

Systèmes Distribués pour le Traitement de Données

Thomas Ropars

thomas.ropars@univ-grenoble-alpes.fr

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

2017

Contexte

Data analytics - Big Data

- 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) des cours de bourses
- 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

Composants classiques des infrastructures Big Data

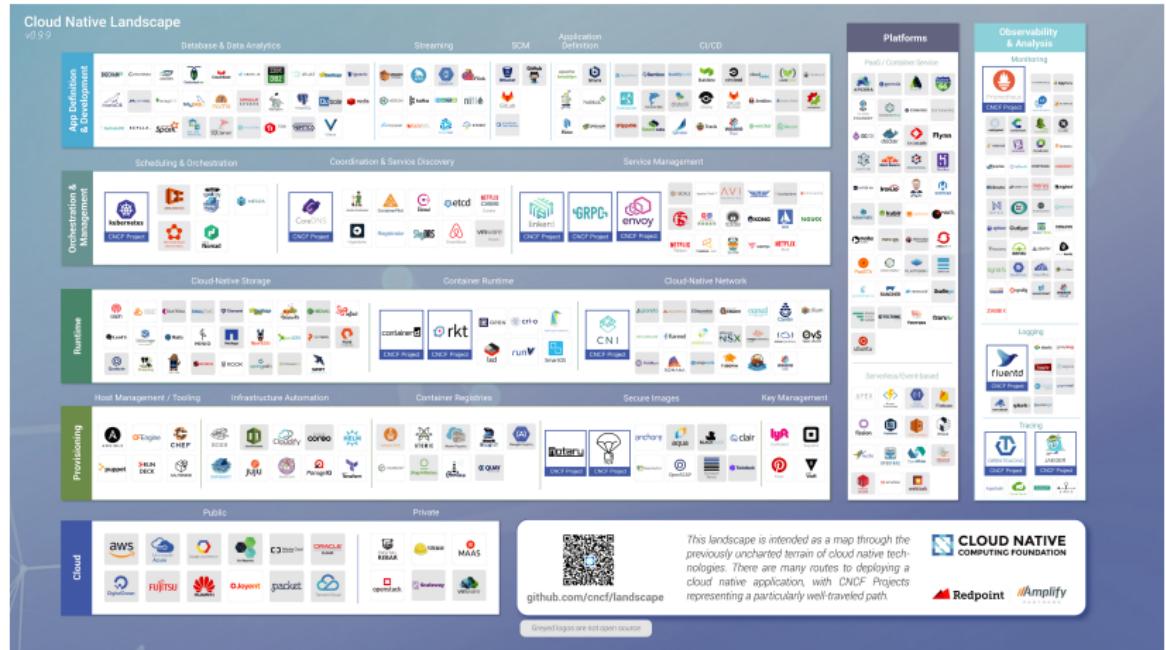
- Collecte des données
- Stockage des données
 - ▶ Système de fichier (distribué)
 - ▶ Base de données
- Traitement des données
 - ▶ Stream processing
 - ▶ Batch processing
- Visualisation des données

Objectifs du module SDTD

Ce que vous devez maîtriser

Objectifs du module SDTD

Et aussi (la vision Cloud)



Projet SDTD

Le projet

- 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

- 5 groupes de 8 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: SMACK

Les composants

- Spark (data processing)
- Mesos (Resource management)
- Akka (Reactive application)
- Cassandra (Distributed database)
- Kafka (Data feeds)

Les cas d'usage

Instructions

- Chaque groupe définit son cas d'usage
- Objectif: Illustrer l'utilisation de la pile logicielle
- Les données:
 - ▶ Utilisation d'un jeu de données *réel*
 - De nombreuses sources disponibles gratuitement
 - ▶ Génération de vos propres données

Exemples de domaines d'applications

- Logs systèmes
- Données médicales
- Réseaux sociaux
- etc.

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
 - ▶ Journée AWSome Day (21/09)

Travail demandé

Plusieurs axes

- Gestion automatisée du cycle de vie et des ressources
- Haute disponibilité
- Analyse de performances
- Scalabilité horizontale (Bonus)
- Extension de la pile logicielle (Bonus)
- Réalisation d'une application de démonstration
- Documentation

Gestion automatisée du cycle de vie et des ressources

Cycle de vie

- Automatisation au travers de scripts
 - ▶ Déploiement automatisé des composants de la pile logicielle
 - Installation des logiciels
 - Configuration
 - ▶ Gestion du cycle de vie
 - Démarrage
 - Arrêt
- Une machine d'administration (Manager)
 - ▶ Exécute les scripts
 - ▶ Toutes les commandes sont à réaliser via cette machine
 - ▶ Commande le déploiement sur tous les nœuds d'exécution

Gestion automatisée du cycle de vie et des ressources

Utilisation d'outils avancés

- Utilisation possible de technologies de conteneurs pour simplifier l'installation, la configuration et le déploiement des logiciels.
 - ▶ Est ce que cela a du sens?
 - ▶ Quels sont les avantages?
- Utilisation d'outils d'orchestration et de déploiement (Kubernetes, Docker Swarm, etc)
 - ▶ Est ce que cela a du sens?
 - ▶ Quels sont les avantages?

