SEANCE 1

I. Architecture des ordinateurs

Comment comprendre l'output de la commande lscpu ?

Mon adresse ip : 192.168.56.101

```
hadoop@ubuntu: $ lpconf
lpconf: command not found
hadoop@ubuntu:~$ lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                3
On-line CPU(s) list:   0-2
Thread(s) per core:    1
Core(s) per socket:    3
Socket(s):             1
NUMA node(s):          1
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 142
Model name:            Intel(R) Core(TM) i5-8365U CPU @ 1.60GHz
Stepping:              12
CPU MHz:               1896.000
BogoMIPS:              3792.00
Hypervisor vendor:     KVM
Virtualization type:   full
L1d cache:             32K
L1i cache:             32K
L2 cache:              256K
L3 cache:              6144K
NUMA node0 CPU(s):     0-2
Flags:                 fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflu
sh mmx fxsr sse sse2 ht syscall nx rdtscp lm constant_tsc rep_good nopl xtopology nonstop_tsc pni pc
lmulqdq ssse3 cx16 pcid sse4_1 sse4_2 x2apic movbe popcnt aes xsave avx rdrand hypervisor lahf_lm ab
m 3dnowprefetch fsgsbase avx2 invpcid rdseed clflushopt
hadoop@ubuntu:~$
```

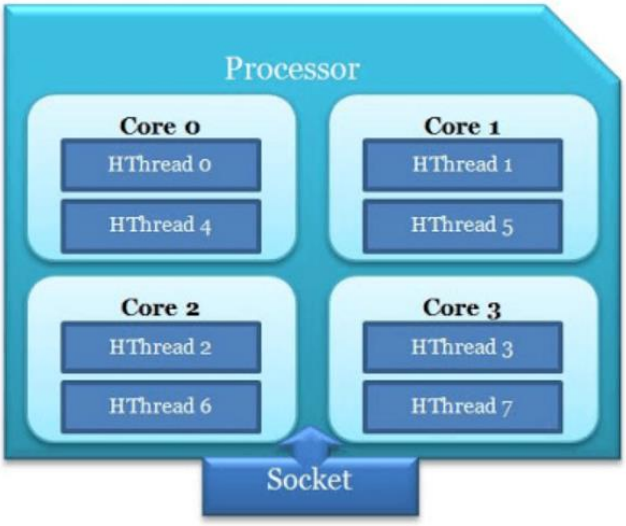
Un socket CPU : connecteur physique sur la carte mère auquel un seul processeur physique (socket CPU) est connecté. Une carte mère possède au moins un socket CPU.

Un CPU ou processeur. C'est le circuit électronique à transistors qui est connecté à une prise. Une CPU exécute des instructions pour effectuer des calculs, exécuter des applications et accomplir des tâches. Lorsque la vitesse des processeurs s'est approchée de la barrière thermique, les fabricants ont modifié l'architecture des processeurs et ont commencé à produire des processeurs à plusieurs cores de processeur.

Un core de processeur est la partie d'un processeur contenant le cache L1. Le cœur du processeur effectue des tâches de calcul de manière indépendante sans interagir avec d'autres core et composants externes d'un « gros » processeur qui sont partagés entre les core. Fondamentalement, un core peut être considéré comme un petit processeur intégré au processeur principal qui est connecté à un socket.

L'hyper-threading est une technologie développée par les ingénieurs d'Intel pour apporter le calcul parallèle aux processeurs qui ont un cœur de processeur. Les débuts de l'hyper-threading remontent à 2002, lorsque le processeur Pentium 4 HT est sorti et positionné pour les ordinateurs de bureau.

Donc CPU(s) = nb socket * nb processeur * nombre de cores *nb treads
Si chaque core a 2 threads, cette architecture représente donc *4 cores physiques et 8 cores logiques appelés CPU(s)*



Un système d'exploitation détecte un processeur monocore avec hyper-threading comme un processeur à deux cœurs logiques (pas des coresphysiques). De même, un processeur à quatre cœurs avec hyper-threading apparaît à un système d'exploitation comme un processeur à 8 cœurs.

II. Des ordinateurs aux nœuds et clusters

Le nœud (node) est l'unité informatique physique autonome réelle dans un cluster informatique distribué. Il possède ses propres processeurs, mémoire, et stockage.

