INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE

INP TOULOUSE ENSEEIHT

Projet Infrastructure pour le Big Data

Machine Learning

Auteurs:
Baptiste AUSSEL
Ayoub ALKARIM
Paul BRUNET
Albin CINTAS
Dylan LEBRETON
Julien SALVADOR

Juin 2021

Contenu du document

Introduction

Ce projet s'inscrit dans le cadre du cours Infrastructure et Big Data. Le but est d'implémenter une plateforme de Machine Learning As A Sevice (ML-AAS).
Après avoir implémenter l'infrastructure et sa gestion, il faudra implémenter une interface d'accès à notre service pour permettre une utilisation de celui-ci. Il faudra ainsi mettre en place un cluster de machines virtuelles avec 4 machines virtuelles réparties au minimum sur 2 machines physiques distinctes. Après avoir mis en place le cluster il faudra déployer Spark/Spark-ML. Une dernière étape constitera en la création d'une interface graphique pour permettre à l'utilisateur les actions suivantes:

- Le chargement de la donnée vers HDFS en spécifiant le chemin
- L'apprentissage et la création de différents modèles
- L'utilisation d'un modèle

Ce document a pour but de retranscrire les différentes étapes de ce projet en détaillant nos choix et les freins que nous avons rencontré.

1 Création des machines virtuelles

La première étape de ce projet a été l'installation d'un hyperviseur pour gérer le déploiements de différentes machines virtuelles. Nous avons choisi d'utiliser l'hyperviseur KVM via Virtual Manager. Nous avons créée 4 machines virtuelles sur 2 machines physiques différentes(2 par machine):

- Deux machines virtuelles (sur la machine de Baptiste) possèdent 5 Go de mémoire, 2 CPU et 4 Go de RAM (deux slaves)
- Deux autres machines virtuelles (sur la machine de Julien) possèdent 10 Go de mémoire, 2 CPU et 4 Go de RAM (un slave et un master)

Deux autres machines virtuelles ont été créées (sur la machine de Paul) car nous avions rencontré des problèmes d'espace mémoire et nous souhaitions nous en prévenir.

2 Connexion entre les différentes machines

Le but de cette partie est de créer une communication entre les machines virtuelles de nos deux machines hôtes en utilisant une connexion filaire entre les deux ports Ethernet de nos machines physiques.

Avant cela et afin de faciliter le téléchargement des fichiers d'installation nous avons relié la source de l'interface réseau de nos VMs en NAT afin d'avoir accès à internet. Il suffit ensuite d'exécuter les commandes suivantes pour finir la configuration.

```
ifconfig enp6s0 up
dhclient
```

2.1 Création d'un bridge depuis la machine hôte

La première étape de la mise en place de cette communication entre les machine est la création d'un bridge vers lequel diriger les activités de nos machines. Nous avons pour cela créé un programme bash `install_bridge.sh` à exécuter à chaque démarrage de nos machine hôte qui crée et configure ce bridge.

```
brctl addbr br0
brctl addif br0 enp2s0f0
fconfig br0 up
ifconfig br0 192.168.0.1
```

Dans l'ordre du code ces commandes bash créent un bridge `br0`, le bridge est ensuite relié à `enp2s0f0` correspondant au port Ethernet de la machine, le bridge est ensuite activité et pour finir une adresse est donnée à la machine afin de l'identifier sur le réseau.

Une fois ce bridge créé il suffit de relier la source de l'interface réseau de nos VMs à celui-ci et d'effectuer un adressage de chaque machine à l'aide la commande `ifconfig br0 adress`. Voici-ci dessous le tableau d'adressage de nos machines.

Machine	**Adresse**
VM1 de PM1	192.168.0.1
VM2 de PM1	192.168.0.2
PM1	192.168.0.3
PM2	192.168.0.4
VM1 de PM2	192.168.0.5
VM2 de PM2	192.168.0.6

Table 1: Adresses de bridge des machines utilisées

Une fois les deux PMs reliées avec un cable Ethernet, il est possible avec cet adressage d'accéder à chaque machine présente dans ce même réseau en `ssh` avec la commande `ssh user@adress` suivit d'un mot de passe.

2.2 Simplification des connexions

Afin de simplifier les prochaines étapes du projet il a été important de rendre cette connexion la plus simple possible. Pour cela nous avons renommé nos machines sur le réseau, crée de nouveau utilisateur et pour finir configuré la connexion ssh afin qu'elle s'exécute sans demande de mot de passe.

