SISTEMAS PARA ALMACENAR GRANDES VOLÚMENES DE DATOS

BIG DATA STORES

AUTOR
SONIA JARAMILLO VALBUENA
Estudiante de doctorado en Ingeniería UPB
*Universidad del Quindío
Profesor
Programa de Ingeniería de Sistemas y Computación
sjaramillo@uniquindio.edu.co
COLOMBIA

***INSTITUCION**
Universidad del Quindío
UNIQUINDIO
Institución Pública
Carrera 15 Calle 12 Norte Armenia,
Quindío wbmaster@uniquindio.edu.co
COLOMBIA

AUTOR
JORGE MARIO LONDOÑO
Doctor en Ciencias de la Computación
**Universidad Pontificia Bolivariana
Profesor
Facultad de Ingeniería Informática y Telecomunicaciones
jorge.londono@upb.edu.co
COLOMBIA

***INSTITUCION**
Universidad Pontificia Bolivariana UPB
Institución Privada
Campus de Laureles Circular 1 No. 70-01
Medellín
información@upb.edu.co
COLOMBIA

INFORMACIÓN DE LA INVESTIGACIÓN O DEL PROYECTO: El estudio comprende una revisión del estado del arte de los sistemas para almacenar grandes volúmenes de datos. Este estudio forma parte de la propuesta de Tesis Doctoral en Ingeniería que se viene cursando en la UPB.

RECEPCIÓN: Febrero 04 de 2014 **ACEPTACIÓN:** Junio 24 de 2014

TEMÁTICA: Ingeniería de software

TIPO DE ARTÍCULO: Artículo de Investigación Científica e Innovación

Forma de citar: Jaramillo Valbuena, S. & Londoño, J. M. (2014). Sistemas para almacenar grandes volúmenes de datos. En R, Llamosa Villalba (Ed.). Revista Gerencia Tecnológica Informática, 13(37), 17-28. ISSN 1657-8236.

RESUMEN ANALÍTICO

La necesidad de almacenar y procesar grandes volúmenes de datos ha dado origen al término Big Data. Estos sistemas manejan información obtenida desde diversas fuentes y formatos, como es el caso de páginas web, redes sociales, el análisis del genoma humano, la física de partículas, entre otros. Estos almacenes de datos presentan dificultades que no pueden ser resueltas mediante el uso de sistemas de gestión de bases de datos tradicionales.

El objetivo de este artículo es revisar el estado del arte en lo referente a técnicas para el almacenamiento de grandes cantidades de datos. Se comparan las características de las bases de datos relacionales y los modelos NoSQL, que han captado la atención durante los últimos años. Todos estos sistemas deben adoptar soluciones de compromiso para lograr características críticas tales como: escalabilidad, fiabilidad, durabilidad, tiempo de respuesta, interfaz de consulta, estructura de los datos almacenados (o carencia de la misma) y esquemas de particionamiento de datos. Se presenta una revisión de las técnicas más representativas y de cómo cada una de ellas permite manejar las características indicadas previamente.

Para concluir se presenta un análisis de las ventajas y limitaciones de los modelos estudiados. Así mismo, se identifican algunos de los problemas que son objeto de investigación activa en el área.

PALABRAS CLAVES: NoSQL, Almacenamiento Clave-Valor, Almacenes de documentos, Almacenamiento por columna, bases de datos de Grafos, SMBD, Bodegas de datos.

ANALYTICAL SUMMARY

The need to store and process very large databases has given origin to the term "big data stores". These are systems that handle information obtained from crawling the web, social networks, the analysis of the genome, particle physics, and many more. These data stores pose many challenging problems that cannot be handled by traditional database management systems. The goal of this survey is to explore the current state-of-the-art solutions to the problem of managing information stores of this scale.

In this paper we compare the characteristics of relational databases against those of newly proposed NoSQL models. This latter has been the center of interest in recent years. All these systems exhibit different trade-offs around critical characteristics of the data store, such as scalability, reliability, durability, response time, query interface, structure (or the lack of) of the stored data, and data partitioning schemes. We present a review of the most representative techniques and how they handle each of these problems.

We conclude by presenting and analysis of the different trade-offs and identifying some of the problems that are still active subject of research.

KEYWORDS: NoSQL, Key Value Stores, Document Stores, Column Family Stores, Graph Databases, DBMS, Data-Warehouses.

INTRODUCCIÓN.

El volumen de información que se maneja de forma sistematizada ha experimentado un crecimiento exponencial a lo largo del tiempo. Este rápido incremento ha excedido las capacidades de las bases de datos relacionales, haciendo evidentes las limitaciones de las mismas respecto a rendimiento en aplicaciones que requieren uso intensivo de datos, y que por ende, necesitan gran cantidad de lecturas y escrituras. Las problemáticas anteriormente mencionadas motivaron el surgimiento de los sistemas NoSQL. Este tipo de sistemas de almacenamiento utilizan una arquitectura distribuida.

En este contexto se necesario mencionar el teorema CAP [8]. Este teorema impone una restricción muy severa a sistemas distribuidos: De las tres propiedades deseables – consistencia, alta disponibilidad y tolerancia a particiones de la red, solo es posible satisfacer 2 de ellas de manera simultánea. Los sistemas NoSQL no garantizan ACID (*Atomicity, Consistency, Isolation, Durability*), contrario a lo que ocurre en las bases de datos relacionales.

NoSQL hace posible el manejo de datos, que pueden ser o no estructurados. La información estructurada tiene un modelo de datos predefinido o un formato estricto. La información no estructurada carece de un modelo predefinido de datos o tiene una estructura que no encaja en el modelo formal de tablas. En caso de que exista una forma de estructura, ésta es definida mediante etiquetas u otro tipo de marcadores que permiten separar elementos semánticos. A este tipo de información se le conoce como semi-estructurada. Ejemplos de información semi-estructurada son los archivos XML y JSON. La información no estructurada normalmente se presenta como texto que contiene datos tales como números, fechas, comentarios. Además, su forma ambigua e irregular dificulta su interpretación mediante el uso de aplicaciones computarizadas convencionales. La información no estructurada tiene gran significado y puede ser utilizada por ejemplo, para realizar predicciones, identificar tendencias y conocer experiencias.

