Homogeneous and heterogeneous architecture for distributed processing of unstructured data with framework hadoop

Javier Artuto Rosas Huacho, Claudio Isaias Huancahuire Bravo, Guido Bravo Mendoza

Abstract— Se requiere nueva tecnología de almacenamiento, para el contexto de sensores, Web 2.0-YouTube, internet de las cosas, redes sociales (facebook, twitter, whatsApp), conllevando exponencialmente a grandes volúmenes de datos, al tratamiento de velocidades extramadanente rápidas y son datos de formatos que no tienen estructura. En compendio se genera un desafío en una dicción titulada “Big Data”, que el SQL no satisface. La propuesta es diseñar e implementar un servidor de mejor prestación para “Big Data”, logrando así dos clústeres de arquitectura de 10 PC homogéneas y 10 PC heterogéneas basados en el framework Hadoop bajo el modelo cliente/servidor en base a Hardware Commodity, HDFS que almacena de manera distribuidad y YARN que procesa en paralelo con el modelo de programación MapReduce. para ello se descargo el código binario de Hadoop 2.9.2, se instalo en sistema operativo RedHat-CentOS7, se compiló el JDK, logrando configurar Java, continuamos con la seguridad SSH-RSA, creando así un servidor de mejores prestaciones para “Big Data”. Las pruebas de rendimiento se realizaron en nuestro servidor localhost, con una población de 6.4 GB y 12.8 GB. Estimando integrar un servidor con PC de escritorio convencionales, como máximo 4000 nodos y no solo con las mismas características de PC.

Palabras clave: HDFS, MapReduce, Hadoop, YARN.

1 INTRODUCCIÓN

Debido al avance tecnológico de almacenamiento de datos digitales en el mundo actual se incrementan de manera exorbitante dichos datos englobando en una dicción “Big Data”, generando un nuevo desafío para almacenar, procesar y analizar un gran volumen de datos. Las tecnologías tradicionales no se convierten en una solución adecuada para procesar. Datos e información irrefutablemente esenciales en esta sociedad digital para tomar decisiones en contexto de Industria, Salud, Transporte, Universidades, Empresas, MYPES y Negocios. Los datos se almacenaban únicamente con formatos estructurados en Excel, Spss, SQL- Server, Oracle, MySql, etc. con equivalencia al 20%. Sin embargo, desde los formatos semi-estructurados y no estructurados. Provenientes de Internet, Web 2.0(YouTube), Google, Amazon, redes sociales (Facebook, Twitter, WhatsApp e Instagram) técnicamente definidos en unidades de PetaByte, ExaByte, ZettaByte, YottaByte, BrontoByte los trata exclusivamente Big Data. Son equivalentes al 80%. Por ello es inevitable e innegable la proliferación exponencialmente alta, de naturaleza heterogénea y con la velocidad a la que se generan dificultan una toma de decisión óptima. La solución diseñada no es por medio de una supercomputadora de Empresas como IBM, Amazon Web Service, Oracle, Microsoft Azuze, Cloudera y Horton-Works. Sino a través de computadoras de precios módicos unidas en redes utilizando tecnología Hadoop (código abierto) la cual permite procesar a las aplicaciones en miles de nodos escalables horizontalmente. Apache Hadoop se ha convertido en la plataforma elegida para el desarrollo de aplicaciones de datos a gran escala. Utilizando desde uno nodo a 4.000 nodos en la versión 1 de Hadoop. Ahora mejora su rendimiento con la administración de recurso YARN.

2 METODOLOGÍA

2.1 HADOOP Y JDK

Se requiere los 2 archivos en binarios y/o ejecutables

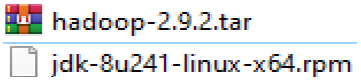

Fig. 1. Hadoop y jdk.

Se crear carpeta y cambiar propietario a hadoop.

```
[root@localhost opt]# mkdir hadoop
[root@localhost opt]# chown hadoop hadoop
[root@localhost opt]#
```

Fig. 2. Configurar hadoop.

Configurar y ejecutar JDK para CentOS 7.

```
[root@localhost ~]# cd /home/hadoop/Descargas
[root@localhost Descargas]# ls -l
total 375076
-rw-r--r--. 1 hadoop hadoop 214092195 may  8 05:48 hadoop-2.7.3.tar.gz
-rw-r--r--. 1 hadoop hadoop 169983496 nov  4 01:54 jdk-8u131-linux-x64.rpm
[root@localhost Descargas]# rpm -ivh jdk-8u131-linux-x64.rpm
```

Fig. 3. Configurar JDK.

