Année 2024 - 2025

Big Data Frameworks

Analyse d’une trace de données de Google

ING3 HPDA

GOUTH Thomas
MILLOT Theo
RICARDE Lucas
SZWARCBART David

Matière encadrée par Juan Angel Lorenzo

Table des matières

1. Introduction

Google a mis à disposition de tous sur GitHub la trace d'utilisation d'un de ses clusters sur le mois de mai 2011. Cela représente plus de 40 Gbytes d'informations, cela a été fait dans le but de fournir des données pour faire progresser l'analyse de grand volumes de données. Les données contenues décrivent les machines, les jobs, les tâches réalisées durant la période. La période est divisée en 500 parties qui correspondent chacune à un fichier. Les données se répartissent en 6 catégories qui correspondent à un dossier de 500 fichiers chacun. On retrouve le descriptifs des machines ainsi que les événements enregistrés par les machines. On a aussi accès aux contraintes d'exécution des tâches. Un autre type d'informations qui est fourni est les jobs et les tâches proposés à l'exécution avec les informations relatives (ID, priorité, user…), dans notre analyse on utilisera la partie des tâches regroupée dans le dossier task_events. Enfin, on retrouve le dossier task_usage, notre analyse se concentrera autour des fichiers présents ici. En effet ce dossier rassemble les information d'exécution des tâches et jobs et notamment les performances utilisé tel que la mémoire ou l'utilisation du CPU.

Notre but va être d'analyser ces données et de sortir des insights intéressants de cette trace. Pour réaliser cela, nous allons dans un premier temps déterminer quels sont les jobs et les tâches dominants sur la trace, c'est-à-dire ceux ayant utilisé en moyenne et au maximum le plus de ressources de mémoire et de CPU. Ces tâches et jobs dominants identifiés nous allons dans un deuxième temps nous intéresser à leur ordre de priorité et regarder si l'on a une corrélation entre l'utilisation de la mémoire et du CPU.

2. Méthode utilisée

a. Choix du langage

Pour effectuer l'analyse de la trace nous avons décidé d'utiliser le langage python. Ici le défi majeur est de traiter un ensemble massif de données, c'est pourquoi nous avons décidé d'opter pour l'utilisation de l'API PySpark qui permet de gérer efficacement des grands volumes de données. En effet PySpark exploite la puissance de calcul distribuée afin de réaliser des tâches qui ne serait pas possible de faire sur des machines uniques.

Pour ce qui est de l'environnement de travail, nous avons dû choisir un environnement qui permettait la parallélisation de Spark et qui pouvait accueillir un grand volume de données. Nous avons en premier lieu opter pour Google Colab mais au fur et à mesure de notre avancement nous avions des problèmes de constances sur la rapidité d'exécution , une même exécution pouvant prendre un temps différent, ce qui rendait compliqué le choix de solutions techniques adaptées. Nous avons donc décidé de changer et de partir sur la solution développée par OpenAI : Deep Note. Dans les fichiers présents sur GitHub, nous avons déposé deux versions de notre notebook, une ayant les chemins adaptés à Google Collab et une ayant les chemins adaptés à Deep Note.

b. Choix des outils

Dans l'API PySpark les données peuvent être traitées de deux manières différentes, soit sous forme de dataframes, soit sous forme de RDD (resilients distributed dataset). Les RDD sont des structures de données immuable qui permettent la partitions et donc le traitement sur un cluster de machines afin d'augmenter la puissance de calcul. Après avoir testé les deux options nous sommes arrivés à la conclusion que les RDD Permettaient une exécution plus rapide, nous sommes donc partis sur cette option pour l'analyse des tâches et des jobs dominants.

Pour le reste de l'étude nous avons choisi de travailler avec seulement les jobs et tâches dominants nous sommes donc repasser avec des dataframes Pandas qui permettent un manipulation et des visualisations plus aisé. Pour les visualisations nous avons utlisés les librairies matplotlib et seaborn.

c. Méthodologie

Nous allons maintenant rentrer dans le descriptif de ce que nous avons fait. Tout d'abord pour l'importation des données nous avons décidé de les télécharger sur les serveurs drive temporaires qui sont mis à notre disposition par Google et OpenAI. Nous n'avons téléchargé que les données nécessaires à notre analyse, c'est-à-dire les dossiers task_usage et task_events. Ce choix nécessite un peu d'attente au lancement du notebook mais permet de lire plus rapidement les données par la suite.

