Abstract

Le but de cette partie est d'implanter une application avec storm. Celle-ci prend en entrée un grand fichier texte et devra produir en sortie un fichier contenant une liste de tuples <mot, comptage>. Il sera donc question dans cette partie de faire varier les spouts afin de paralléliser les entrées-sorties et les Bolts pour paralléliser les traitements dans le but d'obtenir une meilleur performances.

Introduction

Apache Storm est un système en temps réel. Il traiter de manière fiable les flux de données illimitées , faisant pour le traitement en temps réel ce que Hadoop a fait pour le traitement par lots. Storm est simple, peut être utilisé avec n'importe quel langage de programmation. Cela dit une question demeure sur sa performances. Pour répondre à cette question, nous menerons une étude basée sur l'implantation d'une application matérialisant ses conceptes. Notre analyse s'articulera sur les points suivants :

Installation et configurations

Prérequis

Plateformes

- Les plateformes **GNU/Linux** sont supportées comme plateformes de développement et de production. Windows est également supportée comme plateforme de développement.
  - Nos expérimentations ne se basent que sur Linux, notamment **Ubuntu 15.04**

logiciels

- **Java** doit être installée. Les versions recommandées de java sont décrites à l'adresse http://wiki.apache.org/hadoop/HadoopJavaVersions
  - Nous utilisons la version **1.8.0_45 de la jdk**
- **maven**
  - Nous utilisons la version **3.19.0-15-generic**

Installation des logiciels requis

Si vous n'avez pas les logiciels requis, vous devez les installer. Sur Ubuntu, utilisez les commandes suivantes :

```
$ sudo apt-get install openjdk
$ sudo apt-get install maven
```

ajout des librairies necessaire au système

```
$ sudo apt-get install git –y
$ sudo apt-get install libtool -y
$ sudo apt-get install automake -y
$ sudo apt-get install uuid-dev -y
```

- **$ sudo apt-get install g++ -y**
- **$ sudo apt-get install gcc-multilib -y**
- **$ sudo apt-get install libtool-bin -y**
- **$ sudo apt-get install tree**

Téléchargement et configurations globales du système

Téléchargement

La dernière version stable de Apache Storm est téléchargeable à l'adresse : http://storm.apache.org/downloads.html . Pour nos expérimentations, nous avons utilisé la version **apache-storm-0.9.5.tar.gz** .

Nous devons également télécharger zookeeper : http://zookeeper.apache.org/releases.html

