

SMACK: une pile logicielle open source pour le traitement big data

Mathieu Goeminne

Un mot sur le CETIC



Secteurs d'activité

- Ingénierie logicielle / qualité logicielle
- Électronique / Embarqué
- Traitement des données / Infrastructure



Un mot sur l'orateur

- PhD (software engineering) sur les écosystèmes open source
- Cetic depuis 2016 > Équipe Data
 - Projets Big Data et Data Science
 - Manufacturing et transport

Le big data au sein des entreprises

Des données toujours plus nombreuses,
toujours variées, toujours rapides

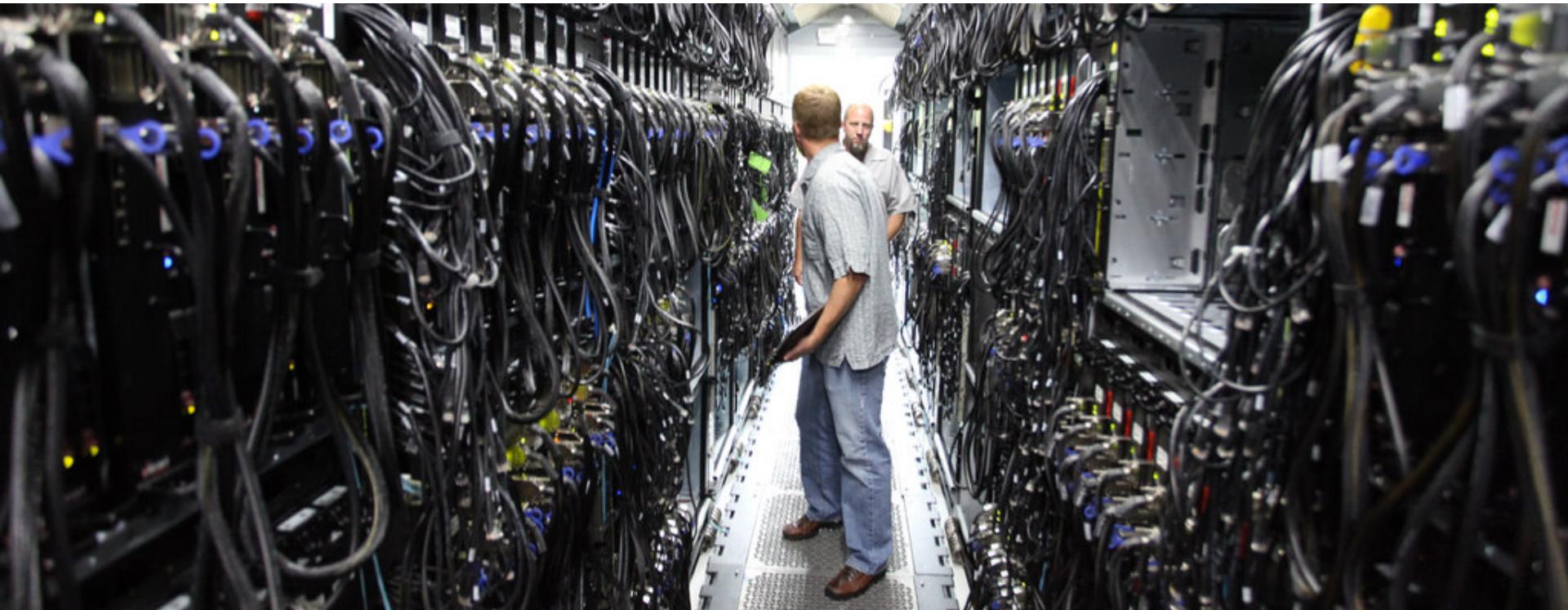
Manufacturing

- $(1-100) \times 1000$ capteurs @ $(1 \text{ min} - 1 \text{ sec})^{-1}$
- Mesures labo, ERP, etc.
- Multiples sites (AGC Europe = 28 implantations)



Data Center

- Des milliers serveurs (OVH: 260 000)
 - Des dizaines de paramètres / serveur
- Des millions de sites hébergés (OVH: 18 000 000)
- (10-100-1000) messages de log applicatifs / seconde / site
- Multiples implantations (OVH: 20)





