UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

ANÁLISIS Y COMPARACIÓN ENTRE EL MOTOR DE BASES DE DATOS ORIENTADO A COLUMNAS INFOBRIGHT Y EL FRAMEWORK DE APLICACIONES DISTRIBUIDAS HADOOP EN ESCENARIOS DE USO DE BASES DE DATOS ANALÍTICAS

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERA CIVIL EN COMPUTACIÓN

ERIKA FERNANDA SILVA BALOCCHI

PROFESOR GUÍA:
EDGARD PINEDA LEONE

MIEMBROS DE LA COMISIÓN:
JORGE PÉREZ ROJAS
DIONISIO GONZÁLEZ GONZÁLEZ

SANTIAGO DE CHILE
MAYO 2014

Resumen

Business Intelligence es la habilidad para transformar datos en información, y la información en conocimiento, de forma que se pueda optimizar la toma de decisiones en los negocios. Debido al aumento exponencial en la cantidad de datos disponibles en los ultimos años y a la complejidad de estos, las herramientas tradicionales de bases de datos y business intelligence pueden no dar a basto, suponiendo numerosos riesgos para las empresas.

El objetivo de la presente memoria fue analizar el uso del framework de aplicaciones distribuidas Hadoop en comparación a la solución actual de Penta Analytics, buscando hacer un mejor uso de la infraestructura y aumentando la disponibilidad de los datos a medida que el volumen de estos crece. Actualmente esta compañía utiliza un motor de bases de datos analíticas llamado Infobright, que permite la ejecución de consultas de manera eficiente dada su estructura columnar, pero a nivel de un único servidor, limitando las capacidades de manejo de datos y uso eficiente de todos los servidores.

Para realizar la comparación se tomaron en cuenta dos casos de procesamiento de datos reales; consultas OLAP y ETL, además de tres casos de consultas estándar. Para cada uno de estos casos se realizaron tres variantes según el volumen a procesar para evaluar el rendimiento según crecían los datos.

La solución Hadoop fue desarrollada en un cluster en la nube, con tres servidores (un maestro y dos esclavos). En el sistema de archivos del cluster se almacenó la información a procesar y se realizaron los sets de consultas mediante dos herramientas Hadoop: Hive e Impala. Los resultados obtenidos arrojaron que Hive presenta tiempo superiores a Impala e Infobright, esto debido al overhead que implica lanzar las tareas map y reduce, sin embargo es el único que ofrece tolerancia ante el fallo de un nodo. Por otro lado Impala presenta la menor latencia, con un tiempo de respuesta mucho menor a Infobright, no obstante presenta la mayor utilización de memoria.

A partir de los resultados se pudo observar que Hive se comporta mejor en trabajos pesados tipo ETL donde la robustez prime sobre el tiempo, e Impala aplica mejor en consultas ligeras donde prime la velocidad.

Se pudo concluir que la combinación de distintas herramientas en un ambiente con tecnología Hadoop pueden ofrecer un buen desempeño, además de mejor utilización de máquinas y eventual tolerancia a fallos. Sin embargo hay que tomar en cuenta la curva de aprendizaje implicada.

Agradecimientos

Ha sido un largo recorrido y se me viene mucha gente a la mente.

Quisiera comenzar agradeciéndole a mi familia. A mis padres Manuel y Erika por la confianza y el apoyo, por impulsarme a lograr mis metas y no rendirme nunca. Son un ejemplo de sacrificio y perseverancia y se se que ha sido difícil para ustedes también, pero acá estamos! A mis hermanitos Victoria, Manuel y Felipe por acompañarme durante esta travesía en Santiago, por el cariño, el apoyo, las risas y por soportarme!. Y a mi hermanita Libertad, gracias por el ánimo y estar siempre preocupada.

A mi pololo Javier por la preocupación, paciencia y cariño, por confiar en mi y darme ánimo en los momentos difíciles.

A la tía María, Nestor, Pilo y la Carola, por recibirme en primer año y preocuparse por mi. Por preguntar como estuvo mi día y estar ahí.

A los amigos que conocí los primeros años de la U, los que pasaron y los que se quedaron. A Elise, Tete, Sandra, Mati, gracias por su amistad y ayudarme a sobrevivir los primeros años de la U.

A los amigos del DCC. Pili, Jaime, Álvaro, Renato, Ronald, Rena. Gracias por las pitiadeces, las dccervezas y el apoyo. Los amigos de cana, Carlos, Furbina y Pola, gracias por las conversaciones, los helados trendy y hacer mas entretenida la practica.

A mi profe guía Edgard por confiar en mi, por su buena disposición y constante ayuda.

Y finalmente, pero no menos importante, gracias a Stackoverflow por la ayuda durante el desarrollo de la memoria y siempre tener una respuesta a mis inquietudes.

Tabla de contenido

Tabla de contenido **v**

Índice de tablas **vii**

Índice de figuras **viii**

1. Introducción **1**
1.1. Antecedentes Generales 1
1.2. Motivación 2
1.3. Objetivo 3
1.4. Metodología 3

2. Antecedentes **5**
2.1. Business Inteligence 5
2.1.1. Data Warehouse 6
2.1.2. ETL 7
2.1.3. OLAP 8
2.2. Plataforma BI 8
2.3. Bases de datos analíticas 9
2.3.1. Modelos 10
2.4. Bases de datos orientadas a columnas 10
2.4.1. Características 11
2.4.2. Ventajas 12
2.4.3. Desventajas 13
2.5. Infobright 13
2.6. Hadoop 14
2.7. Cloud Computing 19
2.8. Amazon Web Services 20

3. Caso de Estudio **22**
3.1. Procesamiento de datos 22
3.1.1. Consultas OLAP 23
3.1.2. Consultas ETL 27
3.2. Almacenamiento de Datos y Modelo Relacional 28
3.2.1. Procesamiento Inicial de Datos 29
3.2.2. Cubo 31
3.3. Planteamiento del Problema 35

4. Diseño e Implementación **36**
4.1. Solución Actual . . . 36
4.2. Descripción de la solución . . . 37
4.3. Infraestructura . . . 39
4.3.1. Máquinas . . . 39
4.3.2. Hadoop . . . 41
4.3.3. Roles . . . 42
4.4. Herramientas/Componentes . . . 43
4.5. Set de Datos . . . 45
4.5.1. Set 1: ETL . . . 45
4.5.2. Set 2: Cubo . . . 46
4.6. Escenarios de Prueba . . . 47
4.7. Set de Consultas . . . 49
4.7.1. Tipo 1: Escaneo . . . 50
4.7.2. Tipo 2: Agregación . . . 50
4.7.3. Tipo 3: Join . . . 51
4.7.4. Tipo 4: Consultas reales OLAP . . . 52
4.7.5. Tipo 5: Consultas reales ETL . . . 53

5. Resultados experimentales y Análisis : Comparación de desempeño entre Infobright, Hive e Impala **55**
5.1. Resultados . . . 55
5.1.1. Tipo 1: Escaneo . . . 55
5.1.2. Tipo 2: Agregación . . . 57
5.1.3. Tipo 3: Join . . . 59
5.1.4. Tipo 4: OLAP . . . 62
5.1.5. Tipo 5: ETL . . . 69
5.2. Análisis . . . 76

6. Conclusiones **78**

7. Trabajo Futuro **80**

Bibliografía **81**

A. cpuinfo **86**
A.1. Máquina Actual . . . 86
A.2. Cluster . . . 91
A.2.1. Master . . . 91
A.2.2. Nodo-1 . . . 94
A.2.3. Nodo-2 . . . 94

B. Consultas ETL **96**

C. Scripts ETL **99**
C.1. Hive Snappy . . . 99
C.2. Hive Sin Comprimir . . . 100
C.3. Impala Parquet . . . 101

C.4. Impala Sin Comprimir 102

Índice de tablas

3.1. fact_ventas ETL 29
3.2. dim_producto ETL 30
3.3. dim_cliente_etiqueta ETL 31
3.4. dim_cliente 32
3.5. dim_sucursal cubo 32
3.6. dim_producto cubo 33
3.7. dim_fecha cubo 33
3.8. fact_ventas cubo 34

4.1. descripción instancias EC2 40
4.2. Direcciones instancias EC2 40
4.3. compatibilidad de formatos y motores 47
4.4. algoritmos de compresión soportados motor y formato de archivo 49
4.5. Escenarios de prueba 49

5.1. Tiempo de ejecución de consultas tipo 1 56
5.2. Almacenamiento (MB) utilizado en consultas tipo 1 56
5.3. Tiempo de ejecución de consultas tipo 2 58
5.4. Almacenamiento (MB) utilizado en consultas tipo 2 59
5.5. Tiempo de ejecución de consultas tipo 3 60
5.6. Almacenamiento (MB) utilizado en consultas tipo 3 61
5.7. Tiempo de ejecución de consultas tipo OLAP 62
5.8. Tiempo de ejecución de consultas tipo ETL 70
5.9. Almacenamiento (MB) utilizado en consultas tipo ETL 70

Índice de figuras

2.1. Esquema de un proyecto de BI. Fuente: elaboración propia. 7
2.2. Esquema de arquitectura Hadoop. Fuente: elaboración propia. 16

3.1. Diseño del cubo. Fuente: Copia de pantalla de Jaspersoft Olap Designer [36] 26
3.2. Vista del cubo vacío. Fuente: Copia de pantalla de Jaspersoft Ad Hoc Report [36] 26
3.3. Consulta clientes periodo actual por estado de fuga. Fuente: Copia de pantalla de Jaspersoft Ad Hoc Report [36] 27
3.4. Consulta Clientes periodo actual por estado de fuga, solo octubre. Fuente: Copia de pantalla de Jaspersoft Ad Hoc Report [36] 27
3.5. Ejemplo de ETL. Fuente: elaboración propia. 28
3.6. modelo Entidad Relación. Fuente: elaboración propia. 34

4.1. Situación actual, esquema de organización de servidores. Fuente: elaboración propia. 37
4.2. Diseño solución OLAP. Fuente: elaboración propia 39
4.3. Instancias en VPC 41
4.4. Roles de las máquinas 44
4.5. Cloudera Manager 46

5.1. Gráfico de tiempo (segundos) por variante de consulta tipo 1. Fuente: elaboración propia 56
5.2. Gráfico de almacenamiento utilizado por variante de consulta tipo 1. Fuente: elaboración propia 57
5.3. Gráfico de tiempo (segundos) por variante de consulta tipo 2. Fuente: elaboración propia 58
5.4. Gráfico de tiempo (segundos) por variante de consulta tipo 2, excluido Hive. Fuente: elaboración propia 58
5.5. Gráfico de almacenamiento utilizado por variante de consulta tipo 2. Fuente: elaboración propia 59
5.6. Gráfico de tiempo (segundos) por variante de consulta tipo 3. Fuente: elaboración propia 60
5.7. Gráfico de tiempo (segundos) por variante de consulta tipo 3, excluido Hive. Fuente: elaboración propia 61
5.8. Gráfico de almacenamiento utilizado por variante de consulta tipo 3. Fuente: elaboración propia 61

5.9. Gráfico de tiempo (segundos) por variante de consulta tipo OLAP. Fuente: elaboración propia 63
5.10. Gráfico de tiempo (segundos) por variante de consulta tipo OLAP, excluyendo Hive. Fuente: elaboración propia 63
5.11. Estado cluster durante consulta tipo OLAP en escenario Hive-Snappy. Fuente: copia de pantalla Cloudera Manager 64
5.12. Estado cluster durante consulta tipo OLAP en escenario Hive-Sin Comprimir. Fuente: copia de pantalla Cloudera Manager 64
5.13. Estado cluster durante consulta tipo OLAP en escenario Impala-Parquet. Fuente: copia de pantalla Cloudera Manager 65
5.14. Estado cluster durante consulta tipo OLAP en escenario Impala-Sin Comprimir. Fuente: copia de pantalla Cloudera Manager 65
5.15. Estado nodo maestro durante consulta tipo OLAP en escenario Hive-Snappy. Fuente: copia de pantalla Cloudera Manager 66
5.16. Estado nodo maestro durante consulta tipo OLAP en escenario Hive-Sin Comprimir. Fuente: copia de pantalla Cloudera Manager 67
5.17. Estado nodo maestro durante consulta tipo OLAP en escenario Impala-Parquet. Fuente: copia de pantalla Cloudera Manager 68
5.18. Estado nodo maestro durante consulta tipo OLAP en escenario Impala-Sin Comprimir. Fuente: copia de pantalla Cloudera Manager 69
5.19. Gráfico de tiempo (segundos) por variante de consulta tipo ETL. Fuente: elaboración propia 70
5.20. Gráfico de almacenamiento utilizado por consultas tipo ETL. Fuente: elaboración propia 71
5.21. Gráfico de almacenamiento utilizado por consultas tipo ETL (solo comprimidos). Fuente: elaboración propia 71
5.22. Estado cluster durante consulta tipo etl en escenario Hive-Snappy. Fuente: copia de pantalla Cloudera Manager 72
5.23. Estado cluster durante consulta tipo etl en escenario Hive-Sin Comprimir. Fuente: copia de pantalla Cloudera Manager 72
5.24. Estado cluster durante consulta tipo etl en escenario Impala-Parquet. Fuente: copia de pantalla Cloudera Manager 73
5.25. Estado cluster durante consulta tipo etl en escenario Impala-Sin Comprimir. Fuente: copia de pantalla Cloudera Manager 73
5.26. Estado nodo maestro durante consulta tipo etl en escenario Hive-Snappy. Fuente: copia de pantalla Cloudera Manager 74
5.27. Estado nodo maestro durante consulta tipo etl en escenario Hive-Sin Comprimir. Fuente: copia de pantalla Cloudera Manager 74
5.28. Estado nodo maestro durante consulta tipo etl en escenario Impala-Parquet. Fuente: copia de pantalla Cloudera Manager 75
5.29. Estado nodo maestro durante consulta tipo etl en escenario Impala-Sin Comprimir. Fuente: copia de pantalla Cloudera Manager 75
5.30. Límite de memoria excedido en nodo-2 durante ETL. Fuente: copia de pantalla Cloudera Manager 76

Capítulo 1

Introducción

1.1. Antecedentes Generales

Durante las últimas décadas ha habido un aumento exponencial en la cantidad de datos disponibles, debido a redes sociales, movilidad, aplicaciones, menor costo de la banda ancha, cloud computing[1], entre otros [1]. Dado el volumen y complejidad de los datos, las herramientas tradicionales de bases de datos y Business Intelligence[2] pueden no dar a basto, implicando numeroso riesgos para la empresa. Esto ha hecho que las empresas recurran a nuevas herramientas y procesos para capturar, almacenar, procesar, analizar y sacarles el mayor provecho a los datos.

Penta Analytics [44] es una empresa que desde el 2002 ha desarrollado servicios y proyectos de alta complejidad en el ámbito de la inteligencia de negocios, apuntando al uso inteligente de los datos para lograr un aumento de utilidades, ventas, tasas de retención de clientes, rentabilidad de campañas comerciales, entre otros. Esto se logra mediante la integración de información valiosa de los datos extraída mediante modelos de minería de datos, con la generación de acciones comerciales.

Dentro de este proceso, el almacenamiento y procesamiento de datos juegan un rol fundamental. En la actualidad la estrategia de almacenamiento de Penta Analytics consiste en servidores de bases de datos que almacenan información de múltiples clientes y un sistema de copia de seguridad semanal, con el fin de mantener un respaldo de los datos ante cualquier eventualidad de los servidores. Sin embargo, no existe una estrategia de alta disponibilidad[3], ni se saca el máximo provecho de los servidores de bases de datos de la empresa.

Con esta memoria se busca analizar el uso de un framework de aplicaciones distribuidas, específicamente Hadoop, para mejorar la situación actual, haciendo mejor uso de la infraestructura y aumentando la disponibilidad de los datos, de manera de satisfacer de mejor manera

[1] Computación en la nube. Detalles en el capítulo Antecedentes, sección Cloud Computing

[2] Inteligencia de negocios. Detalles en el capítulo Antecedentes, sección Business Inteligence

[3] Se entiende por alta disponibilidad al nivel de disponibilidad implícito en el diseño de un sistema, que entregue un determinado nivel de continuidad operacional

la continuidad operacional de los servicios entregados por Penta Analytics a sus clientes.

1.2. Motivación

Existe una relación entre una estrategia eficiente de gestión, análisis de los datos y los resultados de negocio de una empresa, reflejados en una toma de mejores decisiones e identificación de problemas a tiempo.

Para lograr esto es necesaria una buena infraestructura y herramientas que permitan contar con la información donde, cuando y de la forma que se la necesite.

En el caso específico de Penta Analytics, se presenta como un aporte importante el investigar formas de disminuir la probabilidad de pérdida de información, aumentar la disponibilidad del sistema u optimizar tiempos de consultas, mediante el uso más eficiente de la infraestructura disponible. Un framework de aplicaciones distribuidas permitiría mitigar estos problemas de manera notoria optimizando a su vez el uso de los recursos de hardware disponibles.

Actualmente Penta Analytics utiliza un motor de bases de datos analíticas llamado Infobright [42] que permite la ejecución de consultas de manera eficiente debido a su estructura de datos columnar, pero a nivel de un solo servidor, limitando las capacidades de manejo de datos y de uso eficiente de todos los servidores de la empresa, como también mejoras en alta disponibilidad de los datos, lo que puede provocar finalmente altos costos para la empresa ante escenarios de fallas.

Hadoop [52] es un framework de aplicaciones, open-source, inspirado en los papers de Google de MapReduce y Google File System [58], que permite desarrollar aplicaciones que manejan grandes cantidades de datos. Este framework es útil para realizar proyectos que necesiten escalabilidad, porque permite almacenar y procesar petabytes de información. Es perfecto para clusters de servidores, ya que distribuye la información entre nodos, siendo posible disponer de miles de nodos. Al disponer los datos de manera distribuida la búsqueda se puede realizar de forma rápida ya que Hadoop los accede de forma paralela.

Con el presente trabajo se busca analizar el uso de un framework de aplicaciones distribuidas, específicamente Hadoop, definiendo escenarios de uso y de procesos de datos reales, de manera de utilizar de mejor manera todos los recursos de hardware disponibles, aumentando notoriamente la disponibilidad de los datos ante fallas.

Se tomarán en cuenta distintas variables de análisis:

- tamaño de datos
- tiempos de carga de datos
- tipos de consultas
- tiempos de respuesta a consultas
- escenarios de fallas y consistencia de datos

- compresión de datos
- configuraciones de hardware

De esta manera se logrará definir las ventajas y desventajas de cada sistema en distintos escenarios de uso.

1.3. Objetivo

El objetivo de esta memoria es estudiar, analizar y comprender el uso del framework de aplicaciones distribuidas Hadoop al utilizar grandes bases de datos de tipo analíticas y contrastar los resultados con los del motor de bases de datos Infobright.

Los objetivos específicos de este trabajo son los siguientes:

- Definir, analizar y evaluar distintos volúmenes de carga de datos.
- Definir, analizar y evaluar distintos escenarios de consulta y procesamiento de datos.
- Definir, analizar y evaluar distintos escenarios de fallas.
- Definir, analizar y evaluar distintas configuraciones de hardware.
- Medir el desempeño de las soluciones en los distintos escenarios (volumen de datos, tipos de consultas, configuración de los servidores).
- Desarrollo de una aplicación práctica basado en uno de los procesos intensivos de cálculo desarrollados por Penta Analytics y que se ejecuta sobre una Base de Datos Infobright, en el framework Hadoop.
- Realizar comparaciones y análisis de las soluciones en cuanto a tiempo de ejecución, memoria utilizada y configuración de hardware.

1.4. Metodología

Para la realización de este estudio se utilizó una metodología de cuatro pasos[4] consistente en : *investigación* de las tecnologías y situación actual, *planificación* de los ambientes y pruebas a desarrollar, *implementación* de las pruebas y *mediciones* de los resultados.

A continuación se describen los pasos en más detalle:

1. Estudio de bases de datos analíticas y sus modelos.
2. Estudio de motores de bases de datos orientados a columnas, específicamente Infobright.
3. Estudio del framework de aplicaciones distribuidas Hadoop.
4. Levantamiento de situación actual en Penta Analytics: estudio de Bases de Datos, estructura de redes y servidores.
5. Estudio y definición de datos y formas de cargar datos.

[4]Basada en el proceso de investigación cuantitativa[35]

6. Definición de escenarios de ejecución y pruebas.
7. Creación de scripts.
8. Prueba de las soluciones y medición del desempeño en base a los escenarios planteados.
9. Análisis comparativo de los resultados obtenidos.
10. Conclusiones y documentación de la solución.

En el siguiente capítulo se introducen los conocimientos necesarios para abordar este trabajo.

Capítulo 2

Antecedentes

En la siguiente sección se presenta la investigación bibliográfica realizada sobre los conceptos y herramientas necesarias para entender la ejecución del trabajo de memoria.

Se comienza introduciendo los conceptos relacionados con la situación actual como son business inteligence, bases de datos analíticas, bases de datos columnares e Infobright.

Luego se introduce el framework Hadoop, en que servicios consiste y proyectos asociados a este.

Finalmente se menciona cloud computing, en particular los servicios ofrecidos por Amazon, porque fue la plataforma elegida para realizar el desarrollo.

2.1. Business Inteligence

Business Intelligence o inteligencia de negocio es un conjunto de estrategias y herramientas enfocadas a la administración y creación de conocimiento mediante el análisis de datos existentes en una organización. Está compuesto por tres características principales:

Accesibilidad a la información

Los datos son la fuente principal de este concepto. Lo primero que debe garantizar este tipo de herramientas y técnicas es el acceso de los usuarios a los datos con independencia de la procedencia de estos.

Apoyo en la toma de decisiones

Se busca ir más allá en la presentación de la información, de manera que los usuarios tengan acceso a las herramientas de análisis que les permitan seleccionar y manipular sólo aquellos datos que les interesen.

Orientación al usuario final

Se busca independencia entre los conocimientos técnicos de los usuarios y su capacidad para utilizar estas herramientas. El objetivo último es que los usuarios finales que deben tomar decisiones o necesitan la información puedan hacer uso de ella de manera natural, fácil y transparente.

Los sistemas de inteligencia de negocio permiten pasar de los datos a la información, la que tiene un significado de negocio y puede generar conocimiento para la toma de mejores decisiones. Basándose en el conocimiento y la información se pasa a la acción, la toma de alguna decisión, lo que genera resultados reflejados en valor para la compañía. Este proceso produce nuevos datos, comenzando nuevamente el ciclo descrito.

Esquema de una solución de Business Inteligence

En un proyecto de Inteligencia de Negocios, primero se obtienen los datos desde variadas fuentes de información, tales como sistemas operacionales, información semi o no estructurada, ERP (Planificación de Recursos Empresariales), CRM (Administración de la Relación con los Clientes), fuentes externas, entre otros. A través de procesos de integración de datos se extrae la información y se consolida en una estructura llamada Data Warehouse o almacén de datos.

Una vez consolidada la arquitectura necesaria, se pueden realizar análisis mediante distintos dispositivos (móvil, computador, etc.) Para realizar el análisis se pueden emplear diversas técnicas, como Reporting, análisis OLAP, minería de datos, entre otros.

Se puede observar el esquema de un proyecto de Business Inteligence en la figura 2.1.

2.1.1. Data Warehouse

El data warehouse es el núcleo de un proyecto de inteligencia de negocio. Consiste en una colección de datos enfocada a integrar y depurar información de distintas fuentes, para ser procesadas y analizadas desde distintas perspectivas, ayudando a la toma de decisiones en la entidad que la utiliza (empresa, organización, etc). Básicamente es un expediente completo de la entidad, que almacena información transaccional y operacional en una base de datos diseñada para el análisis y divulgación eficiente de los datos. La creación de un data warehouse constituye ocasionalmente el primer paso, desde el punto de vista técnico, para una solución de Business Intelligence.

La ventaja principal de este tipo de bases de datos radica en las estructuras en las que se almacena la información (modelos de tablas en estrella, en copo de nieve, etc). Este tipo de persistencia de la información es homogénea y fiable, y permite la consulta y el tratamiento jerarquizado de la misma.

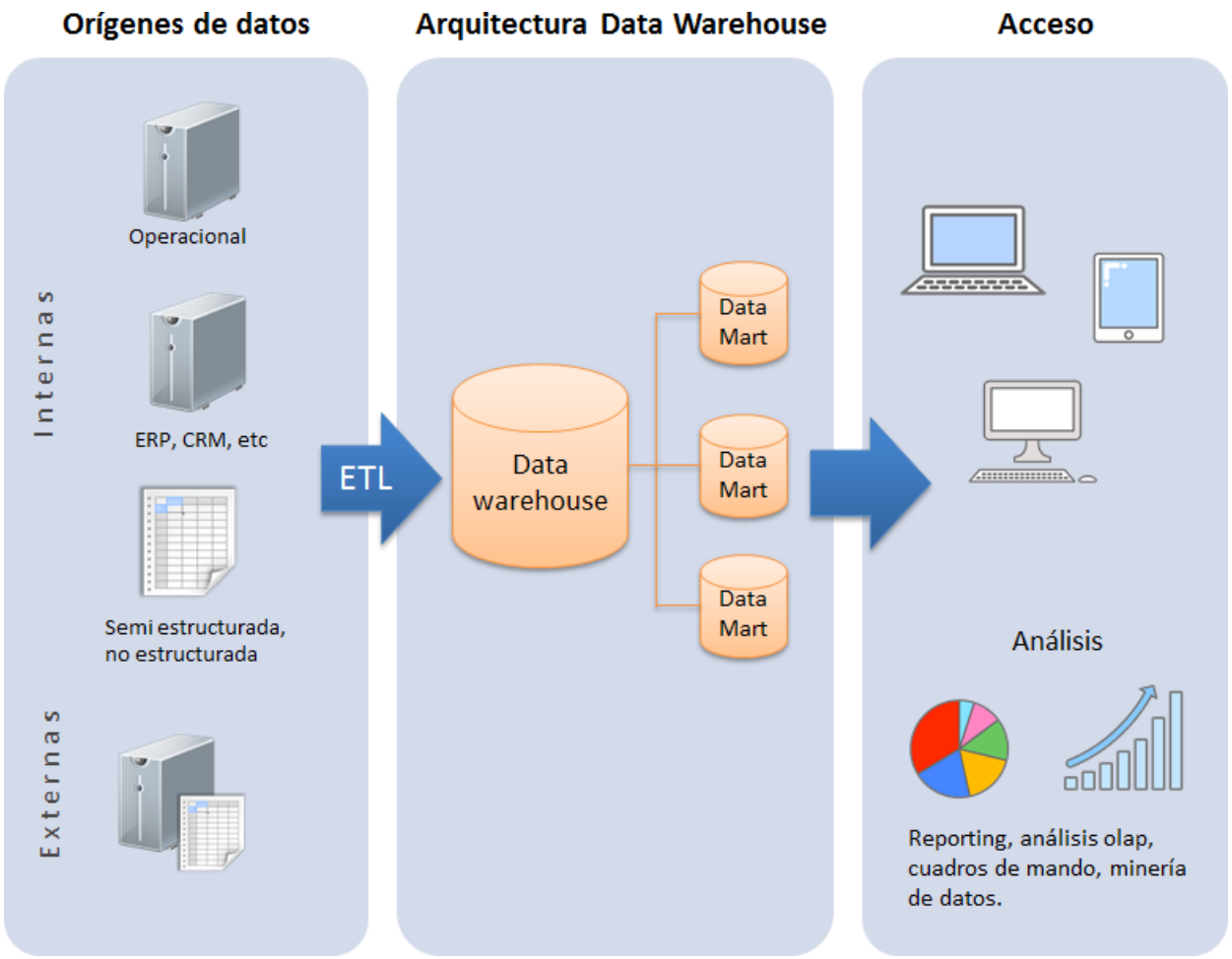

Figura 2.1: Esquema de un proyecto de BI. Fuente: elaboración propia.

2.1.2. ETL

ETL (Extract, Transform and Load ó Extraer, Transformar y Cargar) es un proceso que organiza el flujo de datos entre diferentes sistemas en una organización. Permite mover datos desde múltiples fuentes, modificando el formato, limpiándolos y cargándolos en otra base de datos, data mart[22] o data warehouse para un posterior análisis.

Como su nombre lo dice consta de 3 etapas:

- **Extracción**

  Es la primera fase del proceso y consiste en la extracción de datos desde los sistemas de origen, estos pueden provenir de diferentes sistemas con distintos formatos.

- **Transformación**

  En esta fase se aplica una serie de reglas de negocio o funciones sobre los datos extraídos. Entre las transformaciones más comunes se encuentra la selección de columnas especificas, traducción de códigos, cálculo de valores, unir datos de distintas fuentes, entre otros.

- **Carga**

Es la fase donde los datos de la fase de transformación son escritos en el sistema de destino. Dependiendo de los requerimientos puede abarcar diferentes acciones.

2.1.3. OLAP

OLAP (Online analytical processing ó procesamiento analítico en línea) es una solución de Business Intelligence que, como su nombre lo dice, consiste en procesamiento de la información en línea mediante un método analítico. Esta herramienta permite crear varias vistas y representaciones de los datos, agilizando de manera notable las consultas y evaluaciones de la información de una compañía.

Estos sistemas son utilizados para descubrir tendencias, analizar factores críticos y estadísticas.

Para funcionar, las aplicaciones OLAP utilizan una base de datos denominada cubos OLAP. Un cubo OLAP consiste en una base de datos multidimensional que ordena la información en vectores (cubos), de forma que quede organizada en jerarquías para permitir un acceso organizado, rápido y eficiente a los datos.

Cada una de las dimensiones de un cubo OLAP puede resumirse mediante una jerarquía. Por ejemplo, dentro de una escala temporal se podría tener una dimensión llamada *Enero 2013* , que incluida en otra dimensión *Primer trimestre 2013*, que a su vez se incluye en *Año 2013*. Así mismo pueden utilizarse otras dimensiones del cubo para recabar información referente a situación geográfica, clasificación de productos, partidas de gastos, entre otros.

El analista puede comenzar en un nivel muy resumido, por ejemplo la diferencia entre el resultado real y lo presupuestado, para luego descender en el cubo (en sus jerarquías) para observar con mayor detalle las diferencias por periodo o productos.

Esta forma de organización de la información permite llevar a cabo un análisis completo de diversas situaciones, para hallar las soluciones correctas a los problemas de negocios.

2.2. Plataforma BI

Las plataformas de BI consisten en un conjunto de aplicaciones diseñadas para colaborar con la inteligencia de negocios. Específicamente, herramientas que asisten el análisis y presentación de los datos. Penta Analytics hace uso de Jaspersoft Business Intelligence Suite [36], plataforma de BI de código abierto que permite a empresas generar información basada en datos de gestión para la evaluación y toma diaria de decisiones, de forma dinámica y online. Ofrece informes, dashboards, análisis y servicios de integración de datos. Está diseñada para variados entornos, ya sea en la nube, móviles y big data.

Este conjunto de herramientas permite integrar fácilmente diversas fuentes de datos disponibles en una empresa, y mediante técnicas de análisis multidimensional obtener indicadores

presentados en tableros de control y reportes dinámicos.

La suite de productos Jasper BI está integrado principalmente por:

Jasper Reports

Herramienta para el diseño y presentación de reportes, utiliza diversas clases de fuentes de datos, también permite incluir reportes dentro de otros y exportarlos a diversos formatos como PDF, CSV, XML, HTML. Es librería de mayor utilización en el mundo del código abierto, Integrándose en aplicaciones web, Java y de escritorio.

Jasper Server

Es el contenedor de reportes en un ambiente de acceso seguro a los usuarios y recursos. Puede ser usado stand-alone o desde otras aplicaciones. Se puede ejecutar, programar y distribuir los reportes a usuarios y grupos. Administra los recursos compartidos (imágenes, datos, etc)

Jasper ETL

Es una herramienta de ETL (Extract Transform Load) basada en el IDE Eclipse, permite la selección y procesamiento de múltiples fuentes de datos que alimentan y configuran el data warehouse corporativo. Cuenta con un entorno gráfico, y genera código para las transformaciones. Admite múltiples fuentes de datos (JDBC, Pop3, FTP, etc).

