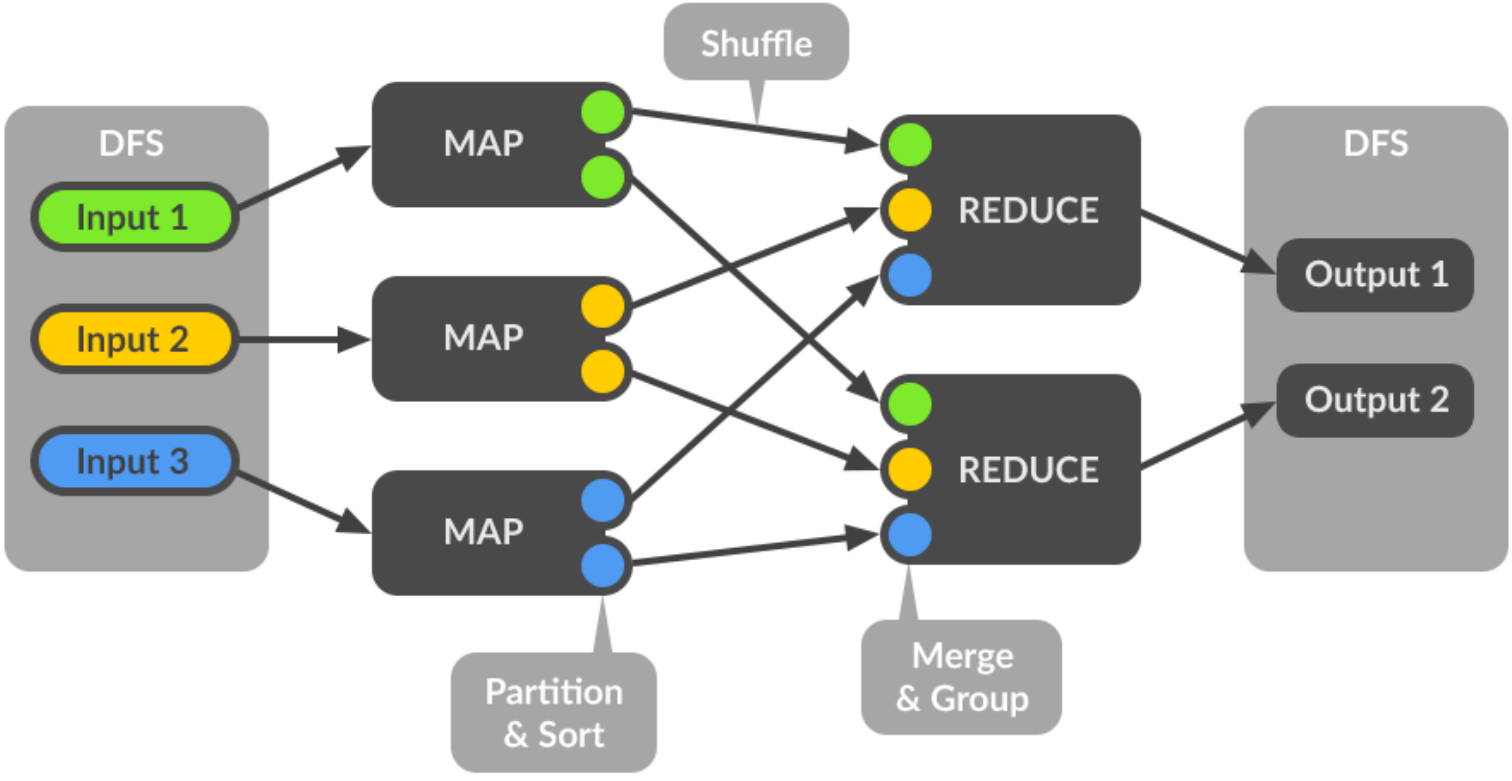
MapReduce

MapReduce est un modèle de programmation issue de travaux de Google

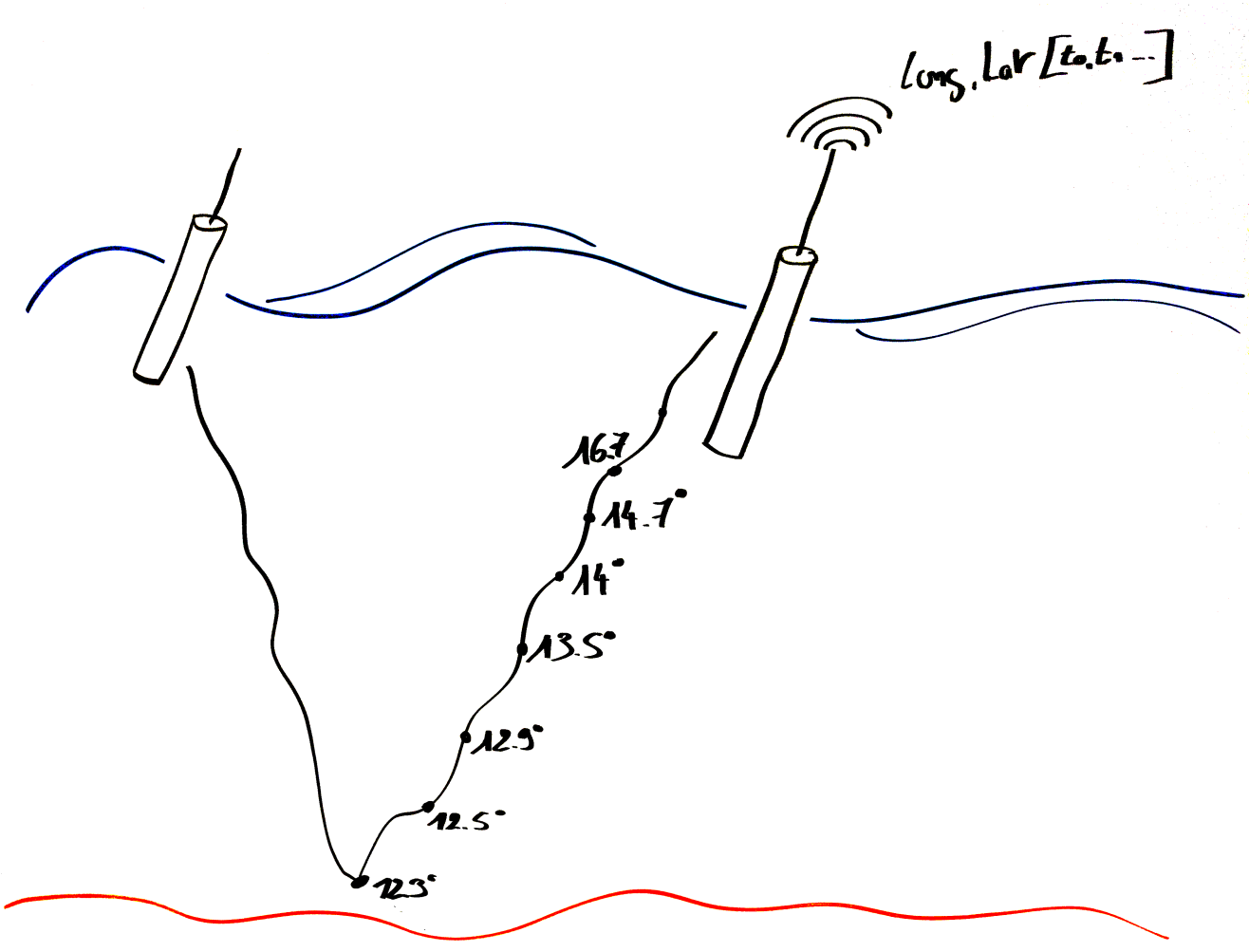
- Traiter d'énorme volume de donnée
- Passage à l'échelle horizontal
- Matériel classique (Commodity Hardware)
- Tolérance à la panne

Deux phases pour le traitement des données Map, puis Reduce.
Tous les problèmes ne peuvent se résoudre avec MapReduce.