Este documento tiene por objetivo evaluar los distintos modelos de almacenamiento de grandes volumenes de datos y establecer los ámbitos de aplicación más apropiados para cada uno de ellos. Se estructura de la siguiente forma: En la sección 1 se presenta un modelo general que clasifica las principales técnicas existentes para el almacenamiento de grandes volúmenes de datos y describe en detalle las más importantes. En la sección 2 se muestran algunos estudios comparativos entre Bases de datos y Sistemas NoSQL. En la sección 3 se presenta un análisis comparativo y finalmente en la sección 4 se dan las conclusiones.

1. MODELOS PARA EL ALMACENAMIENTO DE GRANDES VOLÚMENES DE DATOS.

En este estudio se analizan dos diferentes modelos para el almacenamiento de grandes volúmenes de datos. El primero de ellos son las bases de datos relacionales, que ponen en práctica el modelo relacional. Este modelo tiene como idea fundamental el concepto de *relación*. Una relación puede verse como una tabla formada por filas (registros) y columnas (campos). Estas bases de datos se suelen implementar de forma que cumplan los requisitos ACID.

En segundo lugar, se encuentran los sistemas de almacenamiento Not Only SQL (NoSQL). En éstos se relajan algunas de las características usuales de las bases de datos por ejemplo no es imperativa la normalización [12], ni tampoco se garantiza la consistencia secuencial [7] pues estas características dificultan lograr la escritura escalable [13].

1.1 BASES DE DATOS RELACIONALES.

Las bases de datos relacionales utilizan el lenguaje SQL como lenguaje estándar para su gestión y manejo. Es posible implementar bases de datos tanto paralelas y distribuidas.

1.1.1 Bases de datos paralelas.

En una base de datos paralela es necesario distribuir los datos entre los diferentes nodos de la red. El particionamiento de datos permite el procesamiento concurrente de transacciones y la paralelización de consultas, esto es importante por cuestiones de manejo, mantenimiento y rendimiento. Existen dos estrategias de particionamiento, a saber: vertical (orientación por columnas) y horizontal (orientación por filas).

Las bases de datos paralelas generalmente usan orientación columnar. En la orientación columnar los datos se organizan por columnas, en lugar de por filas, esto quiere decir, que todos los valores para un solo elemento (por ejemplo, identificación de cliente) se guardan de tal forma que pueden ser accedidos como una unidad. La orientación columnar facilita la ejecución de consultas y la generación de reportes. Además, aliviana la carga de E/S y libera ancho de banda.
Algunos aspectos que se destacan de la orientación por columnas se indican a continuación:

- Facilidad de compresión. Es posible que las consultas se realicen mientras los datos se encuentran en formato comprimido, gracias a ello se hace un mejor uso de la CPU y se reducen los accesos a disco.
- En algunos casos se da replicación de columnas para acelerar/eliminar la necesidad de operaciones JOIN y brindar mayor tolerancia a fallos.
- El cálculo de agregados se puede realizar de manera más eficiente, ya que solo se acceden los datos de la columna correspondiente en lugar de transferir registros completos.

Las bases de datos columnares poseen sistemas de indexación por columnas y no requieren que los datos se encuentren pre-agregados para poder realizar análisis. En [1] se indica que las dos principales desventajas del

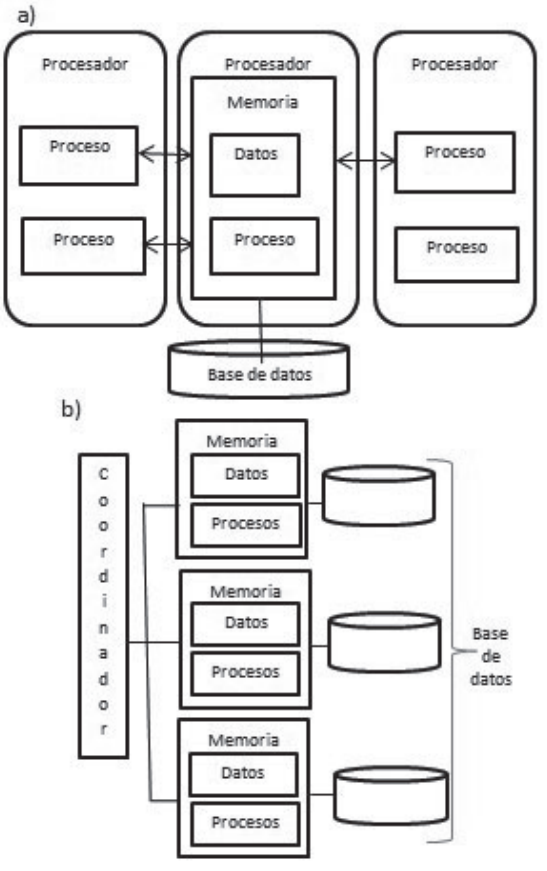
sistema columnar son las operaciones de escritura y la construcción de tuplas. La dificultad en la escritura radica en que cada atributo debe escribirse de forma separada. Por su parte, la construcción de tuplas se considera problemática, dado que la información sobre una entidad lógica se almacena en varias ubicaciones en disco. La mayoría de APIs de interfaz con la base de datos, como por ejemplo ODBC y JDBC, no se ajustan a este esquema. Por ello se hace necesario que en algún punto de la consulta, los datos de varias columnas se combinen en filas para para poder tener acceso a una entidad al tiempo.

Es importante anotar, que también existen sistemas manejadores de bases de datos paralelas que utilizan esquemas de particionamiento horizontal como método de almacenamiento de datos. El particionamiento horizontal permite repartir las filas de una tabla en distintos nodos de almacenamiento, conservando el mismo número de columnas. Cada fragmento obtenido puede considerarse como un subconjunto de las tuplas que hacen parte la relación original. El particionamiento en estos sistemas se logra automáticamente utilizando técnicas tales como: Por funciones hash, por rango o distribución Round-Robin[22].

