10/11/2015

Ecole Nationale Supérieure Polytechnique de Yaoundé (5GI)

Génie informatique : annee academique 2015-2016

*Membres du groupe (groupe Phœnix) :*

* *MOUAFFO KENFACK Reine Vanessa*
* *TABUE Roméo*
* *TEGANTCHOUANG TEUKAP Boris Gaël*
* *TUENO FOTSO Steve Jeffrey (chef de groupe)*

Rapport du projet de Cloud Computing

Prise en main de Docker, Hadoop et Storm

Table de matières

[Présentation des Livrables 2](#_Toc435895308)

[Partie1: Cloud Computing 5](#_Toc435895309)

[I- CONTEXTE 5](#_Toc435895310)

[II- PROBLEMATIQUE 6](#_Toc435895311)

[III- SOLUTION 6](#_Toc435895312)

[1) Description de la solution 6](#_Toc435895313)

[2) Déploiement 8](#_Toc435895314)

[IV- Résumé et perspectives 9](#_Toc435895315)

[Partie2: Big Data 10](#_Toc435895316)

[I- Contexte 10](#_Toc435895317)

[II- Problématique 11](#_Toc435895318)

[III- Solution : cas de Hadoop 11](#_Toc435895319)

[1) Description de la solution 11](#_Toc435895320)

[2) Analyse des résultats et conclusion 12](#_Toc435895321)

[WEBOGRAPHIE, BIBLIOGRAPHIE ET REFERENCES 12](#_Toc435895322)

# Présentation des Livr*ables*

Nous avons envoyé par mail une version réduite des livrables (afin d’être dans la limite de 2 MB), qui contient le strict nécessaire pour comprendre ce que nous avons fait et pour effectuer les tests sur le projet concernant Docker (mise en place du cluster et test de l’Annuaire Web). Elle contient, en plus de ce rapport, un dossier Realisations\_Logicielles constitué de :

