UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLÁN

"Análisis del Framework MapReduce, como herramienta de manejo de grandes conjuntos de datos".

TESIS

QUE PARA OBTENER EL TÍTULO DE

LICENCIADO EN INFORMÁTICA

PRESENTA

Gerson Giovani Álvarez Regino

Asesor: L.C. Carlos Pineda Muñoz

UNAM
CUAUTITLÁN

CUAUTITLÁN IZCALLI, ESTADO DE MÉXICO 2019

Agradecimientos

A mis padres Regina Regino García y Victor Manuel Álvarez Regino, que son los seres más maravillosos del planeta, por su cariño incondicional y cuidar siempre de mí, por confiar constantemente en mí y apoyarme en mis peores momentos, por formarme como un buen ser humano y guiarme en el camino, por ser mi mayor motivación en la vida y que gracias a todas estas cosas pude concluir mis estudios profesionales.

A mi tía Alicia Moreno Álvarez, por socorrerme en los momentos más complicados y brindarme todo su apoyo de manera incondicional, por su apoyo moral y muestras de afecto, sin los cuales no hubiese acabado mi carrera.

A mi prima Miriam Moreno Álvarez, por ser una mujer admirable, que me brindo su ayuda sin esperar nada a cambio, y por apoyarme a concluir con una de las etapas más importantes en la vida.

Al profesor Carlos Pineda, por servirme como guía, por ser una fuente de inspiración y un increíble mentor, por los consejos y la experiencia que me brindo, por las palabras de aliento que constantemente me brindo, por las innumerables oportunidades que me brindo, por el sin fin de conocimientos que obtuve a su lado y que tendré presentes toda mi vida y por darme un grandioso modelo a seguir.

Índice

Objetivos

- Explicar el concepto y funcionamiento de MapReduce
- Explicar el funcionamiento del Framework Hadoop y el sistema de archivos HDFS

Introducción

Con este trabajo se muestra un punto claro y preciso, que sirva de arranque para los trabajos futuros que realice la comunidad interesada en el campo del manejo de grandes volúmenes de información.

Las tecnologías de la información son cada día más empleadas, ya que no es un campo que tenga descanso en el desarrollo de nuevas herramientas para cubrir las exigencias del mundo en el que vivimos; para enfrentar estos retos, se deben maximizar los recursos que se tienen y proponer alternativas de uso libre para que muchas más personas sumen sus esfuerzos para el mejoramiento de estas tecnologías en el beneficio de todos.

Los profesionistas de este ámbito deben estar preparados para afrontar las nuevas problemáticas, y deberán hacerlo entendiendo que estos nuevos paradigmas no serán solucionados de la manera tradicional. Las nuevas tecnologías son y serán aún más cambiantes en el porvenir, así que debemos entenderlas y usarlas para mejorar y proponer soluciones apropiadas en un futuro.

Antecedentes

Olmedo Yolanda presentó como artículo de divulgación ¿Qué es MapReduce? Para el Blog de Sold Q – Big Data[1].

"MapReduce es un Framework que proporciona un sistema de procesamiento de datos paralelo y distribuirlo. Está pensado para la solución práctica de algunos problemas que pueden ser paralelizados, pero se ha de tener en cuenta que no todos los problemas pueden resolverse eficientemente con Map Reduce". MapReduce está orientado a resolver

[1]http://blogs.solidq.com/es/big-data/que-es-mapreduce/

problemas con conjuntos de datos de gran tamaño (PETABYTES) y utiliza el sistema de archivos HDFS (Hadoop Distributed File System).

El Framework está orientado a resolver los nuevos retos de BigData, una tecnología y concepto que hoy en día es imprescindible conocer para los profesionistas e investigadores, ya que respecto al comportamiento de los últimos quince años de acuerdo con un estudio publicado en Science (Marca registrada) en el año 2011 se pretendía cuantificar la cantidad de información generada y almacenada en el mundo. Ese año el CEO de Google en ese momento, Eric Schmidt, afirmó que la Humanidad había creado hasta 2003 una cantidad equivalente a 5 Exabytes, añadiendo que ahora esta cifra se generaba en 2 días.

Las cifras que ofrece el estudio de Science[2] son realmente abrumadoras. Entre algunas de ellas destacan la cantidad de información generada por la humanidad hasta el año 2007 que la estiman en 295 exabytes, aumentando en 2011 a 600 exabytes, o lo que es lo mismo un trillón de bytes, que es la capacidad que pueden contener un millón de ordenadores de sobremesa actuales. El estudio también nos dice que, la tecnología digital domina claramente sobre la analógicas puesto que desde el 2007, el 99,9% de la información generada era en formato digital, que sólo el 0,007% de la información del planeta está en papel. Esto puede confirmar que va a ir creciendo en relación con el flujo de información que generamos día con día. Teniendo en cuenta este tipo de trabajos, se debe aumentar todo tipo de investigaciones en el tema; para los estudiantes de profesiones relacionadas con las ciencias de la información y manejo de tecnología, facilita la definición de conceptos que no son revisados en el actual plano del día con día.

[2] http://science.sciencemag.org/content/332/6025/60

Justificación

Se debe tener en cuenta, que este tipo de tecnologías al ser relativamente recientes y de tan alto grado de especialización alrededor del ámbito de las TIC's, deben ser expuestas y difundidas al público no tan experto y a todos aquellos interesados en el tema.

Los nuevos conceptos, así como todas las nuevas propuestas llevadas a cabo en el actual escrito, aportarán una mayor visión y mejor entendimiento de las nuevas tecnologías.

De acuerdo con el vicerrector de Investigación y Estudios de Posgrado de la BUAP, Ygnacio Martínez Laguna[3], "México presenta un importante rezago tecnológico y carece de innovaciones", no se tiene una cultura de modernización ni adopción de las nuevas tecnologías e investigaciones de éste tipo que ayuden a realizar un mejor aprovechamiento de la tecnología actual y de todo lo que va surgiendo con estas; lo que ayudaría a resolver la enorme problemática del transporte y distribución de los datos.

Uno de los esfuerzos en la licenciatura en Informática, es promover el uso de estas tecnologías, ya que en el plan de estudios está contemplado la impartición de los seminarios en forma de asignaturas optativas que abordan temas de nuevas tecnologías.

[3]http://www.e-consulta.com/nota/2017-11-06/universidades/padece-mexico-rezago-tecnologico-y-de-innovaciones-senalan-en-buap

Capítulo 1

MapReduce para manejo de grandes volúmenes de datos.

1.1 Introducción

El termino BigData se empieza a popularizar en el año 2012, pero no fue ahí en donde podemos decir que empieza la era de grandes volúmenes de datos, fue en el año de 2004 en donde dos ingenieros de Google, Jeffrey Dean y Sanjay Ghemawat, publican un artículo titulado:

"MapReduce: Simplified Data Processing on Large Clusters "[4].

Hablan de un nuevo modelo de programación que permite simplificar el procesamiento de grandes volúmenes de datos. Lo bautizan como MapReduce. Básicamente, es la evolución natural y necesaria que tenían dentro de Google[5] para procesar los grandes volúmenes de datos que ya por aquel entonces manejaban (documentos, referencias web, páginas, etc.). Lo necesitaban, porque a partir de toda esa información, obtenían una serie de métricas que luego les ayudó a popularizar industrias como el SEO (Search Engine Optimization) y SEM (Seach Engine Marketing), principales fuentes de ingreso de lo que hoy vive Google (Alphabet) y lo que le ha permitido ser la empresa de mayor valor bursátil del mundo de acuerdo con la tabla publicada por BrandZ en el años 2018.

Vease figura 1.0

[4] https://static.googleusercontent.com/media/research.google.com/es//archive/mapreduce-osdi04.pdf

[5] Google Marca Registrada

BrandZ™ Top 100 Most

	Brand	Category	Brand Value 2018 $Mil.	Brand Contribution	Brand Value % Change 2018 vs. 2017	Rank Change	Country of Origin
1	Google	Technology	302,063	4	+23%	0	
2		Technology	300,595	4	+28%	0	
3	amazon	Retail	207,594	4	+49%	1	
4	Microsoft	Technology	200,987	4	+40%	-1	
5	Tencent 腾讯	Technology	178,990	5	+65%	3	
6	facebook	Technology	162,106	4	+25%	-1	
7	VISA	Payments	145,611	5	+31%	0	
8	McDonald's	Fast Food	126,044	4	+29%	2	
9	Alibaba Group 阿里巴巴集团	Retail	113,401	3	+92%	5	
10	AT&T	Telecom Providers	106,698	3	-7%	-4	

Figura 1.0, Tomado de:

http://brandz.com/admin/uploads/files/BZ_Global_2018_DL.pdf

La idea que subyace a este nuevo modelo de programación es el siguiente: ante la necesidad de procesar grandes volúmenes de datos, se puede montar un esquema en paralelo de computación que permita así distribuir el trabajo (el procesamiento de datos) entre diferentes máquinas (nodos dentro de una red) para que se pueda reducir el tiempo total de procesamiento. Vease Figura 1.1

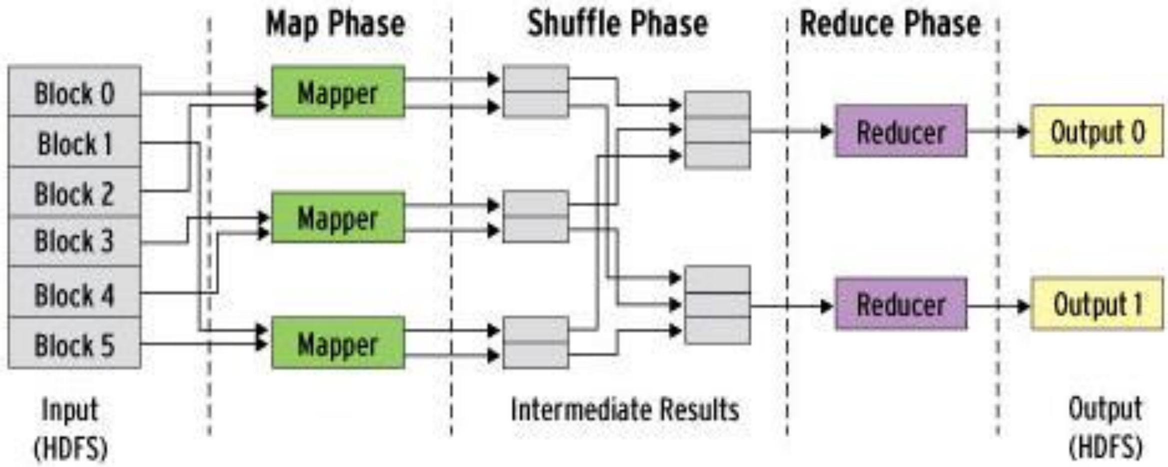


Figura 1.1

Distribución de trabajo a través del modelo MapReduce Tomado de: http://www.admin-magazine.com/HPC/Articles/MapReduce-and-Hadoop

En aquel entonces, estos grandes "visionarios del BigData", se dieron cuenta que este problema que tenía Google en esos momentos, lo iban a tener otras cuantas aplicaciones. Así que decidieron desarrollar un modelo de programación que se desacoplara de las necesidades concretas de Google, y se pudiera generalizar a un conjunto de aplicaciones que pudieran luego reutilizarlo. Pensaron inicialmente en todos los problemas que pudiera tener el propio buscador.

Lo simplificaron tanto que dejaron la preocupación del programador en dos funciones:

- Map
- Reduce

Este paradigma lo adoptó Google en el 2004[6]. Y dado el rendimiento que tenía, se comenzó a emplear en otras aplicaciones.

1.2 Qué es MapReduce

De acuerdo con Yolanda Olmedo en el artículo de divulgación ¿Qué es MapReduce?7 Nos dice que MapReduce es un Framework que proporciona un sistema de procesamiento de datos paralelo y distribuido. Su nombre se debe a las funciones principales que son Map y Reduce.

Este está orientado a resolver problemas con conjuntos de datos de gran tamaño, por lo que utiliza el sistema de archivos distribuido HDFS (Hadoop Distributed File System).

Entonces podemos decir que la Map Reduce es un Framework que reduce el tiempo de trabajo de una gran tarea única a varias tareas distribuidas y simplifica a solo dos pasos lo que conocemos como Map y Reduce.

El Framework MapReduce tiene una arquitectura, maestro / esclavo. Cuenta con un servidor maestro o JobTracker y varios servidores esclavos o TaskTrackers, uno por cada nodo del clúster.

El JobTracker, es el punto de interacción entre los usuarios y el Framework MapReduce. Los usuarios envían trabajos MapReduce al JobTracker, que los

---

[6] https://unpocodejava.wordpress.com/2013/05/22/
[7] http://blogs.solidq.com/es/big-data/que-es-mapreduce/

pone en una cola de trabajos pendientes y los ejecuta en el orden de llegada. El JobTracker gestiona la asignación de tareas y delega las tareas a los TaskTrackers. Los TaskTrackers ejecutan tareas bajo la orden del JobTracker y también manejan el movimiento de datos entre la fase Map y Reduce.

Para ver las diferencias entre JobTracker y TaskTracker vamos a ver las características de cada uno.

Según Google Inc. En donde nos explica cómo funcionan cada uno y en el artículo Explicando MapReduce por Luis Miguel García[8] las siguientes son las características. Vease Tabla 1.0

JobTracker
Capacidad para manejar metadatos de trabajos
Estado de la petición del trabajo
Estado de las tareas que se ejecutan en TaskTracker
Decide sobre la programación
Hay exactamente un JobTracker por clúster

[8]https://unpocodejava.com/2013/05/22/explicando-mapreduce/

Recibe peticiones de tareas enviadas por el cliente
Programa y monitoriza los trabajos MapReduce con TaskTrackers

TaskTracker

- Ejecuta las solicitudes de trabajo de JobTrackers
- Obtiene el código que se ejecutará
- Aplica la configuración específica del trabajo
- Comunicación con el JobTracker:

Envíos de la salida, finalizar tareas, actualización de tareas, etc.

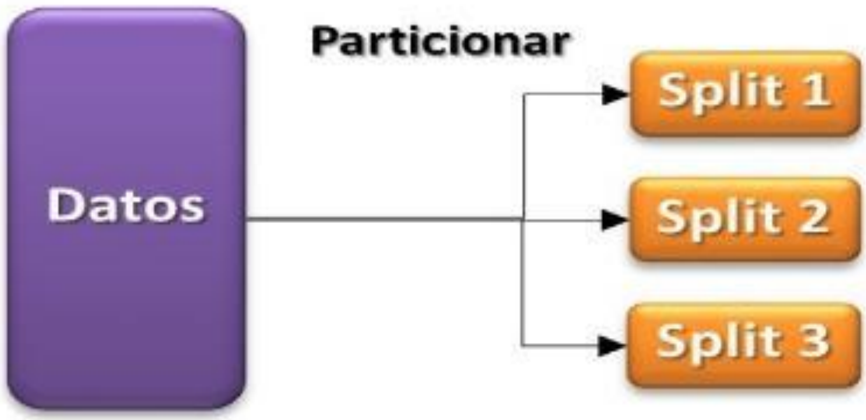
1.3 MapReduce como solución al reto de manejo de grandes volúmenes de datos.

El objetivo principal de MapReduce es, permitir la computación paralela sobre grandes colecciones de datos permitiendo abstraerse de los grandes problemas de la computación distribuida.

MapReduce consta de 2 fases: Map y Reduce. Las funciones Map y Reduce se aplican sobre pares de datos (clave, valor).

Cómo funciona

Lo primero que se hace es tomar los datos de entrada y particionarlos en piezas llamadas Splits.

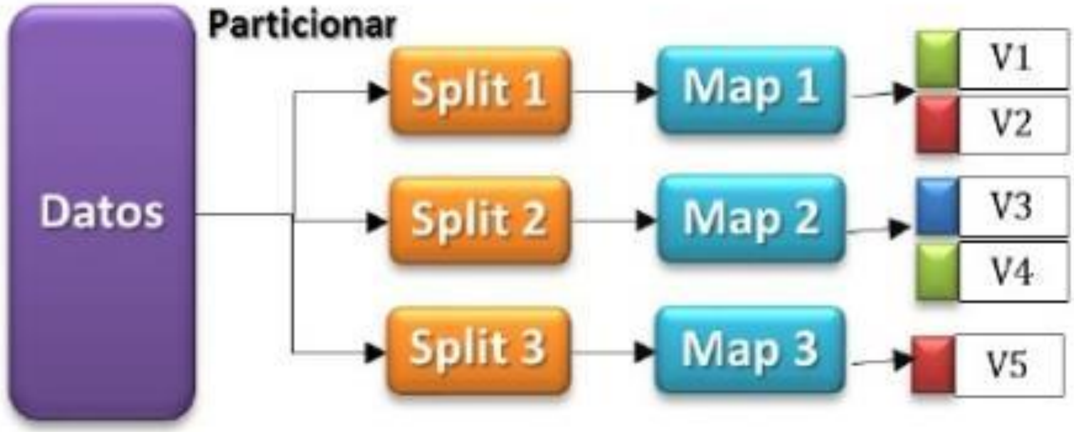


Introducción a MapReduce desde el punto de vista del programador – Parte 1 (Fuente: http://smartbasegroup.com/introduccion-mapreduce/

Proceso de Map

El proceso Map (en realidad son varios procesos Map corriendo exactamente en el mismo código en paralelo en varias máquinas en el clúster) recibe una porción de la data de entrada llamada Split.

En este proceso se produce como salida una lista de pares (clave1, valor1).



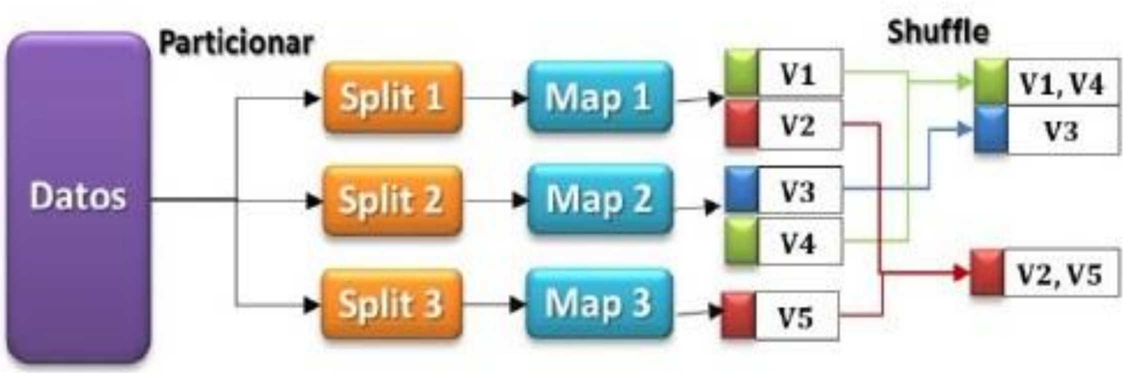
Introducción a MapReduce desde el punto de vista del programador – Parte 1 (Fuente: http://smartbasegroup.com/introduccion-mapreduce/)

En la imagen anterior vemos cómo los datos de entrada son divididos en tres Splits, cada uno de los cuales va a un proceso Map independiente y que corre en paralelo con los otros, y que además cada proceso Map produce su propia salida que depende de los datos de entrada.

