ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"SISTEMA DE ARCHIVOS POR CAPAS EN HADOOP HDFS"

INFORME DE MATERIA INTEGRADORA

Previo a la obtención del Título de:

INGENIERO EN CIENCIAS COMPUTACIONALES

LUIS ANDRÉS CAVIEDES RUIZ

LUIS FERNANDO VÁSQUEZ RUGEL

GUAYAQUIL – ECUADOR

AÑO: 2016

TRIBUNAL DE EVALUACIÓN

Cristina Lucía Abad Robalino (Ph.D)

PROFESOR EVALUADOR

Xavier Ochoa Chehab (Ph.D)

PROFESOR EVALUADOR

DECLARACIÓN EXPRESA

"La responsabilidad y la autoría del contenido de este Trabajo de Titulación, nos corresponde exclusivamente; y damos nuestro consentimiento para que la ESPOL realice la comunicación pública de la obra por cualquier medio con el fin de promover la consulta, difusión y uso público de la producción intelectual"

Luis Andrés Caviedes Ruiz

Luis Fernando Vásquez Rugel

RESUMEN

En este proyecto, presentamos el desarrollo e implementación de un Sistema de Archivos por Capas automatizado, basado en el Hadoop File System. Este sistema se implementó para poder organizar eficientemente y automáticamente ficheros dentro de un sistema de archivos multicapas en HDFS, y está diseñado para servir a cualquier usuario (ya sea para uso personal, como para uso en un sistema empresarial). El sistema trabaja de manera autónoma y es de uso relativamente sencillo para el usuario, ya que puede ser utilizado sin necesidad de conocer por completo el funcionamiento del sistema en cuestión.

ÍNDICE GENERAL

CAPÍTULO 1

1. INTRODUCCIÓN

1.1 ¿Qué es Hadoop File System?

Hadoop File System, la parte esencial de nuestro proyecto, es un sistema de archivos distribuido, de código abierto mantenido y distribuido por Apache.El HDFS ha sidocreado en Java, para el framework Hadoop.

Su versatilidad ha sido de gran ayuda en el mundo de la computación moderna, en especial en la rama del Big Data, donde su robustez y rapidez a la hora de manejar grandes cantidades de archivos lo ha convertido en una de las referencias más grandes en esta rama.

1.2 ¿Cómo funciona Hadoop File System?

El Hadoop File System se maneja equilibrando la carga entre 2 principales tipos de nodos, el “Namenode”, del cual usualmente solo existe uno, y los “Datanodes”, o nodos esclavos, de los cuales pueden existir uno o muchos. El Namenode, nodo maestro, es el encargado de manejar los punteros a los archivos dentro del sistema de archivos; todos los metadatos, configuraciones y ubicaciones se almacenan ahí. Adicionalmente, todas las operaciones se llevan a cabo sobre el Namenode. Los Datanodes, o los esclavos, almacenan los archivos, en forma de bloques.

Cabe recalcar que individualmente los Datanodes no contienen ningún tipo de ordenamiento.El ordenamiento de los archivos, así como su replicación dentro de los Datanodes, es trabajo del Namenode.

1.3 ¿Por qué usar Hadoop File System?

Siendo un sistema distribuido, su uso está más enfocado al manejo de grandes cantidades de datos [1] [2], a diferencia de un sistema de archivos común. Su eficiencia a la hora de manejar grandes cantidades de datos y operaciones se debe a su capacidad de “balancear” el trabajo entre diferentes nodos o máquinas, en vez de usar solo la capacidad de una máquina para balancear el trabajo del sistema.

Es considerado el equivalente a una RAID, solo que está basado en software en lugar de hardware [3]. Aun existiendo más competencia en el área de los sistemas de archivos distribuidos, Hadoop File System ofrece una muy alta capacidad de procesamiento de archivos y operaciones grandes (en el rango de los TB’s) [1].