* **Un dossier « BigDataHadoop »** contenant les réalisations logicielles et les résultats, sous forme de courbes, liés à l’implémentation **sous Hadoop**, de l’application de comptage de mots dans un fichier (WordCount)[**[Section Big Data sous-section Hadoop]**:](#_Solution_:_cas)
  + **Un dossier « Binaires »** contenant les scripts bash d’automatisation des simulations, avec une plage de paramètres, du WordCount sous Hadoop :
    - **« hadoop\_tp.sh »** : qui reçoit des plages de paramètres et le fichier d’entrée et exécute l’implémentation java/hadoop du WordCount pour les différentes valeurs de paramètres prises chacune dans sa plage
    - **« hadoop\_tp\_suite.sh »**
    - **« simulate-hadoop.sh »** : le script principal qui reçoit les plages de paramètres et le fichier d’entrée, appelle hadoop\_tp.sh avec ces plages de paramètres et le fichier d’entrée, récupère les résultats et trace les courbes
  + **Un dossier « Extrait\_Codes\_Sources »** contenant un extrait du code source java du WordCount sous Hadoop :
    - **« WordCount.java »**
  + **Un dossier « Resultats »** contenant les courbes des résultats obtenus à l’issue des simulations
* **Un dossier « BigDataSequentiel »** contenant les réalisations logicielles et les résultats, sous forme de courbes, liés à l’implémentation **séquentielle, en java**, de l’application de comptage de mots dans un fichier (WordCount) **[Section Big Data sous-section Sequentiel]**:
  + **Un dossier « Binaires »** contenant le script bash d’automatisation des simulations, un certain nombre de fois pris en paramètre, du WordCount séquentiel:
    - **« simulate-sequentiel.sh »**
  + **Un dossier « Extrait\_Codes\_Sources »** contenant un extrait du code source java du WordCount sequentiel:
    - **« WordCountSequencial.java »**
  + **Un dossier « Resultats »** contenant la courbe des résultats obtenus à l’issue des simulations : une courbe représentant la répartition, par rapport à la moyenne, des durées d’un certain nombre d’exécution du WordCount sequentiel sur un certain fichier
* **Un dossier « BigDataStorm »** contenant les réalisations logicielles et les résultats, sous forme de courbes, liés à l’implémentation **sous Apache Storm**, de l’application de comptage de mots dans un fichier (WordCount) **[Section Big Data sous-section Apache Storm]**:
  + **Un dossier « Binaires »** contenant le script bash d’automatisation des simulations, avec une plage de paramètres, du WordCount sous Storm :
    - **« simulate-storm.sh »** : script qui reçoit les plages de paramètres et le fichier d’entrée, exécute l’implémentation java/storm du WordCount avec ces plages de paramètres et le fichier d’entrée, récupère les résultats et trace les courbes
  + **Un dossier « Extrait\_Codes\_Sources »** contenant un extrait du code source java du WordCount sous Storm :
    - **« FileSentenceSpout.java »**: l’implémentation du Spout ayant pour source un fichier
    - **« WordSpliterBolt.java » :** l’implémentation du Bolt Découpeur de phrase
    - **« WordCounterBolt.java » :** l’implémentation du Bolt Compteur de mots
    - **« GlobalWordCounterBolt.java » :** l’implémentation du Bolt Compteur Global en charge de la récupération des résultats des WordCounterBolt, de leur synthétisation et de l’écriture du décompte final dans le fichier de résultats
    - **« WordCountTopology.java » :** L’implémentation de la classe principale en charge du découpage du fichier source selon le nombre de Spouts et de la création de la topologie
  + **Un dossier « Resultats »** contenant les courbes des résultats obtenus à l’issue des simulations
* **Un dossier « CloudComputing\_Docker »** contenant les réalisations logicielles liées à l’implémentation d’**un annuaire web J2EE distribué sous Docker** [**[Section Cloud Computing]**](#_Partie1:_Cloud_Computing)**:**
  + **Un dossier « Apache »** contenant les outils de mise en place du container Apache :
    - **« Dockerfile »**: le script d’instanciation du container Apache
    - **« config\_files » :** des fichiers supplémentaires utilisés lors de l’instanciation du container Apache
      * **« 000-default.conf »**
      * **« workers.properties »**
      * **« workers1.properties »**
      * **« apache.sh »**: le script contenant les instructions que la VM apache lancera à chaque démarrage, en particulier le démarrage du serveur Apache2
  + **Un dossier « MySQL »** contenant les outils de mise en place du container MySQL :
    - **« Dockerfile »**: le script d’instanciation du container MySQL
    - **« config\_files » :** des fichiers supplémentaires utilisés lors de l’instanciation du container MySQL
      * **« annuaire.sql»**: le script SQL de création de la BD, des tables, des utilisateurs et des données de base
      * **« mysql.sh »**: le script contenant les instructions que la VM MySQL lancera à chaque démarrage, en particulier le démarrage du serveur mysql
  + **Un dossier « Tomcat »** contenant les outils de mise en place du container Tomcat :
    - **« Dockerfile »**: le script d’instanciation du container Tomcat
    - **« config\_files » :** des fichiers supplémentaires utilisés lors de l’instanciation du container Apache
      * **« server.xml »**
      * **« tomcat7 »**
      * **« tomcat.sh »**: le script contenant les instructions que la VM Tomcat lancera à chaque démarrage, en particulier le démarrage du serveur Tomcat7
  + **Un dossier « Tomcat\_clone »** contenant les outils de clonage d’une VM Tomcat, utilisés dans le cas où plusieurs VM Tomcat sont demandées
    - **« Dockerfile »**: le script de clonage d’une VM Tomcat à partir d’une VM Tomcat existante
  + **Un dossier « Extrait\_Codes\_Sources\_Annuaire »** contenant un extrait du code source du projet J2EE Annuaire Web (uniquement les classes/servlets, les jsp, le web.xml et le script js principal) :
    - **Un dossier « Classes\_Servlets »** contenant les classes/servlets du projet
    - **Un dossier « WEB-INF »** contenant les jsp, le web.xml et le script js principal
  + **« deploy.sh »**: script bash permettant de déployer et démarrer le cluster sous docker en lançant de façon intelligente, les dockerfiles de mise en place des différentes VMs, prenant en paramètre le nombre de tomcats
  + **« nbre\_tomcats »**: fichier utilisé par deploy.sh pour stocker le nombre de tomcats pris en paramètres pour les prochains lancements du cluster
  + **« start.sh » :** script bash en charge du démarrage du cluster sous docker
  + **« stop.sh » :** script bash en charge de l’arrêt du cluster