Jasper Analysis

Este software es utilizado para modelar, manipular y visualizar variados datos mediante cubos OLAP. Esta aplicación le permite al usuario explorar y analizar datos en línea, para poder tomar mejores decisiones rápidamente.

2.3. Bases de datos analíticas

Una Base de Datos es un conjunto de datos relacionados y almacenados para un propósito particular con una estructura lógica. La organización depende de las consultas que se piensan realizar sobre la base de datos.

Las bases de datos operacionales se centran en el almacenamiento de datos, por los que la operación más frecuente es la inserción. Los datos provienen de un proceso automático o repetitivo.

Las bases de datos analíticas tienen como objetivo la búsqueda de información, por lo que están optimizadas para realizar consultas.

2.3.1. Modelos

Esquema en estrella [24]

Es un modelo de datos que tiene una tabla de hechos (fact table) que contiene los datos para el análisis, rodeada de las tablas de dimensiones.

Las tablas de dimensiones tendrán siempre una clave primaria simple, mientras que en la tabla de hechos, la clave principal estará compuesta por las claves principales de las tablas de dimensiones.

El diseño del modelo, en forma de estrella, permite implementar funcionalidades de una base de datos multidimensional utilizando una base de datos relacional.

La simplicidad y rapidez de este esquema hacen que sea ideal para ser utilizada en análisis multidimensionales (OLAP, Datamarts, EIS, etc), permitiendo acceder tanto a datos agregados como detalle.

Es la opción con mejor rendimiento y velocidad pues permite indexar las dimensiones de forma individualizada sin afectar el rendimiento de la base de datos en su conjunto.

Esquema en copo de nieve [23]

Es un modelo de datos similar al anterior, pero más complejo. Se da cuando alguna de las dimensiones se implementa con más de una tabla de datos. La finalidad es normalizar las tablas, para así reducir el espacio de almacenamiento y eliminar la redundancia de datos, pero por otro lado implica generar peores rendimientos, al tener que crear más tablas de dimensiones y más relaciones entre las tablas.

La ventaja de los esquemas en copo de nieve es que las tablas de dimensiones están normalizadas, evitando con esto la redundancia de datos, y ahorrando espacio. Sin embargo, esto implica que las consultas se vuelven más complejas (mayor cantidad de Joins), tardando más tiempo en ejecutarse, por lo que si el tiempo de respuesta es un factor crítico no son recomendables.

2.4. Bases de datos orientadas a columnas

Las bases de datos orientadas a columnas son sistemas de bases de datos que se caracterizan por almacenar los datos en forma de columnas, es decir, por atributos y no por registros como lo hacen los sistemas relacionales.

Una base de datos orientada a filas, maneja internamente sus datos generando grupos de registros en bloques. Al utilizar estos bloques la base de datos obtiene el contenido de toda

una fila cada vez que necesite alguna información de ella, generando un mayor tiempo de gestión de la consulta, lo que hace que los procesos relacionados sean más lentos.

En cambio una base de datos orientada a columnas tiene los datos organizados y almacenados por columnas separadas, por lo que solo es necesario acceder a las columnas requeridas en la consulta, sin perder tiempo consultando por otros datos innecesarios.

La ventaja principal de este tipo de sistema es que se puede acceder como una unidad a los datos de un atributo en particular de la tabla, siento más eficaces para las consultas analíticas y dando una mejor respuesta a data warehouses que manejan grandes volúmenes de datos y soportan incrementos notables de datos.

2.4.1. Características

Tiempo de carga

La velocidad de carga de datos, usualmente medidos en gigabytes por hora, depende de varios factores, entre ellos la naturaleza de los datos y las elecciones realizadas por el usuario. Por ejemplo, algunos sistemas pueden guardar varias versiones de los mismos datos, pudiendo ser más rápidos con menos versiones, pero luego afectar al sistema con consultas más lentas.

Carga Incremental

Los datos deben ser recargados cada vez que sucede una actualización. Muchos sistemas columnares permiten la carga incremental, teniendo solo los registros nuevos o modificados y la fusión de los datos anteriores. Las funciones de carga incremental varían ampliamente, algunas tardan hasta una completa reconstrucción, algunos pueden agregar, pero no cambiar ni eliminar registros. Es recomendable a menudo complementarlas con una reconstrucción completa.

Compresión de datos

La compresión de los datos varía según los sistemas, algunos logran comprimir bastante la fuente de los datos y archivos para ocupar menos espacio, pero esto puede causar un impacto negativo en el rendimiento al realizar la lectura debido a la descompresión de los datos. Otros utilizan menos compresión o almacenan varias versiones de los datos comprimidos, ocupando más espacio, pero teniendo otros beneficios. El enfoque depende de los algoritmos, del hardware y de las características más valoradas al utilizar estos sistemas.

Limitaciones estructurales

Es recomendable tener en cuenta los límites impuestos por un sistema en particular, ya que las necesidades van cambiando y se podrían dificultar a futuro la escalabilidad del sistema. Las bases de datos columnares utilizan diferentes técnicas para responder como una estructura relacional.

Técnicas de acceso

Algunas bases de datos columnares pueden ser accedidas solamente mediante el lenguaje de consulta y herramientas del proveedor, incluyendo capacidades que son difíciles de encontrar usando SQL. Sin embargo, algunas veces faltan funciones especiales y se necesita acceder al sistema con herramientas basadas en SQL, en estos casos se debe determinar exactamente qué funciones SQL y dialectos son compatibles, y asegurarse que el rendimiento también sea compatible con la propia herramienta de consulta.

Rendimiento

Los sistemas orientados a columnas por lo general superan a los sistemas relacionales en casi todas las circunstancias, pero el margen es variado. Consultas que incluyen cálculos o accesos a registros individuales pueden ser igual o más lentos que en un sistema relacional bien indexado.

Escalabilidad

Un punto importante en las bases de datos orientadas a columnas es obtener buenos resultados con grandes volúmenes de datos, es decir, eficientemente escalables. Pero se debe tener en cuenta distintos factores para saber si el sistema podrá escalar a decenas o centenas de terabytes. Por ejemplo, el rendimiento puede depender de la carga de memoria, y no sólo de espacio de almacenamiento, por lo que el equipo debe tener suficiente memoria para el volumen de datos que se desea manejar.

2.4.2. Ventajas

Las principales ventajas de un sistema de bases de datos orientado a columnas son:

- Rápido acceso a los datos, con un mayor rendimiento al gestionar varias consultas simultáneamente.
- Lectura de datos sólo para valores de las columnas necesarias para el procesamiento de una consulta, siendo más eficientes en entornos de almacenes.

- Cambios realizados en el esquema generan un menor impacto. El rendimiento de las consultas no se ve afectado por el movimiento y actualización de datos entre columnas.
- Se evita tener que almacenar distintos índices que no resultan útiles y los datos son comprimidos, mejorando el procesamiento con el ancho de banda del acceso a disco disponible.
- La carga de los datos es de mayor velocidad que en una base de datos orientada a filas.

2.4.3. Desventajas

Las principales desventajas de un sistema de bases de datos orientado a columnas:

- No está orientado a sistemas transaccionales o de mucha escritura de datos aleatoria.
- Los reportes operacionales o de seguimiento pueden ser más ineficientes que con una base de datos orientada a filas, ya que se desea ver toda la información de una relación, que pude estar contenida en muchas tuplas.
- No existe un estándar que unifique los criterios de implementación de este modelo de base de datos por parte de los fabricantes de software.

2.5. Infobright

Infobright[42] es una Base de datos orientada a columnas de alto rendimiento optimizada para el análisis de grandes volúmenes de datos. Proporciona respuesta en un tiempo mínimo y a bajo coste, dada la facilidad de implementación y administración, y la reducción de espacio en disco.

Combina la tecnología de una base de datos orientada a columnas con una red de conocimiento propietaria para ofrecer una arquitectura de auto-gestión de data warehouse optimizado para el análisis de datos multidimensionales.

Resuelve consultas analíticas complejas sin necesidad de índices tradicionales, ajuste manual ni esquemas específicos. En vez de eso, la arquitectura de conocimiento en retícula crea y almacena automáticamente la información necesaria para resolver dichas consultas.

Los datos son estructurados en dos capas. La primera consiste en los datos comprimidos, que se almacenan en segmentos llamados "data packs" o paquetes de datos, y la otra consiste en la información sobre los datos que se almacena, que constituyen los "knowledge nodes" o nodos de conocimiento. Así, para cada consulta, el motor de Infobright utiliza la información del "knowledge node" para acceder a los "data packs" implicados, y a ninguno más. Juntos conforman el "Knowledge grid".

La tecnología de Infobright se basa en los siguientes conceptos:

Orientación a columnas

Como se ha mencionado, Infobright es una base de datos columnar con alta compresión de sus datos. Esto se refleja con una mejor eficiencia en la compresión de los datos, ya que cada columna solo tiene un tipo de datos. Y en eficiencia de tiempo de consulta, ya que la mayoría de ellas se realizan sobre columnas, rara vez por registro.

Paquetes de datos

Los "data packs" o paquetes de datos contienen un conjunto de información estadística sobre los datos almacenados en cada "data pack". El "knowledge node" o retícula de conocimiento tiene la metadata sobre los "data packs".

Retícula de conocimiento

Una capa de que almacena información compacta sobre los contenidos y relaciones entre los "data packs", reemplazando el concepto del tradicional índice de las bases de datos.

Mecanismo de computación granular

El mecanismo granular procesa las consultas haciendo uso de la información del "Knowledge grid" para optimizar el proceso de interrogación. El objetivo es eliminar o reducir significativamente la cantidad de datos que deben ser descomprimidos para responder una solicitud, de forma de acceder solo a los necesarios. Además, cuenta con un caché que resuelve muchas consultas sin necesidad de traer nuevamente la misma información.

2.6. Hadoop

Hadoop[52] es un framework open-source de Apache, que permite el almacenamiento y procesamiento distribuido de grandes volúmenes de datos a través de un clúster de computadores.

Es útil para la realización de proyectos que necesiten escalabilidad, porque puede almacenar y procesar petabytes de información. También es útil para clústers de servidores, debido a que la información es distribuida entre nodos, siendo posible disponer de miles de nodos. De esta forma la búsqueda puede realizarse rápidamente, ya que los datos son accesibles de forma distribuida y paralela.

Técnicamente consiste en 2 servicios importantes: un sistema de almacenamiento de datos llamado HDFS (Hadoop Distributed File System) e inspirado en GFS de Google, y un framework para computación paralela llamado MapReduce[58] para que el trabajo de las aplicaciones se dividan en pequeñas tareas. Ambos están diseñados de tal manera que los fallos en los nodos sean manejados automáticamente por el framework.

MapReduce

Es un framework introducido por Google para dar soporte a la computación paralela. Facilita la manipulación de grandes cantidades de datos en grupos de computadoras.

Tiene 2 algoritmos fundamentales: las funciones Map y las Reduce, que reciben datos estructurados de forma "(clave, valor)" para generar resultados finales o "reducidos".

HDFS

Hadoop Distributed File System (HDFS) es el sistema de almacenamiento principal utilizado por las aplicaciones de Hadoop. Este crea múltiples réplicas de blocks de datos y los distribuye en nodos a través de un clúster.

Arquitectura

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) consiste en un nodo de datos y un rastreador de tareas.

Los nodos maestros supervisan el almacenamiento y procesamiento paralelo de los datos, mientras que los esclavos constituyen la gran mayoría de las máquinas y hacen el trabajo de almacenar datos y ejecutar los cálculos.

Una funcionalidad importante es que cada sistema de archivos reconozca y proporcione su ubicación, es decir el nombre del rack donde está el nodo trabajador. Las aplicaciones pueden hacer uso de esta información para ejecutar el trabajo en el nodo donde están los datos, para así reducir el tráfico de la red troncal. El sistema de archivos también hace uso de esto para replicar los datos, ya que intenta conservar copias de los datos en distintos racks, para así reducir el impacto al fallar, de modo que los datos puedan aún estar legibles.

Proyectos Relacionados de Apache Software Foundation

Existen diversos proyectos de Apache[1] que se encuentran relacionados con el framework Hadoop, algunos de los principales son:

Hive [6]: Data warehouse

Hive provee una infraestructura de data warehouse construida sobre Hadoop, facilitando consultas y análisis de grandes cantidades de datos almacenados en sistemas compatibles

[1] Fundación de Software de Apache [54]

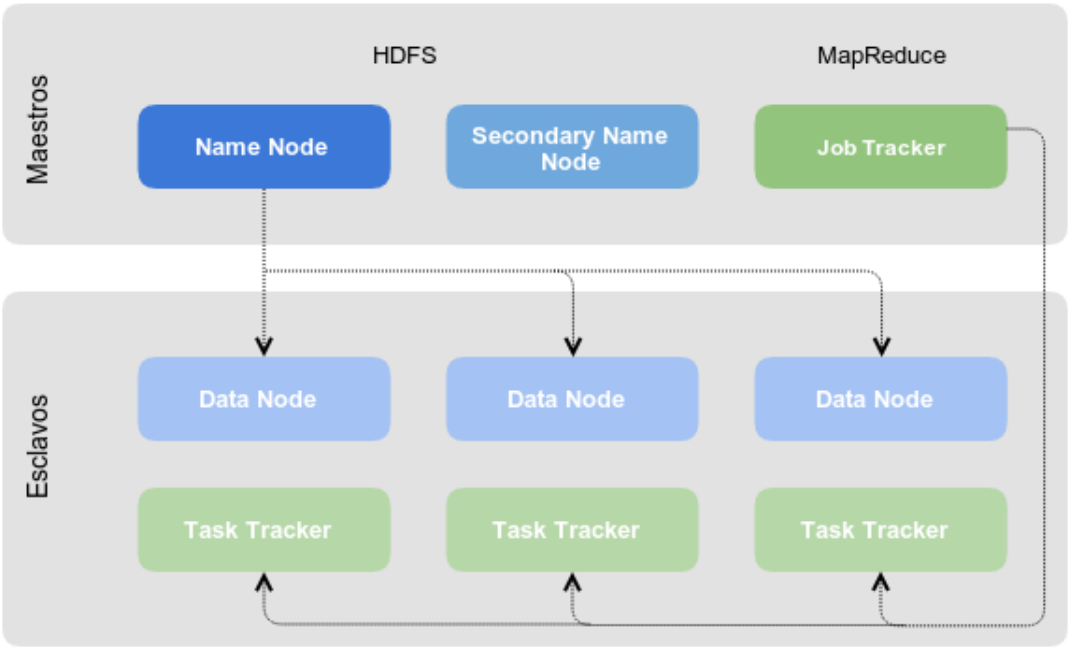

Figura 2.2: Esquema de arquitectura Hadoop. Fuente: elaboración propia.

con Hadoop. Provee un mecanismo de consultas llamado HiveQL, similar a SQL, que permite el acceso a los datos en HDFS y otros fuentes (como Amazon S3) mediante trabajos MapReduce.También permite funciones definidas por el usuario (UDF).

El modelo de datos de Hive ofrece una estructura más familiar para la mayoría de los usuarios que HDFS en bruto. Se basa principalmente en tres estructuras de datos relacionadas: tablas, particiones y buckets, donde las tablas corresponden a directorios HDFS y pueden ser divididos en particiones, que a su vez se pueden dividir en buckets.

Además, permite conexiones JDBC[2] /ODBC[3], por lo que se integra fácilmente con otras herramientas de Business Intelligence.

Hbase [29]: Base de datos

Basado en Bigtable de Google, HBase es una base de datos de Hadoop orientada a columnas, distribuida, no relacional y de código abierto. Se encuentra sobre HDFS y proporciona una forma tolerante a fallos de almacenar grandes cantidades de datos dispersos.

Los datos son tratados en Hadoop mediante MapReduce y se escriben en Hbase, como HBase está totalmente integrado con Hadoop, entiende el sistema de ficheros así como el formato de estos.

[2]Java Database Connectivity. Interfaz que permite a un programa java ejecutar instrucciones SQL dentro de bases de datos relacionales

[3]Open DataBase Connectivity. Estándar de acceso a bases de datos, que permite mantener independencia entre los lenguajes de programación, los sistemas de bases de datos y los sistemas operativos

Sqoop [49]: ETL

Sqoop (SQL to Hadoop) es una herramienta diseñada para transferir datos de forma eficiente, en ambas direcciones, entre sistemas de bases relacionales y HDFS u otros sistemas de almacenamiento Hadoop como Hive o HBase.

Permite importar tablas individuales o bases de datos completas, además, genera clases java gracias a las cuales se puede interactuar fácilmente con los datos importados.

Zookeeper [8] : Sincronización

Zookeeper es la implementación libre del paper Google Chubby, de Google. Básicamente es un sistema que proporciona una infraestructura centralizada para servicios basados en cluster y que necesitan ser sincronizados. Por ejemplo, datos de configuración, jerarquías de nombres, entre otros.

De esta forma, ZooKeeper ofrece un punto de acceso común a una gran variedad de objetos ampliamente utilizados en grandes entornos de cluster.

Pig [53]: Ayuda al Análisis

Es una plataforma que permite simplificar el análisis de grandes volúmenes de datos proporcionando un lenguaje de alto nivel. Consiste en un lenguaje de scripting de alto nivel (Pig Latin) y un entorno de ejecución que permite a los usuarios ejecutar MapReduce en un cluster Hadoop. Al igual que en HiveQL, Pig Latin es un lenguaje que se compila en MapReduce.

El objetivo de Pig es que los usuario puedan concentrarse más en el análisis de los datos y menos en la creación de los programas MapReduce.

Oozie [7] : Planificación de trabajos

Es un planificador de WorkFlows (flujos de trabajo) para gestionar trabajos Hadoop. Permite a los usuarios definir una serie de trabajos escritos en varios lenguajes, como MapReduce, Pig y Hive, creando entre ellos un flujo de procesos (jobs) con lógica. Oozie permite a los usuarios especificar, por ejemplo, que una determinada consulta sólo debe iniciarse después de determinados trabajos previos en los que se basa para recoger datos que se han completado.

Hue [33]: Interfaz web

Hue (Hadoop User Expirience) es un proyecto de código abierto que crea una interfaz web que facilita el uso de Apache Hadoop. Cuenta con un explorador de archivos para HDFS, una aplicación para la creación de flujos de trabajo Oozie, un diseñador de trabajos para MapReduce, una interfaz de usuario de Impala, y mucho más.

Proyectos Relacionados fuera de Apache

Existen proyectos fuera de Apache que se relacionan con Hadoop:

Spark [31] (UC Berkeley)

Spark es un sistema de computación de cluster en paralelo, que puede operar a través de cualquier fuente de entrada de Hadoop: HDFS, HBase, Amazon S3, Avro, etc. Es un proyecto de código abierto desarrollado en la UC Berkeley AMPLab, que tiene como objetivo realizar análisis de datos de manera rápida, tanto para ejecución como escritura. Fue inicialmente desarrollado para dos aplicaciones, donde mantener los datos en memoria ayuda: algoritmos iterativos, que son comunes en el aprendizaje automático, y minería de datos interactiva.

Comparado con MapReduce, si bien Spark también ofrece procesamiento paralelo sobre HDFS y otras fuentes de datos Hadoop, Spark difiere en dos aspectos fundamentales:

- Spark mantiene resultados intermedios en memoria, en vez de escribirlos a disco, reduciendo drásticamente el tiempo de retorno de la consulta.
- Spark soporta más que sólo funciones map y reduce, ampliando en gran medida el conjunto de posibles análisis que pueden ser realizados sobre HDFS.

La primera característica es la clave para realizar algoritmos iterativos en Hadoop: en lugar de leer desde HDFS, ejecutar MapReduce, escribir los resultados nuevamente en HDFS (es decir a disco) y repetir en cada ciclo, Spark lee los datos desde HDFS, realiza el cálculo, y almacena los resultados intermedios en la memoria como conjunto de datos distribuidos redolientes. De esta forma Spark puede ejecutar la siguiente serie de cálculos sobre los resultados almacenados en caché en la memoria, saltándose de este modo los pasos que requieren mucho tiempo, escribir los resultados de la n-ésima ronda a HDFS y leerlos nuevamente para la (n+1)-ésima ronda.

Shark [48] (UC Berkeley)

Shark es escencialmente Hive corriendo sobre Spark. Utiliza una infraestructura Apache Hive, incluyendo Hive metastore y HDFS, pero ofrece a los usuarios los beneficios de Spark (aumento de la velocidad de procesamiento, funciones adicionales además de map y reduce). De esta manera los usuarios pueden ejecutar queries en HiveQL sobre el mismo conjunto de datos en HDFS, pero recibiendo resultados cercanos al tiempo real.

Impala [34] (Cloudera)

Impala es un proyecto de código abierto lanzado por Cloudera, inspirado en el paper de Google en Dremel [63]. El propósito es facilitar consultas en tiempo real de datos en HDFS o HBase. Impala utiliza un lenguaje similar a SQL que, aunque es similar a HiveQL, es actualmente más limitado. Debido a que Impala depende del Hive Metastore, Hive debe estar instalado en el cluster para que funcione.

El secreto detrás de la velocidad de Impala es que elude MapReduce, para acceder directamente a los datos a través de un motor de consulta distribuida especializada muy similar a los encontrados en los RDBMS paralelos comerciales.

Phoenix [43]

Phoenix es una capa SQL sobre HBase, distribuido como un controlador JDBC embebido.

Phoenix provee queries de baja latencia (milisegundos) frente a las operaciones batch vía Map/Reduce.

El motor de queries Phoenix transforma las queries SQL en una o más escaneos HBase y orquesta su ejecución para producir conjuntos de resultados estándar JDBC. El uso directo de la API HBase, junto con coprocesadores y filtros personalizados se traduce en un rendimiento de milisegundos para pequeñas consultas o segundos para decenas de millones de filas.

Redshift [9] (Amazon)

Redshift es un proyecto de Amazon consistente en un Data Warehouse en la nube, optimizado para albergar bases de datos relacionales provenientes de múltiples fuentes y sistemas, ofreciendo un alto rendimiento en procesos de análisis e informes.

Promete rapidez, mediante almacenamiento de datos optimizado y escalabilidad, tolerancia a fallos y bajos costos.

A la fecha se encuentra como *limited preview*[4].

2.7. Cloud Computing

Cloud Computing o Computación en la nube es un paradigma que permite ofrecer diversos servicios computacionales a través de internet. Permite el acceso adecuado y bajo demanda a un conjunto de recursos de cómputo configurables (redes, servidores, almacenamiento, aplicaciones, servicios) que puedan ser rápidamente provistos y puestos a disposición del cliente con un mínimo de esfuerzo de gestión y de interacción con el proveedor del servicio.

Tiene cinco características esenciales:

Auto-servicio bajo demanda: Un cliente puede unilateralmente aprovisionarse de capacidades de cómputo (tales como uso de un servidor, almacenamiento en red, etc.) de acuerdo a sus necesidades, de forma automática y sin precisar de la interacción humana con el proveedor del servicio. (Nota: esta característica es probablemente la más ampliamente exigida y demandada para caracterizar al Cloud Computing y distinguirlo de otros paradigmas precedentes).

Recursos comunes: Los recursos de computo del proveedor son puestos en común para dar servicio a múltiples clientes usando un modelo multi-tenancy (infraestructuras compartidas) con diferentes recursos físicos o virtuales asignados dinámicamente y reasignados de acuerdo a la demanda del cliente. El cliente no tiene control ni conocimiento de la ubicación exacta de los recursos aprovisionados. Ejemplos de recursos de computo son: almacenamiento, procesamiento, memoria, ancho de banda de comunicaciones, maquinas virtuales, etc. Nota: como se observa el cómputo (o procesamiento) es solo uno de los tipos de recursos que se comparten (una de los aspectos en los que se diferencia de la virtualización de servidores)

[4]beta público limitado

Elasticidad rápida: Las capacidades pueden ser provistas (y liberadas) rápida y elásticamente, y en algunos casos automáticamente, de forma que el cliente tiene la visión de tener acceso a recursos ilimitados que puede comprar en cualquier cantidad y en cualquier momento.

Servicio Medible: El uso de los recursos es monitorizado, controlado y medido al nivel de abstracción apropiado para el tipo de servicio o recurso en cuestión (ancho de banda, procesamiento, almacenamiento, cuentas de usuario, etc.). De esta forma, la información del servicio utilizado es clara tanto por el consumidor como para el proveedor.

Acceso por red (Internet): las capacidades de computo están disponibles en la red y son accesibles mediante mecanismos estándares que promueven su uso por equipos de cliente heterogéneos (equipos de sobre mesa, PDAs, móviles, etc.)

2.8. Amazon Web Services

Amazon Web Services (AWS) [4] es un conjunto de servicios de infraestructura TI orientados al cloud computing (servicios web) ofrecidos a través de internet por Amazon.com. Sirve para una amplia gama de proyectos, desde un uso puntual para supercomputación en la nube hasta el alojo de archivos para copias de seguridad e incluso para alojar páginas web. Actualmente es utilizado en aplicaciones populares como Dropbox, Foursquare, HootSuite.

Entre los servicios más conocidos se encuentra EC2 y S3. S3, Simple Storage Service, es un servicio de almacenamiento remoto que permite a desarrolladores almacenar información ilimitada, y tener un control sobre ésta. EC2, Elastic Cloud Computing, permite crear máquinas virtualizadas de forma sencilla para montar la estructura de un servicio en la nube sin necesidad de utilizar ningún servicio físico.

AWS tiene un sistema de cobro por servicio ejecutado, dependiendo de la configuración, tiempo o espacio ocupado. Además, permite desplegar nuevas aplicaciones en el momento, aumentar la capacidad instantáneamente a medida que crezca la carga de trabajo y reducir la capacidad inmediatamente en función de la demanda.

Servicios

Los servicios que ofrece AWS son servicios aplicados al cloud computing; siempre orientados a la escalabilidad. La interacción con ellos se hace generalmente vía API o de una interfaz en el panel de control en la web.

A continuación se listan los servicios más destacados:

EC2

Amazon Elastic Compute Cloud (Amazon EC2) es un servicio web que proporciona capacidad informática con tamaño modificable en la nube. Proporciona un control completo de

los recursos y reduce el tiempo de arranque de sus servidores, lo que permite escalar recursos rápidamente según las necesidades del usuario. Provee herramientas de recuperación de datos y fuerte aislamiento frente a otros procesos realizados en sus máquinas.

Posee una interfaz de servicios web para iniciar y configurar el servicio, permitiendo la utilización de diversos sistemas operativos, personalizarlos, gestionar permisos de acceso a la red, entre otros.

S3

Amazon Simple Storage Service (Amazon S3) es un servicio de almacenamiento en la nube. Está diseñado para facilitar a los desarrolladores recursos informáticos escalables basados en la Web.

Mediante una interfaz sencilla se pueden almacenar y recuperar la cantidad de datos que se desee, cuando se desee y desde cualquier parte de la web. Además, es posible controlar los permisos a los ficheros mediante políticas de acceso.

Esto permite, por ejemplo, almacenar datos o imágenes de un sitio web en un servidor de alta disponibilidad externo a la máquina EC2.

Elastic IP

Elastic IP Addresses (EIP) es un servicio para el mapeo de IP fijas a máquinas EC2.

Una dirección de IP elástica es asociada con la cuenta, no con una instancia EC2 en particular, por lo que se tiene control sobre la dirección hasta que se decide liberarla. Esto permite, por ejemplo, reasignar la dirección a otra instancia si un nodo falla.

VPC

Amazon Virtual Private Cloud (Amazon VPC) es un servicio que permite aprovisionar una sección aislada de forma lógica en nube, posibilitando el lanzamiento de recursos de AWS dentro de una red virtual predefinida.

Proporciona control sobre todos los aspectos del entorno de red virtual, incluida la selección de su propio rango de direcciones IP, creación de subredes y configuración de tablas de enrutamiento y puertas de enlace de red.

Capítulo 3

Caso de Estudio

Actualmente Penta Analytics realiza una serie de servicios enfocados en mejorar la productividad de los recursos de las empresas. Usando datos transaccionales es posible medir el desempeño productivo de diversos factores, de modo de encontrar las condiciones óptimas para mejorar el resultado operacional.

Dentro de los servicios se encuentra el procesamiento de datos mediante procesos ETL[1]. Esto implica extracción de datos de distintas tablas y bases de datos, procesamiento de estos según lo que se necesita, para finalmente cargar el resultado en otra base de datos, quedando disponibles para ser analizados por los usuarios.

Otra parte importante es el servicio OLAP[2] que consiste en la organización de los datos en un vector multidimensional, ordenando la información en jerarquías y permitiendo un acceso rápido, eficiente y organizado.

Dentro del caso de estudio se contempla el análisis de ambos procesos, para evaluar en desempeño a distintos volúmenes y necesidades.

Para llevar a cabo estos procesos, Penta Analytics utiliza Infobright como su motor de base de datos principal en el Data Warehouse.

3.1. Procesamiento de datos

Para el efecto de este trabajo de memoria se tomará en cuenta un cliente correspondiente a una empresa de retail. En la actualidad, esta empresa cuenta con más de 700.000 clientes y 35 sucursales a lo largo del país.

Este cliente mantiene almacenada información de relevancia para poder realizar análisis estadístico posteriormente. Actualmente cuenta con el registro de las transacciones realizadas

[1] Extract Transform Load (Extraer, transformar y cargar) [25]

[2] acrónimo en inglés de procesamiento analítico en línea (On-Line Analytical Processing) [41]

desde comienzos del año 2010 hasta el periodo actual.

Se estudiarán dos casos específicos que representan procesos importantes llevados a cabo en la actualidad, procesamiento por lotes o ETL y procesamiento en linea o consulta OLAP.

A continuación se detallan las consultas a analizar.

3.1.1. Consultas OLAP

Para realizar el análisis en linea de los datos[3] se utilizan los reportes ad hoc[4] de Jaspersoft, presentes en Jaspersoft BI[5]. Esta herramienta lee el esquema del cubo predefinido y se conecta con la fuente de datos (base de datos Infobright), generando una vista de los datos en forma de tabla.

La importancia de estos procesos radica en que los usuarios pueden navegar y profundizar en los datos de forma interactiva, de forma de poder analizar detalles o patrones. Por eso el tiempo de respuesta debe ser acotado.

Este cliente requiere realizar consultas directamente al transaccional y poder visualizar los resultados de ventas mediante una interfaz gráfica. Para esto existe un cubo OLAP[6]on la información histórica de las ventas y los detalles requeridos.

Para la creación del cubo se utilizó la herramienta Jaspersoft OLAP Designer (imagen 3.1), que permite crear cubos de información para análisis multidimensional. Dicho cubo está compuesto por archivos XML donde se definen las dimensiones y las conexiones de los datos. A continuación se describen los componentes que lo conforman.

Dimensiones

Las dimensiones del cubo, las cuales permiten organizar los criterios de búsqueda o filtros de los datos, son las siguientes:

Clientes Estado Fuga

Caracterización del camino de los clientes hacia la fuga. Existen 11 estados posibles: En fuga, fugado, fugado dos meses, fugado nuevo, fugado tres meses, inactivo, nuevo, retornado, spot, spot sin compras, sube o mantiene.

Corresponde a la columna *cli_ estado_fuga* de la tabla *dim_ cliente*.

[3]detalle de los conceptos en el capitulo Antecedentes sección OLAP

[4]tipo de informe que permite a los usuarios finales modificar dinámicamente y profundizar a través de los datos para un potente análisis de la información

[5]detalles en el capítulo Antecedentes sección Plataforma BI

[6]Base de datos multidimensional, en la cual el almacenamiento físico de los datos se realiza en un vector multidimensional [21]

Clientes Segmento

Segmento al que pertenecen los clientes. Existen 5 segmentos distintos en total : A1, A2, A3, A4, A5.

Corresponde a la columna *cli_ etiqueta_ bimensual* de la tabla *dim_ cliente*.

