

Systèmes Distribués pour le Traitement de Données

Thomas Ropars

thomas.ropars@univ-grenoble-alpes.fr

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

2021

Agenda

Introduction

Consignes générales

Consignes techniques

Déroulement du projet

Ressources supplémentaires

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

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.

Modèle utilisé dans ce projet

- Infrastructure-as-a-Service (IaaS)

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

Agenda

Introduction

Consignes générales

Traitements de données

Intégration de l'orchestrateur Kubernetes

Les cas d'usage

Propriétés non fonctionnelles

Consignes techniques

Déroulement du projet

Ressources supplémentaires

Le projet SDTD

Mise en place

- Des groupes de 4 ou 5 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, GCP)
- Mise en œuvre d'un cas d'usage
- Évaluation des caractéristiques non fonctionnelles de la pile mise en place.

2 lignes directrices possibles

- Se concentrer sur la pile de traitement de données
- Pile de traitement de données + Intégration de l'orchestrateur Kubernetes

Consignes générales

Traitement de données

Intégration de l'orchestrateur Kubernetes

Les cas d'usage

Propriétés non fonctionnelles

Objectifs

- 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

Consignes générales

Traitement de données

Intégration de l'orchestrateur Kubernetes

Les cas d'usage

Propriétés non fonctionnelles

Objectifs

- 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

- Certains ou tous les composants de la pile de traitement de données peuvent être déployées au sein du cluster Kubernetes
 - ▶ Pas besoin d'approvisionnement de ressources pour les composants déployés au sein de Kubernetes
- Vous êtes autorisés à avoir une pile plus simple si vous faites l'effort d'intégrer Kubernetes.
 - ▶ 2 composants au lieu de 3
 - ▶ Brique de traitement de données construite *à la main*
 - Tirer partie de l'élasticité

Consignes générales

Traitement de données

Intégration de l'orchestrateur Kubernetes

Les cas d'usage

Propriétés non fonctionnelles

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

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

- Traitement de données en temps réel (Stream Processing)
- Possibilité d'ajouter des calculs de type Batch processing

Jeu de données

- Ne pas hésiter à rejouer un jeu de données réel

Consignes générales

Traitement de données

Intégration de l'orchestrateur Kubernetes

Les cas d'usage

Propriétés non fonctionnelles

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 ?
 - ▶ 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?

Agenda

Introduction

Consignes générales

Consignes techniques

Déroulement du projet

Ressources supplémentaires

Pile de traitement de données à mettre en oeuvre

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

Pile de traitement de données à mettre en oeuvre

Recommandations

- La pile logicielle Kafka-Spark-Cassandra est populaire et plutôt bien documentée
- Vous pouvez faire d'autres choix mais vous devez les faire valider par l'enseignant
 - ▶ Toutes les combinaisons ne sont pas bien supportées
- Pour Spark, beaucoup de connecteurs ne sont maintenus à jour que pour le langage Scala.

Aller plus loin:

- 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.)

A propos de Kubernetes

Déploiement de Kubernetes

- Utilisation de [Kops](#) pour instancier un cluster:
<https://kops.sigs.k8s.io/>
- Alternative possible: Configurer vous même votre cluster en utilisant [Kubeadm](#) (<https://kubernetes.io/docs/reference/setup-tools/kubeadm/>)

Vous n'avez pas le droit d'utiliser les clusters Kubernetes gérés pour les fournisseurs de Cloud.

Le Cloud public

Les fournisseurs considérés

- Amazon Web Services
- Google Cloud Platform

Consignes principales

- Utilisation de machines virtuelles nues
 - ▶ Service EC2 (AWS) ou GCE (GCP)
 - ▶ Utilisation de VMs Linux
 - ▶ **Nous n'utiliserons pas les services avancés fournis pour ces systèmes**

Obtention de crédits

- Infos à venir très vite

Automatisation

Construction de l'infrastructure distribuée dans le Cloud

Approvisionnement de ressources

Réservation et configuration automatisées des machines virtuelles

- Scripts (shell, python, etc.)
- Utilisation d'outils de plus haut niveau pour créer des recettes
 - ▶ Terraform
 - ▶ Ansible
 - ▶ Puppet, Chef, SaltStack, etc.

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 Kubernetes:

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

Agenda

Introduction

Consignes générales

Consignes techniques

Déroulement du projet

Ressources supplémentaires

Travail demandé

5 points principaux

- Construction automatique d'un environnement d'exécution distribué virtuel
- Mise en place et déploiement de l'infrastructure distribuée de traitement de données
- Construction de l'application de démonstration
- Analyse des capacités du système
- Présentation du projet et partage d'expérience avec l'ensemble des groupes

Rapport

Concis mais précis !!

- 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 (Teide)
- 8 octobre: Séance collective
 - ▶ Cours: Présentation des outils pour le traitement de données
 - ▶ Réponse aux questions
- 15 octobre: Rendu TEIDE – Court rapport
 - ▶ Présentation des objectifs du groupe et de la pile logicielle adoptée
 - ▶ Présentation de l'application de démonstration envisagée

Calendrier

- 22 octobre: Séance collective
 - ▶ Suivi (infrastructure)
- 26 Novembre: Séance individuelle (12 minutes par groupe)
 - ▶ Approvisionnement de ressources (Démo)
 - ▶ Déploiement automatique (Démo)
 - ▶ Pile de traitement de données et application de démonstration (Présentation et démo préliminaire)
- 17 décembre: Séance collective
 - ▶ Suivi (application – propriétés non fonctionnelles)
- 14 janvier: Séance individuelle (12 minutes par groupe)
 - ▶ Application de démonstration (Démo – version finale)
 - ▶ Propriétés non fonctionnelles (optionnel)

Calendrier

- 21 janvier: Rendu TEIDE
 - ▶ Rapport, Code
- 1 jour avant la séance collective: Rendu TEIDE
 - ▶ Slides
- 28 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

- 15/10: 10%
- 26/11: 10%
- 14/01: 30%
- 28/01: 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

Poser des questions en dehors des séances

La manière privilégiée pour me contacter:

- Le forum du cours sur Chamilo
- [https://chamilo.grenoble-inp.fr/courses/
ENSIMAG5MMSDTD7/index.php?id_session=0](https://chamilo.grenoble-inp.fr/courses/ENSIMAG5MMSDTD7/index.php?id_session=0)
- Les autres groupes auront peut être la réponse à votre question !!

Me contacter directement par email si la question révèle trop d'informations sur votre solution.

Agenda

Introduction

Consignes générales

Consignes techniques

Déroulement du projet

Ressources supplémentaires

A propos de l'utilisation du Cloud

- Utilisation de machines virtuelles (Linux)
- Authentification par clés ssh permet à tous les membres du groupe de se connecter aux machines si besoin
- Commencer avec des machines petites (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 le cloud pour simplifier le passage d'un compte à un autre
- Pour l'utilisation de tout autre service que les VMs, demander préalablement à l'enseignant.

Quelques 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.

Pour trouver des 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/>