Aquí vemos que el proceso Map1 produce dos pares de clave-valor, el Map2 otros dos pares y el Map3 solo un par.

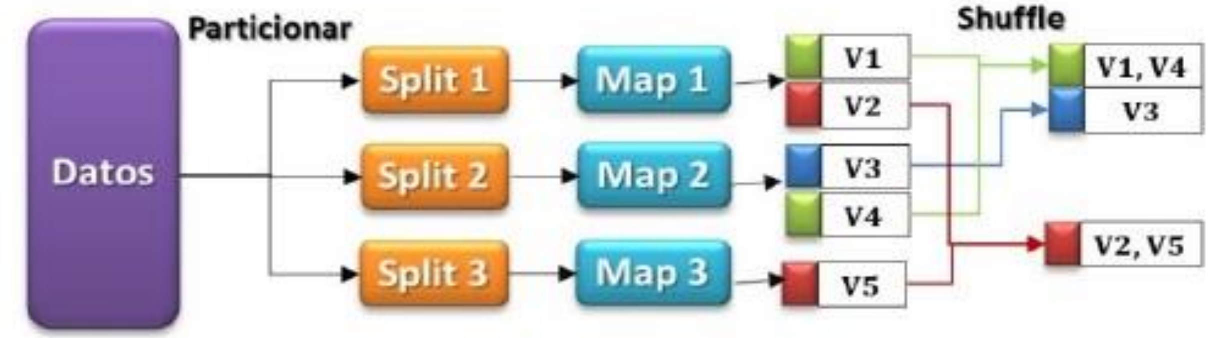
Una vez que el proceso Map ha terminado, comienza el proceso de Shuffle.

Este se encarga de tomar TODOS los pares (clave1, valor1) resultantes de todos los procesos Map ejecutados en paralelo, y agruparlos por clave, es decir, el formato seria (clave1, lista(valor1, valor2,…)), con lo cual todos los



– Parte 1 (Fuente: http://smartbasegroup.com/introduccion-mapreduce/)

Proceso Reduce



Introducción a MapReduce desde el punto de vista del programador – Parte 1 (Fuente: http://smartbasegroup.com/introduccion-mapreduce/)

Los procesos de Reduce (no es solo un proceso Reduce sino varios corriendo exactamente en el mismo código en paralelo en varias máquinas del cluster) reciben los pares (clave1, lista (valor1.1, valor1.2, …) y producen una nueva salida de pares (clave2, valor2).

Es decir, para todos los valores asociados a la clave1, se aplica alguna función que procesa esos valores y se produce una nueva salida (clave2, valor2). Luego todas las salidas de los Reduce son unificadas en una sola.

Nota: en el capítulo 4 se hace la demostración de este funcionamiento.

Capítulo 2

Hadoop y el sistema de archivos HDFS

"Apache Hadoop es un proyecto de software abierto que permite el procesamiento distribuido de grandes conjuntos de datos en clusters de servidores básicos.[9]"

2.1 Introducción a los sistemas de archivos

"Un sistema de archivos son los métodos y estructuras de datos que un sistema operativo utiliza para seguir la pista de los archivos de un disco o partición; es decir, es la manera en la que se organizan los archivos en el disco"[10].

El término también es utilizado para referirse a una partición o disco que se está utilizando para almacenamiento, o el tipo del sistema de archivos que utiliza. Así uno puede decir "tengo diferentes sistemas de archivo" refiriéndose a que tiene dos particiones en las que almacenar archivos, y refiriéndose al tipo del sistema de archivos.

La diferencia entre un disco o partición y el sistema de archivos que contiene es importante. Unos pocos programas trabajan directamente en los sectores crudos del disco o partición; si hay un archivo de sistema existente allí será destruido o corrompido severamente. La mayoría de los programas trabajan sobre un sistema de archivos, y por lo tanto no utilizarán una partición que no contenga uno o que contenga uno del tipo equivocado.

---

[9]https://hadoop.apache.org/

[10] http://www.tldp.org/pub/Linux/docs/ldp-archived/system-admin-guide/translations/es/html/ch06s08.html

Antes de que una partición o disco sea utilizada como un sistema de archivos, necesita ser iniciada, y las estructuras de datos necesitan escribirse al disco. Este proceso se denomina construir un sistema de archivos.

La mayoría de los sistemas de archivos UNIX tienen una estructura general parecida, aunque los detalles exactos pueden variar un poco. Los conceptos centrales son superbloque, nodo-i, bloque de datos, bloque de directorio, y bloque de indirección. El superbloque tiene información del sistema de archivos en conjunto, como su tamaño (la información precisa aquí depende del sistema de archivos). Un nodo-i tiene toda la información de un archivo, salvo su nombre. El nombre se almacena en el directorio, junto con el número de nodo-i. Una entrada de directorio consiste en un nombre de archivo y el número de nodo-i que representa al archivo. El nodo-i contiene los números de varios bloques de datos, que se utilizan para almacenar los datos en el archivo. Sólo hay espacio para unos pocos números de bloques de datos en el nodo-i; en cualquier caso, si se necesitan más espacio para punteros a los bloques de datos son colocados de forma dinámica. Estos bloques colocados dinámicamente son bloques indirectos; el nombre indica que, para encontrar el bloque de datos, primero hay que encontrar su número en un bloque indirecto.

Descripción general de los sistemas de archivos comunes

A continuación, una breve descripción de algunos de los sistemas de archivos más comunes que encontrarás. No es exhaustivo – hay muchos otros diferentes.

FAT32: FAT32 es el sistema de archivos de Windows más antiguo, pero todavía se utiliza en dispositivos de medios extraíbles, solo en los dispositivos más pequeños. Los discos duros externos más grandes de 1 TB o de menos capacidad, probablemente vendrán formateados con NTFS. Sólo desearas utilizar FAT32 con dispositivos de almacenamiento pequeños o para compatibilidad con otros dispositivos como cámaras digitales, consolas de juegos, decodificadores y otros dispositivos que sólo admiten FAT32 y no el nuevo sistema de archivos NTFS.

NTFS: Las versiones modernas de Windows – desde Windows XP – utilizan el sistema de archivos NTFS para su partición del sistema. Las unidades externas se pueden formatear con FAT32 o NTFS.

HFS+: Los usuarios de Mac usan HFS+ para sus particiones internas, y también formatean unidades externas con HFS+. Mac también puede leer y escribir en los sistemas de archivos FAT32, aunque sólo pueden leer de los sistemas de archivos NTFS de forma predeterminada; necesitarían software de terceros para escribir en sistemas de archivos NTFS desde una Mac.

Ext2 / Ext3 / Ext4: A menudo verás los sistemas de archivos Ext2, Ext3 y Ext4 en Linux. Ext2 es un sistema de archivos más antiguo, y carece de características importantes como el registro en diario (si se apaga el equipo o se bloquea una computadora al escribir en una unidad ext2, es posible que se pierdan datos.) Ext3 añade estas características de robustez a costa de cierta velocidad. Ext4 es más moderno y rápido (ahora es el sistema de archivos predeterminado en la mayoría de las distribuciones de Linux , y es más rápido.) Windows y Mac no son compatibles con estos sistemas de archivos; necesitará una herramienta de terceros para acceder a los archivos de dichos sistemas de archivos. Por esta razón, a menudo es ideal para formatear las particiones del sistema Linux como ext4 y dejar dispositivos extraíbles formateados con FAT32 o NTFS si se necesita compatibilidad con otros sistemas operativos. Linux puede leer y escribir en FAT32 o NTFS.

Btrfs: Es un sistema de archivos de Linux que aún está en desarrollo. No es el predeterminado en la mayoría de las distribuciones de Linux en este momento, pero probablemente reemplazará a Ext4 un día. El objetivo es proporcionar características adicionales que permitan a Linux escalar a mayores cantidades de almacenamiento.

Swap: En Linux, el sistema de archivos "swap" no es realmente un sistema de archivos. Una partición formateada como "swap" sólo puede ser utilizada como espacio de intercambio por el sistema operativo (es como el archivo de página en Windows, pero requiere una partición dedicada.)

2.2 Framework Hadoop y sistema de Archivos HDFS

Está diseñado para extender un sistema de servidor único a miles de máquinas, con un muy alto grado de tolerancia a las fallas. En lugar de depender del hardware de alta gama, la fortaleza de este cluster se debe a la capacidad que tiene el software para detectar y manejar fallas al nivel de las aplicaciones.

Hadoop puede manejar diferentes tipos de datos y cargas de trabajo específicos, es sin lugar a duda, una tecnología que sirve para dividir grandes cantidades de datos sin procesar (audio, imágenes, videos), tanto estructurados, no estructurados y semi-estructurados, que se necesiten estudiar y preparar para su análisis.

Hadoop es un sistema distribuido usando una arquitectura Master-Slave, para almacenar su Hadoop Distributed File System (HDFS) y algoritmos de MapReduce para hacer cálculos clave valor.

¿Qué es Hadoop?

"Básicamente, permite desarrollar tareas muy intensivas de computación masiva, dividiéndolas en pequeñas piezas y distribuyéndolas en un conjunto todo lo grande que se quiera de máquinas: análisis de Petabytes de datos, en entornos distribuidos formados por muchas máquinas sencillas[11]"

Hadoop MapReduce[12], Es un sistema basado en hebras (preproceso que se explica abajo) para el procesamiento paralelo de grandes conjuntos de datos; es un marco de software creado con el fin de hacer aplicaciones que puedan procesar grandes cantidades de datos de forma paralela, en un mismo hardware.

Cuando los datos entran para ser procesados se dividen de manera independiente, para su procesamiento, es decir, de manera distribuida en diferente hardware que exista.

Sistema de Archivos HDFS

Sistema de archivos distribuido Hadoop (HDFS): [13] Sistema de archivos tolerante a fallos, escalable y con arquitectura distribuida que puede llegar almacenar 100 TB en un solo archivo. (Brinda la apariencia de estar trabajando en un solo archivo, pero lo que realmente se tiene son archivos distribuidos en varias máquinas para su procesamiento).

"Los datos en un cluster Hadoop se dividen en partes más pequeñas llamadas bloques y, a continuación, se distribuyen en todo el cluster. Los bloques y copias de bloques se almacenan en otros servidores en el cluster Hadoop. Es decir, un archivo individual se almacena como bloques más pequeños que se replican entre varios servidores en el cluster.

Cada HDFS cluster tiene un número de DataNodes habiendo un DataNode para cada nodo en el clúster. Los DataNodes gestionan el almacenamiento que se adjunta a los nodos en los que se ejecutan. Cuando se divide un

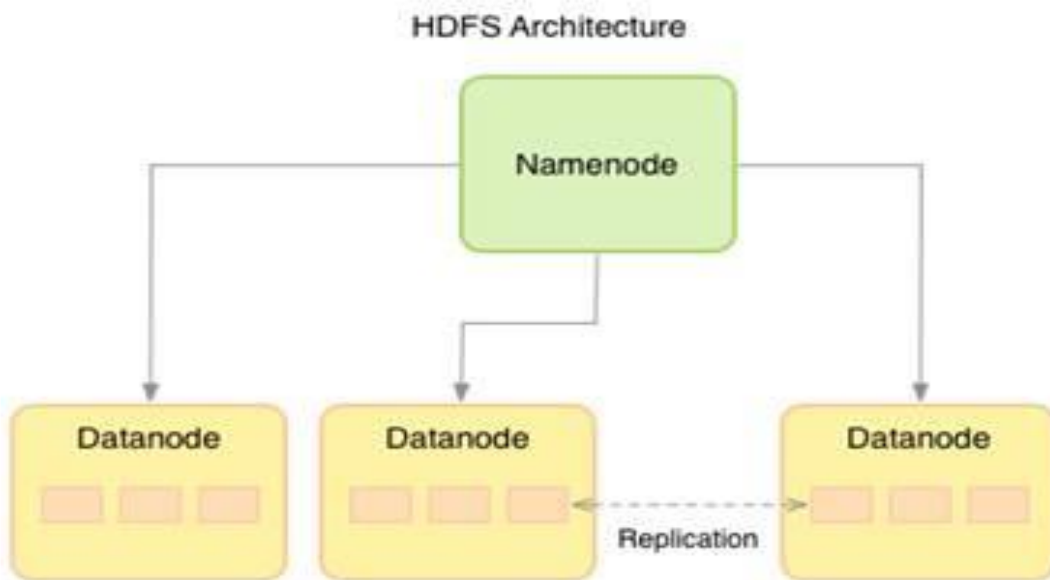
---

[11] Stephen J. Bigelow "http://searchdatacenter.techtarget.com/es/respuesta/Se-impulsara-big-data-con-Hadoop-en-la-nube ".

[12]

[13] https://hadoop.apache.org/docs/r1.2.1/hdfs_user_guide.html

archivo en bloques, éstos se almacenan en un conjunto de DataNodes que se distribuyen en todo el cluster. Los DataNodes son responsables de servir las solicitudes de lectura y escritura de los clientes en el sistema de archivos, y también gestionan la creación, supresión y replicación de bloques.[14]"



NameNodes

El NameNode es el hardware básico que contiene el sistema operativo GNU/Linux y el software NameNode. Es un software que puede ejecutarse en hardware. El sistema que tiene la NameNode actúa como el servidor maestro y no las siguientes tareas:

- Administra el espacio de nombres del sistema de archivos.
- Del cliente regula el acceso a los archivos.
- Además, ejecuta las operaciones del sistema de archivos como el cambio de nombre, cierre y apertura de archivos y directorios.

[14]https://www.ibm.com/support/knowledgecenter/es/SSPT3X_4.1.0/com.ibm.swg.im.infosphere.biginsights.product.doc/doc/c0057606.html

Datanode

La datanode es un hardware de productos básicos con el sistema operativo GNU/Linux y software datanode. Para cada nodo (Commodity hardware/Sistema) de un clúster, habrá un datanode. Estos nodos gestionar el almacenamiento de datos de su sistema.

- Datanodes realizar operaciones de lectura y escritura de los sistemas de archivos, como por petición del cliente.
- Además, permiten realizar operaciones tales como creación, supresión, con lo que la replicación de acuerdo con las instrucciones del namenode.

Bloque

En general los datos de usuario se almacenan en los archivos de HDFS. El archivo en un sistema de archivos se divide en uno o más segmentos y/o almacenados en los nodos de datos. Estos segmentos se denominan como bloques. En otras palabras, la cantidad mínima de datos que HDFS puede leer o escribir se llama un bloque. El tamaño de bloque por defecto es de 64 MB, pero puede ser aumentado por la necesidad de cambiar de configuración HDFS.

2.3 Tendencia de tecnologías YARN

"YARN (Yet Another Resource Negotiator) is a cluster management system. It has been part of Apache Hadoop since v2.0. With the help of YARN arbitrary applications can be executed on a Hadoop cluster. Therefore, the application has to consist of one application master and an arbitrary number of containers. Latter are responsible for the execution of the application whereas the application master requests container and monitors their progress and status.[15]"

Apache Hadoop YARN (por las siglas en inglés de "otro negociador de recursos") es una tecnología de administración de clústeres.

YARN es una de las características clave de la segunda generación de la versión Hadoop 2 del marco de procesamiento distribuido de código abierto de Apache Software Foundation. Originalmente descrito por Apache como un gestor de recursos rediseñado, YARN se caracteriza ahora como un sistema operativo distribuido, a gran escala, para aplicaciones de Big Data.

YARN es una reescritura de software que desacopla las capacidades de gestión de recursos y planificación de MapReduce del componente de procesamiento de datos, permitiendo a Hadoop soportar enfoques más variados de procesamiento, y una gama más amplia de aplicaciones. Por ejemplo, los clúster Hadoop ahora pueden ejecutar consultas interactivas y transmisiones de aplicaciones de datos de forma simultánea con los trabajos de MapReduce. La encarnación original de Hadoop empareja de cerca al sistema de archivos distribuidos Hadoop (HDFS) con el marco de programación MapReduce orientado a lotes, que se ocupa de la gestión

[15] https://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html

de recursos y la planificación de tareas en los sistemas Hadoop, y soporta el análisis y la condensación de conjuntos de datos en paralelo.

YARN combina un administrador central de recursos que reconcilia la forma en que las aplicaciones utilizan los recursos del sistema de Hadoop con los agentes de administración de nodo que monitorean las operaciones de procesamiento de nodos individuales del clúster. Ejecutándose en clústeres de hardware básicos, Hadoop ha atraído un interés particular como zona de espera y de almacenamiento de datos para grandes volúmenes de datos estructurados y no estructurados destinados al uso en aplicaciones de analítica. Separar HDFS de MapReduce con YARN hace al ambiente Hadoop más adecuado para las aplicaciones operativas que no pueden esperar para que terminen los trabajos.

Capítulo 3

Lenguajes de programación para MapReduce y otras alternativas para el manejo de grandes volúmenes de datos

3.1 Introducción a los lenguajes de programación

Un lenguaje de programación es un convenio entre personas que puede definirse así:

Conjunto de reglas o normas que permiten asociar a cada programa correcto un cálculo que será llevado a cabo por un ordenador (sin ambigüedades). Por tanto, un lenguaje de programación es un convenio o acuerdo acerca de cómo se debe de interpretar el significado de los programas de dicho lenguaje, muchas veces se confunden los lenguajes con los compiladores, intérpretes o con los entornos de desarrollo de software

Lenguajes Imperativos

Lenguajes Imperativos
Un cálculo es un conjunto de instrucciones que establecen explícitamente como se debe manipular la información digital presente en memoria, y/o como se debe recoger o enviar información desde/hacia los dispositivos
Ejemplos:

• Ejemplos de lenguajes imperativos: Fortran, Pascal, C, C++, Java.

• La mayoría de los lenguajes usados para desarrollo de software comercial son imperativos.

Lenguajes Funcionales

Un cálculo es el proceso de aplicar una función recursiva a un valor de su dominio para obtener el correspondiente valor del rango (el resultado).

El término función recursiva debe entenderse aquí según se introduce en la teoría de la computabilidad, es decir, como una función calculable con una máquina de turing (no como un subprograma que se invoca a si mismo)

Un programa en estos lenguajes consiste en una especificación de la función recursiva que queremos calcular, junto con los argumentos sobre los que se aplica.

Normalmente, dicha función estará especificada en términos de otras, que también se incluyen en el programa

Ejemplos:

• Ejemplos de lenguajes funcionales son: Lisp, Scheme, ML, Miranda, Haskell

• Menos difundidos que los imperativos para el desarrollo de software comercial

Lenguajes Declarativos

Un cálculo es el proceso de encontrar que elementos de un dominio cumplen determinada relación definida sobre dicho dominio, o bien determinar si un determinado elemento cumple o no dicha relación.