Clientes Rubro

Rubro al que está asociado el cliente, por ejemplo Almacén, confitería, consumo particular, entre otros. Existen en la actualidad 18 rubros distintos.

Corresponde a la columna *cli_ rubro_ nuevo* de la tabla *dim_ cliente*.

Sucursal

Dimensión relacionada con la tabla *dim_ sucursal*.

Dentro de esta dimensión existen 3 jerarquías:

- **Región Sucursal** : Región a la que corresponde la sucursal, en total hay 10 regiones. Corresponde a la columna *sucu_ region*.
- **Gerente Mercado Sucursal** : Gerente de la sucursal correspondiente, existen 13 en total. Corresponde a la columna *sucu_ gerente_ mercado*.
- **Salas** : Identificador único de la sucursal, en total existen 35. Corresponde a la columna *sucu_ nombre*.

Productos

Dimensión relacionada con la tabla *dim_ producto*.

Dentro de esta dimensión existen 3 jerarquías:

- **Subgerencia Productos** : Subgerencia a la que corresponde el producto, existe 7 sub gerencias. Corresponde a la columna *subgerencia*.
- **Categoría Productos** : clasificación del producto, 355 categorías en total. Corresponde a la columna *linea*.
- **Marca Productos**: Marca del producto, 1393 marcas en total. Corresponde a la columna *marca*.

Fechas

Dimensión relacionada con la tabla *dim_ fecha*.

Dentro de esta dimensión existen 2 jerarquías:

- **Año** : Año por el que se desea filtrar. Corresponde a la columna *fecha_ anno*.
- **Mes** : Mes del año correspondiente. Corresponde a la columna *mes*.

Métricas

Las medidas o métricas representan una variedad de indicadores clave de rendimiento, y pueden incluir medidas simples así como medidas más complejas (miembros calculados), realizando cálculos sobre las medidas simples.

Para el caso de este cliente en particular, se definieron las siguientes métricas para realizar el análisis:

- **Clientes Periodo Actual** : Cantidad de clientes en periodo actual. Se calcula contando los distintos valores de la columna *cli_ nkey_ 13*.
- **Clientes Periodo Anterior** : Cantidad de clientes en periodo anterior. Se calcula contando los distintos valores de la columna *cli_ nkey_ 12*.
- **Transacciones Periodo Actual** : Cantidad de transacciones totales realizadas durante el periodo actual. Se calcula sumando los distintos valores de la columna *tran_ id_ 13*.
- **Transacciones Periodo Anterior** : Cantidad de transacciones totales realizadas durante el periodo anterior. Se calcula sumando los distintos valores de la columna *tran_ id_ 12*.
- **Monto Periodo Actual** : Monto total vendido durante el periodo actual. Se calcula sumando los montos de la columna *meas_ monto*.
- **Monto Periodo Anterior** : Monto total vendido durante el periodo anterior. Se calcula sumando los montos de la columna *meas_ monto_ anterior*.
- **Unidades Periodo Actual** : Cantidad total de productos vendidos durante el periodo actual. Se calcula sumando las cantidades de la columna *meas_ unidades*.
- **Unidades Periodo Anterior** : Cantidad total de productos vendidos durante el periodo anterior. Se calcula sumando las cantidades de la columna *meas_ unidades_ anterior*.

Miembros calculados:

- **Ticket Promedio Periodo Actual** : Valor promedio de las transacciones realizadas durante el periodo actual. Se calcula dividiendo la métrica Monto Periodo Actual por la métrica Transacciones Periodo Actual.
- **Ticket Promedio Periodo Anterior** : Valor promedio de las transacciones realizadas durante el periodo anterior. Se calcula dividiendo la métrica Monto Periodo Anterior por la métrica Transacciones Periodo Anterior.
- **Frecuencia Periodo Actual** : Se calcula dividiendo la métrica Transacciones Periodo Actual por la métrica Clientes Periodo Actual.
- **Frecuencia Periodo Anterior** : Se calcula dividiendo la métrica Transacciones Periodo Anterior por la métrica Clientes Periodo Anterior.
- **PVP Periodo Actual** : Se calcula dividiendo la métrica Monto Periodo Actual por la métrica Unidades Periodo Actual.
- **PVP Periodo Anterior** : Se calcula dividiendo la métrica Monto Periodo Anterior por la métrica Unidades Periodo Anterior.

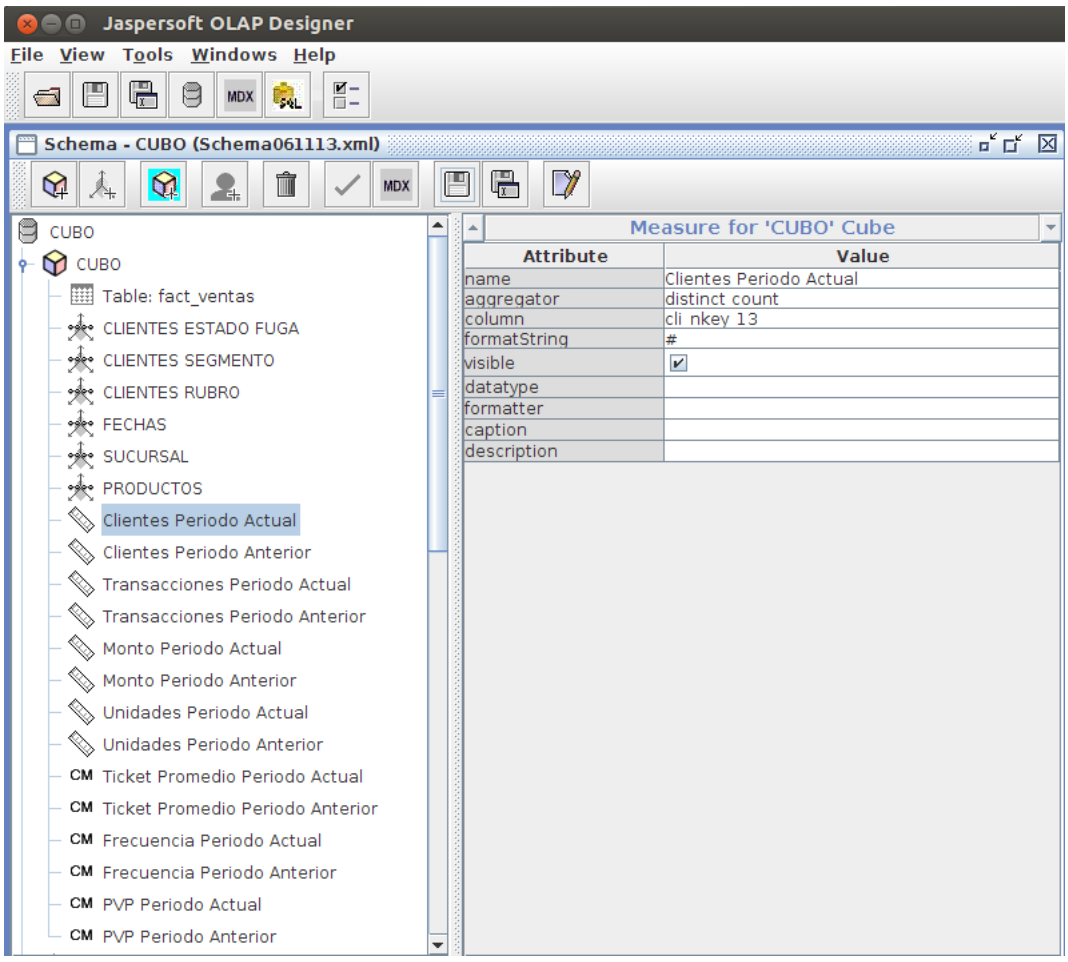

Figura 3.1: Diseño del cubo. Fuente: Copia de pantalla de Jaspersoft Olap Designer [36]

Se pueden ver todas las dimensiones y métricas definidas en el cubo (imagen 3.2), las que se pueden arrastrar hacia los campos Columns (columnas), Rows (filas) o Filters (panel de filtros). De esta forma, a medida que se van arrastrando dimensiones y métricas, se van haciendo consultas a la base de datos y se van mostrando los resultados en forma de tabla.

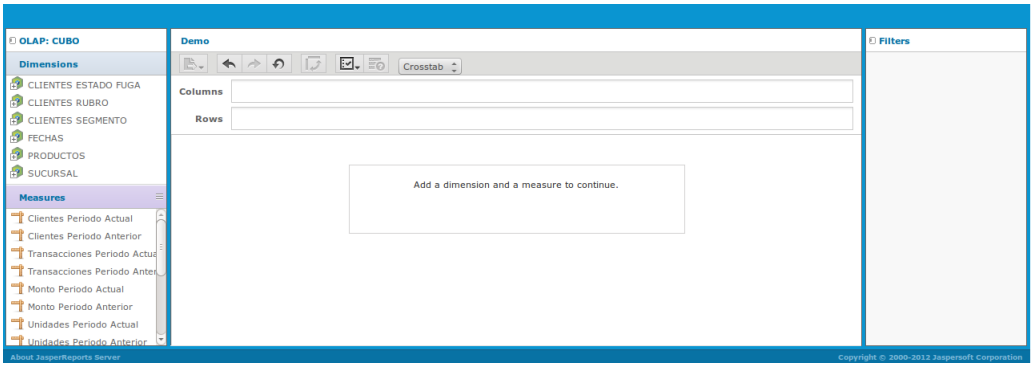

Figura 3.2: Vista del cubo vacío. Fuente: Copia de pantalla de Jaspersoft Ad Hoc Report [36]

Como se puede ver en la imagen 3.3, si se arrastra la métrica Clientes Periodo Actual hacia las columnas y la dimensión Estado Fuga Cliente a las filas, se genera una tabla con

los clientes por estado de fuga. Además, si se arrastra la jerarquía Mes de la dimensión Fecha hacia los filtros y se selecciona Octubre, se tendrá los clientes por estado de fuga del mes de octubre solamente (figura 3.4).

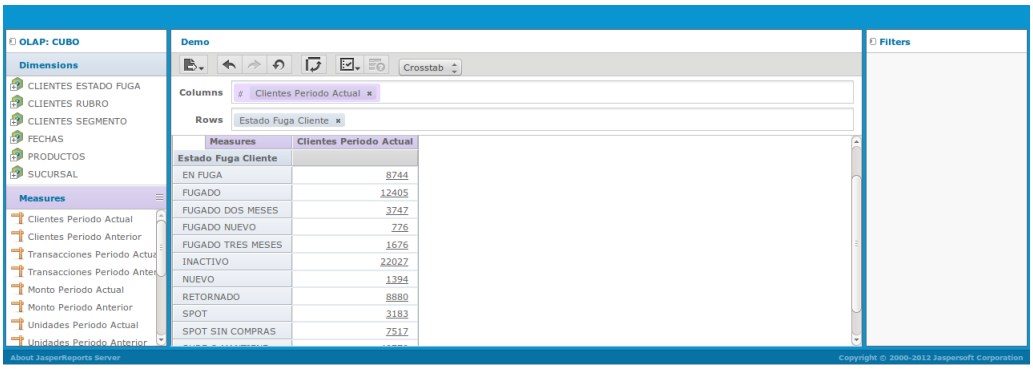

Figura 3.3: Consulta clientes periodo actual por estado de fuga. Fuente: Copia de pantalla de Jaspersoft Ad Hoc Report [36]

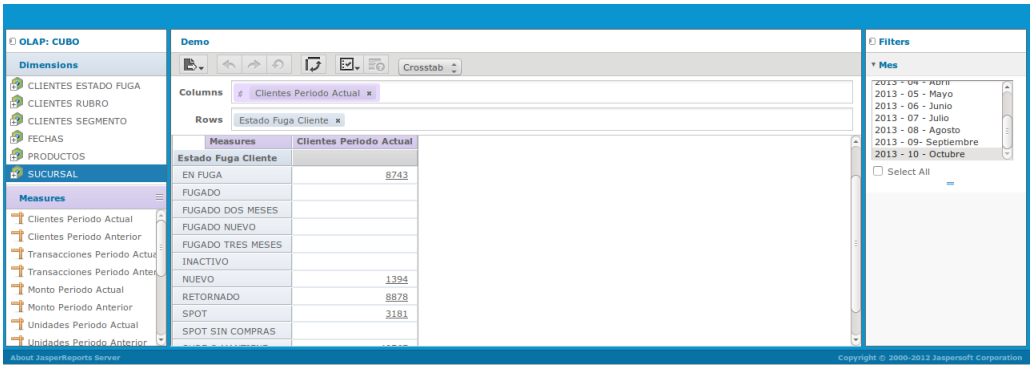

Figura 3.4: Consulta Clientes periodo actual por estado de fuga, solo octubre. Fuente: Copia de pantalla de Jaspersoft Ad Hoc Report [36]

A medida que se van arrastrando más dimensiones y métricas la vista va haciendo consultas más complejas, por lo que si se consultan tablas muy extensas o se agregan muchos parámetros estas pueden tardar mucho.

3.1.2. Consultas ETL

Se realizan procesos ETL para tomar información de distintas fuentes, darles la forma deseada e insertarla en otra fuente.

Este tipo de consulta representa un proceso muy importante, ya que al manejarse muchos datos es necesario recurrir a procesos ETL para prepararlos. Por ejemplo, para abastecer el cubo se deben calcular algunas métricas y eliminar datos de las tablas que no son relevantes para el caso.

En el ejemplo de la figura 3.5 se muestra como, mediante un proceso ETL, se extraen los datos desde las tablas que contienen los datos de las ventas, productos y clientes. Se procesan los datos y el resultado es almacenado en una tabla resumen.

Para las consultas de prueba se tomara parte de un proceso real representativo, que se puede resumir como:

- Extraer datos desde las tablas ventasfact, dim_cliente_etiqueta, dim_producto.
- Se procesan los datos para limitar el alcance a un periodo especifico. Se realizan cálculos como cantidad y monto total vendido en el año actual y anterior.
- Se almacenan los resultados en una nueva tabla.

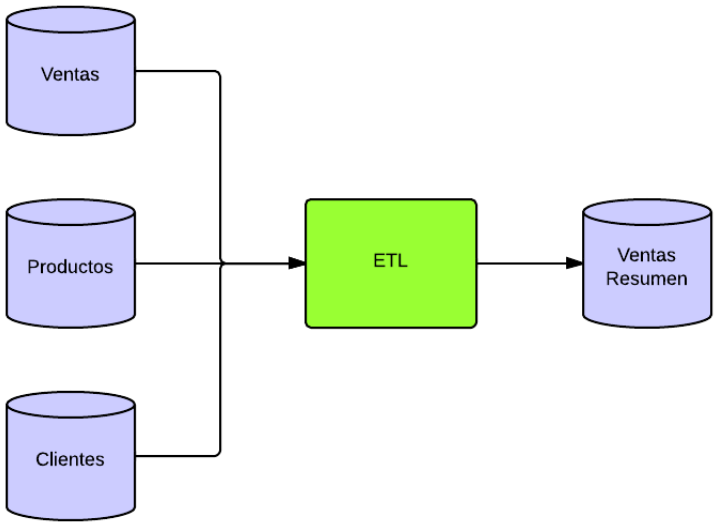

Figura 3.5: Ejemplo de ETL. Fuente: elaboración propia.

3.2. Almacenamiento de Datos y Modelo Relacional

La información histórica de las ventas, productos, clientes, y otros datos relevantes de las transacciones es registrada en bases de datos Infobright. Actualmente se utiliza el mismo servidor tanto para almacenamiento como propósitos operacionales.

Se contará con dos fuentes de datos dependiendo del caso a analizar, por una parte se encuentran los datos que son objeto del procesamiento ETL, distribuidos en distintas bases de datos. La segunda fuente cuenta con los datos ya transformados, desde donde se abastece al cubo.

3.2.1. Procesamiento Inicial de Datos

Para poder disponibilizar los datos y realizar un análisis de manera más fácil se realizan procesos ETL. En el caso a analizar se toman en cuenta las siguientes fuentes de datos:

1. una tupla **Venta** por cada transacción realizada. (tabla 3.1)
2. una tupla **Producto** por cada producto registrado. (tabla 3.2)
3. una tupla de **Cliente** por cada cliente. (tabla 3.3)

Tablas

- **Ventas**

  En la tabla *fact_ ventas* se tiene el registro de las transacciones realizadas desde el 2010 a la fecha, esto es más de 4 años, correspondiente a 280.000.000 registros.

  Cada entidad venta corresponde a la venta de un producto, y cuenta con campos correspondiente al número de la transacción (*tran_ id*), identificador de la fecha (*date_ id*), de la sucursal (*sucu_ id*), del producto (*prod_ id*), el cliente (*cli_ id*), documento (*docu_ id*), canal de venta(*canal_ id*), caja (*caja_ id*), entre otros datos (*desc_ meta_ id*, *canj_ kiosko_ id*, *meas_ desc_ desc_ meta*, *meas_ desc_ canj_ kiosko*, *meas_ unidades*, *meas_ precio_ promedio*, *meas_ costo_ promedio*, *meas_ monto*, *meas_ costo*, *convenio_ id*)

Campo	Tipo
tran_id	int(11)
date_id	int(11)
sucu_id	int(11)
prod_id	int(11)
cli_id	int(11)
docu_id	int(11)
canal_id	int(11)
caja_id	int(11)
desc_meta_id	int(11)
canj_kiosko_id	int(11)
meas_desc_desc_meta	int(11)
meas_desc_canj_kiosko	int(11)
meas_unidades	int(11)
meas_precio_promedio	int(11)
meas_costo_promedio	int(11)
meas_monto	int(11)
meas_costo	int(11)
convenio_id	int(11)

Tabla 3.1: fact_ventas ETL

- **Producto**

En la tabla *dim_producto* se tiene el registro de los productos ofertados durante el transcurso de los 4 años de registros, correspondiente a más de 18.000 registros.

Cada tupla corresponde a un producto, identificado mediante la clave primaria *prod_id*. Este es descrito mediante los campos *prod_nkey*, nombre del producto (*prod_nombre*), familia de productos (*prod_familia*), línea la que pertenece (*prod_linea*), sublinea (*prod_sublinea*), rut del proveedor (*prod_proveedor_rut*), nombre del proveedor (*prod_proveedor_nombre*), área del negocio (*prod_area_negocio*), subgerencia (*prod_subgerencia*), marca (*prod_marca*), entre otros (*prod_cis*, *prod_umb*, *prod_fvt*, *prod_todo*, *prod_nkey_marca*, *prod_cod_categoria*, *prov_id*).

Campo	**Tipo**
prod_id	int(11)
prod_nkey	varchar(18)
prod_nombre	varchar(100)
prod_familia	varchar(100)
prod_linea	varchar(100)
prod_sublinea	varchar(100)
prod_proveedor_rut	varchar(13)
prod_proveedor_nombre	varchar(100)
prod_cis	varchar(100)
prod_umb	varchar(100)
prod_fvt	varchar(100)
prod_todo	varchar(100)
prod_area_negocio	varchar(250)
prod_subgerencia	varchar(250)
prod_nkey_marca	varchar(250)
prod_marca	varchar(250)
prod_cod_categoria	varchar(250)
prov_id	int(11)

Tabla 3.2: dim_producto ETL

- **Cliente**

  En la tabla *dim_cliente_etiqueta* se tiene el registro histórico de los clientes, se agrega a esta la sucursal asignada, tipo, estado de fuga, etiqueta mensual, etiqueta bimensual y mes. Esto se traduce en más de 6.000.000 registros.

  Cada tupla cuenta con el id del cliente (*cli_id*), nkey del cliente (*cli_nkey*), razón social (*cli_razon_social*), datos de contacto (*cli_direccion*, *cli_comuna*, *cli_provincia*, *cli_telefono*, *cli_celular*), entre otros (*cli_credito*, *cli_tipo*, *cli_giro*, *cli_descripcion*, *cli_tipo_desc*, *cli_sap*, *cli_rubro_nuevo*, *cli_tipo_id*, *fecha_minima*, *fecha_maxima*, *cli_sucu_asignada*, *tipo*, *cli_estado_fuga*. *cli_etiqueta_mensual*, *cli_etiqueta_bimensual*, *mes*)

Campo	Tipo
cli_id	int(11)
cli_nkey	varchar(255)
cli_razon_social	varchar(255)
cli_direccion	varchar(255)
cli_comuna	varchar(255)
cli_ciudad	varchar(255)
cli_provincia	varchar(255)
cli_telefono	varchar(255)
cli_celular	varchar(255)
cli_credito	int(11)
cli_tipo	varchar(255)
cli_giro	varchar(255)
cli_descripcion	varchar(255)
cli_tipo_desc	varchar(255)
cli_sap	varchar(255)
cli_rubro_nuevo	varchar(255)
cli_tipo_id	int(11)
fecha_minima	int(30)
fecha_maxima	int(30)
cli_sucu_asignada	varchar(255)
tipo	varchar(255)
cli_estado_fuga	varchar(255)
cli_etiqueta_mensual	varchar(255)
cli_etiqueta_bimensual	varchar(255)

Tabla 3.3: dim_cliente_etiqueta ETL

3.2.2. Cubo

Para abastecer al cubo se procesan los datos mediante procesos ETL, para que solo cuenten con la información necesaria. Esta se resume como sigue:

1. una tupla de **Cliente** por cada cliente. (tabla 3.4)
2. una tupla de **Sucursal** por cada sucursal existente.(tabla 3.5)
3. una tupla **Producto** por cada producto registrado. (tabla 3.6)
4. una tupla **Fecha** por cada periodo de fecha registrado. (tabla 3.7)
5. una tupla **Venta** por cada transacción realizada. (tabla 3.8)

Tablas

Para poder ser analizados con mayor facilidad, los datos son almacenados con un modelo de datos en estrella. Las tablas y el modelo relacional (figura 3.6) son descritos a continuación:

- **Cliente**

  Corresponde a la tabla de dimensión *dim_cliente* del modelo, con clave primaria *cli_id*. Existe en total un aproximado de 300.000 clientes.

  En esta tabla se almacena información referente a los clientes inscritos. Cada entidad cliente cuenta con un identificador (*nkey*), rubro al que pertenece (*cli_rubro_nuevo*), razón social (*cli_razon_social*), sucursal al cual está asignado (*cli_sucu_asignada*), tipo de cliente (*tipo*), estado de fuga (*cli_estado_fuga*) y segmento al que pertenecen (*cli_etiqueta_bimensual*).

Campo	**Tipo**
cli_id	int(11)
cli_nkey	varchar(10)
cli_rubro_nuevo	varchar(40)
cli_razon_social	varchar(50)
cli_sucu_asignada	varchar(30)
tipo	varchar(30)
cli_estado_fuga	varchar(20)
cli_etiqueta_bimensual	varchar(5)

Tabla 3.4: dim_cliente

- **Sucursal**

  Corresponde a la tabla de dimensión *dim_sucursal* del modelo, cuya clave primaria es *sucu_id*. En total hay 35 sucursales.

  En esta tabla se almacena información referente a las sucursales de la empresa. Cada entidad sucursal cuenta con el nombre (*sucu_nombre*), nombre del gerente (*sucu_gerente_mercado*) y región a la que corresponde (*sucu_region*).

Campo	**Tipo**
sucu_id	int(11)
sucu_nombre	varchar(25)
sucu_gerente_mercado	varchar(30)
sucu_region	varchar(30)

Tabla 3.5: dim_sucursal cubo

- **Producto**

  Corresponde a la tabla de dimensión *dim_producto* del modelo, con clave primaria *prod_id*. Existe un total de 18.000 productos aprox.

  En esta tabla se almacena información referente a los productos que ofrece la empresa. Cada entidad producto cuenta con la subgerencia a la que pertenece (*subgerencia*), categoría del producto (*linea*) y marca (*marca*).

Campo	Tipo
prod_ id	int(11)
subgerencia	varchar(30)
linea	varchar(30)
marca	varchar(30)

Tabla 3.6: dim_producto cubo

- **Fecha**

  Corresponde a la tabla de dimensión *dim_fecha* del modelo, con clave principal *date_ id*. Representa períodos de tiempo dentro de los cuales se realizan las transacciones.

  Cada entidad fecha cuenta con campo del mes correspondiente (*fecha_ mensual*), año (*fecha_ anno*), trimestre (*fecha_ trimestre*), semestre (*fecha_ semestre*), e identificadores varios (*mes_ id*, *ano_ id*, *trimestre_ id*)

Campo	Tipo
date_ id	int(11)
fecha_mensual	varchar(10)
fecha_anno	varchar(10)
fecha_mes	varchar(10)
fecha_acum_ini	varchar(10)
fecha_acum_fin	varchar(10)
fecha_ini	varchar(10)
fecha_fin	varchar(10)
fecha_trimestre	varchar(16)
fecha_semestre	varchar(15)
anomes	int(11)
mes	varchar(14)
trimestre_id	int(11)
mes_id	int(11)
ano_id	int(11)

Tabla 3.7: dim_fecha cubo

- **Ventas**

  Corresponde a la tabla de hechos *fact_ ventas* (fact table) del modelo. Cuenta con más de 40.000.000 registros, correspondiente a 2 años de datos.

  Cada entidad venta corresponde a la venta de un producto, y cuenta con campos correspondiente al número de la transacción (*tran_ id*), el identificador del cliente (*cli_ nkey_ 12* y *cli_ nkey_ 13*, dependiendo del año de la venta), el identificador de la transacción (*tran_ id_ 12* y *tran_ id_ 13*), el monto (*meas_ monto* y *meas_ monto_ anterior*) y las unidades de la venta (*meas_ unidades* y *meas_ unidades_ anterior*)

  En esta tabla la clave principal está compuesta por las claves principales *cli_ id*, *sucu_ id*, *prod_ id*, *date_ id*, que corresponden a las claves principales de las tablas *dim_ cliente*, *dim_ sucursal*, *dim_fecha*, *dim_ producto* respectivamente.

Campo	Tipo
cli_id	int(11)
sucu_id	int(11)
prod_id	int(11)
date_id	int(11)
tran_id	int(11)
cli_nkey_13	varchar(10)
cli_nkey_12	varchar(10)
tran_id_13	int(11)
tran_id_12	int(11)
meas_monto	decimal(16,2)
meas_unidades	decimal(16,2)
meas_monto_anterior	decimal(16,2)
meas_unidades_anterior	decimal(16,2)

Tabla 3.8: fact_ventas cubo

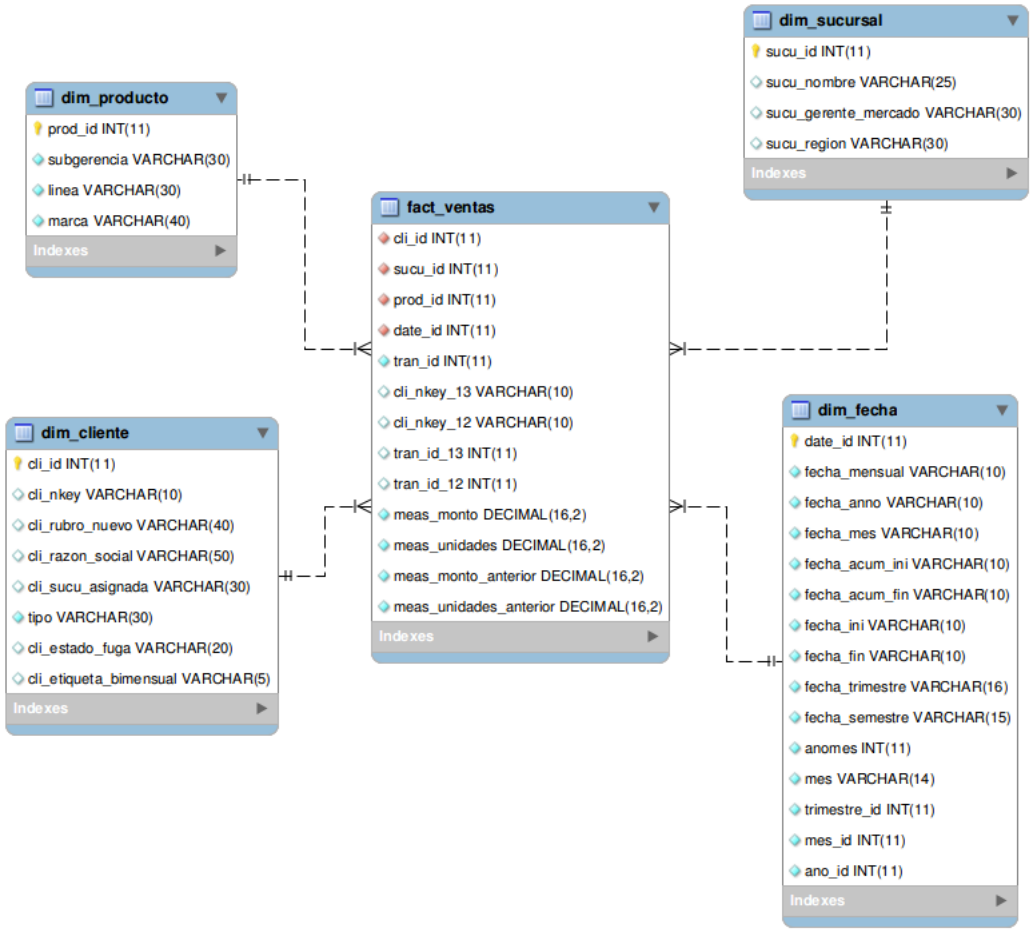

Figura 3.6: modelo Entidad Relación. Fuente: elaboración propia.

3.3. Planteamiento del Problema

En la actualidad se utiliza Infobright como motor de base de datos, este permite la ejecución de las consultas de manera eficiente, pero a nivel de un solo servidor.

Además, conforme pasa el tiempo la cantidad de datos recopilados va creciendo, y es necesario que la velocidad de respuesta de la solución analítica sea suficientemente rápida para lograr obtener la información correcta en el momento preciso.

Podemos ver que los problemas a los que la empresa se enfrenta poseen las características principales de una oportunidad de Big Data [13]:

1. Existe muchos datos almacenados en una única máquina.
2. El tiempo de acceso a la base de datos incrementa drásticamente a medida que la cantidad y complejidad de los datos aumenta.

Por consiguiente se desea estudiar el uso de una plataforma Big Data, como es Hadoop, sin pérdida significativa de rendimiento en los conjuntos de datos existente, con el fin de prepararse para los futuros conjuntos de datos de mayor tamaño. Al realizar esto se verá si es posible incluso lograr una ganancia de rendimiento en los conjuntos actuales y futuros.

Sabemos que Hadoop ofrece una solución que funciona muy bien para analizar grandes volúmenes de datos, pero se desea averiguar como se desempeñará en este caso en particular.

Para esto se realizará en el siguiente trabajo el diseño e implementación de una solución de BI sobre Hadoop, comparando la escalabilidad de la solución con la existente.

Capítulo 4

Diseño e Implementación

El proceso de desarrollo e implementación de la solución al caso de estudio se presenta en 7 partes.

Se comienza describiendo la solución que se utiliza en la actualidad, basada en Infobright. A continuación se describe a grandes rasgos la solución que se implementó con el framework Hadoop, para luego explicar con más detalles cada una de las decisiones de infraestructura y componentes que se tomaron.

La infraestructura contempla las decisiones tanto de hardware como de software necesarias para preparar el ambiente donde se desarrolló la solución Hadoop. A continuación se muestran los componentes de software a utilizados dentro de la solución, es decir herramientas relacionadas al framework. Luego se definen los set de datos sobre los cuales se realizaron las pruebas, para finalizar con el detalle de cada una de las consultas de pruebas.

4.1. Solución Actual

Actualmente Penta Analytics cuenta con una solución de bases de datos columnar llamada Infobright que provee excelentes tiempos de respuestas a consultas para base de datos analíticas.

Las bases de datos de cada cliente son separadas por servidor, de manera agrupada, tratando de balancear a los clientes según su tamaño de datos y el nivel de complejidad de los procesamientos de datos que se necesite en la entrega del servicio analítico.

El el siguiente diagrama (figura 4.1) se muestra un esquema general de la organización de los servidores según los clientes y servicios.