SMACK

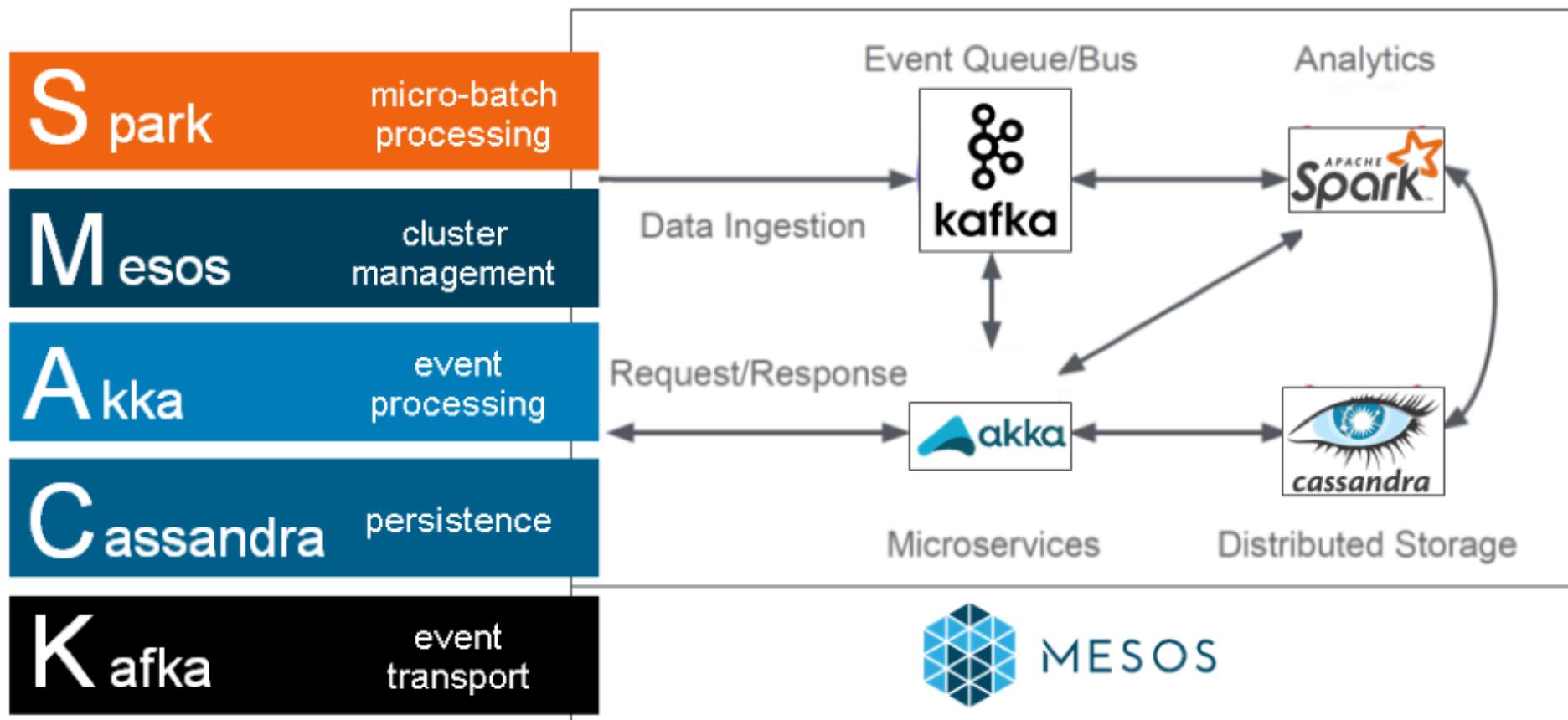
Un ensemble d'outils logiciels pour le traitement de données à large échelle



Quelques propriétés

- Projets Apache (à l'exception de Akka)
- Répondre aux besoins suivants:
 - **Passage à l'échelle:** de plusieurs processeurs à plusieurs *data centers*
 - **Résilience:** N'importe quel élément peut tomber
 - **Réplicabilité:** On doit pouvoir refaire un travail à n'importe quel moment, et obtenir le même résultat
 - **Efficacité:** La donnée perd rapidement de sa valeur, donc un traitement doit être réalisé dans les délais.
 - **Gestion intégrée de plusieurs workflows et plusieurs sources de données:** typiquement, traitement de données historiques et de données « fraîches »

Vue globale de l'architecture



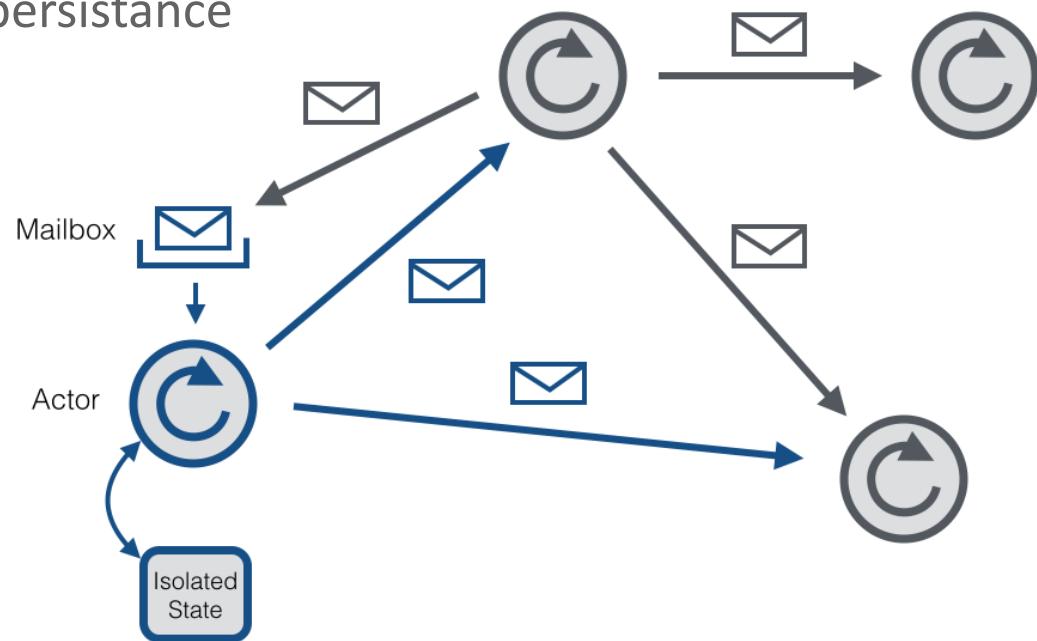
1. Akka

Pour la collecte de données



Akka pour la collecte de données

- Système de traitement distribué
 - Par échange de message, plutôt que par état partagé
 - **Asynchrone, relocalisable, passe à l'échelle.**
- Dans SMACK, chaque acteur représente un petit **service de collecte**
 - Reçoit une donnée (ex: via un service REST)
 - Traitement minimal (ex: validation du format)
 - Transmission à un service de persistance



