

# Systèmes Distribués pour le Traitement de Données

Thomas Ropars

[thomas.ropars@univ-grenoble-alpes.fr](mailto:thomas.ropars@univ-grenoble-alpes.fr)

<http://tropars.github.io/>

2018

# Contexte

## Analyse de données

- Des systèmes de traitement de données sont utilisés dans tous les secteurs.
- Objectif: extraire de la valeur des données

## Exemples d'utilisation

- Analyse (temps réel) de l'opinion publique
- Analyse (temps réel) des données de capteurs
- Analyse (temps réel) de données de monitoring
- Formulation de recommandations (réseaux sociaux)

# L'émergence de l'apprentissage (Machine Learning)

## Apprentissage

- Moyen d'extraire de la valeur des données

## 3 briques principales

- Les algorithmes d'apprentissage
- La brique Big Data
  - ▶ La tuyauterie permettant d'analyser de très grandes quantités de données
- Le Cloud
  - ▶ L'infrastructure fournissant les ressources nécessaires au déploiement de l'application

# Objectifs du projet

Ceci n'est pas un projet sur l'apprentissage et/ou  
l'intelligence artificielle

Ceci est un projet sur les systèmes distribués

L'objectif est de construire et d'étudier l'infrastructure distribuée permettant de mettre en place des algorithmes d'apprentissage.

# Objectifs du projet

- Savoir configurer et utiliser une pile logicielle permettant d'analyser de grandes quantités de données
- Savoir construire une infrastructure (*dans le nuage*) et déployer une pile logicielle sur cette infrastructure
- Comprendre et évaluer les capacités et les limites d'une infrastructure distribué (disponibilité, performance)

# Composants classiques des infrastructures Big Data

- Collecte des données
- Stockage des données
  - ▶ Système de fichier (distribué)
  - ▶ Base de données
- Traitement des données
  - ▶ Stream processing
  - ▶ Batch processing
- Visualisation des données

## Ce que vous devez maîtriser

# Ce que vous devez maîtriser

## La vision Big Data



# Ce que vous devez maîtriser (suite)

## La vision Cloud



# Le projet SDTD

## Travaux à réaliser

- Étude d'une pile logicielle (*stack*) représentative et largement répandue
- Déploiement sur un service de Cloud public (AWS)
- Mise en œuvre d'un cas d'usage

## Mise en place

- Des groupes de 5 étudiants
- Chaque groupe choisit son cas d'usage
- Chaque groupe fait ses choix techniques
- Séance de restitution collective
  - ▶ Partage de l'expérience acquise

## La pile logicielle

Votre pile logicielle doit comprendre au moins:

- Un agent de messages (message broker)
- Une base de données distribuée
- Un framework de traitement de données

Le déploiement de ces briques logicielles pourra être géré par [un orchestrateur d'applications](#).

## La pile logicielle de base: SMACK

- Apache Spark (traitement de données)
- Apache Mesos (orchestration)
- (Akka) (Reactive applications)
- Apache Cassandra (Base de données distribuée NoSQL)
- Apache Kafka (agent de messages)

# Piles logicielles alternatives

Vous pouvez modifier la pile de base en y intégrant un projet concurrent.

**L'intégration d'une de ces alternatives sera prise en compte dans la notation**

- Orchestration
  - ▶ [Kubernetes](#)
  - ▶ Docker Swarm
- Agent de messages
  - ▶ RabbitMQ
- Base de données distribuée
  - ▶ MongoDB
- Traitement de données
  - ▶ [Apache Storm](#)
  - ▶ [Apache Flink](#)

# Piles logicielles alternatives

## Commentaires:

- Les solutions marquées en bleu sont à privilégier
- Toutes les combinaisons ne sont pas nécessairement bien supportées
- Vous pouvez faire d'autres choix mais vous devez les faire valider par l'enseignant

## Les cas d'usage

**Objectif:** Illustrer le fonctionnement de la pile logicielle

- Chaque groupe définit son cas d'usage
  - ▶ Identification des données à analyser
  - ▶ Définition du traitement à appliquer

# Les cas d'usage: données à traiter

## Utilisation de données *réelles*

- Certaines API permettent de récupérer gratuitement des données en temps réel
  - ▶ Ex: API Twitter, pioupiou<sup>1</sup>, etc.
- Recherche de jeux de données:
  - ▶ <https://toolbox.google.com/datasetsearch>
  - ▶ Possibilité de rejouer des données
- Vous pouvez en dernier recours créer votre propre jeu de données

---

<sup>1</sup><http://developers.pioupiou.fr/>

# Le Cloud public

## Amazon Web Services

- Utilisation de machines virtuelles nues
  - ▶ Service EC2
  - ▶ Utilisation de VMs Linux
  - ▶ **Nous n'utiliserons pas les services avancés disponibles sur AWS**
- Utilisation de vos crédits
  - ▶ AWSome Day Ensimag 2018 (27/09)
  - ▶ Obtention des crédits: <https://aws.amazon.com/fr/education/awseducate/apply/>
    - **Ne prenez pas l'option Starter account**

# Travail demandé

## 6 points principaux

- Étude des logiciels à utiliser
- Construction automatique d'un environnement d'exécution distribué virtuel
- Déploiement de la pile logicielle
- Construction de l'application de démonstration
- Analyse des capacités du système
- Présentation du projet et partage d'expérience avec l'ensemble des groupes

# Automatisation

## Construction de l'infrastructure distribuée dans le Cloud Réservation et configuration des machines virtuelles automatisée

- Scripts (shell, python, etc.)
- Utilisation de bibliothèques appelant l'API AWS (boto3)
- Utilisation d'outils de plus haut niveau pour créer des recettes (Ansible, Puppet, Chef, SaltStack, etc.)