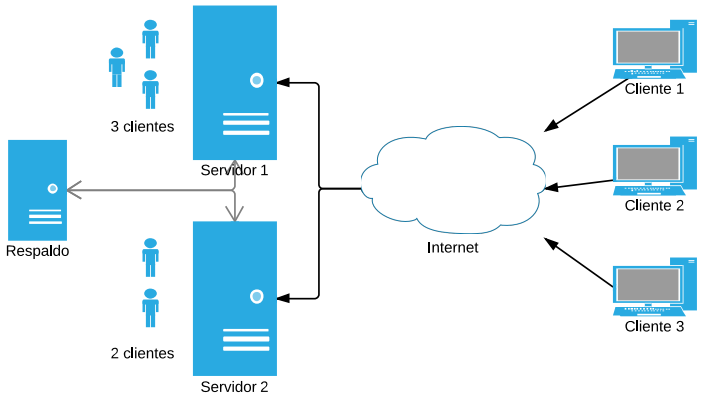

Figura 4.1: Situación actual, esquema de organización de servidores. Fuente: elaboración propia.

En el esquema anterior se ve que un cliente siempre estará en un sólo servidor. Esta organización provoca las siguientes desventajas:

- Al generarse una falla en un servidor se tienen tiempos de respuestas muy elevados debido a la espera en la recuperación de los datos desde respaldos o cargarlos en otro servidor.
- Sobre costos de mantención debido a movimientos de datos entre servidores, que generalmente son realizados de manera manual.
- Potencial pérdidas de datos debido a la periodicidad semanal de generación de respaldos de cada servidor.
- Se desaprovechan los recursos de cálculo y almacenamiento por tener tiempos .ºciosos"de servidores.
- Se requieren esfuerzos periódicos de re-distribución de datos de clientes de manera de poder usar el mayor potencial posible de cada servidor.
- Mayor riesgo de congestión (sobre-utilización) de un servidor debido al aumento de consultas sobre él, dado a que la organización de recursos es reactiva y no proactiva, y manual.

Para el caso a estudiar, los datos y el procesamiento se llevó a cabo en un servidor de la empresa que es utilizado para procesar datos. Este cuenta con un procesador Intel® Core i7-3820 Processor (10M Cache, 3.80 GHz) de 4 núcleos y 8 subprocesos (threads). Además de 32GiB de memoria. (Más detalles en los apéndices).

4.2. Descripción de la solución

Se decidió implementar la solución en un ambiente con infraestructura Hadoop en la nube, esto porque de esta forma es mucho más fácil modificar el número de máquinas que conformarían el clúster y sus características, ya que se puede realizar de forma dinámica.

Sobre Hadoop (HDFS y MapReduce) se agregaron herramientas relacionadas al framework, para facilitar el ingreso de datos, acceso y consultas. Estudiadas las distintas opciones se decidió utilizar las herramientas que ofrece Cloudera para el despliegue de un cluster, ya que provee automáticamente las versiones compatibles y actualizaciones, además de una interfaz de monitoreo. En la sección 4.4: Herramientas/Componentes se da más detalle de estas decisiones.

Además se requirió trabajar con los datos actuales de la empresa y realizar consultas reales, de modo de comparar el desempeño de mejor manera. Para esto se utilizó la herramienta Sqoop, que permite transferir información entre una base de datos y el sistema de archivos de Hadoop (HDFS) directamente, realizando de manera interna la conversión de tipos y formatos.

Para procesar los datos se decidió utilizar Hive e Impala. Hive porque provee una estructura de data warehouse, por lo que la solución modelada no se diferenciaría tanto de la actual y seria mas fácil y transparente migrar los datos y las consultas. Por otro lado Impala provee respuestas en tiempo real, ideal para consultas OLAP. Además, ambos comparten la metadata y no es necesaria ninguna configuración adicional.

Se descartaron las opciones adicionales Spark, Shark y Phoenix mencionadas en el capítulo 2: Antecedentes por un lado, por falta de madurez y documentación y por otro porque no se encuentran aun bien integradas y requiere mucho tiempo realizar cambios adicionales en la configuración del cluster.

Durante el estudio bibliográfico se encontraron distintas fuentes con referencias de pruebas realizadas sobre big data, se decidió basarse en el estudio *Big Data Benchmark*[1] y en el paper *A comparison of approaches to large-scale data analysis* [65] para realizar la creación de pruebas, que contemplan pruebas tipo y pruebas de casos reales (ETL, OLAP). Más detalles en el capítulo Escenarios de Prueba.

Conforme a lo anterior, para realizar las pruebas se tomo un set de datos reales de la empresa, por un lado los datos transaccionales que sirven de fuente de procesos ETL y por otro lado los datos ya procesados que componen en cubo y sirven de fuente para procesamiento en línea.

Una vez almacenados los datos en HDFS estos fueron cargados a Hive e Impala, de manera de darles forma de tabla y poder consultarlos mediante un conector JDBC, de misma manera que se hace actualmente con Infobright.

Una vez preparado el ambiente se creó un esquema de cubo OLAP, y se intentó conectar desde la herramienta Jaspersoft para realizar una vista del cubo. Un esquema de esta solución puede verse en la figura 4.2.

Jaspersoft cuenta con una opción para fuente de datos Hive, pero no está optimizado para Hive 0.10, y a pesar que se instaló el driver necesario para la conexión y que se conectó, no fue posible crear la vista ad hoc del cubo. Esto debido a que Jaspersoft solo soporta una fuente de datos Hadoop Hive para reportes estáticos, y no es posible, por el momento, realizar

[1] realizado por el U.C. Berkeley AMPLab [12]

análisis en linea.

Como vía alternativa de solución al mismo problema se decidió recolectar las consultas realizadas a la base de datos por jaspersoft, de forma de realizar las mismas consultas, pero mediante otra herramienta. En este caso se decidió utilizar SQuirreL SQL, ya que desde esta herramienta se pueden realizar la conexión y las consultas tanto a la base de datos actual Infobright como a Hive e Impala en el clúster.

En la sección Set de Consultas se detallan los sets de consultas realizadas sobre los escenarios.

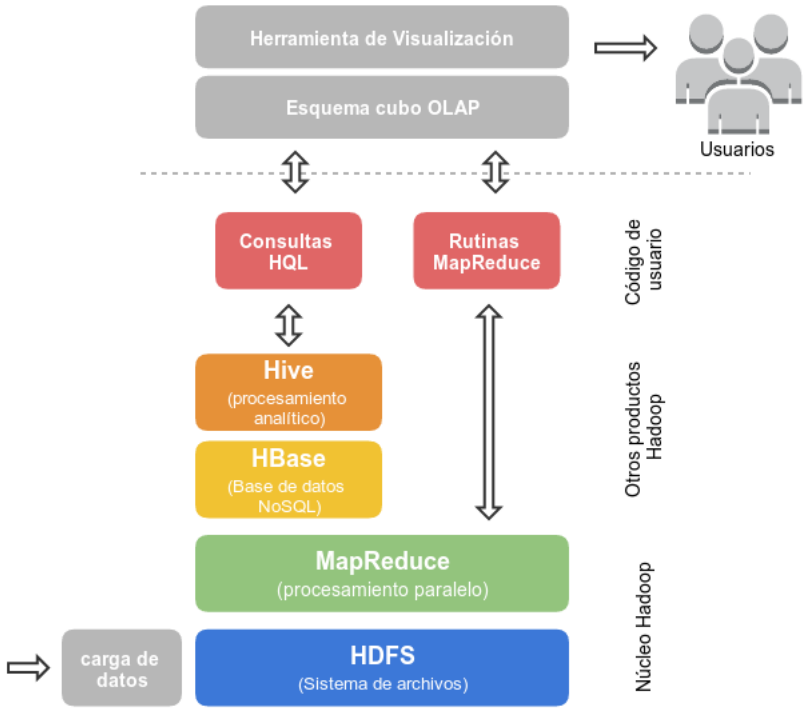

Figura 4.2: Diseño solución OLAP. Fuente: elaboración propia

4.3. Infraestructura

4.3.1. Máquinas

Para la implementación de la solución se requirió un ambiente con Hadoop. La opción más factible fue realizar el desarrollo y pruebas en la nube, específicamente utilizando Amazon Web Services. De esta forma se puede tener control sobre las características y cantidad de las máquinas de forma dinámica, pudiendo realizar pruebas de distintas máquinas hasta encontrar la que satisfaga los requisitos.

Para ejecutar las pruebas se crearon tres instancias EC2, cuyas características se resumen en la tabla 4.1

Todas las instancias fueron creadas con un sistema operativo Ubuntu 12.04.3 LTS (Precise Pangolin) de 64 bit, los procesadores corresponden a procesadores físicos de la familia Intel Xeon. Con un disco duro de 150Gb para los esclavos y 200Gb para el maestro.

El maestro cuenta con 4 CPU virtuales (vCPU) mientras los esclavos con 1 CPU virtual, donde cada vCPU equivale aproximadamente a una CPU física. Además el maestro tiene 8 Unidades de computación EC2 (EC2 Compute Unit o ECU) y los esclavo 2. Cada ECU proporciona la capacidad de una CPU equivalente a un Opteron 2007 o Xeon de 1,0 a 1,2 GHz.

En cuanto a memoria, el maestro cuenta con 15GiB, y los esclavos con 3,75 cada uno.

Nombre	**Sistema Operativo**	**Tipo de Instancia**	**Procesador**	**vCPU**	**ECU**	**Memoria (GiB)**
Master	Ubuntu Server 12.04.3 LTS	m1.xlarge	64 bits	4	8	15
Slave-1	Ubuntu Server 12.04.3 LTS	m1.medium	64 bits	1	2	3,75
Slave-2	Ubuntu Server 12.04.3 LTS	m1.medium	64 bits	1	2	3,75

Tabla 4.1: descripción instancias EC2

Para poder conectar las instancias se decidió crear un VPC, para así tener las instancias dentro de una subred y mantenerlas referenciadas mediante una IP privada. De esta forma no cambia la dirección si detenemos las máquinas y las reiniciamos nuevamente.

Además se configuró el maestro con una IP elástica, ya que debe poder ser accedido desde internet, y no debe cambiar la dirección, para facilitar el uso. En la figura 4.3 se puede apreciar un diagrama de las instancias. En la tabla 4.2 se muestran las direcciones de cada instancia.

Instancia	**Private IP**	**Elastic IP**
Master	10.0.0.62	54.209.56.63
Slave-1	10.0.0.63	
Slave-2	10.0.0.64	

Tabla 4.2: Direcciones instancias EC2

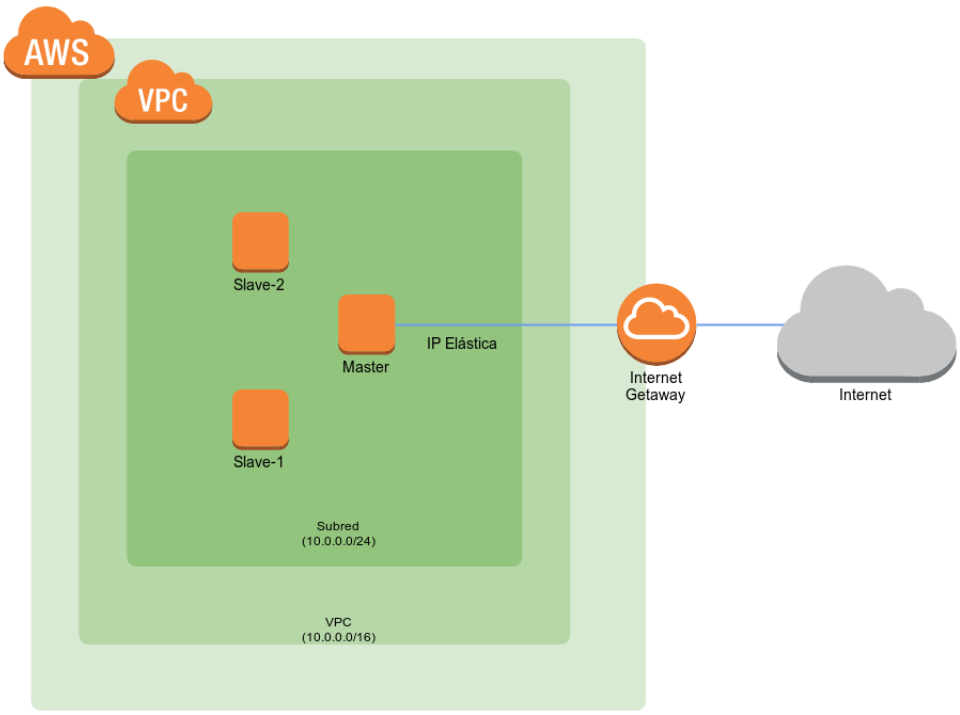

Figura 4.3: Instancias en VPC

4.3.2. Hadoop

Existen distintos proyectos que ofrecen servicios y soporte asociadas a Hadoop, entre ellos se analizaron:

- **Amazon EMR** : Amazon Elastic Map Reduce es un servicio de computación paralela en la nube ofrecido por Amazon, que permite ejecutar trabajos en Hadoop en instancias EC2 pre configuradas.
- **Juju** : Juju es un framework de orquestación para el despliegue de servicios de infraestructura en la nube, permitiendo la instalación orquestada de entornos distribuidos de servicios relacionados. Para lograrlo existen paquetes de servicios (*charms*) que incluyen metadatos del servicio, dependencias, paquetes necesarios, así como la lógica necesaria para gestionar el servicio. De esta forma se puede, desde una máquina, realizar una conexión con instancias EC2 y desplegar servicios Hadoop, armando un cluster con una mínima configuración.
- **Apache BigTop** : Es un proyecto para el desarrollo de paquetes y pruebas del ecosistema Hadoop. Se desarrolla y disponibilizan paquetes, pruebas de integración, entre otros, sobre proyectos de Hadoop según van saliendo nuevas versiones. Existen diversas organizaciones e individuos que utilizan Apache Bigtop como base para sus propias distribuciones.
- **Cloudera** : Cloudera es una compañía de que ofrece software basado en Hadoop, soporte, servicios y capacitación. Incluye CDH (Cloudera Distribution Incuding Apache Hadoop) que integra fácilmente la mayor parte de los componentes de Hadoop (HDFS,

Map/Reduce, Hive, Pig, Mahout, HBase, etc) y Cloudera Manager, la plataforma de administración para CDH que que permite manejar y gestionar el cluster.

Dadas las necesidades del proyecto, se requería poder tener los archivos de manera local en las máquinas (HDFS) y que persistieran luego de detener y reiniciar las instancias, por esta razón se descartó la opción de Amazon EMR. Además, era necesario poder instalar más de un servicio por máquina, por lo que se descartó la opción de juju, que tampoco era compatible con un VPC.

Tras analizar las distintas opciones se decidió utilizar la solución de Cloudera Enterpise Free, que incluye CDH4 junto con Cloudera Manager Free Edition. Este conjunto ofrece una manera fácil de instalar y manejar un cluster Hadoop.

Hadoop se basa en una arquitectura Master/Slave, con nodos del tipo maestro y esclavos. Un cluster tipo tiene un nodo maestro y varios nodos esclavos. Estos se explican a continuación:

Maestro :

A grandes rasgos, el maestro está encargado de la supervisión de las piezas que conforman Hadoop, como el almacenamiento de datos (HDFS) y ejecución de tareas en paralelo (MapReduce).

Esclavos :

Los esclavos constituyen la mayoría de los nodos y son los que realizan el trabajo, como almacenar los datos y ejecutar los cálculos.

Estos tipos de nodos pueden tomar distintos roles, dependiendo las herramientas instaladas y las tareas a realizar. Los posibles roles se describen a continuación:

4.3.3. Roles

HDFS

- Name Node : Supervisa y coordina la función de almacenamiento de datos, manteniendo la metadata de HDFS, información de bloques, información de namespace, etc.
- Secondary Name Node : Asiste al Name Node, haciendo checkpoints en HDFS.
- Data Node : Nodo de datos. Almacena información, recibiendo órdenes del Name Node.

MapReduce

- Job Tracker : Supervisa y coordina el procesamiento paralelo de datos mediante MapReduce.
- Task Tracker : Nodo que acepta trabajos tipo mapReduce desde el JobTracker.

HBase

- HBase Master : (HMaster) es responsable de supervisar las instancias Region Server en un cluster, y es la interfaz para los cambios en los metadatos. En un cluster distribuido se ejecuta normalmente en el Name Node.
- HBase Region Servers : Es responsable de servir y administrar las regiones de las tablas. Se ejecuta junto con el Data Node.
-

Hive

- Hive Metastore : Es un servicio que almacena los metadatos de las tablas y particiones Hive en una base de datos relacional, y ofrece a los clientes (incluyendo hive) acceso a través de una API de servicios.
- Hive Server : Nodo donde se ejecuta Hive.

Zookeeper

- Nodos Zookeeper : Guardan sus datos en un espacio de nombres jerárquico, como hace un sistema de archivos o una datoestructura trie. Los clientes pueden leer y escribir desde/a los nodos y de esta forma tienen un servicio de configuración compartido.

Hue

- Hue server : Nodo donde se ejecuta el servicio web Hue.
- Beeswax : aplicación que permite ejecutar consultas en Hive. Se instala y configura como parte de Hue.

En el cluster construido, el nodo Master realiza las labores de maestro y ejecuta los roles de maestro de los distintos servicios instalados: Name Node, Secondary Name Node, Job Tracker, HBase Master, Hive Metastore Server, Hive Server, Hue Server, Beetwax Server, Zookeeper Server. Adicionalmente es donde está instalado Cloudera Manager. El Master también hace de esclavo, junto a los nodos restantes Slave-1 y Slave-2, ejecutando los roles: Data Node, Task Tracker, HBase Region Server.

Para que el clúster pueda trabajar correctamente se requiere autenticación mediante llave SSH, para que el nodo maestro pueda logearse en los esclavos sin necesidad de insertar un password. Para eso se generó una pareja de claves pública/privada RSA en el maestro y se le pasó a los esclavos.

En la tabla 4.4 se ve un cuadro resumen de las máquinas y los roles que cumplen dentro del cluster.

4.4. Herramientas/Componentes

A continuación se presentan las herramientas con las que se trabajó, tanto para gestión del cluster como para realizar tareas.

Figura 4.4: Roles de las máquinas

Cloudera Manager

Para el manejo del cluster se utilizó la herramienta Cloudera Manager en su versión 4.7. Esta herramienta permite configurar CDH de manera más fácil, ofreciendo una vista en tiempo real de la salud del cluster de forma centralizada, además de estar integrada con herramientas de diagnóstico y reporte. En la imagen 4.5 se puede ver la interfaz del administrador.

CDH

Se utilizó CDH4 (Cloudera Distribution Incuding Apache Hadoop) como distribución de hadoop. De esta forma se asegura que las versiones de los componentes sean compatibles, y que el cluster se puede administrar mediante Cloudera Manager.

Hadoop

Como core de la plataforma se utilizó Hadoop 2.0. Esto incluye Hadoop HDFS y Hadoop MapReduce.

HBase

Como base de datos no relacional CDH4 viene con HBase en su versión 0.94

HCatalog

Se contó con HCatalog como servicio de metadatos. Esta herramienta permite el acceso a los metadatos a herramientas dentro del cluster, proporcionando esquema y tipo de datos compartido para las herramientas hadoop e interoperabilidad a través de herramientas de procesamiento de datos (pig, mapReduce, hive). Se utilizó la versión 0.5

Hue

Interfaz de usuario para herramientas de Hadoop, se utiliza como explorador de archivos en HDFS, explorador de HBase, editor de queries, entre otras cosas. Se utilizó la versión 2.5.

Hive

Como sistema de data warehouse, y para facilitar el acceso y consultas a los datos se utilizó Hive en su versión 0.10.

Zookeeper

Para la coordinación del cluster se contó con la herramienta Zookeeper, que provee un servicio de configuración centralizada y registro de nombres para sistemas distribuidos. Se utilizó en su versión 3.4.5.

Flume

CDH4 incluye Flume 1.4.0 como servicio de gestión de logs distribuidos.

Sqoop

Para la integración de datos CDH4 cuenta con Sqoop 1.4.3. Esta herramienta permite mover datos desde y hacia Hadoop, como bases de datos y almacenes de datos.

4.5. Set de Datos

Los datos a analizar corresponden a la información transaccional de una empresa de retail, correspondiente al registro de ventas, clientes, fechas, productos y sucursales de los últimos 4 años. A continuación se encuentra un resumen de estos, para más detalle ver el capítulo 2: Caso de Estudio.

4.5.1. Set 1: ETL

Corresponde a toda la información histórica de ventas, productos y clientes ingresados durante los últimos 4 años. Está conformado por:

- tabla de hechos (ventasfact) de 281 millones de registros y 18 campos. Almacena la

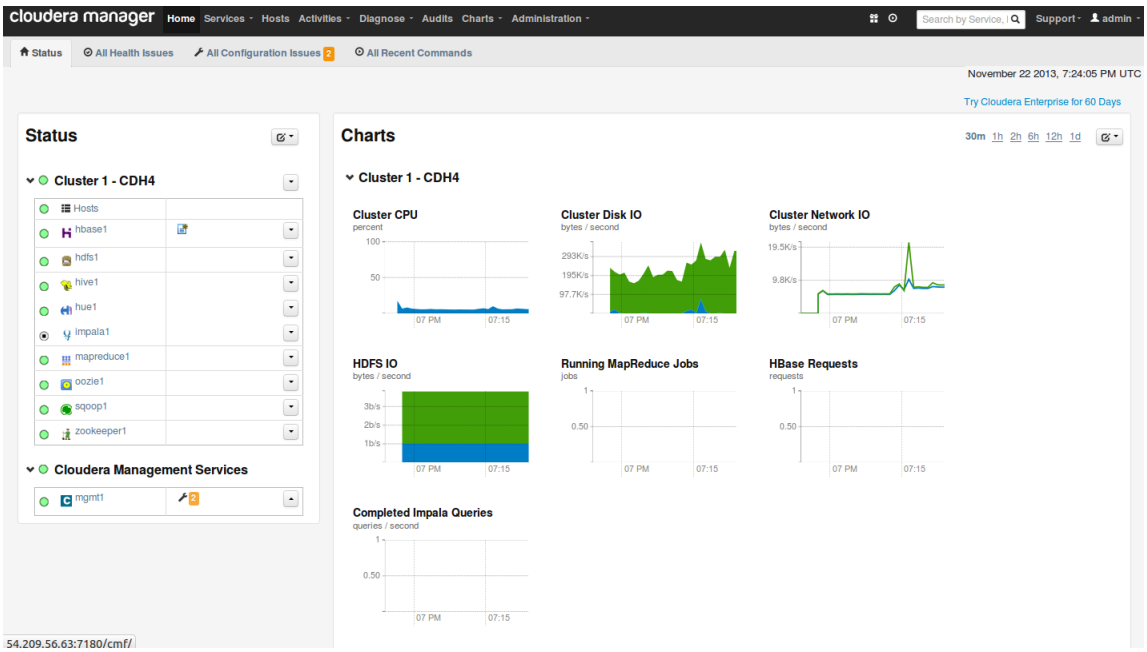

Figura 4.5: Cloudera Manager

información de ventas de 4 años.

- tabla de dimensión de productos (dim_producto) con 18 mil registros y 18 campos. Almacena la información de todos los productos en detalle.
- tabla de dimensión de clientes y etiquetas (dim_cliente_etiquetas) con 6 millones de registros y 24 campos. Almacena la información histórica de los clientes de los últimos 4 años además de información de las etiquetas asociadas a estos.

4.5.2. Set 2: Cubo

En este caso el set de datos está conformado por tablas producidas mediante ETLs, estas cuentan con una versión resumida de las tablas principales, optimizadas para solo contar con la información que se utilizará. Estas tablas son:

- tabla de hechos (fact_ventas) de 62 millones de registros y 13 campos, esta cuenta con la información de las ventas desde enero a octubre del 2013.
- tabla de dimensión cliente (dim_cliente) de 335 mil registros y 8 campos. Este cuenta con la información histórica resumida de los clientes.
- tabla de dimensión de fecha (dim_fecha) de 10 registros y 15 campos. Cuenta con el detalle de las fechas, desde enero hasta octubre del 2013.
- tabla de dimensión de productos (dim_producto) de 3.485 registros y 4 campos. Cuenta con la información resumida de los productos divididos por subgerencia, linea y marca.
- tabla de dimensión de sucursales (dim_sucursal) de 30 registros y 4 campos. Cuenta con la información resumida de las sucursales con el nombre, gerente y región respectivos.

4.6. Escenarios de Prueba

Los escenarios sobre los cuales se desarrollaron las pruebas abarcan distintos aspectos de la solución, a continuación se describen a grandes rasgos cuáles son estos:

Motor de Consulta

Con motor de consulta nos referimos al uso de Impala o Hive para realizar las consultas. Impala se puede describir como SQL sobre HDFS, ya que realiza consultas sobre HDFS mediante MPP(massively parallel processing), sin recurrir a trabajos MapReduce. Por otro lado Hive es SQL sobre Hadoop, ya que consulta los datos almacenados mediante tareas MapReduce. Ambos comparten la metadata, de modo que pueden ver las mismas bases de datos y consultarlas, siempre que el formato sea compatible.

Formato Almacenamiento

Las distintas tecnologías basadas en Hadoop operan con distintos formatos de almacenamiento. Impala y Hive comparten la metadata, por lo que si el formato es compatible los datos pueden ser procesados de igual manera por ambos motores. Sin embargo, existen otros formatos como parquet, optimizado para consultas Impala. A continuación una tabla comparativa con los formatos y compatibilidad:

	Texto	**Sequence**	**RC**	**Parquet**	**Avro**
Hive	si	si	si	si	si
Impala	si	si*	si*	si	si*

* Impala puede crear y consultar en formato Texto y Parquet, pero solo puede consultar datos del tipo Sequence, Avro, RC. Estos tipos de datos deben ser creados en Hive.

Tabla 4.3: compatibilidad de formatos y motores

- **Texto**: El formato de texto es un formato conveniente para ser usado por varias aplicaciones o scripts que producen o leen archivos de texto como CSV o TSV. También es muy flexibles en la definición de las columnas, por ejemplo, si el archivo tiene más campos que los definidos en la tabla, los campos extras se ignoran. Los datos almacenados como texto son voluminosos y no tan eficientes para consultar como formatos binarios.

  CSV: (comma-separated values) es un tipo de documento en formato de texto para representar datos en forma de tabla, en las que las columnas se separan por comas (o punto y coma) y las filas por saltos de línea.

  JSON: (JavaScript Object Notation) Es un formato ligero para el intercambio de datos de la forma atributo-valor. Es usado principalmente para transmitir datos entre un servidor y aplicaciones web.
- **Sequence**:Tradicionalmente Hadoop almacena los datos internamente en archivos de secuencia planos, que es un formato de almacenamiento binario para pares llave-valor. Tiene la ventaja de ser más compacto que el texto y que se ajusta bien a MapReduce. No es óptimo para Hive ya que almacena toda la fila como un solo valor binario, por lo que se debe leer la fila completa aunque solo se solicite una columna.

  Los contra de este formato de almacenamiento es que es lento de leer y escribir, no se

pueden dividir los archivos comprimidos y es necesario leer y descomprimir todos los campos.

- **RC**: RCFile (Record Columnar File) es una combinación sistemática de componentes, incluyendo el formato de almacenamiento de datos, enfoque de compresión y técnicas de optimización para lectura de datos. Divide los datos horizontalmente en grupos de filas, uno o varios grupos se almacenan en un archivo de HDFS, almacenándose los datos del grupo en un formato de columnas. El beneficio de esta organización es que se aplica el paralelismo de hadoop, ya que los grupos de filas están en diferentes archivos distribuidos redundantemente y son procesables al mismo tiempo. De esta forma cada nodo procesa solamente las columnas correspondientes y se salta las irrelevantes. Además la compresión por columna es más eficiente, y se puede aprovechar la similitud de datos por columna.
- **Parquet**: Es un formato de archivo binario orientado a columnas, optimizado para Impala. Es particularmente bueno para consultas de exploración de columnas particulares, como funciones de agregación (SUM, AVG), permitiendo realizar análisis rápidamente y con un E/S mínimo. Dentro del archivo los valores de cada columna están organizados de forma adyacente, permitiendo buena compresión de los valores de la columna. El rendimiento de las consultas depende del número de columnas necesarias para procesar las cláusulas SELECT y WHERE, la forma en que se dividen los datos, la descompresión de cada columna y si se encuentran particionados los datos.
- **Avro**: Es un formato de datos binario, basado en schemas. Es compatible con Hive, permitiendo leer y escribir datos en Avro como tablas, transformando automáticamente los tipos de datos en equivalentes a Hive.

Compresión

También se debe tener en cuenta el nivel de compresión sobre los datos, dentro de las posibilidades se encuentran:

- **Ninguno**: Datos sin compresión, el almacenamiento requiere más espacio pero la utilización de CPU y tiempo de procesamiento disminuye, ya que no se deben realizar tareas de compresión/descompresión.
- **Gzip**: Es una aplicación de software para comprimir/descomprimir archivos. Basado en el algoritmo DEFLATE, combinación de LZ77 y Huffman coding. Buena compresión pero un poco lento. No divisible. Recomendado cuando se desea el mayor nivel de compresión (máximo ahorro de espacio).
- **Bzip2**: programa de compresión que utiliza el algoritmo Burrows-Wheeler. Es más lento que gzip pero de mejor compresión y divisible.
- **Snappy**: Es una librería de compresión/descompresión desarrollada por Google y basada en el algoritmo LZ77. Fue diseñada para ser rápida y estable, pero no para alcanzar un alto ratio de compresión. Muy utilizada en proyectos Google como BigTable y MapReduce. El balance entre ratio de compresión y velocidad de descompresión es muy bueno.
- **LZO**: algoritmo de compresión enfocado en la velocidad de descompresión. La velocidad de compresión es comparable a gzip, pero la descompresión es muy rapida. Divisible.

	Impala	Hive
Parquet	Snappy, Gzip. (Snappy por defecto)	??
Texto	LZO	LZO, Snappy, Gzip, Bzip2
Avro	Snappy, Gzip, Bzip2	Snappy, Gzip, Bzip2
RC	Snappy, Gzip, Bzip2	Snappy, Gzip, Bzip2
Sequence	Snappy, Gzip, Bzip2	Snappy, Gzip, Bzip2

Tabla 4.4: algoritmos de compresión soportados motor y formato de archivo

Dadas las anteriores descripciones y teniendo en cuenta las limitaciones de la herramienta de importación (Sqoop) se realizó un análisis de las combinaciones a utilizar dentro de las pruebas a ejecutar.

Se decidió que la mejor manera de importar los datos era directamente a Hive mediante Sqoop utilizando Snappy como compresor, siendo esta una primera combinación a utilizar. Una vez importados los datos se crearon tablas similares, pero con los datos descomprimidos, constituyendo la segunda combinación. Y finalmente se pasaron estos datos a Parquet (con Snappy como la compresión por defecto) siendo esta la tercera combinación.

De esta forma se constituyeron los escenarios finales de prueba, utilizando las combinaciones de tipos de datos y compresión, con los distintos motores. La tabla a continuación lo resume.

	Formato	Compresión	Motor
Escenario 1	Texto	Snappy	Hive
Escenario 2	Texto	-	Hive
Escenario 3	Texto	-	Impala
Escenario 4	Parquet	Snappy	Impala

Tabla 4.5: Escenarios de prueba

4.7. Set de Consultas

Las consultas de prueba a realizar se basaron en dos modelos. Por un lado están las consultas genéricas o prototipo, basadas en las pruebas del estudio de la Universidad de Berkeley *Big Data Benchmark* [12], en donde se analiza el tiempo de respuesta de distintas soluciones de Big Data sobre consultas relacionales: escaneo, agregación, joins.