Arhivos ejecutables, librerías, licencias de Hadoop.

Fig. 4. Contenido de Hadoop

Corregir la configuración del path de Java, para Hadoop.

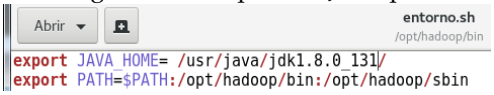

Fig. 5. Path de Java

Seguridad con ssh-keygen, nos permite generar clave pública y clave privada de cada nodo.

```
[hadoop@localhost ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/hadoop/.ssh/id_rsa):
Created directory '/home/hadoop/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/hadoop/.ssh/id_rsa.
Your public key has been saved in /home/hadoop/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:sCUhPlPMA6h59+VVBTF0cPkxISW7sIzd49eWs/F99Og hadoop@localhost.local
The key's randomart image is:
+---[RSA 2048]----+
|    .o+o     .*B=+.|
|   .. o+.     .+=o |
| o  + o..   o . .o|
|o . .o =. = + . .|
| . . ..oSo + +   |
|       . .   . . +|
|              . B+|
|               o.O|
|              .E.+|
+----[SHA256]-----+
[hadoop@localhost ~]$
```

Fig. 6. Ssh-keygen

Resultado de la intefaz de Hadoop

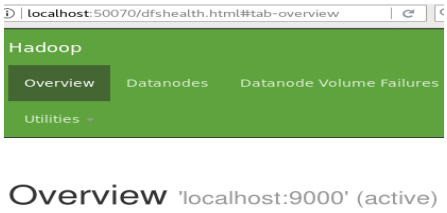

Fig. 7. Inicio de Hadoop

HDFS- Es el sistema de archivos distribuidos de Hadoop

```
<configuration>
        <property>
                <name>fs.defaultFS</name>
                <value>hdfs://localhost:9000</value>
        </property>

</configuration>
```

Fig. 8. Fichero core-site.xml.

La cantidad de réplicas, en los nodos esclavos. Para tener el control de tolerancia a fallas, podemos tener la cantidad de replicas en los nodos esclavos y para ello podemos configurar el fichero de HDFS.

```
<configuration>
        <property>
                <name>dfs.replication</name>
                <value>1</value>
        </property>
        <property>
                <name>dfs.namenode.name.dir</name>
                <value>/datos/namenode</value>
        </property>
        <property>
                <name>dfs.datanode.data.dir</name>
                <value>/datos/datanode</value>
        </property>
</configuration>
```

Fig 9. Fichero hdfs-site.xml.

Comando mkdir como super usuario, Creando los demonios de namenode y datanode.

```
root@localhost:/
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
[hadoop@localhost ~]$ su - root
Contraseña:
Último inicio de sesión:sáb nov  4 05:12:21 -05 20
[root@localhost ~]# cd /
[root@localhost /]# mkdir datos
[root@localhost /]# cd datos
[root@localhost datos]# mkdir namenode
[root@localhost datos]# mkdir datanode
[root@localhost datos]# cd ..
[root@localhost /]# chown -R hadoop:hadoop datos
[root@localhost /]#
```

Fig 10. Namenode y datanode

Pero antes de arrancar el clúster, ubicar el sistema de fichero Hadoop, con el comando hadoop namenode -format del nodo local.

```
hadoop@localhost:/opt/hadoop/etc/hadoop
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
[hadoop@localhost ~]$ cd /opt/hadoop/etc/hadoop
[hadoop@localhost hadoop]$ hadoop namenode -format
```

Fig. 11. Formatear Hadoop.

En el sistema de directorio /sbin se encuentra el script de arranque de HDFS, entonces ahora ejecutamos el comando start-dfs.sh de HDFS, incluido el namenode y datanode y secondarynamenode.

```
[hadoop@localhost hadoop]$ cd /opt/hadoop/sbin
[hadoop@localhost sbin]$ start-dfs.sh
Starting namenodes on [localhost]
```

Fig. 12. Iniciar con HDFS

Confirmar los procesos java que tenemos ejecutando, dentro del jdk se encuentra con el comando jps.

```
[hadoop@localhost sbin]$ jps
4544 NameNode
5225 Jps
4957 SecondaryNameNode
4654 DataNode
[hadoop@localhost sbin]$
hadoop@localhost:/opt/hadoop/sbin
```

Fig. 13. Confirmar la ejecución con JPS

YARN. ahora necesitamos el archivo mapred-site.xml.template. como plantilla para configurar.

```
<configuration>
        <property>
                <name>mapreduce.framework.name</name>
                <value>yarn</value>
        </property>
</configuration>
```

Fig. 14. Configurar MapReduce a YARN.

