CLASIFICACIÓN DE SISTEMAS GESTORES DE BASE DE DATOS SEGÚN EL TEOREMA DE CAP

SERGIO ANDRÉS SEPÚLVEDA CRUZ

INFORME FINAL MONOGRÁFICO

DIPLOMADO EN BIG DATA

UNIVERSIDAD LIBRE

FACULTAD DE INGENIERÍA

PROGRAMA DE INGENIERÍA DE SISTEMAS

BOGOTÁ D.C.

2017

1.TÍTULO

Clasificación de sistemas gestores de base de datos según el teorema de CAP

2. RESUMEN

Hoy en día, el uso de la información digital se encuentra en constante crecimiento, cada vez hay más usuarios de internet, cada vez hay más personas conectadas que requieren información de manera inmediata, para ser capaces de satisfacer dichas necesidades, los sistemas de información han optado por elegir infraestructuras de computación distribuidas, pues un clúster de computadores es capaz de procesar la información en menos tiempo de lo que se tardaría un solo equipo, sin embargo, entre los tipos de sistemas de computación distribuida, se deben identificar las ventajas y desventajas que nos ofrecen cada uno de ellos, de esta manera aparece el teorema de CAP, el cual formula que cualquier entorno de computación distribuida se verá obligado a elegir una de las siguientes opciones: descartar la consistencia, no asegurar disponibilidad, o hacer que el sistema sea intolerante a las particiones, según este teorema, es posible clasificar los sistemas gestores de base de datos utilizados sobre infraestructuras de computación distribuida, entre los cuales se pueden encontrar las bases de datos relacionales y las bases de datos NoSQL, estas últimas se dividen en distintas categorías dependiendo de cómo estructuran la información, pues estas bases de datos no poseen una estructura tan compleja como las bases de datos relacionales, lo que les permite manipular de una manera más eficiente, grandes volúmenes de información que son los que se acostumbran manejar en soluciones orientadas a Big Data.

Palabras clave: Teorema de CAP, Big Data, sistemas gestores de base de datos, NoSQL, sistemas distribuidos de computación.

2. ABSTRACT

Today, the use of digital information is constantly growing, there are more and more Internet users and more people connected who need information immediately, in order to be able to satisfy those needs, information systems have opted to choose distributed computing infrastructures, since a cluster of computers is able to process information in less time than it would take a single computer, however, among the types of distributed computing systems, there are advantages and disadvantages that must be identified, thus appears the CAP theorem, which states that any distributed computing environment will be forced to choose one of the following options: discard consistency, not ensure availability, or make the system intolerant to partitions, according to this theorem, it is possible to classify the database management systems used on distributed computing infrastructures, among which can be found relational databases and NoSQL databases, the latter are divided into different categories depending on how they structure the information, as these databases do not have a structure as complex as relational databases, which allows them manipulate more efficiently large volumes of information which are handled on Big Data-oriented solutions.

Keywords: CAP theorem, Big Data, database management systems, NoSQL, distributed computing systems.

3. INTRODUCCIÓN

Hoy en día la gran mayoría de aplicaciones apuntan a procesar información en periodos muy cortos de tiempo o en tiempo real, sin embargo, el constante aumento de la información generada por los miles de millones de usuarios de internet dificulta el cumplimiento de esta tarea, pues una sola máquina es incapaz de procesar tales volúmenes de información en tiempo real. Es por esto que se emplea el uso de sistemas de computación distribuida, ya que este tipo de infraestructuras reducen enormemente los tiempos de procesamiento de la información, convirtiéndolas en herramientas predilectas para soportar soluciones orientadas a Big Data.

Los sistemas distribuidos pueden clasificarse en 3 diferentes grupos según las características que puedan cumplir entre consistencia, disponibilidad y tolerancia a particiones; existe un teorema llamado teorema de Brewer o mejor conocido como teorema de CAP, dicho teorema dice que un sistema distribuido solamente puede cumplir con dos de las tres características mencionadas anteriormente. Esto supone una elección determinante para la implementación de sistemas distribuidos ya que se debe determinar qué tipo de sistemas son los más adecuados para ciertos tipos de tareas.

En el documento a continuación se describen los tipos de sistemas distribuidos según el teorema de CAP y ejemplos de uso de cada uno de ellos.

