ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

“Análisis de la información de una Base de Datos Transaccional usando Hive sobre Hadoop”

INFORME DE

MATERIA DE GRADUACIÓN

Previo la obtención del Título de:

INGENIERO EN CIENCIAS COMPUTACIONALES

ESPECIALIZACIÓN SISTEMAS TECNOLÓGICOS

Presentada por:

ALCIVAR GUAIGUA MERCEDES VANESSA

ESPINOZA PICO EDGAR IVÁN

GUAYAQUIL – ECUADOR
2012

DEDICATORIA

A Dios, por haberme permitido llegar hasta este punto y haberme dado salud para lograr mis objetivos, además de su infinita bondad y amor.

A mis padres y mis hermanos porque creyeron en mí y porque me sacaron adelante, dándome ejemplos dignos de superación y entrega, porque en gran parte gracias a ustedes, hoy puedo ver alcanzada mi meta, ya que estuvieron siempre impulsándome en los momentos más difíciles de mi carrera.

Alcívar Guaigua Mercedes Vanessa

DEDICATORIA

A Dios, por enseñarme que el camino de la vida está lleno de metas y que la única forma de sentirse orgulloso de alcanzarlas, es con sacrificio.

A mi madre, por haber entregado la vida entera por sus hijos, por sus sabios y afanosos consejos que me han guiado y me han convertido en un hombre de bien, porque demostró con hechos el amor que una madre tiene por sus hijos, por ser ejemplo de rectitud, valores, responsabilidad y perseverancia y por darme la mejor herencia, mi educación.

A mi hermano, mi mejor amigo y consejero, quien es como un padre para mí y a quien le debo muchas de mis metas.

A mi esposa, por acompañarme en todo momento, por ser mi apoyo y darme su amor y comprensión.

Espinoza Pico Edgar Iván

AGRADECIMIENTO

A todas las personas que participaron e hicieron posible este proyecto, muchas gracias por su apoyo y enseñanza.

A nuestros profesores que compartieron con nosotros sus conocimientos y su amor por la computación, especialmente a nuestra Directora de Proyecto la Ingeniera Vanessa Cedeño que nos brindó todo su apoyo y conocimientos en la realización de este proyecto.

DECLARATORIA EXPRESA

“La responsabilidad del contenido de este Trabajo de Graduación, nos corresponde exclusivamente; y el patrimonio intelectual de la misma, a la **Escuela Superior Politécnica del Litoral**”

(Reglamento de Graduación de la ESPOL)

_____

ALCIVAR GUAIGUA MERCEDES VANESSA

_____

ESPINOZA PICO EDGAR IVAN

TRIBUNAL DE SUSTENTACIÓN

Ing. Vanessa Cedeño

PROFESORA DE LA MATERIA DE GRADUACION

Ing. Guido Caicedo.

PROFESOR DELEGADO DEL DECANO

RESUMEN

En este proyecto se pondrá a prueba el rendimiento de dos plataformas empleadas para el análisis de los datos; un motor de base de datos Microsoft y un data warehouse de código abierto distribuido por Apache Foundation.

Basándonos en una configuración de hardware predefinida para cada plataforma se ejecutarán varias consultas a los registros de una base de datos de 16Gb y se medirán sus tiempos de respuesta. Poniendo a prueba el motor de base de datos relacional SQL Server 2008 y el data warehouse Hive soportado por Hadoop.

El proyecto servirá para demostrar la gran capacidad, escalabilidad y bajo costo que tiene Hadoop en la ejecución de análisis del "Big Data[1]" en comparación con una Base de Dato Relacional tradicional.

[1] Big Data, término utilizado al referirse a grandes volúmenes de datos

INDICE GENERAL

INDICE DE FIGURAS

INDICE DE TABLAS

ABREVIATURAS

HDFS	Hadoop Distributed File System
GB	Gigabytes
JRE	Java Runtime Environment
R2	2do Release (lanzamiento)
x64	Plataforma de 64 bits
FV	Factura de Venta
SSH	Secure SHell, intérprete de órdenes segura)

INTRODUCCIÓN

La presente investigación tiene por objetivo realizar un estudio comparativo de los tiempos de respuesta y el rendimiento entre dos herramientas que permiten el análisis de datos; un motor de base de datos relacional de paga y una infraestructura de data warehousing[2] sobre un ambiente distribuido open source[3],

Como configuración definida tendremos varios servidores de diferentes características de hardware instalados con SQL Server 2008 sobre un entorno Windows y un clúster de 4 nodos con computadores de características básicas configurado con Hive y soportado por la plataforma Apache Hadoop sobre CentOS.

Se exportaran tablas desde una base de datos de 16Gb del servidor SQL Server hacia Hive y se ejecutarán consultas sobre registros de tipo transaccional, para este caso, serán consideradas como válidas solo las facturas, ya que como un adicional al proyecto se presentará un rápido

[2] Data Warehousing. En el contexto de la informática, un almacén de datos (del inglés data warehouse) es una colección de datos orientada a un determinado ámbito (empresa, organización, etc.), integrado, no volátil y variable en el tiempo, que ayuda a la toma de decisiones en la entidad en la que se utiliza.
[3] Open Source, (código abierto) es el término con el que se conoce al software distribuido y desarrollado libremente

análisis estadístico a modo gráfico de las ventas mediante una aplicación web.

Las fuentes bibliográficas provienen de la investigación en páginas de internet especializadas en el tema e información proporcionada en el curso

El contenido de este trabajo se distribuye de la siguiente manera: en el capítulo 1 se describe el problema, antecedentes, objetivos, justificación y alcance del presente trabajo. En el capítulo 2 se presentan los fundamentos teóricos de la herramienta Hadoop, su configuración, sus características, proyectos relacionados e información sobre HIVE. En el capítulo 3 se describe el análisis de la solución. En el capítulo 4 se muestra el diseño y se detalla la implementación de la solución presentada. Finalmente, las pruebas y resultados son mostrados en el capítulo 5.

CAPÍTULO 1

1. ANTECEDENTES Y JUSTIFICACIÓN.

1.1 Antecedentes y Descripción del Problema

Antecedentes

Debido al constante crecimiento de datos que generan las empresas hoy en día, se ha vuelto muy necesaria la búsqueda de nuevas plataformas para almacenar y analizar la información, ambientes que consuman menos recursos, que sean más escalables y que provean una alta disponibilidad; Hadoop es el framework[4] que cumple con estos requisitos, gracias a MapReduce es posible dividir los datos y procesarlos de manera paralela, permitiendo a los analistas plantear preguntas, formular hipótesis y explorar el Big Data de manera sencilla.

Descripción del Problema

Se tiene una base de datos transaccional con información comercial desde el año 2003 hasta la fecha y se necesita mejorar los tiempos de respuesta de ciertos reportes que toman demasiado

[4] La palabra inglesa framework define, en términos generales, un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular, que sirve como referencia para enfrentar y resolver nuevos problemas de índole similar.

tiempo al ser ejecutados, sin que para esta mejora sea necesario incurrir en una inversión económica en adquisición de nuevo hardware y licenciamiento.

1.2 Justificación

Generalmente las primeras versiones de los sistemas comienzan delegando la gestión de los datos a una base de Datos relacional, esto simplifica el tiempo empleado en el desarrollo de la solución tecnológica y funciona bien durante una primera etapa, mientras la carga de requerimientos no sea tan alta. Con los años el nivel de data aumenta considerablemente y cada vez es más difícil su administración y análisis, en este punto la Base de Datos empieza a ser un problema y para poder dar servicio de buena calidad es necesario repotenciar los servidores, adquirir licencias compatibles con el nuevo hardware o en el peor de los casos cambiar toda la infraestructura por una más actual, esto funcionará por un tiempo, pero tarde o temprano el ciclo se repetiría.

Tras reconocer la considerable inversión económica necesaria para mejorar el rendimiento de una Base de Datos Relacional, se vuelve necesario buscar soluciones alternativas que permitan aprovechar los recursos de hardware existentes, es decir, varias

máquinas que en conjunto formen un sistema distribuido lo suficientemente robusto para poder soportar grandes volúmenes de datos.

Una primera idea que se presenta es fragmentar los datos distribuyéndolos entre diferentes Bases de Datos, esto implica complicar sustancialmente la lógica del negocio y su gestión, sin solucionar completamente el problema de la escalabilidad, ya que añadir más capacidad sigue siendo una tarea complicada. Es por ello que se suelen buscar alternativas que simplifiquen la escalabilidad del sistema.

Hadoop es una plataforma que proporciona escalabilidad horizontal, para añadir más capacidad basta con agregar más máquinas al sistema, lo cual se realiza de manera transparente y sin complicaciones.

1.3 Objetivos

- Comparar los tiempos de respuesta de una Base de Datos Relacional como SQL SERVER contra una plataforma que soporta multinodos como HIVE sobre HADOOP, ambas herramientas

configuradas en ambientes de hardware controlados, con especificaciones definidas.

- Comprobar la escalabilidad en hardware del uso de la herramienta HIVE sobre Hadoop para el procesamiento de cantidades masivas de registros.

- Demostrar el bajo costo que requiere el uso de una plataforma Open Source para el manejo de una base de datos comparado a la inversión necesaria para levantar un ambiente con software de paga.

- Demostrar la facilidad para crear consultas usando HiveQL en diversas necesidades de análisis de información.

- Realizar un breve análisis de la información de las consultas que se realizaran en HIVE en forma de gráficos estadísticos.

1.4 Alcance

Utilizar un clúster con al menos 4 nodos formado por equipos con hardware básico y un grupo de servidores con diferentes características de hardware

Instalar una plataforma basada en HIVE sobre HADOOP para el procesamiento de registros de la Base de Datos Relacional levantado en el clúster.

Instalar un entorno base en Windows Server 2008 y un motor de base de datos SQL Server 2008 en los servidores

Enfocarse en el análisis de la información transaccional del área comercial, específicamente las facturas que datan desde el 2003

Desarrollar algunos tipos de consultas utilizando el lengua nativo de HIVE, HiveQL, para poder medir los tiempos de respuesta.

Implementar una aplicación web que utilizando los resultados de HIVE presente la información de forma clara y comprensible con gráficos estadísticos.

CAPÍTULO 2

2. FUNDAMENTOS TEÓRICOS HADOOP.

2.1 HADOOP

Hadoop aparece en el 2002 con Doug Cutting y Mike Cafarella, inspirado por los papers de Google en MapReduce y Google File System, empieza como parte de un manejador de datos de un motor de búsqueda (Nutch[5]). Pero es a mediados del 2006 cuando nace el proyecto Apache Hadoop, desarrollado por Yahoo! y usado por LastFM[6], Facebook y The New York Times.

Entre las principales ventajas tenemos:

- Más rápido que un RDBMS[7] para grandes volúmenes de datos, especialmente datos no organizados.
- Mejor rendimiento que un HPC[8] tradicional, ya que implementa optimizaciones teniendo en cuenta la topología de la red.

[5] Nutch es un robot y motor de búsqueda basado en Lucene. Es parte del proyecto Lucene que a su vez es gestionado por la Apache Software Foundation. Nutch es software libre

[6] Last.FM, es una red social, una radio vía Internet y además un sistema de recomendación de música que construye perfiles y estadísticas sobre gustos musicales, basándose en los datos enviados por los usuarios registrados

[7] RDBMS, es un Sistema Administrador de Bases de Datos Relacionales. RDBMS viene del acrónimo en inglés Relational Data Base Management System

[8] HPC, El campo de computación de alto rendimiento (High performance Computing o HPC en inglés) es una herramienta muy importante en el desarrollo de simulaciones computacionales a problemas complejos

- La pérdida de información no es un problema gracias a su configuración de replicación.
- API[9] de fácil aprendizaje.
- Permite trabajar con lenguajes diferentes a JAVA.

Hadoop es un framework que de manera transparente provee fiabilidad y manejo de grandes volúmenes de datos a las aplicaciones, ya que implementa un paradigma computacional llamado Map/Reduce, donde la aplicación es dividida en varios y pequeños fragmentos de tareas, los cuales son ejecutados o re-ejecutados dentro de algún nodo del clúster. Adicional a esto, cuenta con un sistema de archivos distribuido (HDFS) que almacena los datos en los nodos, proporcionando un gran ancho de banda a través del clúster. Tanto el Map/Reduce como el HDFS están diseñados de tal manera que si un nodo falla, el framework se hace cargo de manera automática.

En resumen, Hadoop es accesible, escalable, robusto y simple; tolerante a fallos y distribuido, capaz de almacenar y administrar cualquier tipo y volumen de datos.

[9] API, interfaz de programación de aplicaciones o API (del inglés application programming interface) es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción

2.1.1. ARQUITECTURA DE HADOOP

Hadoop consiste básicamente en el Hadoop Common, el cual proporciona acceso a los sistemas de archivos soportados por Hadoop. El paquete de software The Hadoop Common contiene los archivos .jar[10] y los scripts necesarios para hacer correr Hadoop. El paquete también proporciona código fuente, documentación, y una sección de contribución que incluye proyectos de la Comunidad Hadoop.