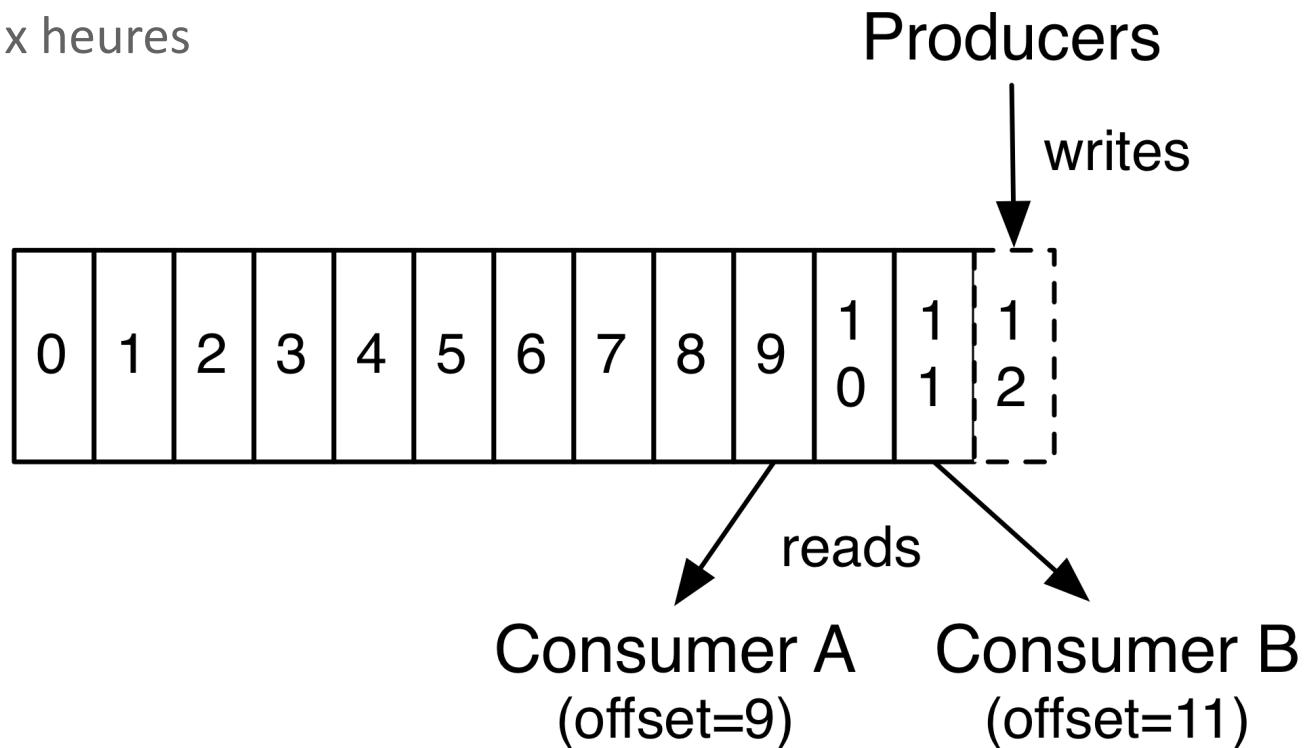
2. Kafka

Pour le transport des messages



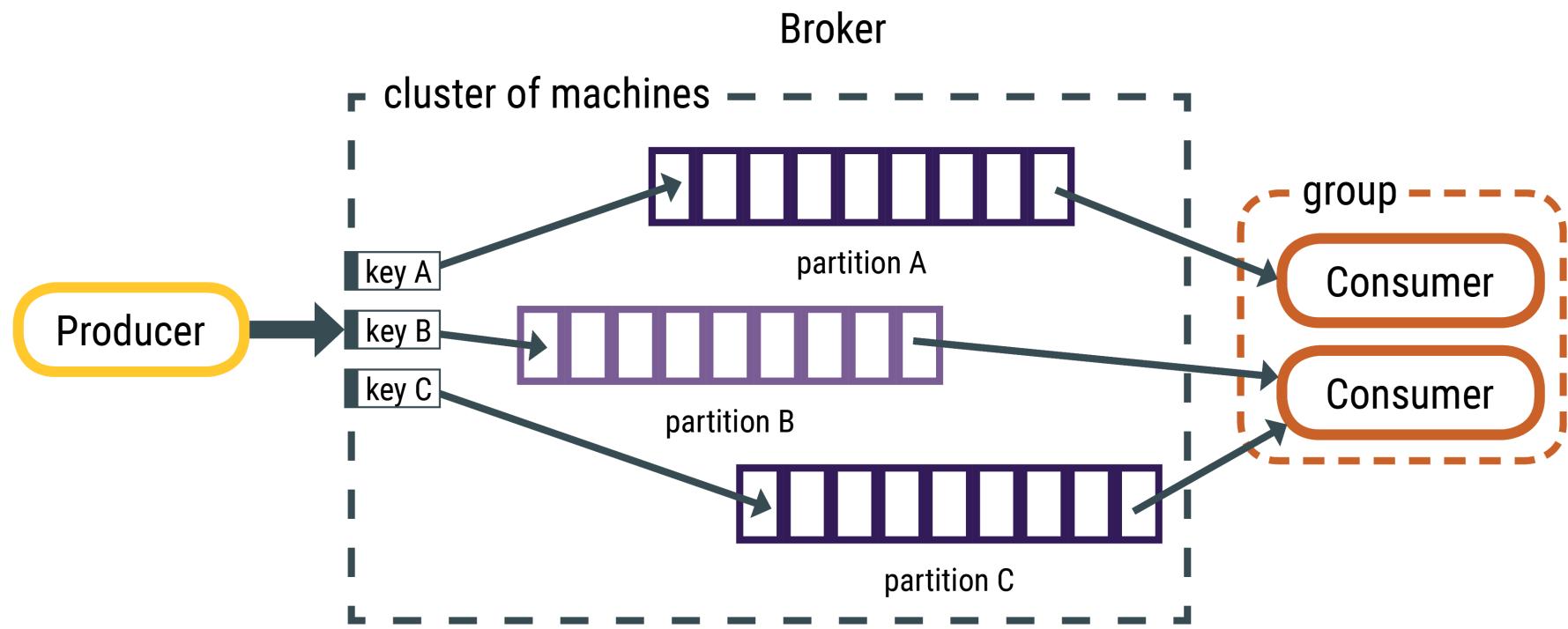
Kafka pour le transport des messages

- Producteurs — Consommateurs
- Optimisé pour écrire à grande vitesse (2 millions d'écritures / seconde sur 3 machines standard).
