NoSQL : projet 03/02/16

William Benhaim
Florian Firmin
Paul Todorov
Lucas Weissert

NoSQL

I. Mise en place des serveurs

II. Cassandra-Spark

III. Visualisation

IV. Manipulation

NoSQL

I. **Mise en place des serveurs**

II. **Cassandra-Spark**

III. **Visualisation**

IV. **Manipulation**

NoSQL

Choix d'une architecture SPARK-CASSANDRA pour :

- Explorer des technos qui ont le vent en poupe
- Profiter de 2 méthodes distribuées
- Profiter de la rapidité de la RAM
- Meilleure tolérance à la panne (réplication)
- Profiter de la recherche optimisée (bloomfilter...)

NoSQL

Utilisation d'un cluster AWS de 6 machines :

- Via une image DATASTAX AMI disponible sur AWS
- Instances : m3.xlarge (4 coeurs - 16 Go de RAM - 80 Go SSD)

Creation datastax AMI

```
--clustername wikinosql
--totalnodes 6
--version enterprise
--username <YourUsername>
--password <YourPassword>
--analyticsnodes 6
--cfsreplicationfactor 2
```

Spark UI

Spark Master at spark://172.31.6.151:7077

URL: spark://172.31.6.151:7077
REST URL: spark://172.31.6.151:6066 *(cluster mode)*
Workers: 6
Cores: 18 Total, 0 Used
Memory: 46.4 GB Total, 0.0 B Used
Applications: 0 Running, 185 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers

Worker Id	Address	State	Cores	Memory
worker-20160126125331-172.31.6.149-43412	172.31.6.149:43412	ALIVE	3 (0 Used)	7.7 GB (0.0 B Used)
worker-20160126125331-172.31.6.150-46984	172.31.6.150:46984	ALIVE	3 (0 Used)	7.7 GB (0.0 B Used)
worker-20160126125331-172.31.6.151-59525	172.31.6.151:59525	ALIVE	3 (0 Used)	7.7 GB (0.0 B Used)
worker-20160126125331-172.31.6.152-37745	172.31.6.152:37745	ALIVE	3 (0 Used)	7.7 GB (0.0 B Used)
worker-20160126125332-172.31.6.147-47142	172.31.6.147:47142	ALIVE	3 (0 Used)	7.7 GB (0.0 B Used)
worker-20160126125332-172.31.6.148-42911	172.31.6.148:42911	ALIVE	3 (0 Used)	7.7 GB (0.0 B Used)

Running Applications

Application ID	Name	Cores	Memory per Node	Submitted Time	User	State	Duration

Completed Applications

Application ID	Name	Cores	Memory per Node	Submitted Time	User	State	Duration
app-20160202172619-0264	wiki_month.py	18	2.0 GB	2016/02/02 17:26:19	www-data	FINISHED	10 s
app-20160202171342-0263	wiki_day.py	18	2.0 GB	2016/02/02 17:13:42	www-data	FINISHED	13 s
app-20160202171118-0262	wiki_day.py	18	2.0 GB	2016/02/02 17:11:18	www-data	FINISHED	11 s