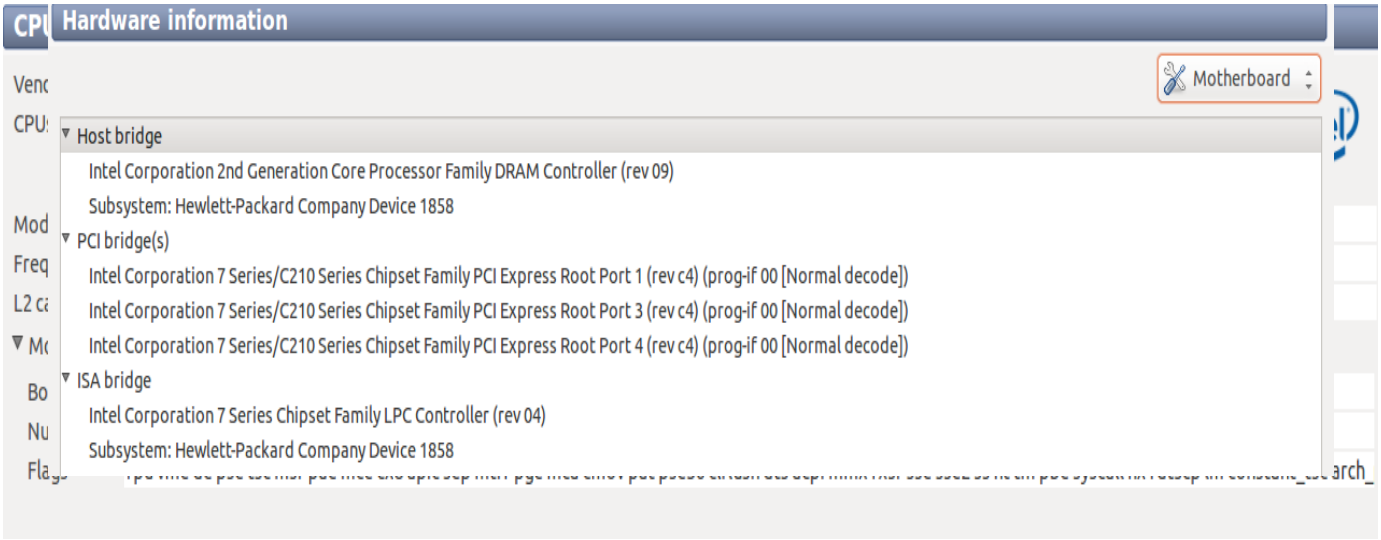
Configurations globales du système

A ce niveau, il s'agit majoritairement de la configuration de certaines variables et de leur ajout dans la variable d'environnement. Les commandes ci-dessous feront le travail.

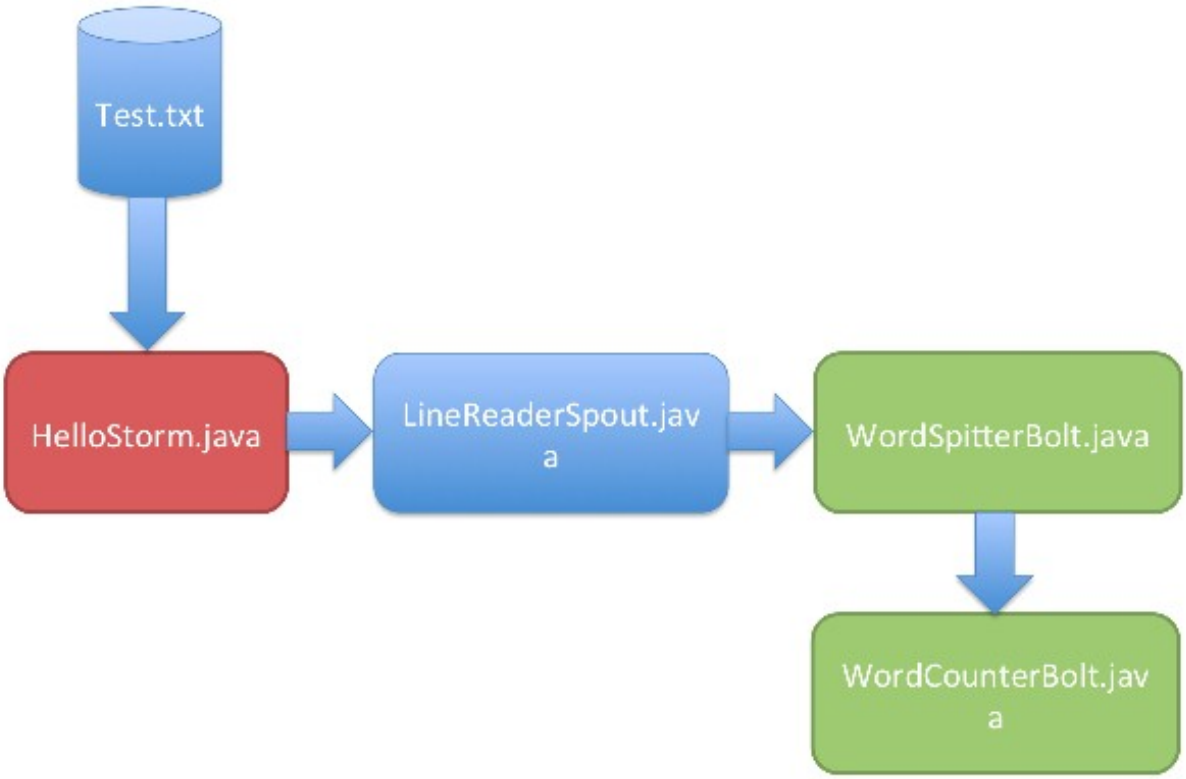
```
sudo gedit ~/.bashr

# Exporter la variable JAVA_HOME et JAVA_JRE
# Ces repertoires sont spécifiques à ma machine, bien vouloir les adapter
#à la votre

export JAVA_HOME=/home/uhkombeul/mywork/jdk1.8.0_45
export JAVA_JRE=/home/uhkombeul/mywork/jdk1.8.0_45/jre
export PATH=$PATH:$JAVA_JRE/bin
export PATH=$JAVA_HOME/bin:$PATH

# Exporter la variable STORM_HOM

export STORM_HOME=/home/uhkombeul/mywork/storm-student/apache-
storm-0.9.5
export PATH=$PATH:$JAVA_HOME/bin:$STORM_HOME

# Exporater la variable ZOOKEEPER_HOME

export PATH=$PATH:$JAVA_HOME/bin:$STORM_HOME/bin:
%ZOOKEEPER_HOME/bin
```