- Organisation des messages en **topics**, chacun d'eux étant géré par un **cluster de machines** (réplication, résilience, etc.)
- Persistance pendant x heures

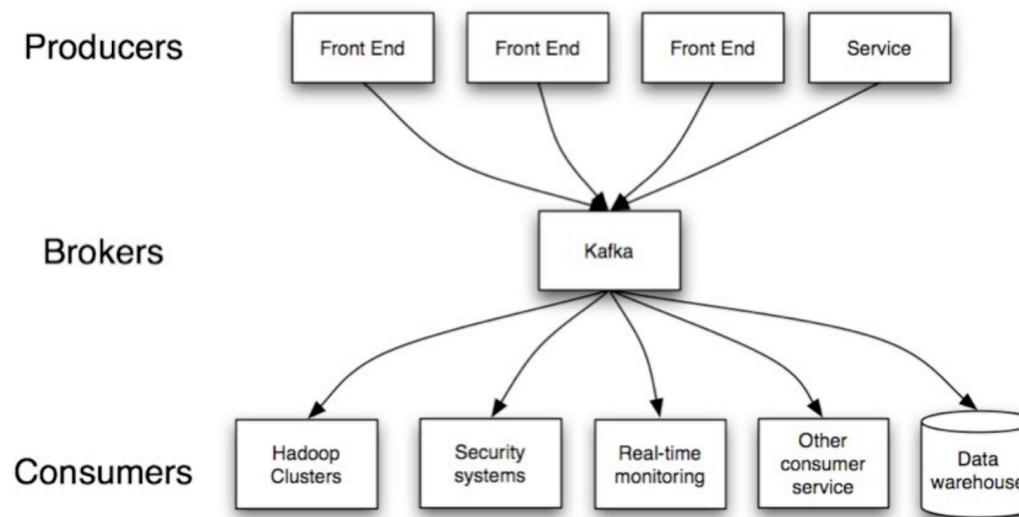
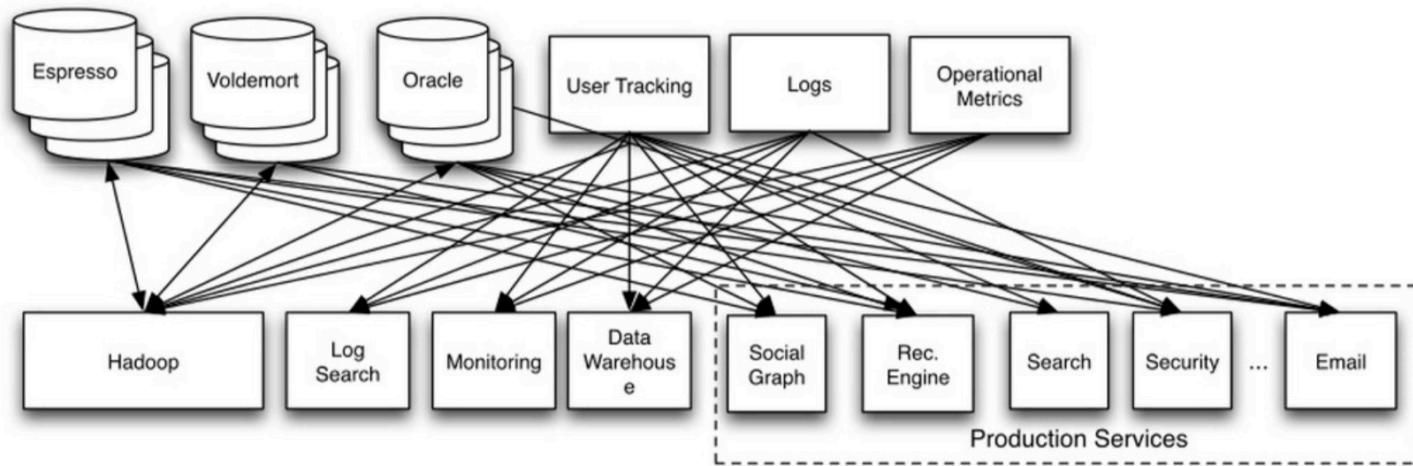