NoSQL

I. Mise en place des serveurs

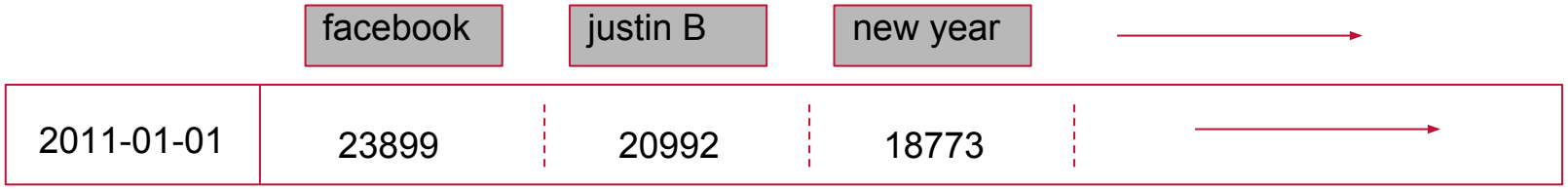
II. **Cassandra-Spark**

III. Visualisation

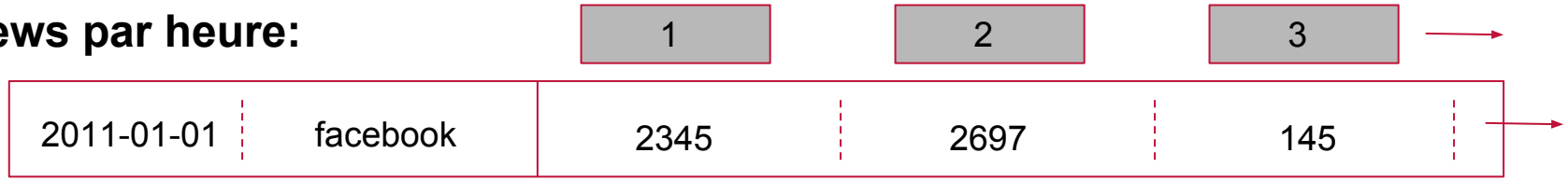
IV. Manipulation

NoSQL - Cassandra - Spark

Top 10 sur 30 jours:



Views par heure:



NoSQL - Cassandra - Spark

```
CREATE KEYSPACE IF NOT EXISTS noSQL_Exams WITH replication = {'class':
'SimpleStrategy','replication_factor' : 2};
```

```
CREATE TABLE IF NOT EXISTS noSQL_Exams.top100_by_day (timestamp int, pagename text,
views int, project text, PRIMARY KEY((timestamp), views, pagename)) WITH CLUSTERING
ORDER BY (views DESC, pagename ASC);

CREATE TABLE IF NOT EXISTS noSQL_Exams.top10_in_30days (timestamp int, pagename text,
views int, project text, PRIMARY KEY((timestamp), views, pagename)) WITH CLUSTERING
ORDER BY (views DESC, pagename ASC);

CREATE TABLE IF NOT EXISTS noSQL_Exams.viewsPage_day (timestamp int, pagename text,
views int, project text, PRIMARY KEY((pagename), timestamp) );

CREATE TABLE IF NOT EXISTS noSQL_Exams.viewsPage_hour (timestamp int, pagename text,
views int, project text, hour int, PRIMARY KEY((pagename, timestamp), hour) );
```

NoSQL - Cassandra - Spark

Lecture des fichiers :

```
# We keep only file started with page
onlyfiles = [f for f in listdir(PATH) if f.startswith("page")]
```

Fonctions utiles :

```
def timestamp_to_minutes(timestampData, formatTimestamp):
    # first date 1970 01 01
    epoch = datetime.utcfromtimestamp(0)
    # get time in the correct format
    timeT = datetime.strptime(timestampData, formatTimestamp)
    # return number of minutes
    return (timeT - epoch).total_seconds() / 60
```

```
def line_to_dict(project, pagename, timestamp, views):
    return Row(
        project=project,
        pagename=pagename,
        timestamp=timestamp_to_minutes(timestamp,FORMAT_DATE),
        views=int(views))
```

Mapping de Spark a Cassandra :

```
rdd_day = sc.textFile(strFiles).map(lambda line: line.split(' '))
                               .filter(lambda line: int(line[2]) > 10)
                               .map(lambda line: ((line[0],line[1]), int(line[2])))
                               .reduceByKey(lambda value1, value2: int(value1) + int(value2))
                               .map(lambda x: (x[1],x[0]) )
                               .sortByKey(False)
                               .map(lambda x: (x[1],x[0]))
                               .map(lambda x: line_to_dict(x[0][0], x[0][1], keys_file,x[1]))
```