Hay dos modelos arquitectónicos para desplegar bases de datos paralelas, ver Figura 1. El primero de ellos es el shared-everything. En este modelo todas las peticiones de E/S y accesos a memoria se canalizan por el mismo bus. Este esquema genera cuellos de botella y problemas de sincronización a medida que se agregan más procesadores se añaden al sistema. En consecuencia, este tipo de sistemas tienen unas posibilidades de crecimiento limitadas, pues el ancho de banda del bus es un cuello de botella que restringe la capacidad de crecimiento del sistema, por lo que solo permiten escalar de forma muy limitada el almacén de datos.

El segundo modelo arquitectónico es el *shared-nothing*, el que cada procesador tiene un sistema de almacenamiento dedicado. Gracias a ello, es posible proveer una arquitectura de procesamiento masivo. Algunos ejemplos comerciales basados en este modelo arquitectónico son Teradata, DataAllegro, Vertica, Greenplum y Aster.

FIGURA 1. Modelos arquitectónicos para desplegar bases de datos. a) Shared-everything b) Shared-nothing.



Fuente: [31]

1.1.2 Bases de datos distribuidas.

En una base de datos distribuida, aunque los datos pertenecen lógicamente a un sistema único, en realidad se encuentran dispersos en diferentes sitios de la red, los cuales se comunican entre sí. La distribución de los datos trae ventajas entre las cuales se destacan: disponibilidad, uso compartido de datos, fiabilidad y descentralización.

1.1.3 HadoopDB: un híbrido que combina tecnologías DBMS y MapReduce [3].

HadoopDB tiene Postgres como capa de base de datos en cada nodo, Hadoop/MapReduce como capa de comunicación para coordinar todos los nodos y Hive [29] como capa de traducción. La combinación de todos estos elementos da como resultado una base de datos paralela *shared-nothing*, en la cual es posible interactuar usando un lenguaje de tipo SQL [20].

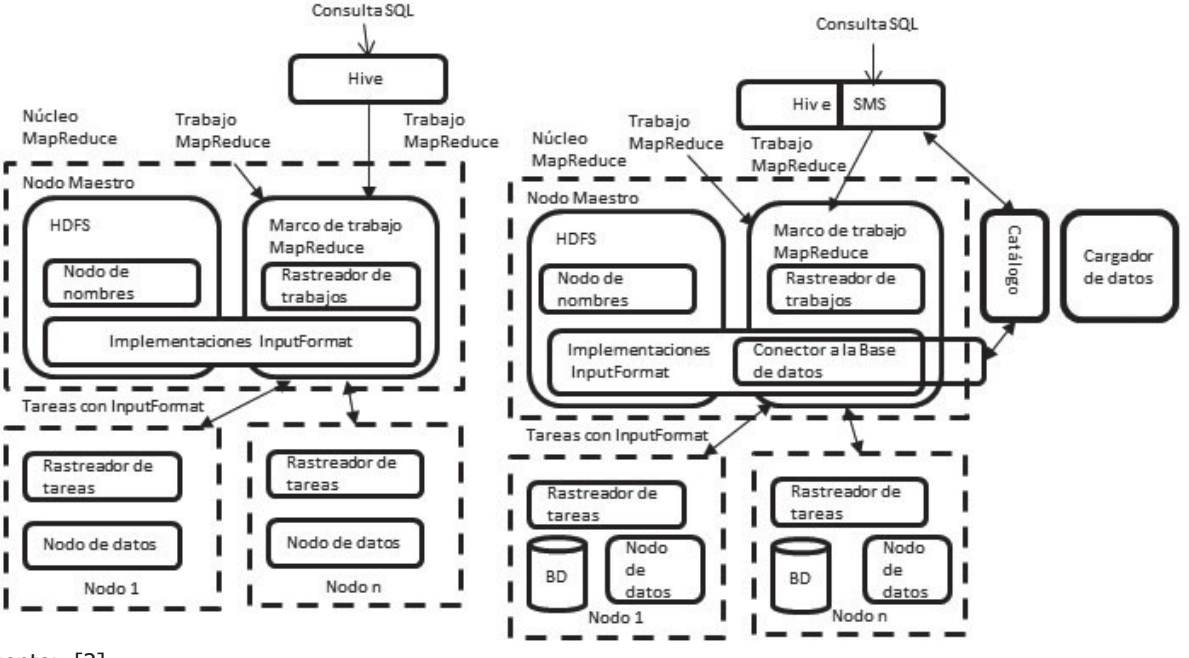
El componente principal de HadoopDB es el framework Hadoop. Hadoop se compone de dos capas, a saber: la capa de almacenamiento (Sistema de Archivos Distribuido HDFS) y la capa de procesamiento de datos (framework MapReduce).

El HDFS se estructura en bloques que son manejados por un NameNode central. Cada archivo es particionado en bloques que se distribuyen en los DataNodes del clúster. Los metadatos con la información referente al tamaño y ubicación de cada uno de los bloques y sus respectivas réplicas son conservados por el Name Node.

La capa de procesamiento MapReduce sigue una arquitectura maestro-esclavo. El maestro (JobTracker) es el encargado de la programación de los trabajos MapReduce y de asignar trabajos a los esclavos (TaskTrackers). Ambas capas, HDFS y MapReduce, se comunican mediante la interfaz InputFormat. Las implementaciones de esta biblioteca permiten analizar archivos de texto y binarios, además, de transformar los datos en parejas clave-valor para que las tareas map puedan realizar el procesamiento [2].

La arquitectura de HadoopDB se muestra en la Figura 2.

FIGURA 2. Arquitectura de Hadoop (izquierda) comparada con arquitectura de HadoopDB (derecha).



