Systèmes Distribués pour le Traitement de Données

Thomas Ropars

thomas.ropars@univ-grenoble-alpes.fr

http://tropars.github.io/

2023

Agenda

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) des données de capteurs
- Analyse (temps réel) de données de monitoring
- Entraînement de modèles d'apprentissage
- Mise en forme de données pour une utilisation future

Mise en place de traitement de données (à large échelle)

3 briques principales

- Les algorithmes de traitement de données
- 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.)

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)
- Platform-as-a-Service (PaaS)

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 d'applications conteneurisées et gestion automatique des ressources
  - Disponibilité
  - Élasticité
  - Équilibrage de charge
- Technologie principale: Kubernetes

Agenda

Le projet SDTD

Mise en place

- Des groupes de 6 étudiants
- 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

Une application de traitement de données

- Mise en place d'un pile logicielle de traitement de données à large échelle
- Mise en œuvre d'un cas d'usage

Déploiement automatique dans un Cloud public

- Approvisionnement de ressources et déploiement automatique
- Google Cloud Platform (GCP)
- Kubernetes?

Fiabilité et performance de l'application

- Supervision de l'application
- Évaluation et amélioration des propriétés non-fonctionnelles

Répartition du travail

3 rôles principaux

- Ingénieur DevOps
  - Automatisation et simplification du déploiement
  - Conteneurisation, orchestration
- Développeur
  - Conception de la pile de traitement logiciel
  - Mise en oeuvre du cas d'usage
- Site Reliability Engineer (SRE)
  - Supervision de l'application en cours d'exécution
  - Évaluation et amélioration de la fiabilité et des performances

Répartition du travail

3 rôles principaux

- Ingénieur DevOps
  - Automatisation et simplification du déploiement
  - Conteneurisation, orchestration
- Développeur
  - Conception de la pile de traitement logiciel
  - Mise en oeuvre du cas d'usage
- Site Reliability Engineer (SRE)
  - Supervision de l'application en cours d'exécution
  - Évaluation et amélioration de la fiabilité et des performances

Au sein de chaque groupe, 2 étudiants assignés à chaque rôle

- Attention: S'adapter aux besoins du projet. A certains moment, il sera nécessaire d'aider les autres rôles

Travail en commun

Des dépendances entre les rôles

- **Important: Avancer quand même en parallèle**
- L'application peut commencer à être développée sur une seule machine
  - Pas nécessaire d'attendre l'approvisionnement automatique
- Certains tests peuvent être fait avec un cluster Kubernetes fourni par GCP
  - Pas nécessaire d'attendre le cluster déployé par l'équipe

Des synergies entre les rôles

- Conteneurisation de l'application (Dev + DevOps)
- Mise en place de stratégies de tolérance aux fautes (SRE + Dev et DevOps)

Agenda

La pile de traitement de données

- Traitement *temps réel* des données (Streaming)
- Construire une pile logicielle comprenant les composants suivants
  - Un agent de messages (message broker)
  - Une brique de traitement de données
  - (Optionnel) Une base de données distribuée

Le choix des technologies

- Agent de messages
  - Kafka
  - (RabbitMQ)
- Base de données distribuée
  - Cassandra
  - MongoDB
  - Redis
- Traitement de données
  - Développer votre propre application de traitement de données en Python
    - Certaines bibliothèques peuvent vous aider (ex: Dask)
  - (Spark)

Les choix recommandés en bleu

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

3 solutions possibles

Sans Kubernetes

- Approvisionnement de VMs avec Terraform
- Configuration des VMs avec Ansible
- Déploiement de l'application directement dans les VMs

En déployant Kubernetes

- Utilisation d'outils pour déployer son propre cluster Kubernetes (Voir annexes)
- Déploiement de l'application avec Kubernetes

En utilisant un cluster Kubernetes fourni par le Cloud provider

- Déploiement de l'application avec Kubernetes

Si vous choisissez la dernière option, les attendus seront plus grands sur les autres parties du projet

Le Cloud public

Le fournisseur considéré

- Google Cloud Platform

Consignes principales

- Utilisation de machines virtuelles nues
  - Service GCE
  - Utilisation de VMs Linux
  - **Nous n'utiliserons pas les services avancés fournis par GCP**
    - Exception pour Kubernetes si c'est votre choix

Obtention de crédits

- Infos à venir très vite

System Reliability Engineer

Les tâches

- Supervision
  - Collecte de métriques systèmes et applicatives
  - Mise en place de *Dashboards* pour la visualisation et d'alertes
- Évaluation de la fiabilité et des performances
  - Performance
  - Fiabilité
  - Coût
  - etc.
- Conception des stratégies de tolérance aux fautes

Supervision de l'infrastructure et de l'application

De nombreuses technos

- Prometheus
- Logstash - Kibana
- Graphana
- cAdvisor
- Etc.