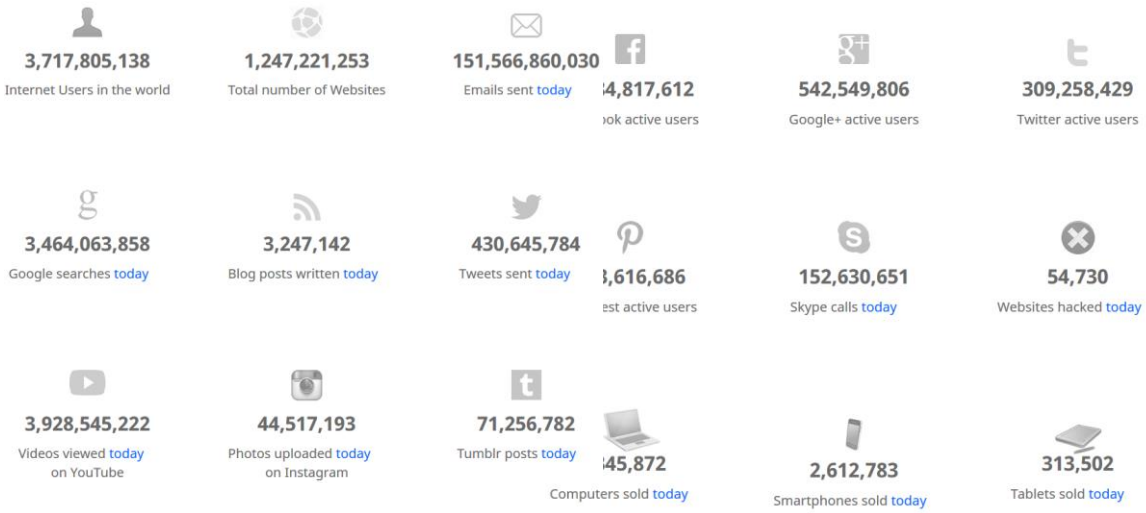
4. DESARROLLO

4.1. Volúmenes de información y entornos distribuidos

La gran mayoría de soluciones de Big Data están diseñadas para procesar y analizar volúmenes masivos de información, por tanto, se requieren aplicaciones que puedan procesar este tipo de cantidades en el menor tiempo posible para que la información analizada sea valiosa. Cabe recalcar que en Big Data se trabaja con volúmenes de datos que alcanzan cantidades superiores a los Petabytes (PB) de información, y aún más hoy en día, pues se transmiten cantidades que alcanzan los Exabytes (EB) es decir 1.000.000.000.000.000.000 bytes; en la figura a continuación se muestran algunas estadísticas del uso de internet que dan una cantidad aproximada de datos generados diariamente.

Figura 1. Cantidad de datos generados diariamente

Fuente: (http://www.internetlivestats.com/ Tomado a las 3:40 PM del 1 de Septiembre de 2017)



Teniendo en cuenta las cifras mostradas en la figura anterior, se hace obvio que una sola máquina es incapaz de procesar tales cantidades de datos en un tiempo óptimo que asegure el valor de la información (dicho valor disminuye a medida que aumenta el tiempo de su obtención), aquí es donde aparecen los entornos distribuidos los cuales permiten reducir enormemente los tiempos de procesamiento

de los datos; pero ¿Qué son los entornos distribuidos de datos? Los entornos distribuidos de datos o, mejor dicho, los sistemas de computación distribuida son “Sistemas en los cuales sus componentes están ubicados en computadoras comunicadas en red que coordinan sus acciones únicamente mediante la transmisión de mensajes” (George Coulouris, Jean Dillimore, Tim Kindberg, Gordon Blair, 2011). Lo anterior quiere decir que los sistemas de computación distribuida se componen de una serie de equipos cuyas características son similares, dichos equipos se comunican mediante una red para conformar una sola máquina la cual tiene especificaciones mucho mayores que los computadores que la conforman.

El aumento significativo de la capacidad de procesamiento de la información que ofrecen los sistemas de computación distribuidos, hace que sean la herramienta ideal para la implementación de soluciones de Big Data que requieren procesar máximas cantidades de datos en mínimos periodos de tiempo. No obstante, esta capacidad de procesamiento conlleva ciertas desventajas dependiendo del tipo de solución que se quiera implementar, para identificar estas desventajas y determinar qué solución es más adecuada para cual empresa se emplea el teorema de CAP el cual es una herramienta vital para identificar qué tipo de tecnología es la más adecuada para cada tipo de solución.

4.2. NoSQL como almacenamiento óptimo para Big Data

Hoy en día, la gran mayoría de las aplicaciones requiere atender todas las peticiones que reciben en tiempos de respuesta mínimos, incluso en tiempo real. Además de esto se requiere manejar cierto nivel de consistencia en los datos para que cada usuario acceda a la información que le corresponde, si bien, las bases de datos tradicionales manejan una fuerte consistencia de los datos, estas flaquean en temas de velocidad de respuesta, más aún cuando hablamos de cantidades masivas de información que son las que se tienen en entornos orientados a Big Data, pues las bases de datos tradicionales deben cargar todos los registros que extraen del disco duro, es decir desde el almacenamiento a la memoria RAM, esta tarea asegura la consistencia de los datos pero en detrimento de la velocidad de

respuesta, mientras que las bases de datos NoSQL (Not Only SQL), acceden directamente al almacenamiento para mostrar la información, esta alternativa reduce enormemente los tiempos de respuesta de cualquier aplicación, incluso cuando se hacen peticiones que contienen grandes volúmenes de datos. Las bases de datos NoSQL se definen como “Bases de datos que no son relacionales, de código abierto, son distribuidas por naturaleza, y del mismo modo poseen alto rendimiento, de un modo lineal que es escalable horizontalmente.” (Vatika Sharma, Meenu Dave, 2012) Gracias a su capacidad para reducir tiempos de respuesta, las bases de datos NoSQL adquirieron un gran auge y popularidad, este tipo de base de datos toma los principios básicos de las bases de datos tradicionales, tales como que cada objeto del mundo real se puede representar con una tabla y que dentro de una tabla hay un campo identificador de cada registro conocido como llave, sin embargo, se separa del sistema relacional que utilizan las bases de datos tradicionales, pues estas generalmente crean demasiadas restricciones que resultan incómodas para el diseño de una base de datos y sobre todo para su escalabilidad, este último siendo un aspecto fundamental en cualquier entorno de Big Data donde los volúmenes de información tienden a ser altamente variables. Esto se logra manejando cada registro como un objeto que posee un indicador y para consultar a dicho objeto, éste es ubicado gracias a su indicador, sin embargo, en una tabla de una base de datos NoSQL un indicador es capaz de repetirse, evento que sería inconcebible en una base de datos relacional, dado que la redundancia es inaceptable. Otro evento que se presenta en una tabla de una base de datos NoSQL es que los diversos registros dentro de una tabla pueden contener diversos formatos, esta característica permite almacenar información desde diferentes fuentes lo cual asegura la variedad, que es un aspecto fundamental que debe manejar cualquier entorno de Big Data. Entre los tipos de bases de datos NoSQL más utilizados podemos encontrar:

- Bases de datos tipo clave-valor
- Bases de datos documentales

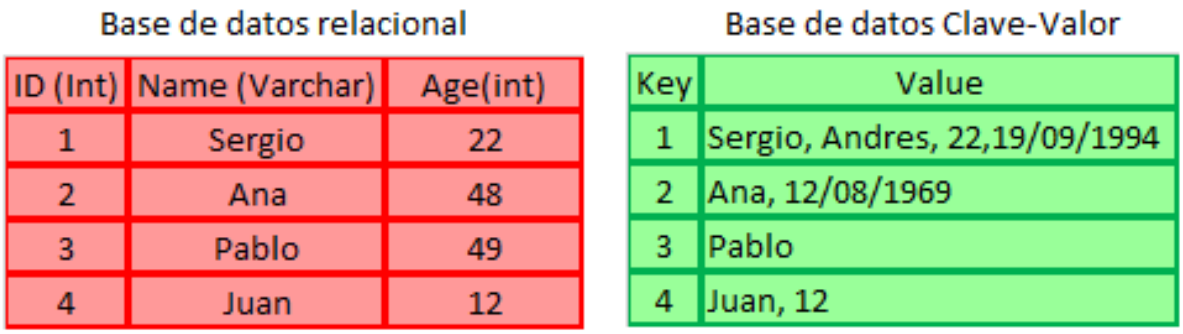
- Bases de datos orientadas a columnas
- Bases de datos orientadas a grafos

4.2.1. Bases de datos tipo clave-valor

Este tipo de bases de datos funciona mediante el uso de arreglos o listas asociativas mejor conocidas como diccionarios, dichos diccionarios contienen una gran cantidad de registros que se encuentran identificados por un valor único, dichos registros pueden poseer diversas variedades de campos, por ejemplo, en una base de datos tipo clave-valor puede existir un registro con solo campos numéricos y simultáneamente tener otro registro con campos numéricos y alfanuméricos, dicho ejemplo se ilustra en la figura a continuación:

Figura 2. Comparación entre base de datos relacional y Clave-Valor

Fuente: (Autor, 2017)



Este tipo de base de datos se puede clasificar en cómo manejan la consistencia de sus datos, sus clasificaciones son: Eventualmente consistentes, de almacenamiento en Memoria RAM y de Almacenamiento en discos rotativos o discos de estado sólido. Mientras que las bases de datos de consistencia eventual aseguran completa disponibilidad, las dos últimas aseguran consistencia en la información. Algunos ejemplos de estas bases de datos son:

- Dynamo DB
- Redis

- Riak

4.2.2. Bases de datos documentales

Son bases de datos que se encuentran orientadas a la administración, registro y consulta de la información que contienen diversos documentos, este tipo de base de datos podría considerarse como una derivación de las bases de datos clave-valor, pero se diferencian de estas por el modo en el que tratan los datos, pues en las bases de datos documentales, se emplea la meta data de un archivo (datos que describen a un documento) para optimizar las consultas de información. Dentro de este tipo de bases de datos se suelen registrar archivos de tipo XMM o JSON, e incluso se pueden almacenar archivos de tipo PDF, Word, etc. Este tipo de bases de datos almacenan los documentos como si se tratasen de objetos a los cuales les asigna un identificador único, este tipo de manejo permite buscar rápidamente grandes cantidades de información que pueden estar contenidas en un solo documento. Algunos de los ejemplos más destacados de este tipo de base de datos son.

- MongoDB
- CouchDB
- SimpleDB

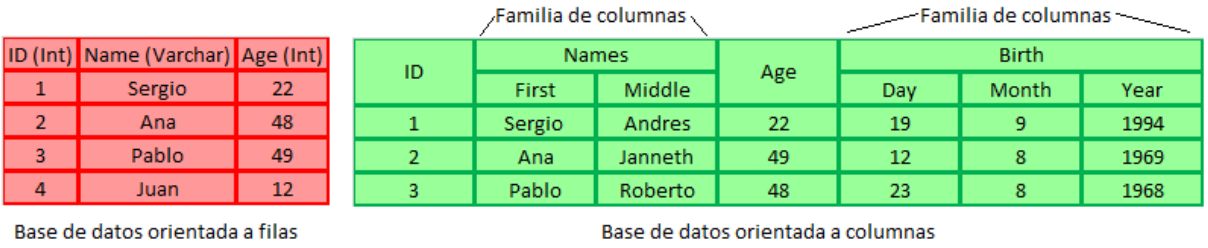
4.2.3. Bases de datos orientadas a columnas