1.4 Objetivos

Las tecnologías de Big Data están siendo adoptadas cada vez más en el mundo. Un elemento clave son los Sistemas de Archivos para grandes datos, los cuales son implementados de manera distribuida.Dichos sistemas de archivos proporcionan acceso rápido a los datos. En nuestra investigación, nuestro principal objetivo es implementar un sistema de archivos por capas dentro del Hadoop File System, en el cual nos ayudamos con conocimiento del framework Hadoop, así como con nuestro conocimiento previamente adquirido de Java y el sistema operativo Linux.

CAPÍTULO 2

2. METODOLOGÍA

2.1 Preparación de ambiente y pruebas

Lo primero que se implementó en este proyecto fue el ambiente en el cual se iba a desarrollar el programa, en este caso, se empezó con pruebas de un solo clúster en una sola máquina corriendo Ubuntu 14.04 LTS. Luego instalamos y compilamos desde el código fuente Hadoop 2.7.1, la versión estable más reciente (en Noviembre del 2015) del framework de Apache Hadoop, como se muestra en la Figura 2.1.Esta instalación, aparte de ser necesaria para la primera ronda de pruebas y para el trabajo, también nos ayudó a entender mucho más la configuración de Hadoop File System.

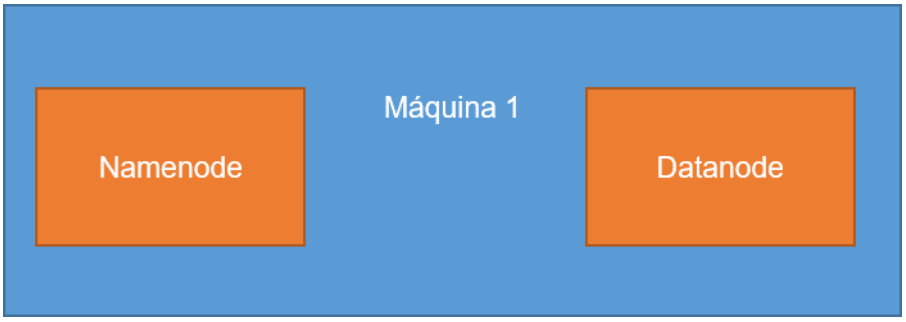


Figura 2.1: Hadoop File System instalado en un Solo Nodo

Una vez instalado Hadoop File System, y habiendo leído y comprendido más sobre el sistema de archivos, procedimos a ejecutar la misma instalación en un diseño multimodal, el cual requería más de una computadora. Ya que físicamente implementar esta topología iba a resultar imposible por tiempo y por recursos, recurrimos a Amazon Web Services. Haciendo el uso del servicio de máquinas virtuales en la nube (EC2) de Amazon, pudimos montar un clúster de 3 máquinas todas con 10 GB de Disco (2 de ellas SSD, una HDD) y con 1GB de RAM, con Ubuntu 14.0 LTS Server, las cuales representaron nuestros Data y Name Nodes, respectivamente (1 Namenode y 2 Datanodes).

Una vez completado el levantamiento del clúster, se procedió a instalar y configurar el mismo framework usado en el paso anterior, solo que con consideraciones ahora que se trata de un clúster, y no de una sola máquina, como se muestra en la Figura 2.2.