Una funcionalidad clave es que para la programación efectiva de trabajo, cada sistema de archivos debe conocer y proporcionar su ubicación: el nombre del rack (más precisamente, del switch) donde está el nodo trabajador. Las aplicaciones Hadoop pueden usar esta información para ejecutar trabajo en el nodo donde están los datos y, en su defecto, en el mismo rack/switch, reduciendo así el tráfico de red. El sistema de archivos HDFS usa esto cuando replica datos, para intentar conservar copias diferentes de los datos en racks diferentes. El objetivo es reducir el impacto de un corte de energía de rack o de fallo de interruptor de modo que incluso si se producen estos eventos, los datos todavía puedan ser legibles.

[10] Un archivo JAR (por sus siglas en inglés, Java ARchive) es un tipo de archivo que permite ejecutar aplicaciones escritas en lenguaje Java. Las siglas están deliberadamente escogidas para que coincidan con la palabra inglesa "jar" (tarro).

Un clúster típico Hadoop incluye un nodo maestro y múltiples nodos esclavo. El nodo maestro consiste en jobtracker (rastreador de trabajo), tasktracker (rastreador de tareas), namenode (nodo de nombres), y datanode (nodo de datos). Un esclavo o compute node (nodo de cómputo) consisten en un nodo de datos y un rastreador de tareas (Ver figura 2.1). Hadoop requiere tener instalados entre los nodos en el clúster JRE 1.6 o superior, y SSH[11].

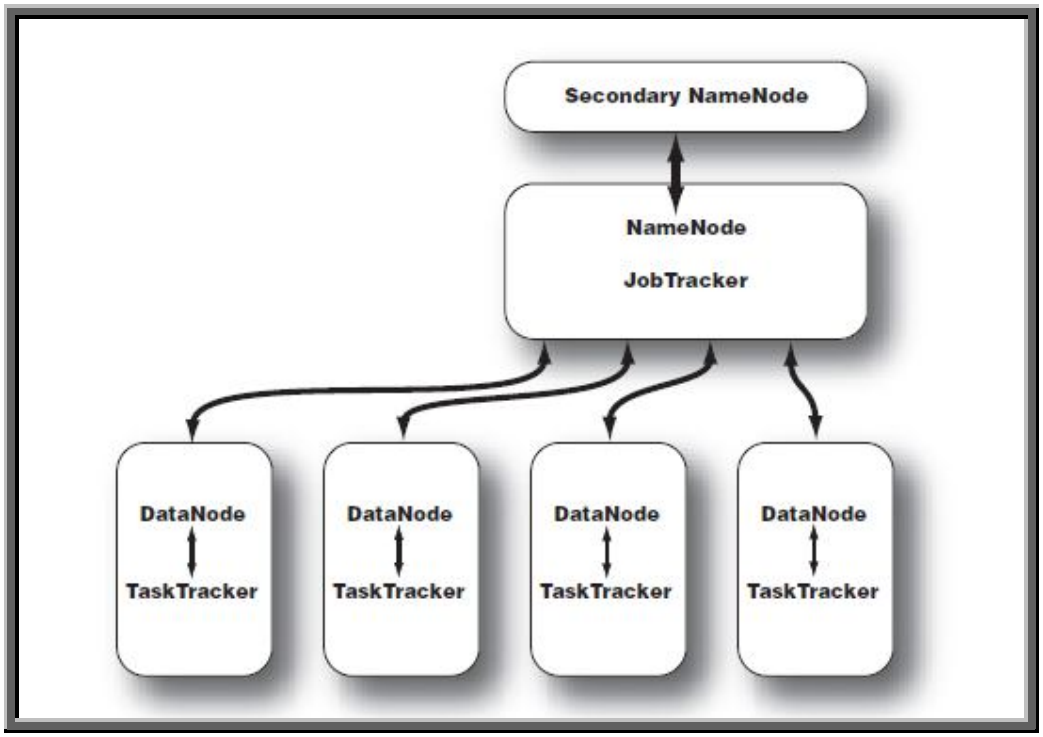


Figura 2-1-Arquitectura Master-Esclavo

[11] SSH, es el nombre de un protocolo y del programa que lo implementa, y sirve para acceder a máquinas remotas a través de una red

2.1.2 Componentes de Hadoop

Hadoop es un framework que ejecuta aplicaciones en grandes clústeres de hardware dedicado. El framework proporciona a las aplicaciones de forma transparente fiabilidad y movilidad de datos. Hadoop implementa un paradigma computacional llamado map/reduce, donde la aplicación se divide en muchos pequeños fragmentos de trabajo, cada uno de los cuales se pueden ejecutar o volver a ejecutar en cualquier nodo del clúster (Ver figura 2.2).

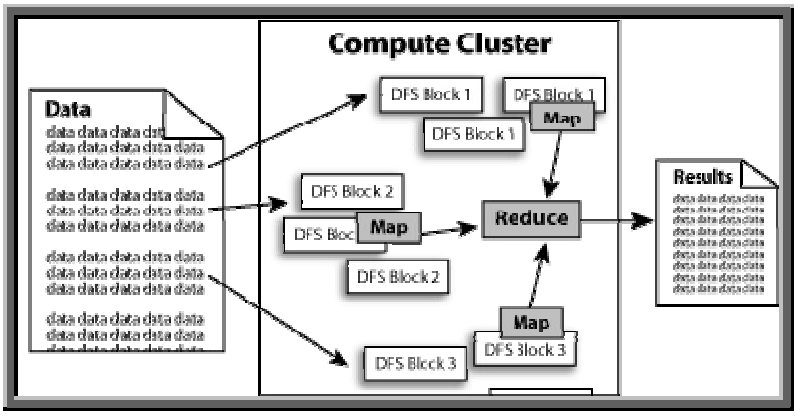


Figura 2-2-Generalidades Hadoop

Además, proporciona un sistema de archivos distribuido que almacena los datos en los nodos de cómputo, produciendo un alto ancho de banda agregado en todo el clúster. Ambos, map/reduce y el sistema de archivos distribuidos, están diseñados de manera que las fallas de nodo se gestionan automáticamente mediante el framework.

2.1.3 HDFS

HDFS (Hadoop Distributed File System) es un sistema de archivos distribuido diseñado para correr en cualquier hardware. Entre sus principales características tenemos:

1. Fallas de hardware, una instancia HDFS podría consistir en cientos o miles de servidores, cada uno almacenando parte de la data del sistema de archivos. El hecho de que exista un gran número de componentes y que cada uno tenga una mínima probabilidad de falla significa que algún componente del HDFS siempre está no-operativo. Por lo tanto, la detección de fallos, y su rápida y automática recuperación forma parte del objetivo de la arquitectura del núcleo de HDFS.

2. Streaming de acceso a datos, HDFS está diseñado para procesamiento de datos tipo batch, en lugar de aplicaciones interactivas. El énfasis está en un alto rendimiento de acceso a los datos y no en accesos de baja latencia.

3. Grandes volúmenes de datos, las aplicaciones que corren sobre HDFS manejan sets de datos del orden de gigabytes a petabytes. Debe proveer un gran ancho de banda de datos y escalar a cientos de nodos en un clúster único, debe soportar decenas de millones de archivos en un única instancia

4. Modelo de coherencia, las aplicaciones HDFS necesitan el modelo de acceso a archivos tipo “escribir una vez – leer muchas veces”, es decir, un archivo una vez creado, escrito y cerrado no necesita ser cambiado, esta asunción simplifica los problemas de la coherencia de los datos y permite un alto rendimiento al acceder a los datos.

5. “Computación en movimiento es más barato que datos en movimiento”, un requerimiento de cálculo de una aplicación es mucho más eficiente si es ejecutado cerca de la data sobre la cual opera, lo cual se confirma cuando el tamaño de la data es bastante considerable. Esto minimiza la congestión en la red e incrementa el rendimiento global del sistema. HDFS provee interfaces para que las aplicaciones puedan migrarse cerca de donde se encuentra alojada la data.

6. Portabilidad a través de hardware heterogéneo y plataformas de software, HDFS ha sido diseñado para ser fácilmente portable de una plataforma a otra, esto facilita la adopción generalizada de HDFS como la mejor opción para un sin número de aplicaciones que manejen cualquier nivel de data.

El Namenode ejecuta operaciones del namespace de archivos del sistema, como apertura, cierre y cambio de nombre de archivos y directorios, también determina la asignación de bloques a los Datanodes (Ver figura 2.3).

Los Datanodes son los responsables de atender las solicitudes de lectura y escritura de los clientes del sistema de archivos. También llevan a cabo la creación de bloques, eliminación, y replicación al recibir la instrucción del Namespace.

Una implementación típica tiene un equipo dedicado que solo ejecuta el software Namenode. Cada una de las otras máquinas en el clúster ejecuta una instancia del software de Datanode.

HDFS está construido utilizando el lenguaje Java, cualquier equipo que soporte Java puede ejecutar el Namenode o el software Datanode.

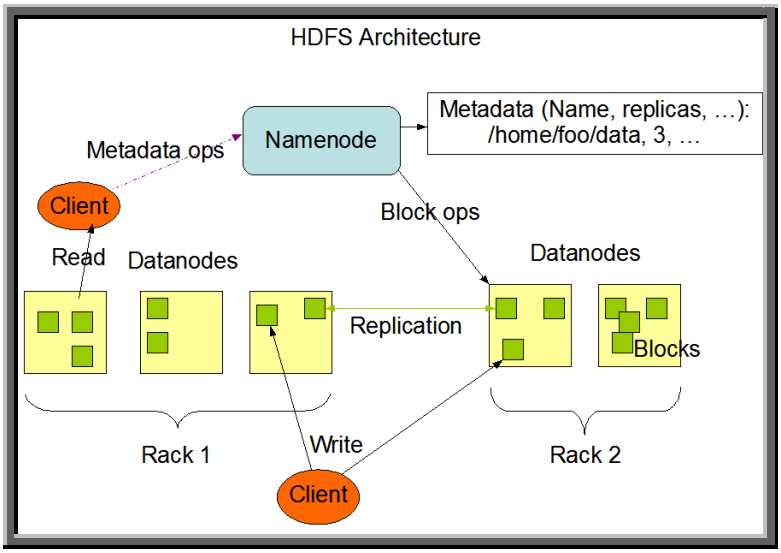


Figura 2-3-HDFS.

Replicación de Datos:

HDFS está diseñado para almacenar de forma confiable archivos muy grandes a través de las máquinas en un gran clúster. Almacena cada archivo como una secuencia de bloques, todos los bloques en un archivo, excepto el último bloque son del mismo tamaño. Los bloques de un archivo se replican para tolerancia a fallos, además el tamaño del bloque y el factor de replicación son

configurables por archivos. Los archivos HDFS tienen estrictamente un escritor en cualquier momento. El NameNode toma todas las decisiones con respecto a la replicación de los bloques, que periódicamente recibe un HeartBeat[12] y blockreport de cada uno de los Datanodes. La recepción de un HeartBeat implica que el DataNode está funcionando correctamente. Un Block report contiene una lista de todos los bloques en un DataNode (Ver figura 2.4).

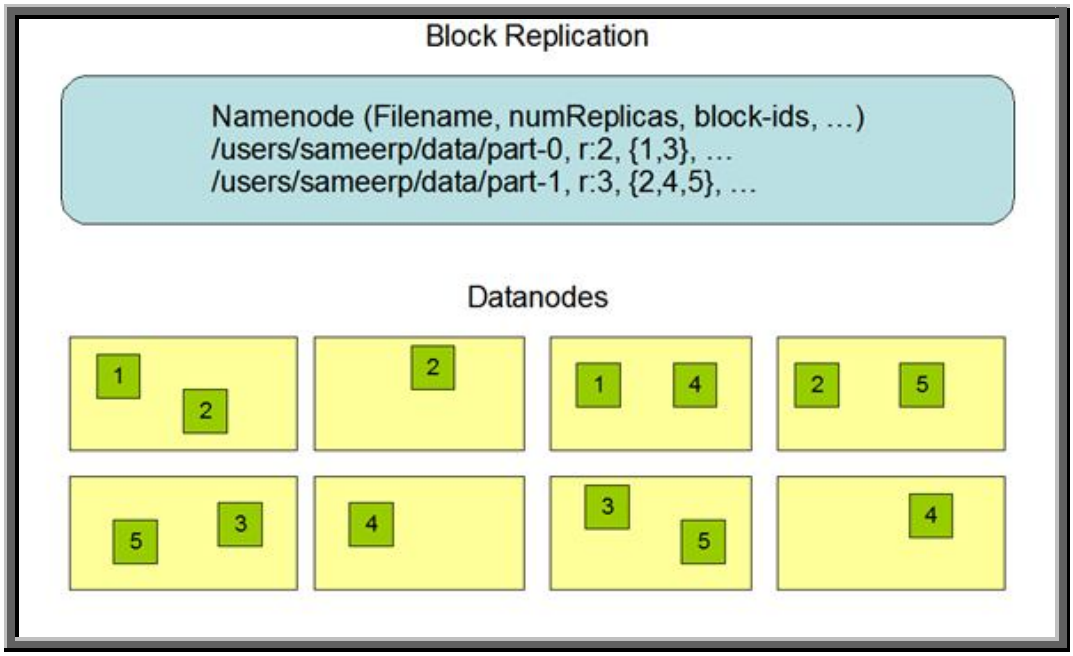


Figura 2-4-Block Replication.