• Un programa en estos lenguajes consiste en una especificación de la relación que queremos calcular

• Normalmente, dicha relación estará especificada en términos de otras, que también se incluyen en el programa

El lenguaje declarativo por excelencia es Prolog

• Ejecutar un programa consiste en buscar recursivamente en una base de datos de relaciones

• Prolog está especialmente indicado para aplicaciones muy específicas como:

– sistemas expertos
– demostración de teoremas
– consulta de bases de datos relacionales,
– procesamiento del lenguaje natural

3.2 Lenguajes de programación que se pueden usar en el manejo de grandes volúmenes de datos no aplicados en la investigación SCALA, Python, R

Scala

“Scala es un lenguaje de programación moderno multiparadigma diseñado para expresar patrones de programación comunes de una forma concisa, elegante, y de tipado seguro. Integra fácilmente características de lenguajes orientados a objetos y funcionales.[16]”

Spark es una aplicación que está programada utilizando el lenguaje de programación Scala. Según la Universidad de Barkley, Scala utiliza mecanismos sencillos para describir procesos concurrentes; lo que hace más eficiente y simple el desarrollo de aplicaciones para el entorno mencionado.

Una de las herramientas de procesamiento de grandes volúmenes de datos tolerante a fallas es “Apache Spark”, que surge como una alternativa a “Apache Hadoop” y su metodología MapReduce.

[16]https://docs.scala-lang.org/es/tutorials/tour/tour-of-scala.html.html

El creador de este lenguaje de programación (SCALA) el Dr. Martín Odersky fue un desarrollador distinguido que realizo el compilador "javac", hoy es una herramienta de gran utilidad en el mundo JAVA[17].

El conoce las ventajas de JAVA así que decidió realizar un lenguaje que aprovechara las fortalezas de JAVA, así nace SCALA, que se ejecuta en una plataforma JAVA utilizando JVM (Java Virtual Machine) y que es 100% compatible con JAVA. En 2011 la compañía Typesafe Inc. Desarrolla módulos y nuevas versiones de SCALA para el cómputo distribuido introduciendo conceptos innovadores en el lenguaje principalmente para el contexto el modelo de actores (Model Actor) en la biblioteca AKKA.

El modelo de actor para el soporte de procesos concurrentes en Scala, tiene características que promueven hoy el uso de Scala, como el soporte en los procesos distribuidos tolerantes a fallas con grandes volúmenes de datos (Big Data).

El modelo de actores son entidades concurrentes muy ligeras, procesan mensajes de forma asíncrona utilizando un ciclo de recepción basado en eventos[18]. La coincidencia de patrones en los mensajes es una forma conveniente de expresar el comportamiento de un actor. Elevan el nivel de abstracción y hacen que sea más sencillo probar, comprender y mantener sistemas concurrentes o distribuidos.

Según el Dr. Gabriel Guerrero "Entornos Big Data Para herramientas como Spark, las ventajas de utilizar Scala sobre Java son abrumadoras (una versión Scala de un programa será usualmente 5-10 veces más corto que el programa Java equivalente)[19]"

[17]http://www.saxsa.com.mx/por-que-scala-para-big-data/

[18] https://yujikiriki.github.io/2014/11/18/futuros_y_actores.html

[19]http://www.saxsa.com.mx/por-que-scala-para-big-data/

-Moshe Kranc (CTO, Ness Digital Engineering) en DZone.

Características de Scala en entornos Big Data

- Apache Spark está escrito en Scala y el Shell es Scala
- La Librería AKKA es muy poderosa y está escrita en Scala que hoy en día es una de las piezas angulares del cómputo distribuido concurrente de aplicaciones.

Python

"Python es un lenguaje de programación interpretado, orientado a objetos de alto nivel y con semántica dinámica. Su sintaxis hace énfasis en la legibilidad del código, lo que facilita su depuración y, por tanto, favorece la productividad. Ofrece la potencia y la flexibilidad de los lenguajes compilado. Aunque Python fue creado como lenguaje de programación de uso general, cuenta con una serie de librerías y entornos de desarrollo para cada una de las fases del proceso de Data Science. Esto, sumado a su potencia, su carácter open source[20]"

Python fue creado por Guido Van Rossum en 1991 y, como curiosidad, debe su nombre a la gran afición de su creador por las películas del grupo Monty Python.

Es administrado por la Python Software Foundation. Posee una licencia de código abierto, denominada Python Software Foundation License, que es compatible con la Licencia pública general de GNU a partir de la versión.

---

[20] LUCA AI Powered Desicions, Paloma Recuero

Big Data es tan amplio y tiene tanta información para trabajar en ella, sacar información estructurada y analizar concretamente los datos existen muchas maneras de hacerlo, algunas más eficientes que otras y por supuesto cada día más completas.

La complejidad del Big Data es siempre alta sea cual sea el profesional que desee aprender, sin embargo, hay herramientas que son un apoyo crucial para los desarrolladores y profesionales especializados, estas permiten simplificar y maximizar procesos, un claro ejemplo de estas es Python.

Python según la comunidad de desarrolladores es un lenguaje más sencillo de entender, profesionales del área lo manejan y esto ahorra mucho tiempo en capacitación y entrenamiento a nuevo talento humano que deseé aprendan este lenguaje.

Python además es compatible con Hadoop, con el paquete PyDoop ofrece acceso para la API HDFS para Hadoop y permite escribir programas y aplicaciones en Hadoop MapReduce. Usando el API HDFS se pueden conectar tus programas para una instalación HDFS, haciendo posible leer, escribir y obtener información de los archivos y directorios, y las propiedades globales del sistema de archivos.

Los paquetes que ofrece el lenguaje tienen un amplio rango para la ciencia de datos y el análisis. Algunos de los paquetes más poderosos y populares son los siguientes.

- Pandas - a Python data analysis library that offers a range of functions for dealing with data structures and operations like manipulating numerical tables and time series.
- Scipy - library for scientific and technical computing. SciPy contains modules for common data science and engineering tasks like linear algebra, interpolation, FFT, signal and image processing, ODE solvers.

Lenguaje de programación *R*

R es un entorno y lenguaje de programación con un enfoque al análisis estadístico. R es una implementación de software libre del lenguaje S pero con soporte de alcance estático.

R es parte del sistema GNU y se distribuye bajo la licencia GNU GPL. Está disponible para los sistemas operativos Windows, Macintosh, Unix y GNU/Linux.

“Fue desarrollado inicialmente por Robert Gentleman y Ross Ihaka del Departamento de Estadística de la Universidad de Auckland en 1993.1 Sin embargo, si se remonta a sus bases iniciales, puede decirse que inició en los Bell Laboratories de AT&T y ahora Alcatel-Lucent en Nueva Jersey con el lenguaje S. Este último, un sistema para el análisis de datos desarrollado por John Chambers, Rick Becker, y colaboradores diferentes desde finales de 1970. La historia desde este punto es prácticamente la del lenguaje S. Los diseñadores iniciales, Gentleman y Ihaka, combinaron las fortalezas de dos lenguajes existentes, S y Scheme. En sus propias palabras: "El lenguaje resultante es muy similar en apariencia a S, pero en el uso de fondo y la semántica es derivado desde Scheme". El resultado se llamó R en parte al reconocimiento de la influencia de S y en parte para hacer gala de sus propios logros”.

-R Project

“R es un lenguaje con una curva de aprendizaje compleja, pero muy robusto y efectivo para el manejo de datos estadísticos. Es un lenguaje orientado a objetos, muy similar a las sintaxis C y C++. Para los desarrolladores especializados en estos lenguajes, puede ser sencillo. Además, R es un lenguaje de programación que está en constante evolución y del que se dispone de una amplia documentación.[21]"

Es de nombrar que R está muy presente en los ejercicios formativos en Big Data y Data Science: al ser un lenguaje gratuito acaba entrando muy fuerte en los círculos académicos. Se asimila esta manera de programar también por los altos costos de las alternativas de pago. Con lo cual, a pesar de ser un lenguaje un tanto complejo, se va adaptando a amplitud de necesidades y demandas.

Siendo un conjunto muy vasto de funciones estadísticas, con una base de programación no estructurada, su presencia se vio muy pronto consolidada en la escritura de programas para el análisis especial y profundo de los datos. Presentando, además, varias características atractivas: funciona con comandos, sirve para analizar datos y generar gráficos de gran calidad (que se pueden visualizar y guardar en varios formatos de manera directa), es integrable en distintas bases de datos (dado que puede correr sobre diferentes hardware y software), tiene versatilidad en el momento de llamar a paquetes de datos y bibliotecas mediante una gama generosa de funcionalidades, muestra resultados estadísticos en pantalla pero permite guardar resultados intermedios y exportarlos, la relativa facilidad para dar respuesta a necesidades gracias al tamaño de la comunidad R en el mundo, etc.

Por otra parte, y esto es muy importante, las funciones de R interactúan con otros programas y con el Big Data. Es decir, en el caso de librerías, permite trabajar en entornos distribuidos.

[21]https://bbvaopen4u.com/es/actualidad/taller-para-novatos-en-r-ventajas-instalacion-y-paquetes

Desde R podemos comunicarnos con otros entornos y trabajar, por ejemplo, en Hadoop y Spark. La ventaja de conectar varias máquinas (servidores de memoria) y hacer que todas trabajen como una sola (en la misma tarea), nos proporciona la capacidad de atacar bases de datos muy grandes mediante particiones. Entre las listas de lenguajes de programación más populares, tenemos la del Instituto de Ingeniería Eléctrica y Electrónica (Institute of Electrical and Electronics Engineers o IEEE), que cuenta con alrededor de 425.000 miembros en 160 países. Esta lista se basa en los lenguajes más usados para desarrollos web, soluciones para empresas, aplicaciones móviles, etc. Su índice estrella para 2018 indica que los lenguajes más usados (entre una lista de 48) son Python, C, Java, C++, C#, R, JavaScript, PHP, Go y Swift[22].

3.3 PIG BIG DATA

"Pig es una plataforma de scripting desarrollado originalmente por Yahoo! en el 2006 y fue adoptado un año después por Apache Software Foundation para convertirse en un subproyecto de Apache.

Pig provee un lenguaje de alto nivel para crear flujos de datos llamado Pig Latin el cual permite realizar programas MapReduce de forma simple y en pocas líneas de código. La gran ventaja es que no es necesario saber programar en Java, pues Pig posee dentro de su infraestructura un compilador capaz de producir secuencias MapRecude, lo que permite a los usuarios Hadoop enfocarse más en el análisis de los datos que el desarrollo mismo de los programas.[23]"

Pig Latin

---

22 IEEE Espectrum https://spectrum.ieee.org/at-work/innovation/the-2018-top-programming-languages

23 Devorando datos con Pig, "http://blog.jacagudelo.com/pig-hadoop/

Pig Latin es un lenguaje de flujos de datos que trabaja en paralelo. Lo que quiere decir que permite a los programadores describir cómo los datos provenientes de una o más entradas deben ser leídos, procesados y luego almacenados a uno o más flujos de salida en paralelo.

La sintaxis de Pig Latin es muy similar a la de SQL, aunque Pig Latin es un lenguaje de transformación de datos y, por lo tanto, es similar a los optimizadores de consultas de base de datos de los sistemas de bases de datos actuales.

Uno de los principales objetivos de Pig es que Pig Latin sea el lenguaje natural para los entornos de procesamientos de datos paralelos como Hadoop, ya que Pig en si ofrece varias ventajas sobre el uso de MapReduce directamente.

La programación en Pig tiene al menos 3 pasos:

- El primer paso en un programa Pig es cargar los datos que desea manipular desde HDFS.
- A continuación, ejecuta los datos a través de un conjunto de transformaciones (que, bajo las cubiertas, se traducen en un conjunto de tareas de asignación y reductor en MapReduce).
- Finalmente, DUMP para ver los datos de resultado en pantalla o STORE para almacenar los resultados en un archivo en alguna parte[24].

[24] Devorando datos con Pig, “http://blog.jacagudelo.com/pig-hadoop/

De acuerdo con M. Jones miebro de IBM Developers Works “Apache Pig modifica esto al crear una abstracción de lenguaje de procedimiento más simple sobre MapReduce para exponer una interfaz más parecida con Structured Query Language (SQL) para aplicaciones Hadoop. Entonces, en vez de escribir una aplicación de MapReduce separada, se puede escribir un único script en Pig Latin que es automáticamente paralelizado y distribuido a través de un clúster[25]”.

[25] https://www.ibm.com/developerworks/ssa/library/l-apachepigdataquery/index.html

Capítulo 4

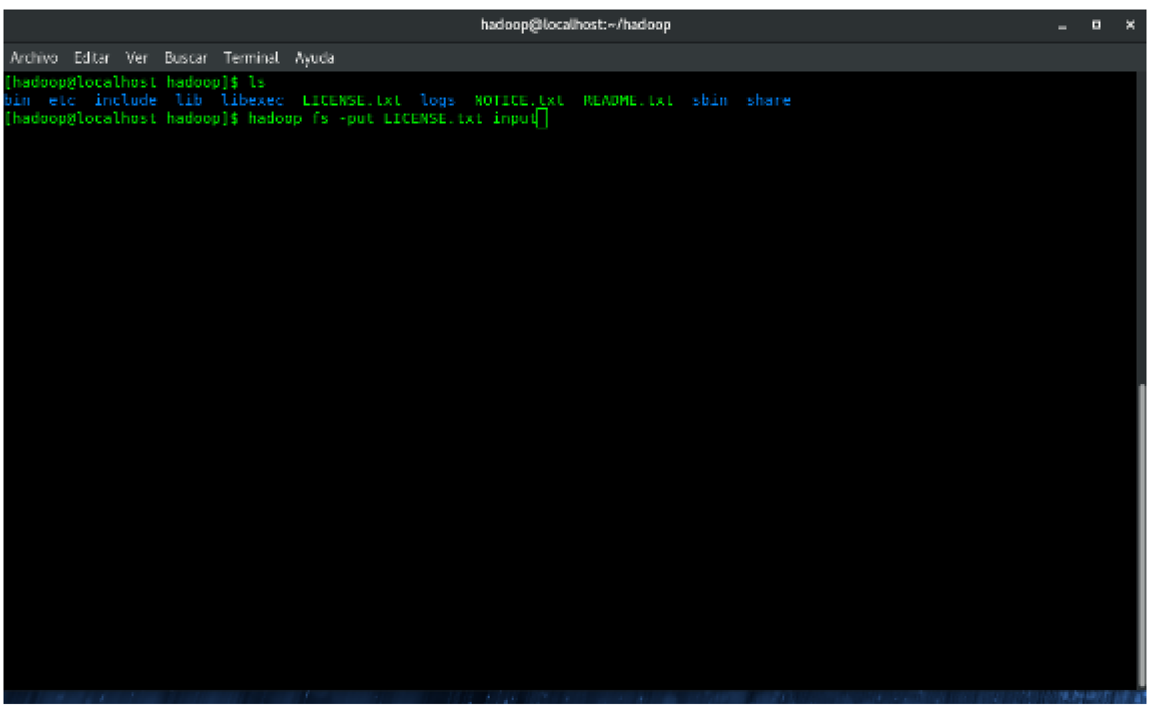
Funcionamiento de MapReduce

4.1 Ejemplos de Funcionamiento MapReduce JAVA

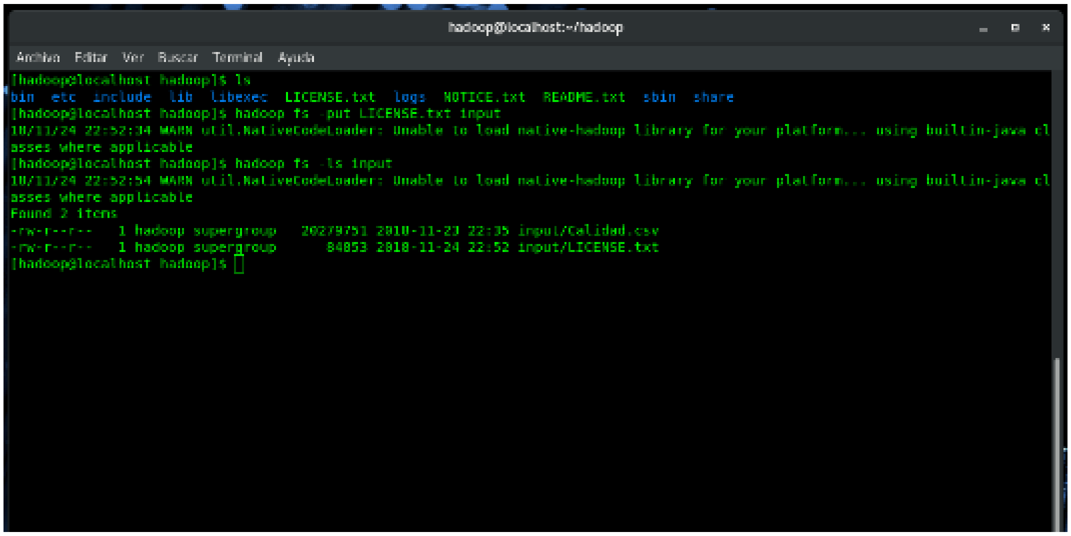
EJEMPLO WORDCOUNT HADOOP MAPREDUCE JAVA

Para comenzar a probar MapReduce ejecutando algún programa y resolviendo un problema, usaremos un ejemplo famoso en el contexto Hadoop y MapReduce el cual es el programa Word Count.

El programa Word Count consiste en contar y ordenar cada palabra de un archivo. Lo primero que se tiene que hacer es subir el archivo al HDFS.