Como su nombre lo indica, estas bases de datos se enfocan en almacenar la información en columnas y no en filas como en las bases de datos tradicionales, esto quiere decir que cada columna en una tabla será un nuevo registro, diseños de bases de datos como este permiten que las tablas, o mejor dicho, familias de columnas sean altamente escalables puesto que cada columna es independiente de las demás y puede guardar datos con un dato completamente distinto al de otras columnas de su familia, esto causa que los datos se puedan comprimir de una manera mucho más fácil que en una base de datos orientada a filas. En la figura a

continuación, se muestra un comparativo de una tabla de una base de datos tradicional y una familia de columnas en una base de datos orientada a columnas.

Figura 3. Comparación entre base de datos orientada a filas y orientada a columnas

Fuente: (Autor, 2017)



Esta manera de manejar la información ofrece como ventaja un fácil y rápido acceso al almacenamiento en disco, pues es posible realizar operaciones directamente sobre un disco duro sin tener que pasar por una memoria RAM, esto causa que el tiempo que requiere una consulta para extraer los datos se reduce significativamente, lo que se traduce en tiempos de respuesta mucho más cortos para cualquier aplicación. Algunos de los ejemplos más utilizados para este tipo de base de datos son:

- Cassandra
- HBase
- Google Bigtable

4.2.4. Bases de datos orientadas a grafos

Si bien la mayoría de bases de datos NoSQL se separan del concepto de relaciones entre elementos, las bases de datos orientadas a grafos hacen todo lo contrario, enfocándose en analizar las relaciones que existen entre los registros de una base de datos para representar la información en un grafo, donde los nodos son los datos como tal y las aristas que unen a los nodos son las relaciones que hay entre ellos, dicho grafo se utiliza para recorrer la base de datos de manera eficiente. Teniendo

en cuenta lo mencionado anteriormente, este tipo de estructuración de los datos permite que una base de datos orientada a grafos mantenga su rendimiento, sin importar que tanto crezca la base de datos, pues al realizarse una consulta, la base de datos revisará únicamente los nodos relevantes para dicha consulta e ignorará a los demás. Este tipo de consultas se logran añadiendo un puntero a cada registro el cual lo relaciona a otro, con esto se evitan el uso de índices en estas bases de datos. Algunos de los ejemplos más destacados son:

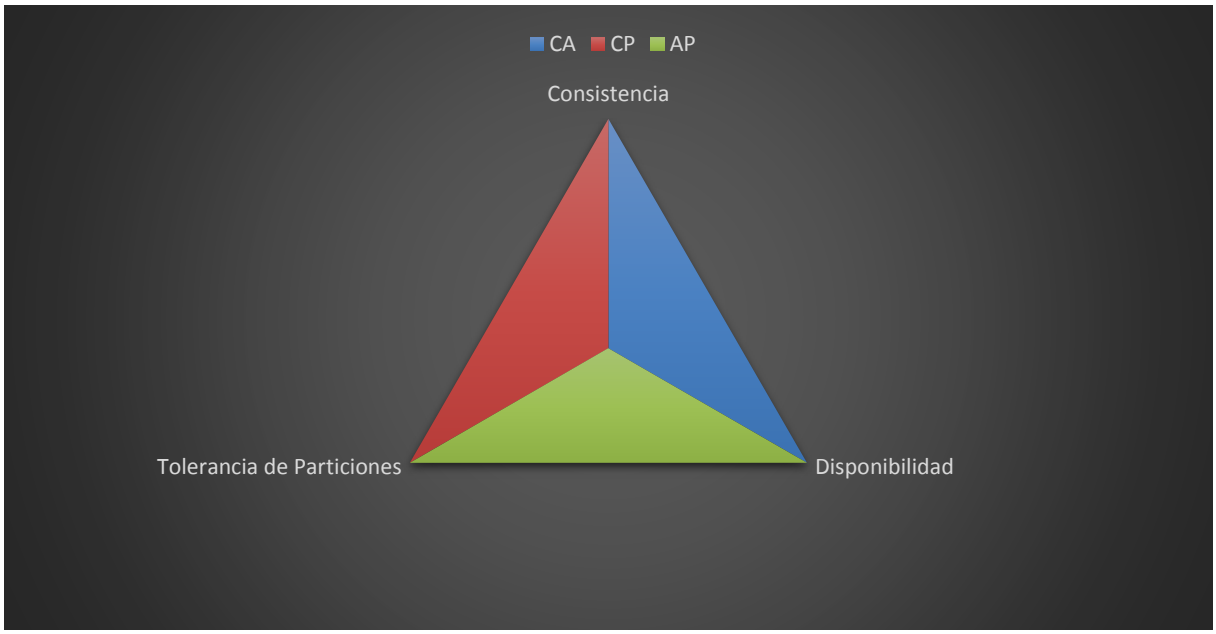
- Neo 4j
- Sparksee
- Titan

4.3. El teorema de CAP

El teorema de Brewer (Nombrado así gracias al científico Eric Brewer, quien formuló este teorema en el 2000), o mejor conocido como teorema de CAP afirma que en un sistema de computación distribuida sólo es posible asegurar 2 de 3 características fundamentales de estos sistemas, estas son: Consistencia (Consistency), Disponibilidad (Availability) y Tolerancia a particiones (Partition Tolerance). Esto quiere decir que cualquier solución de Big Data que sea implementada sobre un sistema de computación distribuido se verá obligada a descartar alguna de estas 3 características; dicho de manera más específica: un entorno distribuido que asegure la consistencia y la disponibilidad carecerá de tolerancia a particiones (sistemas tipo CA), un sistema que posea disponibilidad y tolerancia a particiones será poco consistente (sistemas tipo AP), así mismo un sistema consistente y tolerante a particiones no podrá estar siempre disponible (sistemas tipo CP). la figura a continuación muestra de manera resumida el aseguramiento de características que propone el teorema de CAP.

Figura 3. Tipos de sistemas distribuidos según teorema CAP

Fuente: (Autor, 2017)



4.3.1. Consistencia en el teorema de CAP

Como un sistema de computación distribuido es la interconexión de distintas computadoras que trabajan como una sola se debe asegurar que todos sus nodos registren la misma información y que dicha información esté disponible del mismo modo para cada equipo del sistema distribuido; a modo de ejemplo, si un nodo (A) realiza un registro de datos (X), éste debe comunicar a los demás nodos que X fue registrado en el sistema, esto asegura que si se consulta X desde un nodo (B) diferente de A, B sabrá que existe un registro de X y podrá acceder a él; además para asegurar esta consistencia la información registrada se duplicará para así tener una copia de seguridad de los datos.

Existen distintos tipos de modelos de consistencia para sistemas distribuidos los cuales aseguran dicha característica, estos son:

- Consistencia Estricta: cualquier operación realizada debe ser visto por todos los nodos simultáneamente (representa la mejor consistencia, pero siempre consumirá mayor tiempo de procesamiento)
- Consistencia Secuencial: Las operaciones no tienen que ser vistos por todos los nodos, pero todos los nodos deben ser capaces de ver la secuencia de operaciones ejecutadas.
- Consistencia Ocasional: Este modelo categoriza las operaciones en operaciones que están casualmente relacionadas y las que no, determinando que únicamente las operaciones que se encuentren relacionadas casualmente deben mostrar su secuencia de ejecución a todos los nodos.

4.3.2. Disponibilidad en el teorema de CAP

A pesar de que en un sistema distribuido todas las máquinas trabajen coordinadamente como una sola, no se debería dar el escenario en el que el malfuncionamiento de uno de los nodos del sistema cause el fallo de todo el entorno, esto se garantiza mediante la redundancia de datos, para crear copias de seguridad que garanticen que todos los nodos de un sistema distribuido tengan acceso a la información registrada así uno de ellos no se encuentre disponible.

Cabe recordar que para asegurar la disponibilidad de un sistema se debe detectar y reparar en el menor tiempo posible nodos que presenten fallas, porque si bien un entorno distribuido puede seguir funcionando así tenga un nodo faltante, la capacidad de procesamiento del sistema se verá reducida con cada nodo que no se encuentre en pleno funcionamiento.

4.3.3. Tolerancia a particiones del teorema de CAP

Como los sistemas distribuidos son conectados mediante una infraestructura de red, es posible inferir que no todos los nodos de un sistema distribuido no se encontrarán dentro de la misma ubicación geográfica, debido a esto pueden presentarse errores

de conexión entre los nodos de un sistema, dichas fallas causan que un sistema distribuido se divida, es decir que pase a tener diversas particiones; la tolerancia a particiones, es la capacidad de un sistema distribuido de poder seguir funcionando normalmente, incluso con fallas de conectividad entre sus nodos; nuevamente el hecho de tener copias de respaldo de la información en distintos nodos asegura que el sistema funcione normalmente, aun con fallas de conexión.

4.4. Tipos de sistemas distribuidos según teorema de CAP

Como ya se ha mencionado anteriormente el teorema de CAP divide a los sistemas distribuidos en 3 tipos según la característica que adolezcan, los sistemas CA son intolerantes a particiones, los sistemas CP no siempre están disponibles y los sistemas AP, pueden presentar inconsistencias en algunos momentos; desde luego estas desventajas pueden ser desalentadoras a primera vista, sin embargo, se debe comparar con una enorme ventaja que nos ofrecen los entornos de computación distribuida que es reducir enormemente los tiempos de procesamiento de la información para así poder entregar resultados en tiempo real, cabe recalcar que la antigüedad de la información es un factor clave para determinar su valor y si se eligen las herramientas correctas para hacer Big Data, una empresa puede contar con una herramienta determinante para sus decisiones corporativas.