Fuente: [2]

HadoopDB extiende Hadoop para proveer 4 componentes [2]:

- Catálogo: Mantiene información acerca de las bases de datos. Esto incluye parámetros de conexión y metadatos tales como: conjuntos de datos contenidos en el clúster, localización de las réplicas y propiedades de particionamiento de datos. La información del catálogo es almacenada en un archivo XML en el HDFS.
- El cargador de datos: Particiona los datos y maneja carga paralela de los datos en los sistemas de bases de datos. El cargador de datos consta de dos componentes principales: el hasher global y hasher local. El primero de ellos ejecuta un trabajo MapReduce que se encarga de leer los datos del archivo almacenado en el HDFS y a continuación los particiona en tantos pedazos como número de nodos haya en el clúster. Por su parte el hasher local copia cada partición desde

el HDFS al sistema de archivos local de cada nodo y posteriormente particiona el archivo en trozos de tamaño más pequeños, cada uno de ellos de aproximadamente 1 GB. Estos datos son importados en las bases de datos locales.

- Planificador SQL-*to*-MapReduce-*to*-SQL (SMS): Es una base de datos paralela frontal que extiende Hive para proveer una interfaz de consultas SQL. Hive es un sistema de almacén de datos que facilita el análisis de grandes conjuntos de datos almacenados en sistemas de archivos compatibles con Hadoop [9]. Hive provee un mecanismo para para hacer las consultas a la base de datos usando un lenguaje similar a SQL, denominado HiveQL. Este lenguaje soporta un subconjunto de operaciones SQL, entre ellas selección, proyección, join, agregación y unión. No soporta borrado y actualización de filas en tablas. Las consultas realizadas en HiveQL son compiladas en trabajos MapReduce [14]. Hive posee un catálogo del sistema, llamado Hive-Metastore, que guarda estadísticas y esquemas, necesarios para la optimización de consultas y la exploración de datos.

El conector a la base de datos: Es la interface que conecta sistemas de bases de datos independientes residentes en los nodos del clúster y Hadoop.

1.2 NOT ONLY SQL (NOSQL).

El movimiento NoSQL propone la utilización de sistemas que permitan gestionar grandes cantidades de información de una forma eficiente y económica. Partiendo de este planteamiento han surgido los sistemas de almacenamiento NoSQL.

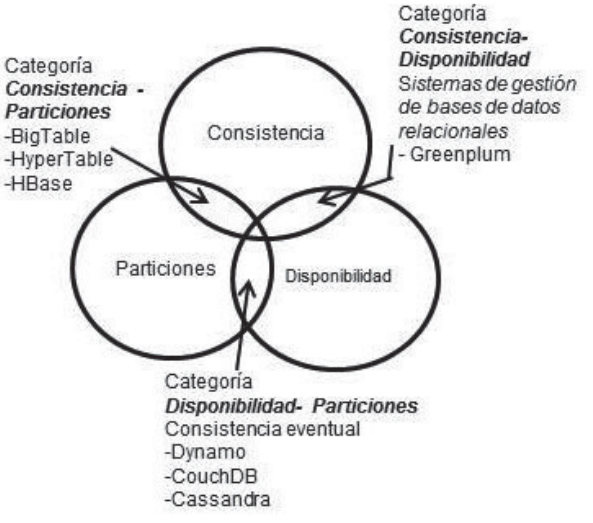
Estos sistemas se caracterizan por cumplir generalmente 6 características [10]:

- La capacidad de escalar horizontalmente a lo largo de muchos servidores.
- La capacidad para replicar y distribuir datos a través de múltiples servidores.
- Una simple interfaz de nivel de llamada o protocolo (en contraste con SQL *binding*).
- Un modelo de concurrencia débil.
- Uso eficiente de índices distribuidos y memoria RAM para almacenamiento de datos, y
- La posibilidad de añadir dinámicamente nuevos atributos a los registros de datos.

Cassandra, BigTable, HBase, MongoDB, CouchDB y Dynamo, entre otros, son sistemas de almacenamiento NoSQL.

Los sistemas NoSQL normalmente se apoyan en esquemas débiles de consistencia, como es el caso de transacciones limitadas a elementos de datos simples o consistencia eventual. Esto en consecuencia del teorema CAP [8] explicado previamente, ver Figura 3.

FIGURA 3. Teorema CAP.



Fuente: [16]

En el esquema de consistencia débil el sistema no asegura que accesos posteriores retornen el valor actualizado. Por su parte, los modelos de consistencia eventual contemplan máxima disponibilidad y tolerancia al particionado pero sacrifican la consistencia estricta en favor de estas dos características[16]. En los sistemas que implementan el modelo de consistencia eventual (conocida también como BASE- *Basically Available Soft-state Eventual Consistency*) es posible que luego de una operación las distintas réplicas queden inconsistentes durante el periodo de tiempo necesario para que las actualizaciones de propaguen a todos los nodos. Cuando se completa este proceso el sistema vuelve a un estado consistente. La restauración de la consistencia se logra mediante el uso de transacciones de compensación para el manejo de fallas.

La compensación de transacciones permite reemplazar una operación que fue ejecutada, pero que falló por algún motivo. La operación que sustituye a la inicial puede proveer capacidades similares a la original, o en su defecto, deshacer los resultados de la operación original [27].

Algunos de los sistemas NoSQL ofrecen la posibilidad de personalizar la consistencia y la disponibilidad de

acuerdo a las necesidades. Dichos sistemas hacen uso de replicación optimista (consistencia débil). Este modelo garantiza consistencia sobre un conjunto de operaciones, en lugar de sobre escrituras o lecturas aisladas. La replicación optimista permite configurar el nivel de consistencia de acuerdo a la cantidad de nodos que deben devolver un mismo valor en un proceso de lectura o a los nodos que deben hacer COMMIT en un proceso de escritura [15].

Los 4 modelos de sistemas de almacenamiento NoSQL más comúnmente usados, se muestran a continuación:

a. Depósitos llave-valor: Este tipo de almacén es similar a un diccionario o mapa, en el que por cada llave hay un valor asociado. El valor normalmente se guarda en forma binaria (BLOB), de esta forma es posible almacenar valores carentes de estructura. No obstante, algunas implementaciones llave-valor soportan listas como valores. Algunos de los beneficios de usar la aproximación llave-valor son su simplicidad y la facilidad para escalar.