[12] HertBeat, función definida por Ethernet para verificar la calidad de la señal SQE

2.1.4 MAP-REDUCE

En el Proceso MapReduce se tiene un solo maestro que controla la ejecución del trabajo de varios esclavos, los mapeadores se colocan preferentemente en el mismo nodo o rack que su bloque de entrada lo que minimiza el uso de la red, además los mapeadores almacenan las sálidas en el disco local antes de servir a los reductores lo que permite la recuperación si un reductor falla. El proceso MapReduce es el siguiente (Ver figura 2.5)

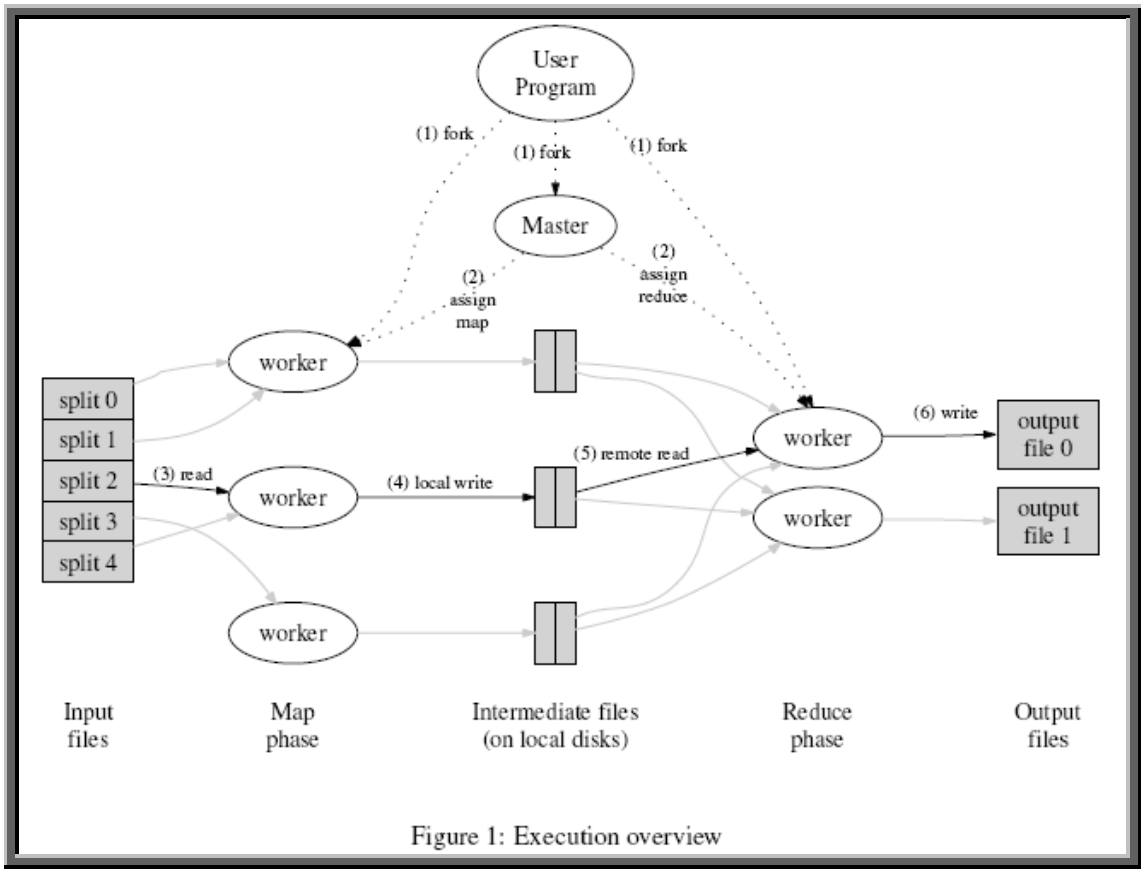


Figura 2-5–Modelo de Programación MapReduce.

- La librería de MapReduce en el programa de usuario divide la entrada de datos en M pedazos de normalmente 64 MB por pieza. Luego varias copias del programa se pondrán en funcionamiento en un clúster de máquinas. Una de las copias del programa es especial, el maestro, el resto son trabajadores que se les asigna trabajo mediante el maestro.

- Hay M tareas Map y R tareas Reduce para asignar, el maestro recoge los trabajadores libres y asigna a cada uno una tarea Map o una tarea reducir. Periódicamente los pares del buffer se escriben en el disco local, dividido en regiones R por la función partición. La ubicación de estos pares almacenados en el disco local se envía de nuevo al maestro que es responsable de enviar estas ubicaciones a los trabajadores reductores.

- Un trabajador que se le asigna una tarea Map lee el contenido de la entrada correspondiente. Analiza pares clave-valor de los datos de entrada y pasa cada par a la función Map. Los pares clave-valor intermedios producidos por la función Map e almacenan en memoria.

- Cuando un trabajador reductor es notificado por el maestro sobre estas ubicaciones, lee todos los datos intermedios y ordena por las claves intermedias de modo que todas las apariciones de la misma clave se agrupan.

- El trabajador reductor para cada clave intermedia única encontrada pasa la clave y el correspondiente conjunto de valores intermedios para la función reducir del usuario. La salida de la función reducir se adjunta a un archivo de salida.

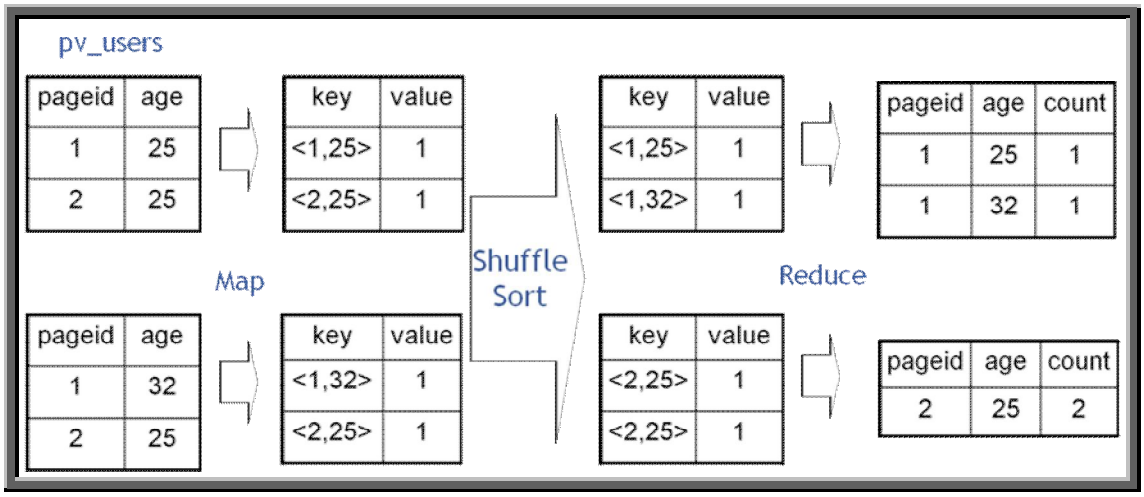


Figura 2-6– Ejemplo MapReduce.

2.2 HIVE

Apache HIVE es una infraestructura de almacenamiento de datos construida sobre Hadoop para proveer resumen de datos, consultas y análisis de grandes volúmenes de datos almacenados en Hadoop o en sistemas de archivos distribuidos compatibles.

Entre sus características tenemos:

- Herramientas que permiten la extracción/transformación/carga de datos de manera fácil
- Mecanismo para imponer estructura sobre una variedad de datos
- Acceso a los archivos almacenados directamente en Apache HDFS o en otros sistemas de almacenamiento de datos, tales como Apache HBase[13]
- Ejecución de consultas haciendo uso de MapReduce

Hive define un lenguaje tipo SQL llamado HiveQL que da soporte a los usuarios familiarizados con SQL para realizar consultas de los datos. Al mismo tiempo, también permite a los programadores que trabajan con el framework de MapReduce el poder conectarse con

[13] HBase es una base de datos open source, no relacional y distribuida escrita en Java. Es desarrollada como parte del proyecto Apache Hadoop y se ejecuta sobre el HDFS, provee capacidad para manejar grandes volúmenes de datos.

sus mappers y reducers de tal manera que puedan realizar un análisis más sofisticado de la información. HiveQL tiene un alcance limitado a consultas tipo insert, select y create table, cuenta con soporte básico para índices, tales como *bitmap index*[14] para acelerar los queries, pero carece de soporte para transacciones y vistas, se interrelaciona con Hadoop a través de un compilador que traduce la sentencia en un *DAG*[15] de los jobs de MapReduce.

Al igual que las bases de datos tradicionales Hive contiene tablas, donde cada tabla se compone de un número de filas y cada fila se compone de un número específico de columnas. Cada columna tiene un tipo asociado, el tipo puede ser int, float o string.

2.2.1 Arquitectura de Hive

Los siguientes componentes son los elementos principales de Hive (Ver figura 2.7):

[14] Bitmap index, es una clase especial de índice de base de datos que utiliza bitmaps, empleado con aplicaciones de data warehousing

[15] DAG (Directed acyclic graph), está formado por una colección de vértices y aristas dirigidas, cada arista conecta un vértice a otro, de tal manera que no hay forma de iniciar en algún vértice y seguir una secuencia de aristas sin que eventualmente se termine en el mismo vértice.

Metastore: El componente que almacena el sistema catálogo y almacena toda la información sobre las tablas, particiones, esquemas, columnas y tipos, etc.

Driver: El componente que gestiona el ciclo de vida de una declaración
HiveQL y además mantiene un identificador de sesión de estadísticas.

Compilador: Los datos almacenados en el Metastore son utilizados por el compilador para generar un plan de ejecución. Primero Analiza para generar la sintaxis abstracta para la consulta, luego obtiene la información de todas las tablas, verifica los tipos y realiza el análisis semántico, después realiza la optimización generando una serie de tareas Map-Reduce y tareas HDFS.

Motor de Ejecución: El componente que ejecuta ordenadamente las tareas producidas por el compilador según su dependencia, cada tarea dependiente solo se ejecuta si todos sus requisitos han sido ejecutados interactuando con Hadoop.

Hive Server: Proporciona la integración de Hive con otras aplicaciones.

Clientes: Tales como la interface de líneas de comandos, la interfaz de usuario web y el Driver JDBC/ODBC.

.

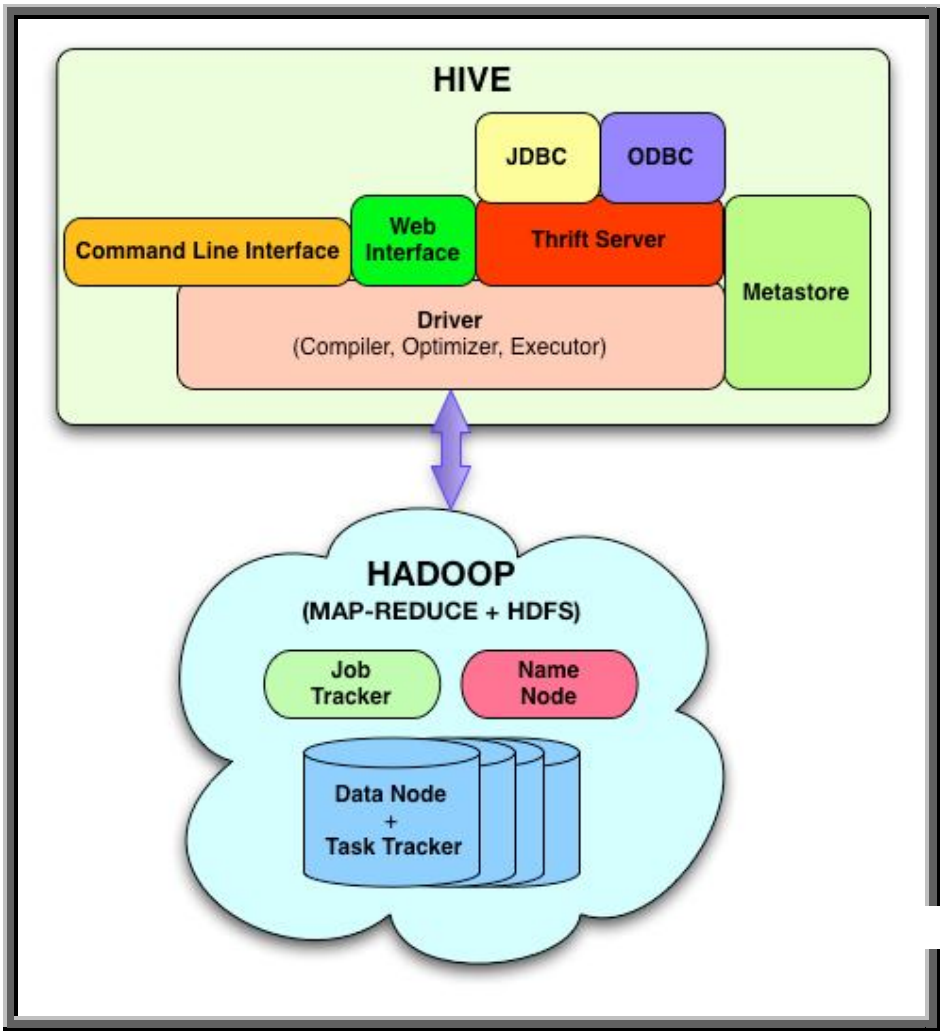


Figura 2-7– Arquitectura Hive.

CAPÍTULO 3

3. ANÁLISIS DE LA SOLUCIÓN.

Debido al constante crecimiento de la base de datos transaccional de la empresa analizada y a los altos tiempos de respuesta en la generación de reportes, se requiere usar una herramienta que permita el almacenamiento y análisis de datos masivo a bajo costo y con menor tiempo de respuesta. A continuación detallamos la infraestructura actual (Ver Figura 3.1.).