Por otro lado se encuentran las consultas reales, basadas en consultas que se realizan actualmente sobre los datos de la empresa. Estas constan de consultas en tiempo real, realizadas mediante el uso de los reportes ad-hoc a medida que se inspecciona el cubo sobre el Set 1: ETL, y procesamiento por lotes, que corresponde a las consultas realizadas por el ETL sobre el Set 2: Cubo.

Las consultas fueron divididas por tipo y variante, según el tipo de consulta y el tamaño de datos que abarca.

4.7.1. Tipo 1: Escaneo

Consultas de tipo exploratoria. Este tipo de query escanea, filtra y recupera set de datos de una tabla y almacena los resultados en otra. Estas consultas principalmente prueba el rendimiento con el que cada solución puede leer y escribir en una tabla de datos.

En este caso la consulta consiste en la selección de ventas realizadas después de una fecha variable:

```
SELECT tran_id, date_id, prod_id
FROM fact_ventas
WHERE date_id > X
```

Se definen tres variantes, dependiendo de la cantidad de resultados obtenidos, esto para probar la capacidad de escalamiento de cada sistema.

Variante A: BI-Like

```
SELECT tran_id, date_id, prod_id
FROM fact_ventas
WHERE date_id > 2191
```

Los resultados son pequeños, pudiendo ser almacenados en memoria en una herramienta de Inteligencia de Negocios.

Variante B: Intermedia

```
SELECT tran_id, date_id, prod_id
FROM fact_ventas
WHERE date_id > 2009
```

El tamaño de los resultados es mediano, y podrían no ser almacenados en memoria en un único bloque.

Variante C: ETL-Like

```
SELECT tran_id, date_id, prod_id
FROM fact_ventas
WHERE date_id > 1828
```

El tamaño de los resultados son grandes y requieren varios bloques para ser almacenados.

4.7.2. Tipo 2: Agregación

Estas consultas realizan un cálculo sobre un conjunto de datos y regresan un solo valor.

En este caso la consulta consiste en el cálculo del monto promedio de compra por cliente, agrupando el resultado por cliente. Para variar el tamaño de los datos, la consulta se realiza sobre un rango de fechas variables:

```
SELECT ventasfact.cli_id, AVG(ventasfact.meas_monto)
FROM consorcio_alvi_bdm.ventasfact ventasfact
WHERE ventasfact.date_id>X
GROUP BY ventasfact.cli_id
ORDER BY ventasfact.cli_id
```

Este tipo de consulta también contempla las variantes:

Variante A:

```
SELECT ventasfact.cli_id, AVG(ventasfact.meas_monto)
FROM consorcio_alvi_bdm.ventasfact ventasfact
WHERE ventasfact.date_id>2191
GROUP BY ventasfact.cli_id
ORDER BY ventasfact.cli_id
```

Variante B:

```
SELECT ventasfact.cli_id, AVG(ventasfact.meas_monto)
FROM consorcio_alvi_bdm.ventasfact ventasfact
WHERE ventasfact.date_id>2009
GROUP BY ventasfact.cli_id
ORDER BY ventasfact.cli_id
```

Variante C:

```
SELECT ventasfact.cli_id, AVG(ventasfact.meas_monto)
FROM consorcio_alvi_bdm.ventasfact ventasfact
WHERE ventasfact.date_id>1828
GROUP BY ventasfact.cli_id
ORDER BY ventasfact.cli_id
```

4.7.3. Tipo 3: Join

Corresponden a consultas combinadas, que permiten consultar y ordenar información que se encuentra distribuida en varias tablas mediante joins.

En esta consulta se realiza el cálculo del promedio de ventas por usuario, también mostrará la razón social del cliente, que se encuentra en otra tabla. Nuevamente se realiza sobre tres intervalos de fecha para variar el tamaño de los resultados.

```
SELECT dim_cliente_etiquetas.cli_razon_social, AVG(ventasfact.
   meas_monto)
FROM consorcio_alvi_bdm.ventasfact ventasfact INNER JOIN
   consorcio_alvi_clientes_etiquetas.dim_cliente_etiquetas
   dim_cliente_etiquetas ON ventasfact.cli_id = dim_cliente_etiquetas
   .cli_id
WHERE date_id>X
GROUP BY dim_cliente_etiquetas.cli_razon_social
```

```
ORDER BY ventasfact.cli_id
```

Variante A:

```
SELECT dim_cliente_etiquetas.cli_razon_social, AVG(ventasfact.
   meas_monto)
FROM consorcio_alvi_bdm.ventasfact ventasfact INNER JOIN
   consorcio_alvi_clientes_etiquetas.dim_cliente_etiquetas
   dim_cliente_etiquetas ON ventasfact.cli_id = dim_cliente_etiquetas
   .cli_id
WHERE date_id>2191
GROUP BY dim_cliente_etiquetas.cli_razon_social
ORDER BY ventasfact.cli_id
```

Variante B:

```
SELECT dim_cliente_etiquetas.cli_razon_social, AVG(ventasfact.
   meas_monto)
FROM consorcio_alvi_bdm.ventasfact ventasfact INNER JOIN
   consorcio_alvi_clientes_etiquetas.dim_cliente_etiquetas
   dim_cliente_etiquetas ON ventasfact.cli_id = dim_cliente_etiquetas
   .cli_id
WHERE date_id>2009
GROUP BY dim_cliente_etiquetas.cli_razon_social
ORDER BY ventasfact.cli_id
```

Variante C:

```
SELECT dim_cliente_etiquetas.cli_razon_social, AVG(ventasfact.
   meas_monto)
FROM consorcio_alvi_bdm.ventasfact ventasfact INNER JOIN
   consorcio_alvi_clientes_etiquetas.dim_cliente_etiquetas
   dim_cliente_etiquetas ON ventasfact.cli_id = dim_cliente_etiquetas
   .cli_id
WHERE date_id>1828 GROUP BY dim_cliente_etiquetas.cli_razon_social
ORDER BY ventasfact.cli_id
```

4.7.4. Tipo 4: Consultas reales OLAP

Corresponden a consultas reales realizadas sobre el Set 1: ETL, mediante el cubo OLAP a medida que se va bajando en la jerarquía del cubo.

Se escogieron estas consultas ya que son representativas de lo que el cliente requiere, en este caso el ticket promedio por cliente. Además, tienen nivel de complejidad que va aumentando a medida que se va profundizando en las dimensiones y van tomando mas tiempo.

Se dividen en tres variantes, desde la más genérica a la más específica.

Variante A: Ticket Promedio Periodo Actual por Segmento Clientes y Subgerencia Productos

```
SELECT dim_clientes_nov.cli_etiqueta_bimensual AS segmento,
   dim_producto.prod_subgerencia AS subgerencia, dim_producto.
   prod_linea AS categoria, sum(fact_ventas.meas_monto)/count(
   DISTINCT fact_ventas.tran_id_13 ) AS ticketPromedio
FROM dim_cliente AS dim_cliente, fact_ventas AS fact_ventas,
   dim_producto AS dim_producto
WHERE fact_ventas.cli_id = dim_cliente.cli_id AND fact_ventas.prod_id
    = dim_producto.prod_id
GROUP BY segmento, subgerencia, categoria
```

Variante B: Ticket Promedio Periodo Actual por Segmento Clientes, Subgerencia Productos, Categoría Productos

```
SELECT dim_cliente.cli_etiqueta_bimensual AS segmento, dim_producto.
   prod_subgerencia AS subgerencia, dim_producto.prod_linea AS
   categoria, dim_producto.prod_marca AS marca, sum( fact_ventas.
   meas_monto ) / count( DISTINCT fact_ventas.tran_id_13 ) AS
   ticketPromedio
FROM dim_cliente AS dim_cliente, fact_ventas AS fact_ventas,
   dim_producto AS dim_producto
WHERE fact_ventas.cli_id = dim_cliente.cli_id AND fact_ventas.prod_id
    = dim_producto.prod_id
GROUP BY segmento, subgerencia, categoria, marca
```

Variante C: Ticket Promedio Periodo Actual por Segmento Clientes, Subgerencia Productos, Categoría Productos, Marca Productos

```
SELECT dim_sucursal.sucu_region AS sucursal, dim_cliente.
   cli_etiqueta_bimensual AS segmento, dim_producto.prod_subgerencia
   AS subgerencia, dim_producto.prod_linea AS categoria, dim_producto
   .prod_marca AS marca, sum( fact_ventas.meas_monto ) / count(
   DISTINCT fact_ventas.tran_id_13 ) AS ticketPromedio
FROM dim_sucursal AS dim_sucursal, dim_cliente AS dim_cliente,
   fact_ventas AS fact_ventas, dim_producto AS dim_producto
WHERE fact_ventas.sucu_id = dim_sucursal.sucu_id AND fact_ventas.
   cli_id = dim_cliente.cli_id AND fact_ventas.prod_id = dim_producto
   .prod_id
GROUP BY sucursal , segmento , subgerencia , categoria , marca
```

4.7.5. Tipo 5: Consultas reales ETL

Se refiere a consultas reales realizadas sobre el Set 2: Cubo. Estas corresponden a la extracción de datos de distintas tablas, transformarlos y cargar los resultados en otra tabla.

Se decidió realizar esta prueba ya que es representativa y es una de las pruebas que toman

más tiempo, ya que abarca una gran cantidad de datos. Además, es de gran importancia dentro de la empresa.

Para esta prueba en particular se realiza una parte de la consulta ETL actual, que consiste en extraer datos de ventas, clientes y productos para un periodo específico, cruzar los datos y calcular valores como monto y costo total. Finalmente se crea una tabla donde se almacenan los resultados en el formato deseado, reuniendo y resumiendo los datos de las fuentes para el periodo seleccionado.

Se compone de 4 partes que se describen a continuación:

- Creación y poblamiento de tabla dim_estado_compra_0_0 . Esta query obtiene un resumen de las ventas. Para eso hace un join de ventasfact con dos tablas, clientes_etiqueta y dim_producto, para obtener los datos de las ventas, cliente, producto y cálculo de valores como monto y costo total.
- Creación y poblamiento de tabla dim_estado_compra_0_1 . Esta consulta realiza un join entre la tabla anterior y la tabla que contiene un resumen de los clientes actuales, para crear una tabla aún más detallada con datos del cliente.
- Creación y poblamiento de tabla dim_estado_compra_0_4 . En esta query se crea una nueva tabla, similar a la anterior pero que contiene nuevas columnas para indicar el monto, costo y unidades durante el periodo actual y anterior. Además de la columna tiene_monto_actual que indica si la transacción es actual.
- Finalmente la tabla dim_estado_compra_mensual_0_5 realiza una suma sobre los montos para sacar el total y hace un cast sobre estos para darles formato, agrupando por cliente, y producto.

Estas consultas también se divide en tres tipos, variando el periodo de la consulta:

Variante A: Consulta por mes : El ETL contempla el mes de Julio del 2013.

Variante B: Consulta por trimestre : El ETL contempla el cuarto trimestre del 2013, es decir desde Julio a Septiembre.

Variante C: Consulta por semestre : El ETL contempla el segundo semestre del 2013, es decir desde Julio a Diciembre.

Las consultas completas se encuentran en los Apéndices.

Capítulo 5

Resultados experimentales y Análisis : Comparación de desempeño entre Infobright, Hive e Impala

Para poder comparar el desempeño de las soluciones se realizaron mediciones de tiempo de ejecución y de espacio de almacenamiento de los resultados en los distintos ambientes, de modo de poder contrastarlos con la solución existente.

En específico, se llevaron a cabo las pruebas descritas en la sección Set de Consultas sobre los escenarios descritos en la sección Escenarios de Prueba y sobre el escenario actual.

Las pruebas fueron ejecutadas de manera secuencial para asegurar el acceso y recursos exclusivos y evitar una sobrecarga de tareas en paralelo. Luego de cada ejecución se verificó el estado de los nodos y la integridad de los datos. Cada uno de los tiempos de ejecución representa el promedio de tres ejecuciones.

Además, para las soluciones Hadoop se utilizó un índice de replicación igual a tres, que es el mínimo recomendado y la cantidad de nodos del cluster.

En el siguiente capítulo se presentan las tablas, gráficos e imágenes de los resultados obtenidos de la ejecución de las distintas pruebas divididas por tipo. Posteriormente y en base a lo anterior se desarrolla el análisis.

5.1. Resultados

5.1.1. Tipo 1: Escaneo

Esta consulta realiza una exploración de toda la tabla, seleccionando y almacenando los resultados que cumplen con el rango de fechas. Se debe tener en cuenta que las variantes

simbolizan el crecimiento en la cantidad de registros a obtener.

La figura 5.1 y la tabla 5.1 muestra el tiempo (en segundos) que tomaron las consultas en realizarse mientras que la figura 5.2 y tabla 5.2 la cantidad de registros resultantes y el espacio (en MB) utilizado en cada escenario.

Tiempo

	Hive+Snappy	Hive+sc	Impala+Parquet	Impala+sc	Infobright
A	241,25	252,00	4,75	18,50	0,22
B	332,00	387,00	10,00	42,25	19,06
C	492,00	529,00	17,50	68,25	36,61

Tabla 5.1: Tiempo de ejecución de consultas tipo 1

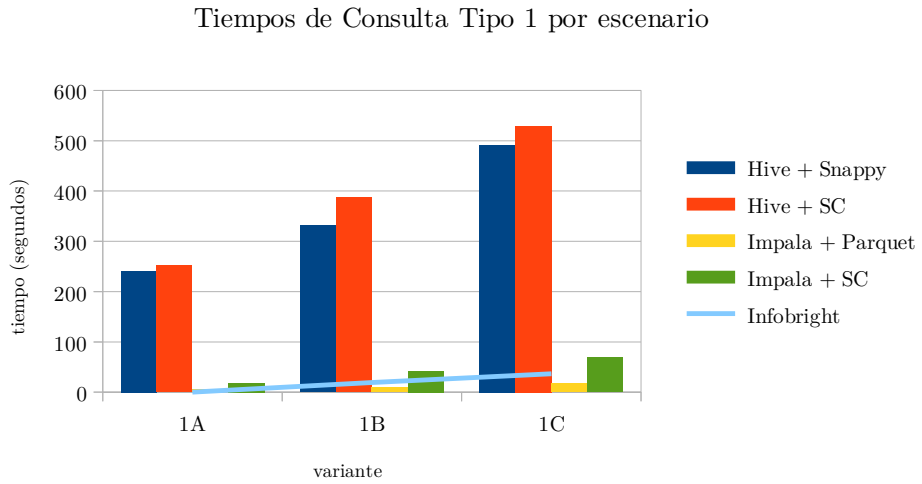

Figura 5.1: Gráfico de tiempo (segundos) por variante de consulta tipo 1. Fuente: elaboración propia

Almacenamiento

	# Registros	Snappy	Sin Comprimir	Parquet	Infobright
A	270442	1,70	5,00	0,89	0,53
B	37131481	262,50	685,90	127,70	91,93
C	72179170	504,70	1333,00	234,10	176,85

Tabla 5.2: Almacenamiento (MB) utilizado en consultas tipo 1

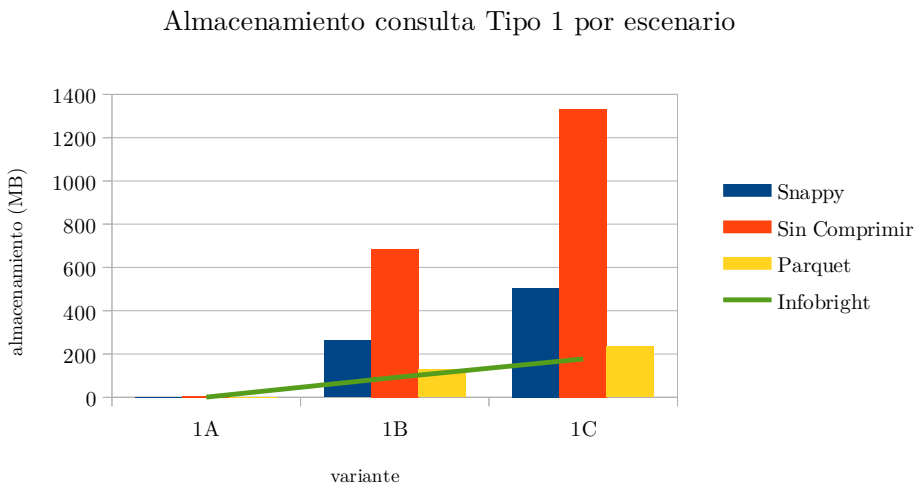

Figura 5.2: Gráfico de almacenamiento utilizado por variante de consulta tipo 1. Fuente: elaboración propia

Esta consulta prueba principalmente el rendimiento con el que cada escenario puede leer y escribir datos en una tabla.

Se observa (figura 5.1) que Infobright parte siendo la solución que toma menos tiempo (variante A) donde la cantidad de registros es menor. Sin embargo, a medida que va creciendo el conjunto de datos el escenario Impala+Parquet va tomando ventaja, debido a la compresión columnar, que permite pasar por alto los campos que no son necesarios. El escenario Impala con datos sin comprimir se va alejando cada vez más del escenario con Parquet, debido al aumento en el peso de los datos (figura 5.2) que debe ir persistiendo en el disco. Los escenarios con Hive se alejan por mucho de los demás.

5.1.2. Tipo 2: Agregación

Las consultas del tipo agregación ejecutan un escaneo sobre una tabla, mientras que realizan cálculos sobre una columna y agrupa según los resultados.

La figuras 5.3, 5.4 y la tabla 5.3 reflejan el tiempo (en segundos) de las consultas por variante, mientras que la figura 5.5 y tabla 5.4 la cantidad de registros resultantes y el espacio (en MB) utilizado en cada escenario.

Tiempo

	Hive+Snappy	Hive+sc	Impala+Parquet	Impala+sc	Infobright
A	247,75	256,00	10,25	15,00	0,18
B	367,50	391,00	8,00	12,00	1,56
C	480,25	512,00	7,50	13,75	2,37

Tabla 5.3: Tiempo de ejecución de consultas tipo 2

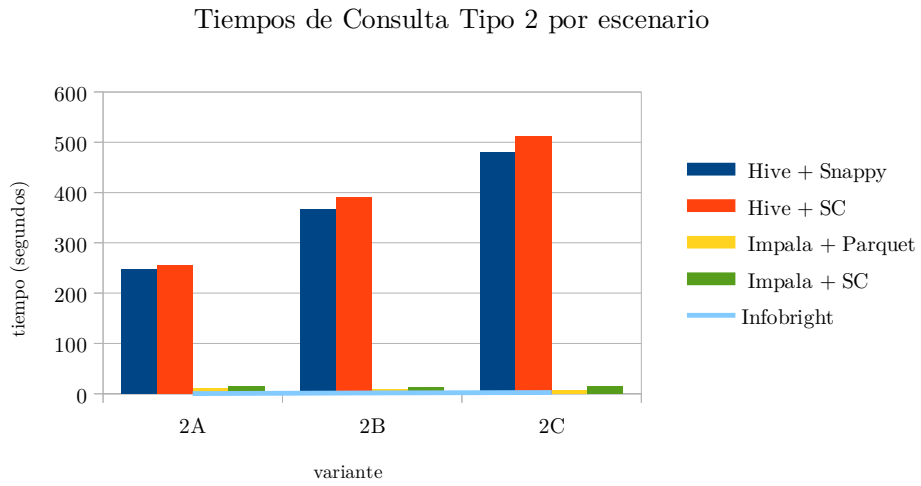

Figura 5.3: Gráfico de tiempo (segundos) por variante de consulta tipo 2. Fuente: elaboración propia

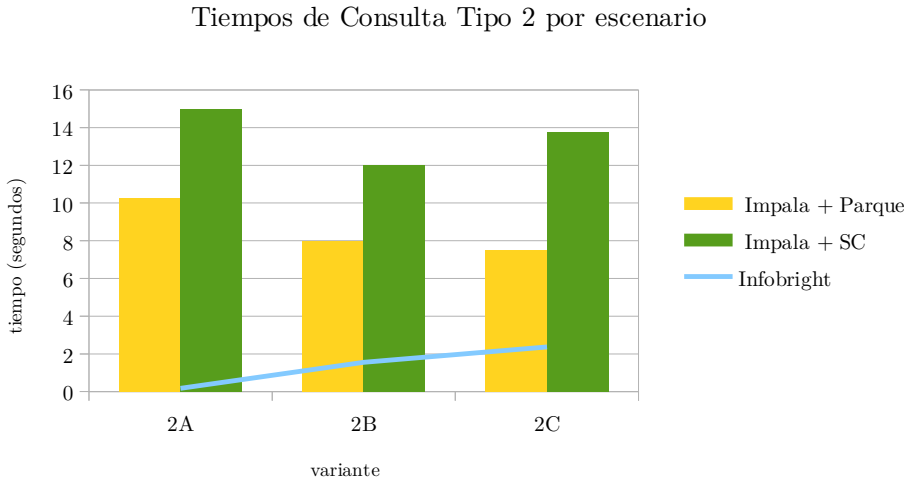

Figura 5.4: Gráfico de tiempo (segundos) por variante de consulta tipo 2, excluido Hive. Fuente: elaboración propia

Almacenamiento

	# Registros	Snappy	Sin Comprimir	Parquet	Infobright
A	19855	0,23	0,37	0,23	0,08
B	290973	4,00	6,20	3,10	1,78
C	361168	5,10	7,80	3,80	2,22

Tabla 5.4: Almacenamiento (MB) utilizado en consultas tipo 2

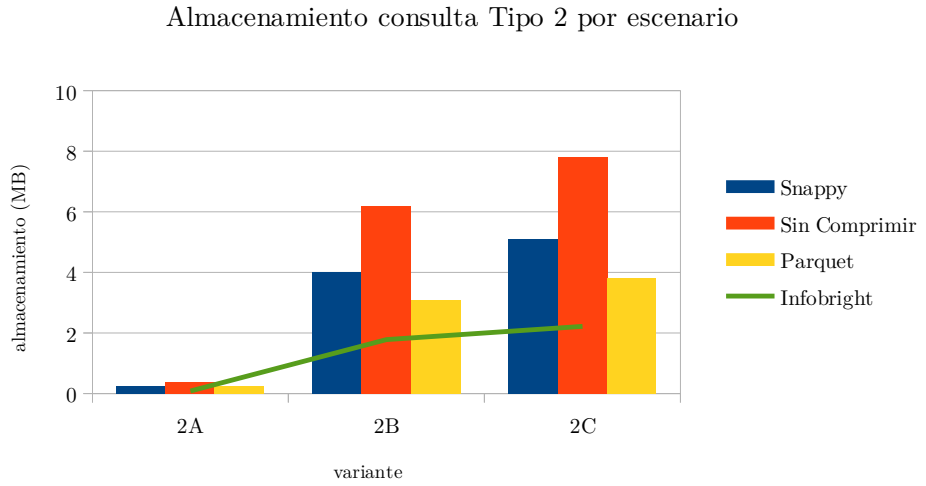

Figura 5.5: Gráfico de almacenamiento utilizado por variante de consulta tipo 2. Fuente: elaboración propia

En esta consulta se aplica la función promedio a los montos y luego agrupa por cliente, dentro del rango de fechas descrito.

Se observa en los resultados que nuevamente los escenarios con motor Hive se alejan por mucho del resto (figura 5.3), por lo que en la imagen 5.4 se grafican los tiempos sin este escenario.

En este tipo de consulta Infobright obtiene el menor tiempo de respuesta, mientras que las consultas con Impala obtienen un tiempo mayor, pero más constante. Al parecer la aplicación de la función y la agregación presentan un costo base mayor para Impala. Nuevamente la compresión columnar presenta una ventaja frente al formato sin comprimir.

5.1.3. Tipo 3: Join

Esta consulta realiza un cruce entre una tabla grande (ventasfact) y una tabla más pequeña (dim_cliente) y luego ordena los resultados.

La figuras 5.6, 5.7 y la tabla 5.5 reflejan el tiempo (en segundos) de las consultas por variante, mientras que la figura 5.8 y tabla 5.6 la cantidad de registros resultantes y el espacio (en MB) utilizado en cada escenario.

Tiempo

	Hive+Snappy	**Hive+sc**	**Impala+Parquet**	**Impala+sc**	**Infobright**
Mes	326,75	338	14,25	27	3,094
Trimestre	1346,25	1334	25,5	37	112,72
Semestre	2280,25	2300	37,25	47,75	206,78

Tabla 5.5: Tiempo de ejecución de consultas tipo 3

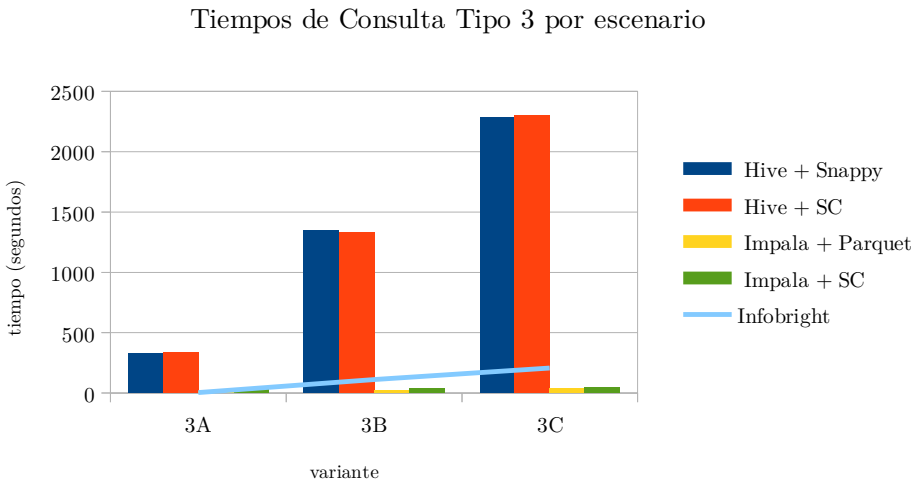

Figura 5.6: Gráfico de tiempo (segundos) por variante de consulta tipo 3. Fuente: elaboración propia

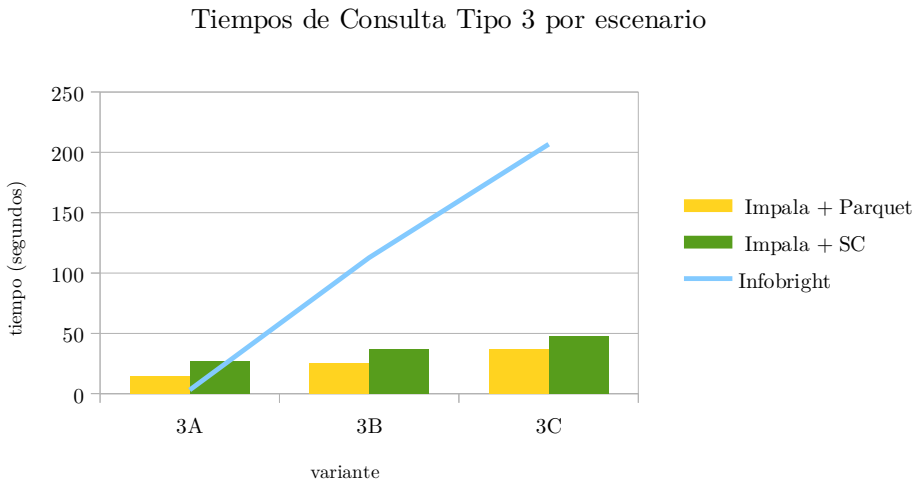

Figura 5.7: Gráfico de tiempo (segundos) por variante de consulta tipo 3, excluido Hive. Fuente: elaboración propia

Almacenamiento

	# Registros	Snappy	Sin Comprimir	Parquet	Infobright
A	10727	0,22	0,37	0,22	0,12
B	101830	2,40	2,90	2,10	1,38
C	119889	2,90	4,60	2,50	1,62

Tabla 5.6: Almacenamiento (MB) utilizado en consultas tipo 3

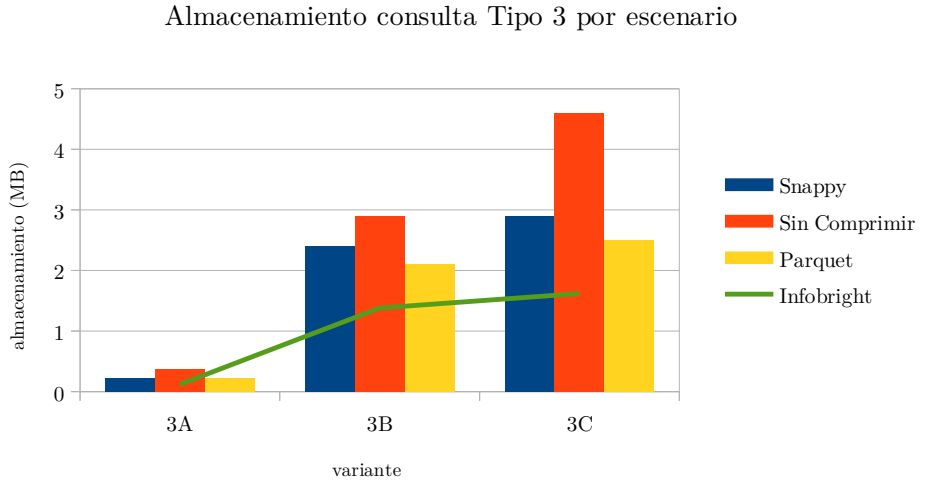

Figura 5.8: Gráfico de almacenamiento utilizado por variante de consulta tipo 3. Fuente: elaboración propia

Cuando se realiza la unión de las tablas sobre el conjunto menor (variante A) los motores pasan la mayor parte del tiempo escanendo los datos y realizando comparaciones de fecha. Para uniones mas grandes, la exploración inicial se convierte en una fracción menos significativa del tiempo de respuesta total.

En los resultados se notar que una vez superado el coste inicial, la brecha entre los tiempos de Impala son menores a las demás (figura 5.7). Infobright parte con un tiempo muy pequeño, pero la brecha va a aumentando en mayor medida. Los tiempos del motor Hive siguen superando a los demás, siendo mayores a 12 veces el más lento de los demás (figura 5.6).

5.1.4. Tipo 4: OLAP

En estas consultas se pretende emular el desempeño de una consulta tipo OLAP o en tiempo real, por lo que no se almacenan los resultados, sino que solo se mide el tiempo de consulta.

Los tiempos de respuesta se muestran en la tabla 5.7 y en los gráficos 5.9 y 5.10. Además, se presentan los gráficos de desempeño generados por Cloudera Manager tanto para el estado del cluster como del nodo Maestro.

Cabe mencionar que para las variantes B y C en el motor Impala se debieron hacer cambios en el tipo de instancia de los nodos esclavos, igualando al maestro.

Tiempo

	Hive+Snappy	**Hive+sc**	**Impala+Parquet***	**Impala+sc***	**Infobright**
A	2041,20	1886,66	1,26	1,25	54,74
B	2140,53	1881,61	1,42	1,35	61,99
C	2185,52	1983,27	1,39	1,29	64,36

* Variantes B y C ejecutadas con nodos esclavos iguales al master.