Nous avons ensuite chargé les données du dossier task_usage sous forme de RDD comme discuté précédemment et nous avons divisé le RDD en deux parties : une pour identifier les jobs dominants en mémoire et en utilisation du CPU et l'autre pour identifier les tâches dominantes. Pour chacun des RDD on a rassemblé les données et calculé la moyenne d'utilisation du CPU et de la mémoire ainsi que le maximum d'utilisation du CPU et de la mémoire.

On a récupéré les tâches les plus dominantes afin d'analyser plus en profondeur. A partir de cette étape nous n'utilisons plus les RDD mais des dataframes pythons vu que notre étude se concentre sur 60 tâches (les 15 plus dominantes par catégorie). D'abord nous analysons la répartition des priorités des tâches. Pour cela nous chargeons les données de task_events dans laquelle se situent ces informations. Ensuite, nous avons analysé la corrélation entre l'utilisation moyenne du CPU et de la mémoire pour cela nous utilisons la librairie statsmodel afin de calculer tous les paramètres d'une régression linéaire entre ces deux paramètres.

3. Résultats obtenus

a. Jobs Dominants

Les 10 jobs identifiés comme dominants en usage moyenne du CPU sont :

```
[5501666014, 6184967359, 6243111979, 2509801316, 4974863111, 4974863530, 4974862840, 4974863723, 4974862115, 4974863081]
```

Les 10 jobs identifiés comme dominants en usage maximale du CPU sont :

```
[5022579678, 4665897969, 4665712396, 5501666014, 6184967359, 6176114691, 4665712499, 6251803864, 6127642576, 4974912787]
```

Les 10 jobs identifiés comme dominants en usage moyenne de la mémoire sont :

```
[6024393894, 6184967406, 6184967359, 4472621957, 5840251953, 259235987, 3638155221, 5323518598, 2509801316, 3405957923]
```

Les 10 jobs identifiés comme dominants en usage maximale de la mémoire sont :

```
[6024393894, 6184967406, 6184967359, 4472621957, 5840251953, 259235987, 3638155221, 5323518598, 2509801316, 3405957923]
```

b. Tâches dominantes

Les 10 tâches identifiées comme dominantes en usage moyenne du CPU sont :

job ID	task index
6251620795	5
4665897969	44
6251684909	5
4665897969	46
4665897969	46
6251658528	2
4665897969	228
4665897969	254
6251684909	7
4665897969	191
4665897969	79
4665897969	278
4665897969	262
4665897969	111
6249153835	32
4665897969	222

Les 10 tâches identifiées comme dominantes en usage maximale du CPU sont :

job ID	task index
5022579678	78
4665897969	46
4665897969	46
4665712396	17
4665897969	320
5022579678	224
4665897969	156
4665897969	200
4665897969	61
4665897969	33
4665897969	48
4665897969	242
4665897969	23
4665897969	304
4665897969	255
4665897969	96

Les 10 tâches identifiées comme dominantes en usage moyenne de la mémoire sont :

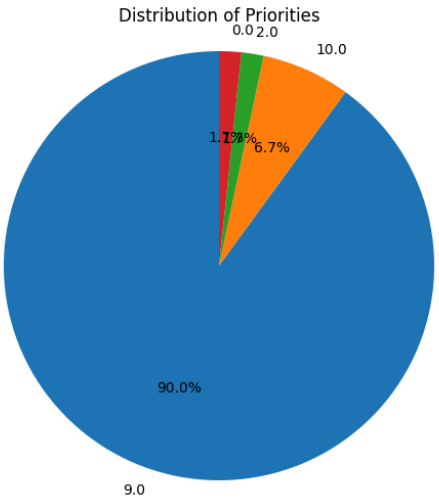
job ID	task index
6024393894	1
6024393894	1
6184967406	1
6184967406	1
6184967406	2
6184967406	2
3638155221	1
3638155221	1
5400430074	7
5400430074	7
5400430074	9
5400430074	9
5400430074	8
5400430074	8
5400434834	6
5400434834	6