Kafka pour le transport des messages

- Chaque topic est divisé en **partitions**
- Les partitions sont réparties sur plusieurs nœuds
 - Répartition des messages selon une **clé de partition**
 - Un nœud leader pour chaque partition
- Les consommateurs lisent une ou plusieurs partitions de leurs topics d'intérêt



Kafka en tant que point central de communication



3. Cassandra

Pour le stockage distribué



Cassandra pour le stockage

- Base de données NoSQL: pas votre SGBD habituel
 - Optimisée pour l'**écriture** à grande vitesse
 - Dictionnaire de dictionnaires de valeurs
 - Accès obligatoire par la clef de ligne, puis optionnellement par clef de colonne
 - Chaque ligne peut avoir des colonnes différentes
- BDD **distribuée**, masterless
 - Chaque instance possède un sous-ensemble des lignes
 - Les lignes sont dupliquées (typiquement: 3x): vitesse et résilience
- Alimentée par un agent (acteur Akka), qui consomme un topic Kafka.

Cassandra: stockage de séries chronologiques

```
CREATE TABLE event (
    tag text,
    time timestamp,
    value double,
    quality double,
)
PRIMARY KEY (tag, time)
```

```
SELECT *
FROM event
WHERE tag IN ('sensor1', 'sensor2') AND
      time >= '2018-01-12 12:00:00' AND
      time < '2018-01-13 00:00:00';
```

Clef primaire composée.

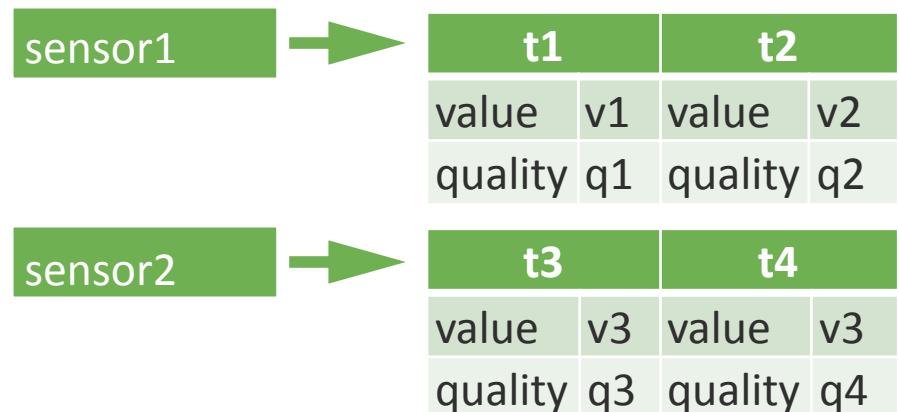
tag=**clef de partition** (de ligne)
timestamp=**clef de clustering**

Pour chaque tag, les timestamps sont ordonnés.

Ce qu'on voit

tag	time	value	quality
sensor1	t1	v1	q1
sensor1	t2	v2	q2
sensor2	t3	v3	q3
sensor2	t4	v4	q4