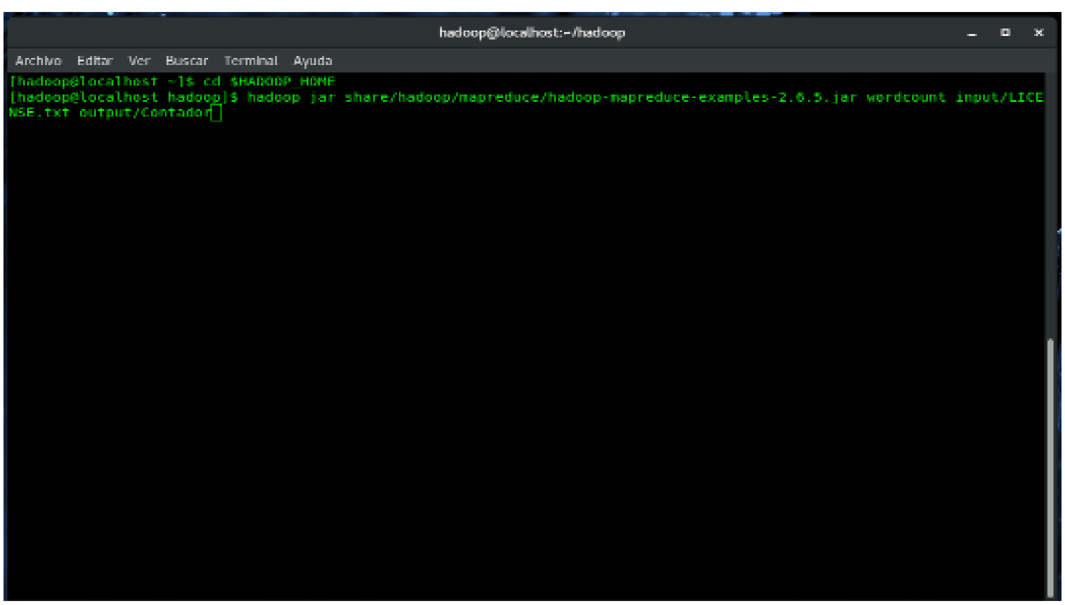


Verificamos que no exista algún error, y que el archivo que subimos se encuentre en la carpeta que indicamos de HDFS la cual se llama “input”

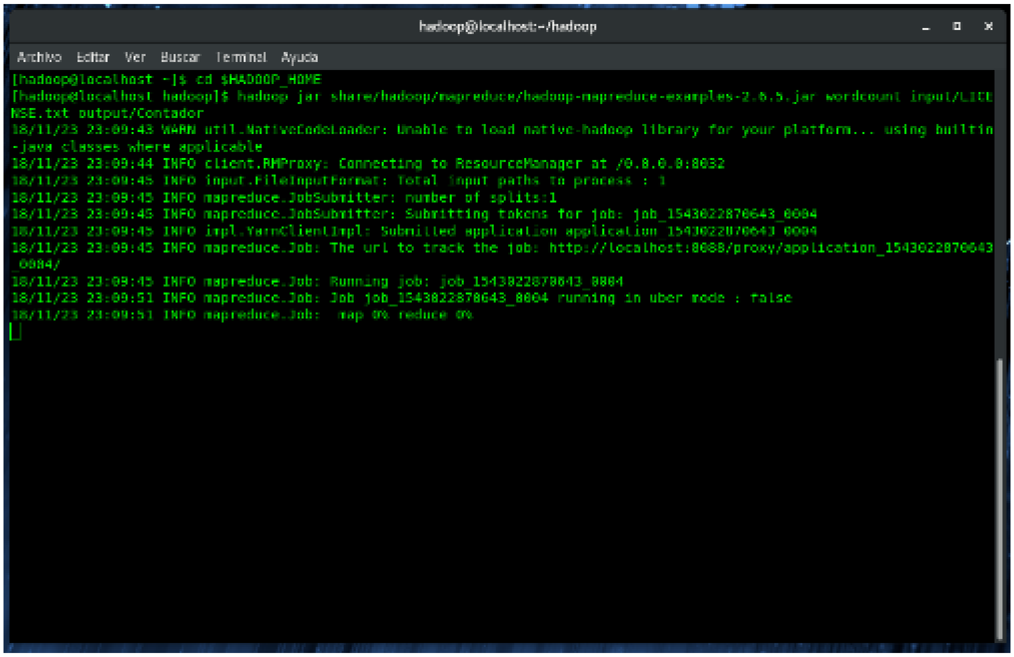


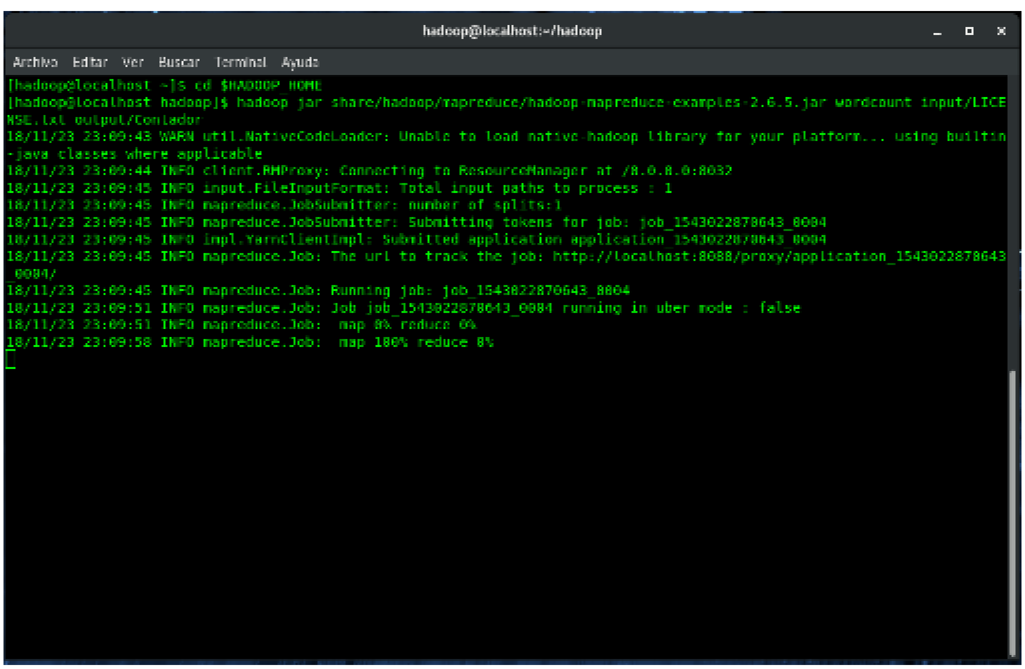
Una vez confirmado el archivo que colocamos en la ruta de HDFS, ejecutamos el programa Word Count pasándole:

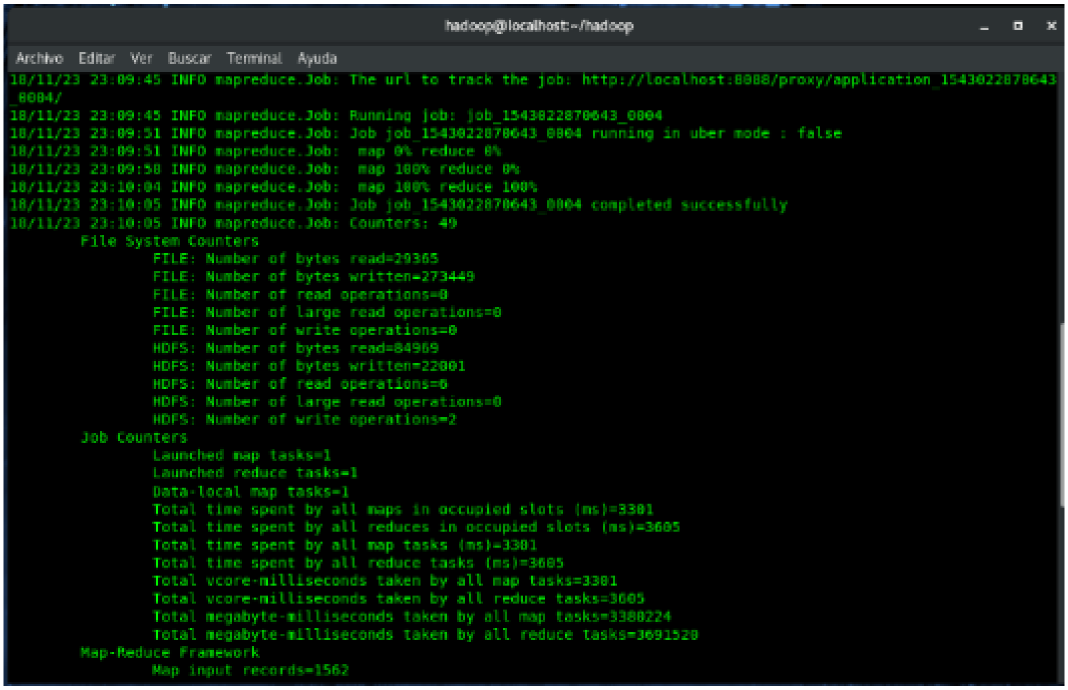
1. La ruta del jar que vamos a ejecutar.
2. La ruta del archivo que va a tomar para que el programa lea.
3. La ruta de salida del archivo HDFS con el resultado del programa Word Count ejecutado después de hacer el proceso Map y Reduce



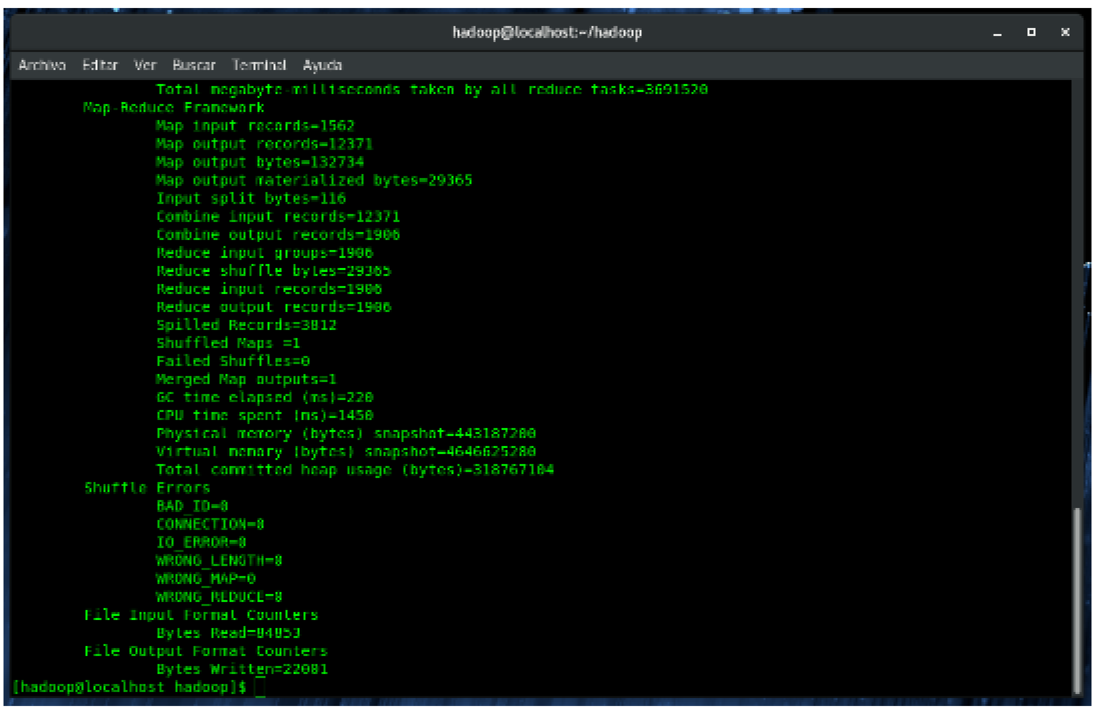
La salida en la terminal después de ejecutar el programa, deberá ser algo parecido a las imágenes que se muestran a continuación.





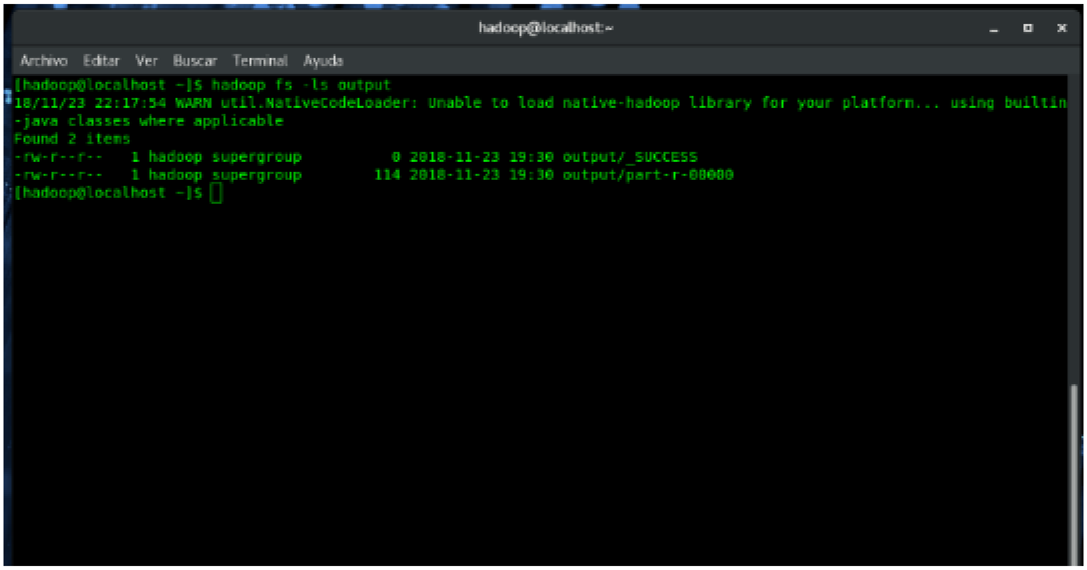


```
hadoop@localhost:~/hadoop
Archivo Editar Ver Buscar Terminal Ayuda
18/11/23 23:09:45 INFO mapreduce.Job: The url to track the job: http://localhost:8088/proxy/application_1543022870643
_0004/
18/11/23 23:09:45 INFO mapreduce.Job: Running job: job_1543022870643_0004
18/11/23 23:09:51 INFO mapreduce.Job: Job job_1543022870643_0004 running in uber mode : false
18/11/23 23:09:51 INFO mapreduce.Job:  map 0% reduce 0%
18/11/23 23:09:58 INFO mapreduce.Job:  map 100% reduce 0%
18/11/23 23:10:04 INFO mapreduce.Job:  map 100% reduce 100%
18/11/23 23:10:05 INFO mapreduce.Job: Job job_1543022870643_0004 completed successfully
18/11/23 23:10:05 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=29365
                FILE: Number of bytes written=273449
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=84969
                HDFS: Number of bytes written=22081
                HDFS: Number of read operations=6
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=1
                Launched reduce tasks=1
                Data-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=3301
                Total time spent by all reduces in occupied slots (ms)=3605
                Total time spent by all map tasks (ms)=3301
                Total time spent by all reduce tasks (ms)=3605
                Total vcore-milliseconds taken by all map tasks=3301
                Total vcore-milliseconds taken by all reduce tasks=3605
                Total megabyte-milliseconds taken by all map tasks=3380224
                Total megabyte-milliseconds taken by all reduce tasks=3691520
        Map-Reduce Framework
                Map input records=1562
```

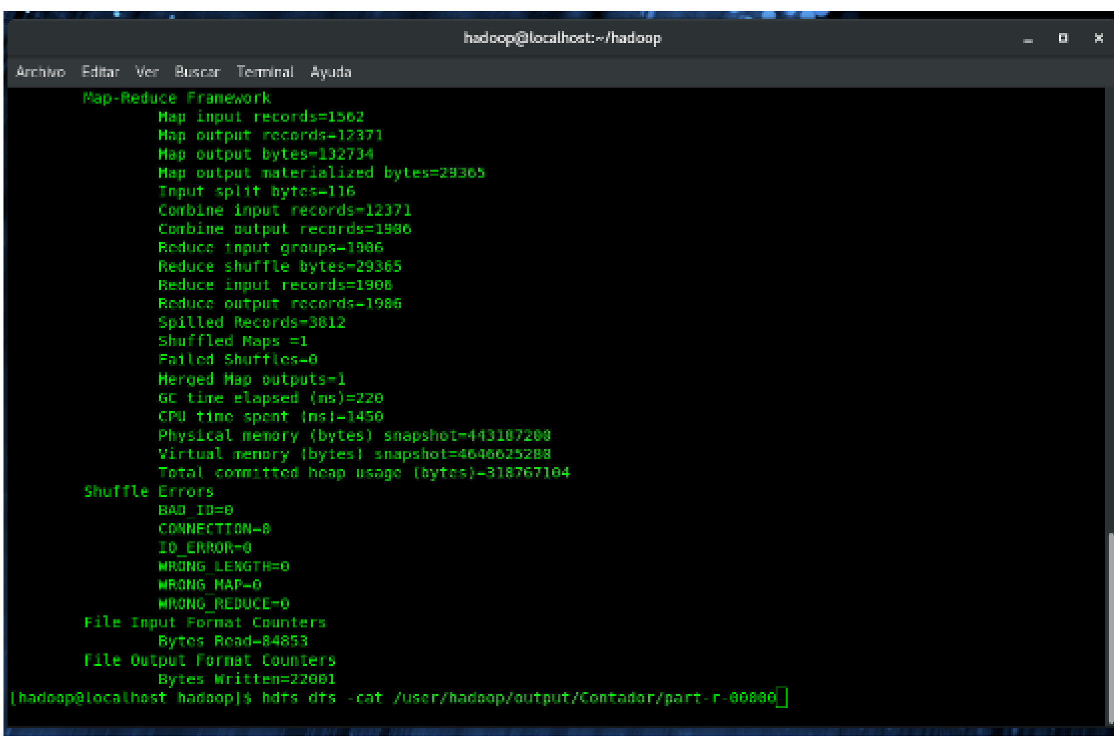


```
hadoop@localhost:~/hadoop
Archivo Editar Ver Buscar Terminal Ayuda
                Total megabyte-milliseconds taken by all reduce tasks=3691520
        Map-Reduce Framework
                Map input records=1562
                Map output records=12371
                Map output bytes=132734
                Map output materialized bytes=29365
                Input split bytes=116
                Combine input records=12371
                Combine output records=1906
                Reduce input groups=1906
                Reduce shuffle bytes=29365
                Reduce input records=1906
                Reduce output records=1906
                Spilled Records=3812
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=220
                CPU time spent (ms)=1450
                Physical memory (bytes) snapshot=443187200
                Virtual memory (bytes) snapshot=4646625280
                Total committed heap usage (bytes)=318767104
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=84853
        File Output Format Counters
                Bytes Written=22081
[hadoop@localhost hadoop]$
```

Una vez finalizado el proceso del comando que ejecuto el programa sin errores, podremos observar el resultado en el directorio de salida que indicamos.

