Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 11, No.12, Mes Diciembre, 2018
ISSN: 2306-2495 | RNPS: 2343
Pág. 47-59
http://publicaciones.uci.cu

Tipo de artículo: Artículo original
Temática: seleccionar la temática a partir de las líneas editoriales de la revista
Recibido: 18/09/16 | Aceptado: 20/11/16 | Publicado: 20/12/2018

Propuesta de clusterización en base de datos nosql cassandra

Propuesta de clusterización en base de datos nosql cassandra

Ismael Gómez González[1], José Leandro González[2]

[1] DATEC, Facultad CITEC, Universidad de las Ciencias Informáticas.
[2] DATEC, Facultad CITEC, Universidad de las Ciencias Informáticas..

* Autor para correspondencia: ismaelgg@uci.cu

Resumen

Las bases de datos constituyen el fundamento de los sistemas de información y el soporte para la gestión de los datos y la toma de decisiones. Ellas facilitan: el almacenamiento de grandes cantidades de datos, la organización y reorganización de la información y su impresión y distribución de varias formas. La clusterización en los sistemas de bases de datos relacionales es uno de los temas más novedosos de las ciencias informáticas en estos días. Puesto que permite obtener un mayor rendimiento a medida que se incrementan las peticiones de los usuarios, debido a que siempre se debe prever que las bases de datos lleguen a crecer hasta el punto de que tanto la carga del servidor como el espacio pueden llegar a convertirse en un problema. Sin embargo hay algunas necesidades concretas donde las bases de datos SQL no cubren suficientemente las peticiones requeridas y por lo tanto se han desarrollado bases de datos no relacionales para poder cubrirlas. La presente investigación realiza un estudio exhaustivo sobre el impacto de la clusterización en bases de datos no relacionales resaltando las ventajas que brinda la implementación de un clúster en el sistema de bases de datos Cassandra con respecto a los sistemas de bases de datos relacionales, específicamente Postgres SQL.

Palabras clave: Clúster, Clusterización, Bases de datos, NoSQL, Cassandra, Nodo

Abstract

Las bases de los datos se basan en el fondo de los sistemas de información y el soporte para la gestión de los datos y la toma de decisiones. Ellas facilitan: el almacenamiento de grandes cantidades de datos, la organización y la reorganización de la información y su impresión y distribución de varias formas. La agrupación en los sistemas de bases de datos se relaciona con uno de los temas más novedosos de las ciencias informáticas en estos días. ¿Cómo

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 11, No.12, Mes Diciembre, 2018
ISSN: 2306-2495 | RNPS: 2343
Pág. 47-59
http://publicaciones.uci.cu

puedo obtener un mayor rendimiento? ¿Se incrementa? ¿Se incrementa? ¿Se incrementa? ¿No? en un problema Sin embargo, hay algunas necesidades en donde las bases de datos no cubren suficientemente las peticiones requeridas y así se han desarrollado las bases de datos no relacionadas para poder cubrirlas. La presente investigación realiza un estudio exhaustivo sobre el impacto de la agrupación en las bases de datos no relacionales las ventajas que brinda la aplicación de un clúster en el sistema de bases de datos Cassandra con respecto a los sistemas de bases de datos relacionales, específicamente Postgres SQL

***Keywords**: Clúster, Clusterización, Bases de datos, NoSQL, Cassandra, Nodo*

Introducción

El desarrollo de la informática ha posibilitado el surgimiento de metodologías y productos capaces de ayudar al hombre a realizar sus tareas cotidianas, contribuyendo así a una adecuada organización de su trabajo debido a que permiten adquirir un mejor control de sus recursos e información. La informática traza nuevas metas a las ciencias modernas y de forma particular a las de la información, las cuales son un instrumento imprescindible del desarrollo social, político y económico de los países. En el mundo informatizado de hoy, se observa un alto nivel de desarrollo científico-técnico, donde simples datos estadísticos constituyen una balanza de fortuna para las grandes compañías, ya que el análisis e interpretación que se les den a estos datos determina en un gran porciento el éxito de sus proyectos. Por lo que es necesario desarrollar búsquedas lo más exactas posibles dentro del enorme caudal de información que existe, para lo que deben ser empleados mejores métodos y herramientas para el almacenamiento y gestión de los datos (HAN *et al.* 2011).

