

Systèmes Distribués pour le Traitement de Données

Thomas Ropars

thomas.ropars@univ-grenoble-alpes.fr

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

2020

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 traiter de grandes quantités de données.
(et éventuellement 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, coût, etc.)

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

BIG DATA & AI LANDSCAPE 2018



VI - Last updated 6/19/2018

© Matt Turck (@mattturck), Demilade Obayomi (@demi_ obayomi), & FirstMark (@firstmarkcap) mattturck.com/bigdata2018

FIRSTMARK
EARLY STAGE VENTURE CAPITAL

Infrastructures distribuées et Cloud computing

Les infrastructures de type Cloud fournissent les ressources matérielles nécessaires à l'exécution d'applications distribuées.

Différent modèles d'accès aux ressources

- Infrastructure-as-a-Service (IaaS)
- Platform-as-a-Service (PaaS)
- Software-as-a-Service (SaaS)
- Container-as-a-Service (CaaS)
- Function-as-a-Service (FaaS)

Infrastructures distribuées et Cloud computing

Plusieurs technologies permettent de simplifier le déploiement et la gestion d'application distribuées (dans le Cloud).

Les conteneurs

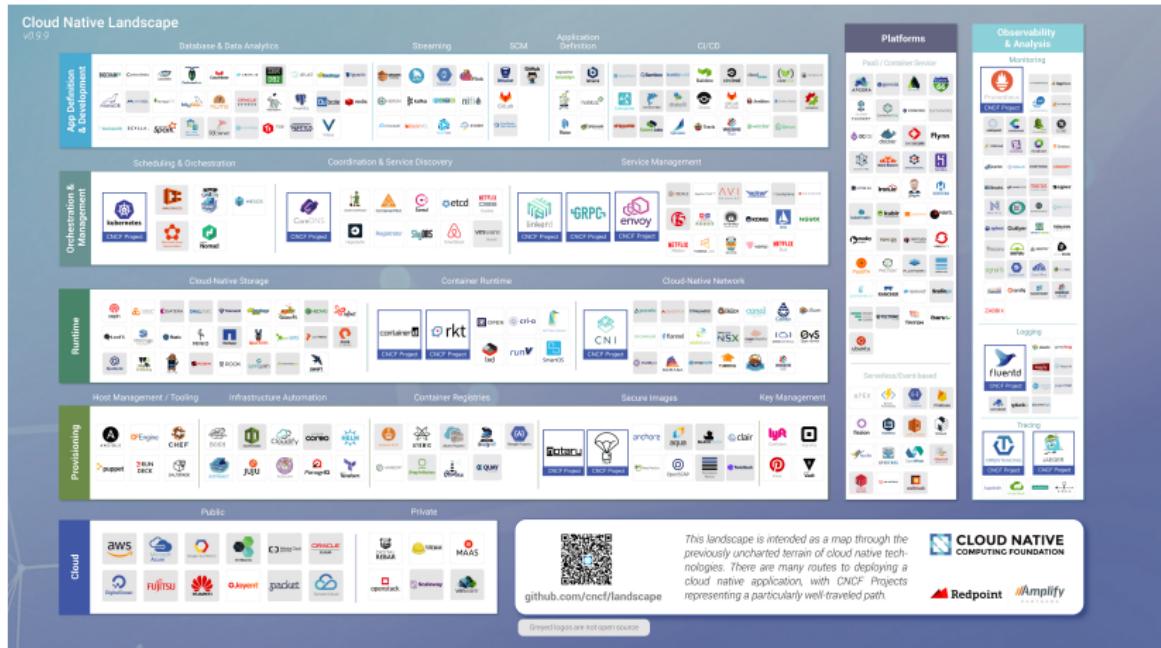
- Outil permettant de packager un logiciel, ses dépendances et sa configuration dans une image.
 - ▶ Exécution simplifiée et reproductible (Infrastructure-as-code)
 - ▶ Exécution isolée
- Technologie principale: Docker

Les orchestrateurs

- Déploiement et gestion automatique des ressources pour des applications conteneurisées
 - ▶ Disponibilité
 - ▶ Élasticité
 - ▶ Équilibrage de charge
- Technologie principale: Kubernetes

Ce que vous devez maîtriser (suite)