2.2.1 Nommage des machines

Afin d'éviter à chaque connexion en ssh de devoir rentrer l'adresse de la machine cible il est possible d'associer des noms à chaque adresse dans le fichier `/etc/hosts`. Voici-ci ci dessous le nommage effectué.

Machine	**Nom**	**Adresse**
VM1 de PM1	slave2	192.168.0.1
VM2 de PM1	slave3	192.168.0.2
PM1	*Non Nommé*	192.168.0.3
PM2	*Non Nommé*	192.168.0.4
VM1 de PM2	slave1	192.168.0.5
VM2 de PM2	master	192.168.0.6

Table 2: Adresses de bridge et Nommage des machines utilisées

Une fois le fichier `/etc/hosts` configuré sur l'ensemble des machines, il est maintenant possible de se connecter avec la commande `ssh user@machine` et un mot de passe.

2.2.2 Création de nouveaux utilisateurs

Avec la commande `adduser user` nous avons crée un utilisateur `hadoop` sur chacune des machines virtuelles. En utilisant le même nom de user sur chaque machine il est possible de se connecter en ssh à ce même user sur une autre machine sans avoir besoin de le préciser. Avec cette étape la commande se réduit en `ssh machine` et un mot de passe.

2.3 Connexion par clé de sécurité

La dernière étape est de configurer la connexion afin qu'elle se fasse sans demande de mot de passe. Pour cela nous avons utilisé des clé privé/pubilc associé à chaque user de machine.

Le principe est le suivant, pour qu'un user1 puisse se connecter en ssh sur un user2 sans mot de passe, la clé public du user1 doit être présente dans le fichier des clés autorisés du user2. La clé public se trouve dans le fichier `/home/user/.ssh/id-rsa.pub` et la liste des clé

autorisées se trouve dans le fichier `/home/user/.ssh/authorized_keys`. Ces fichiers de clés se génèrent grâce à l'aide de la commande `ssh-keygen`

Une fois que les fichiers `authorized_keys` de chaque user sont configurés il est alors possible de se connecter en ssh entre machine vers le même user avec la commande simplifiée `ssh machine`

3 Déploiement de Spark-ML

3.1 Installations préliminaires

Tout d'abord, nous avons automatisé les installations nécessaires à l'utilisation de Spark-ML sur les machines virtuelles. Le but était de permettre un travail asynchrone entre les équipes puisque l'installation et la connexion entre les machines virtuelles n'étaient pas configurées au début du projet, il fallait donc pouvoir démarrer le travail sur la partie Spark-ML tout en ayant fixé la configuration des installations sur les machines virtuelles.

Ces installations préliminaires comportent :

- Permettre la connexion à localhost via ssh
- Installation d'hadoop
- Installation de java
- Installation de spark
- Configuration de ./.bashrc pour définir les variables d'environnements suivantes : JAVA_HOME, HADOOP_HOME et SPARK_HOME et réaliser quelques tests
- Vérification des installations
- Configuration des installations

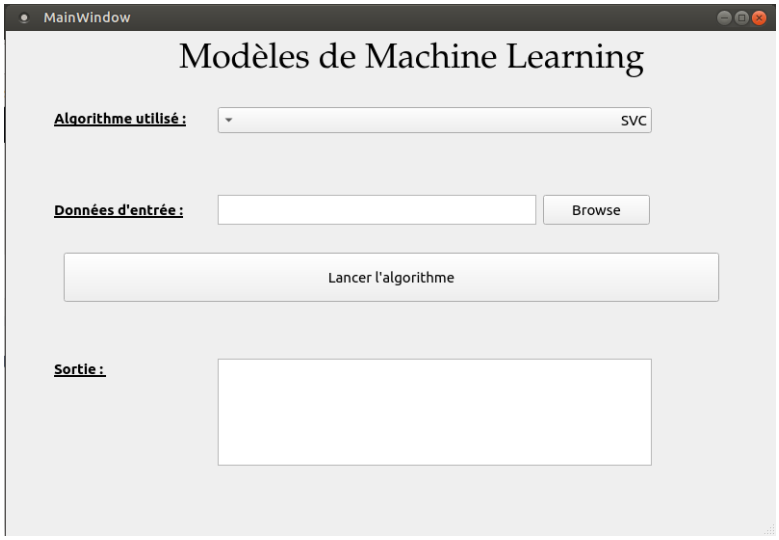
3.2 Application de SparkML pour la reconnaissance d'images