   Los depósitos llave-valor son útiles para realizar operaciones simples, basadas en atributos llave únicamente. Consultas complejas o por rango resultan muy ineficientes en este tipo de depósitos. Esto se debe a que sus API permiten la obtención de información únicamente haciendo uso de operaciones *get, put* y *delete*. Si se desean funcionalidad adicionales, están deben ser implementadas como una capa superior, lo cual llevar a reducir el rendimiento y a aumentar la complejidad del sistema [15].

   Algunos ejemplos de sistemas de almacenamiento llave-valor son DynamoDB [5], MemcacheDB, SimpleDB, Voldemort y Redis.

b. Basada en Documentos: De forma similar a un sistema llave-valor, hay una llave única y una información. La diferencia con el modelo anterior radica en que la información es un documento con formato JSON o XML.

   Una limitación de este tipo de bases de datos es que la mayoría de sus implementaciones no pueden realizar uniones (JOIN) o transacciones que abarquen varias filas o documentos, por ello se ven obligadas a recurrir a la de-normalización de datos. Esta restricción es deliberada, porque facilita a la base de datos realizar particionado automático, el cual es útil para escalar [17]. Aquí es importante anotar, que aunque la de-normalización facilita el escalamiento, incrementa el costo de las actualizaciones a la base de datos y puede llevar a pérdida de consistencia y de propiedades transaccionales. Además, requiere sincronización global para poder mantener consistentes las copias.

   MongoDB y CouchDB son ejemplos de sistemas de almacenamiento de este tipo.

c. Tabular: En lugar de guardar la información en filas, se guarda en columnas. Las columnas, pueden a su vez concentrarse en un conjunto denominado familia de columnas. Gracias a ello es posible ganar velocidad en lo referente a lecturas. Son ampliamente usadas para el cálculo de datos agregados.

   Cassandra [19], BigTable [11] y HBase [6] son ejemplos pertenecientes a esta categoría.

d. Orientadas a Grafos: Estas bases de datos almacenan los datos en forma de grafo, es decir, con nodos (entidades) y aristas (relaciones). Esto permite darle importancia no solo a los datos, sino a las relaciones entre ellos. De hecho, las relaciones también pueden tener atributos y es posible efectuar consultas directas a relaciones, en vez de a los nodos.

   Bases de datos tales como Neo4j, AllegroGraph y FlockDB ejemplifican este modelo.

Los sistemas NoSQL, en lo que respecta particionamiento, se apoyan en dos estrategias, a saber: hashing consistente y particionamiento basado en rango. El método hash divide el conjunto de datos mediante la aplicación de una función de localización, la cual retorna el nodo donde se va a almacenar la información. Depósitos llave-valor tales como Voldemort, Redis y DynamoDB utilizan hashing consistente, al igual que lo hace el sistema de almacenamiento basado en documentos CouchDB, el sistema tabular Cassandra y el sistema orientado a grafos FlockDB. El particionamiento por rango distribuye el conjunto de datos teniendo en cuenta el rango de sus claves. MongoDB, HBase e Hypertable usan una estrategia de particionamiento basada en rango.

Pasando a los sistemas de almacenamiento basados en grafos, se observa que en la mayoría de ellos realizar operaciones de particionamiento es demasiado complejo. Esto se debe principalmente a dos aspectos. El primer de ellos es que para obtener información del grafo es necesario considerar las relaciones entre entidades y el segundo, que los nodos deben distribuirse uniformemente entre los servidores teniendo como consideración adicional que nodos relacionados fuertemente no pueden separados a grandes distancias. Debido a los anteriores inconvenientes y a que el grafo puede cambiar muy rápidamente algunos sistemas,

como por ejemplo GraphDB y Neo4J, no ofrecen posibilidades de particionamiento [15].

2. COMPARATIVOS PREVIOS ENTRE BASES DE DATOS Y SISTEMAS NOSQL.

Se han realizado diversos análisis comparativos entre Bases de datos y sistemas NoSQL. En [10][28] se efectúa una comparación en donde se analizan aspectos tales como modelo de datos, mecanismos de consistencia y de almacenamiento, garantías de persistencia, disponibilidad y soporte a consultas. En [32] se reporta un cotejo entre MySQL y un sistema orientado a Grafos, llamado Neo4j, que incluye comparaciones subjetivas partiendo de la experiencia y la documentación. Además, se analiza la velocidad de procesamiento basado en consultas predefinidas, requisitos de espacio en disco y escalabilidad.

En [25] se analizan 5 diferentes sistemas de almacenamiento: RDBMS, depósitos llave-valor, Tabular, Basada en Documentos, y orientadas a grafos. Aquí se examinan 8 diferentes implementaciones de esos 5 sistemas de almacenamiento, mediante el uso un dataset real proveniente de contenido de social media. Se consideran criterios como escalabilidad y procesamiento de datos y consultas. Finalmente, en [24] se estudia el problema de cómo los sistemas requieren nuevas formas de manejar sus recursos para poder escalar. Se reflexiona sobre el concepto de *sharding* y se analizan los modelos ACID y BASE.

La comparación realizada en el presente artículo difiere de las anteriores en que en éste la información es más actualizada y se analizan más características. Además, se incluyen sistemas que no habían sido comparados en publicaciones previas (entre ellos HadoopDB) y se establece los ámbitos de aplicación más apropiados para cada uno de estos modelos.

3. ANÁLISIS COMPARATIVO.

El modelo relacional ha traído consigo innumerables ventajas tales como gran expresividad, integridad, compatibilidad, estandarización, fiabilidad, garantía de independencia de datos y facilidad de conectividad con los lenguajes de programación estándar [4]. Este modelo pone a disposición una infinidad de posibilidades para realizar consultas, entre ellas unir tablas con base a relaciones, obtener agregados y seleccionar subconjuntos de datos con un criterio de búsqueda. La llegada de las aplicaciones web y las redes sociales ha sacado a flote fallas en estos modelos para trabajar en este tipo de ambientes: es demasiado estricto, no da soporte a estructuras de datos complejas, la realización de JOINS es ineficiente. Además, de que tienen limitaciones para escalar. Su problema para escalar radica en la distribución horizontal de datos y carga. A estas aplicaciones se les dificulta lograr *sharding* automático, esta funcionalidad no es proporcionada de forma nativa. El *sharding* requiere que entidades de datos diferentes se distribuyan a través de un número de nodos de la base de datos y que también éstas se procesen y escriban de forma independiente, ver Figura 4 [18].