Vous pouvez retrouver, en vous rendant sur <https://github.com/stuenofotso/Cloud-Computing-Project-Ressources> La version complète des livrables. Cette version complète est obligatoirement nécessaire pour reproduire les simulations concernant les implémentations du WordCount en séquentiel, sous Hadoop et sous Apache Storm car contient, en plus, le code source complet de chacune des implémentations, les binaires « .jar » des implémentations incluant les librairies utilisées, les fichiers de résultats, bref, tout l’attirail que nous avons mis en place durant ces deux mois sur le projet.

# Partie1: Cloud Computing

## CONTEXTE

Le cloud computing (informatique en nuage) est une infrastructure dans laquelle la puissance de calcul et le stockage sont gérés par des serveurs distants auxquels les usagers se connectent via une liaison Internet sécurisée. L’ordinateur de bureau ou portable, le téléphone mobile, la tablette tactile et autres objets connectés deviennent des points d’accès pour exécuter des applications ou consulter des données qui sont hébergées sur les serveurs. Le cloud se caractérise par sa souplesse qui permet aux fournisseurs d’adapter automatiquement la capacité de stockage et la puissance de calcul aux besoins des utilisateurs [1].

Il apparaît donc clairement que le cloud permet de stocker dans “le nuage” nos applications, données et tout autre genre d'informations. Mais avant de conserver nos applications dans ces serveurs, nous désirons être certains que cette dernière qui fonctionne parfaitement en local sur nos machines de développement, fonctionnera tout aussi bien sur le serveur du cloud qui a une architecture et une configuration très différentes de celles de notre environnement de développement. L’architecture est différente dans le sens où elle offre :

* **L’abstraction sur la localisation :** en effet l’application est « quelque part » sur l’une des machines constitutives de la plateforme de virtualisation. Si cette plateforme utilise des mécanismes de réplication sur des Datacenters distants, les risques de désastre (incendies, inondations) sont couverts.
* **L’élasticité :** il est possible d’allouer des ressources supplémentaires à une application proche de la saturation, dans les limites physiques de la plateforme.
* **Le “Pay As You Go”:** on ne paye que pour les ressources utilisées.
* **Le “self-service” :** les ressources sont disponibles dans des délais relativement courts

Etant limités en ressources, il faudrait trouver une solution qui nous permettrait une utilisation optimale de l'espace sur nos machines de développement c'est là qu'intervient Docker.

Docker offre la possibilité de faire des tests qu’on n’oserait pas faire sur sa propre machine, simuler, mutualiser un même serveur en isolant des utilisateurs sur une même machine physique. Il le fait à travers le mécanisme de containers dont il facilite la gestion. Ces derniers pouvant être vu comme des machines virtuelles très basiques, qui vont encapsuler les applications au sein d’un espace isolé. Chaque container va partager le kernel du système d’exploitation qui l’héberge. Docker va donc nous permettre de créer des containers applicatifs,  de les packager, et de les déployer facilement.

## PROBLEMATIQUE

Nous souhaitons dans cette partie prendre en main docker et pour ce faire nous nous sommes reposés sur un petit exemple pour application du concept. L'exemple proposé est divisé en 3 segments:

* un serveur de base de données MySQL pour le stockage des données ;
* un module Tomcat qui va se charger du traitement des requêtes ;
* un module Apache qui s'occupera du load balancing.

Cette divisions en 3 blocs nous permettra d’avoir une bonne visibilité des parties de notre application, facilitera la maintenance et la réutilisation des composants.

Comment mettre en œuvre un outil de déploiement d'une architecture JEE (Apache + Tomcat + BD) sur un ensemble de conteneurs Docker? Telle est la problématique dans cette partie.