Les 10 tâches identifiées comme dominantes en usage maximale de la mémoire sont :

job ID	task index
6024393894	1
6024393894	1
6184967406	1
6184967406	1
6184967406	2
6184967406	2
3638155221	1
3638155221	1
5400430074	7
5400430074	7
5400430074	9
5400430074	9
5400430074	8
5400430074	8
5400434834	6
5400434834	6

c. Priorité des tâches dominantes

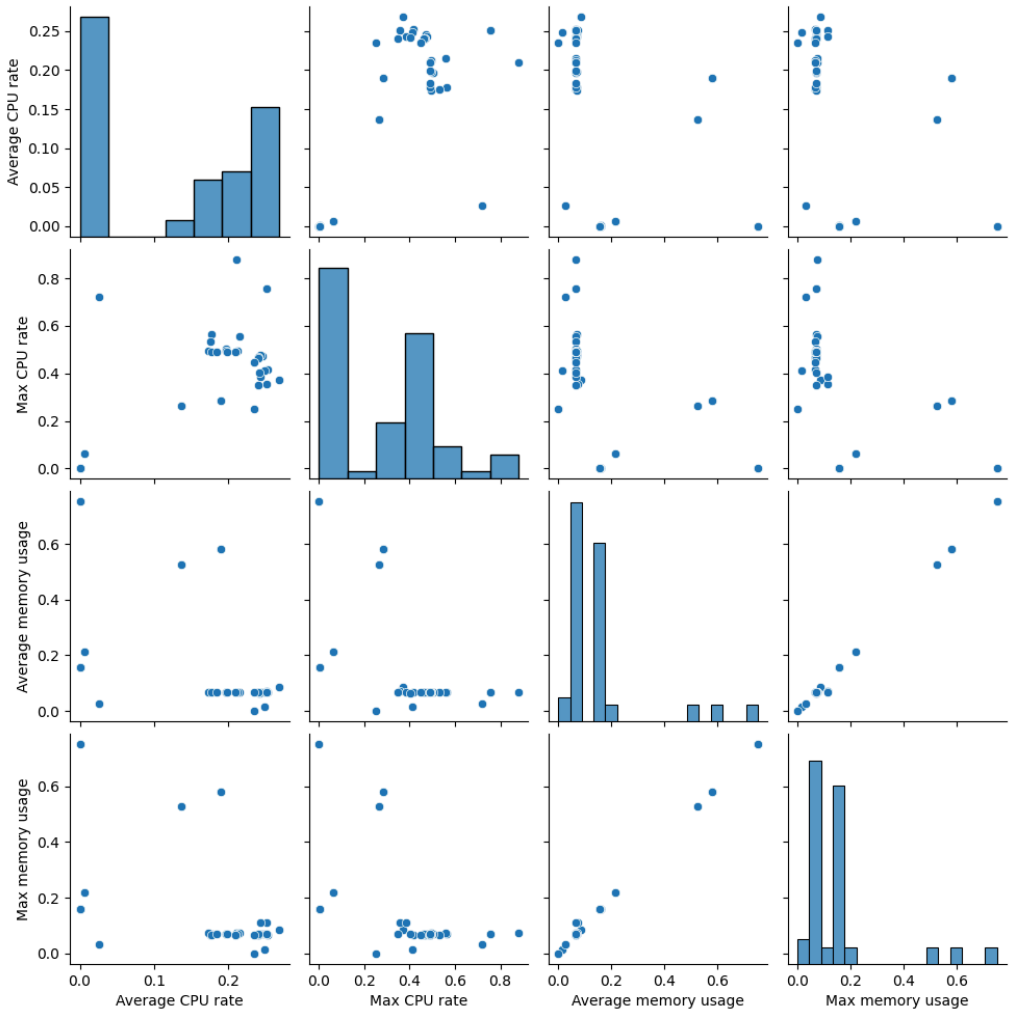
La priorité est l'importance de la tâche, en d'autres termes est-il important de l'exécuter rapidement ou non. Plus la priorité est élevée, plus la tâche sera exécutée rapidement par le cluster. Voici la répartition des ordres de priorité sur les tâches dominantes :



On peut observer que la majorité des tâches dominantes ont une priorité élevée.On note aussi qu'il y a seulement 3 % de tâches dominantes avec une priorité inférieure à 9, leur priorité est d'ailleurs très faible puisqu'elle est de 2 et 0.

d. Corrélation entre consommation mémoire vive et CPU

Sur le graphique ci-dessous on observe qu'il semble avoir une forte corrélation entre l'utilisation moyenne et maximale du CPU, et dans un moindre mesure de la mémoire. Cependant, il ne semble pas y avoir de lien entre la consommation de mémoire et de CPU.