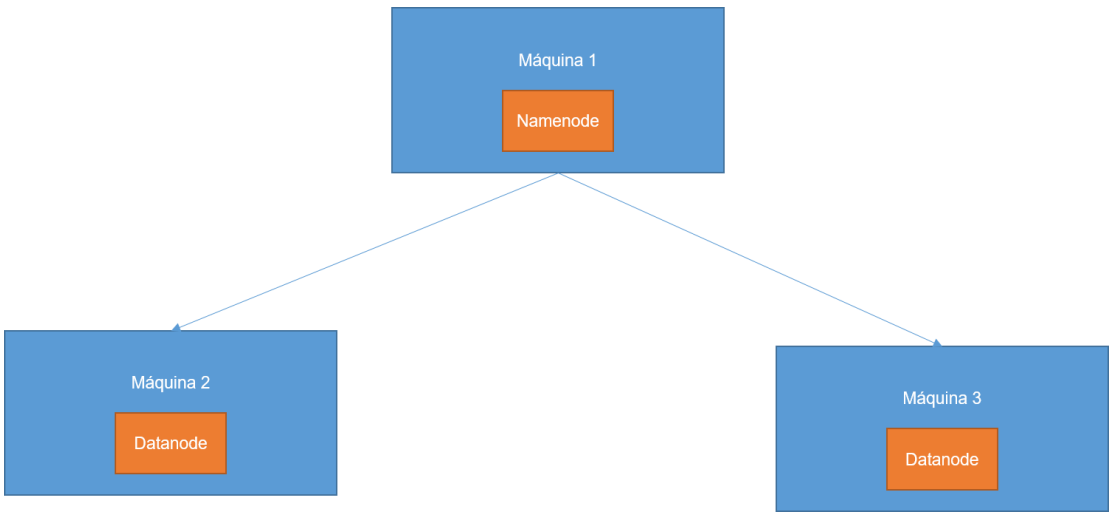


Figura 2.2: Hadoop File System instalado en varios nodos

Una vez configurado el clúster, se hace la última prueba para asegurar que el clúster esta levantado, esta prueba la visualización de los Nodos dentro del WebUI de Hadoop File System.

Una vez configurada y probada, se procede a cargar el código de nuestro programa, en el Namenode, como se verá en la siguiente sección.

2.2 Código Fuente y pruebas

La elaboración del código fuente se llevó a cabo en Java, ya que en este lenguaje trabaja Hadoop, y las librerías que el proyecto usa. Para ejecutar el código fuente se implementó un script en bash, el cual se ejecutará cada día, en un horario en el cual no se hagan modificaciones a los archivos, para que el programa no tenga conflictos con el sistema.El código fuente en sí, trata de modelar un sistema de archivos por capas, en las cuáles las capas son representadas por volúmenes de almacenamiento diferentes. En nuestro proyecto se usan 2 capas, que corresponden a los tipos de almacenamiento disponibles: SSD y Magnético. El fin de estas capas es facilitar la lectura y escritura de los archivos más usados, colocándolos en el SSD, y los menos usados en los volúmenes magnéticos; lo cual se aprecia visualmente en la Figura 2.3.

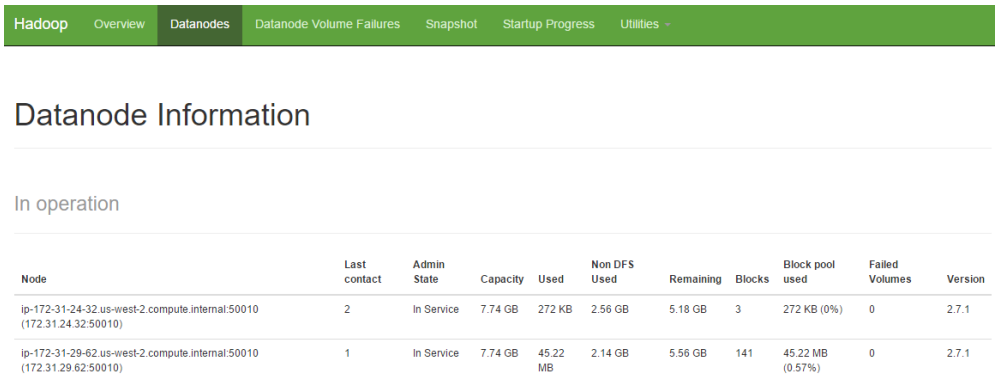


Figura 2.3: WebUI del Sistema 1

Se ha logrado esto usando el conocimiento obtenido del Issue[HDFS-2832] en los foros de Apache Hadoop [5]. Internamente Hadoop File System usa “capas” de almacenamiento, a las cuales mueve los datos que el usuario ordene. Nuestro script automatiza este proceso, calificando y moviendo archivos según su uso, cada día. Los archivos que se hayan usado en ese día tendrán un tag “ALL_SSD”, la cual mueve los archivos y todas sus réplicas a volúmenes con Discos Duros de Estado Sólido.