La vision Cloud



Le projet SDTD

Mise en place

- Des groupes de 4 étudiants
 - ▶ **Au moins un étudiant suivant le cours Cloud Avancé**
- Chaque groupe choisit son cas d'usage
- Chaque groupe fait ses choix techniques
- Séance de restitution collective
 - ▶ Partage de l'expérience acquise

Travail à réaliser

Objectifs généraux

- Mise en place d'un pile logicielle de traitement de données à large échelle
- Approvisionnement de ressources et déploiement automatique de la pile dans un Cloud public. (AWS)
- Mise en œuvre d'un cas d'usage

Deux approches possibles:

- Projet orienté *infrastructure*
- Projet orienté *traitement de données*

Projet infrastructure

Objectifs du projet

- Comprendre le fonctionnement de l'orchestrateur Kubernetes
- Être capable de configurer et déployer une application distribuée dans le Cloud en utilisant Kubernetes
- Construire une application tirant partie des fonctionnalités de Kubernetes (élasticité, équilibrage de charge, disponibilité, etc.)

Les consignes principales

- Déploiement d'un cluster Kubernetes sur AWS
 - ▶ Utilisation de **Kubeadm** pour instancier un cluster au dessus d'un ensemble de VMs préalablement provisionnées
- Application de traitement de donnée conçue pour mettre en évidence les fonctionnalités de Kubernetes
 - ▶ Pile logicielle pouvant être simple (2 composants)
 - ▶ Possibilité d'ajouter des composants simple de transformation des données
 - Extraction et mise en forme de données

Projet traitement de données

Objectifs du projet

- Construire une pile logicielle complète de traitement de données
- Déployer automatiquement son application dans le Cloud
- Évaluer les capacités de la pile logicielle (Performance, disponibilité)

Les consignes principales

- Construire une pile logicielle comprenant au moins 3 composants
 - ▶ Un agent de messages (message broker)
 - ▶ Une base de données distribuée
 - ▶ Un framework de traitement de données
- Pile déployable automatiquement dans le Cloud
 - ▶ Approvisionnement de ressources et configuration automatique
- Possibilité de gérer au moins une partie de l'infrastructure en utilisant Kubernetes
 - ▶ Utilisation de [Kops](#) pour le déploiement automatique d'un cluster Kubernetes

Pile logicielle à mettre en oeuvre

Chaque groupe fait ses propres choix et est capable de les justifier

- Orchestration
 - ▶ [Kubernetes](#)
- Agent de messages
 - ▶ [Kafka](#)
 - ▶ [Redis](#)
 - ▶ [\(RabbitMQ\)](#)
- Base de données distribuée
 - ▶ [Cassandra](#)
 - ▶ [MongoDB](#)
 - ▶ [Redis](#)
- Traitement de données
 - ▶ [Spark](#)
 - ▶ [Flink](#)
 - ▶ [\(Storm\)](#)
 - ▶ [\(Samza\)](#)

La pile logicielle Kafka-Spark-Cassandra est populaire.

Pile logicielle à mettre en oeuvre

Commentaires:

- Les solutions marquées en bleu sont des solutions très largement utilisées.
 - ▶ Toutes les combinaisons ne sont pas nécessairement bien supportées
 - ▶ Pour Spark, beaucoup de connecteurs ne sont maintenus à jour que pour le langage Scala.
 - ▶ Vous pouvez faire d'autres choix mais vous devez les faire valider par l'enseignant
- Vous pouvez ajouter des composants en plus à votre pile
 - ▶ Visualisation (Graphana, Kibana, etc.)
 - ▶ Cache distribué (Redis, Memcached, etc.)
 - ▶ Supervision de l'infrastructure (Telegraph-InfluxDB-Graphana, Zabbix, etc.)
 - ▶ Indexation de données (ElasticSearch, etc.)

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 des traitements à appliquer
 - ▶ Définition du pipeline de traitement
 - Architecture de votre application

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, etc.
- Recherche de jeux de données:
 - ▶ Voir liste de références en annexe
 - ▶ **Possibilité de rejouer des données**
- Vous pouvez en dernier recours créer votre propre jeu de données

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 2020 (24/09)
 - ▶ 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 l'infrastructure distribuée
- 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.)
- Approvisionnement de ressources:
 - ▶ Utilisation d'outils de plus haut niveau pour créer des recettes
 - Terraform
 - Ansible
 - Puppet, Chef, SaltStack, etc.
 - ▶ Utilisation d'une bibliothèque appelant l'API AWS (boto3)