Tabla 5.7: Tiempo de ejecución de consultas tipo OLAP

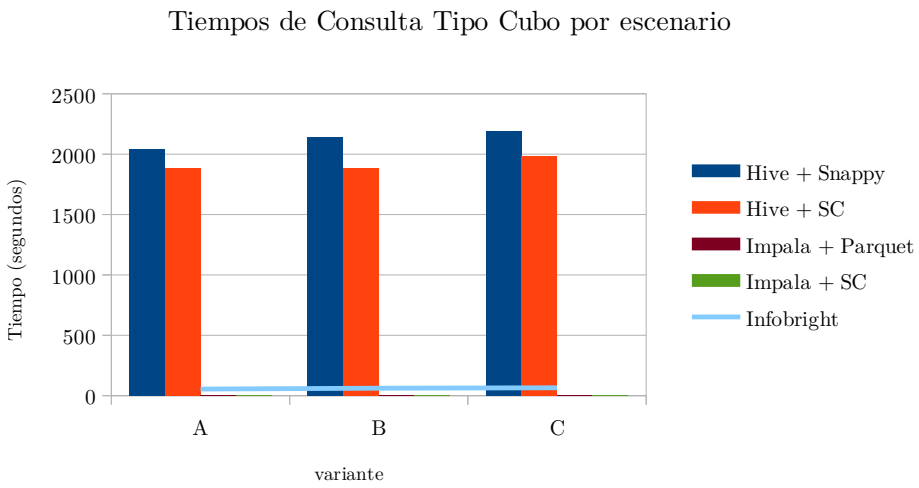

Figura 5.9: Gráfico de tiempo (segundos) por variante de consulta tipo OLAP. Fuente: elaboración propia

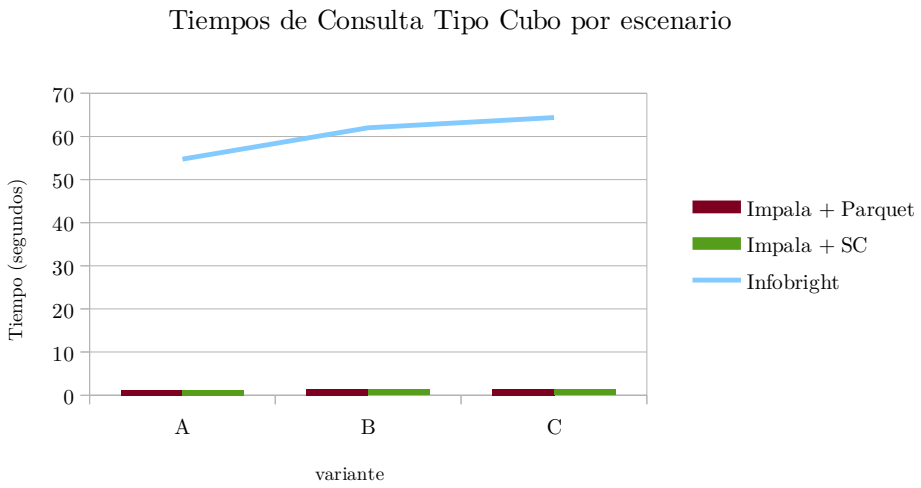

Figura 5.10: Gráfico de tiempo (segundos) por variante de consulta tipo OLAP, excluyendo Hive. Fuente: elaboración propia

Desempeño Cluster

En las siguientes imágenes se grafica a nivel de clúster el uso de CPU, IO y E/S durante el desarrollo de las queries, dividido por escenario de ejecución.

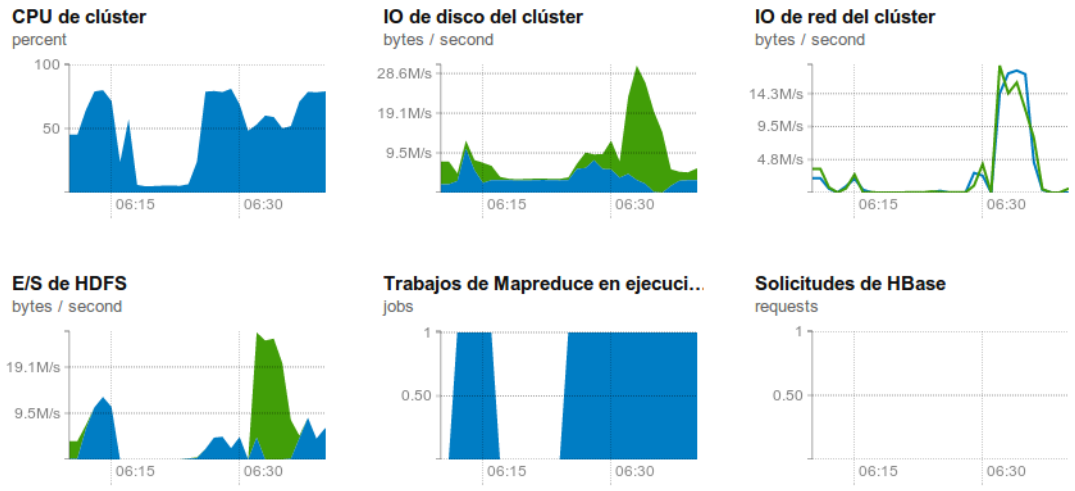

Figura 5.11: Estado cluster durante consulta tipo OLAP en escenario Hive-Snappy. Fuente: copia de pantalla Cloudera Manager

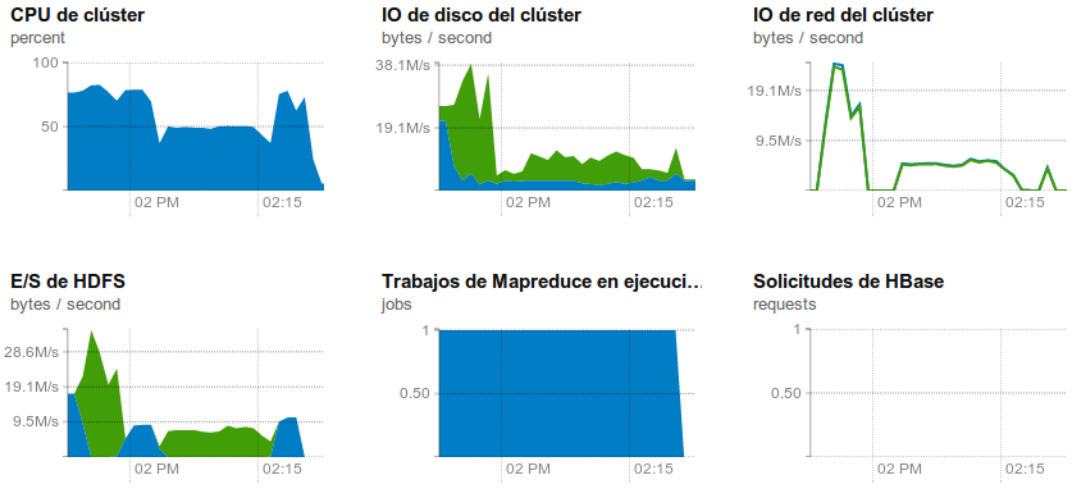

Figura 5.12: Estado cluster durante consulta tipo OLAP en escenario Hive-Sin Comprimir. Fuente: copia de pantalla Cloudera Manager

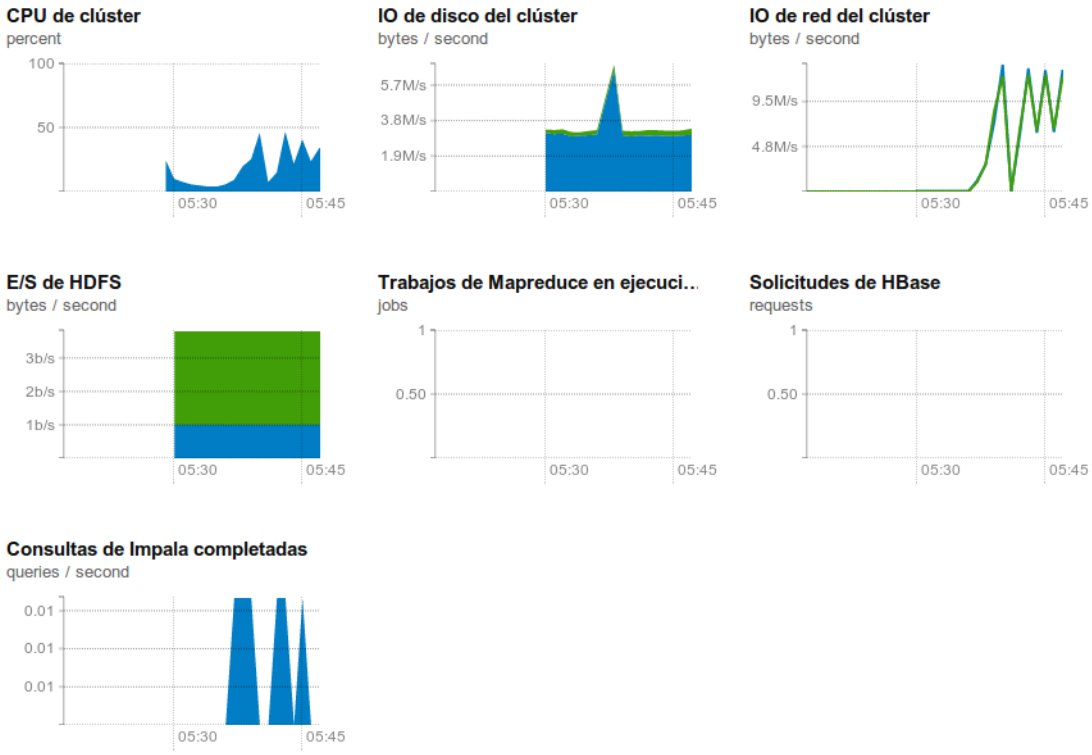

Figura 5.13: Estado cluster durante consulta tipo OLAP en escenario Impala-Parquet. Fuente: copia de pantalla Cloudera Manager

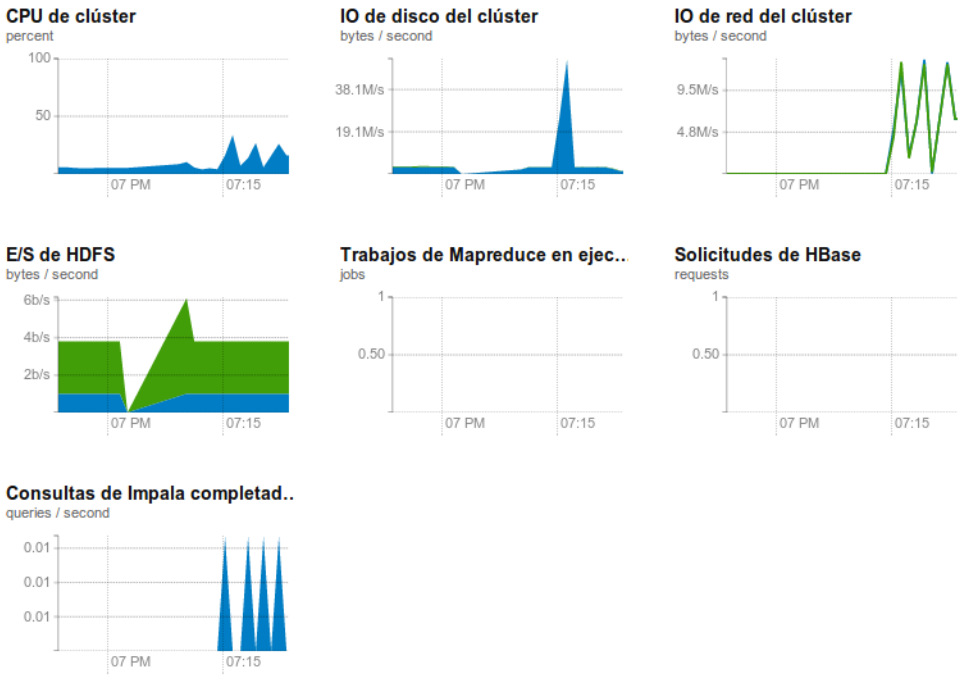

Figura 5.14: Estado cluster durante consulta tipo OLAP en escenario Impala-Sin Comprimir. Fuente: copia de pantalla Cloudera Manager

A continuación, se muestra el uso de CPU, Memoria y Disco en el nodo maestro durante la ejecución de las consultas en los distintos ambientes.

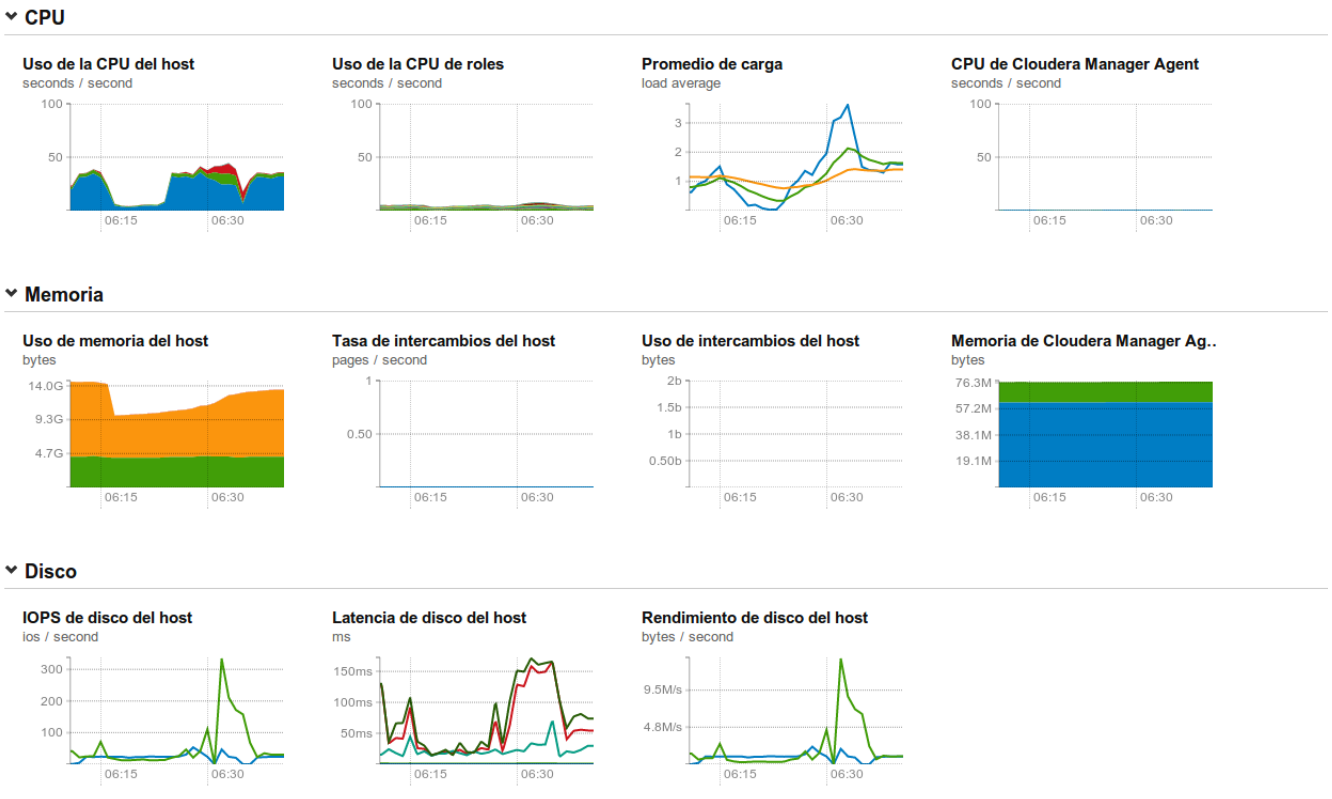

Figura 5.15: Estado nodo maestro durante consulta tipo OLAP en escenario Hive-Snappy. Fuente: copia de pantalla Cloudera Manager

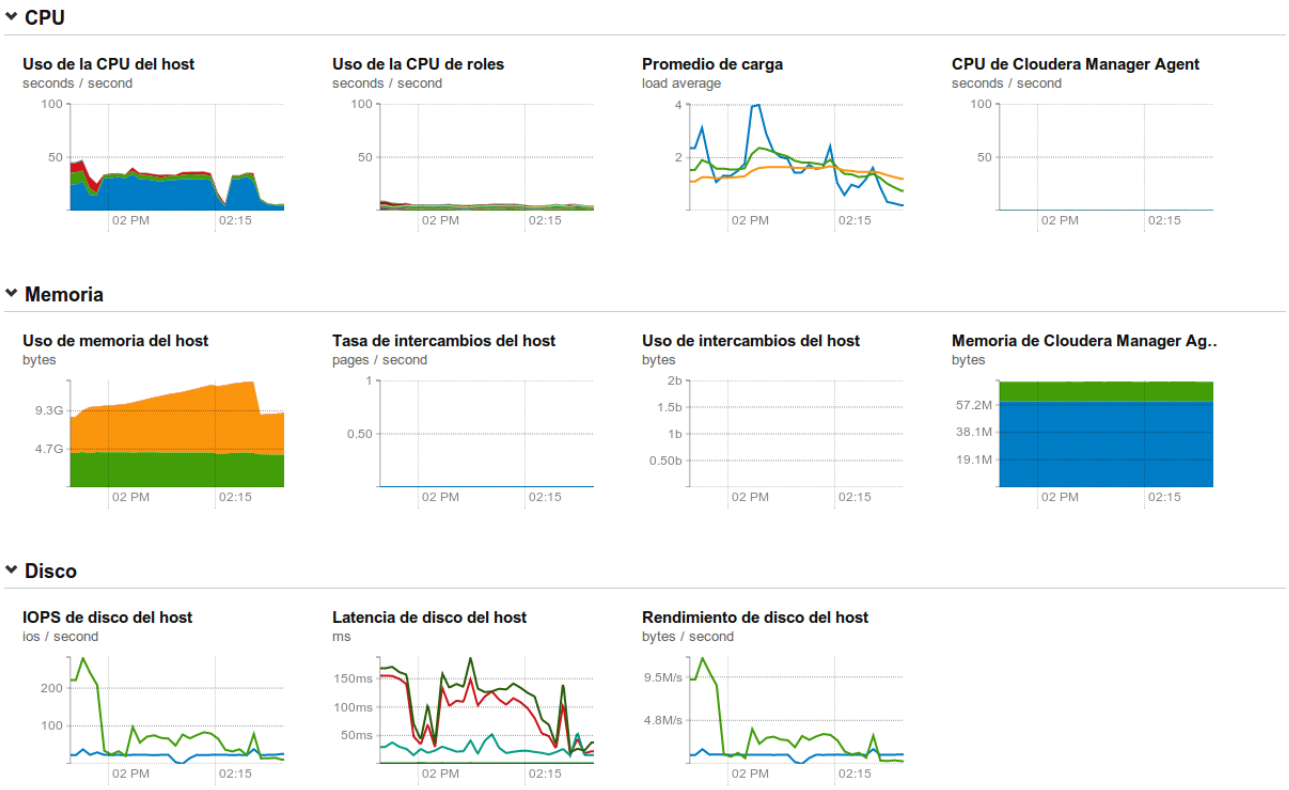

Figura 5.16: Estado nodo maestro durante consulta tipo OLAP en escenario Hive-Sin Comprimir. Fuente: copia de pantalla Cloudera Manager

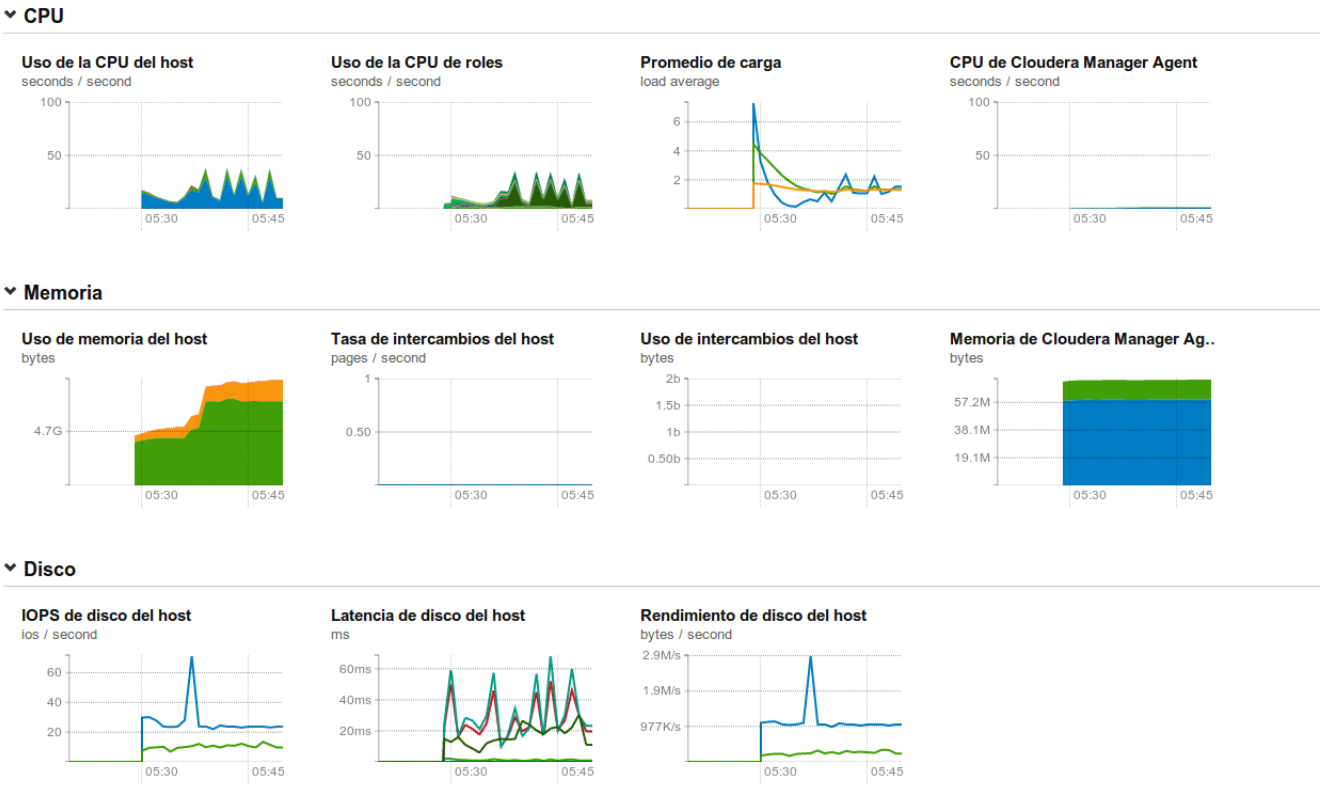

Figura 5.17: Estado nodo maestro durante consulta tipo OLAP en escenario Impala-Parquet. Fuente: copia de pantalla Cloudera Manager

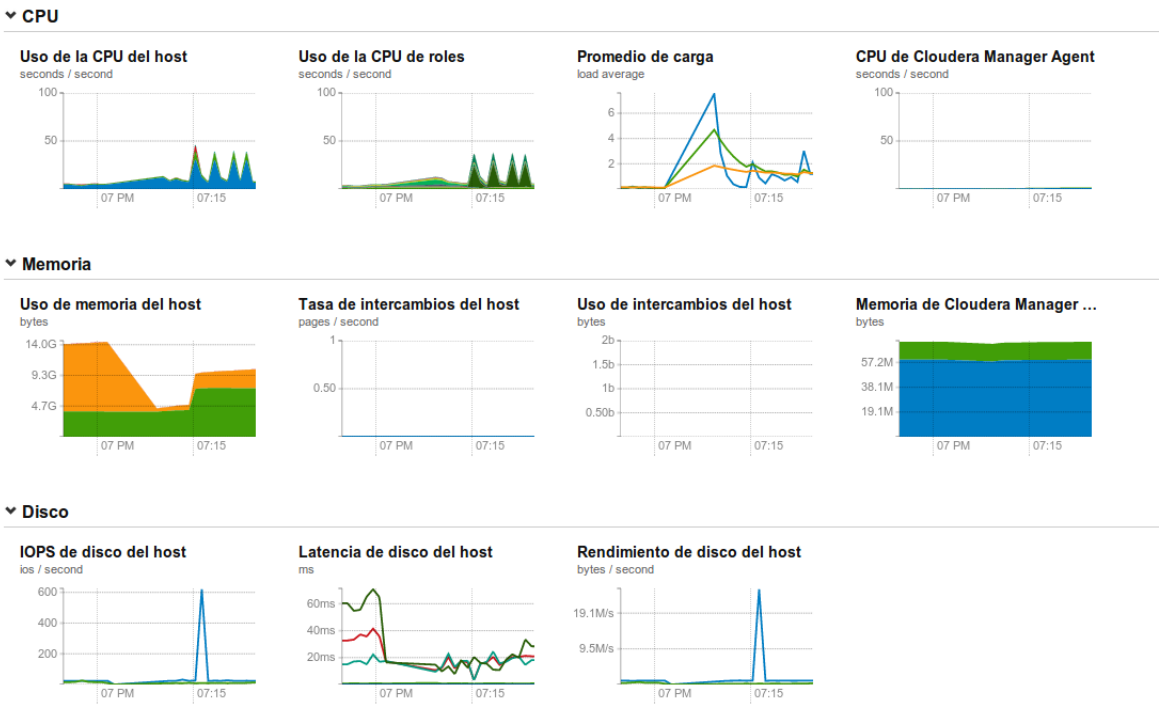

Figura 5.18: Estado nodo maestro durante consulta tipo OLAP en escenario Impala-Sin Comprimir. Fuente: copia de pantalla Cloudera Manager

Los tiempos de este tipo de consulta muestran grandes diferencias. Para comenzar, los tiempos con Hive son cerca de 35 veces mayores a los de Infobright (figura 5.9). Por otro lado, la diferencia entre los tiempos de los escenarios Impala (formatos Parquet y sin comprimir) tienen una diferencia imperceptible, siendo estos aproximadamente 45 veces menores a los presentados por la solución Infobright (figura 5.10).

Desde el punto de vista del cluster se aprecia que en los escenarios Hive el uso de CPU es mucho mayor, así como el I/O de HDFS (figuras 5.11 y 5.12). Esto se debe a que al ejecutarse los trabajos MapReduce los resultados parciales van siendo escritos en el disco y luego leídos para continuar con la siguiente tarea. No se ve una diferencia significativa en el desempeño de la opción comprimida y la sin comprimir.

Con respecto al estado del nodo Maestro, se observa un aumento significativo en el uso de memoria por parte de los escenarios Impala (figuras 5.17 y 5.18).

5.1.5. Tipo 5: ETL

Las consultas tipo ETL se presentan en varios pasos, e incluyen lectura, procesamiento y escritura de los datos. Estos se realizan en lotes de distintos tamaños: Mensual, Trimestral y Semestral.

Los tiempos se presentan en la tabla 5.8 y gráfico 5.19. Los pesos de la tabla de resultados se muestran en la tabla 5.9 y gráficos 5.20 y 5.21. Además, se muestran los gráficos de

desempeño del cluster y del nodo Maestro por cada escenario.

Nuevamente, para las variantes B y C ejecutadas con el motor Impala se debieron hacer cambios en el tipo de instancia de los nodos esclavos, igualando al maestro.

Tiempo

	Hive+Snappy	Hive+sc	Impala+Parquet*	Impala+sc*	Infobright
Mes	2255	2024,5	109,33	114,5	125,8
Trimestre	7597	6491	233,33	454	4314,25
Semestre	16620,5	14056,5	385,67	887,67	7516,02

* Variantes Trimestre y Semestre ejecutadas con nodos esclavos iguales al master.

Tabla 5.8: Tiempo de ejecución de consultas tipo ETL

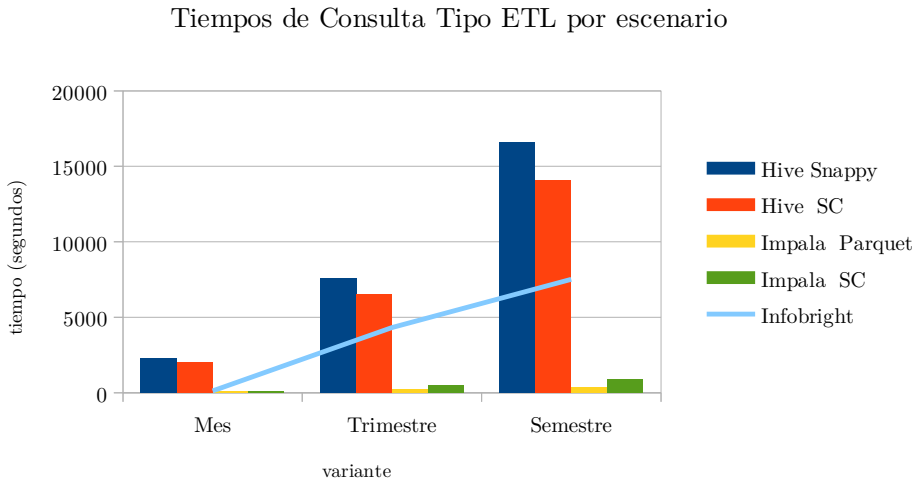

Figura 5.19: Gráfico de tiempo (segundos) por variante de consulta tipo ETL. Fuente: elaboración propia

Almacenamiento

	# Registros	Snappy	Sin Comprimir	Parquet	Infobright
Mes	3523805	451,70	1757,70	289,30	246,67
Trimestre	8511953	1457,80	8721,10	1250,30	1215,57
C	13421209	3173,70	19347,10	2423,20	1931,71

Tabla 5.9: Almacenamiento (MB) utilizado en consultas tipo ETL

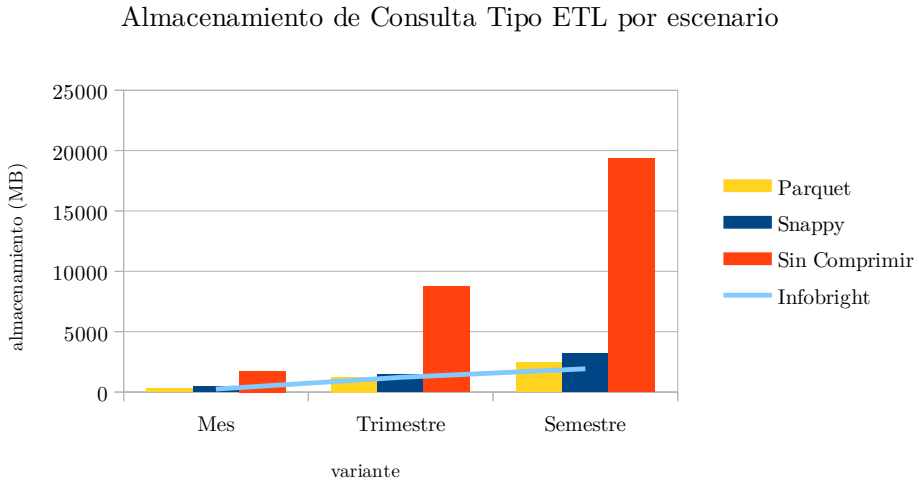

Figura 5.20: Gráfico de almacenamiento utilizado por consultas tipo ETL. Fuente: elaboración propia

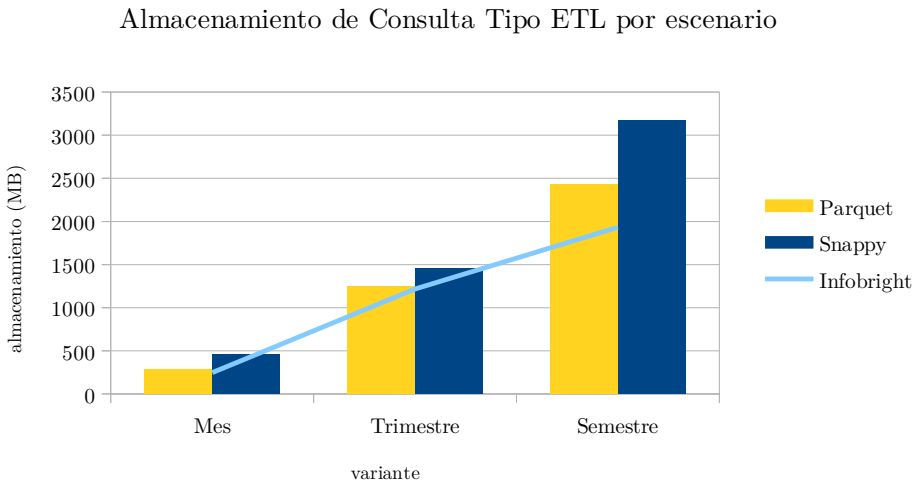

Figura 5.21: Gráfico de almacenamiento utilizado por consultas tipo ETL (solo comprimidos). Fuente: elaboración propia

Desempeño Cluster

En las siguientes imágenes se grafica a nivel de clúster el uso de CPU, IO y E/S durante el desarrollo de las queries, dividido por escenario de ejecución.