NoSQL - Cassandra - Spark

Récupérations des données :

```
df30 = sqlContext.read.format("org.apache.spark.sql.cassandra").options(keyspace='nosql_exams', table='top100_in_30days').load();
df30 = df30.filter(df30.timestamp >= int(timeMinutes)).filter(df30.timestamp <= int(timeMinutes))
df30.toPandas().to_csv("/var/www/wiki_month.csv",mode = 'w', index=False)
```

Données pour le trend

```
dfN = df.filter(df.timestamp >= int(start))
        .filter(df.timestamp <= int(end))
        .join(df30, (df.pagename == df30.pagename) & (df.project == df30.project), 'inner')
        .drop(df30.pagename).drop(df30.views).drop(df30.timestamp).drop(df30.project)
        .toPandas()
        .pivot(index='timestamp', columns='pagename', values='views')
        .fillna(0)
        .reset_index()
dfN["timestamp"] = dfN["timestamp"].apply(lambda x : datetime.fromtimestamp(x * 60)).apply(lambda x : str(x)[0:10])
dfN = dfN.rename(columns={'timestamp': 'date'})
dfN.to_csv("/var/www/wiki_month_line.csv",mode = 'w', index=False)
```

NoSQL

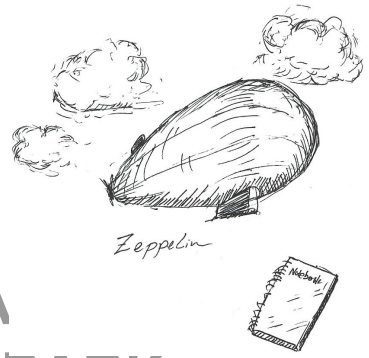
I. Mise en place des serveurs

II. Cassandra-Spark

III. **Visualisation**

IV. Manipulation

NoSQL - Visualisation



Choix initial : utiliser ZEPPELIN

- Essai en local -> cluster AWS et sur le master A
  - -> Echec de liaison de Zeppelin au master SPARK

Choix alternatif :

- page WEB sur un serveur WAMP : Apache2+PHP
- Lancement du script PySpark via une requête PHP
- Affichage via D3JS