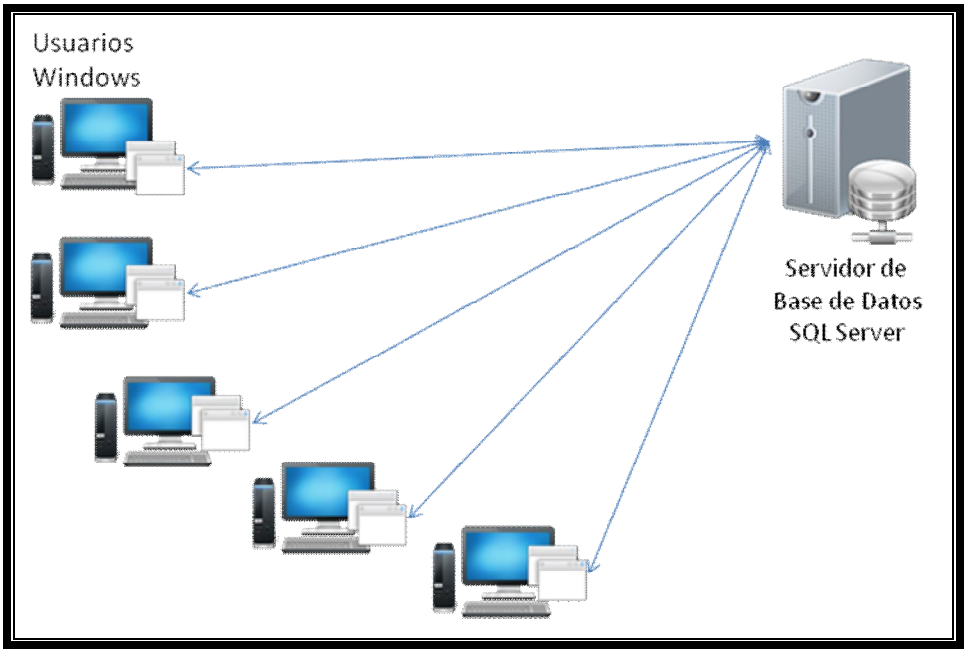


Figura 3-1– Arquitectura del Sistema Actual.

3.1 Requerimientos.

La solución debe ser Open Source y tener como principal característica el soporte para el almacenamiento de datos masivo a bajo costo. Se definen las características principales que debe cumplir esta herramienta.

3.1.1 Requerimientos Funcionales.

- La Herramienta escogida debe permitir la consulta y carga de grandes cantidades de información, al momento una base de datos de 16Gb con múltiples tablas y registros que a futuro puede crecer.

- Debe brindar alguna forma de importar la información desde la base de datos relacional.

- El tiempo de respuesta de las consultas realizadas debe ser menor en comparación al obtenido en los servidores de base de datos relacional.

- El resultado de las consultas realizadas debe generar un archivo plano, con los datos en un formato legible, que pueda ser leído por la aplicación web que genera los gráficos estadísticos.

3.1.2 Requisitos No Funcionales.

Escalabilidad

La posibilidad de una infraestructura que pudiera escalar junto con el crecimiento de los datos de forma sencilla.

Velocidad

Es decir poder aumentar la velocidad de la extracción de la información y lograr hacer consultas que se ejecuten en el menor tiempo posible.

Economía

Con las bases de Datos tradicionales a medida que el tiempo transcurre se hace cada vez más costoso su mantenimiento, por lo tanto la herramienta debe ser lo más económico a pesar de los años.

3.2 Capacidades de la Herramienta HIVE para el Análisis de Bases de Datos Transaccional.

Apache Hive se encuentra levantado sobre un sistema basado en Hadoop, el cual es el encargado de llevar el trabajo a donde residen los datos, de tal manera que se pueda hacer un uso eficiente de los discos

locales en los múltiples nodos, además esta soportado por HDFS, un sistema de archivos distribuido, resistente y escalable.

Hive brinda soporte para tablas (a diferencia de las bases de datos NoSQL[16]) y admite consultas expresadas en un lenguaje tipo SQL. Está pensado para manipular enormes cantidades de información de manera muy rápida y está preparado para escalar horizontalmente sin perder rendimiento, suele funcionar bastante bien en hardware de bajo costo (PCs normales y corrientes), y permite el escalado horizontal añadiendo nuevas máquinas en caliente (idealmente sin necesidad de reinicio del sistema).

Y si agregamos a estas características el hecho de ser una herramienta Open Source podemos concluir que esta es la mejor alternativa para ser aplicada en nuestro proyecto.

[16] NoSQL (not only sql) es un término usado en informática para agrupar una serie de almacenes de datos no relacionales que no proporcionan garantías ACID. Normalmente no tienen esquemas fijos. En bases de datos se denomina ACID a un conjunto de características necesarias para que una serie de instrucciones puedan ser consideradas como una transacción

CAPÍTULO 4

4 DISEÑO E IMPLEMENTACIÓN DEL ANALISIS DE LA INFORMACION DE UNA BASE DE DATOS TRANSACCIONAL USANDO HIVE.

4.1 Diseño del Análisis de la información de una Base de Datos transaccional usando Hive sobre Hadoop.

En esta sección se explicará todo lo referente al diseño de nuestra solución para realizar el análisis de la información de la Base de Datos Transaccional usando Hive, además Se describe el diseño General utilizando el paradigma MAP-REDUCE sobre Hadoop.

4.1.1 Diseño General

A continuación presentamos los principales puntos de diseño (Ver figura 4.1):

Dataset

Son los archivos de entrada TXT (input) y los resultados del proceso MapReduce (output).

Clúster

Son los recursos computacionales necesarios para correr nuestra solución. Se implementa la computación paralela dada por Hadoop donde los Map y los Reduce se ejecutan.

Request

Cuando se ejecutan las consultas solicitadas.

Response

Es la respuesta que el clúster provee a la aplicación. En el response vienen los datos necesarios para generar un archivo los cuales serán utilizados para generar gráficos estadísticos.

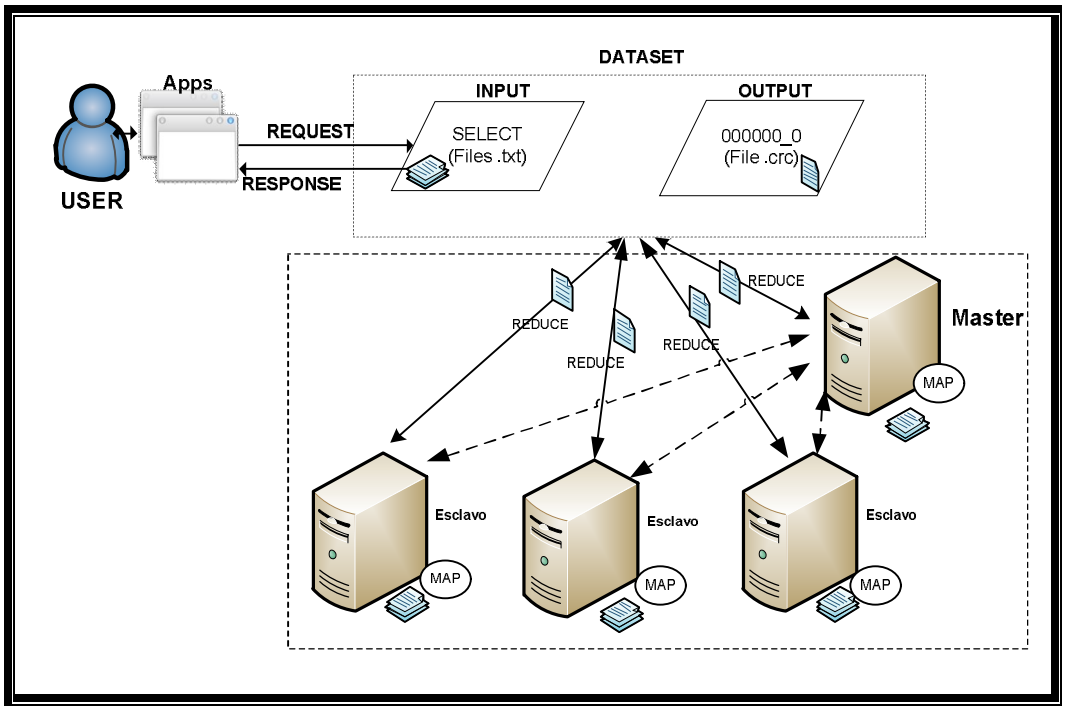


Figura 4-1– Diseño de la Solución.

4.1.2 Diseño MapReduce Implementado por Hadoop.

El proceso MapReduce se inicia con la consulta escrita por el usuario en Hive, entonces busca las coincidencias y retorna un archivo formado por el resultado del Queries. El diseño general se muestra en el gráfico (Ver figura 4.2):

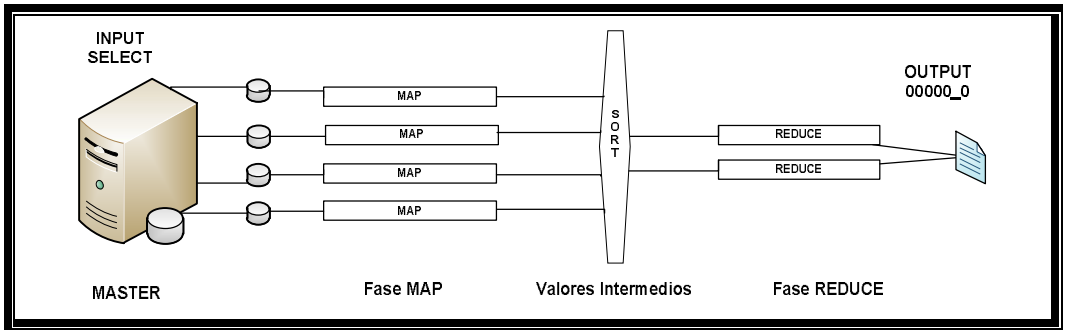


Figura 4-2– Proceso Map-Reduce.

Fase Map

El map de la solución trabaja sobre tres parámetros: las tablas, los campos elegidos, los campos de unión de las tablas si tomamos como ejemplo un join. Internamente la función Map toma cada tabla que se seleccionó y en cada una realiza una búsqueda de los campos solicitados por el usuario en el contenido de la consulta, generando de esta forma pares <clave, valor>, en donde la clave corresponde al campo que se repite en las tablas del join y el valor contiene los campos del resultado del select ingresado.

Fase Reduce

Cada función reduce se encarga de concatenar todas las ocurrencias encontradas en la misma tabla. Generando así los pares <Tabla, ocurrencia | ocurrencia2 |……>.

4.2 Modelamiento de Datos para su Procesamiento

Para poder realizar nuestra solución debemos realizar un proceso de extracción de la información de las tablas de la Base de Datos hacia archivos textos (Ver figura 4.3).

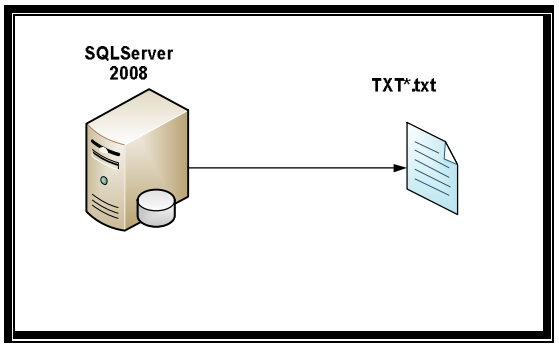


Figura 4-3– Conversión de Archivos de Entrada.

4.3 Plan de Pruebas

Para las pruebas que se realizaran en el sistema tomaremos en consideración dos factores importantes como son el número de nodos

que conforman el clúster y los tiempos de respuesta resultantes de acuerdo al nivel de complejidad de los queries.

Para poder probar la solución se debe tener previamente los queries para las consultas, y tener cargada la información en Hive, luego ejecutar los queries con 2, 3 y 4 nodos midiendo el tiempo de cada ejecución.

Una vez que se cuente con los resultados de ambas herramientas se verificará con cuántos nodos los tiempos de respuesta se los podría considerar mejores que los obtenidos con SQL Server en sus distintas configuraciones.

4.4 Implementación del Análisis de la información de una Base de Datos transaccional usando Hive sobre Hadoop

Para realizar el análisis de la información de la Base de Datos transaccional que usamos en este proyecto hemos dividido el proceso en 6 fases:

a) Instalación del Sistema Operativo, CentOS[17].

[17] CentOS (Community ENTerprise Operating System) es una bifurcación a nivel binario de la distribución Linux Red Hat Enterprise Linux RHEL, compilado por voluntarios a partir del código fuente liberado por Red Hat.

b) Configuración de Apache Hadoop y Hive.

c) Exportación de las tablas involucradas en los queries desde la base de datos origen, SQL Server 2008, hacia archivos planos

d) Importación a HIVE de las tablas en formato .txt de la base de datos origen.

e) Ejecutar los queries requeridos en el ambiente de pruebas.

f) Presentar en forma gráfica el resultado de los queries.