implémentation

Propriétés matérielles de la machine qui a servi de test



Architecture de l'application



Nous avons donc en entrée un fichier texte . Une classe HelloStorm.java qui est la classe principale de l'application. Un spout LineReaderSpout qui est charger de recuperer les lignes du fichier et des les transmetre à notre premier bolt (WordSpitterBolt) qui est chargé de normaliser les mots. Un deuxième bolt (WordCounterBolt) qui est chargé de compter les occurrences de chaque mots. Notons également que nous aurons un fichier en sortie qui aura le resultat.

Préparation des test

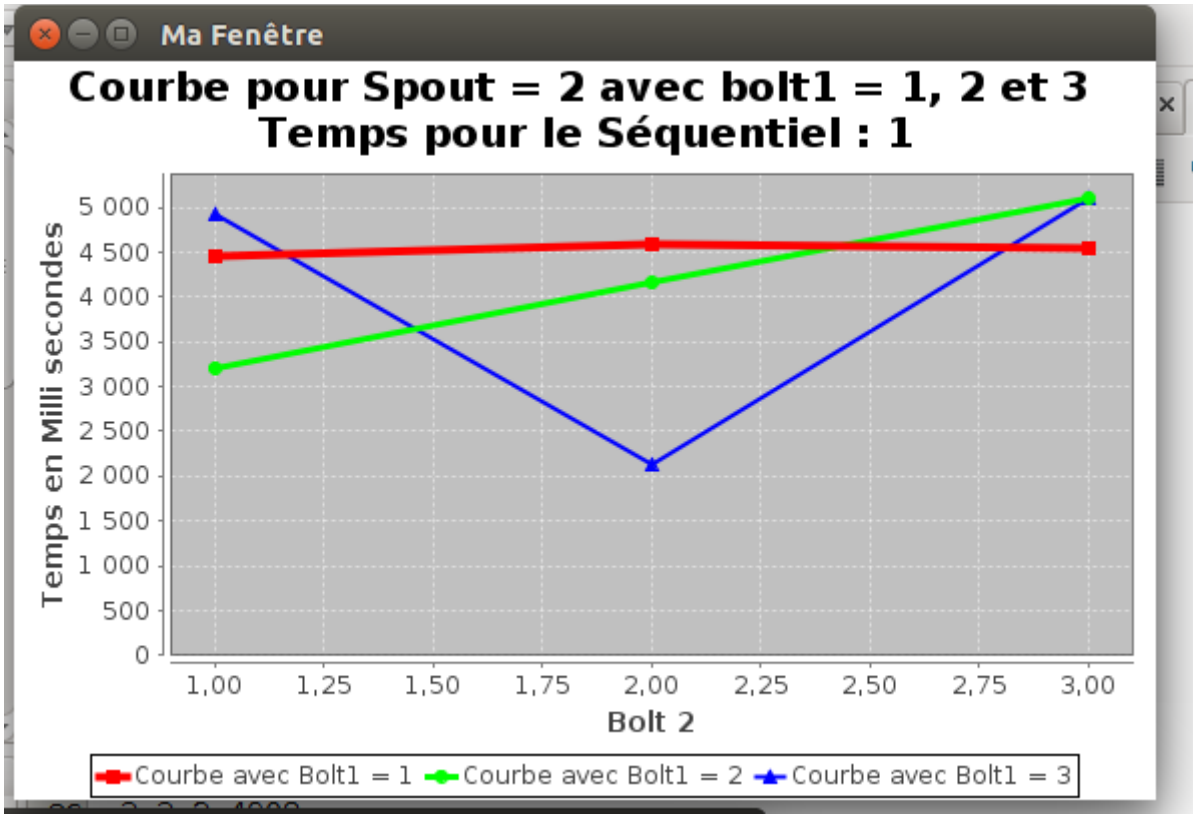
Nous avons modifier la wordcounter de storm en créant trois classe : une pour le spout et deux pour les deux bolts. Nous avons également ajouté le parametre de temps qui est représenté dans notre programme par la variable static long time qui permet de donner le temps d'exécutions.
Nos données de test serons constituées des fichiers (3M0, 5M0, 10M0, 91M0), en entrée, nous ferons varier le nombre de spouts et bolts. A chaque spouts et bolts sera associé un temps d'éxécution. Les tableaus suivants contientiennent donc le nombre de spouts, bolt1, bolt2, le temp d'une exécution et son équivalent en mode séquentiel.