El siguiente fichero a editar es con el contenido siguiente de 3 propiedades como son: El gestor del manager y el nombre del host manager, la aplicación manager, de servicios y que clase manager.

```
<configuration>

<!-- Site specific YARN configuration properties -->
        <property>
                <name>yarn.resourcemanager.hostname</name>
                <value>nodo1</value>
        </property>
        <property>
                <name>yarn.nodemanager.aux-services</name>
                <value>mapreduce_shuffle</value>
        </property>
        <property>
                <name>yarn.nodemanager.aux-services.mapreduce_shuffle.class</n
                <value>org.apache.hadoop.mapreduce.ShuffleHandler</value>
        </property>
</configuration>
```

Fig. 15. Configurar Servidor.

Arrancar el YARN. Primero arrancamos con el HDFS, con el comando start-dfs.sh, los demonios a encontrar son:

```
3457 Jps
3079 DataNode
3271 SecondaryNameNode
2958 NameNode
```

Fig.16. Los demonios de HDFS.

Segundo con el comando start-yarn.sh iniciamos los proceso java de YARN, levantado todos los procesos java o demonios a encontrar en un solo nodo maestro con el comando jps.

```
[hadoop@localhost ~]$ jps
3616 ResourceManager
3079 DataNode
3271 SecondaryNameNode
3932 Jps
2958 NameNode
3743 NodeManager
[hadoop@localhost ~]$
```

Fig. 17. Los demonios del clúster en un solo nodo.

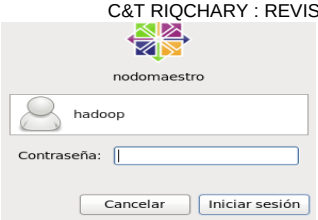

Fig. 18. Sesión de inicio.

El modelo MapReduce se basa en 6 fases estas son: primero la entrada, cada nodo cargaría los bloques con los datos que tuviesen en su sistema de ficheros HDFS localmente. Segundo el split, obtiene una unidad de trabajo, par clave-valor, que comprende una sola tarea Map. Tercero Map, procesa los pares y produce un conjunto de pares intermedio. Cuarto shuffle y sort, estos resultados intermedios (pares) son agrupados y ordenados por clave. Quinto reduce, esta obtención ordenada de pares en clave es aún procesados por otra serie de tareas denominado REDUCE para producir el resultado.

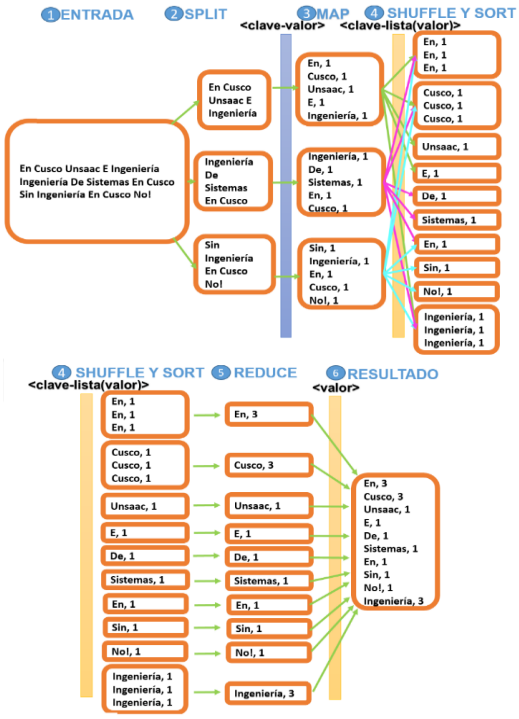

Fig. 19. Ejecución lógica del procesamiento Hadoop v2

2.2 INFRAESTRUCTURA HOMOGÉNEA DE HADOOP.

La infografía muestra 10 nodos, con características homogéneas, el primer nodo es maestro y los 9 nodos son esclavos.

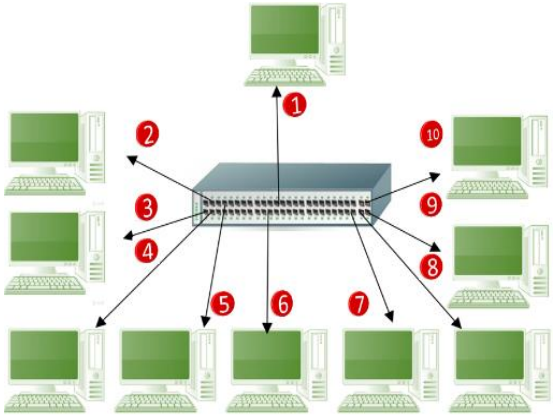

Fig. 20 Clúster homogéneo

TABLA 1

Características de cada PC de clúster homogéneo.