A medida que transcurren los años se evidencia cada vez más cómo estas herramientas de búsqueda y gestión de la información se van desarrollando en función de las necesidades actuales de la sociedad. Las bases de datos como parte de ellas, hoy en día ocupan un lugar determinante en cualquier área del quehacer humano, comercial, y tecnológico; ya que se han convertido en un producto estratégico de primer orden, al constituir el fundamento de los sistemas de información y el soporte para la gestión de los datos y la toma de decisiones. Ellas facilitan: el almacenamiento de grandes cantidades de datos, la organización y reorganización de la información y su impresión y distribución de varias formas (CATTELL 2011), (LEAVITT 2010).

La mayoría de los sistemas de bases de datos más populares en la actualidad se basan en la arquitectura relacional, y todos ellos utilizan el lenguaje de consultas SQL para operar con los datos. Tanto es así, que SQL se ha convertido con el paso de los años en un estándar "de facto", debido a su uso (STRAUCH *et al.* 2011), (TIWARI 2011).

Resulta innegable el buen rendimiento que comprenden estas herramientas de gestión de almacenamiento, principalmente porque se ha logrado aprender técnicas para normalizarlas en la medida de lo posible y escalarlas según crece la demanda. Pero con el surgimiento y desarrollo de las aplicaciones web llegaron los problemas de alta escalabilidad. Si bien los modelos relacionales se pueden adaptar para hacerlos escalar incluso en los entornos más difíciles, sí que es cierto que, a menudo, se hacen cada vez menos intuitivos a medida que aumenta la complejidad. Triples y cuádruples JOINs en consultas SQL que asustan nada más verlas, a veces poco eficientes, y sistemas de almacenamiento de resultados en cachés para acelerar la resolución de las peticiones y evitar ejecutar cada vez estas pesadas operaciones, son el pan de cada día en muchos de estos proyectos de software con el uso de estas herramientas (HECHT and JABLONSKI 2011), (TUDORICA and BUCUR 2011), (MONIRUZZAMAN and HOSSAIN 2013), *(MAR-CORNELIO et al. 2019).*

Los sistemas no Relacionales (NoSQL) intentan atacar este problema proponiendo una estructura de almacenamiento más versátil, aunque sea a costa de perder ciertas funcionalidades como las transacciones que engloban operaciones en más de una colección de datos, o la incapacidad de ejecutar el producto cartesiano de dos tablas teniendo que recurrir a la desnormalización de datos. La presente investigación, describe la implementación de cluster en el sistema Cassandra *(CORNELIO and GULÍN 2018), (MAR, O et al. 2016).*

Materiales y métodos

Para implementar la propuesta de cluster se utiliza Cassandra siendo un sistema distribuido y descentralizado que permite ejecutarse sobre múltiples máquinas,. Todos los nodos en el clúster o grupo de máquinas funcionan del mismo modo. A ese comportamiento típico de esta base de datos se le denomina servidor simétrico (OKMAN *et al.* 2011), (OKMAN *et al.* 2011),

La descentralización tiene dos ventajas claves: Puede ser más fácil de operar y mantener un almacén de datos descentralizado que uno Administrador/Esclavo ya que todos los nodos son iguales, y no se requieren conocimientos adicionales para escalarlo; configurar 50 máquinas no es diferente a configurar una sola. Como en Cassandra todos los nodos son idénticos, la inutilización de uno no interrumpe su funcionamiento. En pocas palabras, como Cassandra es distribuido y descentralizado, no hay un punto de fallo lo que genera una alta disponibilidad.