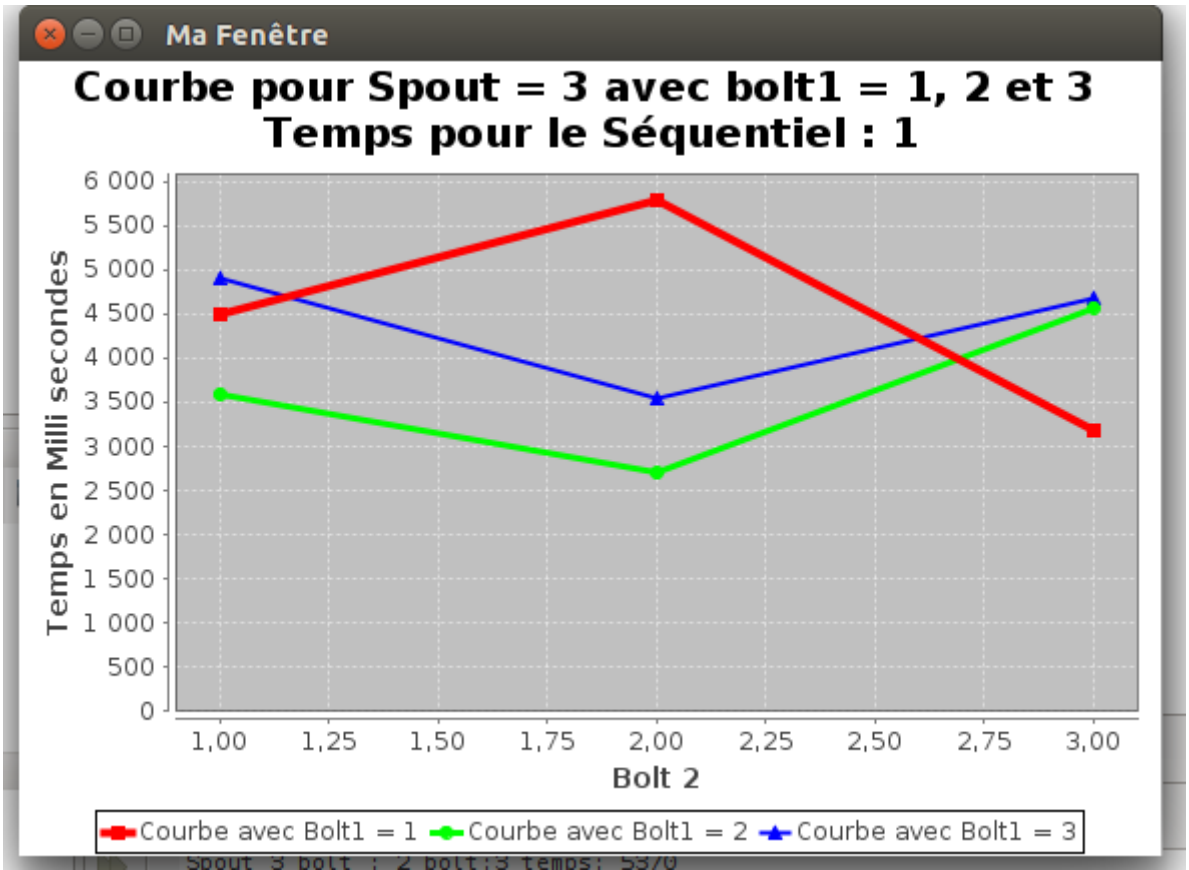
Fichier de 3M0

Taille du fichier	Spouts (LineReaderSpout)	Bolt1(Word SpitterBolt)	Bolt2 (WordCounterBolt)	Temps d'exécution (ms)	Temp d'éxécution en mode séquentiel (ms)	Variation de temps (ms) (temps sequentiel – temps avec storm)
300Ko	1	1	1	7714	327	Baisse
300Ko	1	1	2	4589	327	Baisse
300Ko	1	1	3	4539	327	Baisse
300Ko	1	2	1	4455	327	Baisse
300Ko	1	2	2	4159	327	Baisse
300Ko	1	2	3	5110	327	Baisse
300Ko	1	3	1	3208	327	Baisse
300Ko	1	3	2	2120	327	Baisse
300Ko	1	3	3	5116	327	Baisse
300Ko	2	1	1	4916	327	Baisse
300Ko	2	1	2	5789	327	Baisse
300Ko	2	1	3	3182	327	Baisse
300Ko	2	2	1	4485	327	Baisse
300Ko	2	2	2	2697	327	Baisse
300Ko	2	2	3	4570	327	Baisse
300Ko	2	3	1	3589	327	Baisse
300Ko	2	3	2	3534	327	Baisse
300Ko	2	3	3	4686	327	Baisse
300Ko	3	1	1	4898	327	Baisse
300Ko	3	1	2	5832	327	Baisse
300Ko	3	1	3	5367	327	Baisse
300Ko	3	2	1	6256	327	Baisse
300Ko	3	2	2	4319	327	Baisse
300Ko	3	2	3	5370	327	Baisse

300Ko	3	3	1	4215	327	Baisse
300Ko	3	3	2	4908	327	Baisse
300K0	3	3	3	6276	327	Baisse