4.4.1. Sistemas de tipo CA

Este tipo de sistemas de computación distribuida están diseñados para asegurar la consistencia de la información que tratan, gracias a diferentes protocolos de seguridad y de confirmación, como replicar la información de cada transacción realizada en distintos nodos para crear copias de seguridad constantemente o que una transacción sea confirmada por distintos nodos del sistema antes de realizarse; todas estas políticas para asegurar la consistencia hacen que el sistema sea altamente intolerante a las particiones, pues una falla de conexión entre los nodos, imposibilitaría las validaciones necesarias para asegurar la consistencia de la información y por tanto este tipo de sistemas deja de trabajar con normalidad al

tener particiones en las redes que conectan a sus nodos. Por tanto, la mejor manera de trabajar con estos tipos de sistemas es asegurando la conectividad de la red y así mismo soportarlas en una topología de red poco segmentada para reducir el riesgo de perder conectividad entre sus nodos, pues a mayor segmentación en la red de un sistema, mayores son las posibilidades de que este sea particionado debido a fallas de conexión.

4.4.1.1. ¿Qué tipo de soluciones son ideales para sistemas distribuidos CA?

Debido a la exigente consistencia que demandan este tipo de sistemas, se requiere usar una tecnología de almacenamiento de información que sea capaz de crear ciertas reglas que puedan asegurar que ésta sea coherente y no redundante. Dicho lo anterior, se hace evidente que las bases de datos relacionales tradicionales se convierten en la mejor solución, pues gracias al planteamiento de las formas normales, es posible crear una serie de reglas que estructuren la información de tal manera que se pueda asegurar un alto nivel de consistencia en los datos, a partir de esto emerge la siguiente pregunta: ¿Qué función desempeñan las bases de datos relacionales dentro del Big Data? Dicha pregunta surge a partir del paradigma de que es extremadamente complicado hacer que una solución de Big Data sea manejada por los sistemas de almacenamiento tradicionales, no obstante, toda solución de Big Data realiza un análisis de enormes volúmenes de datos los cuales son alimentados mediante varias fuentes de información, entre ellas se encuentran las bases de datos estructuradas, las cuales proveen información consistente y precisa que no necesita de un proceso exhaustivo de filtrado para ser analizada, pues su estructura evita que muchos datos innecesarios o “ruido” sea registrado. Por otra parte, este tipo de información debe estar siempre disponible, pues son datos que se utilizan con alta frecuencia y el que no esté disponible supondría un alto en la operación de la empresa (La mayoría de esta información son datos destinados al uso interno de las compañías). Los mejores ejemplos de sistemas de bases de datos relacionales son, Oracle Database, Microsoft SQL Server, MySQL y PostgreSQL.

4.4.1.2. Ejemplos de uso de sistemas distribuidos CA

Cualquier empresa que trabaje con información sensible que provenga de personas naturales, productos o propiedades e incluso otras empresas deberá asegurar que los datos que procese sean consistentes en todo momento y que correspondan correctamente a cada objeto de la vida real que están representando. Dados estos requerimientos, la consistencia se convierte en una característica crucial, la cual asegura que esta información sensible será tratada correctamente y así mismo se relacionará de una forma adecuada, el presentar información sensible de manera inconsistente puede acarrear problemas muy graves a cualquier empresa, tales como posibles demandas, pérdida de la fiabilidad de la empresa, pérdidas económicas e incluso incurrir en un acto delictivo. Como ejemplo las empresas que hacen un mayor uso de este tipo de tecnologías son las entidades bancarias ya que manejan información de las cuentas de sus clientes, así como las transacciones que hacen, también encontramos a las bases de datos de nómina como un claro ejemplo del uso de bases de datos relacionales.

4.4.2. Sistemas de tipo CP

Este tipo de entornos se caracterizan por priorizar la consistencia de los datos que manejan, a tal grado que se contempla el sacrificar la disponibilidad del sistema, haciendo que este mantenga en espera todas las operaciones que va a realizar mientras se asegura que todos los nodos están al tanto de las mismas, es decir que en caso de presentarse un particionamiento en el sistema, éste esperará a que los nodos se reconecten para continuar operando con normalidad. A pesar de tener ciertas fallas en materia de disponibilidad, este tipo de sistemas son altamente veloces ya que manejan bases de datos de tipo NoSQL que, al poseer una estructura simple para manejar los datos, estos son procesados más fácilmente en dichos sistemas que en una base de datos relacional.

4.4.2.1. ¿Qué tipo de soluciones son ideales para sistemas distribuidos CP?

Este tipo de sistemas son apetecidos gracias a su capacidad para procesar y entregar información consistente a partir de grandes volúmenes de información, en periodos muy cortos de tiempo, sin embargo, no se usan generalmente para manejar datos sensibles, debido a las falencias que poseen en cuanto a disponibilidad. No obstante, las altas velocidades de procesamiento que manejan las tecnologías orientadas a sistemas CP los convierten en sistemas muy necesarios para ciertas aplicaciones que requieren tiempos de respuesta inmediatos. Las bases de datos más adecuadas para este tipo de tareas son algunas bases de datos NoSQL, que, al manejar una estructura sencilla de datos, pueden procesarlos muy rápidamente mientras mantienen un buen nivel de consistencia en todo momento. Las bases de datos más utilizadas para este tipo de sistemas son: MongoDB, una base de datos documental que almacena documentos de tipo JSON, los cuales contienen toda la información relacionada con dicho documento. HBase, una base de datos orientada a columnas la cual puede distribuirse en varios servidores y es tolerante a las fallas de los mismos, almacena los datos en columnas las cuales agrupa dinámicamente como regiones. Redis, una base de datos de tipo clave-valor la cual se carga en memoria RAM para agilizar la carga de sitios web o de operaciones constantes dentro de una máquina.