Los archivos que se hayan usado en esa semana, pero no en el mismo día, tendrán el tag HOT, donde todas las réplicas pasan a los discos magnéticos de mayor capacidad (llamados DISK por el sistema de archivos). Por último, todos los archivos que no hayan sido usados esa semana tendrán como tag COLD, el cuál ordena al archivo y todas sus réplicas en discos magnéticos

diseñados para archivar ficheros de manera comprimida (ARCHIVE para Hadoop).

La implementación de la solución funciona de tal manera que automáticamente, después de un período de tiempo ajustable (del cual se discutirá más adelante) el script seleccione todos los archivos del HDFS y los categorice por su último tiempo de acceso (más no de modificación). Después de esto se va cambiar, según sea necesario, los TAGS de cada uno de los archivos los cuáles necesiten ser cambiados (hayan sido accesado recientemente o no hayan sido accesado en un largo tiempo). Luego que se modifican los tags de los archivos, se procede a llamar a la función MOVER del HDFS, el cual automáticamente, según el tag, mueve los archivos a sus respectivos dispositivos de almacenamiento físico (en caso de ser usados recientemente, a SSD's, si no a HDD's según lo especificado en las políticas). Este proceso se repite mientras el script este activo, después de un período de espera.

Los tiempos de ejecución del script son tomados a partir del posible uso de estos en aplicaciones administrativas, estos procesos son mejor ejecutados cuando no se interpongan con otros procesos críticos del día a día del ambiente.

Después de implementar el código, se procede a implementar las pruebas de rendimiento, para poder visualizar los resultados del programa. Se usa la herramienta de Benchmarking DFSIO, la cual viene incluida en el paquete de hadoop instalado.DFSIO es una prueba de rendimiento dentro del framework

de hadoop, la cual escribe archivos dentro del sistema, y luego trata de leerlos. Todo esto genera un reporte, el cual leerá la velocidad de lectura y escritura del sistema. Precisamente usamos esta herramienta por su efectividad y porque es la mejor herramienta que nos presenta el framework.

CAPÍTULO 3

3. RESULTADOS

3.1 Resultados sin usar el programa

Sin usar el programa creado, solo se creó y se leyó los archivos directamente del disco duro magnético, que es el disco duro por defecto del sistema de archivos, usando la prueba DFSIO WRITE. Los resultados se muestran a continuación, con 10,20,40,60y 100 archivos de 10 MB cada uno. Cabe destacar que ya que se está usando una subscripción económica a Amazon Web Services, la memoria solo permite las operaciones de hasta 100 archivos. Más que esto es demasiado para las memorias RAM de 1GB de las máquinas virtuales. Cabe recalcar que los tiempos de escritura son iguales para ambos casos, por lo cual solo vamos a tomar en cuenta los tiempos de lectura. Los tiempos de ejecución se los puede apreciar en el