NoSQL - Visualisation

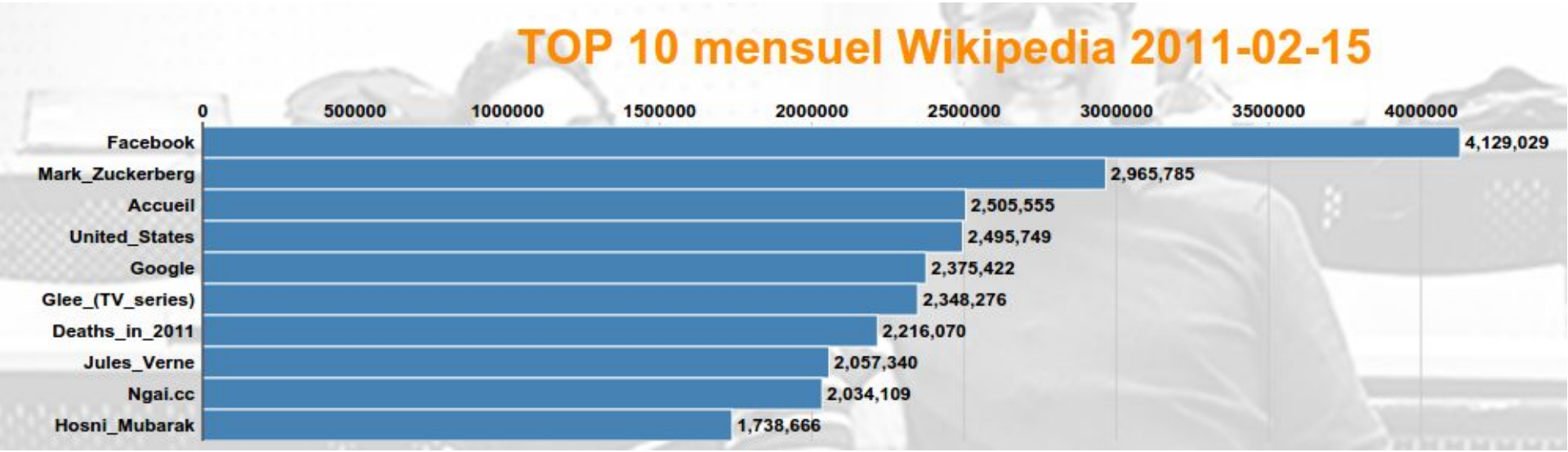
Récupération de la date

Chargement des données

Affichage d'un chart

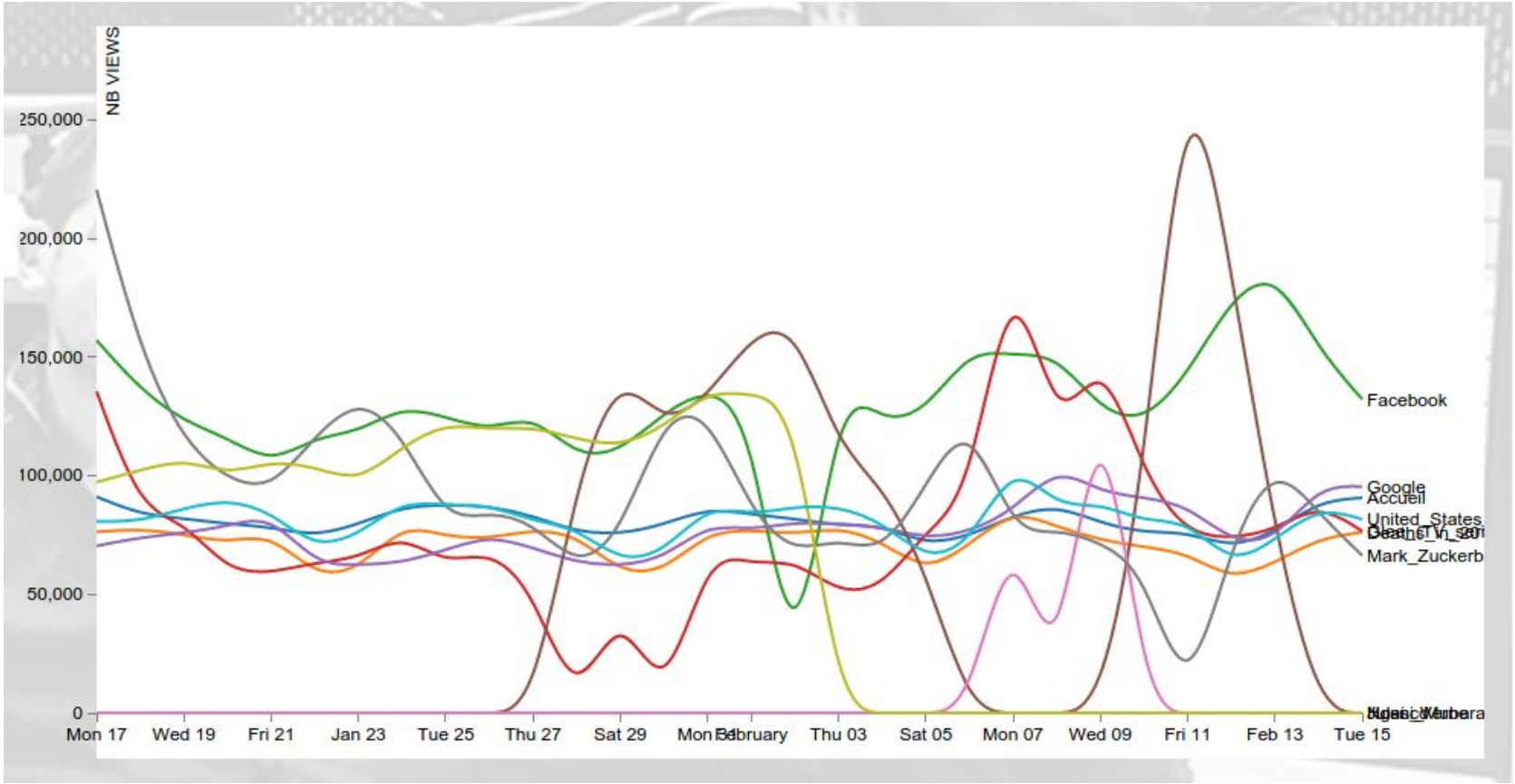
NoSQL - Visualisation

Affichage du top 10 mensuel / quotidien :