A continuación desarrollaremos cada uno de los pasos:

- **Instalación del Sistema Operativo**

Dado que la herramienta elegida para el proyecto necesita funcionar sobre un ambiente Linux, es necesario instalar el sistema operativo CentOS 6.0 y crear el usuario Hadoop (en cada una de los nodos/máquinas), el cual nos servirá para trabajar con el clúster (Ver figura 4.4).

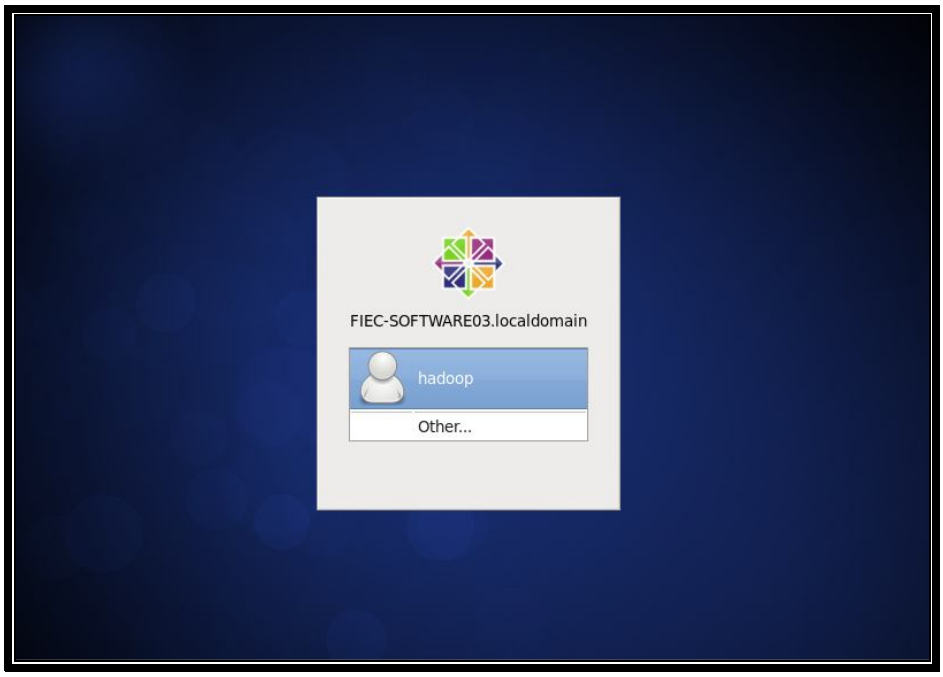


Figura 4-4– Sistema Operativo CentOS 6.0 configurado usuario hadoop.

- **Configuración de Hadoop y Hive.**

Una vez iniciada el CentOS ingresamos con el usuario Hadoop, y luego de haber instalado Java, configuramos Hadoop y Hive, para esto extraemos en el home del usuario los archivos hadoop-0.20.203.tar.gz y hive-0.7.0.tar.gz, los cuales pueden ser descargados de http://hadoop.apache.org/common/releases.html y http://hive.apache.org/releases.html, respectivamente

En todas las máquinas del clúster realizamos las siguientes configuraciones necesarias:

1. En el archivo Hadoop-env.sh colocar el PATH del JDK

```
JAVA_HOME= /usr/lib/jvm/java-1.6.0-openjdk-1.6.0.0/jre
```

2. En el archivo Hdfs-site.xml las rutas para la información del Namenode y el Datanode.

```
  <property>
       <name>dfs.replication</name>
       <value>1</value>
   </property>
  <property>
       <name>dfs.name.dir</name>
       <value>/home/hadoop/dfs/name</value>
   </property>
   <property>
       <name>dfs.data.dir</name>
       <value>/home/hadoop/dfs/data</value>
 </property>
```

3. En el archivo Core-site.xml el nombre del nodo Master

```
   <property>
       <name>fs.default.name</name>
       <value>hdfs://FIEC-SOFTWARE00:8020</value>
   </property>
```

4. En el archivo Mapred-site.xml el nombre de la máquina Jobtracker que en este caso será la máquina master, y el directorio.

```
<property>
     <name>mapred.job.tracker</name>
     <value>FIEC-SOFTWARE00:8021</value>
</property>
<property>
     <name>mapred.system.dir</name>
```

```
        <value>/home/hadoop/mapred/system</value>
    </property>
```

Además en la máquina Master en el archivo masters colocamos el nombre de la máquina master, y en el archivo slaves colocamos todos los nombres de las máquinas esclavos, en las máquinas esclavos estos archivos deben estar vacíos.

```
MASTERS
FIEC-SOFTWARE00
```

```
SLAVES
FIEC-SOFTWARE01
FIEC-SOFTWARE02
```

Todas las máquinas del clúster deben tener acceso automático entre sí y consigo misma a través del ssh con el usuario Hadoop sin necesidad de ingresar la clave, para esto primero en cada máquina ingresamos las IP y los nombres de todas las máquinas del clúster en el archivo /etc/hosts, luego realizamos los siguientes pasos en la consola:

```
ssh-keygen -t dsa -P '' -f ~/.ssh/id_dsa
cd
cd .ssh
scp id_dsa.pub hadoop@fiec-software02:~/
ssh hadoop@fiec-software02
cat id_dsa.pub >>.ssh/authorized_keys
chmod 700 ~/.ssh
chmod 600 ~/.ssh/authorized_keys
```

Luego de terminar la configuración, ya estamos listos para empezar a trabajar con Hadoop, nos colocamos en la carpeta donde instalamos Hadoop en la máquina master y realizamos el format al Namenode con el siguiente comando:

```
bin/hadoop namenode -format
```

Observamos que se crean los directorios /home/hadoop/dfs/name en el master y /home/hadoop/dfs/data en los esclavos (Ver figura 4.5).

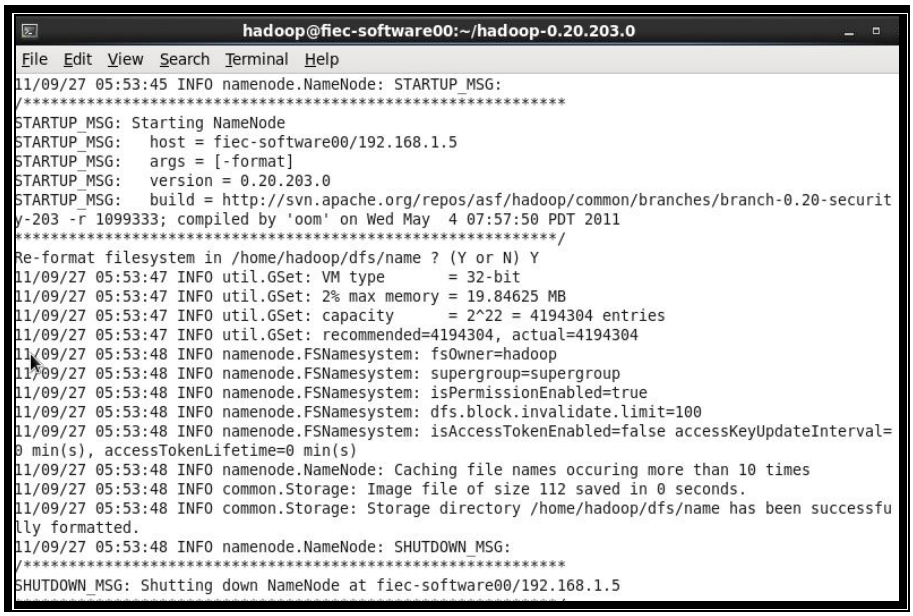


Figura 4-5– Resultado del Format al Namenode.

Luego levantamos los servicios del HDFS en la máquina master con el siguiente comando:

```
bin/start-dfs.sh
```

Debe aparecer el siguiente log :

```
[hadoop@fiec-software00 hadoop-0.20.203.0]$ bin/start-dfs.sh
starting namenode, logging to /home/hadoop/hadoop-0.20.203.0/bin/../logs/hadoop-hadoop-namenode-f
iec-software00.out
FIEC-SOFTWARE02: starting datanode, logging to /home/hadoop/hadoop-0.20.203.0/bin/../logs/hadoop-
hadoop-datanode-fiec-software02.out
FIEC-SOFTWARE01: starting datanode, logging to /home/hadoop/hadoop-0.20.203.0/bin/../logs/hadoop-
hadoop-datanode-fiec-software01.out
FIEC-SOFTWARE00: starting secondarynamenode, logging to /home/hadoop/hadoop-0.20.203.0/bin/../log
s/hadoop-hadoop-secondarynamenode-fiec-software00.out
[hadoop@fiec-software00 hadoop-0.20.203.0]$
```

Figura 4-6– Iniciando el HDFS.

Y levantamos el servicio del MapRed para iniciar el jobtracker en el master y el tasktracker en los nodos con el siguiente comando:

```
bin/start-mapred.sh
```

Debe aparecer el siguiente log:

```
[hadoop@fiec-software00 hadoop-0.20.203.0]$ bin/start-mapred.sh
starting jobtracker, logging to /home/hadoop/hadoop-0.20.203.0/bin/../logs/hadoop-hadoop-jobtrack
er-fiec-software00.out
FIEC-SOFTWARE01: starting tasktracker, logging to /home/hadoop/hadoop-0.20.203.0/bin/../logs/hado
op-hadoop-tasktracker-fiec-software01.out
FIEC-SOFTWARE02: starting tasktracker, logging to /home/hadoop/hadoop-0.20.203.0/bin/../logs/hado
op-hadoop-tasktracker-fiec-software02.out
[hadoop@fiec-software00 hadoop-0.20.203.0]$
```

Figura 4-7– Iniciando MapRed.

Si se necesitan detener los servicios sustituimos el START por STOP en los comandos anteriores.

Es importante recordar que Hadoop proporciona varias interfaces web en las que se pueden consultar el estado de los nodos y el avance de la ejecución de las consultas.

Para ver si se levantaron correctamente los nodos: http://<localhost>:50030/jobtracker.jsp (Ver figura 4.8).

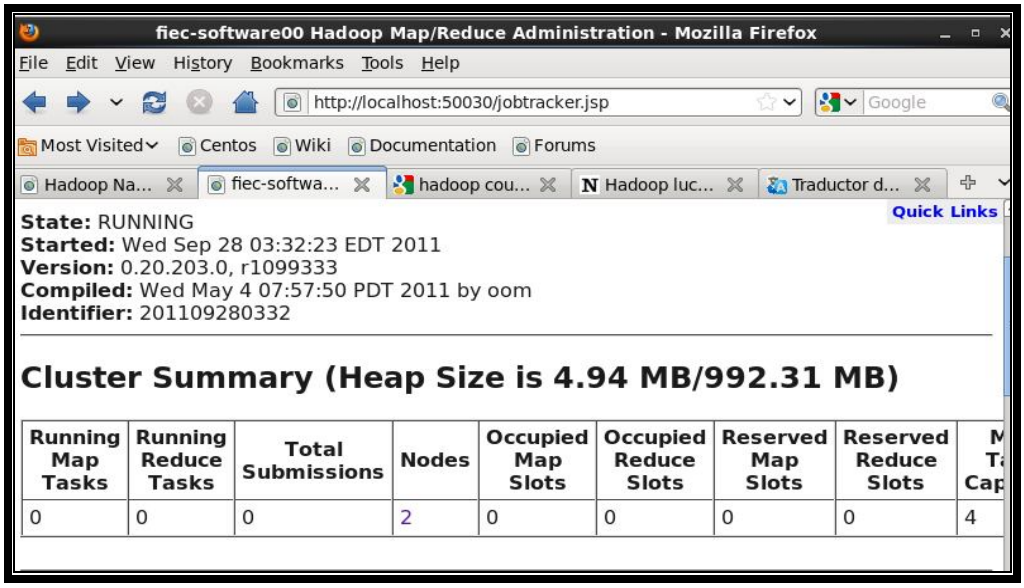


Figura 4-8– Interface Web para ver los estados del Job.

Para verificar el estado del sistema de archivos: http://<localhost>:50070/dfshealth.jsp (Ver figura 4.9).

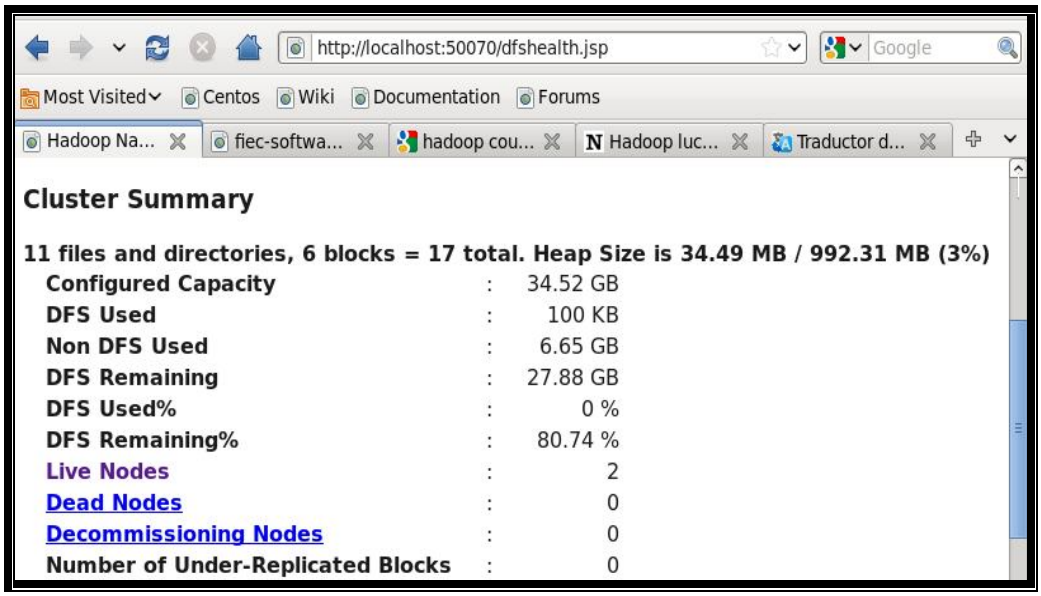


Figura 4-9– Interface Web para ver la memoria del clúster.