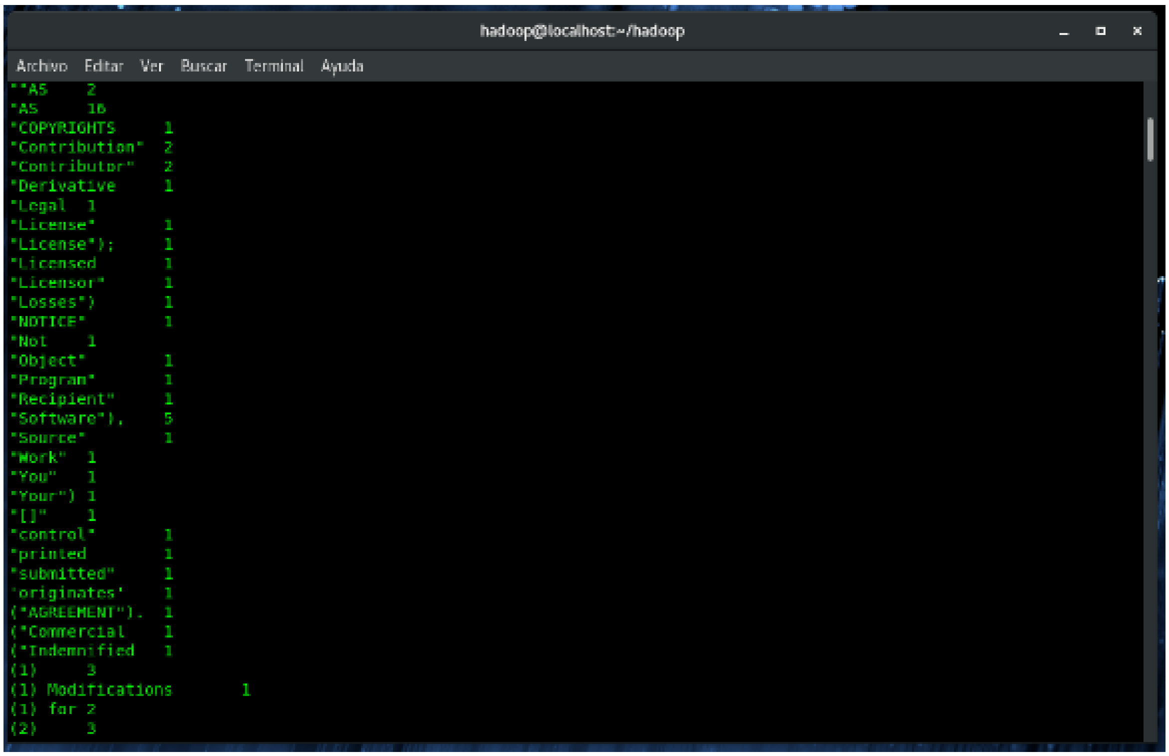


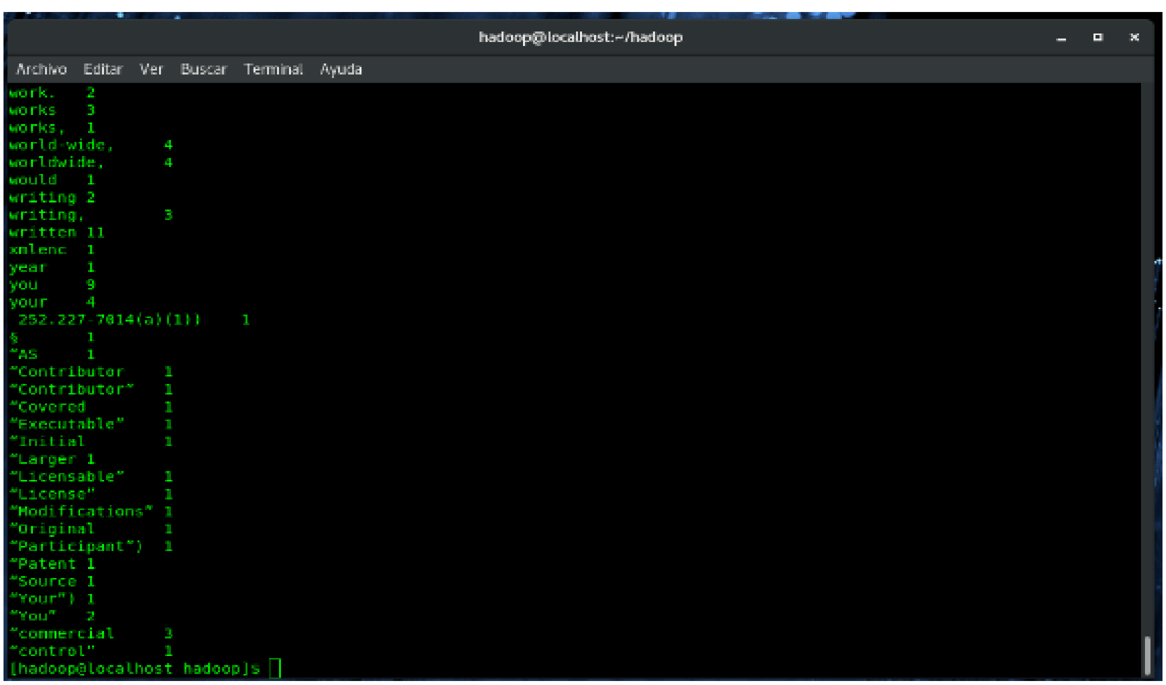
Ahora que se comprueba que hay archivos en la ruta de salida que indicamos vamos a observar el resultado que arroja el programa.



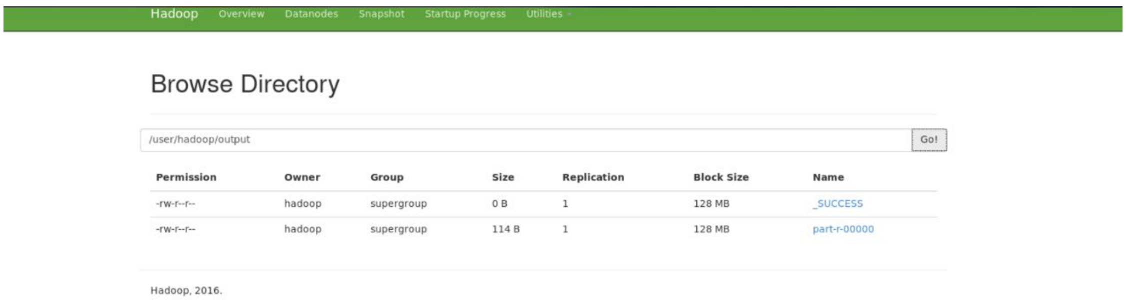
Lo que hacemos con el comando cat, es visualizar el contenido del archivo en la terminal, observamos palabras y un número a continuación de cada una de ellas.

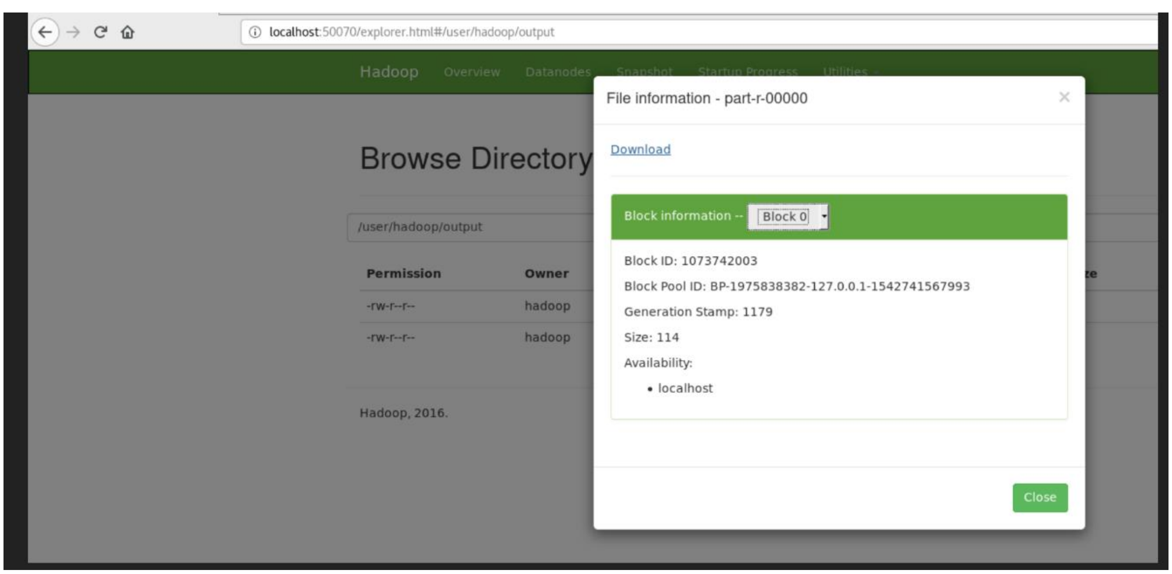
Lo cual quiere decir que la palabra aparece en el texto tantas veces como indica el número.



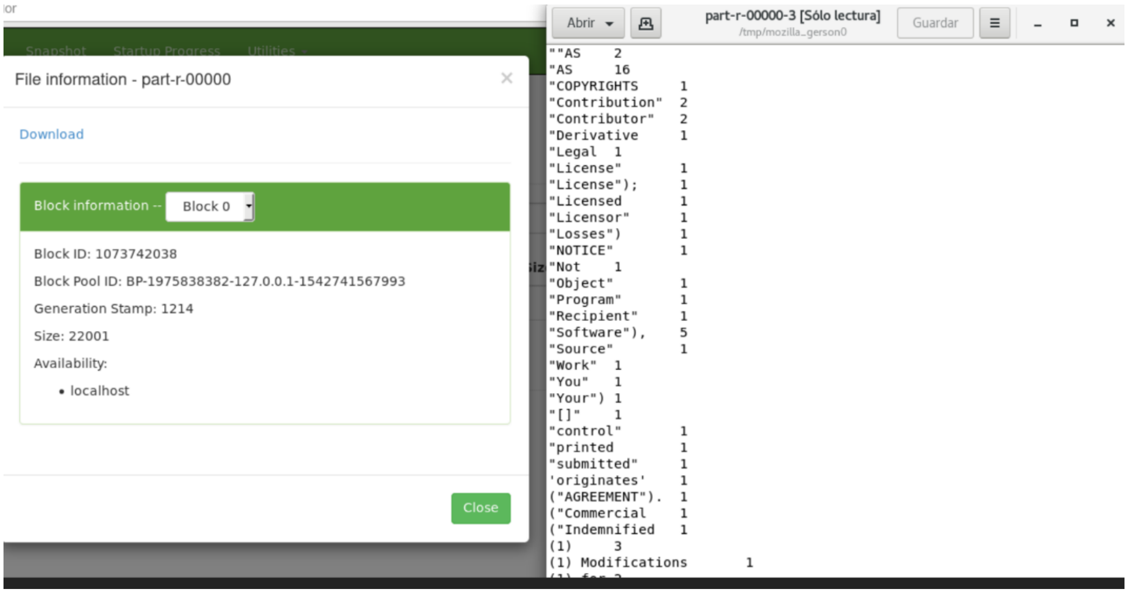


El resultado del programa lo podemos igual visualizar en el navegador y Web Browser que tiene Hadoop.

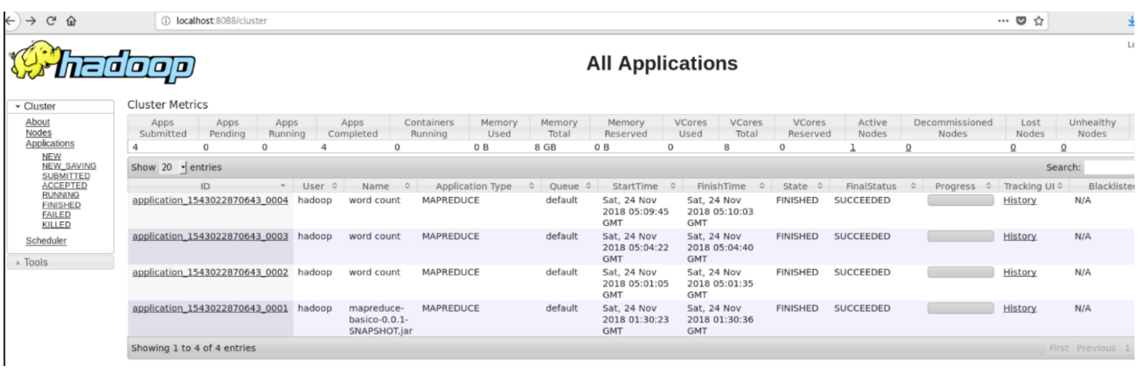


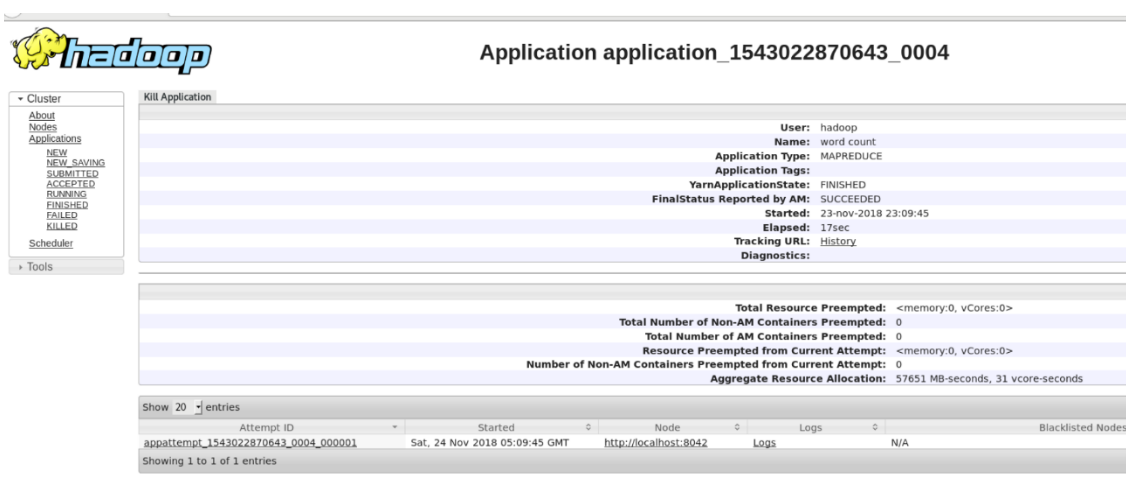


Incluso podemos descargar desde el navegador el archivo de salida que genera el programa y visualizarlo.



De igual manera podemos ver las aplicaciones ejecutadas en el cluster de Hadoop en el navegador. Y detalles de cada una de ellas.

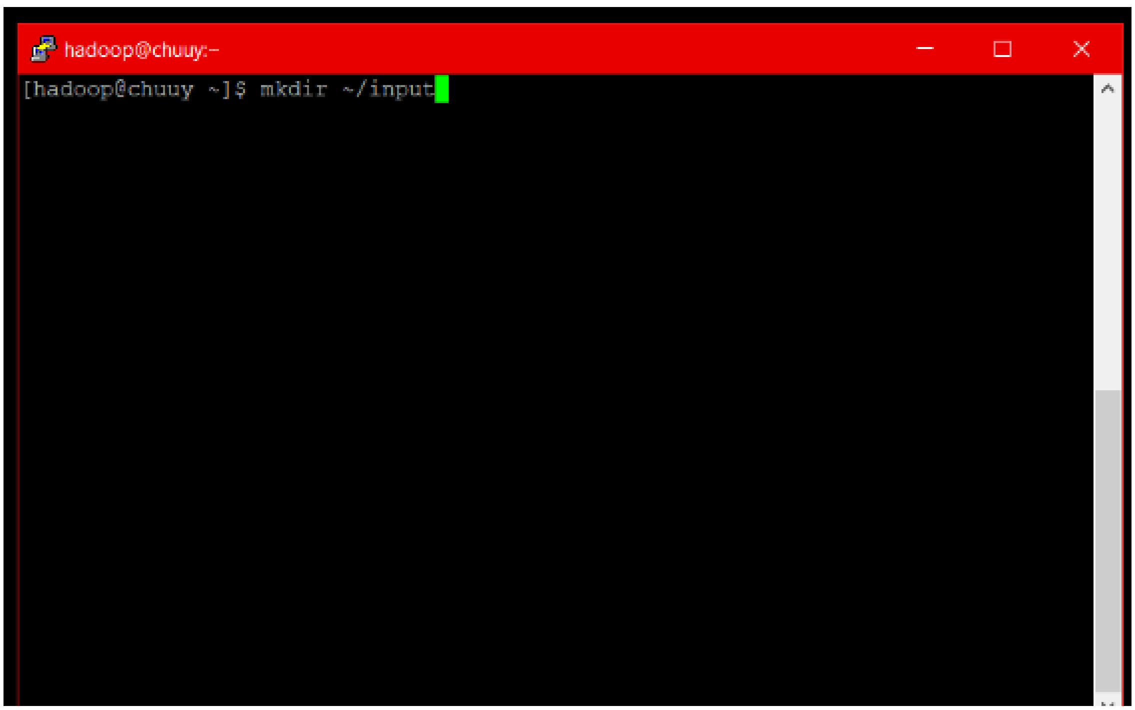


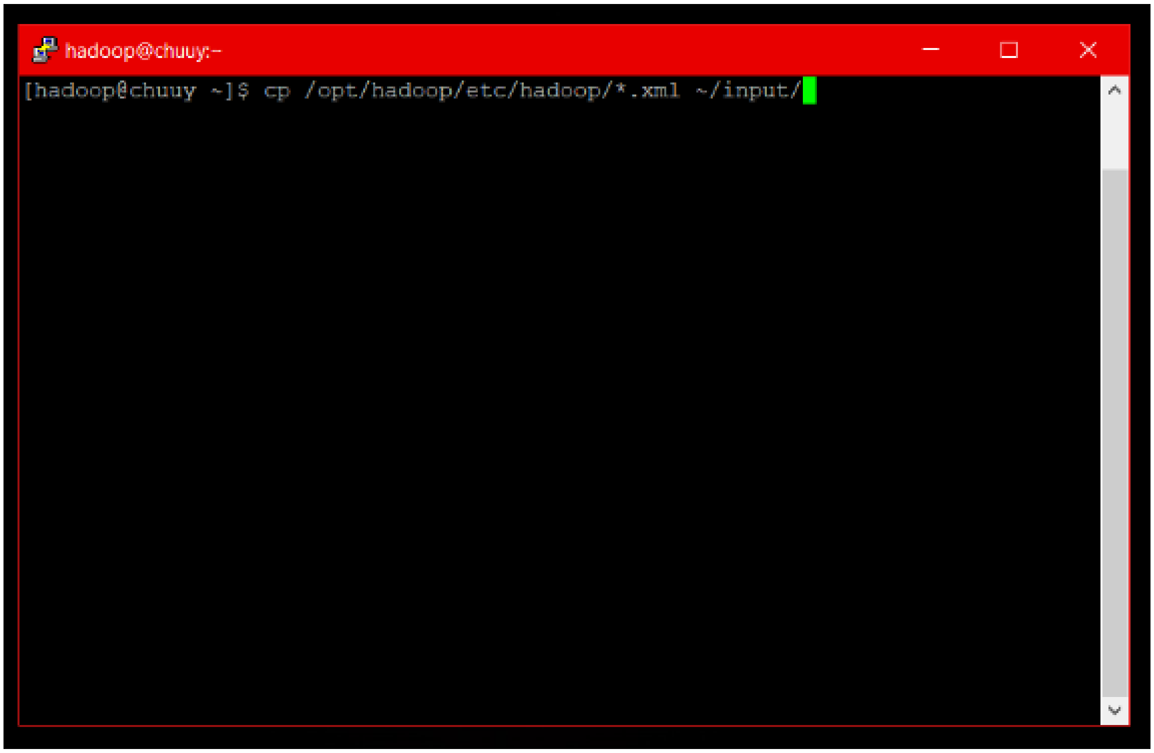


EJEMPLO "BUSQUEDA DE UNA PALABRA EN ARCHIVOS".

Haremos un ejercicio que consiste en encontrar todas las apariciones de una palabra usando una expresión regular. Para hacerlo, crearemos un directorio llamado input en nuestro directorio de inicio y copiaremos los archivos de configuración (los xml) de Hadoop para usar esos archivos como nuestros datos de entrada.