Ce qui est stocké



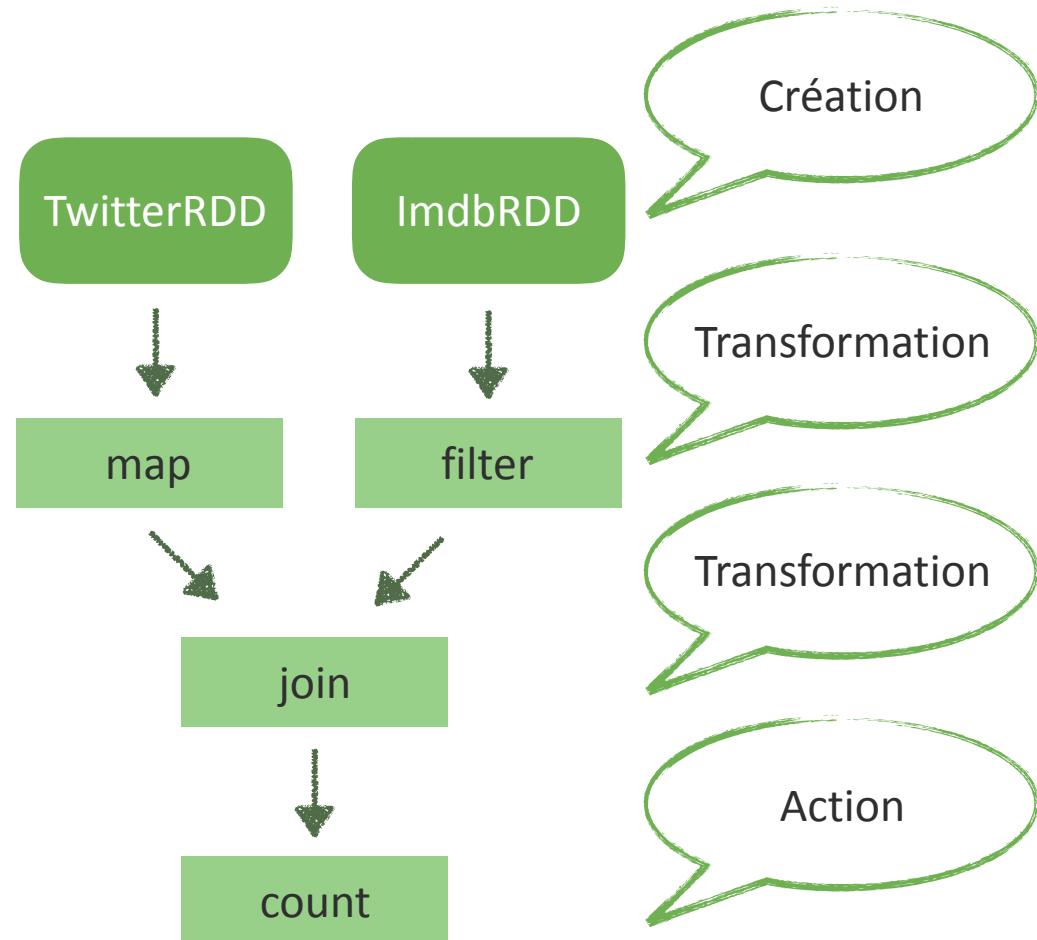
4. Spark

Pour le traitement des données

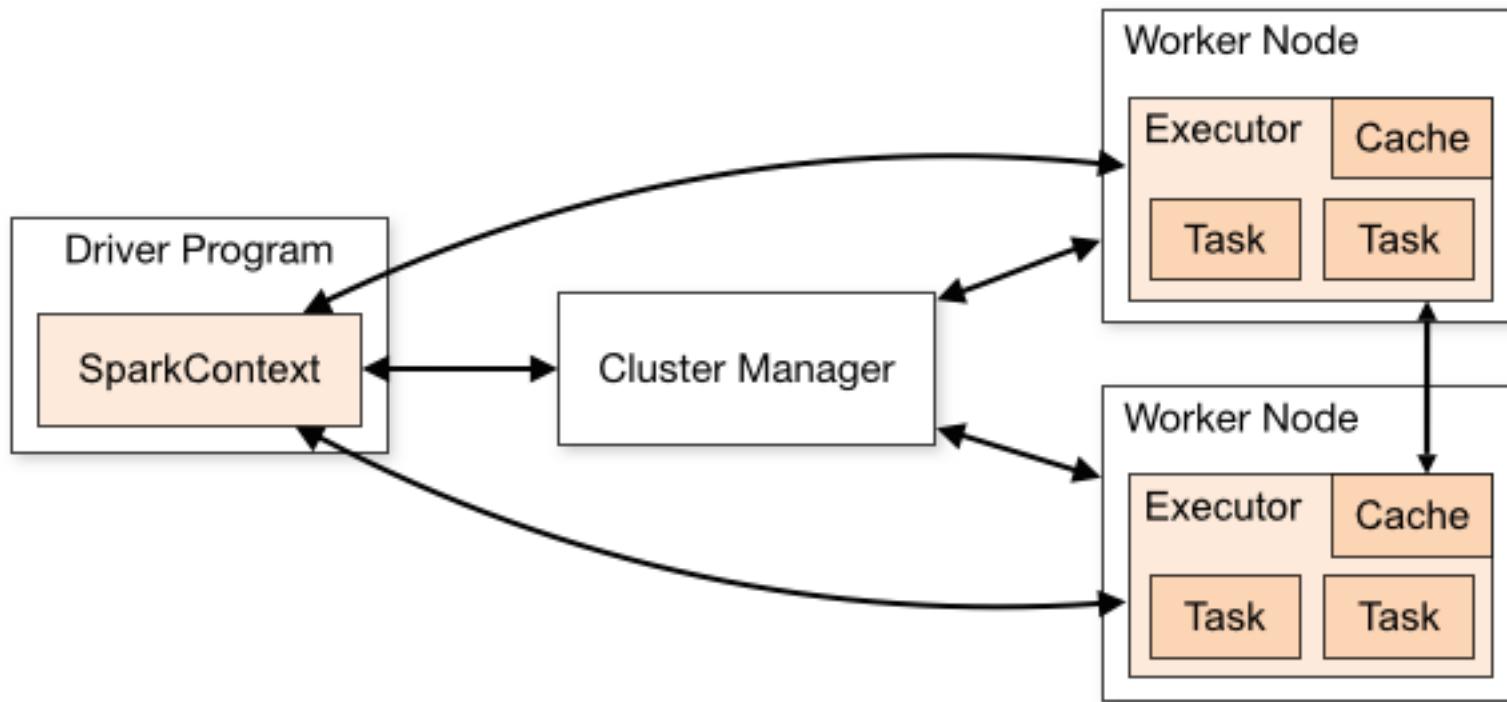


Spark pour le traitement des données

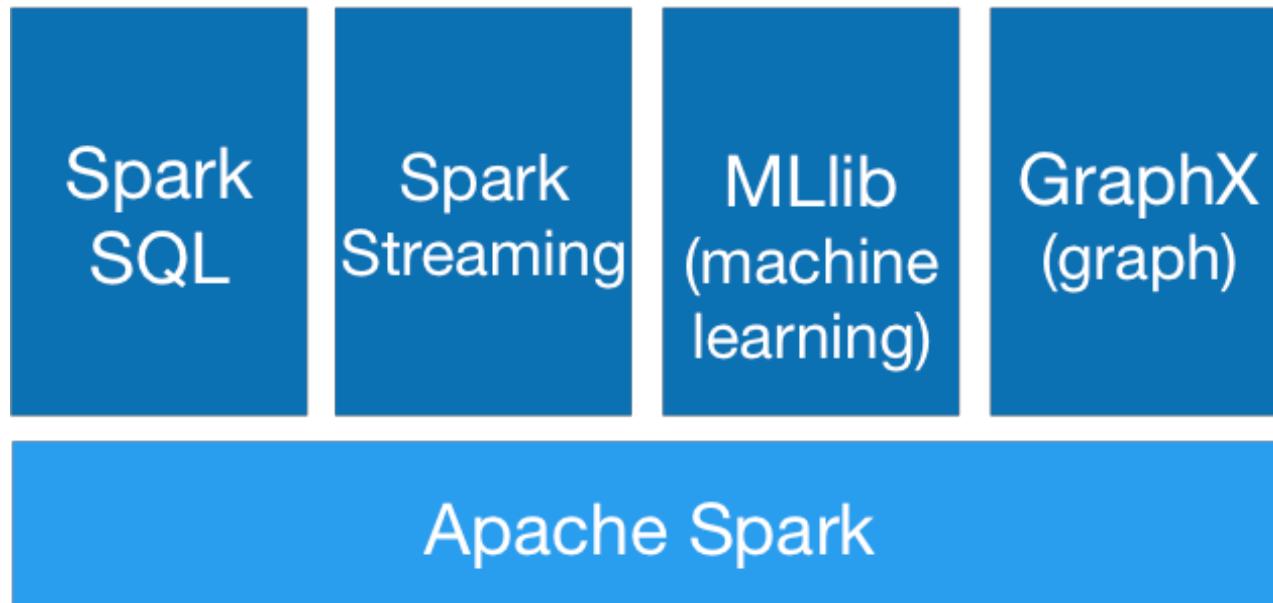
- Resilient Distributed Dataset
 - Collections immuables
 - Partitions distribuées sur le cluster
- Transformations
 - Appliquées sur chaque partition
 - Programmation fonctionnelle