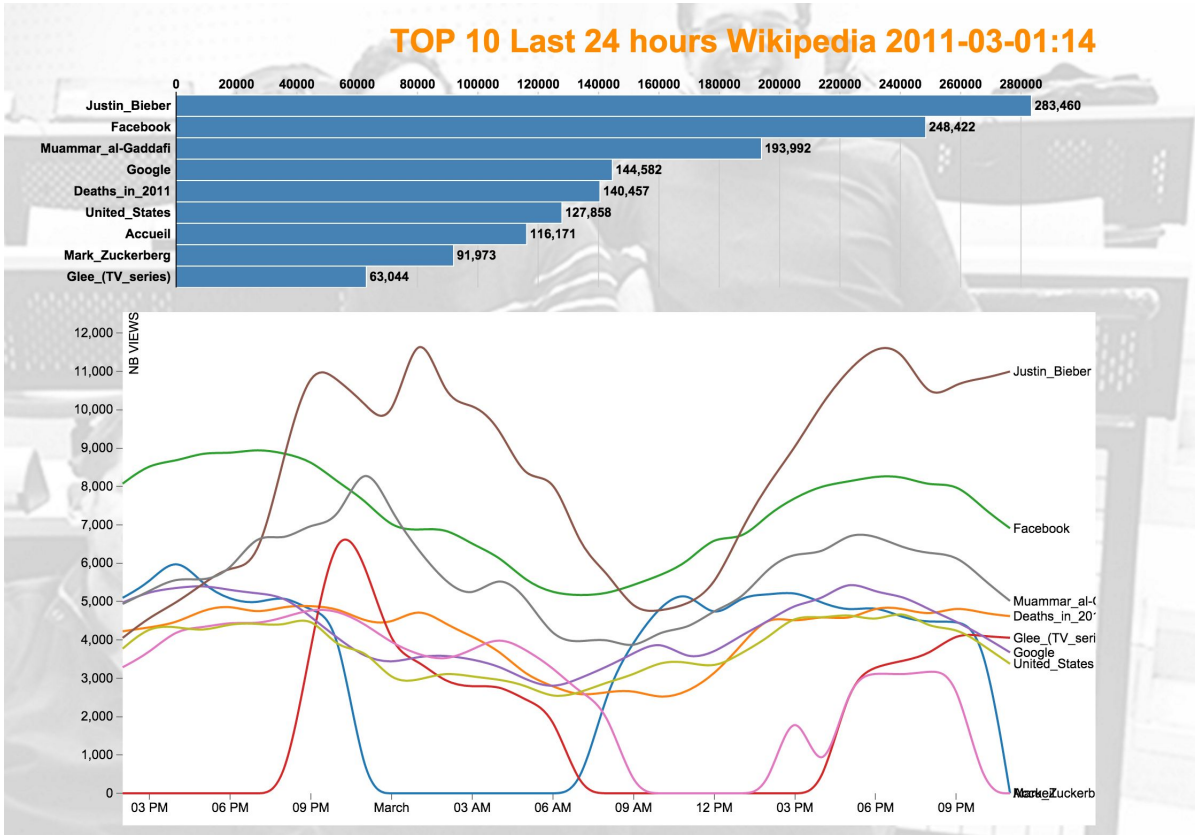
NoSQL - Visualisation

Affichage du Trend :



NoSQL - Visualisation

Affichage de la page: (Sur-brillance au contact du nom ou de la courbe)



NoSQL

I. Mise en place des serveurs

II. Cassandra-Spark

III. Visualisation

IV. **Manipulation**