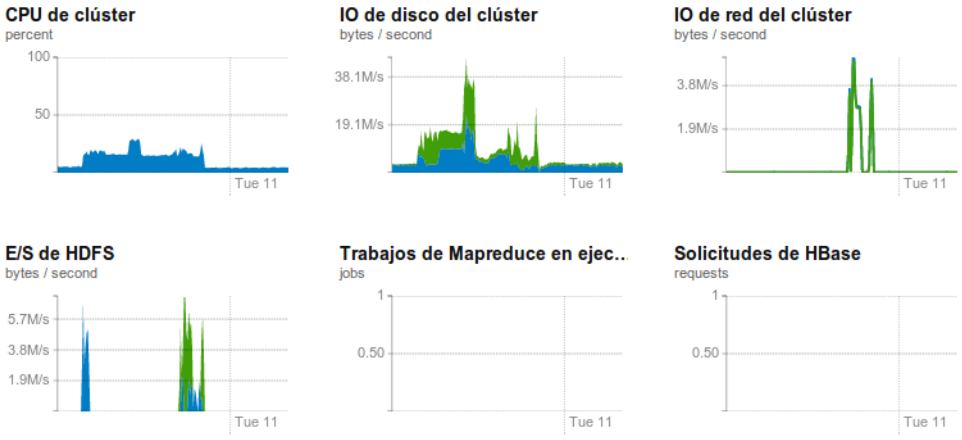

Figura 5.22: Estado cluster durante consulta tipo etl en escenario Hive-Snappy. Fuente: copia de pantalla Cloudera Manager

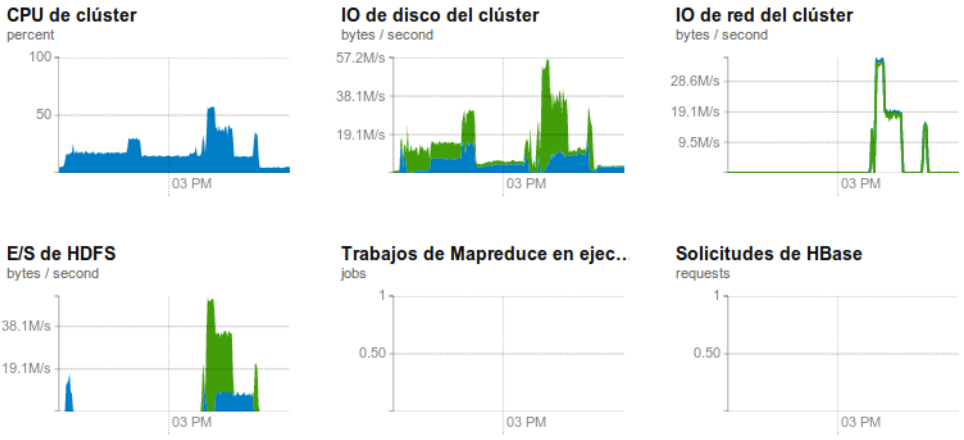

Figura 5.23: Estado cluster durante consulta tipo etl en escenario Hive-Sin Comprimir. Fuente: copia de pantalla Cloudera Manager

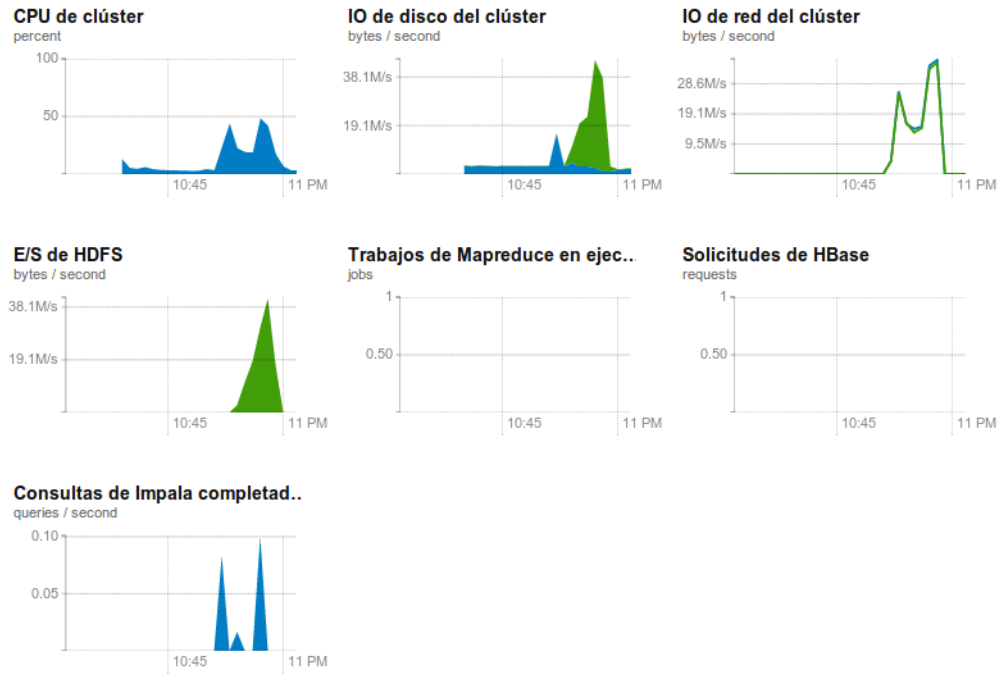

Figura 5.24: Estado cluster durante consulta tipo etl en escenario Impala-Parquet. Fuente: copia de pantalla Cloudera Manager

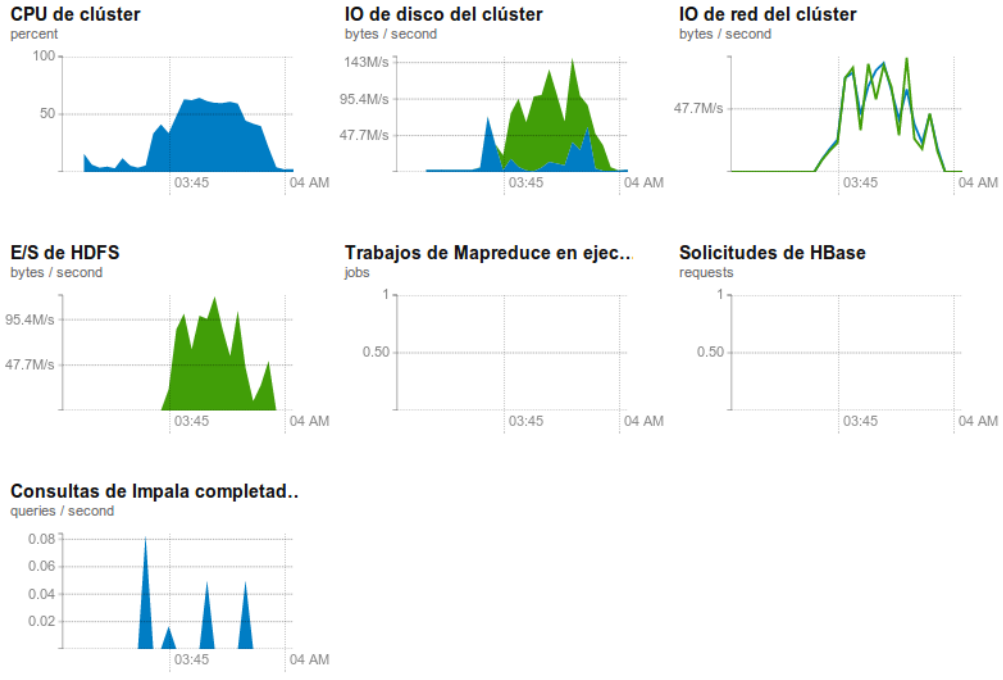

Figura 5.25: Estado cluster durante consulta tipo etl en escenario Impala-Sin Comprimir. Fuente: copia de pantalla Cloudera Manager

A continuación, se muestra el uso de CPU, Memoria y Disco en el nodo maestro durante

la ejecución de las consultas, en particular la variante Semestre, en los distintos ambientes.

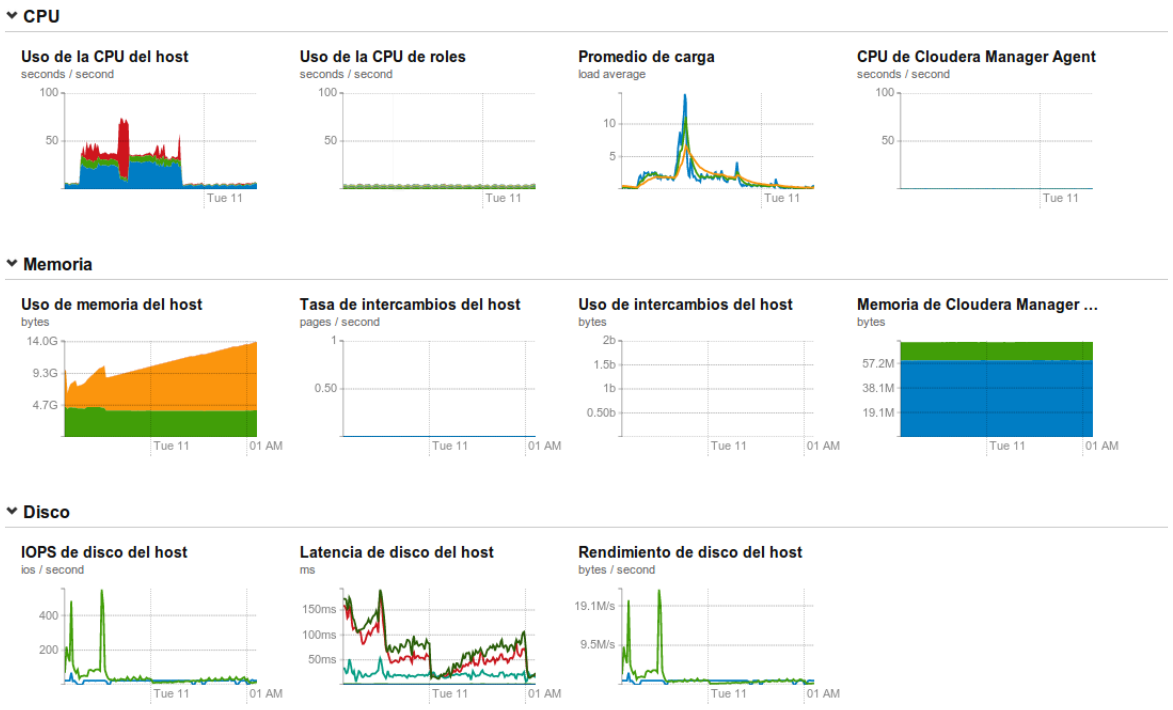

Figura 5.26: Estado nodo maestro durante consulta tipo etl en escenario Hive-Snappy. Fuente: copia de pantalla Cloudera Manager

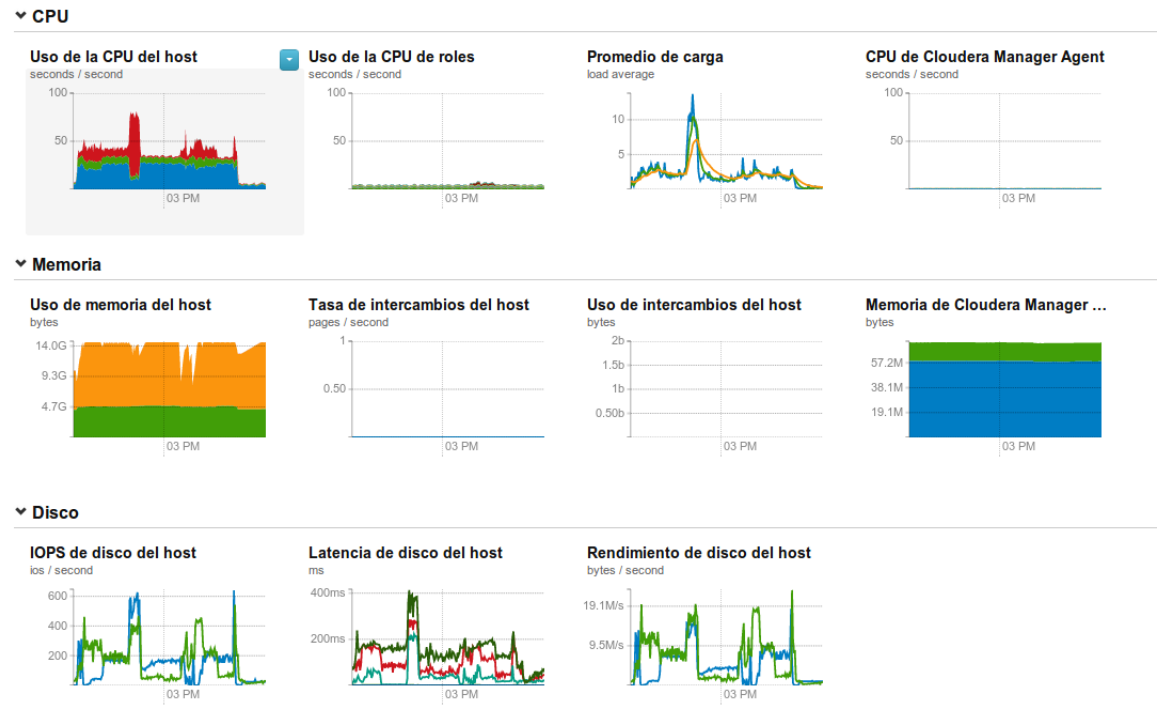

Figura 5.27: Estado nodo maestro durante consulta tipo etl en escenario Hive-Sin Comprimir. Fuente: copia de pantalla Cloudera Manager

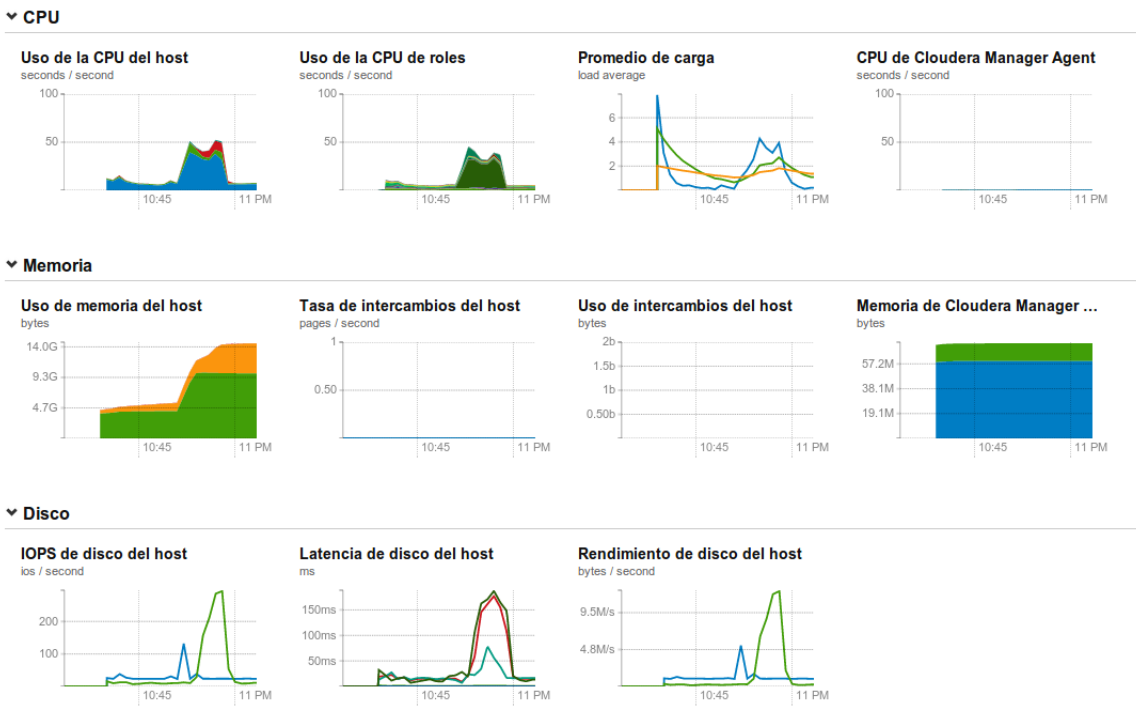

Figura 5.28: Estado nodo maestro durante consulta tipo etl en escenario Impala-Parquet. Fuente: copia de pantalla Cloudera Manager

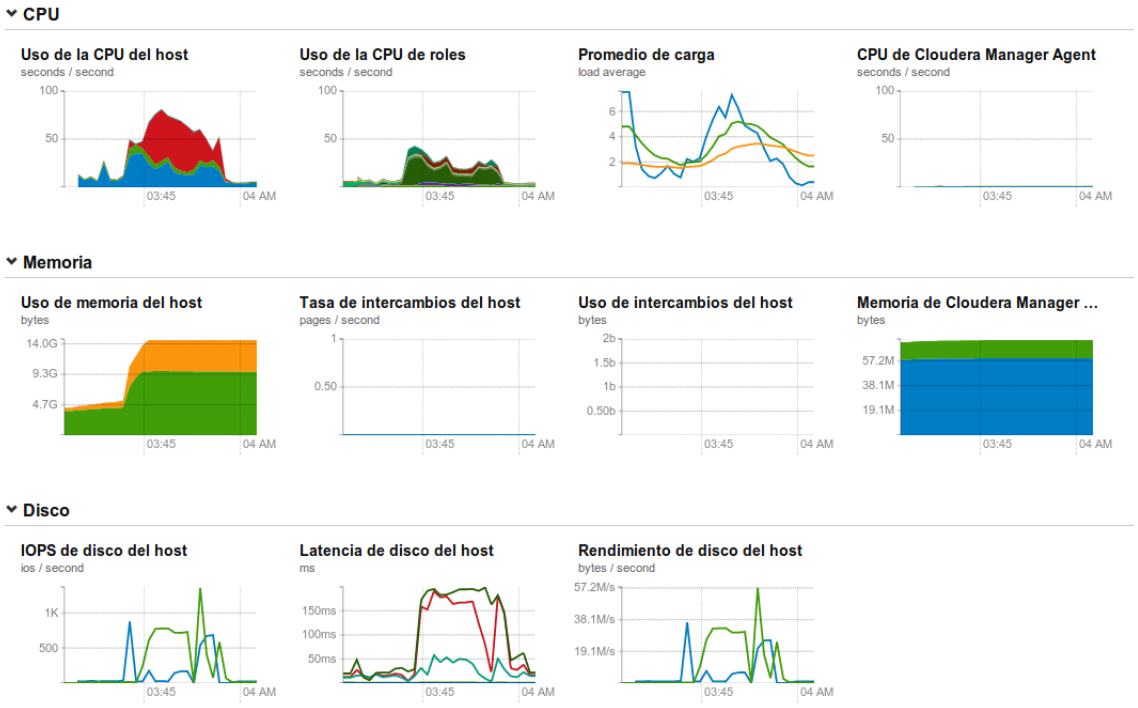

Figura 5.29: Estado nodo maestro durante consulta tipo etl en escenario Impala-Sin Comprimir. Fuente: copia de pantalla Cloudera Manager

Figura 5.30: Límite de memoria excedido en nodo-2 durante ETL. Fuente: copia de pantalla Cloudera Manager

De los resultados se puede apreciar que en este tipo de prueba es cuando los tiempos de los escenarios Hive más se acercan al actual, siendo aproximadamente el doble (figura 5.8).

Para la variante Mes el tiempo de Infobright es ligeramente mayor al de Impala, pero a medida que crece la cantidad de datos la diferencia va aumentando hasta ser casi 20 veces mayor.Nuevamente la opción Impala muestra los menores tiempos de respuesta.

Con respecto al uso de los recursos, el uso de CPU del cluster durante el transcurso de las queries es bastante similar en las opciones Hive e Impala (figuras 5.22 y 5.24).

En cuanto al uso de memoria de los nodos (Master en particular) se observa un aumento significativo en los escenarios Impala, siendo aproximadamente el doble de la memoria ocupada por las consultas con motor Hive (figuras 5.28 y 5.29).

Cabe mencionar que al intentar correr las consultas Trimestre y Semestre con motor Impala y la configuración inicial, estas terminaban abruptamente por falta de memoria en los nodos esclavos. La imagen 5.30 muestra el estado del nodo-2 durante una consulta finalizada por error de memoria.

Cloudera Impala aplica mejor en consultas más ligeras donde prime la velocidad sobre la fiabilidad, cualquier fallo en algún nodo o proceso obligaría a relanzar la consulta.

Hive es mejor para trabajos pesados de tipo ETL (Extract, Transform and Load) donde no nos interesa tanto la velocidad como la robustez de la ejecución, ya que la alta tolerancia a fallos que presenta evita la necesidad de relanzamientos al fallar algún nodo.

5.2. Análisis

De los resultados obtenidos se puede observar que los escenarios Hive presentan una latencia mayor a los demás, esto debido al overhead que presenta realizar las consultas mediante Map-Reduce.

Al realizar joins con map-reduce se generan tareas por cada unión y se almacenan temporalmente en el disco los resultados parciales, de modo que tiene un costo asociado a la

lectura/escritura de estos. También influye el orden en que se realiza el join de las tablas, ya que si bien el resultado final es el mismo, el tamaño de los resultados parciales varía.

Impala no utiliza map-reduce, sino que un motor distribuido de queries que consultan directamente a HDFS. Además, los resultados parciales son almacenados en memoria y no en el disco. Esta combinación permite que las consultas sean realizadas de manera rápida, obteniendo la menor latencia de los test. Por otra parte, las consultas están limitadas por el tamaño de la memoria de los nodos, y en caso de que uno de estos falle toda la consulta termina abruptamente.

De la consulta de tipo cubo se puede apreciar que la diferencia de tiempos se hace mucho mayor, ya que hecho de no tener que escribir los resultados de vuelta a HDFS le da mayor ventaja al motor Impala.

De los tiempos con Infobright se puede notar que parten muy bajos para pocos datos, pero al ir aumentando los registros los tiempos crecen en mayor medida, alejándose por mucho de los tiempos con Impala, pero aun por debajo de Hive.

Con respecto a la compresión de datos, si bien el tipo no comprimido debería ahorrar recursos al no tener que descomprimir las tablas y comprimir los resultados, esto no se ve reflejado en los tiempos totales de ejecución. Donde si se ve reflejado en en el espacio utilizado en HDFS, ya que el tamaño es mucho mayor al no estar comprimido, y esto se replica por 3.

Tanto snappy como parquet presentaron un buen desempeño. El formato columnar parquet permitió un ahorro tanto en almacenamiento como en acceso a las columnas. Snappy presentó un nivel de compresión un poco menor a parquet, pero tampoco influyó negativamente en su desempeño.

El nivel de compresión de Infobright es el más eficiente manteniéndose, en cuanto a espacio utilizado, siempre por debajo de las demás opciones.

En cuanto al uso de CPU y Memoria, durante las consultas con Hive los aumentos no fueron excesivos y las consultas pudieron llevarse a cabo sin problemas. Por lo que se puede preveer el comportamiento en consultas futuras dada la configuración actual. Donde se visibilizaron cambios más significativos fue en la E/S de HDFS, como era de esperar.

Las consultas realizadas con el motor Impala provocaron un aumento abrupto en el uso de memoria de los nodos, esto ocasionó que durante una consulta (con la configuración inicial) un nodo esclavo se quedara sin memoria y se abortara la consulta. Por lo que hay que tener especial cuidado con las limitaciones que el hardware implica.

Hay que tomar en cuenta que Hive es la única solución que ofrece tolerancia a fallos, ya que los datos se encuentran replicados y las referencias son almacenadas en el maestro, y en caso de fallar un esclavo el maestro puede localizar las copias. Si bien estos también se encuentran replicados para Impala, este no esta optimizado para reponerse de las fallas.

Capítulo 6

Conclusiones

La solución construida durante este trabajo de memoria evidencia que la combinación de distintas herramientas en un ambiente con tecnología Hadoop puede ofrecer un buen desempeño y mejor utilización de máquinas que la solución actual, en los casos de uso abordados.

Si se observan estrictamente los resultados numéricos se puede concluir que el motor Impala es el que ofrece el mejor desempeño en casi todos los escenarios a excepción de las consultas de tipo agregación donde es superado por Infobright. La diferencia más clara se ve en las consultas tipo OLAP, en las cuales el tamaño de los datos es menor y los accesos son de sólo lectura, en esos casos presenta tiempos hasta 45 veces menores a Infobright.

Por otra parte, se puede ver que realizar consultas con Hive (trabajos de tipo Map Reduce) requiere un overhead que no se superó en las pruebas. Por esta razón este sistema no está hecho para realizar trabajos en tiempo real. Sin embargo, muestra una mejora en los tiempos con las consultas tipo ETL, aunque sin superar a las otras herramientas analizadas.

En lo que respecta al uso de los recursos, los requerimientos de memoria de Impala son superiores a los de Hive. Estos dependen del tipo de consulta, y no existe una regla general para determinar la correlación, ya que Impala no carga tablas completas en memoria, por lo que la cantidad de memoria disponible no limita directamente el tamaño de las tablas que puede manejar. Sin embargo, Impala genera una menor carga de CPU del cluster que Hive.

Con respecto a la compresión Infobright presenta el mejor ratio, sin embargo, la compresión columnar de Parquet mostró una buena relación entre el nivel de compresión y tiempos de respuesta, superando en rendimiento a las pruebas sobre archivos sin comprimir.

En los casos de Hive y de Impala es importante mencionar que las consultas son optimizables, esto permite aprovechar al máximo el potencial del hardware. Las optimizaciones se pueden realizar de variadas formas, tales como el orden en que se hacen los joins, creación de particiones, elegir el formato mas adecuado, entre otros.

Se debe considerar que Hive e Impala no son excluyentes sino que completamente compatibles, ya que ambos trabajan sobre HDFS y comparten la metadata, de modo que no se debe comenzar de cero al momento de requerir una solución Impala. Todo el hardware que

se utiliza para MapReduce puede utilizarse para Impala.

Se puede concluir que Impala aplica mejor en consultas ligeras, donde prime la velocidad de los resultados sobre la robustez.

A su vez Hive se comporta mejor en trabajos pesados de tipo ETL, donde la robustez sea lo primordial, y no el tiempo de ejecución. Esto porque es la única solución que presenta tolerancia a fallos, evitando la necesidad de relanzar los trabajos al fallar un nodo.

En lo que respecta a ambas soluciones (Infobright y Hadoop), Hadoop tiene la ventaja de ser escalable, ya que basta con agregar un nodo para aumentar la capacidad de almacenamiento y computación del cluster. Esto implica ventajas en costos de hardware igualmente, ya que el software es opensource y el escalamiento horizontal permite construir un cluster poderoso sin tener una máquina poderosa. Sin embargo, debe tenerse en cuenta el obstáculo que presenta la empinada curva de aprendizaje de Hadoop y el costo que implica.

A pesar de los beneficios, los clústers Hadoop no son útiles para cualquier organización. Aunque se requiera análisis intensivo de datos, si se cuenta con pocos datos pueden no verse beneficiados.

Otro punto importante a considerar es que Hadoop está basado en la idea de que los datos puede dividirse y ser analizados por procesos paralelos ejecutándose en distintos nodos. Si el análisis no se adapta al modelo de procesamiento paralelo, la herramienta no es útil.

Capítulo 7

Trabajo Futuro

Si bien con el proyecto se cumplieron los objetivos y se obtuvieron resultados concluyentes, existen otras tecnologías asociadas a Hadoop que por razones de poca madurez y alcance de la memoria no fueron cubiertos.

Por una parte sería interesante estudiar una solución utilizando Phoenix [43], que ofrece una capa SQL sobre HBase, permitiendo utilizar APIs JDBC standard para el manejo de los datos en vez de las APIs regulares de HBase. De esta forma se podrían aplicar las mismas consultas actuales sin modificación. Además, esta herramienta promete optimizar las consultas para obtener una baja latencia, obteniendo un rendimiento del orden de milisegundos para pequeñas consultas o segundos para decenas de millones de filas.

Otro proyecto que sería interesante comparar es Redshift [9] de Amazon. Apareció el 2013 y consiste en un sistema de data warehousing como un servicio en la nube que promete las características y capacidades de un sistema de data warehouse relacional completo. Además, ofrece un rendimiento diez veces mayor a una décima del costo de los data warehouses usados comúnmente [3]. Actualmente está certificado por Jaspersoft y MicroStrategy, con herramientas de inteligencia empresarial adicionales próximamente.

Una segunda linea de continuación de este trabajo es el desarrollo de soluciones más eficientes. Como se explicó en las conclusiones, las consultas son optimizables, por lo que las soluciones implementadas podrían ser mejoradas. Para ello se podría realizar un estudio más a fondo de la naturaleza de las consultas y buscar la mejor forma de realizarlas para aprovechar al máximo las capacidades del hardware disponible y las capacidades de cada solución.

Bibliografía

[1] 15 important big data facts IT professionals need to know - webopedia.com. http://www.webopedia.com/quick_ref/important-big-data-facts-for-it-professionals.html.

[2] Adentrándose en la nube con amazon web services (AWS) y cloud computing | init developers. http://blog.theinit.com/2011/09/14/nube-amazon-web-services-aws-y-cloud-computing/.

[3] Amazon launches redshift data warehousing as a service - InformationWeek. http://www.informationweek.com/cloud/software-as-a-service/amazon-launches-redshift-data-warehousing-as-a-service/d/d-id/1108703?

[4] Amazon web services (AWS) - cloud computing services. http://aws.amazon.com/.

[5] Apache bigtop. http://bigtop.apache.org/index.html.

[6] Apache hive TM. http://hive.apache.org/.

[7] Apache oozie workflow scheduler for hadoop. https://oozie.apache.org/.

[8] Apache ZooKeeper - home. http://zookeeper.apache.org/.

[9] AWS | amazon redshift - cloud data warehouse solutions. http://aws.amazon.com/redshift/.

[10] AWS developer forums: Connecting pentaho report designer to ... https://forums.aws.amazon.com/thread.jspa?threadID=129715#.

[11] Bases de datos columnares | business intelligence, data warehouse, monterrey, méxico : Gravitar. http://gravitar.biz/bi/base-datos-columnar/.

[12] Big data benchmark. https://amplab.cs.berkeley.edu/benchmark/#.

[13] Big data, claves de negocio y retos tecnológicos. http://www.actibva.com/magazine/mas-que-economia/big-data-claves-de-negocio-y-retos-tecnologicos.

[14] bizo developer blog: 4 tips from the trenches of amazon elastic MapReduce and hive. http://dev.bizo.com/2011/12/4-tips-from-trenches-of-amazon-elastic.html.

[15] Build a data library with hive. http://www.ibm.com/developerworks/library/bd-hivelibrary/index.html.

[16] Build a data warehouse with hive. http://www.ibm.com/developerworks/library/bd-hivewarehouse/.

[17] Charming hadoop | JavaCruft. http://javacruft.wordpress.com/2012/08/16/charming-hadoop/#comments.

[18] Charming hadoop | MAAS. https://maas.ubuntu.com/2012/08/16/charming-hadoop/.

[19] Cloud computing: una definición formal (y la más ampliamente aceptada) | tissat I+D+i. http://tissat.wordpress.com/2012/02/20/cloud-computing-una-definicion-formal-y-la-mas-ampliamente-aceptada/.

[20] Cost of a hadoop cluster with AWS and EC2 - cloud computing | data science and cloud computing. http://www.semantikoz.com/blog/hadoop-cluster-cost-amazon-ec2-vs-emr/.

[21] Cubo OLAP - wikipedia, la enciclopedia libre. http://es.wikipedia.org/wiki/Cubo_OLAP.

[22] Data mart - wikipedia, la enciclopedia libre. http://es.wikipedia.org/wiki/Data_mart.

[23] Esquema en copo de nieve - wikipedia, la enciclopedia libre. http://es.wikipedia.org/wiki/Esquema_en_copo_de_nieve.

[24] Esquema en estrella - wikipedia, la enciclopedia libre. http://es.wikipedia.org/wiki/Esquema_en_estrella.

[25] Extract, transform and load - wikipedia, la enciclopedia libre. http://es.wikipedia.org/wiki/Extract,_transform_and_load.