 - filter, map, reduce, ...
- Actions
 - Rapatriement sur le driver



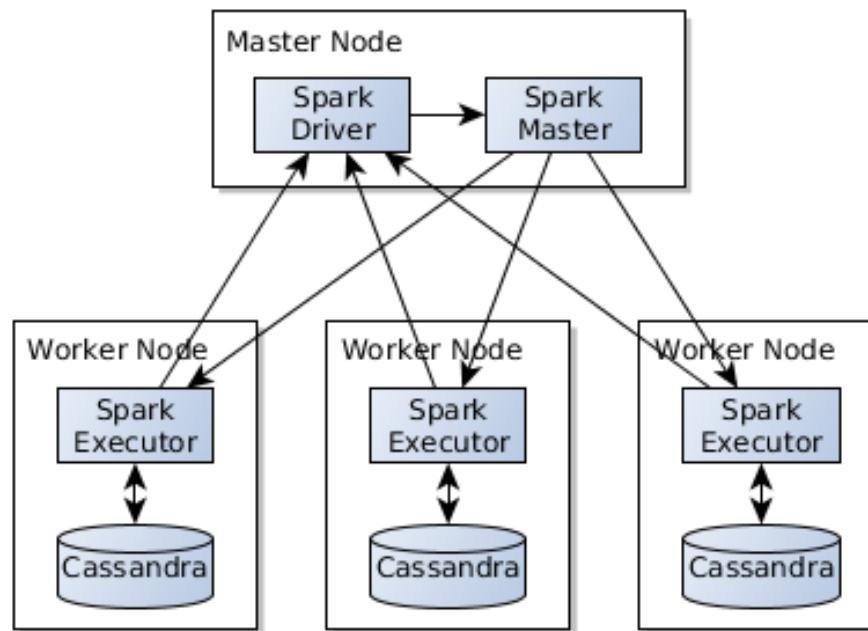
Spark : Cluster Mode



Spark : une plateforme pour les gouverner tous



Spark ❤️ Cassandra



```
val df = spark
  .read
  .cassandraFormat("events", "test")
  .load
```

```
val dfWithPushdown = df.filter($"tag" === "sensor1") and ($"time" >= "2018-01-12")
```

```
val nLines = dfWithPushdown.count
```