MapReduce schema (from Microautomata)



Flotteur ARGO



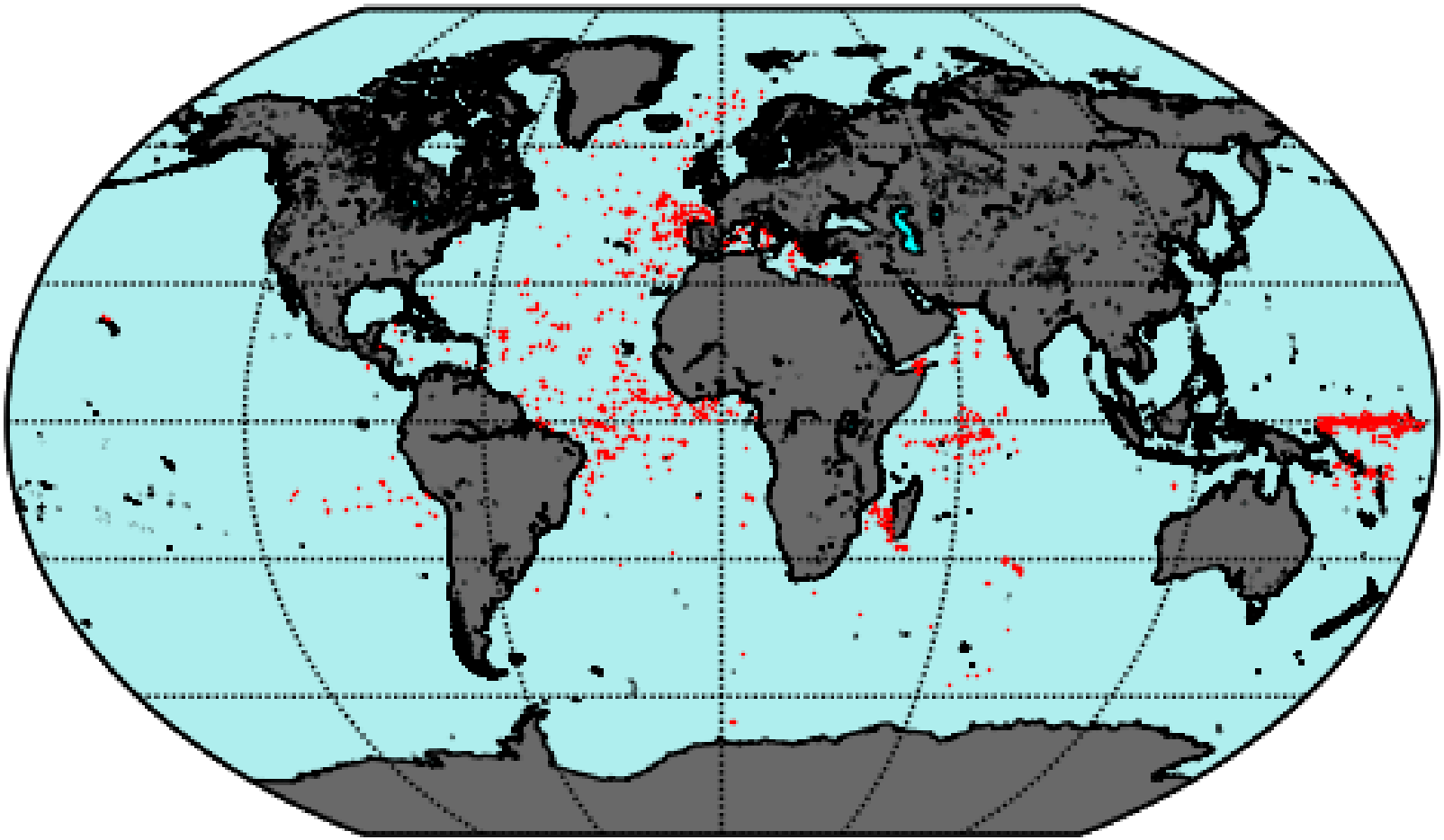
Flotteur ARGO

```
head argo-profiles-cvs-all.csv
-50.853,109.365,1.0,5.83300,0.99068
-50.853,109.365,7.0,5.84800,6.93481
-50.853,109.365,16.0,5.8499,15.85100
-50.853,109.365,26.0,5.8520,25.75787
-50.853,109.365,36.0,5.8530,35.66475
-50.853,109.365,46.0,5.8509,45.57162
-50.853,109.365,56.0,5.8420,55.47850
-50.853,109.365,66.0,5.7960,65.38537
-50.853,109.365,76.0,5.7090,75.29225
-50.853,109.365,86.0,5.5669,85.19912
```