Propriétés non fonctionnelles

Haute disponibilité

- Utilisation d'un script pour simuler la perte d'un composant logiciel
 - ▶ Votre plateforme doit continuer de fournir son service
- Redémarrage automatisé des composants si nécessaire
 - ▶ Certaines briques sont nativement hautement disponibles
 - ▶ Redémarrage local

Mesure et analyse de performance (Benchmarking)

- Réponse à des questions du type: "Combien de traitements par secondes le système est-il capable de réaliser?"
 - ▶ Génération de workloads synthétiques pour stresser le système
- Automatisation au travers de scripts
 - ▶ Lancement des tests
 - ▶ Collecte des données
 - ▶ Traitement et mise en forme des résultats

Propriétés non fonctionnelles (Bonus)

Scalabilité horizontale

- Mesure du taux d'utilisation des ressources
- Ajout/retrait *à chaud* de ressources pour s'adapter à la charge

Extension de la pile logicielle (Bonus)

D'autres piles logicielles majeures existent dans le paysage du Big Data:

- Elastic (Elasticsearch, Kibana, Logstash, Beats)
 - ▶ Stockage de documents textuels (logs systèmes?)
 - ▶ Recherche avancée, visualisation, etc.
- TICK (Telegraf, InfluxDB, Chronograf, Kapacitor)
 - ▶ Séries temporelles (métriques de monitoring)
 - ▶ Traitement, visualisation, etc.
- Autres?
- Intégration de certains éléments de ces piles logicielles en plus de SMACK:
 - ▶ Ajout de nouvelles fonctionnalités pour les utilisateurs?
 - ▶ Amélioration de la gestion de votre plateforme

Travail demandé (Fin)

Application de démonstration (cas d'usage)

- Simple
- Illustrative
- Réaliste (Utilisation de données réelles ou générées)

Documentation (concise mais précise)

- Description des différents 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.

Calendrier

- 27 octobre: présentation des projets
 - ▶ Constitution des groupes
- 24 novembre: séance individuelle (30 minutes par groupe)
 - ▶ Présentation des composants du système (gestion des données et des ressources)
 - ▶ Présentation de l'application de démonstration envisagée
 - ▶ Documentation (version préliminaire)
 - Présentation des composants du système
- 8 décembre: séance individuelle (30 minutes par groupe)
 - ▶ Déploiement automatisé (Démo)
 - ▶ Gestion du cycle de vie (Démo)
 - ▶ Application de démonstration (Démo – version préliminaire)

Calendrier

- 12 janvier: séance individuelle (30 minutes par groupe)
 - ▶ Application de démonstration (Démo – version finale)
 - ▶ Haute disponibilité (Démo)
 - ▶ Réalisations bonus (Démo)
 - ▶ Reste à faire: Benchmarking
- 19 janvier: Rendu TEIDE
 - ▶ Documentation, Code, Retours sur le projet
- 1 jour avant la séance collective: Rendu TEIDE
 - ▶ Slides
- 26 janvier(?): Séance collective (40 minutes par groupe)
 - ▶ Présentation du projet
 - ▶ Partage de l'expérience entre les groupes
 - ▶ Présence d'un jury

Notation

Chaque étape est notée

- 24/11: 20%
- 08/12: 20%
- 12/01: 20%
- 26/01: 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 et conseils

Les groupes

- 5 groupes de 8 étudiants
- Un chef désigné par groupe
- Définir les rôles au sein du groupe

Infos pratiques et conseils

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
- Utilisation possible de Amazon Virtual Private Cloud (VPC) pour simplifier la configuration de votre système
 - ▶ Assigner des adresses privées statiques à vos VMs
- Pour l'utilisation de tout autre service AWS, demander préalablement à l'enseignant.

Infos pratiques et conseils

Bibliothèques/langages à utiliser

- Scripts: Shell ou Python
- Possibilité d'utiliser Monit (<https://mmonit.com/monit/>) pour la détection de défaillances
- Pour tout le reste, demander préalablement à l'enseignant

Remarques complémentaires

- Mettre à jour le /etc/hosts de chaque machine (scripter) afin d'utiliser des noms symboliques pour les machines (dans les fichiers de configuration et les scripts).
- Ne jamais utiliser d'adresses IP dans les scripts
- Utilisation de ssh pour les connexions entre les différentes machines du système
- Le déploiement et la configurations des bibliothèques doit être prévu dans les scripts