Descripción	Carácteristicas
Almacenamiento	Sobre disco
Procesa datos	Batch
Sistema	Lenovo.
RAM.	16 GB.
Disco HDD.	Intel(R)Xeon(R)
CPU.	64 bits.
Sistema Operativo.	Centos 7.
Hadoop.	Versión 2.9.2
Java	Version JDK 1.8.
Dirección IP	Clase C.

TABLA 2

Tiempo de rendimiento en nodos homogéneos (fuente propia)

Nodos-PC	Tiempo de rendimiento
1	5 minutos y 33 segundos.
3	2 minutos y 29 segundos.
5	1 minutos y 29 segundos.
7	1 minutos y 18 segundos.
10	1 minutos y 2 segundos.

2.3 INFRAESTRUCTURA HETEROGÉNEA DE HADOOP

La infraestructura con las mismas caracteristcas no es vasto, para obtener un servidor de mejores prestaciones, se requieres de diferentes características de PC. La infografía de figura 21, muestra 10 nodos, con características heterogéneas, el primer nodo es maestro y los 9 nodos son esclavos, con sus diferentes características.

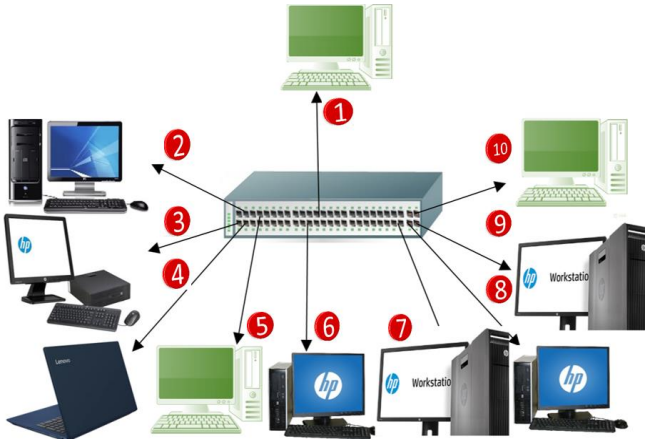

Fig. 21. Clúster heterogéneo

TABLA 3

Descripción	Carácteristicas PC 1,5 y 10
Sistema	Lenovo
RAM	16 GB
Disco HDD	1 T
Procesador	Intel(R)Xeon(R)
Arquitectura	64 bits
Sistema Operativo	Centos 7
Hadoop	2.9
Java	version JDK 1.8
Dirección IP	Clase C

TABLA 4

Descripción	Carácteristicas PC 2
Sistema	ECS
RAM	8 GB
Disco HDD	1 TB
Procesador	Intel(R) core (TM) i5- 3330 CPU @ 3.00Ghz
Arquitectura	64 bits
Sistema Operativo	Centos 7
Hadoop	2.9
Java	version JDK 1.8
Dirección IP	Clase C

Publicado por Ing. Informática y Sistemas - UNAMBA

TABLA 5

Descripción	Carácteristicas PC 3
Sistema	HP
RAM	10GB
Disco HDD	1 TB
Procesador	Intel(R) core(TM) i5-2500 CPU @ 3.30Ghz
Arquitectura	64 bits
Sistema Operativo	Centos 7
Hadoop	2.9
Java	version JDK 1.8
Dirección IP	Clase C

TABLA 6

Descripción	Carácteristicas PC 4
Sistema	Lenovo
RAM	4 GB
Disco HDD	500GB
Procesador	Intel(R)core(TM) i5-4300U
Arquitectura	64 bits
Sistema Operativo	Centos 7
Hadoop	2.9
Java	version JDK 1.8
Dirección IP	Clase C

TABLA 7

Descripción	Carácteristicas PC 6 y PC 8
Sistema	Lenovo
RAM	16 GB
Disco HDD	1,8 T
Procesador	Intel(R)core (TM) i7-7700
Arquitectura	64 bits
Sistema Operativo	Centos 7
Hadoop	2.9
Java	version JDK 1.8
Dirección IP	Clase C