## SOLUTION

### Description de la solution

Nous avons mis sur pied un ensemble de scripts permettant de déployer le cluster décrit précédemment sur tout terminal linux. Pour tester le bon fonctionnement de ce cluster une fois déployé, nous avons conçu et développé en J2EE un annuaire web exploitant les 3 étages du cluster et permettant à des utilisateurs de pouvoir consulter, ajouter, modifier et supprimer des contacts dans une base de données MySQL.

Nous rappelons que notre cluster est constitué de:

Une machine contenant essentiellement Apache et plus particulièrement son module de répartition de charge Mod\_JK et permettant de répartir les requêtes des utilisateurs entre les différents serveurs d'application Tomcat

* Une machine contenant essentiellement MySQL, en charge du stockage des données
* Une ou plusieurs machines contenant chacune essentiellement Tomcat7, en charge du traitement des requêtes et de la génération des réponses HTTP

NB : Pour des raisons de rapidité et pour assurer la montée en charge, le nombre de Tomcat peut être paramétré.

Les scripts de déploiement du cluster peuvent se regrouper en deux catégories:

#### Les Dockerfiles

Il s'agit de scripts qui, pour chaque conteneurs, permettent de décrire la séquence d'instructions à exécuter par docker pour mettre en place la VM, de la récupération de l'image de base au script de démarrage en passant par l'installation et la configuration des paquets applicatifs.

A chaque VM (Virtual Machine) du cluster correspond donc un fichier de script dockerfile qui se retrouve dans un dossier portant le nom de la VM ainsi que les fichiers supplémentaires nécessaires à la mise en place de cette dernière:

* VM MySQL:
* Le script de création de la BD, de création de l'utilisateur, d'allocation des droits, de création des tables et d'insertion des données de base
* Le script de démarrage de la VM
* VM Tomcat:
* Fichier servers.xml de configuration des ports d'écoute et des workers
* Le script de démarrage de la VM
* VM Apache:
* Fichier workers.properties de configuration des workers, mis à jour dynamiquement en fonction du nombre paramétré de VMs Tomcat
* Fichier de mapping des urls aux workers
* Script de démarrage de la VM

#### Les Scripts BASH

Nous avons mis sur pied:

Un script global de déploiement de l'architecture (Apache/Mod\_JK + Tomcat + MySQL) en charge du séquencement des actions de création du cluster. Ce script s'appelle deploy.sh et se charge d'appeler les Dockerfiles de création des VMs et de créer les liens entre ces conteneurs ; liens qui sont utilisés au moment du démarrage des conteneurs, afin de procéder à la configuration de ces derniers. Finalement, ce script appelle le script de démarrage du cluster.

NB: Ce script installe et configure docker dans le cas où ce n'est pas encore fait.

* Un script en charge du démarrage du cluster qui se termine en affichant l'adresse URL permettant d'accéder à l'annuaire Web. Nous nous sommes assurés de rendre la prise en main et l'utilisation de cette application vraiment aisées: listing, édition, insertion, suppression de contacts.
* Un script en charge de l'arrêt du cluster

### Déploiement

Notre projet se trouve dans le dossier CloudComputing\_Docker dans lequel on retrouve les dossiers

* MySQL: qui a les fichiers de configuration de MySQL et un Dockerfile qui crée la machine virtuelle MySQL
* Tomcat: qui a les fichiers de configuration de Tomcat et un Dockerfile qui crée la machine virtuelle Tomcat
* Tomcat\_clone: qui gère la multiplicité des Tomcat
* Apache: qui a les fichiers de configuration d'Apache et un Dockerfile qui crée la machine virtuelle Apache

Et les fichiers deploy.sh, start.sh et stop.sh.

**Etapes :**

a) Commencez par vous positionner dans le dossier du projet;

b) Pour le déploiement du cluster:

sudo ./deploy.sh nbre\_de\_parametres

Contraintes: nbre\_de\_parametres >=1

Ceci fait, la création de toutes les machines virtuelles a été faite et les liens entre elles ont été établis. A la fin deploy.sh appelle start.sh qui lance le cluster, vous devez donc copier l'url qui apparaît et le coller dans votre navigateur, ceci vous redirigera directement vers la page de notre application ou vous aurez donc la possibilité d'insérer, éditer, supprimer ou de consulter les contacts de l'annuaire.