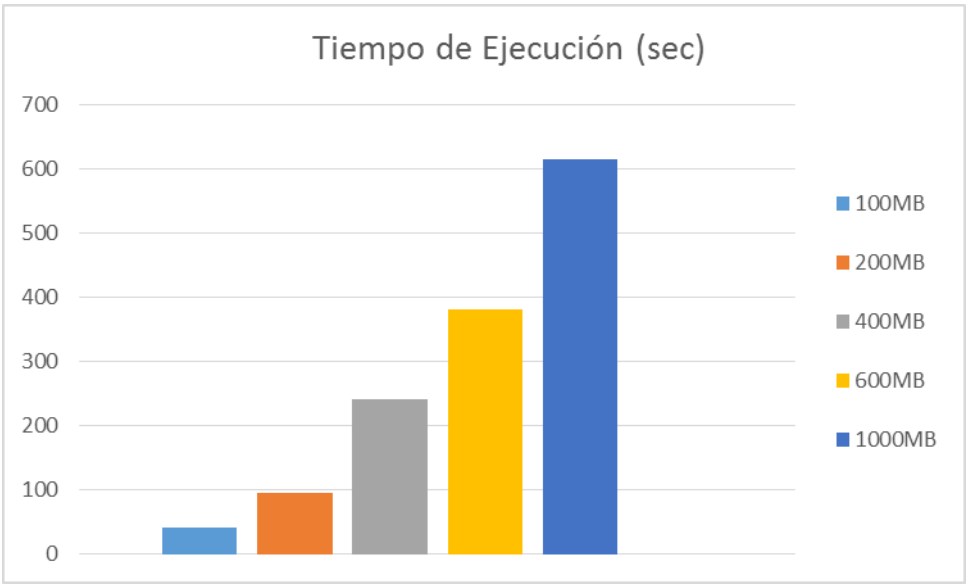


Gráfico 3.1.

3.2 Resultados usando el programa

Después de ejecutar el programa automático y esperar que los archivos sean procesados y movidos a sus capas respectivas (En este caso, el programa movió a todos los archivos a SSD ya que habían sido recientemente usados). Los resultados de la prueba de lectura fueron los siguientes:

Gráfico 3.1: Tiempos de Ejecución de DFSIO Read para

Gráfico 3.2: Tiempos de Ejecución de DFSIO Read para SSD, post-balanceado

Con los gráficos 3.1 y 3.2 mostrados a continuación se puede observar una notable diferencia en tiempos de ejecución para procesos de lectura entre HDD's y SSD's, los cuales aumentan a medida que aumentan el número de

datos. Esto se debe a la mayor velocidad de las SSD’s [6]. Junto con las capacidades de Hadoop de manejar grandes cantidades de datos eficientemente. Se espera que a mayor cantidad de datos (mayor a un 1TB), la diferencia sea aún más considerable.

Sin embargo los HDD siguen siendo más económicos, y para cantidades de datos menores a los cientos de GB, la diferencia es mínima comparada con el precio.

CONCLUSIONES Y RECOMENDACIONES

Conclusiones

1. Se diseñó un programa que automatiza el movimiento de archivos entre capas en Hadoop File System, facilitando con esto al usuario el ordenamiento de sus archivos en el volumen correspondiente a su uso.
2. Gracias a las pruebas se pudo demostrar que hay un cambio de rendimiento positivo en almacenar los archivos más usados dentro de una memoria más rápida dentro de Hadoop File System, ya que almacenar todos dentro de una memoria de acceso rápido puede llegar a resultar costoso, se busca un punto de equilibrio entre precio y eficiencia.

Recomendaciones

1. En futuras revisiones se recomienda usar equipos virtuales de mayor tamaño y rendimiento, ya que con estos se podría tener pruebas a mayor escala que demuestren en forma general nuestros objetivos.
2. En futuras revisiones, después de probar minuciosamente, se aspira a poder subir nuestro programa al foro de desarrollo de Apache Hadoop, en el cual podría llegar a tomar parte de algún próximo release del programa.

BIBLIOGRAFÍA

[1]M. Musatoiu, "An approach for choosing the right distributed file system Microsoft DFS vs. Hadoop DFS", Postgraduate, Faculty of Computing Blekinge Institute of Technology, 2015.

[2]M. Satyanarayanan, "Distributed file systems", *Addison-Wesley and ACM Press*, no. 821, pp. 145-154, 1993.