4.4.2.2. Ejemplos de uso de sistemas distribuidos CP

Las altas velocidades que se logran con este tipo de sistemas, los hacen ideales para soportar aplicaciones de mensajería instantánea, por ejemplo, Facebook utiliza HBase para su aplicación de mensajería instantánea Messenger, puesto que su modelo orientado en columnas, el cual está basado en un modelo clave-valor, le permite identificar rápidamente el contenido que hay entre cada chat que mantiene un usuario. MongoDB es usado para realizar procesos de analítica y obtener una vista generalizada de toda la información de una empresa, también es utilizado para internet de las cosas mediante la lectura y análisis de datos que entregan múltiples sensores instalados en diversos artefactos, estos análisis son útiles para

diagnosticar y realizar mantenimientos predictivos, además también tiene usos en catálogos para e-commerce y el procesamiento de datos en tiempo real, siendo The Weather Channel un gran ejemplo pues utiliza MongoDB para atender en tiempo real, las múltiples peticiones que se le hacen a cada momento. El uso de almacenamiento en memoria de Redis, lo hace una herramienta ideal para manejar los datos que hay en el caché de las páginas web, esto aligera las cargas en el servidor y permite acceder más rápido a ellas, igualmente puede usarse para almacenar las sesiones de usuario, los contenidos del carrito de compras dentro de un e-commerce, crear contadores estadísticos en tiempo real y listas de elementos recientes, por ejemplo Twitter usa Redis para actualizar el timeline de cada usuario y Pinterest lo utiliza para administrar los tableros de imágenes que construye cada usuario.

4.4.3. Sistemas de tipo AP

Este tipo de sistemas distribuidos se caracterizan por manejar un tipo de consistencia llamada consistencia eventual, esta prioriza la disponibilidad del sistema sobre la validación de los datos, pues esta se irá corrigiendo con el paso del tiempo a medida que todos los nodos puedan verificar los datos, es decir, la información se publicará primero para mantener el sistema disponible y esta será verificada eventualmente. Este tipo de sistemas tienen como objetivo proveer una alta escalabilidad, mientras se mantienen constantemente disponibles. Nuevamente para este tipo de sistemas se prefiere el uso de bases de datos NoSQL, ya que las bases de datos tradicionales tienen unas reglas tan estrictas que las hacen muy poco escalables.

4.4.3.1. ¿Qué tipo de soluciones son ideales para sistemas distribuidos AP?

Este tipo de sistemas consideran la disponibilidad como característica fundamental que siempre debe cumplirse, por tanto, pueden permitirse la posibilidad de tener inconsistencias en la información durante algunos instantes, ya que el sistema seguirá funcionando incluso si existen nodos que no han verificado la información,

caso contrario a los sistemas tipo CP, pues estos detendrán la operación del sistema hasta que todos los nodos hayan validado la información que se va a operar, estas tecnologías implementan este modelo de consistencia ya que están diseñadas para trabajar sobre sistemas altamente distribuidos donde la posibilidad de que ocurran particiones es muy alta y gracias a la consistencia eventual aseguran que el sistema se mantenga siempre disponible. Este tipo de bases de datos son las indicadas para los sistemas distribuidos AP, entre ellas están presentes: Riak, la cual es una base de datos tipo clave-valor, la cual está diseñada para tener una alta disponibilidad y escalabilidad, estas son soportadas gracias a una alta replicación de clusteres. Otra base de datos bastante útil es Cassandra, que es una base de tipo clave-valor la cual está diseñada para tener una escalabilidad lineal, esto lo consigue comunicando los nodos bajo un protocolo P2P, en el que se replica constantemente la información, esto genera una redundancia alta de los datos que agiliza los procesos de sincronización de los nodos y de validación de los datos, cuando ocurre una partición de red en el sistema.

4.4.3.2. Ejemplos de uso de sistemas distribuidos AP

Este tipo de bases de datos brillan gracias a la disponibilidad constante que proveen y a su alta escalabilidad, por ejemplo, Cassandra es usado por Netflix para almacenar todos sus datos de streaming y el historial de vistas de los usuarios, otro buen ejemplo es Spotify, quien usa Cassandra para gestionar su sistema de sugerencias musicales a sus usuarios, del mismo modo eBay, utiliza Cassandra para soportar su motor de recomendaciones. Otra tecnología que es ampliamente usada es Riak la cual es usada por Riot Games para soportar el sistema de estadísticas para cada partida, el sistema de chat in-game de League Of Legends, un juego con más de 100 millones de usuarios activos, otro ejemplo es Best Buy, quien usa Riak para soportar su catálogo de productos, también es usado por The Weather Channel para recopilar información de diversos sensores que usan para realizar mediciones climáticas.