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 11, No.12, Mes Diciembre, 2018
ISSN: 2306-2495 | RNPS: 2343
Pág. 47-59
http://publicaciones.uci.cu

Un elemento necesario en el proceso de clustarizacion lo representa el nivel de escalabilidad, puede ser alcanzado mediante escalabiliad horaizontal y escalabilidad Vertical. El principio de escalabilidad vertical es atribuido al crecimiento en el procesamiento (Terminar idea), sin embargo la escalabilidad horaizontal parte del principio de adicionar mayor cantidad de nodos. (ABRAMOVA and BERNARDINO 2013)

La descentralización tiene dos ventajas claves: Puede ser más fácil de operar y mantener un almacén de datos descentralizado que uno Administrador/Esclavo ya que todos los nodos son iguales, y no se requieren conocimientos adicionales para escalarlo; configurar 50 máquinas no es diferente a configurar una sola. Como en Cassandra todos los nodos son idénticos, la inutilización de uno no interrumpe su funcionamiento. En pocas palabras, como Cassandra es distribuido y descentralizado, no hay un punto de fallo lo que genera una alta disponibilidad *(MAR CORNELIO et al. 2016).*

Un elemento necesario en el proceso de clustarizacion lo representa el nivel de escalabilidad, puede ser alcanzado mediante escalabiliad horaizontal y escalabilidad Vertical. El principio de escalabilidad vertical es atribuido al crecimiento en el procesamiento (Terminar idea), sin embargo la escalabilidad horaizontal parte del principio de adicionar mayor cantidad de nodos.

Partiendo del proceso de descentalizacion con una escalabilidad horizontal, casandra implementa principios para su aplicación. La figura 1, muestra el cunjunto de principios teóricos que sustentan la clusterizacion en la propuesta presentada. Uno de los puntos fuertes de Cassandra es su escalabilidad. Mientras muchos otros proyectos necesitan código adicional para poder crecer en capacidad, Cassandra ha sido diseñado para ser totalmente descentralizado, y crecer elásticamente añadiendo nodos adicionales. La facilidad de Cassandra de crecer y reducir el tamaño del clúster elásticamente, es uno de los puntos más fuertes de esta base de datos NoSQL. Aunque se puede realizar, no resulta provechoso ejecutar Cassandra en un solo nodo porque se estarían prescindiendo de estas características que la distinguen de los demás sistemas de bases de datos *(MAR, OMAR et al. 2015)*.

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 11, No.12, Mes Diciembre, 2018
ISSN: 2306-2495 | RNPS: 2343
Pág. 47-59
http://publicaciones.uci.cu

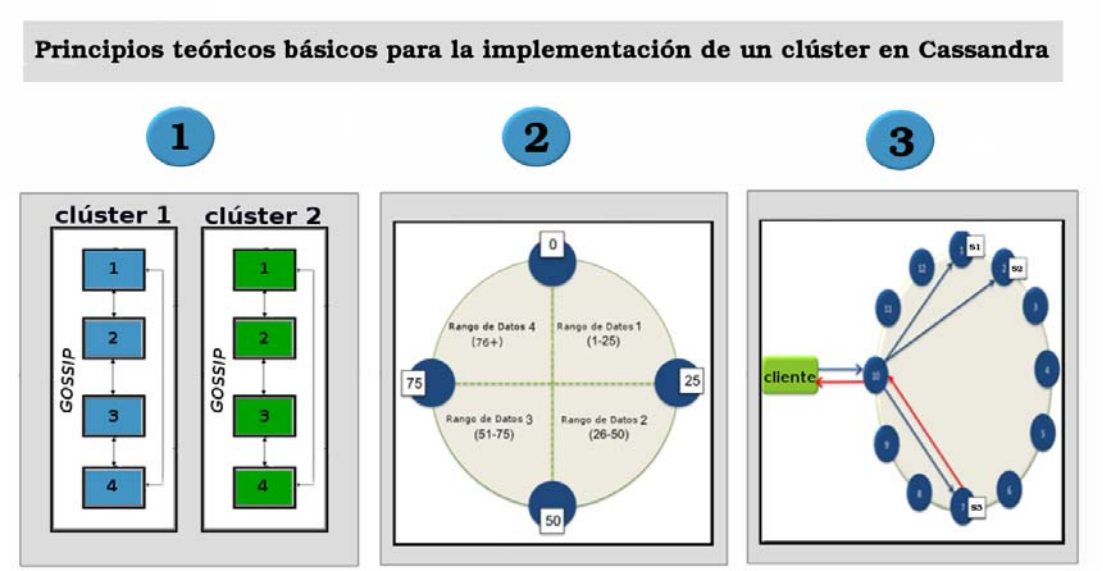

Figura 1: principios teóricos de la clusterización.