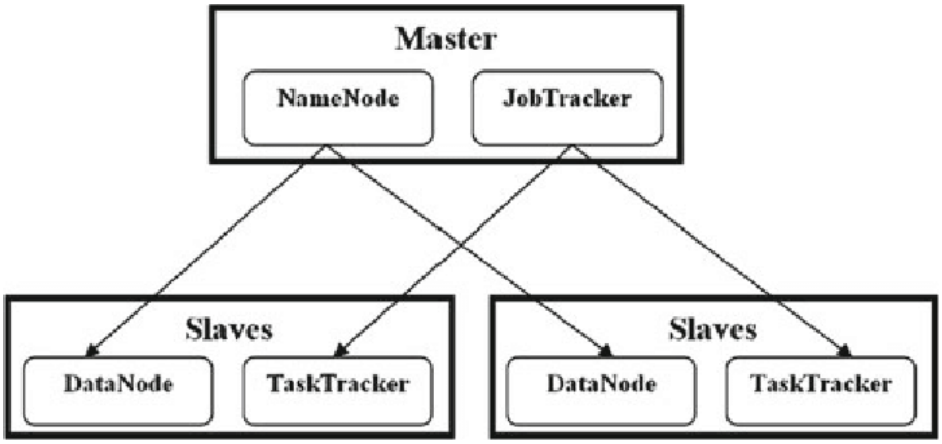
Lorsque les informaticiens utilisent le terme «nœud», cela signifie généralement un **serveur**, qui peut avoir plusieurs processeurs partageant une mémoire monolithique mais apparaissant comme un seul ordinateur pour un programmeur d'applications.

Chaque nœud est une machine ou un serveur à entité unique.

Lorsque plusieurs nœuds sont configurés pour effectuer un ensemble d'opérations, nous l'appelons Cluster.

Exemple : Un cluster hadoop

Un cluster Hadoop typique comprend un seul Master node et plusieurs slaves nodes . Le nœud maître qui est un capitaine du cluster se compose du suivi des tâches, du suivi des tâches et du nœud de nom.

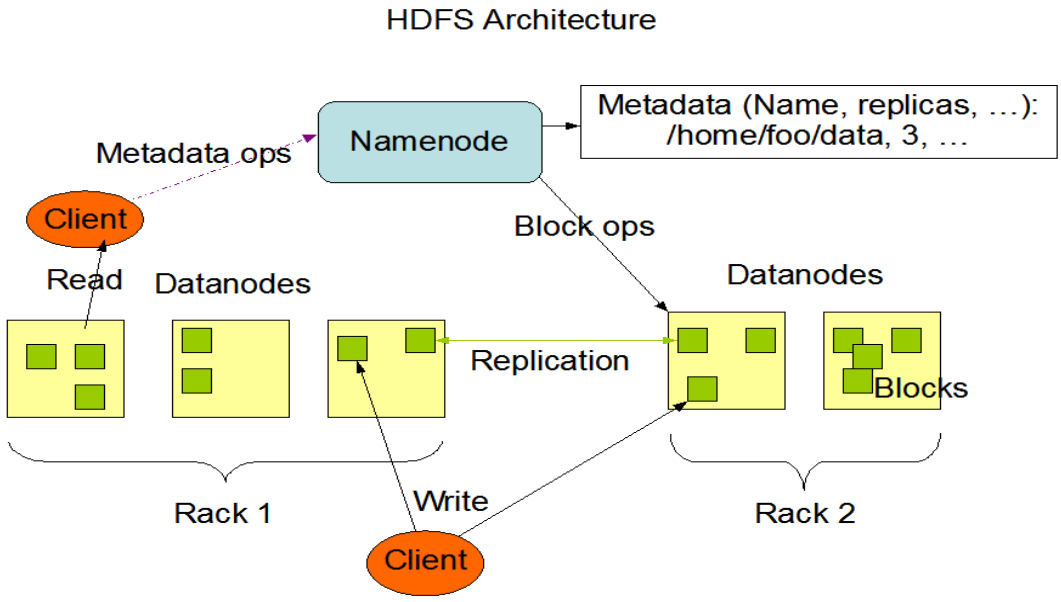


III. Hadoop et Map Reduce

Hadoop est un framework pour stockage et processing pour Big Data.