Une fois le déploiement terminé

b) si vous souhaitez relancer le cluster:

sudo ./start.sh

c) si vous souhaitez stopper le cluster:

sudo ./stop.sh

## Résumé et perspectives

Il était question dans cette partie de prendre la main avec Docker, pour se faire nous avons implémenté un annuaire web distribué. Cette petite application a été subdivisée en 3 parties : une pour la gestion des données, une autre pour le traitement des requêtes et une dernière pour le load balancing des requêtes. Pour chaque étage de notre annuaire, nous avons eu à créer des containers (machines virtuelles) sous lesquelles il s’exécuterait. Ces machines virtuelles sont instanciées à travers l’exécution de docker files et scripts bash.

Il ressort de notre travail, la méthodologie à utiliser pour une mise en place claire, simple et facilement maintenable sous Docker d’une application avec une architecture multicouche.

Nous avons retenu plusieurs avantages dans l’utilisation de Docker et pouvons le recommander pour :

* Le partage d’un environnement de travail entre plusieurs personnes,
* Garder son système hôte propre, en installant tout ce dont on n’est pas sûr sous Docker,
* Avoir des versions spécifiques d’une librairie, d’un serveur, d’une base de données,
* La création, le paquetage et le déploiement facile des applications.

# Partie2: Big Data

## Contexte

Chaque jour, nous générons 2,5 trillions d’octets de données [2]. A tel point que 90% des données dans le monde ont été créées au cours des deux dernières années seulement. Ces données proviennent de partout : de capteurs utilisés pour collecter les informations climatiques, de messages sur les sites de médias sociaux, d'images numériques et de vidéos publiées en ligne, d'enregistrements transactionnels d'achats en ligne et de signaux GPS de téléphones mobiles. Le Big Data couvre quatre dimensions: volume, vélocité, variété et véracité [2].

Volume : les entreprises sont submergées de volumes de données croissants de tous types, qui se comptent en téraoctets, voire en pétaoctets.

* Transformer les 12 téraoctets de Tweets créés quotidiennement en analyse poussée des opinions sur un produit
* Convertir les 350 milliards de relevés annuels de compteurs afin de mieux prédire la consommation d'énergie

Vélocité : parfois, 2 minutes c'est trop. Pour les processus chrono sensibles tels que la détection de fraudes, le Big Data doit être utilisé au fil de l'eau, à mesure que les données sont collectées par l’entreprise afin d'en tirer le maximum de valeur.

* Scruter 5 millions d'événements commerciaux par jour afin d'identifier les fraudes potentielles
* Analyser en temps réel 500 millions d’enregistrements détaillés d’appels quotidiens

Variété : le Big Data se présente sous la forme de données structurées ou non structurées (texte, données de capteurs, son, vidéo, données sur le parcours, fichiers journaux, etc.). De nouvelles connaissances sont issues de l’analyse collective de ces données.

* Utiliser les centaines de flux vidéo des caméras de surveillance pour contrôler les points d'intérêt
* Tirer parti de la croissance de 80 % du volume de données image, vidéo et documentaires pour améliorer la satisfaction client

Véracité : 1 décideur sur 3 ne fait pas confiance aux données sur lesquelles il se base pour prendre ses décisions [2]. Comment s’appuyer sur de l'information en laquelle on n’a pas confiance? Etablir la confiance dans les Big Data représente un défi d'autant plus important que la variété et le nombre de sources augmentent.

## Problématique

Il est question ici d’étudier les performances de deux implantations d’une application de comptage de mots avec deux solutions du Big Data: Hadoop et Apache Storm.

L'application doit prendre en entrée un grand fichier texte et délivrer en sortie un fichier contenant une liste de tuples <mot, décompte>. Pour chaque implantation, le travail principal consiste à chercher à obtenir les performances maximales avec le même nombre de processeurs; ainsi, il s’agit de rechercher les paramètres permettant d'atteindre les meilleures performances.

