Sujet de TP Systèmes distribués à large échelle

Système de gestion d'ETL à grande échelle

Dans ce TP nous allons construire un système ETL à grande échelle.
1- Qu'est qu'est un ETL (Extract, Transform, Load) ?
1-a) commencer par voir ce qu'est un ETL (voir par exemple https://www.lebigdata.fr/etl-definition)
1-b)- Nous allons concevoir un système de gestion d'ETL à grande échelle. Nous supposons que nous avons un ensemble de tache T1,T2,T3, …(le nom des taches est quelconque).
Chaque tache est définie de la façon suivante :

```
Class T1:
     def __init__(params):
          #params is a dictionary that contains all variables and
          #parameters needed to execute the task. In particular
          #params contains all input values to the task
          ….
     def extract():
          …
     def load():
          ….
```

La variable `params` contient tous les paramètres nécessaires à l'execution de la tache ETL.

1-c) Nous supposons que le fichier `taches.py` contient la définition de toutes les taches. Ainsi un objet de la classe T1 peut être instanciée par la commande :

```
object= taches.T1(params)
```

Il est notable que python permet d'instancier une objet dont le nom est défini dans une variable c'est à dire que les commandes suivantes instancient aussi un objet de la classe T1

```
object_name= 'taches.T1'
object= object_name(params)
```

Ainsi si nous connaissons le nom d'une tache est les paramètres de celle-ci nous pouvons l'executer dans n'importe processus et n'importe ou dans un cluster de calcul.

Tester les concepts précédents sur un ensemble de taches.

1-d)- supposez que vous devez effectuer une séquence de taches ETL, par exemple T1 suivi de T2, suivi de T3, etc sur des données. On peut supposer que la tache T1 génère dans sa méthode load() un dictionnaire contenant `object_name= 'taches.T2'`, et les paramètres d'entrée pour executer la tache T2 et qu'il les met dans une file qu'on appellera `task_queue`. Ensuite le programme lit dans cette file et instancie la prochaine tache à faire. Quand le traitement est fini la file devient vide.

Implantez la conception précédente et testez la.

2- REDIS et cache partagée
Vous savez déjà ce qu'est REDIS et comment utiliser REDIS en utilisant python. Sinon lire

https://www.ionos.fr/digitalguide/hebergement/aspects-techniques/redis-tutoriel/
Et
https://realpython.com/python-redis/

2-a) vous avez besoin d'une instance de REDIS dans un docker. Veuillez donc utiliser un container de REDIS (par exemple celui-ci https://hub.docker.com/_/redis)

Installer une instance de REDIS et assurez vous que vous pouvez y accéder de votre machine par le biais d'un script python

2-b) Nous allons supposer que la file `task_queue` est mise en place dans REDIS sont forme d'une liste (voir dans https://redis.io/docs/manual/data-types/ le type list) et que chaque tache écrit à la fin de son execution dans la list redis `task_queue` une entrée de la forme `task_name:hash_name` ou la première partie réfère au nom de la tache a effectuer et la seconde partie refère au nom d'une variable de hash dans REDIS qui contient le dictionnaire des paramètres nécessaire à l'execution de la tache `task_name`.
Changez le code rédigé dans la partie 1-d pour qu'il puisse utiliser la file de tache dans REDIS.

3)- parallélisme massif

Python ne permet pas de faire du parallélisme au niveau thread.
3-a) pourquoi ? Lisez au sujet de l'infame GIL et comprenez ce que cela signifie

3-b) nous allons utiliser la libraire de multiprocessing plutôt que le multi-threading. Veuillez lire la documentation dans https://zetcode.com/python/multiprocessing/

3-c) nous allons mettre le code dans la partie 2-b dans plusieurs process grace à la librairie multiprocessing. Ainsi chaque process ira sur redis lire une tache a effectuer, effectuera la tache et ré-ecrira le resultat dans la file task_queue. Si un process trouve la file vide pendant une durée de temps de 10 secondes il s'arretera. Le programme est fini quand il n'y plus aucune tache à faire.

Implantez ce mécanisme et testez le.

4-Map Reduce.

4-a)-Nous allons refaire le TP que vous avez fait l'année dernière sur map-reduce (lecture d'un très long texte et comptage du nombre de mot disctint et de l'histogramme des mots.

4-b) refléchissez à comment synchroniser les taches dans cette environnement distribués? Comment faire une tache qui reçoit le résultat de plusieurs sous taches ? Est ce qu'un mécanisme pubsub pourrait être utile ? Comment ?

4-c) voir que REDIS permet de faire du pubsub. Compléter votre code avec la synchronisation par pubsub.

Vous avez maintenant un redoutable système de Système de gestion d'ETL à grande échelle que vous avez développé de vos mains.

Si vous êtes arrivé ijusqu'ici et qu'il faut encore beau, vous pouvez vous acheter une grande glace italienne (pas une glace vénitienne une gélateria) et allez la déguster sur les bords du lac car vous l'avez mérité …..