1. Stockage

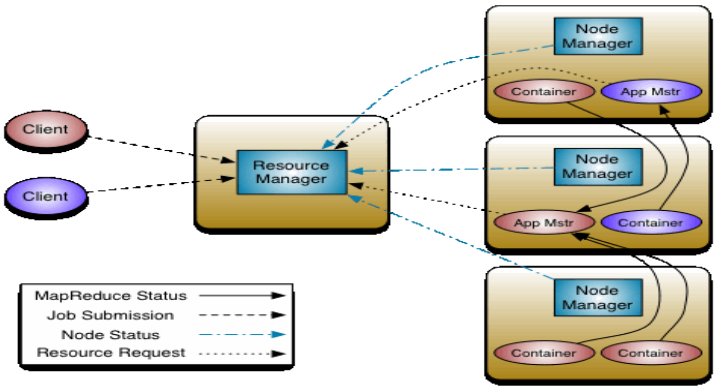
HDFS est le principal stockage distribué utilisé par les applications Hadoop. Un cluster HDFS se compose principalement d'un **NameNode** qui gère les métadonnées du système de fichiers et de **DataNodes** qui stockent les données réelles.



Source : https://i.stack.imgur.com/yoBCX.png

2. Processing

Map reduce est un framework de traitement et d' des données distribuées. The fundamental idea of MRv2 is to split up the two major functionalities of the JobTracker, resource management and job scheduling/monitoring, into separate daemons.



Source : https://i.stack.imgur.com/Bej3P.gif

IV. Premiers pas notre Serveur

1. Connexion à Oracle virtual box
2. Ajouter l'image du VM Ubuntu server Single node
3. Allocation de ressources au VM
4. Demarré le VM
5. Recuperation l'adresse ip du VM « ipconfig » « ip addr show »
6. Se connecter sur putty avec l'ip
7. Lancer les services hadoop

   - Lancer le data file system : start-dfs.sh (pour arrêter le dfs : stop-dfs.sh)
   - Lancer le ressource manager Yarn : start-yarn.sh (pour arrêter yarn : stop-yarn.sh)
   - Contrôler les processus Java : jps
   - Commande groupée : start-dfs.sh && start-yarn.sh / stop-dfs.sh && stop-yarn.sh

8. Accéder aux interfaces

Après avoir récupérer l'IP du VM (xxx.xxx.xxx.xxx), il est possible d'accéder aux interfaces web en utilisant les adresses suivantes (les services étant déjà lancés sur le serveur) :

- Web UI Hadoop du namenode(hdfs) : xxx.xxx.xxx.xxx:50070
- Web UI Hadoop du moniteur d'applications : xxx.xxx.xxx.xxx:8088
- Web UI Hbase du master : xxx.xxx.xxx.xxxx:16010
- Web UI du moniteur d'application SPARK : xxx.xxx.xxx.xxx:4040
- Web UI du Master Spark : xxx.xxx.xxx.xxx:8080 (si mode spark cluster)
- Notebook jupyter : xxx.xxx.xxx.xxx:8888

Pour accéder au dossier partagé sur le VM, tapez dans l'explorateur de fichier l'adresse suivante : \\xxx.xxx.xxx.xxx\public\

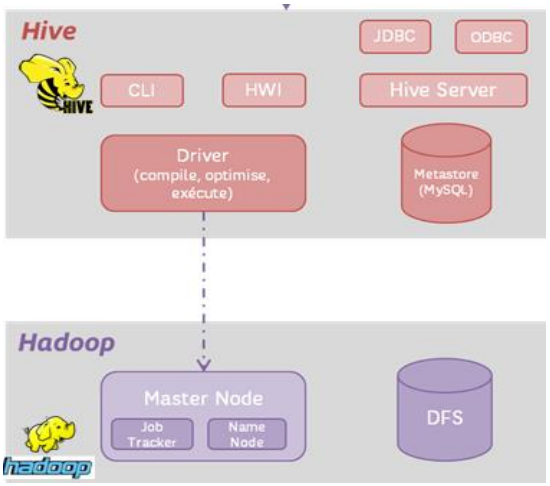
V. HDFS : Hadoop Shell

Le shell hadoop permet d'interagir avec les fichiers présents sur le HDFS (trop sembabl aux commandes lunix). Ci-dessous quelques Exemples de commandes:

- Parcourir le HDFS : **hadoop fs –ls**
- Créer un dossier : **hadoop fs mkdir**
- Copier un fichier sur le HDFS : **hadoop fs –put sourcelocale desthdfs**
- Copier un fichier à partir du HDFS : **hadoop fs –get sourcehdfs sourcelocale**
- Effacer un fichier sur le HDFS : **hadoop fs –rm nomdufichier**
- Effacer un dossier sur le HDFS : **hadoop fs –rm r nomdudossier**
- Contrôle des données sur le HDFS : **hdfs fsck /**
- Killer un job hadoop : **yarn application –kill « application-id »**
- Se déplacer dans un dossier : **hadoop fs –cd « chemin »**
- Déplacer des éléments : **hadoop fs –mv « origine » « destination »**

VI. HIVE : big data warehouse

Hive est une solution de datawarehousing qui reposer sur l'infrastructure Hadoop. Les données y sont structurées et représentées comme en tables. Les données sont manipulées en HQL une syntaxe très proche du langage SQL.



Les composants Hive

Driver Hive : Permet d'interpréter, de compiler, d'optimiser et d'exécuter des commandes HQL.

Metastore : composant indispensable à Hive. Il s'agit d'une base de données (généralement MySQL) annexe permettant de stocker la structure, l'emplacement, le schéma et les métadonnées des tables Hive.

Les interfaces EXTERNES

Le CLI **(**Command Line Interface) est le moyen le plus simple d'interagir avec Hive. Il s'agit d'une interface de type commande, dans laquelle il est possible d'entrer des requêtes Hive.

HWI **(**Hive Web Interface) est une interface Web simple permettant d'accéder à Hive à distance.

JDBC **(**Java DataBase Connectivity) et ***ODBC*** **(**Open Database Connectivity) servent à de lancer des requêtes Hive à partir d'autres applications (par exemple R, SAS), grâce à leurs drivers respectifs.

Avec Hive, il s'agit d'un Schema on read, lire tous éléments de dossiers, lire tous le dossiers. Traditionnement schema on write. Mise à jour régulier de la base de données

Exemple :

CREATE DATABASE UPEC_2022;

Use upec_2022;

```
CREATE TABLE svi_data (

        calldate string,
        calltime string,
        callqueue int,
        product string,
        callnumber string,
        callid int )
        ROW FORMAT DELIMITED
        FIELDS TERMINATED BY '\;'
        STORED AS TEXTFILE;
```

```
LOAD DATA INPATH '/user/hadoop/data/svi/' INTO TABLE svi_data;
```

```
SET hive.cli.print.header=true;
```

Autres notes

- Un node = une machine(ou un serveur)
- **Hadoop** = Une couche hdfs : gérer les fichiers en local et Une couche YARN : gestion des calculs
- **Kafka : ingestion de données et data streaming**
- **Echantillon descript pour se déplacer dans le vm et hadoop**

```
hadoop fs  -rm  -r /user/hadoop/data
```

```
hadoop fs  -mkdir /user/hadoop/data
```

```
hadoop fs  -mkdir /user/hadoop/data/svi
```

```
hadoop fs  -put dat_svi_data.csv user/hadoop/data/svi/
```

hadoop fs -put _datasets/dat_svi_data.csv **/user/hadoop/data/svi/**

- **Dans la racine users, il y a hadoop et hive**
- **Par defaut , ce fichier est deposé sur hadoop sans être transformé**
- **Un node = worker (un node peut avoir plusieurs workers)**

Le véritable problème est la scalabilité ! pas la parralelization!!

SPARK MET DIRECTEMENT EN MEMEOIRE

Lancer spark-sql avec yarn : spark-sql --master yarn

le nombre de partitions : joue sur la rapidité

SET spark.sql.shuffle.partitions=8;

Possibibilite de lancer du spark sans passer par yarn et d'ailleurs c'est la plus optimale

Si tous les nœuds on install spark.

http://192.168.56.101:50070/

http://192.168.56.101:8088/