Por otra parte, se encuentran los sistemas de almacenamiento NoSQL Este tipo de sistemas poseen importantes características, tales como: rapidez y escalabilidad horizontal cerca al orden lineal. En [26] se describe una arquitectura para escalar Sistemas NoSQL, la cual consiste en combinar mensajería asíncrona con un sistema de bases relacionales capaz de proveer una poderosa infraestructura en la que los desarrolladores puedan construir aplicaciones altamente escalables y ágiles para proveer tolerancia a particiones y alta disponibilidad mientras proporcionan un alto nivel de consistencia eventual.

Los sistemas NoSQL presentan algunas desventajas. Una de ellas es que no se cuenta con un único modelo de datos a nivel de sistema.

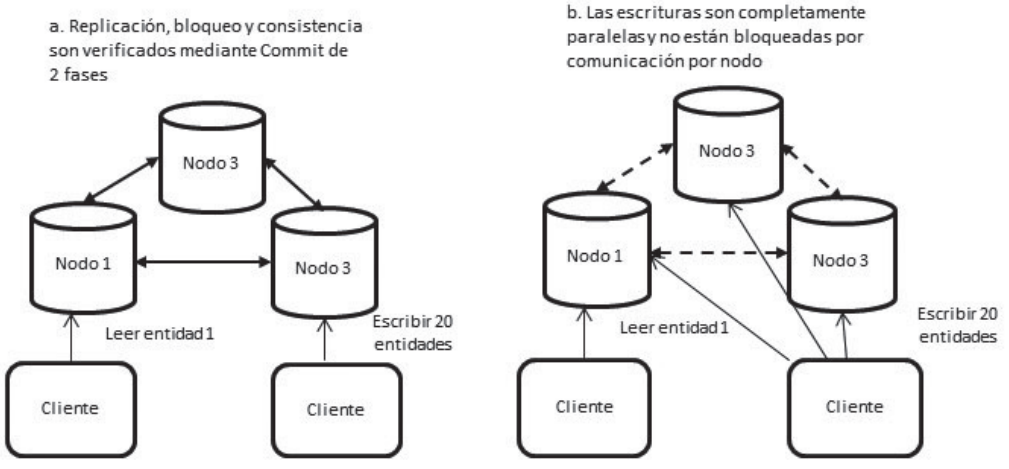
Adicionalmente, en lo que respecta a arquitectura se evidencia poca estandarización de interfaces para servicios NoSQL, carencia de una semántica estándar y esto, por ende, lleva a problemas de interoperabilidad.

En lo que respecta a cuando usar cada uno de estos sistemas, los NoSQL son apropiados cuando se necesita atender a millones de usuarios manteniendo el rendimiento, como es el caso de redes sociales y negocios basados en tecnologías de las Web 2.0. Aquí deben realizarse lecturas y escrituras de grandes cantidades de datos. Los sistemas NoSQL proveen una solución rápida (al no tener que desarrollar un modelo detallado), fácil de usar, que no es prohibitivamente costosa y que es escalable usando hardware de bajo costo. Sin embargo, si lo que se requiere es integridad referencial, utilizar conexiones estándar entre servidores y clientes, consultas arbitrarias y joins, una comunidad para soporte, estandarización, herramientas de análisis y pruebas de rendimiento de informes para validar la aplicación, entonces los sistemas de bases de datos son la opción apropiada.

La TABLA 1, resume el comparativo entre los sistemas relacionales y los NoSQL.

En la TABLA 2 se analizan diferentes implementaciones de estos sistemas, de acuerdo con sus características de diseño.

FIGURA 4. Distribución de entidades en a) Bases de datos y b) NoSQL.



Fuente: [18]

TABLA 1. Comparativo entre sistemas de almacenamiento de grandes volúmenes de datos.

Característica	Base de datos relacional	Sistemas de almacenamiento NoSQL
Modelo de almacenamiento de datos y Esquema	Considera 3 aspectos: relaciones (tablas), atributos (columnas de una relación) y dominios (valores de un atributo). El esquema, estructura y tipos de datos se debe especificar antes de poder empezar a agregar datos. Si se requiere modificar el modelo es necesario alterar la base de datos completa, quedando durante este proceso en estado "fuera de línea".	Varía dependiendo del sistema de almacenamiento, siendo posible manejar información estructurada, no estructurada o semiestructurada. Se pueden almacenar datos disímiles juntos. Formatos como BLOB, XML, JSON, o cualquier otro son admisibles. No requieren un esquema definido, esto en general, puede hacerse de forma dinámica. Al no tener un modelo de datos definido pueden presentarse problemas en caso de migración.
Modelo de desarrollo	Código abierto código cerrado	Generalmente de código abierto.
Soporte para transacciones y recuperación de fallas	Puede ser configurado para que la operación finalice completamente o no se ejecute en absoluto.	Se da dependiendo de las circunstancias y en determinados niveles, por ejemplo a nivel de elementos de datos simples. Algunos sistemas dan soporte de transacciones ACID, mientras que otros tales como MongoDB no ofrecen transacciones, en su reemplazo tienen 2 opciones: operaciones atómicas y commit de 2 fases.
Manipulación de datos, interfaces e interoperabilidad	Ya está estandarizado. Hay un lenguaje específico, SQL.	Ofrecen diferentes APIs orientadas a objetos para interactuar con datos. Las interfaces para los servicios NoSQL no se han estandarizado, aún no se tiene semántica estándar.
Consistencia	Consistencia estricta	Es configurable dependiendo del producto. Algunos ofrecen consistencia estricta y otros consistencia eventual.
Escalabilidad	Usualmente vertical, puede escalar horizontalmente pero es más restrictivo	Horizontal
Soporte para almacenar datos a través de múltiples equipos, Auto-sharding	Capacidad no proporcionada de forma nativa, pero puede lograrse.	En su mayoría ofrecen soporte nativo y automático de auto-sharding.
Ámbito operacional	Operación sistemática y autónoma.	Si algo falla para diagnosticar el problema se requiere pasar por la cadena, incluso es posible que se requiera llegar a nivel de desarrollador.
Replicación	Soportada	En su mayoría, este proceso es automático

Fuente: Basado en [21]

TABLA 2. Comparativo entre diferentes implementaciones de sistemas relacionales y NoSQL.