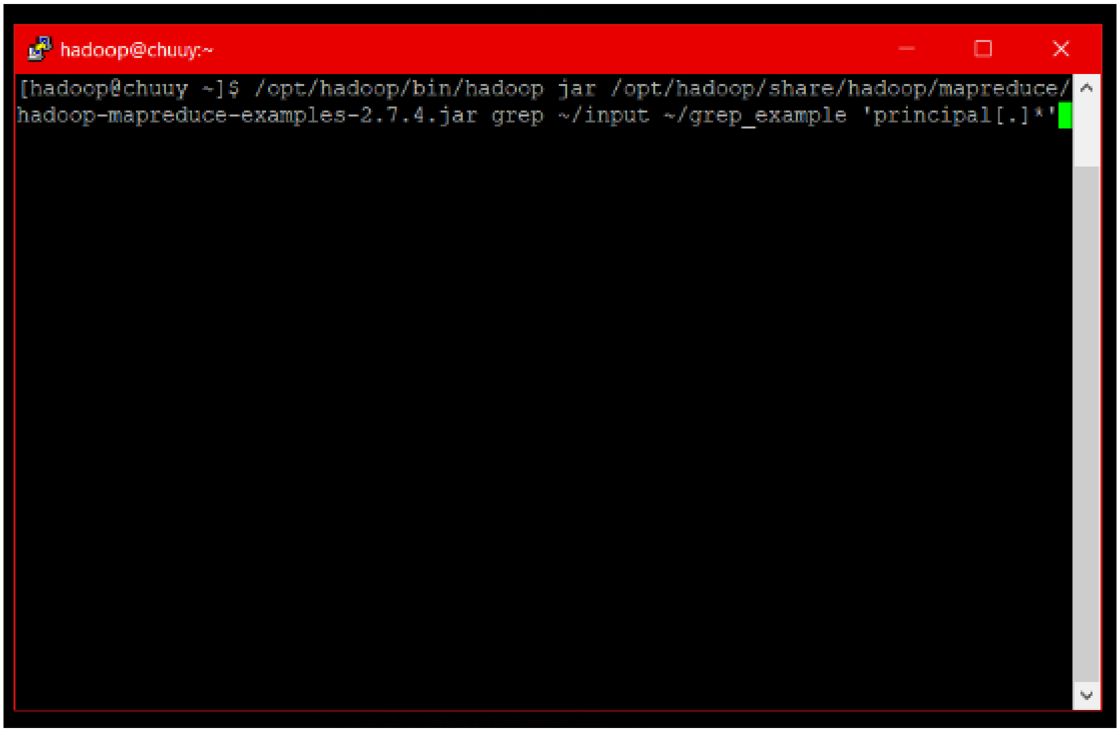
En la terminal de nuestro Sistema Operativo para este caso GNU/Linux CentOS7 escribiremos:





*Recordemos que (~) en terminal representa toda la ruta de nuestro Home

Podemos usar el siguiente comando para ejecutar el programa (jar) hadoop-mapreduce-examples, un archivo Java con varias opciones. Invocaremos grep, uno de los muchos ejemplos incluidos en hadoop-mapreduce-examples, seguido por el directorio de entrada input y el directorio de salida grep_example. El programa MapReduce grep contará las coincidencias de una palabra literal o expresión regular. Finalmente, proporcionaremos una expresión regular para encontrar ocurrencias de la palabra ***principal*** dentro o al final de una oración declarativa. La expresión distingue entre mayúsculas y minúsculas, por lo que **no** encontraríamos la palabra si estuviera en mayúscula al comienzo de una oración:

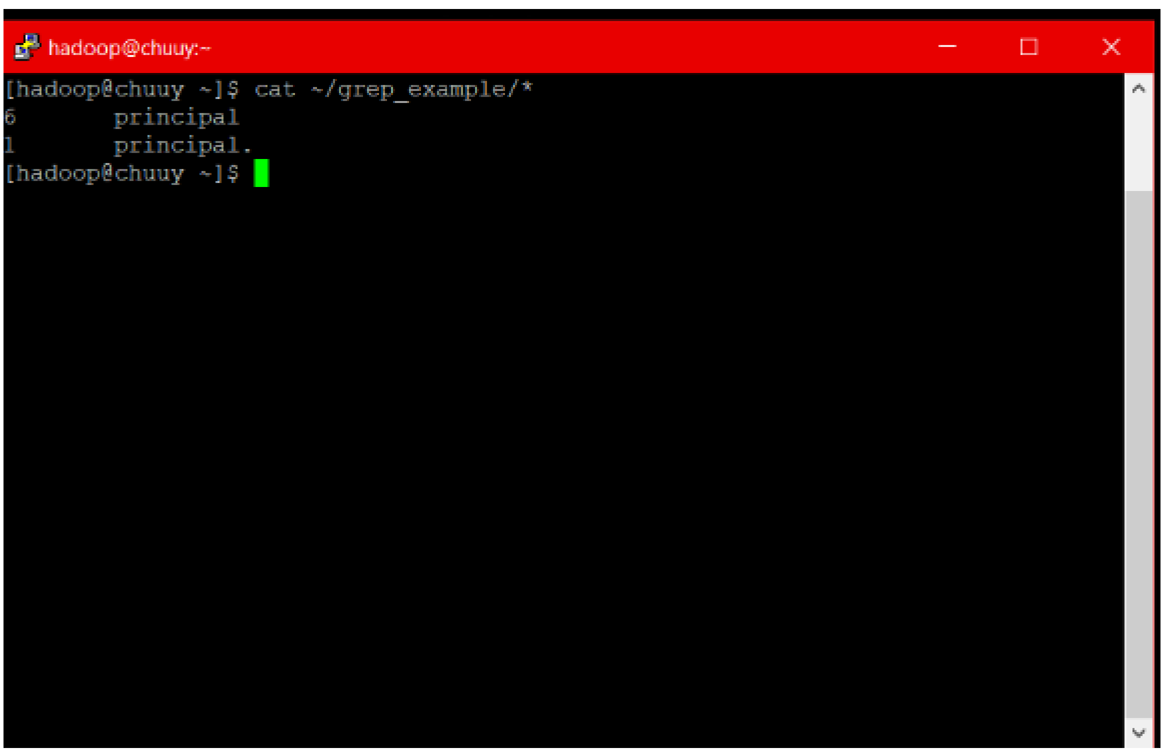


1. **/usr/local/hadoop/bin/hadoop:** Es el directorio donde está el ejecutable de hadoop en el sistema.
2. **jar:** Le indica a hadoop que deseamos ejecutar una aplicación empaquetada de Java. (Jar)
3. **/usr/local/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.0.1.jar :** Es la ruta donde está el Jar que deseamos ejecutar. Note que la versión del jar depende de la versión de hadoop instalada.
4. **grep**: Es un parámetro de los muchos que se le pueden pasar al Jar de ejemplos que trae Hadoop. grep sirve para encontrar y contar ocurrencias de strings haciendo uso de expresiones regulares.
5. **~/input**: El directorio de entrada. Es donde el programa va a buscar los archivos de entrada a la tarea de map-reduce. Aquí copiamos unos archivos de prueba en un comando anterior.

6. **~/grep_example:** El directorio de salida. Es donde el programa va a escribir el resultado de la corrida de la aplicación. En este caso, la cantidad de veces que la palabra del parámetro siguiente, aparece en los archivos de entrada.

7. **‘principal[.]*’:** Es la expresión regular que deseamos buscar. Esta expresión regular en particular coincide con la palabra ‘principal’ y ‘principal.’ (con y sin punto al final).

Los resultados se almacenan en el directorio de salida (~/grep_example/) y se pueden verificar ejecutando cat en el directorio de salida:



4.2 Ejemplo MapReduce para reducir, aprovechar un archivo de más de 400,000 líneas con información del aire en León y Castilla España.

Antes de implementar cualquier línea de código, debemos analizar lo que pretendemos extraer del archivo de datos. Si abrimos el archivo CSV que hemos descargado, esto es lo que nos aparece.

```
DIA;CO(mg/m3);NO(ug/m3);NO2(ug/m3);O3(ug/m3);PM10(ug/m3);SH2(ug/m3);PM25(ug/m3);PST
(ug/m3);SO2(ug/m3);PROVINCIA;ESTACION
    01/01/1997;1.2;12;33;63;56;;;;19;ÁVILA;Ávila
    02/01/1997;1.3;15;35;59;47;;;;17;ÁVILA;Ávila
    03/01/1997;1.5;18;43;54;65;;;;19;ÁVILA;Ávila
    04/01/1997;1.6;56;73;50;74;;;;22;ÁVILA;Ávila
    05/01/1997;1.4;11;33;63;54;;;;18;ÁVILA;Ávila
    06/01/1997;1.6;28;46;56;60;;;;20;ÁVILA;Ávila
    07/01/1997;1.5;19;41;58;47;;;;23;ÁVILA;Ávila
    08/01/1997;1.8;35;58;43;43;;;;27;ÁVILA;Ávila
    09/01/1997;1.1;14;34;60;55;;;;17;ÁVILA;Ávila
    10/01/1997;1.3;35;53;38;61;;;;18;ÁVILA;Ávila
    11/01/1997;1.3;27;40;25;59;;;;16;ÁVILA;Ávila
    12/01/1997;1.5;33;41;24;61;;;;16;ÁVILA;Ávila
    ....
```

El archivo tiene una línea por cada día de medición de diferentes medidas relativas a diversos contaminantes como son el monóxido de carbono, monóxido de nitrógeno, dióxido de nitrógeno, ozono, etc. por provincia y población donde se encuentra la estación donde se ha tomado la medida.

Para hacer una prueba dentro de MapReduce vamos a calcular la media de las muestras tomadas desde 2012 de Monóxido de Carbono (CO) agrupados por provincia, este cálculo es uno de algunos que podemos realizar con la herramienta además se constata que es una gran opción para hacer este tipo de operaciones manejando gran cantidad de datos.

Este es un cálculo que aporta valor, así podemos saber qué provincia de Castilla y León tiene un menor nivel de presencia de esta sustancia. Sin ser ningún entendido en química, se escogió este gas por ser una sustancia que se desprende al quemar gasolina, petróleo, carbón, etc. y que representan un problema al estar presentes en el aire.

El ranking final de las provincias menos contaminadas que resulte del cálculo que vamos a realizar, no se debe tomar como un estudio serio para determinar la calidad del aire presente en ellas, ya que se ha decidido tomar este indicador como podría haber sido cualquier otro.

Dentro del paradigma MapReduce, en la tarea mapper se reciben los bloques de datos que contienen la información. Es aquí donde debemos extraer la información que nos irá llegando línea a línea. Nuestra clase AirQualityMapper, extenderá de la clase Mapper definiendo los formatos de entrada de la clave y valor y los tipos de salida que devolverá la función. La salida del mapper será la entrada del reducer. En nuestro caso el valor de la clave contendrá el offset del archivo, es decir el puntero que va recorriendo el archivo entero y extrae cada una de las líneas del mismo. La línea nos la pasará en el campo value. Al heredar de Mapper debemos implementar el método map cuyos parámetros de entrada corresponderán con los tipos genéricos la que definimos anteriormente.

El método map es muy sencillo, considerando que en el campo value tenemos una línea del archivo de entrada, la troceamos separando por el token ";". Nos quedaremos con los valores que nos interesan, para nuestro análisis nos quedamos con el valor correspondiente al Monóxido de Carbono (CO) y con la provincia.

Aparte de la tarea de extracción del dato, el mapper también realiza la tarea de filtrado, en nuestro caso no emitimos ningún valor que no sea numérico. Hacemos esto porque hay líneas que no tienen todas las medidas bien informadas supongo que porque ese día la muestra tomada no es fiable de acuerdo a las directivas que regula la medición.

Una parte muy importante en la fase de map es la información que devuelve el propio método, como es lógico. Esto se realiza escribiendo en el objeto context siempre una tupla compuesta por una clave y un valor. La clave será la provincia y como valor el dato relativo a la medición de Monóxido de Carbono (CO). Los tipos que se escriben en el context son del tipo Writable. Es un tipo de datos específico en Hadoop, ya que por temas de rendimiento redefine los tipos de Java, por ejemplo, si la clave es de tipo string tenemos que crear un tipo Text, si el valor es de tipo Double, debemos crear un DoubleWritable. También podremos crear nuestros propios Writables para trabajar con objetos compuestos.

El reducer es la parte del algoritmo que recoge agrupadas por clave todos los valores emitidos en la fase map. En nuestro caso a la tarea reduce le llegará algo como (ÁVILA, [1.2, 1.3, 1.5, 1.6, 1.4…]). En esta fase debemos realizar la operación que calcule la media de los valores recibidos para la provincia.

De forma muy parecida al map, en este caso extendemos de Reducer y definimos los valores de entrada y de salida del método reduce que debemos implementar obligatoriamente. Los valores de entrada deben coincidir con los valores de salida de la función map. La salida del reduce será de tipo texto resultando la tupla que contendrá la provincia y el valor medio de las muestras tomadas de Monóxido de Carbono (CO) desde 1997.

Recorremos con un bucle for el iterable que Hadoop nos pasa con los valores de las muestras tomadas. En el campo key vendrá la provincia. Si existen medidas para la provincia se emitirá el resultado escribiendo en el objeto context. De forma similar al map, se escribe siempre una tupla key/value. En nuestro caso la provincia y la media obtenida redondeando en 2 decimales.

Ya que tenemos todo, sólo nos faltaría ejecutarlo. Para eso nos creamos una función Driver que no es más que un método main que creará un Job configurando todo lo necesario para ejecutar la tarea: función map, función reduce, tipos de datos que manejan, entrada y salida de datos, etc. Utilizaremos también la clase ToolRunner de Hadoop, una utilidad que se encarga de pasar en la invocación por línea de comandos de nuestra clase Driver una serie de parámetros de configuración que le pasemos.

CODIGO RESULTANTE

AirQualityManager.java

```
package com.tesis.example;

//imports ...

public class AirQualityManager extends Configured implements Tool {

  /** Map */
  public static class AirQualityMapper extends Mapper<Object, Text, Text, DoubleWritable> {

    private static final String SEPARATOR = ";";

    public void map(Object key, Text value, Context context) throws IOException, InterruptedException {
      ...
    }
  }

  /** Reduce */
  public static class AirQualityReducer extends Reducer<Text, DoubleWritable, Text, Text> {

    public void reduce(Text key, Iterable<DoubleWritable> coValues, Context context) throws IOException, InterruptedException {
      ...
    }
  }

  @Override
  public int run(String[] args) throws Exception {

    if (args.length != 2) {
      System.err.println("AirQualityManager required params: {input file} {output dir}");
      System.exit(-1);
    }

    deleteOutputFileIfExists(args);

```

AirQualityManager.java

```java
    deleteOutputFileIfExists(args);

    final Job job = new Job(getConf());
    job.setJarByClass(AirQualityManager.class);
    job.setInputFormatClass(TextInputFormat.class);
    job.setOutputFormatClass(TextOutputFormat.class);

    job.setMapperClass(AirQualityMapper.class);
    job.setReducerClass(AirQualityReducer.class);

    job.setMapOutputKeyClass(Text.class);
    job.setMapOutputValueClass(DoubleWritable.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(Text.class);

    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));

    job.waitForCompletion(true);

    return 0;
  }

  private void deleteOutputFileIfExists(String[] args) throws IOException {
    final Path output = new Path(args[1]);
    FileSystem.get(output.toUri(), getConf()).delete(output, true);
  }

  public static void main(String[] args) throws Exception {
    ToolRunner.run(new AirQualityManager(), args);
  }
}
```

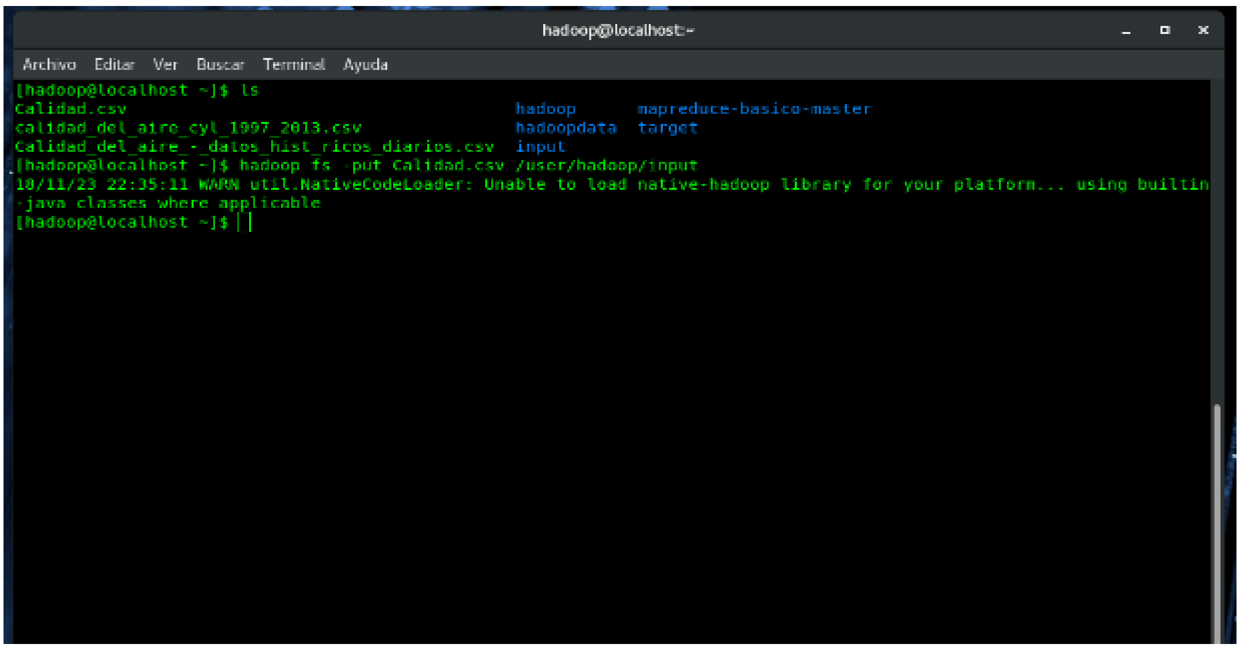
AirQualityManager.java* 1:1 CRLF UTF-8 Java

(El código completo esta en el anexo A)

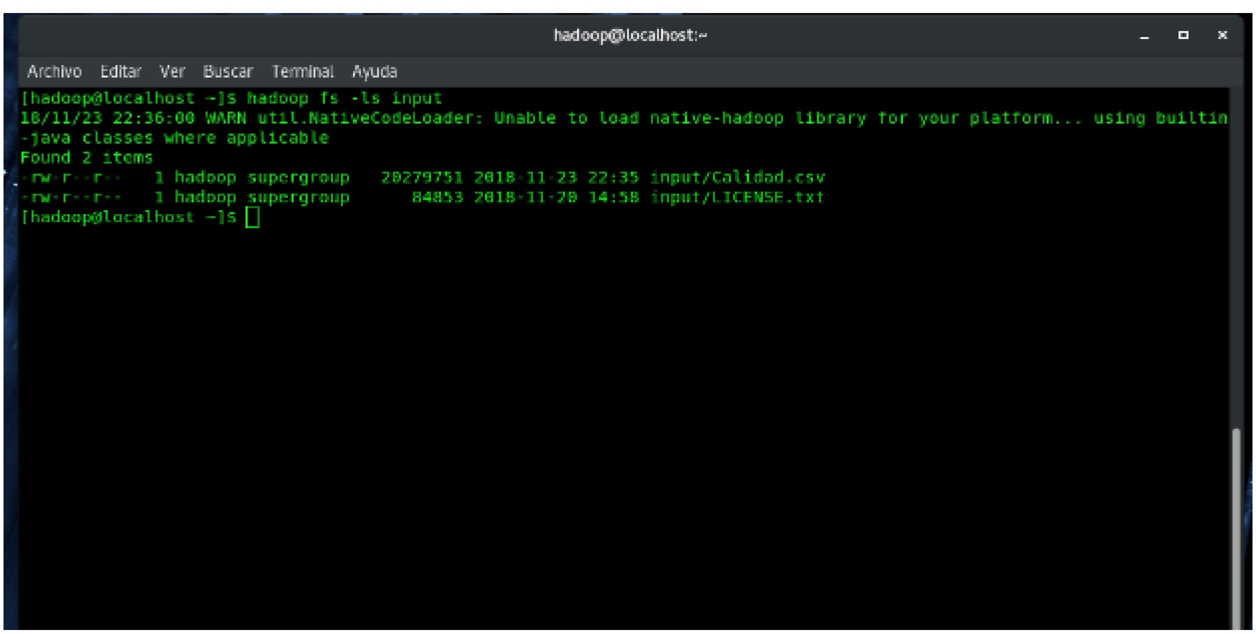
Creamos la clase AirQualityManager, que para poder invocarse a través de ToolRunner debe extender de Configured e implementar el interfaz Tool. Define el método main que servirá de punto de entrada a nuestra función. En el main invocamos al ToolRunner creando una instancia de nuestra clase manager pasándole los argumentos que recogemos por línea de comandos.