4.4.4. Comparación entre los tipos de sistemas distribuidos

Todos estos tipos de sistemas distribuidos presentan alguna desventaja según el teorema de CAP, sin embargo, también presentan ventajas de peso para elegirlos. Por ejemplo, los sistemas de tipo CA son muy poco escalables debido a su carente tolerancia a particiones, no obstante, aseguran una disponibilidad constante y una fuerte consistencia, ideal para el manejo de datos sensibles, los cuales se usan con suma frecuencia y no se pueden presentar inconsistencias en ellos. Los sistemas de tipo CP, tienen una disponibilidad que presenta intermitencias, sin embargo, la fácil escalabilidad que ofrecen, junto con la alta velocidad de operaciones lo convierten en excelentes herramientas para aplicaciones que requieran funcionamiento en tiempo real, y los sistemas CP a pesar de que no son constantemente consistentes, compensan esta desventaja con una amplia escalabilidad y disponibilidad constantes. La siguiente figura ilustra una comparación resumida entre los tipos de sistemas distribuidos según el teorema de CAP.

Tabla 1. Comparación de sistemas distribuidos

Fuente: (Autor, 2017)

Característica	Sistemas CA	Sistemas CP	Sistemas AP
Ventajas	-Disponibilidad constante -Consistencia estricta	-Consistencia sólida -Amplia escalabilidad	-Disponibilidad constante -Amplia escalabilidad
Desventajas	Poca escalabilidad	Disponibilidad sujeta a conexiones de nodo	Los datos no son consistentes en todo momento
Tipo de BD que utiliza	Bases de datos Relacionales	Bases de datos tipo NoSQL	Bases de datos tipo NoSQL
Ejemplos de BD	-MySQL -Oracle Database -SQL Server	-Mongo DB -HBase -Redis	-Cassandra -Riak -CouchDB

5. CONCLUSIONES

La masificación de internet ha provocado un aumento exponencial en la cantidad de información digital que se genera diariamente, como consecuencia, las bases de datos relacionales se hacen cada vez menos eficientes, esto se debe a que sus tiempos de respuesta se hacen demasiado largos al procesar grandes volúmenes de información, en respuesta a esta problemática han aparecido las bases de datos NoSQL, las cuales se usan cada vez más, gracias a que reducen ampliamente los tiempos de respuesta al manejar grandes cantidades de datos.

Si bien el teorema de CAP plantea tres posibles modelos de sistemas distribuidos, basados en el cumplimiento de 2 de 3 características, los entornos orientados a Big Data reducen estas posibilidades a solamente dos modelos de sistemas distribuidos, los sistemas tipo CP y los de tipo AP, es decir que se convierte en una elección entre disponibilidad y consistencia, esto se debe a que se requiere utilizar clusterización para manejar datos de manera masiva, generalmente estos clústeres se encuentran altamente segmentados, lo que se traduce en una alta probabilidad de tener particiones entre los nodos y como consecuencia, los sistemas distribuidos de tipo CA son poco viables para Big Data.

Las bases de datos relacionales, aún tienen una fuerte presencia dentro de las empresas, pues gracias a sus estrictas reglas y relaciones, aseguran una fuerte consistencia en los datos, cualidad que es muy apetecible para soluciones que manejan frecuentemente datos sensibles, los cuales en ningún momento pueden presentar inconsistencias, esto sumado a que la mayoría de bases de datos NoSQL, son open source, a pesar de que el uso de este tipo de aplicaciones reduce enormemente los costos, no gustan entre las empresas debido a que ofrecen muy poco, o ningún soporte técnico.

Entre los sistemas distribuidos orientados a Big Data, existen dos características que sobresalen y que se hacen necesarias, la escalabilidad, que viene junto a la tolerancia a particiones y la velocidad de respuesta, pues cada vez más aplicaciones

requieren funcionar en tiempo real, estas dos cualidades hacen que la tolerancia a particiones se convierta en un requerimiento crucial para cualquier infraestructura de Big Data.

6. REFERENCIAS

6.1. Bibliografía

George Coulouris, Jean Dollimore, Tim Kindberg, Gordon Blair (2011, Noviembre)

Distributed Systems, concepts and design 5th edition [Sistemas distribuidos, conceptos y diseño 5ta edición]. Publicado en: https://github.com/tsani/school/blob/master/general/george-coulouris-distributed-systems-concepts-and-design-5th-edition.pdf

Vatika Sharma, Meenu Dave (2012, Agosto)

SQL and NoSQL Databases [Bases de datos SQL y NoSQL], Publicado en: https://s3.amazonaws.com/academia.edu.documents/33559632/V2I800154.pdf?AWSAccessKeyId=AKIAIWOWYYGZ2Y53UL3A&Expires=1504657308&Signature=24ApKjHYGGBMb9bWFVQnGXCsSyk%3D&response-content-disposition=inline%3B%20filename%3DSQL_and_NoSQL_Databases.pdf

Raúl Hernanz Gómez (2014, Junio)

Bases de datos NoSQL: Arquitectura y ejemplos de aplicación. Publicado en:

https://e-archivo.uc3m.es/bitstream/handle/10016/22895/PFC_raul_herranz_gomez_2014.pdf

6.2. Webgrafía

http://www.julianbrowne.com/article/viewer/brewers-cap-theorem

http://blog.nahurst.com/visual-guide-to-nosql-systems

https://code.facebook.com/posts/321111638043166/hydrabase-the-evolution-of-hbase-facebook/

https://redis.io/topics/whos-using-redis

https://www.mongodb.com/use-cases

https://hbase.apache.org/poweredbyhbase.html

https://www.datastax.com/resources/casestudies

http://basho.com/use-cases/

http://basho.com/tag/case-study/