Base de datos		Consistencia (+)Strong (/)BASE (-Tuneable)	Modelo de datos	Arquitectura	Particionamiento
Relacionales	Vertica	+	Tabular	Share-nothing	Mixta--Híbrido Fila/Columna
	Teradata	+	Tabular	Share-nothing	Mixta--Híbrido Fila/Columna
	HadoopDB	+	Tabular	Share-nothing	*
NoSQL--Tabular	Cassandra	+/-	Columnas, grupo de columnas correspondientes a una clave (supercolumnas)	Share-nothing	Hashing consistente
	BigTable	+	Mapa multidimensional, cada valor del mapa está indexado por 3 valores: *row key*, *column key* y *timestamp*	Share-nothing	Rango
	HBase	+	Grupo de columnas (clon de BigTable)	Share-nothing	Rango
NoSQL--Documentos	MongoDB	+/-	Documentos con información semiestructurada almacenada en colecciones	*	Rango
	CouchDB	/	Documentos como una lista de ítems llamados JSON	*	Hashing consistente
NoSQL Clave-valor	Dynamo	+/-	Grupo de parejas Clave-Valor	Share-nothing	Hashing consistente
NoSQL Grafos	Neo4j	+	Grafo con nodo y aristas	*	*
	AllegroGraph	+	Grafo con nodo y aristas	*	*

Como se mencionó previamente NoSQL posibilita manejar datos estructurados o no estructurados. Respecto a estos últimos, vale la pena resaltar el interrogante de cómo enfrentar el problema de carencia de estructura en los casos en cuales no pueden convertirse los datos a semiestructurados o estructurados. En la actualidad se están trabajando técnicas de Aprendizaje Automático tales como Text Mining, Web Content Mining y Web Structure Mining. Todas ellas importantes para lograr el objetivo de buscar información relacionada, personalizar información, crear nueva información y aprender a partir de los usuarios o consumidores.

El Big Data plantea problemáticas tales como exploración de grandes volúmenes datos, predicción de acciones de los usuarios, detección de amenazas de seguridad, visibilidad en tiempo real sobre las operaciones realizadas por los usuarios y ampliación de las estructuras de almacenamiento existentes. La elección del sistema de almacenamiento apropiado en este tipo de ambientes depende de las necesidades particulares de cada caso. Los sistemas NoSQL son una alternativa para manejar grandes cantidades de información proveniente de buscadores, redes móviles, sistemas de información geográfica, aplicaciones web y redes sociales. Ventajas

tales como eficiencia, elasticidad y manejo de datos no estructurados, los hacen propicios para trabajar con textos, mapas, imágenes y sonidos. Su posibilidad para escalar y funcionar de manera distribuida permiten su uso en la computación en nube. Ahora bien, los sistemas de bases de relacionales, por su capacidad de mantener diversos niveles de consistencia y garantizar que las tareas se efectúen de forma eficiente y confiable, son utilizados en aplicaciones de gestión, administrativas y comerciales. Finalmente, hay casos en los que coexisten ambos sistemas, un ejemplo de ello es Twitter [30].

4. CONCLUSIONES.

A pesar de la madurez que han alcanzado las tecnologías de DBMS paralelos en los últimos años, hay aspectos referentes a su desempeño que aún no han sido resueltos, uno de ellos es el escalamiento. Esta limitación ha motivado al surgimiento de otro tipo de sistemas, denominados NoSQL, los cuales pueden escalar horizontalmente, debido a su flexibilidad, al particionamiento de datos (*sharding*), al buen manejo del concepto de redundancia y a la capacidad de procesar tareas concurrentemente. Los sistemas NoSQL han hecho que los desarrolladores, para lograr un alto rendimiento, adopten soluciones de compromiso entre aspectos tales como alta disponibilidad mediante el uso de replicación, balanceo de carga, particionamiento y el modelo de consistencia. En lo que respecta a consistencia, se destaca que estos sistemas son capaces de soportar modelos tales como consistencia eventual y consistencia débil.

Es importante anotar, que aún existen muchas preguntas abiertas y áreas de investigación activas alrededor de los Sistemas NoSQL. Por ejemplo se tiene que los almacenes clave-valor presentan problemas para realizar consultas complejas, algunas de las bases de datos orientada a grafos no pueden realizar particionamiento y la mayoría de las implementaciones de bases de datos documentales y tipo BigTable son incapaces de efectuar uniones o transacciones que abarquen varias filas o documentos.

Finalmente no se descarta que en el futuro, ambos sistemas, bases de datos relacionales y almacenes de datos NoSQL, se complementen y que la tendencia sea la construcción de sistemas híbridos compuestos por múltiples almacenes de datos cada uno de ellos basado en diferentes principios [23].

5. REFERENCIAS BIBLIOGRÁFICAS.

[1] Abadi, D., Boncz, P., & Harizopoulos, S. (2009). Column-oriented database systems. ACM Proceedings VLDB Endowment, 2(2), 1664-1665.

[2] Abouzeid, A., Bajda- Pawlikowski, K., Abadi, D., Silberschatz, A., & Rasin, A. (2009). HadoopDB: an architectural hybrid of MapReduce and DBMS technologies for analytical workloads. ACM Proceedings VLDB Endowment, 2(1), 922-933.

[3] Abouzied, A., Bajda-Pawlikowski, K., Huang, J., Abadi, D., & Silberschatz, A. (2010). HadoopDB in action: building real world applications. SIGMOD '10 Proceedings of the 2010 ACM SIGMOD International Conference on Management of data, 1111-1114.