Évaluation des capacités du système

Propriétés non fonctionnelles

Réponse a certaines des questions suivantes (liste non exhaustive)

- 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

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
- Supervision et analyse des capacités du système
- Présentation du projet et partage d'expérience avec l'ensemble des groupes

Rapport

Concis mais précis !!

- Architecture logicielle globale et description des composants logiciels utilisés
- Application de démonstration
- Présentation des mécanismes d'automatisation et d'orchestration
- Présentation des mécanismes de supervision et évaluation des propriétés non fonctionnelles
- Documentation des principaux problèmes techniques rencontrés et des solutions employées pour les résoudre.

Calendrier

- 18 septembre: présentation des projets
  - Constitution des groupes (Teide)
- 29 septembre: Séance individuelle
  - Présentation des objectifs du groupe et de la pile logicielle adoptée
  - Présentation de l'application de démonstration envisagée
  - Rapport + Slides (à déposer sur Teide)
    - Présenter les connaissances antérieures de chacun des membres du groupe
- 27 octobre: Séance collective
  - Suivi (focus infrastructure/devops)

Calendrier

- 24 novembre: Séance individuelle
  - Approvisionnement de ressources (Démo)
  - Déploiement automatique (Démo)
  - Pile de de traitement de données et application de démonstration (Présentation et démo préliminaire)
  - Supervision (Démo)
- 15 décembre: Séance collective
  - Suivi (application – propriétés non fonctionnelles)
- 12 janvier: Séance individuelle
  - Application de démonstration (Démo – version finale)
  - Propriétés non fonctionnelles

Calendrier

- 22 janvier: Rendu TEIDE
  - Rapport, Code
- 1 jour avant la séance collective: Rendu TEIDE
  - Slides
- 26 Janvier: Séance collective (20 minutes par groupe)
  - Présentation du projet
  - Partage de l'expérience entre les groupes
  - Présence d'un jury

Notation

Chaque étape est notée

- 29/09: 15%
- 24/11: 10%
- 12/01: 25%
- 26/01: 50%

Sont pris en compte:

- Qualité de l'infrastructure logicielle (Automatisation, gestion des ressources, etc)
- Réalisations (haute disponibilité, performances, etc)
- Qualité de l'application de démonstration
- Qualité des présentations
- Qualité des réponses aux questions

Agenda

Infos pratiques

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.

A propos de Kubernetes

Déploiement de Kubernetes

- Plusieurs solution pour automatiser le déploiement: `https://kubernetes.io/fr/docs/setup/custom-cloud/`
- Alternative possible: Configurer vous même votre cluster en utilisant Kubeadm (`https://kubernetes.io/docs/reference/setup-tools/kubeadm/`)

Quelle distribution Kubernetes?

- K8s: la version de Kubernetes par défaut
- K3s: Une alternative à considérer
  - Attention: Des avantages et des inconvénients

A propos de Kubernetes

Déploiement des briques au dessus de Kubernetes

- Des solutions existent pour simplifier le déploiement de certains composants (Message broker, Base de données, etc.)
  - Helm charts
  - Operators

A propos de l'automatisation

Pour des VMs:

- 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 d'outils de déploiement ne nécessite pas de provisionner l'infrastructure préalablement.

A propos de votre brique de traitement de données

Des questions à vous poser

- Ai-je besoin d'effectuer des calculs complexes qui agrègent un grand nombre de données?
  - Ou bien est-ce-que chaque donnée est traitée de manière indépendante?

- Est-ce-que je veux fournir des garanties spécifiques sur le traitement des données?
  - Au moins une fois
  - Au plus une fois

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/