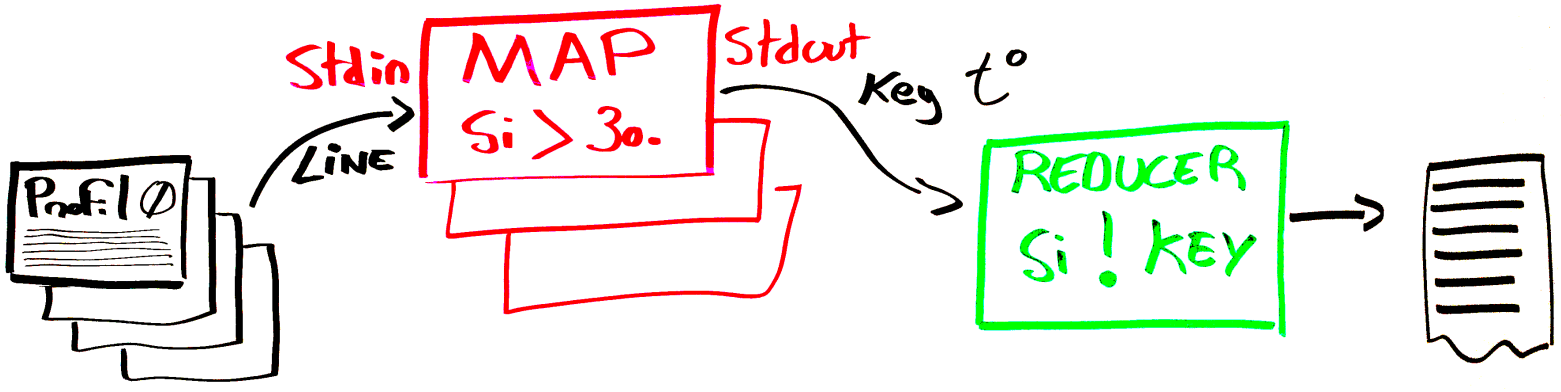
Algorithme classique (simplifié)

```
csvfinder = "./argo-profiles-cvs/*.csv"
profileslist = glob.glob(csvfinder)
print("%s profiles trouvé" % len(profileslist))
match = []
for p in profileslist:
    csvfile = open(p, 'r')
    data = csv.reader(csvfile, delimiter=',')
    for level in data:
        if float(level[3]) > 30.0:
            match.append([level[0],level[1]])
            break
```

Result WorldMap



En MapReduce (principe)

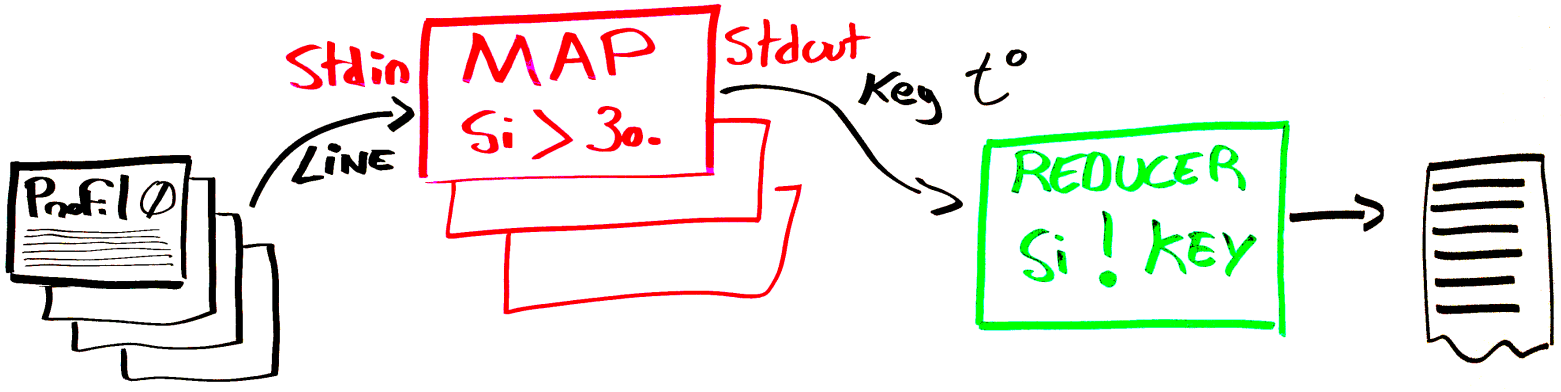


1. Réception des données en streaming (stdin)
2. Map : Si level > à 30 renvoyer les long,lat et temp (stdout)
3. Reduce : Si la clef [long,lat] non encore enregistrés, enregistrement