[4] Alvarez, S., & Bravo, S. Archivos y Bases de Datos. (2009). Universidad de Salamanca. Recuperado (2014, abril 03) de http://ocw.usal.es/ensenanzas-tecnicas/aplicaciones-informaticas-para-humanidades/contenidos/Temas/Tema7-_Archivos_y_BBDD_-_2ppt.pdf

[5] Amazon Web Services. (2012). DynamoDB. Seattle, WA, Estados Unidos. Recuperado (2014, febrero 3) de http://aws.amazon.com/es/dynamodb/

[6] Apache Software Foundation. (2012). HBase. Delaware, Estados Unidos. Recuperado (2014, febrero 03) de http://hbase.apache.org/

[7] Arredondo, P. (2011). NoSQL (Not only SQL). México: Universidad Veracruzana. Recuperado (2014, febrero 3) de http://www.uv.mx/universo/448/infgral/infgral_08.html

[8] Brewer. (2000). Principles of Distributed Computing. Nineteenth ACM Symposium on Principles of Distributed Computing.

[9] Brown, R. A. (2009). Hadoop at home: large-scale computing at a small college. SIGCSE '09 Proceedings of the 40th ACM technical symposium on Computer science education, 41(1), 106-110.

[10] Cattell, R. (2011). Scalable SQL and NoSQL data stores. ACM SIGMOD Record, 39(4), 12-27.

[11] Chang, F., Dean, J., Ghemawat, S., Hsieh, W. C., Wallach, D. A., Burrows, M., & Gruber, R. (2008). Bigtable: A Distributed Storage System for Structured Data. ACM Transactions on Computer Systems (TOCS), 26(2), 4.

[12] Dans, E. (2011). Entender el futuro: la evolución de las bases de datos. España. Recuperado (2014, enero 22) de http://www.enriquedans.com/2011/11/entender-el-futuro-la-evolucion-de-las-bases-de-datos.html

[13] De seta, L. (2010). NoSQL y varias alternativas a las bases de datos. Buenos Aires, Argentina. Recuperado (2014, enero 22) de http://www.dosideas.com/noticias/base-de-datos/864-nosql-una-alternativa-a-las-bases-de-datos.html

[14] Dean, J., & Ghemawat, S. (2008). MapReduce: simplified data processing on large clusters. Communications of the ACM - 50th anniversary issue: 1958 - 2008, 51(1), 107-113.

[15] Hecht, R., & Jablonski, S. (2011). NoSQL evaluation A use case oriented survey. Cloud and Service Computing, International Conference, 336-341.

[16] Init Developers. (2012). Introducción a NO-SQL: Cassandra y CouchDB. Recuperado (2014, junio 19) de http://blog.theinit.com/2012/04/24/introduccion-a-no-sql-cassandra-y-couchdb/

[17] Kleppmann, M. (2009). Should you go Beyond Relational Databases. Estados Unidos: Treehouse Island, Inc. Recuperado (2014, enero 22) de http://thinkvitamin.com/code/should-you-go-beyond-relational-databases/

[18] Kopp Michael (2011). NoSQL or RDBMS? – Are we asking the right questions?. Recuperado (2014, junio 19) de http://apmblog.compuware.com/2011/10/05/nosql-or-rdbms-are-we-asking-the-right-questions/.

[19] Lakshman, A., & Malik, P. (2010). Cassandra: a decentralized structured storage system. ACM SIGOPS Operating Systems Review, 44(2), 35-40.

[20] Lorica. (2009). HadoopDB: An Open Source Parallel Database. Estados Unidos: O'Reilly Media, Inc. Recuperado (2014, enero 22) de http://strata.oreilly.com/2009/07/hadoopdb-an-open-source-parallel-database.html

[21] MongoDB. (2014). Recuperado (2014, junio 19) de http://www.mongodb.com/nosql-explained.

[22] Ozsu, M., & Valduriez, P. (2011). Principles of Database Systems, Third Edition. USA: Prentice-Hall Segunda edición.

[23] Pokorny, J. (2011). NoSQL databases: a step to database scalability in web environment. iiWAS '11 Proceedings of the 13th International Conference on Information Integration and Web-based Applications and Services, 278-283.

[24] Pritchett, D. (2008). BASE: An Acid Alternative. Magazine Queue - Object-Relational Mapping Queue Homepage archive, 6(3), 48-55.

[25] Ruflin, N., Burkhart , H., & Rizzotti, S. (2012). Social-Data Storage-Systems. Proceeding DBSocial '11 Databases and Social Networks, 7-12.

[26] Rys, M. (2011). Scalable SQL. Communications of the ACM, 54(6),48-53.

[27] Schafer, M., Dolog, P., & Nejdl, W. (2008). An environment for flexible advanced compensations of Web service transactions. ACM Transactions on the Web (TWEB), 2(2), 14:1-14:36.

[28] Sharma, V., & Dave, M. (2012). SQL and NoSQL Databases. International Journal of Advanced Research in Computer Science and Software Engineering, 2 (8).

[29] Thusoo, A., Sarma, J. Jain, N., Shao, Z., Chakka, P., & Murthy, R. (2009). Hive: a warehousing solution over a Map-Reduce framework. Proceedings of the VLDB Endowment, 2(2), 1626-1629.

[30] Weil, Kevin (2011). How Twitter Uses NoSQL. Recuperado (2014, junio 19) de http://readwrite.com/2011/01/02/how-twitter-uses-nosql.

[31] Wikibooks. (2010).Oracle and DB2, Comparison and Compatibility/Database Scaling/Shared Architectures ?. Recuperado (2014, junio 19) de http://en.wikibooks.org/wiki/Oracle_and_DB2,_Comparison_and_Compatibility/Database_Scaling/Shared_Architectures.

[32] Zhao, Z., Vicknair, C., Macias, M. , Nan, X., Chen, Y.,& Wilkins, D. (2010). A Comparison of a Graph Database and a Relational Database. ACM SE '10 the 48th Annual Southeast Regional Conference, 42.