Esto llamará a nuestro método run que valida los datos recibidos como entrada, ya que es necesario introducir el archivo que contiene los datos y también el directorio de salida donde se escribirá el resultado. A continuación, borra el directorio si ya existe (Hadoop da un error si ya está creado) y crea un nuevo Job pasándole la configuración recogida por línea de comandos si la hubiera. Al job hay que configurarle una serie de parámetros como son el formato de entrada y salida, en nuestro caso una entrada de texto línea a línea, el mapper y el reducer, los tipos de datos que se manejan, los archivos de entrada y el directorio de salida. Invocamos y esperamos mediante la llamada al método waitForCompletion. El valor true indica el modo verbose.

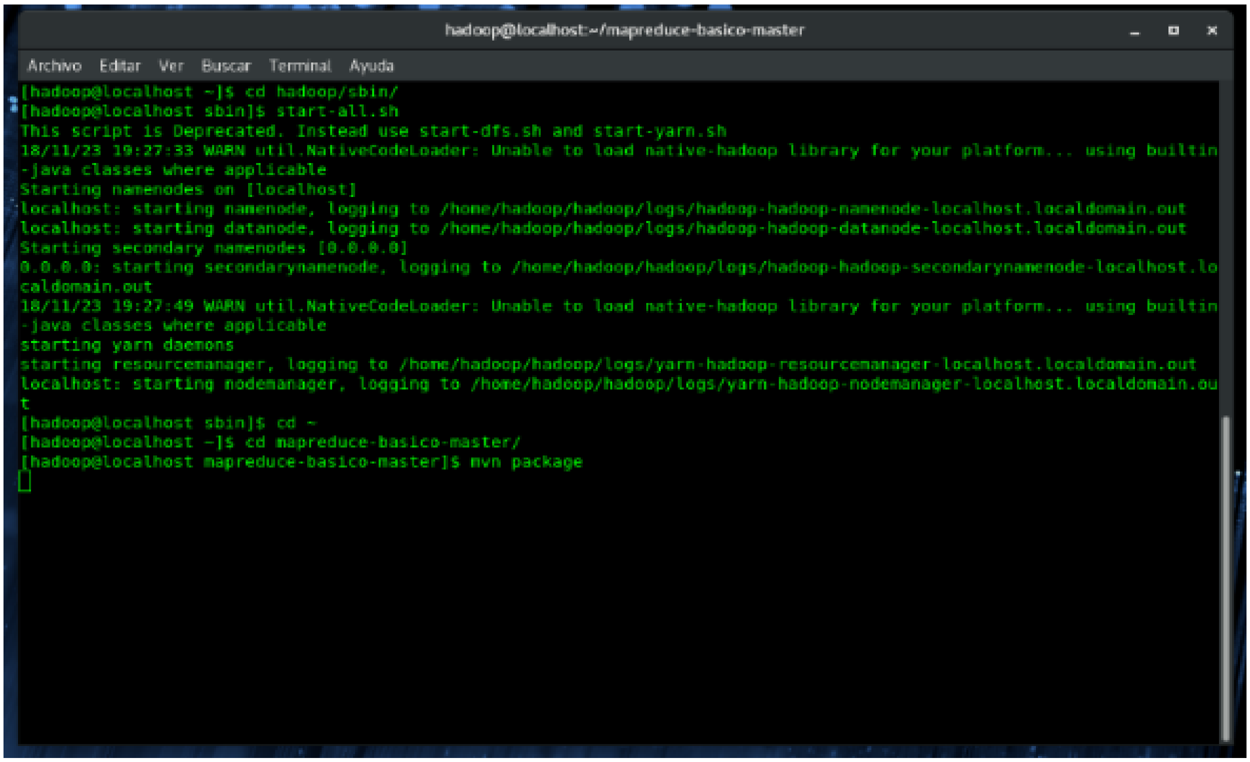
Ya sólo queda ejecutar y esperar el resultado. Para ello antes debemos crear el directorio y subir al HDFS el archivo que contiene los datos mediante el comando:

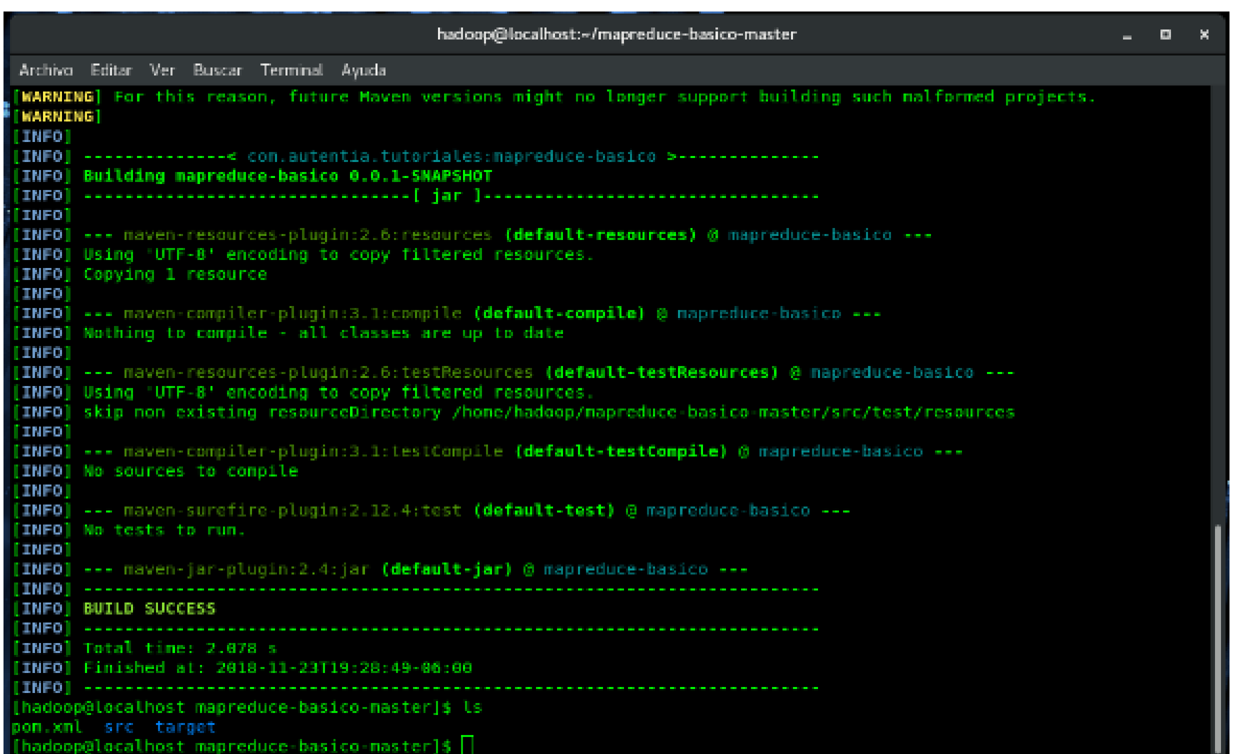


Podemos comprobar si se ha subido bien mediante el comando:

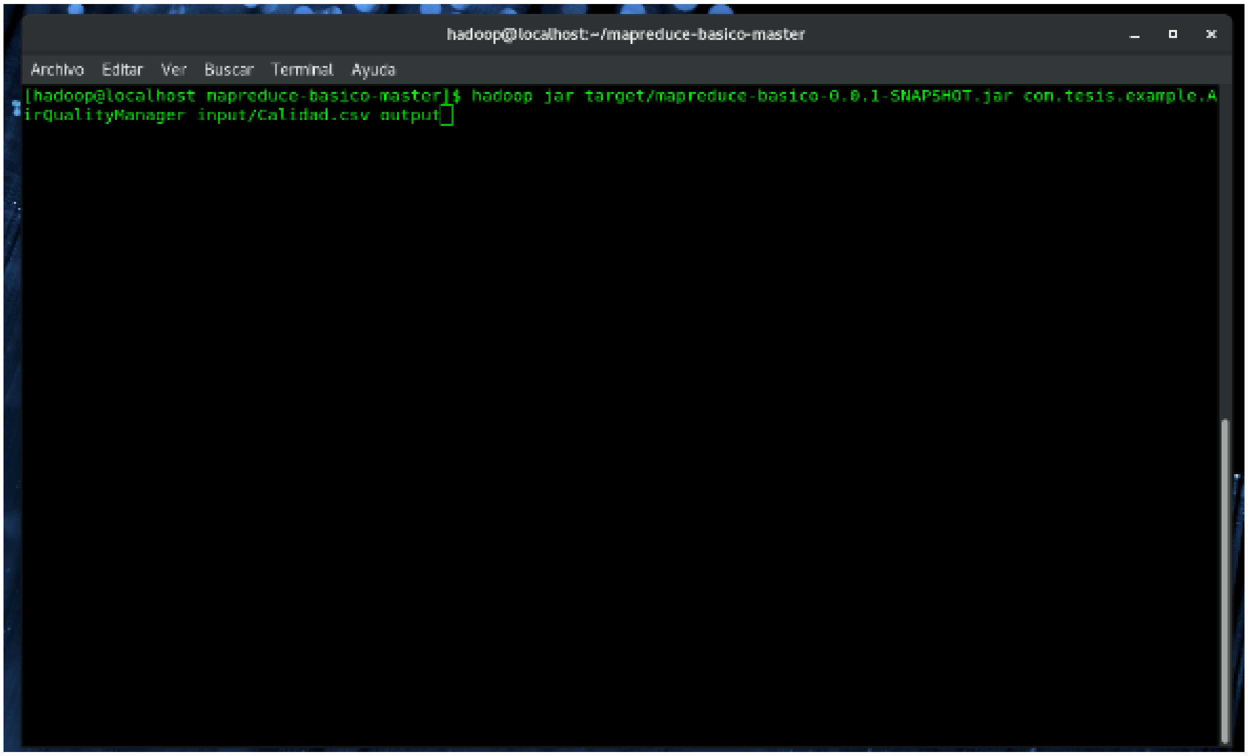


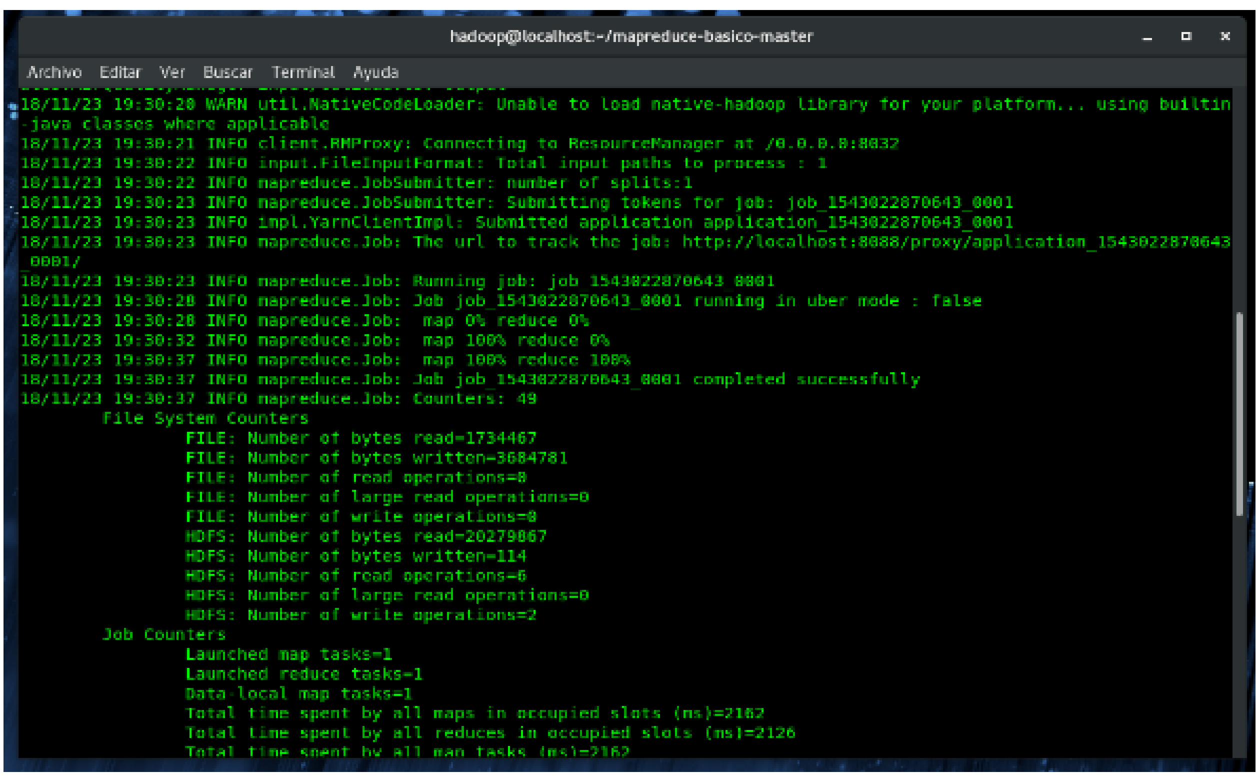
El siguiente paso será generar un jar que será la forma de pasarle a Hadoop el código que tiene que ejecutar. Para ello usamos maven haciendo un **mvn package.**

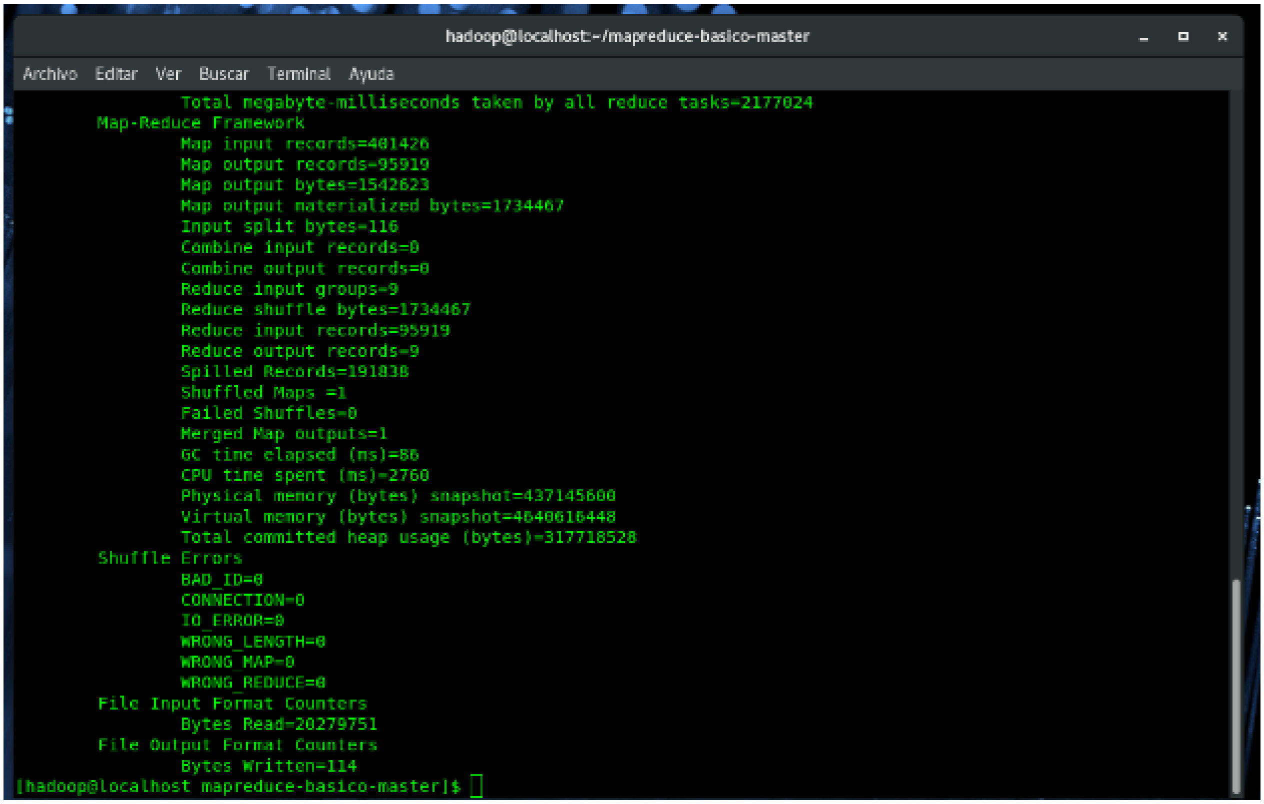




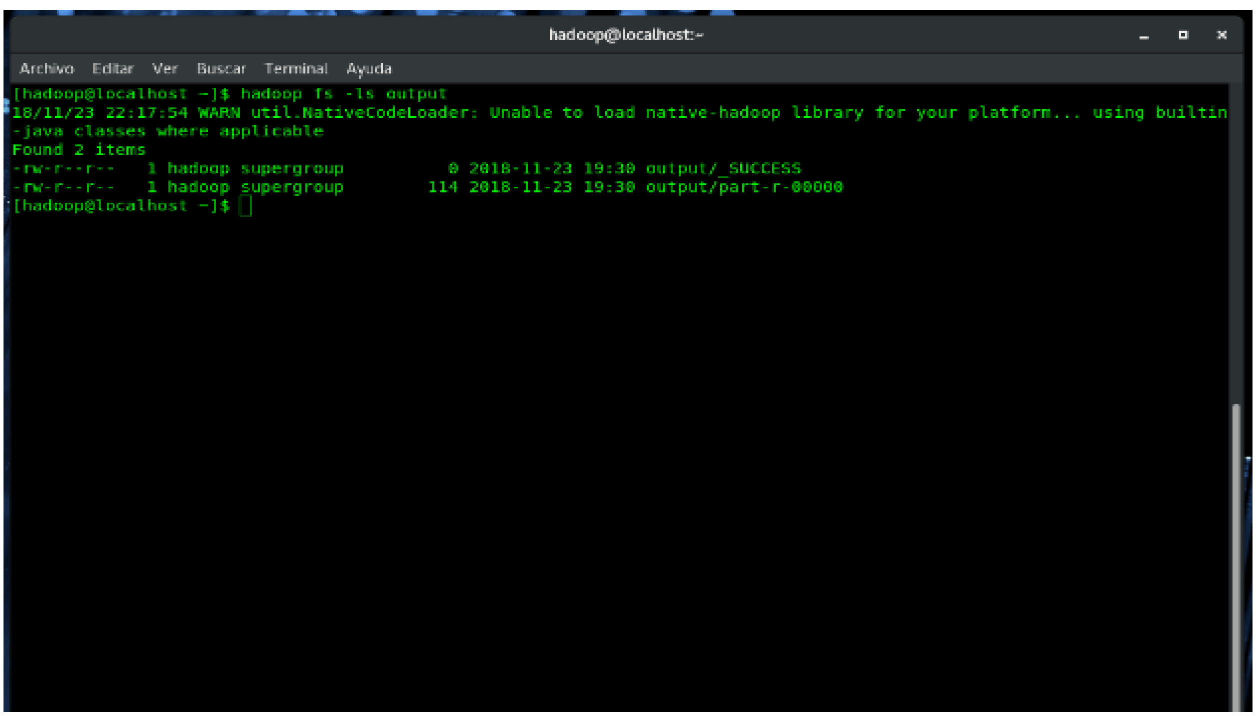
Una vez hecho esto ejecutamos el siguiente comando para lanzar el Job.