Luego de verificar que está corriendo correctamente nuestro clúster, procedemos a levantar Hive, primero realizamos las configuraciones iniciales desde el directorio de Hive con los siguientes comandos en la consola:

```
export HIVE_HOME=/home/hadoop/hive-0.7.0    (path hive)
export PATH=$HIVE_HOME/bin:$PATH
export         HADOOP_HOME=/home/hadoop/hadoop-0.20.203.0
(PATH hadoop)
Crear las carpetas en el HDFS para HIVE
  -   $HADOOP_HOME/bin/hadoop fs -mkdir /tmp
  -              $HADOOP_HOME/bin/hadoop     fs     -mkdir
/user/hive/warehouse
Dar permisos a las carpetas de HIVE
  -  $HADOOP_HOME/bin/hadoop fs -chmod g+w /tmp
  -        $HADOOP_HOME/bin/hadoop    fs    -chmod    g+w
/user/hive/warehouse
```

Procedemos a levantar Hive y ya estamos listos para empezar a trabajar con Hive:

```
export HADOOP_HOME=/home/hadoop/hadoop-0.20.203.
export HIVE_HOME=/home/hadoop/hive-0.7.0
$HIVE_HOME/bin/hive
```

- **Exportación de las tablas de la Base de Datos SQL Server a TXT.**

  Ingresamos al SQL Server 2008, elegimos la base datos de la que vamos a extraer la información y damos clic derecho en Tasks → Export Data (Ver Figura 4.10)

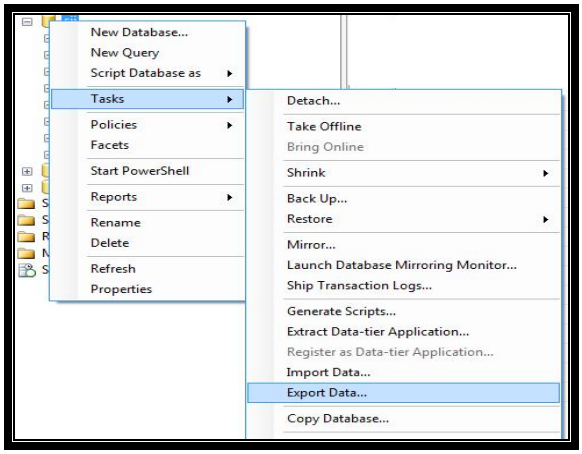


Figura 4-10-Extracción de los datos de SQL Server a TXT.

Se debe ingresar el origen de los datos, servidor y base de datos, acompañado con la autenticación respectiva (Ver Figura 4.11)

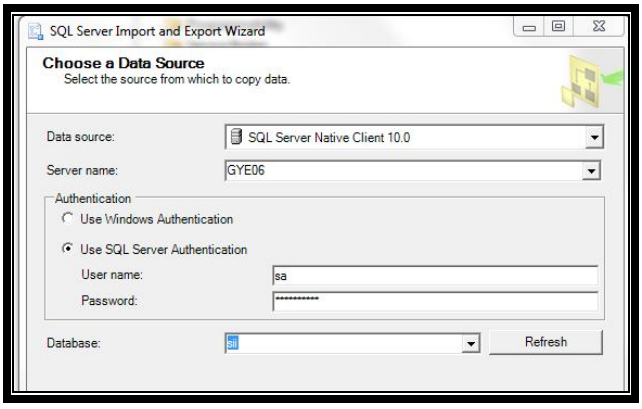


Figura 4-11-Selección de origen de datos y autenticación.

Paso seguido se define como destino el archivo de texto y la ubicación donde se lo va a guardar (Ver Figura 4.12)

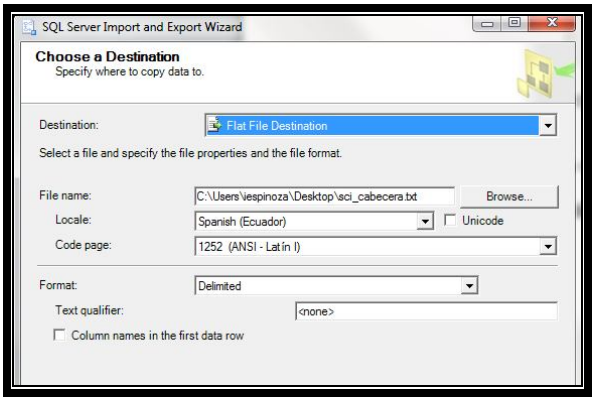


Figura 4-12-Selección de destino, nombre y ubicación del txt.

Verificamos el resultado de la operación donde se muestra el número de registros exportados y el estado de cada etapa de exportacion.

Se ejecuta el mismo proceso de exportación con las 4 tablas restantes, requeridas para poder realizar los queries.

- **Replicar la información para los análisis en HIVE.**

  Para poder importar las tablas exportadas desde la base de datos origen, es necesario crearlas en Hive, debemos considerar un detalle muy importante, el separador que va a tener cada campo, en este caso, la coma (ver Anexo A):

```
CREATE TABLE sci_kardex (
cod_empresa int,
cod_local string,
nom_inventario string,
val_precio_distribuidor int,
por_desc_politica int,
cod_politica_descto string)
  ROW FORMAT DELIMITED
    FIELDS TERMINATED BY ','
 STORED AS TEXTFILE;
```

Luego de copiar las tablas en formato texto al path /bases/, procedemos a cargar la información a Hive con el siguiente comando:

```
hive> LOAD DATA LOCAL INPATH './bases/sci_kardex.txt'
      OVERWRITE INTO TABLE sci_kardex;
```

Hive enviará un mensaje indicando si la carga se ha realizado con éxito (Ver figura 4.13).

```
hive> LOAD DATA LOCAL INPATH './bases/sci_kardex.txt'
    >       OVERWRITE INTO TABLE sci_kardex;
Copying data from file:/home/hadoop/hive-0.7.0/bases/sci_kardex.txt
Copying file: file:/home/hadoop/hive-0.7.0/bases/sci_kardex.txt
Loading data to table default.sci_kardex
OK
Time taken: 445.94 seconds
```

Figura 4-13– Carga de Datos de Tabla a Hive.

- **Realizar los Queries requeridos.**

  Se ingresan los queries en Hive indicando el directorio en el que queremos el resultado del query (Ver Anexo B):

  La ejecución de cada query dará como resultado un archivo llamado 00000_0 en el directorio indicado.

- **Presentar en forma gráfica el resultado de los queries.**

  A fin de presentar una funcionalidad adicional a la herramienta analizada en este proyecto, se realizó la implementación de una pequeña aplicación web, desarrollada con PHP, que presenta un gráfico estadístico de los archivos generados en las pruebas.

  La aplicación tiene como función principal el leer los archivos resultantes de los queries ejecutados en el ambiente Hive y

generar un archivo XML que es utilizado por AmCharts[18] para generar un gráfico estadístico.

La interfaz es muy sencilla, consta de un componente HTML tipo “file” y un botón para subir el archivo al servidor (Ver figura 4.14).

Figura 4-14– Interface para generación de gráficos.

Elegimos el archivo, en este caso el correspondiente al Query de ventas, y lo subimos al servidor web (Ver figura 4.15)

[18] AmCharts herramienta Open Source que comprende un conjunto de gráficos desarrollados con Javascript (HTML5) y flash, soporta datos extraídos de archivos CSV o XML o generados dinámicamente con PHP, .NET, Java, Ruby, Perl, ColdFusion, entre otros

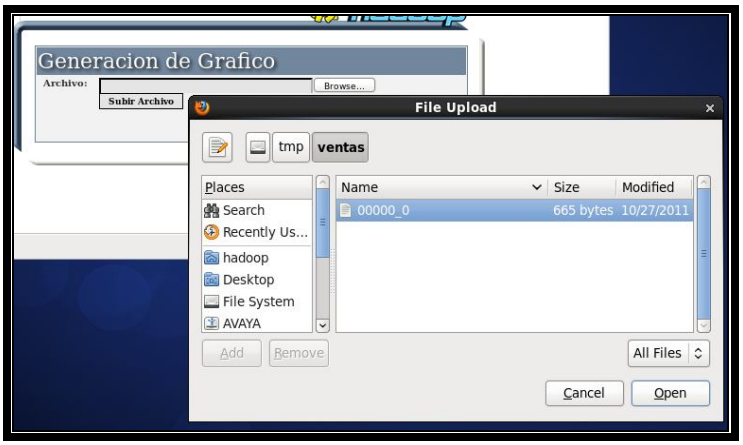


Figura 4-15– Elegir el archivo generado por Hive luego de ejecutar el query.

Una vez subido el archivo presentamos los registros ordenados a modo de tabla para la revisión previa antes de generar el archivo XML. Se presenta como opción dos tipos de gráficos, de barras y de líneas (Ver figura 4.16)

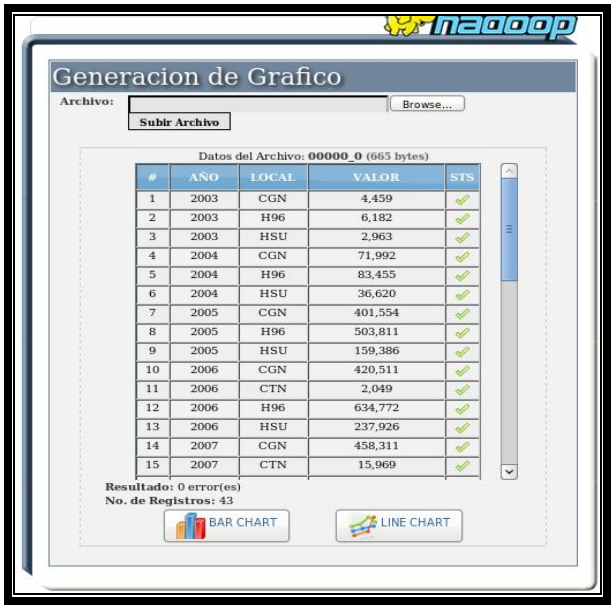


Figura 4-16– Visualización de los datos del archivo subido en el servidor.

Para el ejemplo elegimos el gráfico de barras, la aplicación procede a generar el archivo XML, necesario para crear el gráfico, (Ver figura 4.17)

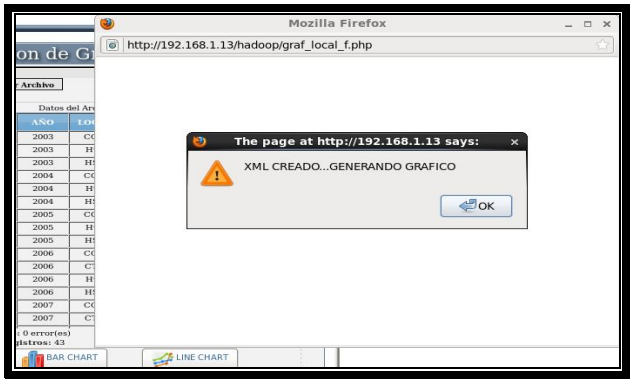


Figura 4-17– Generando el archivo XML.

Con el archivo XML ya tenemos listo el gráfico estadístico (Ver figura 4.18)

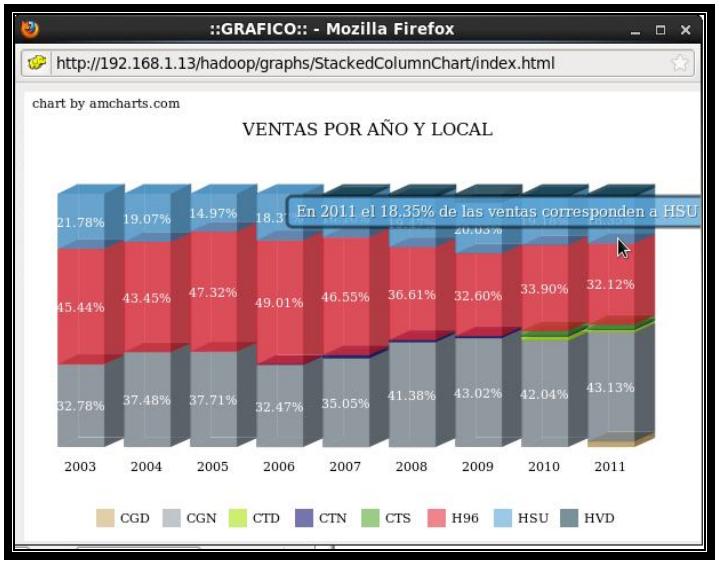


Figura 4-18– Gráfico de barras.