En MapReduce (Map)

```
import sys
for line in sys.stdin:
    line = line.strip()
    unpacked = line.split(",")
    longitude, latitude, temp = line.split(",")
    results = [longitude, latitude, temp]
    if float(temp) > 30:
                print("\t".join(results))
```

En MapReduce (principe)



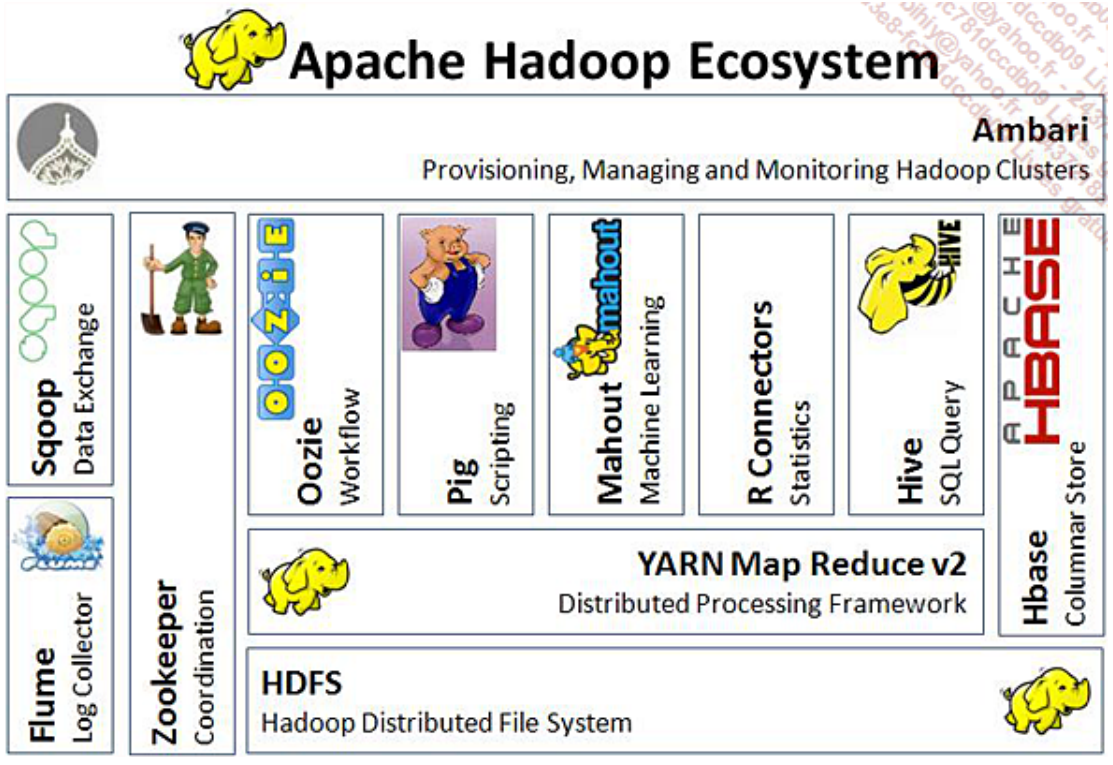
1. Réception des données en streaming (stdin)
2. Map : Si level > à 30 renvoyer les long,lat et temp (stdout)
3. Reduce : Si la clef [long,lat] non encore enregistrés, enregistrement

En MapReduce (Reduce)