## Déploiement des briques logicielles

Installation automatique des différents logiciels

- Scripts
- Configurations de conteneurs
- Construction de recettes pour configurer des VMs (Vagrant)

# Gestion avancée du déploiement

## Orchestration de conteneurs

Outils automatisant le déploiement de briques logicielles conteneurisées

- Mesos
- Kubernetes
- Docker Swarm

**L'utilisation de tels outils n'est pas obligatoire mais recommandée**

- Utilisation de scripts

# Évaluation des capacités du système

Propriétés non fonctionnelles

## Réponse aux questions suivantes

- Mon application fonctionne-t-elle toujours en cas de panne matérielle ?
  - ▶ Haute disponibilité
  - ▶ Considérez et testez différents scénarios
    - Simulez des défaillances
- Combien de données par secondes mon application est-elle capable de traiter ?
  - ▶ Débit
  - ▶ Possibilité de considérer différentes configurations
- Quel est le temps nécessaire pour traiter une nouvelle donnée arrivant dans le système?
  - ▶ Temps de réponse
  - ▶ Certaines applications peuvent avoir des contraintes *temps réel* faibles

## Fonctionnalités avancées (optionnel)

### Scalabilité horizontale

- Mesure du taux d'utilisation des ressources et/ou des variations dans les temps de réponse
- Ajout/retrait à *chaud* de ressources pour s'adapter à la charge

### Amélioration de la disponibilité du système

- Identification des *single point of failure* de votre infrastructure
- Mise en place de stratégie pour palier le problème

# Application de démonstration

## Objectif

- Démontrer l'utilisation des composants logiciels
  - ▶ Simple
  - ▶ Réaliste
- Ne pas passer trop de temps sur l'interface graphique

## Types de calcul

- Stream processing
- Batch processing
- Les deux

## Jeu de données

- Ne pas hésiter à rejouer un jeu de données réel

## Concise mais précise !!

- Description des composants logiciels utilisés
- Architecture logicielle globale
- Application de démonstration
- Gestion du cycle de vie et des ressources
- Documentation des principaux problèmes techniques rencontrés et des solutions employées pour les résoudre.
- Évaluation des propriétés non fonctionnelles

# Calendrier

- 24 septembre: présentation des projets
  - ▶ Constitution des groupes
- 12 octobre: Séance individuelle (30 minutes par groupe)
  - ▶ Présentation des composants du système
  - ▶ Présentation de l'application de démonstration envisagée
  - ▶ Documentation (1ère partie)
    - Présentation des composants du système

## Calendrier

- 26 octobre: Séance collective
  - ▶ Suivi (infrastructure)
- 30 novembre: Séance individuelle (30 minutes par groupe)
  - ▶ Déploiement automatisé (Démo)
  - ▶ Gestion du cycle de vie (Démo)
- 14 décembre: Séance collective
  - ▶ Suivi (application – propriétés non fonctionnelles)
- 11 janvier: Séance individuelle (30 minutes par groupe)
  - ▶ Application de démonstration (Démo – version finale)
  - ▶ Haute disponibilité (Démo)
  - ▶ Autres propriétés non fonctionnelles (optionnel)

## Calendrier

- 18 janvier: Rendu TEIDE
  - ▶ Documentation, Code
- 1 jour avant la séance collective: Rendu TEIDE
  - ▶ Slides
- 25 janvier(?): Séance collective (30 minutes par groupe)
  - ▶ Présentation du projet
  - ▶ Partage de l'expérience entre les groupes
  - ▶ Présence d'un jury

# Notation

Chaque étape est notée

- 12/10: 10%
- 30/11: 20%
- 11/01: 30%
- 25/01: 40%

Sont pris en compte:

- Réalisations (haute disponibilité, benchmarking, etc)
- Qualité de l'application de démonstration
- Qualité de la documentation
- Qualité des présentations
- Qualité de l'infrastructure logicielle (Automatisation, gestion des ressources, etc)

# Infos pratiques

## A propos d'AWS

- Utilisation de machines virtuelles EC2 (Linux)
- Authentification par clés ssh permet à tous les membres du groupe de se connecter aux machines si besoin
- Commencer avec des machines t2.micro
  - ▶ Augmenter la taille si trop limité
- Penser à bien arrêter vos machines virtuelles avant de terminer votre session
  - ▶ Utiliser une CB virtuelle pour vous enregistrer
  - ▶ Bien documenter la configuration et automatiser le déploiement sur AWS pour simplifier le passage d'un compte AWS à un autre
- Pour l'utilisation de tout autre service AWS, demander préalablement à l'enseignant.

# Annexes

## Références : sources de données

- <https://www.programmableweb.com/news/62-real-time-apis-twitter-thruthu-and-pusher/2012/04/17>
- <https://freestartupkits.com/articles/technology/tech-tips-and-tricks/best-free-open-data-sources-2018/>