Lo mismo ocurre para el tipo de gráfico de líneas (Ver figura 4.19)

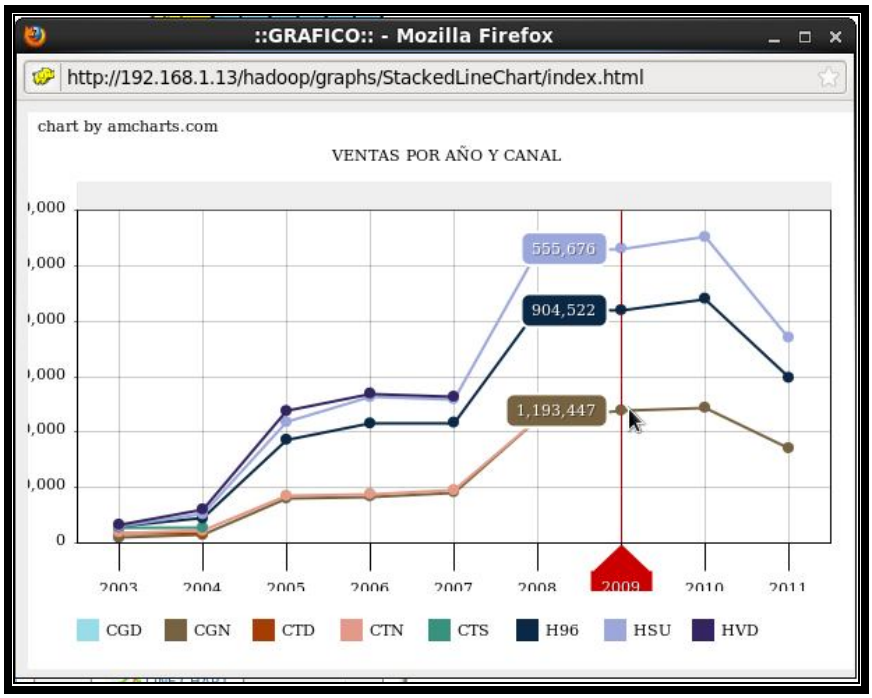


Figura 4-19– Gráfico de líneas.

4.4.1 Herramientas a utilizarse

Se eligieron las herramientas seleccionadas porque nos permitían cumplir con los objetivos impuestos y por su capacidad de adaptarse a futuros cambios.

Como motor de base de datos relacional se utilizó a Microsoft SQL Server Standard 2008 R2 levantado sobre un Windows Server 2008 Standard R2, que es el servidor de producción de la empresa cuyos datos están siendo utilizados para el presente estudio.

El sistema Operativo para cada uno de los nodos fue CentOS 6.0, basado en Red Hat Enterprise Linux (RHEL), el cual es una plataforma Open Source de clase empresarial, sumamente estable y eficiente.

Microsoft Visio 2010, programa que sirvió para aplicar ingeniería inversa a la base de datos de SQL Server, obteniendo el diagrama entidad relación.

Para la ejecución de procesos de tipo MapReduce se utilizó la versión 2.3 de Apache Hadoop, que implementa dicho paradigma y es un marco de trabajo para procesamiento en arquitecturas distribuidas Open Source, esta herramienta nos proporciona la escalabilidad deseada.

Para la duplicación de la base de datos se instaló el data warehouse Apache Hive en su versión 0.7.0, esta solución construida sobre Hadoop permite realizar consultas con gran rapidez y el análisis de grandes cantidades de datos.

Para la generación de los gráficos basados en los resultados obtenidos a partir de HIVE se programó una pequeña aplicación en PHP 5.0 soportado con la herramienta AmCharts que permite generar una gran variedad de gráficos estadísticos con cierto nivel de configuración a partir de archivos .txt.

CAPÍTULO 5

5. PRUEBAS Y RESULTADOS.

5.1 Ejecución de las pruebas

Para crear el clúster se usaron 4 equipos y como servidor principal se consideraron 3 configuraciones diferentes, con las siguientes características:

	Nodos	Servidor 1	Servidor 2	Servidor 3 (pc)
CPU	Core i3 2.27Ghz	Xeon 2.4Ghz (4 n.)	Xeon 3.06Ghz (2 n.)	Core i3 2.2Ghz
Memoria	4 Gb	12 Gb	2 Gb	4 Gb
HDD	400 Gb	1.2 Tb RAID 5	80 Gb RAID 5	500 Gb

Tabla I- Características de hardware del ambiente de pruebas

Para implementar los diferentes entornos de pruebas es necesario realizar la siguiente inversión:

	Hardware	Software	Inversión
Nodos	$450 (costo unitario) ***Total***: $1800 (4 nodos)	$0 (CentOS Apache Hadoop Hive) ***Total***: $0	**$1800**
Servidor 1	$6366	$923 (Windows Server) $1095.68 (SQL Server)	**$8597.99**
Servidor 2	$3500	$32.31 x usuario (CAL Windows) $181 x usuario (CAL SQL Server)	**$5731.99**
Servidor 3 (pc)	$460	***Total:*** $2231.99 (1 usuario CAL)	**$2691.99**

Tabla II– Costos aproximados de implementación de cada ambiente

Se ejecutaron los queries en Hive con 1, 2, 3 y 4 nodos y se obtuvieron los siguientes resultados:

Nodo/ Query	Ventas x Sucursal	Ventas x Negocio	Ventas Full	Rentabilidad
1	60	326	561	223
2	52	237	477	156
3	37	202	384	134
4	35	193	318	124

Tabla III– Tiempos de respuesta de los queries ejecutados en Hive.

Luego se ejecutaron los queries en cada configuración de servidor con SQL Server obteniendo lo siguiente:

Servidor/ Query	Ventas x Sucursal	Ventas x Negocio	Ventas Full	Rentabilidad
Servidor 1	5	11	33	25
Servidor 2	67	101	371	182
Servidor 3 (pc)	70	271	529	168

Tabla IV– Tiempos de respuesta de los queries ejecutados en SQL Server.

Si comparamos los resultados de los queries ejecutados en Hive con 4 nodos comparados con los tiempos obtenidos en SQL Server obtenemos la siguiente tabla:

	Ventas x Sucursal	Ventas x Negocio	Ventas Full	Rentabilidad
HIVE 4 nodos	35	193	318	124
Servidor 1	5	11	33	25
Servidor 2	67	101	371	182
Servidor 3 (pc)	70	271	529	168

Tabla V– Tiempos de respuesta de los queries Hive vs SQL Server 2008.

5.2 Análisis de los resultados

Con la información obtenida a partir de las pruebas, podemos observar que a mayor número de nodos menor es el tiempo de respuesta al ejecutar las consultas, lo cual es más notorio a partir del 2do nodo (Ver figura 5-1)

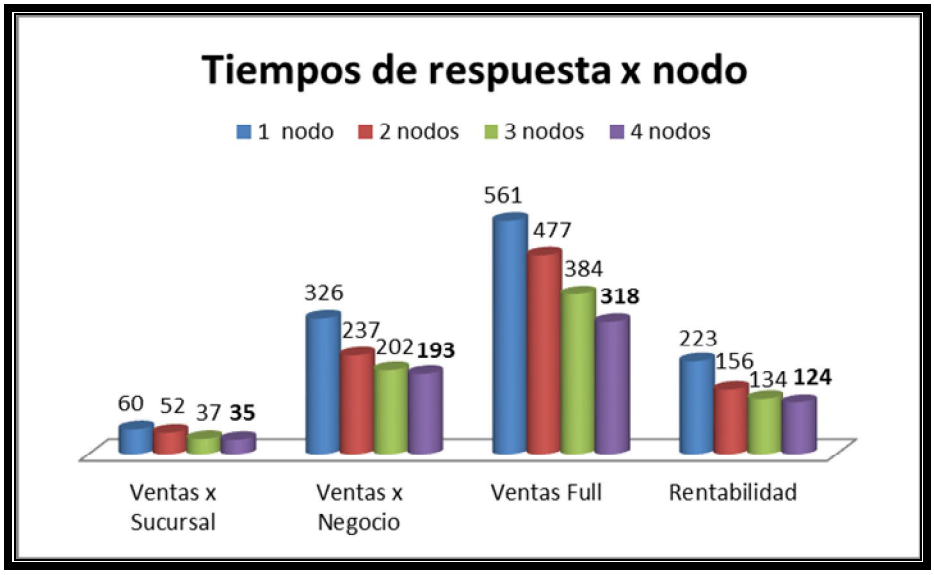


Figura 5-1-Ejecución de Queries en Hive con distintos nodos

Luego al comparar los tiempos de respuesta de SQL Server 2008 con Hive (4 nodos) logramos mejorar a dos de las tres configuraciones de servidores. (Ver figura 5.2)

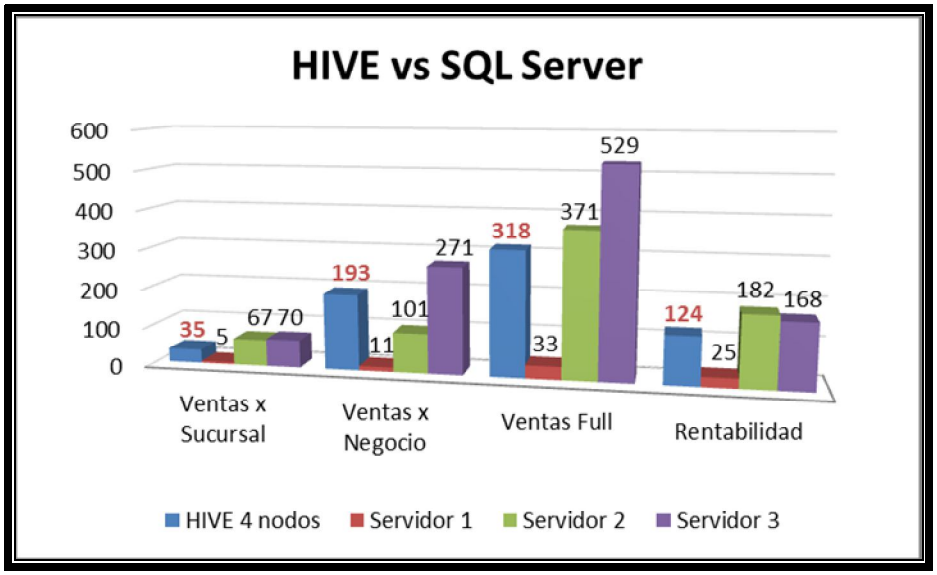


Figura 5-2- Ejecución de Queries Hive (4 nodos) vs SQL Server

Si nos enfocamos en la ejecución de un solo query como por ejemplo el de Ventas x Sucursal y lo comparamos con dos configuraciones del servidor SQL Server nos damos cuenta que con tan solo tener un nodo agregado al clúster el tiempo de respuesta es mejor (Ver figura 5.3).

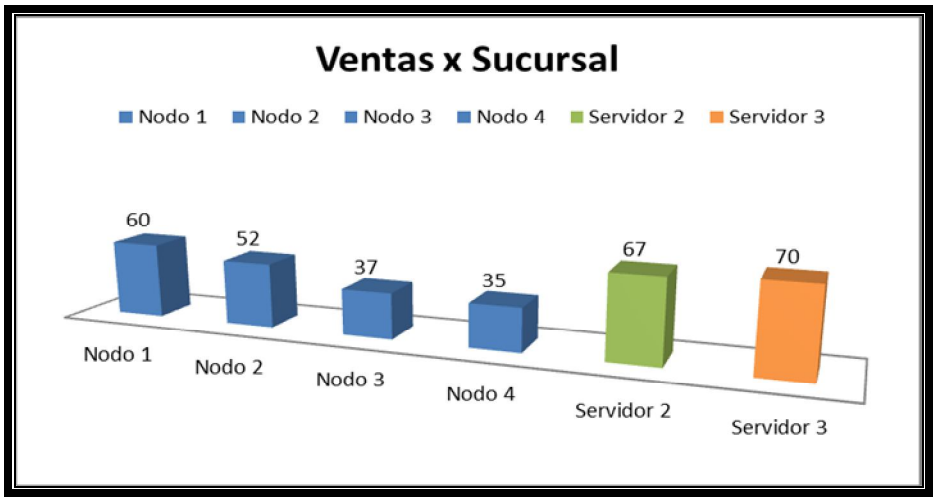


Figura 5-3- Ejecución de query “Ventas x Sucursal” en Hive con varios Nodos vs SQL Server en dos configuraciones

Similar resultado obtenemos al ejecutar el query “Ventas Full”, y en esta ocasión con 3 nodos equiparamos al mejor de los dos servidores (Ver figura 5.4)

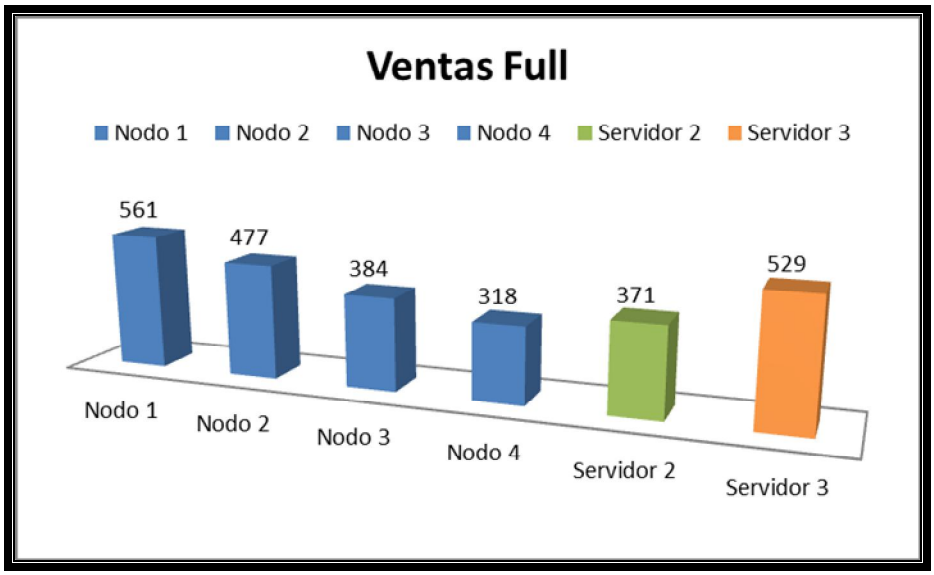


Figura 5-4- Ejecución de query “Ventas Full” en Hive con varios Nodos vs SQL Server en dos configuraciones

Dado que el ambiente de pruebas solo se pudo armar con 4 nodos y con dicha configuración no fue posible equiparar al Servidor 1, consideramos que es necesario determinar el número de nodos requeridos para igualar o mejorar los tiempos de respuesta del Servidor 1, por lo tanto, haciendo uso de la extrapolación exponencial de datos pudimos obtener los siguientes resultados:

	Ventas x Sucursal	Ventas x Negocio	Ventas Full	Rentabilidad
Servidor 1	5	11	33	25
No. de Nodos	13	19	16	12

Tabla VI- Tiempos de procesamiento de Datos Hive vs SQL Server 2008

Mostramos a continuación los gráficos de extrapolación de uno de los queries (ver figura 5-5).