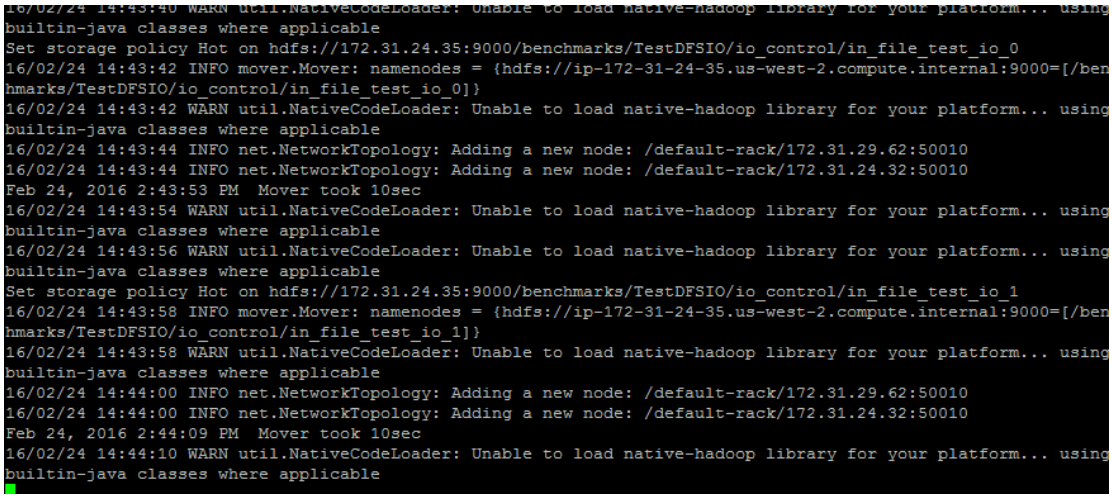
[3]T. White, *Hadoop: The Definitive Guide*, 3rd ed. O'Reilly Media, Inc., 2012.

[4]A. Chaudhary and P. Singh, "BIG DATA – IMPORTANCE OF HADOOP DISTRIBUTED FILESYSTEM", *International Journal of Scientific & Engineering Research*, vol. 4, no. 11, p. 234, 2013.

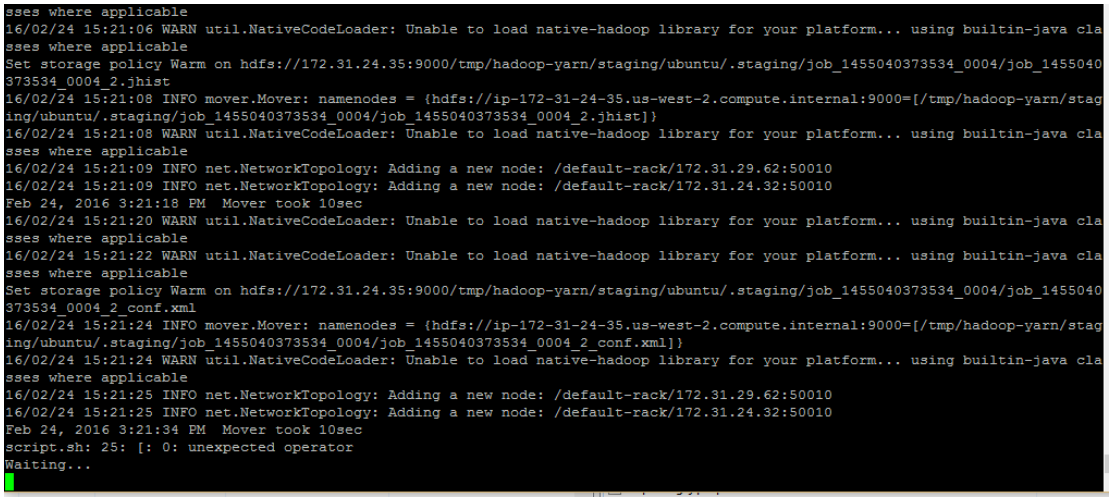
[5]A. Agarwal, "[HDFS-2832] Enable support for heterogeneous storages in HDFS - DN as a collection of storages - ASF JIRA", *Issues.apache.org*, 2013. [Online]. Available: https://issues.apache.org/jira/browse/HDFS-2832. [Accessed: 15- Nov- 2015].

[6] S. Kang and D. Koo, "A Case for Flash Memory SSD in Hadoop Applications",International Journal of Control and Automation, vol. 6, no. 1, 2013.

ANEXOS



Anexo 1: Captura de pantalla de ejecución del programa



Anexo 2: Captura de pantalla de terminación del programa