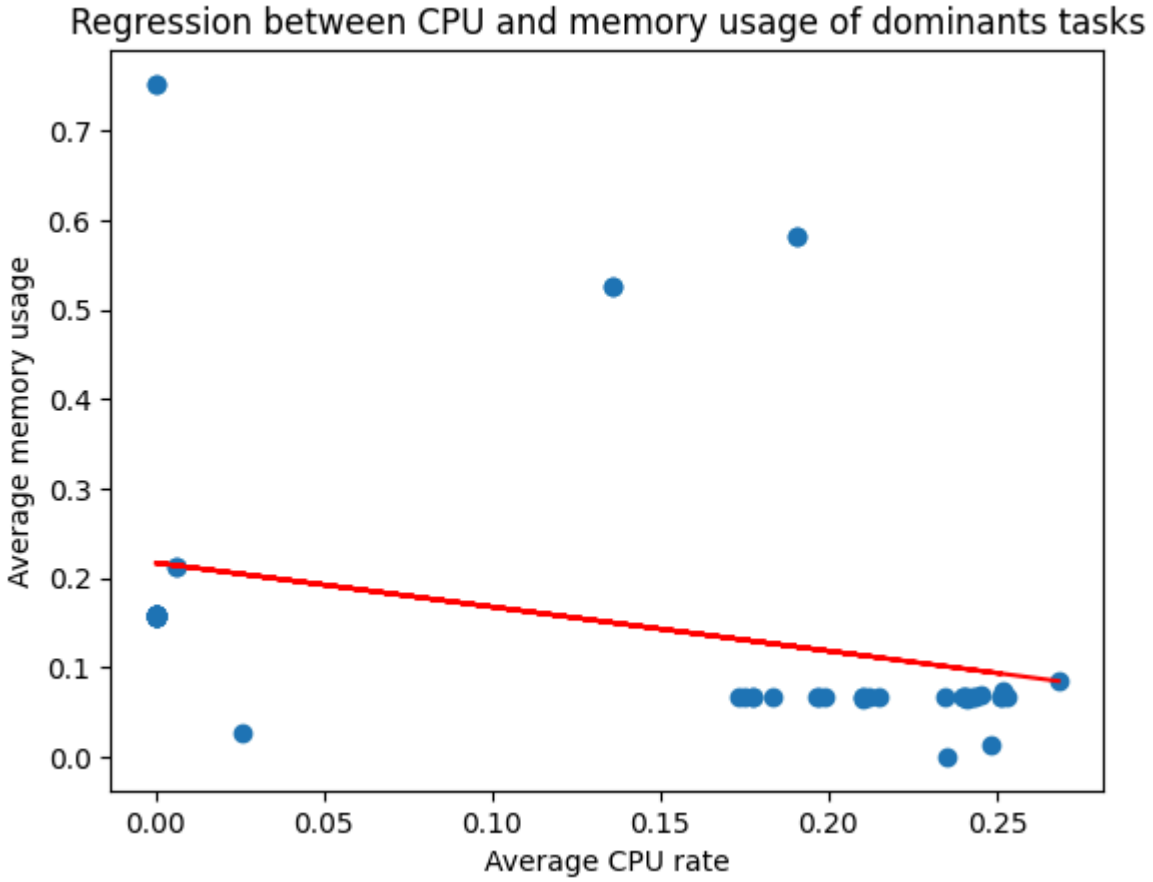


Nous réalisons alors une régression linéaire entre l'utilisation moyenne du CPU et l'utilisation moyenne de la mémoire dont voici les résultats :

OLS Regression Results

Dep. Variable:	Average memory usage	R-squared:	0.108
Model:	OLS	Adj. R-squared:	0.092
Method:	Least Squares	F-statistic:	6.988
Date:	Sun, 23 Feb 2025	Prob (F-statistic):	0.0105
Time:	21:52:28	Log-Likelihood:	26.565
No. Observations:	60	AIC:	-49.13
Df Residuals:	58	BIC:	-44.94
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.2169	0.030	7.182	0.000	0.156	0.277
Average CPU rate	-0.4929	0.186	-2.644	0.011	-0.866	-0.120



On obtient un R^2 de 10.8% qui rejoint l'analyse graphique sur la faible corrélation entre utilisation de la mémoire et utilisation du CPU.

4. Lien

Vous pouvez retrouver les notesbook sur le répertoire GitHub suivant :

https://github.com/tollim311/BDF_Groupe3

Pour exécuter les notebooks, nous vous conseillons d'aller sur GoogleColab pour BDF_projet_Groupe3_GC et sur Deepnote pour BDF_projet_Groupe3_DN