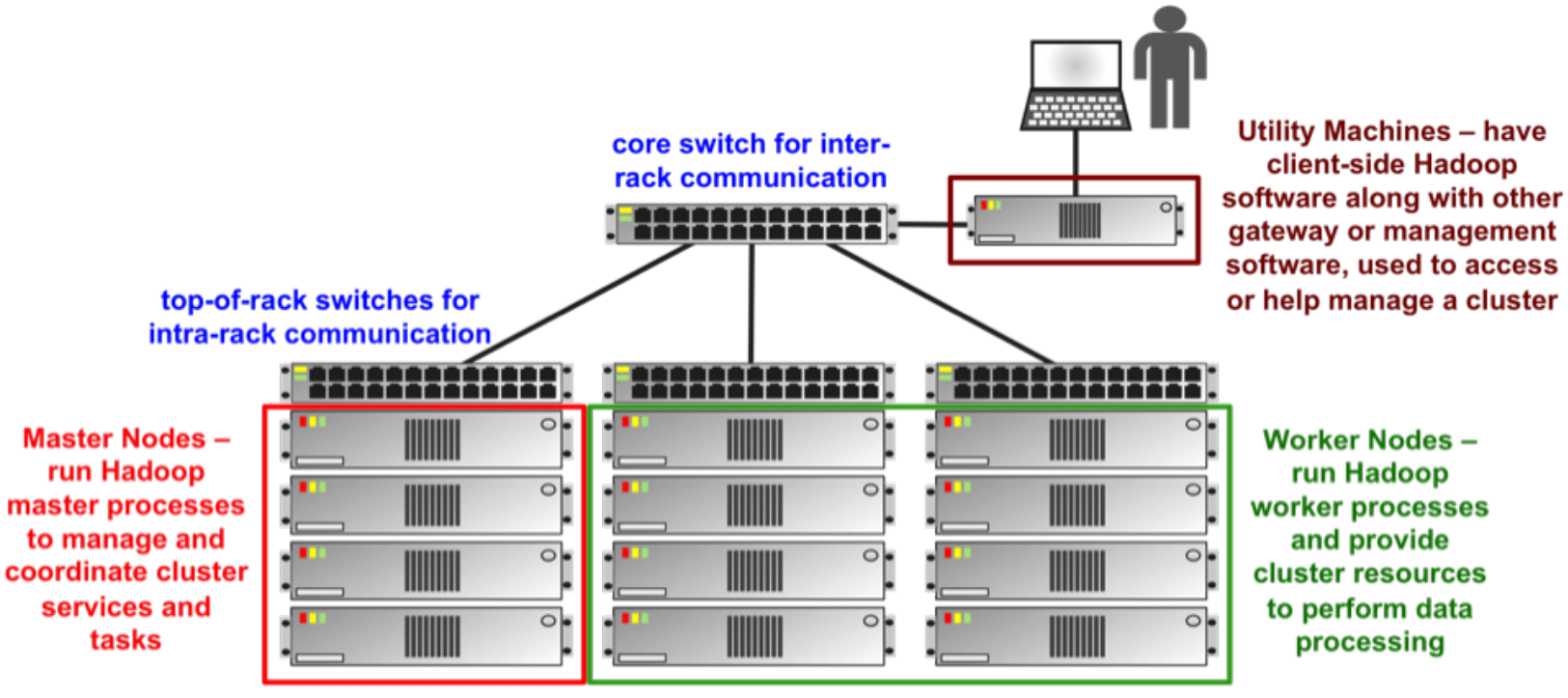
```
import sys
match=[]
for line in sys.stdin:
    line = line.strip()
    longitude, latitude, temp = line.split("\t")
    if not match:
        match = []
    if not ([longitude,latitude] in match):
                match.append([longitude,latitude])

for p in match:
        print("%s,%s" % (p[0],p[1]))
```

Presentation

- MapReduce
- Common
- Yarn
- HDFS



Topologie (from HortonWorks)

