Multimedia data management course 3

Johnny Nguyen

***Abstract*— Ce document test une synthèse du troisième cours de gestion de données multimedia.**

I. INTRODUCTION

Comment gérer deux flux en parallèle (vidéo + son) ? Nous devons un utiliser un plugin nommé decode bin, il va fournir de l'audio et de la vidéo. Un thread est lancé pour chaque flux (l'exécution est donc en parallèle).

- Prendre le mixeur et mettre le bon paramêtre,
- Prendre la video, lui spécifier sa résolution et l'envoyer au mixeur,

Nous pouvons utiliser ces plugins pour analyser une vidéo en temps réel.

Rappel sur HDFS, Spark et zookeeper.

II. HADOOP AVEC HDFS

C'est un système de fichier distribué, l'intérêt est qu'il y a une redondance de bloc. Il va essayer de le recopier au minimum trois fois. Eviter de faire trop de copie quand même. HDFS n'est pas forcément reconnu par n'importe quel langage. Nativement, nous devons écrire en Java, Spark ou Python (mais il va falloir faire un gros effort). A priori, il y a trois type de serveur :

- Namenode : il va gérer où on met les blocs,
- Secondary namenode : optimiser le contenu pour que l'accès au fichier soit plus rapide,
- Datanode : répondre à toutes les requêtes pour que le namenode réponde correctement,

Le fait de gérer la redondance et les fichiers sur plusieurs machine il va falloir le gérer.

Kerberos permet de sécuriser un système Hadoop. L'authentification est utilisé ici. Le Big Data doit être fait en interne pour que tout soit sécurisé.

WebHDFS pour faire du REST API. Nous utilisons aussi des adresses URI. C'est le protocole classique a effectuer.

YARN utilisé surtout pour des lignes de textes. Un ressource manager (RM) a un rôle de coordonateur.

Map-Reduce découpe les tâches en plusieurs étapes. YARN et HDFS l'utilise.

Streaming permet à Python d'être utilisé pour manipuler Hadoop. Par exemple, nous testons le HDFS, nous répartissons nos jobs sur nos clusters.

L'algorithme pour splitter en Python consiste à capter la vidéo en entrée, décompose la vidéo en image en la distribuant sur HDFS et génère une sortie avec un index (clé) et le nom du fichier vidéo. Le mappage est effectué sur le texte et utilise un fichier temporaire. La réduction collecte et fusionne les résultats sur le HDFS.

III. SPARK

Permet d'effectuer des calculs à partir de données HDFS. A priori trois modes de fonctionnement :

- En *standalone*, c'est à dire, en local avec plusieurs threads et en cluster (sur plusieurs machines). Utilisation des RDD.
- En *Apache-Mesos*, vu avec Fillatre. Distribution Spark sur chaque noeud avec un maître et un esclave.
- En *Hadoop-YARN* pour utiliser HDFS et l'allocation de ressources. C'est moins rapide que celle de Fillatre. Il ne faut pas mettre Spark sur chacune des machines, les échanges sont font nativement directement.

L'idée de **SPARK streaming** est d'ajouter un flux important de données. C'est un découpage temporel. Le context de SPARK doit être démarré au début et droit être arrété à la fin du traitement. Alors que dans le contexte du streaming, il doit être arrêté sans le détruire. La notion de point de rendez-vous, il était évident qu'il y aura des raté, mais cela a été prévu. Nous pouvons redémarrer en conséquence, et reprendre les RDDs qui n'ont pas été traité. Cela permet de savoir l'état à certain moment du traitement en streaming.

A. Zoopkeeper

C'est un service qui va faire de la maintenance et de la distribution d'information. Il va distribuer et synchroniser soit la configuration ou l'authentification. C'est de l'information centralisé pour que ce soit simple et réparti pour avoir de la redondance. Nous pouvons avoir des informations divergentes pour voter pour l'information la plus pertinente. Simplement, c'est un petit plugin qui envoie de petit message pour avoir une faible latence et une source d'information orienté configuration à disposition de l'ensemblement des serveurs distribués.

IV. AUTRES

A. Alluxio

Essaie de résoudre les différents problèmes présent dans SPARK (beaucoup d'échange mémoire à condition d'être sur la bonne machine). Rétablir rapidement une panne. Nativement le SSD est géré. Une interface web et aussi configurable en ligne de commande (Client Java, GO et Python). C'est un des systèmes les plus rapide et les plus fiables.

B. Apache Flume

Un système de log vers un sytème centralisé. Nous avons deux agents qui peuvent collaborer et ils échangent pour quelque chose s'exécute sur une machine à distance. La

plupart du temps, les logs nous permettent de retrouver des variations qui nous intéressent. A partir des éléments récupérés, nous pouvons faire du Machine Learning en particulier.

C. Apache Avro

Permet de sérialiser les données. Aucune programmation n'est demandé.

D. Apache Ranger

C'est la possiblité de protéger, centraliser les informations sur la protection. Les plugins s'adaptent aux serveurs et applications. Toutes les trente seconde le serveur se met à jour. C'est un système descriptif des droits que nous pouvons avoir.

E. Apache Knox

Quel est la différence avec Ranger ? C'est intéressant de tout protéger à partir des requêtes lancés. C'est compliqué, il faut prévoir tout qu'il y a dedans. Si tout est géré, c'est compliqué. Là pour chaque service, nous créons un petit tuyau pour gérer les intéractions, c'est bien mais c'est pas la meilleure solution.

V. Conclusion

Nous nous verrons à l'IUT, nous allons manipulé des fichiers audios et vidéos. Nous allons faire un peu de Map-Reduce, car c'est facile. Un peu de SPARK mais ce n'est moins probant. Pour comprendre faire du Big Data d'une à dix machine. (dix machines permet de savoir le faire sur cent machines). C'est totalement différent sur une machine que sur plusieurs machines. Le problème est que nos connaissances en réseaux et en Python n'est pas optimale. Les quatres séances suivantes seront faites à l'IUT. GStreamer globalement nous ne rencontrons pas de problème mais dès que nous traitons des données Big Data la diffuculté s'étend relativement. Désole, de vous faire finir trop tôt.

References

[1] http://iutsa.unice.fr/ mathieu/unice/SI/