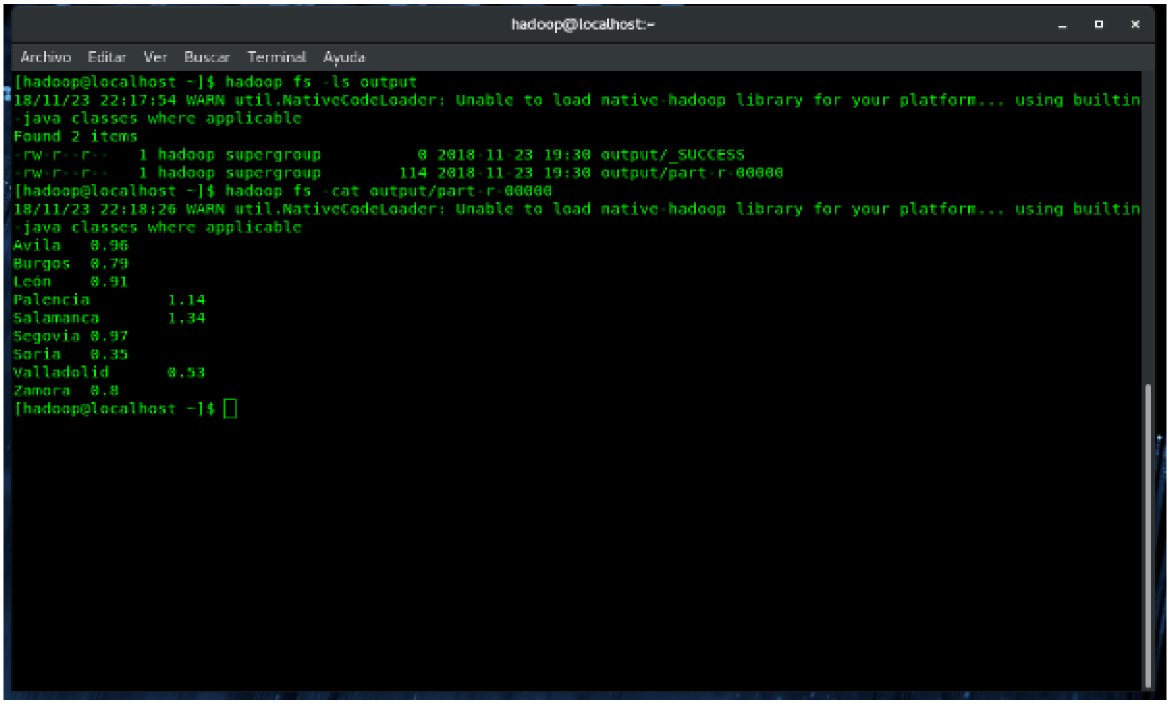




Después de unos pocos segundos nos dejará el resultado en el directorio output.

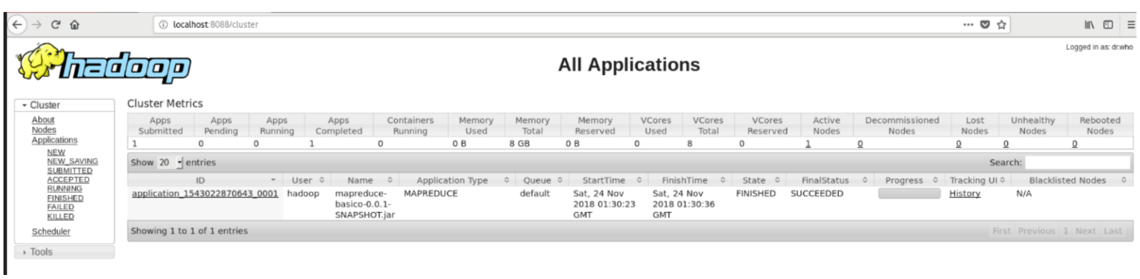


Lo abrimos y tendremos el resultado del cálculo de la media de Monóxido

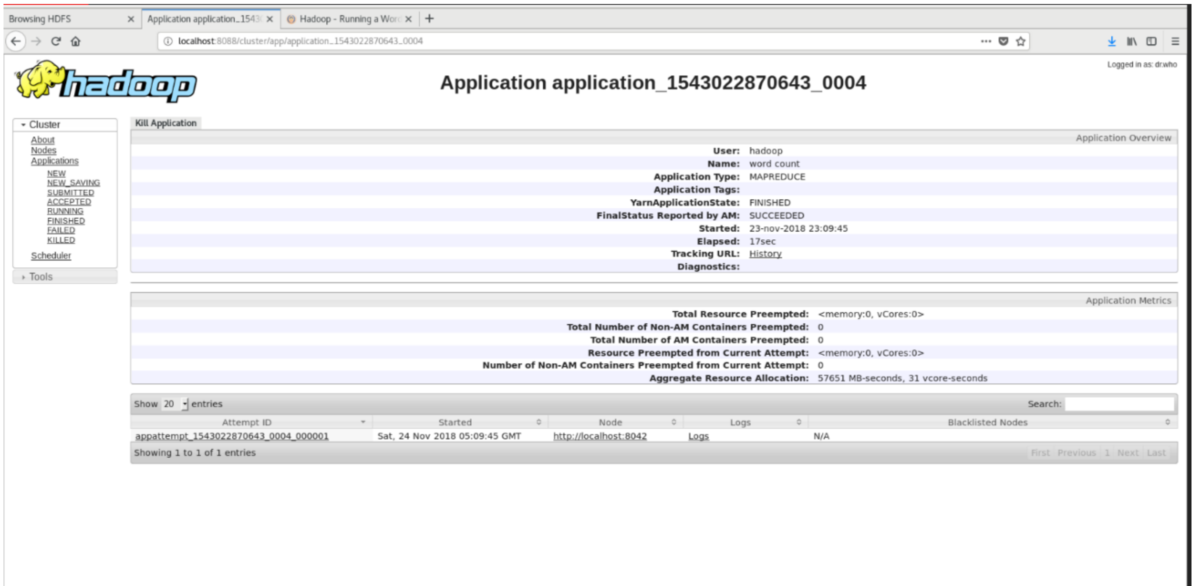


Como ya vimos podemos visualizar tanto la aplicación y sus detalles en el

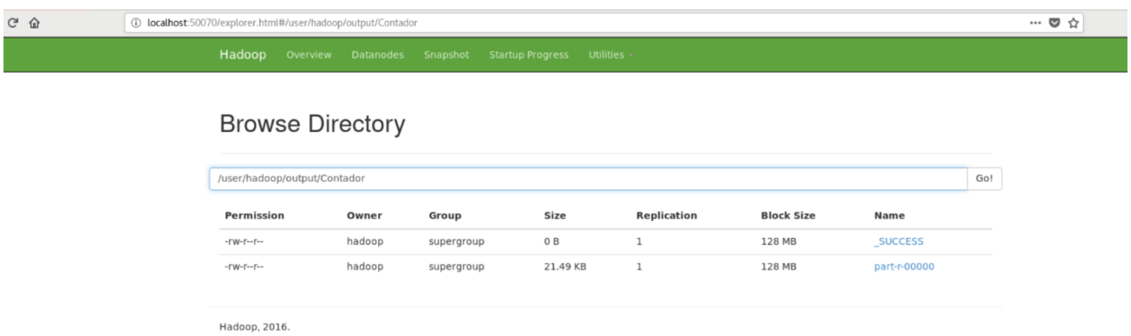
Navegador.

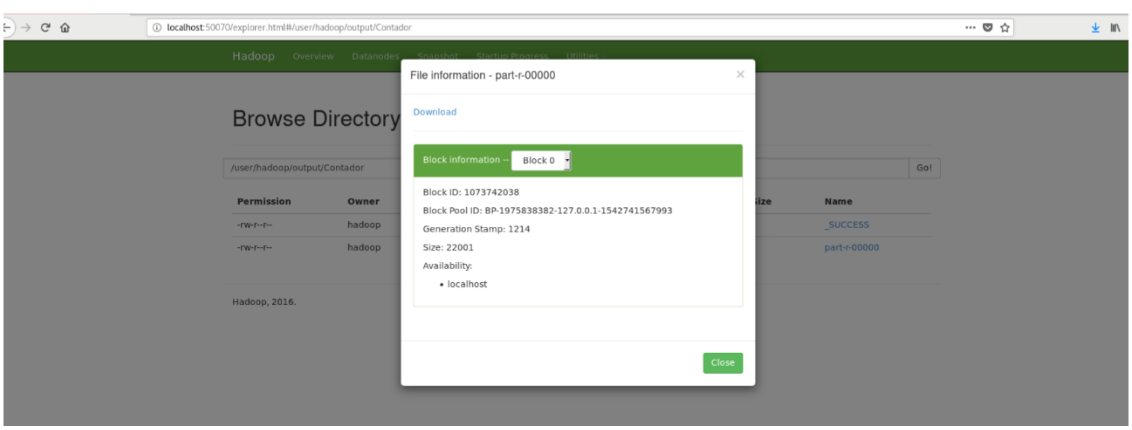


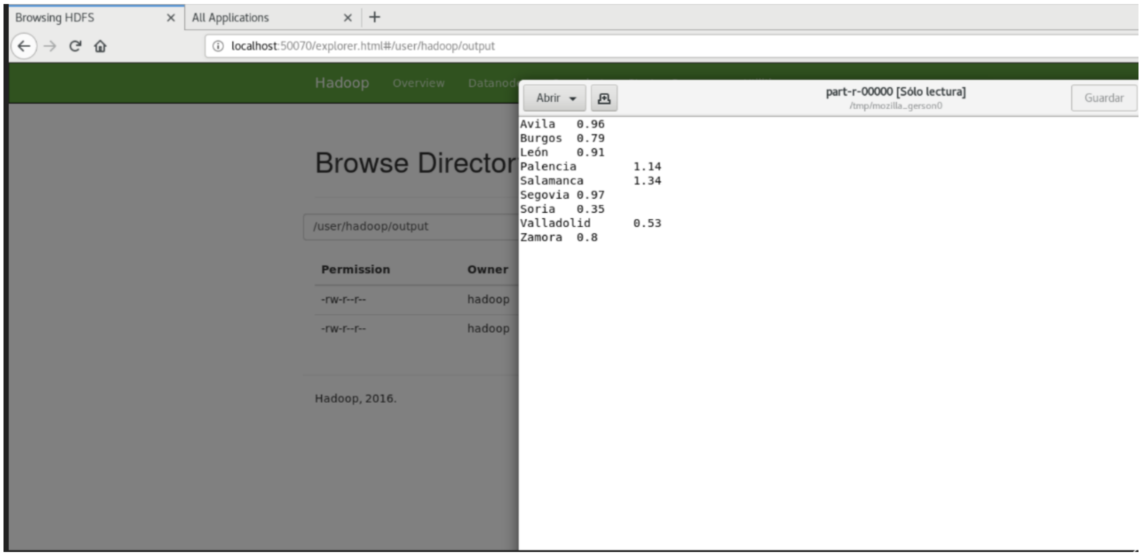
En este caso nos muestra la aplicación que ejecutamos y su estado que es finalizado, y en detalles podemos observar más información.



Los resultados obtenidos del Programa los podemos visualizar y descargar desde el Web Browser de Hadoop:







Hemos visto como crear un Job en Hadoop y extraer información de un archivo que a simple vista no nos dice gran cosa a menos que se procese. El archivo con el que hemos trabajado era pequeño, unas 400.000 líneas (19 MB). Usando archivos de cientos de millones de líneas, quizá la tarea no tardaría unos segundos en terminar sino horas o días.

Conclusiones

Conocer a fondo el funcionamiento de MapReduce, permite entender la revolución que se está presentando en la acumulación y utilización de datos con la aplicación de la digitalización, la informática y la Internet, en el presente trabajo se presenta con detalle este funcionamiento y como ha sido utilizado el MapReduce para afrontar los retos de lo que ahora se llama BigData (manejo de grandes volúmenes de datos).

En este trabajo también se plantean diversas herramientas utilizadas de forma común en la comunidad como HDFS, Hadoop, Spark o Yarn, que fueron probadas ya que los sistemas de archivos y procesamiento de datos actuales se ven rebasados y no dan los resultados requeridos.

El material presentado en este trabajo también muestra el impacto que ha tenido esta revolución en la generación de nuevos lenguajes de programación como SCALA, Python, R y desde luego PIG BIG DATA que es un lenguaje de alto nivel para crear flujos de datos llamado Pig Latin el cual permite realizar programas MapReduce de forma simple y en pocas líneas de código.

Se utilizaron muestras visuales Web de Hadoop que permiten reconocer el funcionamiento adecuado de to el código utilizado así como el desempeño de las nuevas herramientas, se espera que la revisar las pruebas realizadas sea motivante para que se utilicen de forma más amplia estas tecnologías que nos acerquen al desarrollo tecnológico que se está dando a nivel mundial.

Líneas de Trabajo

El presente trabajo realizado cumple como análisis de MapReduce respecto al objetivo de explicar el funcionamiento. Para efectuar mejores programas y propuestas innovadoras, una vez comprendido como trabaja y funciona, es posible incluso tomar alguna alternativa como YARN o SPARK. Entendiendo de igual manera que con otros lenguajes de programación, pero llevados a ejecutarse bajo el modelo de programación MapReduce pueden incluso ser más veloces y eficientes a la hora de construir aplicaciones.

Bibliografía

“Lublinsky Boris”, “Smith Kevin”, Hadoop: Soluciones Big Data, 2014

“Jose Aguilar“ Big Data: Fundamentos, Estrategias, Gestión de datos (Hadoop, Map/Reduce, NoSQL), Análisis de datos (Spark, Splunk), 2013

“Dirk deRoos”, Haddop for Dummies, 2014

“Miner Donald”, “Shook Adam”, MapReduce Design Patterns: Building Effective Algorithms and Analytics for Hadoop and Other Systems, 2012

“White Tom”, Hadoop: The Definitive Guide, 2012

“Alan Kaminsky”, Big CPU, Big Data, 2015

“Jimmy Lin”,”Chris Dyer”, Data-Intensive Text Processing with Map Reduce, 2010

“Andy Konwinski”, Learning Spark: Lightning-Fast Big Data Analysis, 2015

“Uri Laserson”, Advanced Analytics with Spark, 2015

“Lazy Programmer”, Big Data, MapReduce, Hadoop, and Spark with Python: Master Big Data Analytics and Data Wrangling with MapReduce Fundamentals using Hadoop, Spark, and Python (English Edition), 2015

“Yolanda” Olmedo”, Blog de Solid Q - Big Data, Agosto 20, 2012,

http://blogs.solidq.com/es/big-data/que-es-mapreduce.html

Anexos

1. Codigo completo de la clase AirQualityManager

```
public class AirQualityManager extends Configured implements Tool {

	public static class AirQualityMapper extends Mapper<Object, Text, Text, DoubleWritable> {

		private static final String SEPARATOR = ";";

		/**
		 * DIA; CO (mg/m3);NO (ug/m3);NO2 (ug/m3);O3 (ug/m3);PM10 (ug/m3);SH2 (ug/m3);PM25 (ug/m3);PST (ug/m3);SO2 (ug/m3);PROVINCIA;ESTACIÓN <br>
		 * 01/01/1997; 1.2; 12; 33; 63; 56; ; ; ; 19 ;ÁVILA ;Ávila
		 */
		public void map(Object key, Text value, Context context) throws IOException, InterruptedException {
			final String[] values = value.toString().split(SEPARATOR);

			// final String date = format(values[0]);
			final String co = format(values[1]);
			// final String no = format(values[2]);
			// final String no2 = format(values[3]);
			// final String o3 = format(values[4]);
			// final String pm10 = format(values[5]);
			// final String sh2 = format(values[6]);
			// final String pm25 = format(values[7]);
			// final String pst = format(values[8]);
			// final String so2 = format(values[9]);
			final String province = format(values[10]);
			// final String station = format(values[11]);
```

```
                if (NumberUtils.isNumber(co.toString())) {
                    context.write(new Text(province), new
DoubleWritable(NumberUtils.toDouble(co)));
                }
            }

            private String format(String value) {
                return value.trim();
            }
        }

        public static class AirQualityReducer extends Reducer<Text, DoubleWritable, Text, Text> {

            private final DecimalFormat decimalFormat = new DecimalFormat("#.##");

            public void reduce(Text key, Iterable<DoubleWritable> coValues, Context context)
throws IOException, InterruptedException {
                int measures = 0;
                double totalCo = 0.0f;

                for (DoubleWritable coValue : coValues) {
                    totalCo += coValue.get();
                    measures++;
                }

                if (measures > 0) {
                    context.write(key, new Text(decimalFormat.format(totalCo /
measures)));
                }
            }
```

```
	}

	@Override
	public int run(String[] args) throws Exception {

		if (args.length != 2) {
			System.err.println("AirQualityManager required params: {input file} {output
dir}");
			System.exit(-1);
		}

		deleteOutputFileIfExists(args);

		final Job job = new Job(getConf());
		job.setJarByClass(AirQualityManager.class);
		job.setInputFormatClass(TextInputFormat.class);
		job.setOutputFormatClass(TextOutputFormat.class);

		job.setMapperClass(AirQualityMapper.class);
		job.setReducerClass(AirQualityReducer.class);

		job.setMapOutputKeyClass(Text.class);
		job.setMapOutputValueClass(DoubleWritable.class);
		job.setOutputKeyClass(Text.class);
		job.setOutputValueClass(Text.class);

		FileInputFormat.addInputPath(job, new Path(args[0]));
		FileOutputFormat.setOutputPath(job, new Path(args[1]));
```

```
        job.waitForCompletion(true);

        return 0;
    }

    private void deleteOutputFileIfExists(String[] args) throws IOException {
        final Path output = new Path(args[1]);
        FileSystem.get(output.toUri(), getConf()).delete(output, true);
    }

    public static void main(String[] args) throws Exception {
        ToolRunner.run(new AirQualityManager(), args);
    }
}
```