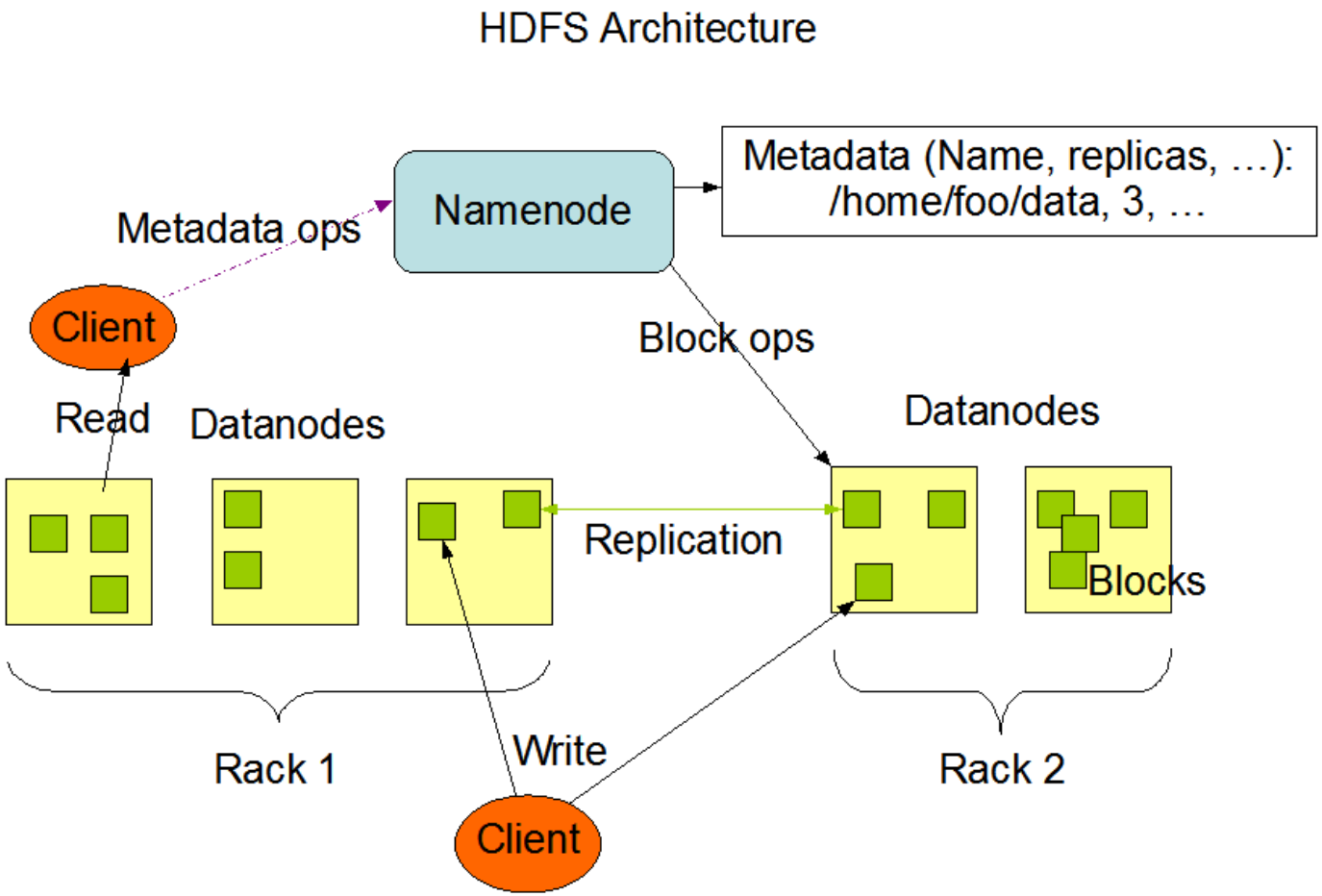


Hadoop Distributed FileSystem

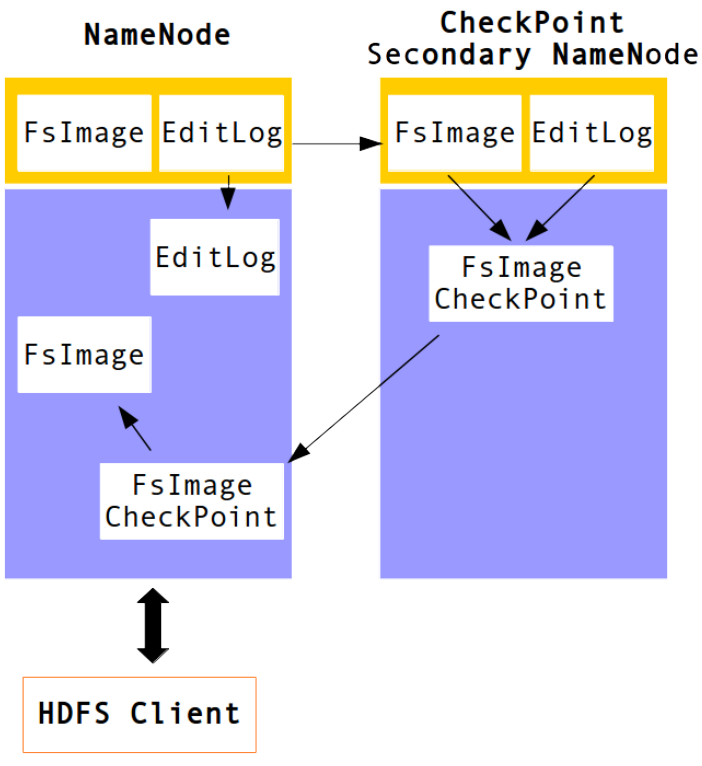
Système de fichier distribué résilient et extensible horizontalement

- Gros blocs
- Réplication (résilience)
- Distribution (scalabilité)
- Optimisation streaming
- Optimisation débit vs accès
- Interface scheduler
- Largesse avec POSIX
- Write Once Read Many

Architecture (from Apache Hadoop)



NameNode stuffs



- Metadonnées en mémoire
- Image sur disque (FsImage)
- Changement sur disque (EditLog)
- Lecture et commit lors du startup
- SecondaryNameNode/CheckPoint assiste le NameNode

Ecriture (en x3)

Lors de l'écriture d'un bloc, le placement sera

- Ecriture sur un datanode d'un rack
- Ecriture sur un datanode d'un autre rack
- Ecriture sur un datanode du même autre rack

Lecture

"Moving Computation is Cheaper than Moving Data"

Une API permet aux logiciels (ie Hadoop MapReduce) d'adapter son comportement.

Rack-Awareness

- Configuration topology.script.file.name
- Renvoi le path de l'host /dc0/rack3
- Si non configuré usage de "/default-rack"

Configuration HDFS (hdfs-site.xml)

```
<property>
 <name>dfs.replication</name>
 <value>3</value>
</property>
<property>
  <name>dfs.namenode.name.dir</name>
  <value>file:/srv/hadoop-data/namenode</value>
</property>
<property>
 <name>dfs.datanode.data.dir</name>
 <value>file:/mnt/chunk1,/mnt/chunk2</value>