Présentation des courbes de test

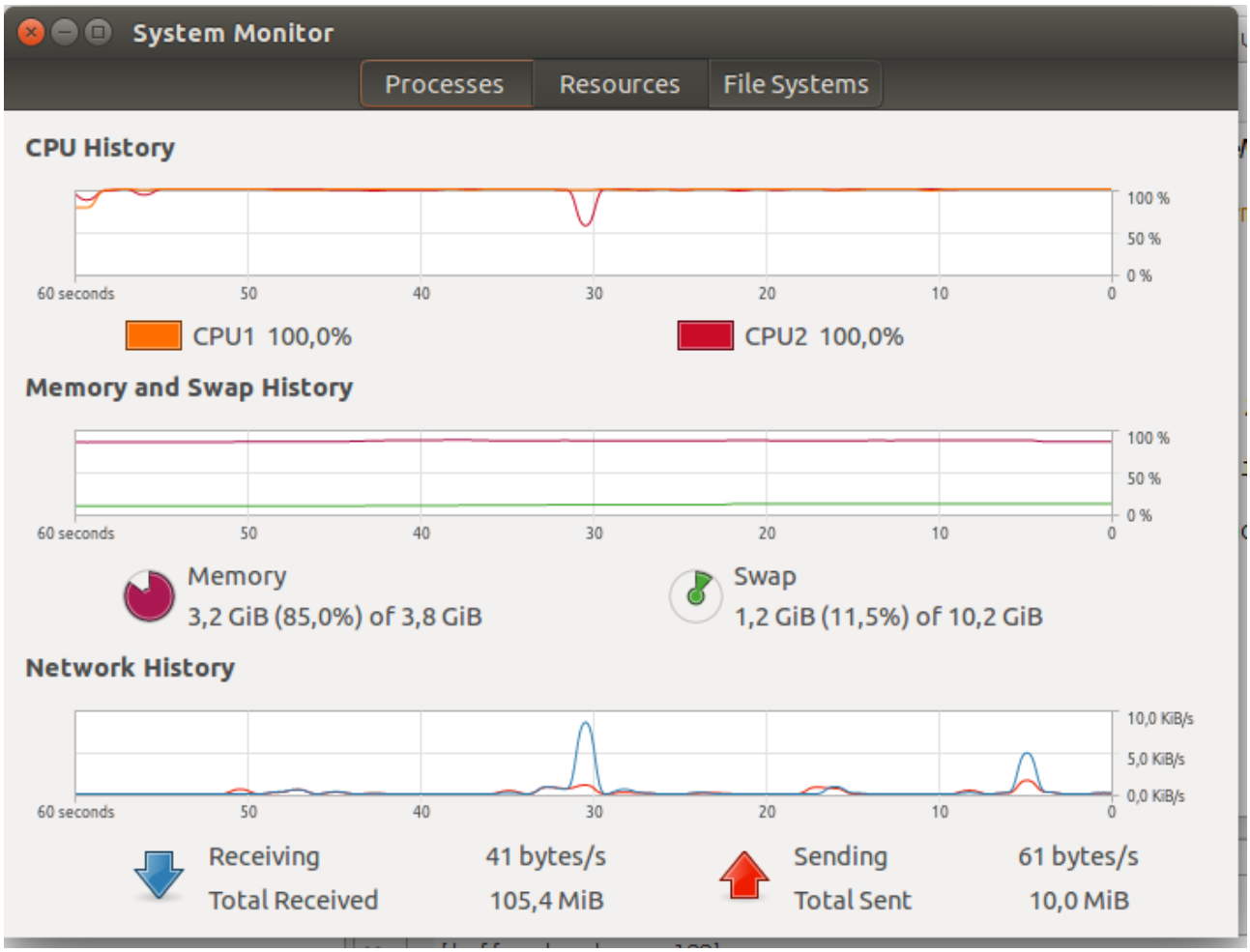




Analyse des résultats

D'après les résultats précedents nous pouvons dire que storm demande beaucoup de ressource pour fonctionner de façon optimale. Le temps d'éxécution séquentiel est supérieur au temps d'exécution avec storm pour des fichier de petites tailles. Cependant lorsque la tatille du fichier augmente considérablement storm devient plus rapide. Pour des fichiers assez lourd (par rapport à la configuration matérielle de la machine) nous constatons que le resultat est interompus et même biaisé car tous les mots ne sont plus comptés. Il est préférable d'utiliser storm dans un cluster pour mieux repatir les tâches.

Pour la configuration matérielle de notre machine lorsque nous augmentons la taille du fichier elle plante voir la figure suivantes pour un fichier de 20Mo



Conlusion

En définitive nous pouvons dire que storm est adapté pour un grand volume de donnée. Cependant il est très gourment en ressouce. Donc le résultat de Storm dépend du matétériel ; c'est la raison pour laquelle il est préférable de l'utiliser dans un réseau d'ordinateur pour profiter de ses alogorithmes.