Dans cette partie:

- On a déployé des modèles de Maching Learning pour faire de la reconnaissance d'images sur nos machines virtuelles en utilisant le module SparkML.
- Les modèles déployés sont: RandomForest, Arbre de décision, réseau de neurones et support vector machine (SVC).
- On s'est inspiré du TP SparkML et TP Scalability pour faire le pré-traitement des données et la programmation en Java des modèles ML
- On a crée un fichier DataReader.java pour lire les données et répartir les données en images d'apprentissage et labels.
- On a crée un fichier TestSVMImage.java pour vérifier que les données entrées par l'utilisateur correspondent bien à des images.
- On a crée un fichier DataGenerateFile pour faire le pré-traitment des données et les mettre sous format .txt pour les importer dans hdfs.
- On a crée les fichier <model>.java pour déployer nos modèles.

4 Création d'une interface graphique

Dans cette partie, l'objectif était de créer une interface graphique permettant à l'utilisateur de l'infrastructure d'obtenir un accès graphique à celle-ci. Grâce à cela, l'utilisateur peut ainsi avoir la possibilité d'entraîner un algorithme de Machine Learning sur le réseau précédemment déployé à l'aide de SparkML.
Cette interface graphique permet de choisir dans un premier temps un algorithme parmi les 4 suivants : Régression logistique, Forêt aléatoire, Arbre de décision et Réseau de neurones. Puis dans un second temps, de choisir les fichiers qui seront envoyés sur le master afin d'entraîner le modèle de ML. Enfin, un bouton permet de se connecter au master par ssh et va alors permettre de lancer l'algorithme choisi. Les résultats sont affichés en dessous dans la section "sortie". Vous pouvez retrouver l'interface ci-dessous.



Lors de la mise en place de l'interface nous avons fait le choix de ne pas l'héberger sur un serveur web, mais plutôt de la réaliser en local grâce au langage python. Plus particulièrement, nous avons utilisé la librairie PyQT et l'API PyQT-Designer afin de construire le frontend de l'interface. De ce fait, l'interface commence par envoyer les données sur le master. Elle va ensuite se connecter en ssh au master et réaliser les tâches nécessaires pour faire tourner l'algorithme sur les différentes machines virtuelles.

Lors de ce projet il a été difficile de mettre en lien l'interface et le master, et de permettre à celle-ci de réaliser des commandes shell. Heureusement, la librairie subprocess de python permet de réaliser des commandes à partir de python. De plus,

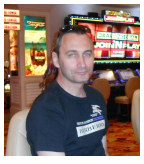
Conclusion

Du déploiement de machines virtuelles à l'élaboration d'une interface graphique en passant par le déploiement et l'utilisation de Spark-ML, ce projet nous aura montré toute la difficulté de la coordination du travail d'équipe. Chaque grande partie de ce projet pouvait demander une certaine abstraction vis-à-vis des autres parties, et, pour autant, nous avons réussi à emboîter le travail de chaque équipe en fin de course.

Certaines améliorations seraient pensables. Par exemple, dans notre cas, l'interface graphique est à installer sur la machine cliente et doit communiquer avec la master du cluster. Nous aurions pu proposer une interface graphique hébergée sur serveur web pour plus de praticité. Aussi, nous pourrions également proposer un service plus abstrait de données, ne se limitant pas à l'étude d'images mais à une structure de données dédiée comme les dataframes.

Malgré ces améliorations possibles, l'intérêt profond du projet était de proposer au client une solution de machine learning distribuée et développée d'un bout à l'autre par nos soins. Mission réussie !

Pour le côté technique, la partie Networking aura posé le plus de problèmes, en effet la configuration d'un bridge et d'interfaces réseaux n'auront pas été simple. Autoriser toutes les connections entre chaque machine et simplifier les noms de nos machines nous a fait gagner beaucoup de temps. L'installation d'Hadoop, Java et Spark n'a pas posé beaucoup de problème. Cependant de mauvaises configurations d'HDFS et Spark ont été dures à remarquer, nous aurions pu nous simplifier la tâche en liant la configuration du master avec celle des slaves afin de gagner du temps. Nous avons eu du mal a différencier exécution locale et exécution via hdfs dans l'implémentation des méthodes en Java. Nous avons eu quelques problèmes pour assurer des commandes d'un script python vers le shell.



MERCI

A VOUS