</property>
```

Utilisation de HDFS (CLI)

```
hdfs dfs -mkdir /user
hdfs dfs -mkdir /user/hduser
hdfs dfs -copyFromLocal argo-profiles-cvs-all.csv \
        /user/hduser/
hdfs dfs -copyToLocal /user/hduser/output-gt30/001

hdfs dfs -getMerge /user/hduser/output-gt30
```

Passerelles

HDFS NFS Gateway

- Passerelle NFSv3
- Upload et download possible
- Streaming client > passerelle
- Ajout possible mais écriture aléatoire non

API Restfull HTTP (WebHDFS RestAPI)

Specification

Apache Hadoop dispose de son système de fichier, HDFS, mais d'autre FS sont possible

- Local (POSIX)
- BlobStore, stockage object (S3, Swift)
- FS fournissant une implémantation de *org.apache.hadoop.fs.FileSystem*

L'objectif a minima est de répondre aux tests Apache Hadoop et de fournir au framework MapReduce les informations de localité des données.

Système de fichiers HCFS ready

Certains vendor, fournissent le Jar d'implémentation comme :

- HDFS
- GlusterFS
- OrangeFS
- SwiftFS
- GridGain

D'autre non-officiel : BeeGFS, CephFS, Lustre...

Exemple avec CephFS

Download et installation dans le CLASSPATH (HADOOP HOME/lib) du JarFile Ceph

1. fs.default.name : ceph://cephmaster:6789/
2. fs.defaultFS : ceph://cephmaster:6789/
3. ceph.conf.file : /etc/ceph/ceph.conf
4. ceph.auth.keyring : /etc/ceph/client-keyring.key
5. ceph.data.pools : hadoopx3,[hadoopx1]

Conclusion