Déploiement des briques logicielles

Installation automatique des différents logiciels

- Scripts
- Configurations de conteneurs (Docker)

A propos de l'automatisation

Pour tous les projets:

- Utilisation de **Terraform** et **Ansible** fortement recommandée pour l'approvisionnement et la configuration de VMs.
- Exemples disponibles ici (pour le cas de GCP):
<https://roparst.gricad-pages.univ-grenoble-alpes.fr/cloud-tutorials/>
 - ▶ Création automatique d'un ensemble de VMs avec Terraform:
<https://roparst.gricad-pages.univ-grenoble-alpes.fr/cloud-tutorials/terraform/>
 - ▶ Création et configuration de VMs avec Terraform et Ansible (installation de MPI): <https://roparst.gricad-pages.univ-grenoble-alpes.fr/cloud-tutorials/mpi/>

Pour les projets traitement de données:

L'utilisation de **Kops** ne nécessite pas de provisionner l'infrastructure préalablement.

Évaluation des capacités du système

Propriétés non fonctionnelles

Réponse à certaines des questions suivantes

- Mon application fonctionne-t-elle toujours en cas de panne matérielle ?
 - ▶ Haute disponibilité
 - ▶ Considérez et testez différents scénarios
 - Simulez des défaillances
 - ▶ Identification des *single point of failure* de votre infrastructure
 - ▶ Mise en place de stratégie pour palier le problème
- Combien de données par secondes mon application est-elle capable de traiter ?
 - ▶ Débit
 - ▶ Possibilité de considérer différentes configurations

Évaluation des capacités du système

Propriétés non fonctionnelles

- La charge est-elle équitablement répartie entre les noeuds de mon infrastructure?
 - ▶ Équilibrage de charge
 - ▶ Mesure du taux d'utilisation des ressources
- Mon infrastructure s'adapte-t-elle aux variations de charge?
 - ▶ Élasticité / Scalabilité horizontale
 - ▶ Ajout/retrait à *chaud* de ressources pour s'adapter à la charge
- Quel est le coût de mon infrastructure?
 - ▶ Combien a été dépensé pour mettre en place l'infrastructure?
 - ▶ Combien coûte mon infrastructure en fonctionnement normal?
 - Quels sont mes gains liés aux mécanismes d'élasticité?

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

Documentation

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

- 21 septembre: présentation des projets
 - ▶ Constitution des groupes
- 9 octobre: Séance individuelle (12 minutes par groupe)
 - ▶ Présentation des composants du système
 - Projet infra: Composant Kubernetes
 - Projet données: Pile logicielle
 - ▶ Présentation de l'application de démonstration envisagée
 - ▶ Documentation (1ère partie)
 - Présentation des composants du système

Calendrier

- 13 novembre: Séance collective
 - ▶ Suivi (infrastructure)
- 30 novembre: Séance individuelle (12 minutes par groupe)
 - ▶ Approvisionnement de ressources (Démo)
 - ▶ Déploiement automatique (Démo)
 - ▶ Application de démonstration (Présentation avancée et démo préliminaire)
- 11 décembre: Séance collective
 - ▶ Suivi (application – propriétés non fonctionnelles)
- 15 janvier: Séance individuelle (12 minutes par groupe)
 - ▶ Application de démonstration (Démo – version finale)
 - ▶ Propriétés non fonctionnelles (optionnel)

Calendrier

- 22 janvier: Rendu TEIDE
 - ▶ Documentation, Code
- 1 jour avant la séance collective: Rendu TEIDE
 - ▶ Slides
- 29 janvier: Séance collective (15 minutes par groupe)
 - ▶ Présentation du projet
 - ▶ Partage de l'expérience entre les groupes
 - ▶ Présence d'un jury

Notation

Chaque étape est notée

- 09/10: 10%
- 30/11: 10%
- 15/01: 30%
- 29/01: 50%

Sont pris en compte:

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

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
 - ▶ 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.

Références

- *Designing Distributed Systems*, by B. Burns, 2018.
- *Designing Data-Intensive Applications*, by M. Kleppmann, 2017.
- *Site Reliability Engineering*, by B. Beyer et al, 2016.

Annexes

Références : sources de données

- <https://www.dataquest.io/blog/free-datasets-for-projects/>
- <https://github.com/awesomedata/awesome-public-datasets>
- <https://toolbox.google.com/datasetsearch>
- <https://crawdad.org/>