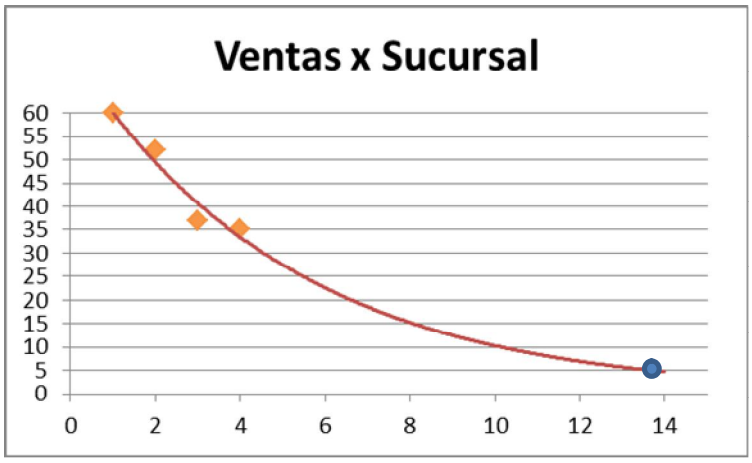


Figura5-5 Extrapolación de datos del query “Ventas x Sucursal”.

Tomando como muestra el número de nodos necesarios para igualar los 5 segundos de tiempo de respuesta del Servidor 1 podemos notar que sigue siendo más económica la configuración del clúster con 14 nodos:

	Hardware	Software	Inversión
Nodos	$450 (costo unitario) ***Total***: $6300 (14 nodos)	$0 (CentOS Apache Hadoop Hive) ***Total***: $0	**$6300**
Servidor 1	$6366	$923 (Windows Server) $1095.68 (SQL Server) $32.31 x usuario (CAL Windows) $181 x usuario (CAL SQL Server) ***Total:*** $2231.99 (1 usuario CAL)	**$8597.99**

Tabla VII– Costos comparativos entre Hive (14 nodos) y Servidor 1

Si mantenemos las mismas configuraciones durante un periodo 5 años (tiempo de vida útil de los equipos), la inversión necesaria en el caso de Windows y SQL Server 2008 (licenciamiento renovable cada 3 años mediante contrato) sería muy elevado, por lo tanto ratificamos el bajo costo de implementar un ambiente Open Source como lo es Hive sobre Hadoop.

CONCLUSIONES

Las conclusiones son:

1. La configuración inicial de un ambiente Hive sobre la plataforma Hadoop es, hasta cierto punto, manejable. Pero al momento de realizar las pruebas modificando el número de nodos entre cada ejecución de las consultas, se tuvo que ejecutar en la consola administrativa algunos pasos repetitivos que no siempre daban el resultado deseado, lo cual afectó la fluidez operativa en las pruebas.

2. Basados en el ambiente de pruebas configurado, podemos concluir que los tiempos de respuesta en Hive (como promedio 3 nodos) son menores en comparación a los obtenidos con dos de las tres configuraciones del motor de base de datos SQL Server 2008. Para poder igualar el desempeño del Servidor 1 es necesario implementar un clúster con un promedio de 14 nodos

3. Los costos para implementar un ambiente basado en Hive son menores en comparación con la implementación de una Base de Datos Relacional de paga, tanto en hardware como software (licenciamiento).

4. El uso de Hive proporciona escalabilidad, a medida que aumentamos mas nodos al clúster, los tiempos de respuesta mejoraron en cada una de las consultas.

5. La creación de consultas para el análisis de la información en HIVE es sencilla, solo se necesita tener conocimientos básicos de SQL.

RECOMENDACIONES

Las recomendaciones son:

1. Investigar la posibilidad del uso de conexiones automáticas de HIVE con la base de Datos Origen.

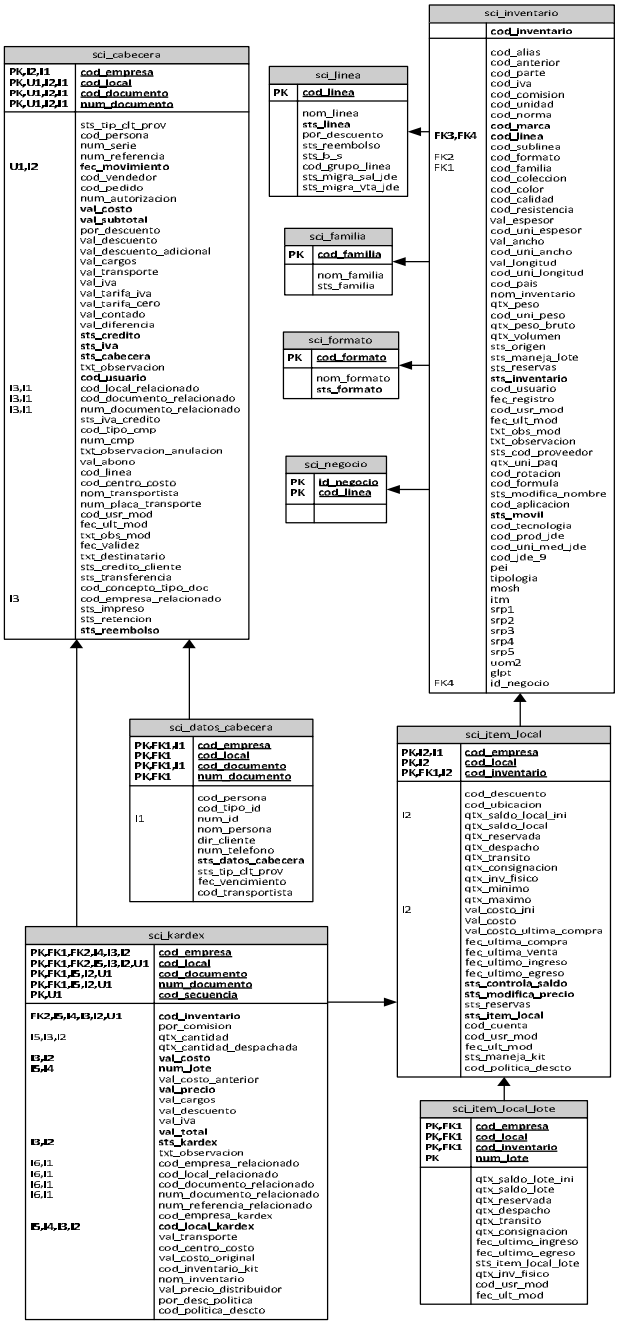
2. Implementar una interfaz más amigable que permita a usuarios que no sean expertos realizar automáticamente el levantamiento de la plataforma para realizar las consultas que requieran.

3. Sería recomendable que la solución se pueda implementar en un clúster con un mayor número de nodos físicos propios de la Espol, es decir un DataCenter especializado en la tecnología Apache Hadoop para poder llegar a resultados sin la necesidad de extrapolar datos.

ANEXOS

ANEXO A

Tablas Creadas



ANEXO B

Queries realizados

Ventas por Negocio

```
INSERT OVERWRITE LOCAL DIRECTORY
'/tmp/sum3'
SELECT YEAR(a.fec_movimiento) AS ANIO, d.id_negocio as NEGOCIO,
SUM(b.qtx_cantidad) AS TOTAL
FROM sci_cabecera as a JOIN sci_kardex as b ON a.cod_empresa =
b.cod_empresa and
a.cod_local = b.cod_local
and a.cod_documento = b.cod_documento
and a.num_documento = b.num_documento
JOIN sci_inventario as c ON b.cod_inventario = c.cod_inventario
JOIN sci_negocio as d ON c.cod_linea = d.cod_linea
where a.cod_empresa = 20 and a.cod_documento = 'FV'
GROUP BY YEAR(a.fec_movimiento), d.id_negocio, b.cod_documento;
```

Ventas por Local

```
INSERT OVERWRITE LOCAL DIRECTORY
'/tmp/sum'
SELECT year(fec_movimiento)AS ANIO, cod_local as SUCURSAL,
SUM(val_subtotal) AS TOTAL
FROM sci_cabecera
where cod_empresa = 20 and cod_documento = 'FV'
GROUP BY year(fec_movimiento), cod_local, cod_documento;
```

Ventas Full

```
INSERT OVERWRITE LOCAL DIRECTORY
'/tmp/sum1'
SELECT YEAR(a.fec_movimiento) ANIO, MONTH(a.fec_movimiento) MES,
DAY(a.fec_movimiento) DIA, a.cod_local SUCURSAL, a.cod_documento
TIPO_DOC, a.num_documento NUM_DOC, d.nom_linea LINEA,
e.nom_formato FORMATO, b.cod_inventario ITEM, c.nom_inventario
DETALLE, SUM(b.qtx_cantidad) CANTIDAD, SUM(a.val_subtotal) TOTAL,
SUM(f.qtx_saldo) SALDO FROM sci_cabecera a JOIN sci_kardex b ON
```

a.cod_empresa=b.cod_empresa and a.cod_local=b.cod_local and a.cod_documento=b.cod_documento and a.num_documento=b.num_documento JOIN sci_grupo_item f ON b.cod_empresa=f.cod_empresa and b.cod_inventario=f.cod_inventario and YEAR(a.fec_movimiento) = f.num_anio and MONTH(a.fec_movimiento)= f.num_mes JOIN sci_inventario c ON b.cod_inventario=c.cod_inventario JOIN sci_linea d ON c.cod_linea=d.cod_linea JOIN sci_formato e ON c.cod_formato=e.cod_formato WHERE a.cod_empresa = 20 and a.cod_documento = 'FV' GROUP BY a.fec_movimiento, a.cod_local, a.cod_documento, a.num_documento, d.nom_linea, e.nom_formato, b.cod_inventario, c.nom_inventario, b.qtx_cantidad

Rentabilidad por Año

```
INSERT OVERWRITE LOCAL DIRECTORY
'/tmp/sum2'
SELECT a.cod_local SUCURSAL,
YEAR(b.fec_movimiento) ANIO,
(a.qtx_cantidad * (a.val_precio - a.val_descuento)) RENTABILIDAD
FROM sci_kardex a JOIN sci_cabecera b ON
a.cod_empresa = b.cod_empresa AND
a.cod_local = b.cod_local AND
a.cod_documento = b.cod_documento
AND a.num_documento = b.num_documento
JOIN sci_inventario c ON a.cod_inventario = c.cod_inventario
WHERE (a.cod_empresa = 20) AND
(a.cod_documento = 'FV') AND
(a.cod_local_kardex IN ('202', '203', '224',
'205', 'HSS', 'HSU', 'H96', 'HVD', '222', '223',
'CTN', 'CGN', '208', '209', 'BE3', 'BE2', 'BE1', 'BE4', 'MUE'))
ORDER BY ANIO, SUCURSAL;
```

BIBLIOGRAFÍA

[1] Jason Venner, Pro HADOOP Build scalable, distributed applications in the cloud, Estados Unidos, 2009, Páginas.

[2] Chuck Lam, Hadoop in Action, Estados Unidos, 2011, Páginas 3-160, 246-265.

[3]The Apache Software Foundation: “Apache Hadoop” <http://hadoop.apache.org/> [Consultado :martes 4 de octubre del 2011].

[4]The Apache Software Foundation: “Apache Hive” < http://hive.apache.org/> [Consultado :martes 4 de octubre del 2011].

[5]Wikipedia < http://en.wikipedia.org/wiki/Apache_Hadoop > [Consultado :martes 4 de octubre del 2011].

[6]VMware < http://www.vmware.com/ > [Consultado :martes 4 de octubre del 2011].

[7]SlideShare <http://www.slideshare.net/glud/hadoop-en-accion-9249763> [Consultado :martes 4 de octubre del 2011].

[8]Humbug <http://www.humbug.in/serverfault/es/que-es-hadoop-y-que-es-utilizado-para--27829.html> [Consultado :martes 4 de octubre del 2011].