Pour Hadoop, nous allons varier la taille des blocs, de 1MB à 1GB par puissance de 2, et mesurer les temps d’exécution pour un fichier texte de 900MB.

Pour Apache Storm, nous allons varier le nombre de Spouts et le nombre de Bolts Découpeurs de 1 à 7 par pas de 2 et le nombre de Bolts Compteurs de mots de 1 à 101 par pas de 10 et mesurer les temps d’exécution pour un fichier texte de 30 MB. Réaliser cette tâche nécessitant de mettre en place une implémentation de l’application de comptage de mots où les spouts lisent les phrases dans un fichier passé en paramètres et surtout qui est capable de s’arrêter à la fin du fichier (le fichier étant de taille dynamique) et afficher la durée de son exécution.

Vu l’immensité du nombre de paramètres, nous avons aussi cru nécessaire de mettre en place des scripts d’automatisation de la collecte des données, du tracée des courbes et de la sauvegarde de celles-ci sous forme d’images exploitables à des fin d’analyse.

## Solution : cas de Hadoop

### Description de la solution

Nous avons utilisé l’application Java WordCount telle que donnée en cours.

L'objectif étant de trouver la taille des blocs manipulés dans les datanodes par Hadoop permettant de minimiser le temps de traitement, plusieurs scripts bash ont été développés, permettant, du moment où Hadoop est correctement installé et configuré, à partir de la plage des valeurs de la taille des blocs, du pas de variation dans la plage et surtout du nom du fichier source, d’automatiquement :

* Stopper les daemons d’Hadoop s’ils étaient déjà lancés ;
* formater le système de fichiers hdfs ;
* Démarrer les daemons ;
* Créer les répertoires dans le système de fichiers hdfs nécessaires à l’exécution (*/user/$USER/input) ;*
* Exécuter l’application Java WordCount pour chaque valeur de la taille des blocs dans la plage ;
* De récupérer les différents temps d’exécution et de tracer les courbes correspondantes

Il s’agit de hadoop\_tp.sh, hadoop\_tp\_suite.sh et simulate-hadoop.sh présents dans le sous répertoire Binaires du répertoire BigDataHadoop.

Pour refaire la simulation pour une plage et un fichier texte donnés, il faut télécharger la version complète des livrables, se rendre dans le sous dossier Binaires du répertoire BigDataHadoop, *contenant launch\_hadoop\_perf.jar (servant au tracé des courbes à partie du fichier de résultats), wc.jar, hadoop\_tp.sh, hadoop\_tp\_suite.sh et simulate-hadoop.sh,* et lancer la simulation en saisissant la commande :

chmod 755 \* #pour donner les droits d’exécution sur les scripts

./simulate-hadoop.sh taille\_min pas taille\_max nom\_fichier

Où taille\_min est un entier, puissance de 2, supérieur ou égal à 1048576 représentant la valeur minimale de la taille des blocs, pas est un entier, puissance de 2, représentant le quotient de la division entre deux valeurs successives de la taille des blocs et taille\_max est un entier, puissance de 2, supérieur à taille\_min, représentant la valeur maximale de la taille des blocs.

L’on obtient en sortie un fichier de données donnant pour chaque taille de blocs le temps mis pour l’exécution ainsi qu’un fichier image, capture de la courbe de variation du temps d’exécution en fonction de la taille de blocs.

### Analyse des résultats et conclusion

Nous avons fait varier la taille des blocs, de 1MB à 1GB par puissance de 2, et nous avons mesuré les temps d’exécution pour un fichier texte de 900MB. Chaque mesure s’est faite 10 fois, pour réduire les risques d’erreurs et les résultats finaux sont la médiane des résultats obtenus.

# WEBOGRAPHIE, BIBLIOGRAPHIE ET REFERENCES

[1] <http://www.futura-sciences.com/magazines/high-tech/infos/dico/d/informatique-cloud-computing-11573/>

The Docker Book by James Turnbull, 16/08/2015, version 1.8.0

[2] https://www-01.ibm.com/software/fr/data/bigdata/