5. Mesos

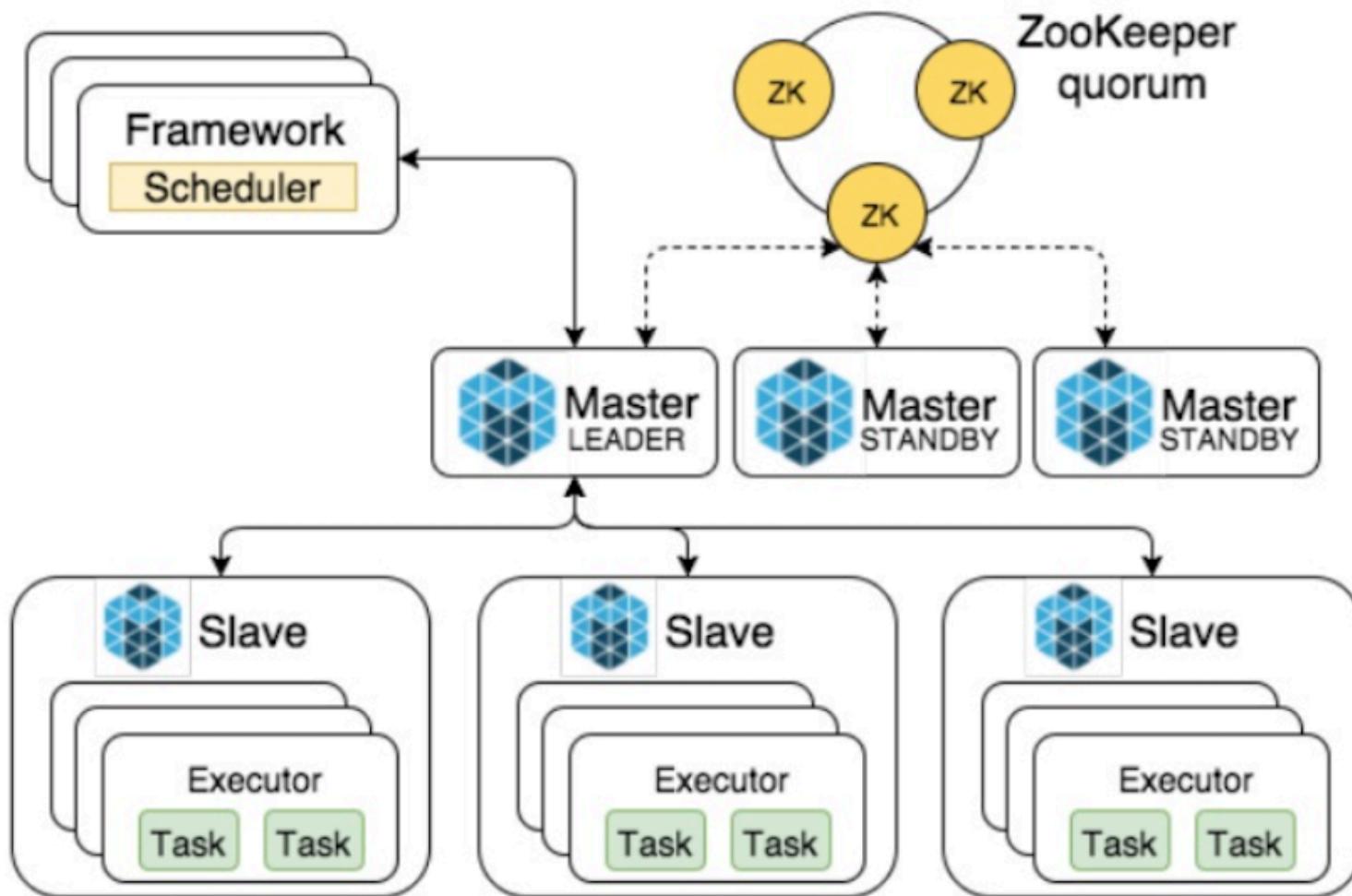
Pour gérer les ressources du cluster



Mesos pour gérer les ressources du cluster

- **Efficace**
 - Partage efficace des ressources entre les applications
- **Flexible**
 - Supporte de nombreux frameworks actuels et à venir
 - Supporte des demandes de ressources de différentes natures
- **Passe à l'échelle**
 - Des clusters de quelques nœuds à des milliers de nœuds
- **Robuste**
 - Tolérance aux erreurs: si ça plante, on recommence sur le nœud d'à côté
 - Le scheduler lui-même est redondant

Mesos pour gérer les ressources du cluster



Démonstration!

<https://youtu.be/QzJI4FeXns4>



Your Connection to **ICT** Research

Aéropole de Charleroi-Gosselies
Avenue Jean Mermoz 28
B-6041 Charleroi - Belgique



twitter.com/@CETIC twitter.com/@CETIC_be



linkedin.com/company/cetic



info@cetic.be



+32 71 490 700

www.cetic.be

Merci

Mathieu Goeminne

mathieu.goeminne@cetic.be