TABLA 8

Descripción	Carácteristicas PC 6 y PC 8
Sistema	Lenovo
RAM	16 GB
Disco HDD	1,8 T
Procesador	Intel(R)core (TM) i7-7700
Arquitectura	64 bits
Sistema Operativo	Centos 7
Hadoop	2.9
Java	version JDK 1.8
Dirección IP	Clase C

TABLA 9

Descripción	Carácteristicas PC 7 y PC 9
Sistema	HP
RAM	30 GB
Disco HDD	1,8 T
Procesador	Intel(R)Xeon(R) W-2123
Arquitectura	64 bits
Sistema Operativo	Centos 7
Hadoop	2.9
Java	version JDK 1.8
Dirección IP	Clase C

TABLA 10

Tiempo de rendimiento en nodos heterogéneos

Nodos-PC	Tiempo de rendimiento
1	5 minutos y 33 segundos.
3	4 minutos y 31 segundos.
5	2 minutos y 44 segundos.
7	2 minutos y 44 segundos.
10	1 minutos y 31 segundos.

3 RESULTADOS

Se logró implementar el almacenamiento y procesamiento distribuido de datos no estructurados sobre Hadoop Distributed File System (HDFS) y ejecutar el proceso paralelo con el modelo de programación MapReduce y administrar con YARN, sobre los cluter de características homogéneas y heterogéneas. Las prestaciones de servidores con precios exorbitantes, ahora esta al alcance de todos, poqrue Hadoop es open-source, con hardware convencional de PC de precios básicos.

4 REFERENCIAS

[1] K. Rattanaopas y S. Kaewkeeree(2017). Mejora del rendimiento de Hadoop MapReduce con compresión de datos: un estudio con Wordcount Job, Thailandia: IEEE.

[2] J. Bhimani y M. Leeser(2017). Aceleración de aplicaciones de big data utilizando un marco de virtualización ligero en la nube empresarial, Waltham,USA: IEEE.

[3] A. Shah y M. Padole(2018). Equilibrio de carga a través de la política de reorganización de bloques para el clúster heterogéneo de Hadoop, India: IEEE.

[4] Revista 2015 Universidad Politécnica de Madrid. Augsburger Becerra.

[5] M. A. (2014) PARALELIZACIÓN DE UN ALGORITMO PARA LA DETECCIÓN DE CÚMULOS Universidad de Chile Facultad de Ciencias Físicas y Matemáticas Departamento de Ciencias de la Computación.

[6] C. Verma (2016). Big Data representation for Grade Analysis Through, 6th International Conference - Cloud System and Big Data Engineering (Confluence)

[7] R. K. Sidhu(2016). Efficient Batch Processing of Related Big Tasks using Persistent MapReduce, India: ACM.

[8] S. Prabhu (2015). Performance Enhancement of Hadoop MapReduce Framework for Analyzing BigData, India: IEEE.

[9] M. R. Ghazi(2015), Hadoop, MapReduce and HDFS: A Developers Perspective, India: ELSEVIER.

[10] Jean-Pierre (2013). ORACLE: BIG DATA FOR THE ENTERPRISE RED WOOD: ORACLE ENTERPRISE

[11] Sanjay Agrawal (2014). AN EXPERIMENTAL APPROACH TOWARDS BIG DATA FOR ANALYZING MEMORY UTILIZATION ON A HADOOP CLUSTER USING HDFS AND MAPREDUCE. First International Conference on Networks & Soft Computing.

[12] A López Borrull, A Canals (2013). LA COLABORACIÓN CIENTÍFICA EN EL MARCO DE NUEVAS PROPUESTAS CIENTÍFICAS: BIG DATA. Universitat Oberta de Catalunya.

[13] Melanie Swan (2015). PHILOSOPHY OF BIG DATA. First International Conference on Big Data Computing Service and Applications

- **Javier Arturo Rozas Huacho,** Maestro en Ciencias con Mención en Informática. Docente principal a dedicación exclusiva del Departamento Académico de Ingeniería Informática de la UNSAAC. Docente de la Escuela de Pos grado de la UNSAAC

- **Claudio Isaias Huancahuire Bravo,** Jefe de practica en la Universidad Nacional Micaela Bastidas de Apurimac, docente auxiliar de la Universidad Nacional Jose maria Arguedas y actualmente docente de la Universidad Nacional San Antonio Abad del Cusco.

- **Guido Bravo Mendoza,** Ingeniero electricista, Magister en Proyectos de Inversión Universidad Nacional San Antonio Abad del Cusco. Egresado de la Maestría en Administración Educativa. Universidad Nacional Micaela Bastidas de Apurímac.