[26] Fast GDELT queries using impala and parquet | global database of events, language, and tone (GDELT). http://blog.gdelt.org/2013/11/06/fast-gdelt-queries-using-impala-and-parquet/.

[27] Get set up for amazon EC2 - amazon elastic compute cloud. http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/get-set-up-for-amazon-ec2.html.

[28] GettingStarted - apache hive - apache software foundation. https://cwiki.apache.org/confluence/display/Hive/GettingStarted#GettingStarted-BrowsingthroughTables.

[29] HBase - apache HBase home. http://hbase.apache.org/.

[30] HiveServer2 clients - apache hive - apache software foundation. https://cwiki.apache.

org/confluence/display/Hive/HiveServer2+Clients.

[31] Home | apache spark. http://spark.incubator.apache.org/.

[32] How-to: Create a CDH cluster on amazon EC2 via cloudera manager | apache hadoop for the enterprise | cloudera. http://blog.cloudera.com/blog/2013/03/how-to-create-a-cdh-cluster-on-amazon-ec2-via-cloudera-manager/.

[33] Hue - the UI for apache hadoop. http://gethue.com/.

[34] Impala. http://www.cloudera.com/content/cloudera/en/products-and-services/cdh/impala.html.

[35] Investigación cuantitativa - wikipedia, la enciclopedia libre. http://es.wikipedia.org/wiki/Investigaci%C3%B3n_cuantitativa.

[36] Jaspersoft bi enterprise edition?:: Jaspersoft business intelligence software. https://www.jaspersoft.com/jaspersoft-bi-enterprise-edition.

[37] Juju documentation - using constraints. https://juju.ubuntu.com/docs/charms-constraints.html.

[38] LanguageManual DDL - apache hive - apache software foundation. https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DDL.

[39] Launch an amazon EC2 instance - amazon elastic compute cloud. http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-launch-instance_linux.html.

[40] Machine learning in hive. http://www.baynote.com/2013/03/machine-learning-in-hive/.

[41] OLAP - wikipedia, la enciclopedia libre. http://es.wikipedia.org/wiki/OLAP.

[42] Open source data warehouse, column database software, improve sql performance - infobright.org. http://www.infobright.org/.

[43] Overview | apache phoenix. http://phoenix.incubator.apache.org/.

[44] Penta analytics | inteligencia de negocios. http://www.analytics.cl/.

[45] Phoenix: Capa SQL sobre HBase | un poco de java. http://unpocodejava.wordpress.com/2013/01/31/phoenix-capa-sql-sobre-hbase/.

[46] Rittman mead consulting » blog archive » OBIEE 11.1.1.7, cloudera hadoop & Hive/Impala part 2 : Load data into hive tables, analyze using hive & impala. http://www.rittmanmead.com/2014/01/obiee-11-1-1-7-cloudera-hadoop-hiveimpala-part-2-load-data-into-hivehcatalog-anal

[47] Scenarios for using amazon VPC - amazon virtual private cloud. http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_Scenarios.html.

[48] Shark - lightning fast hive SQL on spark. http://shark.cs.berkeley.edu/.

[49] Sqoop user guide (v1.4.1-incubating). http://sqoop.apache.org/docs/1.4.1-incubating/SqoopUserGuide.html.

[50] Step 2: Set up the VPC and internet gateway - amazon virtual private cloud. http://docs.aws.amazon.com/AmazonVPC/latest/GettingStartedGuide/Wizard.html.

[51] Ubuntu in the cloud: Getting started with juju | ITworld. http://www.itworld.com/cloud-computing/345362/ubuntu-cloud-getting-started-juju.

[52] Welcome to apache? hadoop®! http://hadoop.apache.org/.

[53] Welcome to apache pig! http://pig.apache.org/.

[54] Welcome to the apache software foundation! http://www.apache.org/.

[55] ¿Qué es cloud computing? cloud computing america. http://cloud-america.com/?page_id=257.

[56] Dan Graham Amr Awadallah. Hadoop and the data warehouse: When to use which.

[57] Jeffrey Dean and Sanjay Ghemawat. MapReduce: a flexible data processing tool. 53(1):72?77.

[58] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: Simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, January 2008.

[59] Marissa Rae Hollingsworth. Hadoop and hive as scalable alternatives to RDBMS: a case study.

[60] jetlore. Spark and shark: Lightning-fast analytics over hadoop and hive data.

[61] Ralph Kimball and Margy Ross. *The Data Warehouse Toolkit the Definitive Guide to Dimensional Modeling.* Wiley.

[62] George Lumpkin Maria Colgan. Conducting a data warehouse benchmark.

[63] Sergey Melnik, Andrey Gubarev, Jing Jing Long, Geoffrey Romer, Shiva Shivakumar, Matt Tolton, and Theo Vassilakis. Dremel: Interactive analysis of web-scale datasets. In *Proc. of the 36th Int'l Conf on Very Large Data Bases*, pages 330–339, 2010.

[64] Owen O'Malley. Optimizing hive queries.

[65] Andrew Pavlo, Erik Paulson, Alexander Rasin, Daniel J. Abadi, David J. DeWitt, Samuel Madden, and Michael Stonebraker. A comparison of approaches to large-scale data analysis. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, page 165?178.

[66] Ashish Thusoo, Joydeep Sen Sarma, Namit Jain, Zheng Shao, Prasad Chakka, Ning

Zhang, Suresh Antony, Hao Liu, and Raghotham Murthy. Hive-a petabyte scale data warehouse using hadoop. In *Data Engineering (ICDE), 2010 IEEE 26th International Conference on*, page 996?1005.

Apéndice A

cpuinfo

A.1. Máquina Actual

```
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 45
model name      : Intel(R) Core(TM) i7-3820 CPU @ 3.60GHz
stepping        : 7
microcode       : 0x705
cpu MHz         : 1200.000
cache size      : 10240 KB
physical id     : 0
siblings        : 8
core id         : 0
cpu cores       : 4
apicid          : 0
initial apicid  : 0
fpu             : yes
fpu_exception   : yes
cpuid level     : 13
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr
   pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm
    pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs
   bts rep_good nopl xtopology nonstop_tsc aperfmperf pni pclmulqdq
   dtes64 monitor ds_cpl vmx est tm2 ssse3 cx16 xtpr pdcm pcid dca
   sse4_1 sse4_2 x2apic popcnt tsc_deadline_timer aes xsave avx
   lahf_lm ida arat epb xsaveopt pln pts dtherm tpr_shadow vnmi
   flexpriority ept vpid
bogomips        : 7199.63
clflush size    : 64
cache_alignment : 64
address sizes   : 46 bits physical, 48 bits virtual
```

```
power management:

processor       : 1
vendor_id       : GenuineIntel
cpu family      : 6
model           : 45
model name      : Intel(R) Core(TM) i7-3820 CPU @ 3.60GHz
stepping        : 7
microcode       : 0x705
cpu MHz         : 3601.000
cache size      : 10240 KB
physical id     : 0
siblings        : 8
core id         : 1
cpu cores       : 4
apicid          : 2
initial apicid  : 2
fpu             : yes
fpu_exception   : yes
cpuid level     : 13
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr
   pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm
    pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs
   bts rep_good nopl xtopology nonstop_tsc aperfmperf pni pclmulqdq
   dtes64 monitor ds_cpl vmx est tm2 ssse3 cx16 xtpr pdcm pcid dca
   sse4_1 sse4_2 x2apic popcnt tsc_deadline_timer aes xsave avx
   lahf_lm ida arat epb xsaveopt pln pts dtherm tpr_shadow vnmi
   flexpriority ept vpid
bogomips        : 7199.47
clflush size    : 64
cache_alignment : 64
address sizes   : 46 bits physical, 48 bits virtual
power management:

processor       : 2
vendor_id       : GenuineIntel
cpu family      : 6
model           : 45
model name      : Intel(R) Core(TM) i7-3820 CPU @ 3.60GHz
stepping        : 7
microcode       : 0x705
cpu MHz         : 1200.000
cache size      : 10240 KB
physical id     : 0
siblings        : 8
core id         : 2
cpu cores       : 4
apicid          : 4
initial apicid  : 4
```

```
fpu             : yes
fpu_exception   : yes
cpuid level     : 13
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr
   pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm
    pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs
   bts rep_good nopl xtopology nonstop_tsc aperfmperf pni pclmulqdq
   dtes64 monitor ds_cpl vmx est tm2 ssse3 cx16 xtpr pdcm pcid dca
   sse4_1 sse4_2 x2apic popcnt tsc_deadline_timer aes xsave avx
   lahf_lm ida arat epb xsaveopt pln pts dtherm tpr_shadow vnmi
   flexpriority ept vpid
bogomips        : 7199.47
clflush size    : 64
cache_alignment : 64
address sizes   : 46 bits physical, 48 bits virtual
power management:

processor       : 3
vendor_id       : GenuineIntel
cpu family      : 6
model           : 45
model name      : Intel(R) Core(TM) i7-3820 CPU @ 3.60GHz
stepping        : 7
microcode       : 0x705
cpu MHz         : 1200.000
cache size      : 10240 KB
physical id     : 0
siblings        : 8
core id         : 3
cpu cores       : 4
apicid          : 6
initial apicid  : 6
fpu             : yes
fpu_exception   : yes
cpuid level     : 13
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr
   pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm
    pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs
   bts rep_good nopl xtopology nonstop_tsc aperfmperf pni pclmulqdq
   dtes64 monitor ds_cpl vmx est tm2 ssse3 cx16 xtpr pdcm pcid dca
   sse4_1 sse4_2 x2apic popcnt tsc_deadline_timer aes xsave avx
   lahf_lm ida arat epb xsaveopt pln pts dtherm tpr_shadow vnmi
   flexpriority ept vpid
bogomips        : 7199.47
clflush size    : 64
cache_alignment : 64
address sizes   : 46 bits physical, 48 bits virtual
power management:
```

```
processor       : 4
vendor_id       : GenuineIntel
cpu family      : 6
model           : 45
model name      : Intel(R) Core(TM) i7-3820 CPU @ 3.60GHz
stepping        : 7
microcode       : 0x705
cpu MHz         : 1200.000
cache size      : 10240 KB
physical id     : 0
siblings        : 8
core id         : 0
cpu cores       : 4
apicid          : 1
initial apicid  : 1
fpu             : yes
fpu_exception   : yes
cpuid level     : 13
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr
   pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm
    pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs
   bts rep_good nopl xtopology nonstop_tsc aperfmperf pni pclmulqdq
   dtes64 monitor ds_cpl vmx est tm2 ssse3 cx16 xtpr pdcm pcid dca
   sse4_1 sse4_2 x2apic popcnt tsc_deadline_timer aes xsave avx
   lahf_lm ida arat epb xsaveopt pln pts dtherm tpr_shadow vnmi
   flexpriority ept vpid
bogomips        : 7199.46
clflush size    : 64
cache_alignment : 64
address sizes   : 46 bits physical, 48 bits virtual
power management:

processor       : 5
vendor_id       : GenuineIntel
cpu family      : 6
model           : 45
model name      : Intel(R) Core(TM) i7-3820 CPU @ 3.60GHz
stepping        : 7
microcode       : 0x705
cpu MHz         : 1200.000
cache size      : 10240 KB
physical id     : 0
siblings        : 8
core id         : 1
cpu cores       : 4
apicid          : 3
initial apicid  : 3
fpu             : yes
```

```
fpu_exception   : yes
cpuid level     : 13
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr
   pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm
    pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs
   bts rep_good nopl xtopology nonstop_tsc aperfmperf pni pclmulqdq
   dtes64 monitor ds_cpl vmx est tm2 ssse3 cx16 xtpr pdcm pcid dca
   sse4_1 sse4_2 x2apic popcnt tsc_deadline_timer aes xsave avx
   lahf_lm ida arat epb xsaveopt pln pts dtherm tpr_shadow vnmi
   flexpriority ept vpid
bogomips        : 7199.46
clflush size    : 64
cache_alignment : 64
address sizes   : 46 bits physical, 48 bits virtual
power management:

processor       : 6
vendor_id       : GenuineIntel
cpu family      : 6
model           : 45
model name      : Intel(R) Core(TM) i7-3820 CPU @ 3.60GHz
stepping        : 7
microcode       : 0x705
cpu MHz         : 1200.000
cache size      : 10240 KB
physical id     : 0
siblings        : 8
core id         : 2
cpu cores       : 4
apicid          : 5
initial apicid  : 5
fpu             : yes
fpu_exception   : yes
cpuid level     : 13
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr
   pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm
    pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs
   bts rep_good nopl xtopology nonstop_tsc aperfmperf pni pclmulqdq
   dtes64 monitor ds_cpl vmx est tm2 ssse3 cx16 xtpr pdcm pcid dca
   sse4_1 sse4_2 x2apic popcnt tsc_deadline_timer aes xsave avx
   lahf_lm ida arat epb xsaveopt pln pts dtherm tpr_shadow vnmi
   flexpriority ept vpid
bogomips        : 7199.46
clflush size    : 64
cache_alignment : 64
address sizes   : 46 bits physical, 48 bits virtual
power management:
```

```
processor       : 7
vendor_id       : GenuineIntel
cpu family      : 6
model           : 45
model name      : Intel(R) Core(TM) i7-3820 CPU @ 3.60GHz
stepping        : 7
microcode       : 0x705
cpu MHz         : 1200.000
cache size      : 10240 KB
physical id     : 0
siblings        : 8
core id         : 3
cpu cores       : 4
apicid          : 7
initial apicid  : 7
fpu             : yes
fpu_exception   : yes
cpuid level     : 13
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr
   pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm
    pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs
   bts rep_good nopl xtopology nonstop_tsc aperfmperf pni pclmulqdq
   dtes64 monitor ds_cpl vmx est tm2 ssse3 cx16 xtpr pdcm pcid dca
   sse4_1 sse4_2 x2apic popcnt tsc_deadline_timer aes xsave avx
   lahf_lm ida arat epb xsaveopt pln pts dtherm tpr_shadow vnmi
   flexpriority ept vpid
bogomips        : 7199.46
clflush size    : 64
cache_alignment : 64
address sizes   : 46 bits physical, 48 bits virtual
power management:
```

A.2. Cluster

A.2.1. Master

```
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 62
model name      : Intel(R) Xeon(R) CPU E5-2651 v2 @ 1.80GHz
stepping        : 4
microcode       : 0x415
cpu MHz         : 1800.034
cache size      : 30720 KB
physical id     : 1
siblings        : 4
```

```
core id         : 12
cpu cores       : 1
apicid          : 57
initial apicid  : 57
fpu             : yes
fpu_exception   : yes
cpuid level     : 13
wp              : yes
flags           : fpu de tsc msr pae cx8 sep cmov pat clflush mmx
   fxsr sse sse2 ss ht syscall nx lm constant_tsc rep_good nopl pni
   pclmulqdq ssse3 cx16 sse4_1 sse4_2 x2apic popcnt
   tsc_deadline_timer aes f16c rdrand hypervisor lahf_lm arat epb pln
    pts dtherm fsgsbase erms
bogomips        : 3600.06
clflush size    : 64
cache_alignment : 64
address sizes   : 46 bits physical, 48 bits virtual
power management:

processor       : 1
vendor_id       : GenuineIntel
cpu family      : 6
model           : 62
model name      : Intel(R) Xeon(R) CPU E5-2651 v2 @ 1.80GHz
stepping        : 4
microcode       : 0x415
cpu MHz         : 1800.034
cache size      : 30720 KB
physical id     : 1
siblings        : 4
core id         : 12
cpu cores       : 1
apicid          : 57
initial apicid  : 57
fpu             : yes
fpu_exception   : yes
cpuid level     : 13
wp              : yes
flags           : fpu de tsc msr pae cx8 sep cmov pat clflush mmx
   fxsr sse sse2 ss ht syscall nx lm constant_tsc rep_good nopl pni
   pclmulqdq ssse3 cx16 sse4_1 sse4_2 x2apic popcnt
   tsc_deadline_timer aes f16c rdrand hypervisor lahf_lm arat epb pln
    pts dtherm fsgsbase erms
bogomips        : 3600.06
clflush size    : 64
cache_alignment : 64
address sizes   : 46 bits physical, 48 bits virtual
power management:

processor       : 2
```

```
vendor_id       : GenuineIntel
cpu family      : 6
model           : 62
model name      : Intel(R) Xeon(R) CPU E5-2651 v2 @ 1.80GHz
stepping        : 4
microcode       : 0x415
cpu MHz         : 1800.034
cache size      : 30720 KB
physical id     : 1
siblings        : 4
core id         : 12
cpu cores       : 1
apicid          : 57
initial apicid  : 57
fpu             : yes
fpu_exception   : yes
cpuid level     : 13
wp              : yes
flags           : fpu de tsc msr pae cx8 sep cmov pat clflush mmx
   fxsr sse sse2 ss ht syscall nx lm constant_tsc rep_good nopl pni
   pclmulqdq ssse3 cx16 sse4_1 sse4_2 x2apic popcnt
   tsc_deadline_timer aes f16c rdrand hypervisor lahf_lm arat epb pln
    pts dtherm fsgsbase erms
bogomips        : 3600.06
clflush size    : 64
cache_alignment : 64
address sizes   : 46 bits physical, 48 bits virtual
power management:

processor       : 3
vendor_id       : GenuineIntel
cpu family      : 6
model           : 62
model name      : Intel(R) Xeon(R) CPU E5-2651 v2 @ 1.80GHz
stepping        : 4
microcode       : 0x415
cpu MHz         : 1800.034
cache size      : 30720 KB
physical id     : 1
siblings        : 4
core id         : 12
cpu cores       : 1
apicid          : 57
initial apicid  : 57
fpu             : yes
fpu_exception   : yes
cpuid level     : 13
wp              : yes
flags           : fpu de tsc msr pae cx8 sep cmov pat clflush mmx
   fxsr sse sse2 ss ht syscall nx lm constant_tsc rep_good nopl pni
```

```
   pclmulqdq ssse3 cx16 sse4_1 sse4_2 x2apic popcnt
   tsc_deadline_timer aes f16c rdrand hypervisor lahf_lm arat epb pln
    pts dtherm fsgsbase erms
bogomips        : 3600.06
clflush size    : 64
cache_alignment : 64
address sizes   : 46 bits physical, 48 bits virtual
power management:
```

A.2.2. Nodo-1

```
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 62
model name      : Intel(R) Xeon(R) CPU E5-2651 v2 @ 1.80GHz
stepping        : 4
microcode       : 0x415
cpu MHz         : 1795.717
cache size      : 30720 KB
physical id     : 1
siblings        : 1
core id         : 0
cpu cores       : 1
apicid          : 32
initial apicid  : 32
fpu             : yes
fpu_exception   : yes
cpuid level     : 13
wp              : yes
flags           : fpu de tsc msr pae cx8 sep cmov pat clflush mmx
   fxsr sse sse2 ss ht syscall nx lm constant_tsc up rep_good nopl
   pni pclmulqdq ssse3 cx16 sse4_1 sse4_2 x2apic popcnt
   tsc_deadline_timer aes f16c rdrand hypervisor lahf_lm ida arat epb
    pln pts dtherm fsgsbase erms
bogomips        : 3591.43
clflush size    : 64
cache_alignment : 64
address sizes   : 46 bits physical, 48 bits virtual
power management:
```

A.2.3. Nodo-2

```
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 62
model name      : Intel(R) Xeon(R) CPU E5-2651 v2 @ 1.80GHz
stepping        : 4
```

```
microcode       : 0x415
cpu MHz         : 1800.040
cache size      : 30720 KB
physical id     : 0
siblings        : 1
core id         : 13
cpu cores       : 1
apicid          : 27
initial apicid  : 27
fpu             : yes
fpu_exception   : yes
cpuid level     : 13
wp              : yes
flags           : fpu de tsc msr pae cx8 sep cmov pat clflush mmx
   fxsr sse sse2 ss ht syscall nx lm constant_tsc up rep_good nopl
   pni pclmulqdq ssse3 cx16 sse4_1 sse4_2 x2apic popcnt
   tsc_deadline_timer aes f16c rdrand hypervisor lahf_lm arat epb pln
    pts dtherm fsgsbase erms
bogomips        : 3600.08
clflush size    : 64
cache_alignment : 64
address sizes   : 46 bits physical, 48 bits virtual
power management:
```

Apéndice B

Consultas ETL

A continuación se muestran las consultas que conforman el ETL, donde las variables DATE_INICIO, DATE_FINAL Y DIM_CLIENTE_ACTUAL varían según el tipo de consulta.

Mes

DATE_INICIO : 2009

DATE_FINAL : 2039

DIM_CLIENTE_ACTUAL : dim_cliente_actual_mes

Trimestre

DATE_INICIO : 2009

DATE_FINAL : 2100

DIM_CLIENTE_ACTUAL :dim_cliente_actual_trimestre

Semestre

DATE_INICIO : 2009

DATE_FINAL : 2192

DIM_CLIENTE_ACTUAL : dim_cliente_actual_semestre

```
DROP TABLE IF EXISTS selector_clientes.dim_estado_compra_mensual_0_0;
CREATE TABLE selector_clientes.dim_estado_compra_mensual_0_0
(cli_nkey STRING,
 prod_sublinea STRING,
 prod_marca STRING,
 prod_proveedor_nombre STRING,
 monto BIGINT,
```

```
 costo BIGINT,
 unidades BIGINT,
 trx BIGINT);

DROP TABLE IF EXISTS selector_clientes.dim_estado_compra_mensual_0_1;
CREATE TABLE selector_clientes.dim_estado_compra_mensual_0_1
(descripcion1 STRING,
 descripcion2 STRING,
 descripcion3 STRING,
 descripcion4 STRING,
 descripcion5 STRING,
 cli_nkey STRING,
 prod_sublinea STRING,
 prod_marca STRING,
 prod_proveedor_nombre STRING,
 monto BIGINT,
 costo BIGINT,
 unidades BIGINT,
 trx BIGINT);

INSERT overwrite table selector_clientes.
   dim_estado_compra_mensual_0_0
SELECT  dc.cli_nkey, dp.prod_sublinea, dp.prod_marca, dp.
   prod_proveedor_nombre,  sum( vf.meas_monto ) AS monto, sum( vf.
   meas_costo ) AS costo, sum( vf.meas_unidades ) AS unidades, COUNT(
   distinct vf.tran_id) as trx
FROM selector_clientes.ventasfact vf
JOIN selector_clientes.dim_cliente_etiquetas dc ON (vf.cli_id = dc.
   cli_id)
JOIN selector_clientes.dim_producto dp ON (vf.prod_id = dp.prod_id)
WHERE vf.date_id BETWEEN ${hiveconf:DATE_INICIO} AND ${hiveconf:
   DATE_FINAL}
GROUP BY dc.cli_nkey, dp.prod_sublinea, dp.prod_marca, dp.
   prod_proveedor_nombre
ORDER BY monto DESC;

INSERT overwrite table selector_clientes.
   dim_estado_compra_mensual_0_1
SELECT dc.cli_razon_social, dc.cli_direccion, dc.cli_comuna, dc.
   cli_telefono, dc.cli_celular, vf.*
FROM selector_clientes.dim_estado_compra_mensual_0_0 vf
JOIN selector_clientes.${hiveconf:DIM_CLIENTE_ACTUAL} dc ON (vf.
   cli_nkey = dc.cli_nkey);

DROP TABLE IF EXISTS selector_clientes.dim_estado_compra_mensual_0_4;

CREATE TABLE selector_clientes.dim_estado_compra_mensual_0_4
(cli_nkey STRING, descripcion1 STRING, descripcion2 STRING,
   descripcion3 STRING, descripcion4 STRING, descripcion5 STRING,
   prod_sublinea STRING, prod_marca STRING, prod_proveedor_nombre
```

```
   STRING, monto_anterior BIGINT, monto_actual BIGINT, costo_anterior
    BIGINT, costo_actual BIGINT, unidades_anterior BIGINT,
   unidades_actual BIGINT, trx_anterior BIGINT, trx_actual BIGINT)
PARTITIONED BY (tiene_monto_actual INT);

INSERT overwrite table selector_clientes.
   dim_estado_compra_mensual_0_4
PARTITION (tiene_monto_actual=1)
SELECT  cli_nkey, descripcion1, descripcion2, descripcion3,
   descripcion4, descripcion5, prod_sublinea, prod_marca,
   prod_proveedor_nombre, 0 as monto_anterior, monto as monto_actual,
    0 as costo_anterior, costo as costo_actual, 0 as
   unidades_anterior, unidades as unidades_actual, 0 as trx_anterior,
    trx as trx_actual
FROM selector_clientes.dim_estado_compra_mensual_0_1;

DROP TABLE IF EXISTS selector_clientes.dim_estado_compra_mensual_0_5;

CREATE TABLE selector_clientes.dim_estado_compra_mensual_0_5(cli_nkey
    STRING, descripcion1 STRING, descripcion2 STRING, descripcion3
   STRING, descripcion4 STRING, descripcion5 STRING, prod_sublinea
   STRING, prod_marca STRING, prod_proveedor_nombre STRING,
   monto_actual BIGINT, monto_anterior BIGINT, costo_actual BIGINT,
   costo_anterior BIGINT, unidades_actual BIGINT, unidades_anterior
   BIGINT, trx_actual BIGINT, trx_anterior BIGINT, tiene_monto_actual
    INT);

INSERT overwrite table selector_clientes.
   dim_estado_compra_mensual_0_5
SELECT cli_nkey, descripcion1, descripcion2, descripcion3,
   descripcion4, descripcion5, prod_sublinea, prod_marca,
   prod_proveedor_nombre, SUM(monto_actual) as monto_actual, SUM(
   monto_anterior) as monto_anterior, SUM(costo_actual) as
   costo_actual, SUM(costo_anterior) as costo_anterior, SUM(
   unidades_actual) as unidades_actual,SUM(unidades_anterior) as
   unidades_anterior, SUM(trx_actual) as trx_actual, SUM(trx_anterior
   ) as trx_anterior, SUM(tiene_monto_actual) as tiene_monto_actual
FROM selector_clientes.dim_estado_compra_mensual_0_4
GROUP BY cli_nkey, descripcion1, descripcion2, descripcion3,
   descripcion4, descripcion5, prod_sublinea, prod_marca,
   prod_proveedor_nombre;
```

Apéndice C

Scripts ETL

Existe un script distinto por cada combinación de formato+motor disponible.

Cada script recibe un parametro que indica el tipo de consulta a realizar: 1 para mes, 2 para trimestre y 3 para semestre.

C.1. Hive Snappy

Se ejecuta las queries del archivo *hive_ snappy.sql*, que utiliza una base de datos comprimidos con Snappy.

Se explicita en las opciones que la salida debe ser comprimida con Snappy Codec.

```
#!/bin/bash
# se mide el tiempo que demora el ETL en correr
# se debe ingresar la opción 1 para mes, 2 para trimestre o 3 para
   semestre.
START=$(date +%s)
# SET hive.exec.compress.output=true; SET mapred.output.compression.
   codec=org.apache.hadoop.io.compress.SnappyCodec; SET mapred.output
   .compression.type=BLOCK;
fecha_inicio=2009
if [ "$1" == 1 ]; then
# julio 2013
fecha_final=2039
dim_cliente="dim_cliente_actual_mes"
echo "opcion mes"
elif [ "$1" == 2 ]; then
# tercer trimestre 2013
fecha_final=2100
dim_cliente="dim_cliente_actual_trimestre"
echo "opcion trimestre"
else
```

```
# segundo semestre 2013
fecha_final=2192
dim_cliente="dim_cliente_actual_semestre"
echo "opcion semestre"
fi
# set DATE_INICIO=1948; set DATE_FINAL=$fecha_final;

sudo -u hive hive -hiveconf DATE_INICIO=$fecha_inicio -hiveconf
   DATE_FINAL=$fecha_final -hiveconf DIM_CLIENTE_ACTUAL=$dim_cliente
   -hiveconf hive.exec.compress.output=true -hiveconf mapred.output.
   compression.codec=org.apache.hadoop.io.compress.SnappyCodec -
   hiveconf mapred.output.compression.type=BLOCK -f hive_snappy.sql
END=$(date +%s)
DIFF=$(( $END - $START ))
echo "fecha inicio: $fecha_inicio , fecha final: $fecha_final"
echo "tiempo total: $DIFF segundos"
```

C.2. Hive Sin Comprimir

Se ejecuta las queries del archivo *hive_ sc.sql*, que utiliza una base de datos en formato texto, sin comprimir.

No se especifican parámetros de compresión por lo que los archivos resultantes serán almacenados en texto sin comprimir.

```
#!/bin/bash
# se mide el tiempo que demora el ETL en correr
# se debe ingresar la opción 1 para mes, 2 para trimestre o 3 para
   semestre.
START=$(date +%s)
fecha_inicio=2009
if [ "$1" == 1 ]; then
# julio 2013
fecha_final=2039
dim_cliente="dim_cliente_actual_mes"
echo "opcion mes"
elif [ "$1" == 2 ]; then
# tercer trimestre 2013
fecha_final=2100
dim_cliente="dim_cliente_actual_trimestre"
echo "opcion trimestre"
else
# segundo semestre 2013
fecha_final=2192
dim_cliente="dim_cliente_actual_semestre"
echo "opcion semestre"
fi
```

```
# set DATE_INICIO=1948; set DATE_FINAL=$fecha_final;
sudo -u hive hive -hiveconf DATE_INICIO=$fecha_inicio -hiveconf
   DATE_FINAL=$fecha_final -hiveconf DIM_CLIENTE_ACTUAL=$dim_cliente
   -f hive_sc.sql
END=$(date +%s)
DIFF=$(( $END - $START ))
echo "fecha inicio: $fecha_inicio , fecha final: $fecha_final"
echo "tiempo total: $DIFF segundos"
```

C.3. Impala Parquet

Se ejecuta las queries del archivo *impala_parquet.sql*, que utiliza una base de datos comprimidos con parquet.

No se puede especificar en las opciones el codec de compresión ni formato, pero si en el SQL se especifica que las tablas deben ser almacenadas como PARQUETFILE.

```
#!/bin/bash
START=$(date +%s)
fecha_inicio=2009
# no se puede hacer set de DATE_INICIO y DATE_FINAL, las fechas estan
    puestas en el sql.
if [ "$1" == 1 ]; then
# julio 2013
fecha_final=2039
script="impala_parquet_mes.sql"
echo "opcion mes"
elif [ "$1" == 2 ]; then
# tercer trimestre 2013
fecha_final=2100
script="impala_parquet_trimestre.sql"
echo "opcion trimestre"
else
# segundo semestre 2013
fecha_final=2192
script="impala_parquet_semestre.sql"
echo "opcion semestre"
fi
impala-shell -f $script
END=$(date +%s)
DIFF=$(( $END - $START ))
echo "fecha inicio: $fecha_inicio , fecha final: $fecha_final"
echo "tiempo total: $DIFF segundos"
```

C.4. Impala Sin Comprimir

Se ejecuta las queries del archivo *impala_ sc.sql*, que utiliza una base de datos en formato texto, sin comprimir.

No se puede especificar en las opciones el codec de compresión ni formato, pero si en el SQL se especifica que las tablas deben ser almacenadas como TEXTFILE.

```
#!/bin/bash
START=$(date +%s)
fecha_inicio=2009
# no se puede hacer set de DATE_INICIO y DATE_FINAL, las fechas estan
    puestas en el sql.
if [ "$1" == 1 ]; then
# julio 2013
fecha_final=2039
script="impala_sc_mes.sql"
echo "opcion mes"
elif [ "$1" == 2 ]; then
# tercer trimestre 2013
fecha_final=2100
script="impala_sc_trimestre.sql"
echo "opcion trimestre"
else
# segundo semestre 2013
fecha_final=2192
script="impala_sc_semestre.sql"
echo "opcion semestre"
fi
impala-shell -f $script
END=$(date +%s)
DIFF=$(( $END - $START ))
echo "fecha inicio: $fecha_inicio , fecha final: $fecha_final"
echo "tiempo total: $DIFF segundos"
```