La figura 1, representa los factores que pueden contribuir al alcance de un mayor rendimiento en la implementación de la propuesta constituyendo un pilar para la correcta compresión acerca de la forma en que se internamente en su proceso de clusterización. A continuación se describne las tres principios del proceso:

- **Principio 1: Comunicación entre nodos**

Cassandra usa un protocolo llamado Gossip para descubrir la localización y estado acerca de los otros nodos que integran el clúster del sistema. Dicho protocolo establece una comunicación punto-a-punto en la cual los nodos periódicamente intercambian información acerca de sí mismos y acerca de otros nodos que conocen. El proceso que ejecuta el protocolo Gossip corre a cada segundo y permite intercambiar los mensajes de estado a un máximo de tres nodos del clúster. Cada nodo intercambia información acerca de su estado y también acerca del estado de otros nodos que conoce, permitiendo el rápido estudio del comportamiento de los datos en los nodos que componen el clúster.

Cuando un nodo inicia por primera vez, se debe de editar su fichero de configuración para determinar el nombre del clúster de Cassandra y a qué nodo llamado semilla es al que pertenece, para de esta manera obtener información acerca de los otros nodos en el clúster. El fichero de configuración que tiene estos puntos de contacto es el llamado "cassandra.yaml". Al nodo que está en contacto con otros nodos cuando estos se ponen en marcha por primera vez y se unen al clúster, es al que se le denomina ***nodo semilla***.

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 11, No.12, Mes Diciembre, 2018
ISSN: 2306-2495 | RNPS: 2343
Pág. 47-59
http://publicaciones.uci.cu

Un clúster o anillo puede tener varios nodos semillas. Cassandra utiliza el antes mencionado protocolo gossip para descubrir la ubicación y la información de estado acerca de los otros nodos que participan en un clúster de Cassandra. Cuando un nodo se inicia por primera vez, entra en contacto con un nodo semilla para iniciar el proceso de comunicación gossip. El propósito del nodo semilla es el inicio del proceso de comunicación gossip para los nuevos nodos que se unen al clúster. Los nodos semillas no son un punto único de fallo, ni tampoco tienen ningún otro propósito especial en las operaciones del clúster más allá de la secuencia de arranque de los nodos.

- **Principio 2: Distribución de los datos**

Una vez asignados los nodos y almacenada cierta cantidad de datos en uno de ellos, se procede a la distribución de los mismos dentro del clúster -esto no implica de ninguna forma que luego de distribuidos los datos no se puedan asignar nuevos nodos al clúster-. Para saber hasta que rango de datos (entiéndase rango como porción de datos dentro de una tabla) es responsable cada nodo, el mismo debe conocer sus propios tokens (símbolos) y los símbolos de aquellos que pertenecen a los otros nodos del clúster.

Luego de inicializo el grupo de máquinas que comprende el sistema, se deben generar tokens para todo el clúster y asignar un token inicial para cada nodo antes de la puesta en marcha. Cada nodo hará público su token a los demás. A la hora de inicializar un clúster de Cassandra, se debe seleccionar cómo los datos serán divididos a través de los nodos en el clúster. Esto se hace mediante la elección de un partidor para el clúster. Cassandra utiliza un algoritmo de Tabla de Hash Distribuido (DHT) para determinar cuando los datos se almacenan en un nodo con un token determinado. La tabla 1, visualiza la de distribucin de nodos en el cluster con sus respectivos rangos de datos.

Tabla 1: Distribución de los nodos

Servidor	Token Inicial	Rango Responsable
0	0	2**127/4 * 3 - 0
1	2**127/4	0 - 2**127/4
2	2*(2**127/4)	2**127/4 - 2*(2**127/4)
3	3*(2**127/4)	2*(2**127/4) - 3*(2**127/4)

Cada servidor de Cassandra se configura asignando un valor simbólico en la sentencia initial_token del fichero de configuración. Esta asignación debe estar regida por la siguiente fórmula matemática:

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 11, No.12, Mes Diciembre, 2018
ISSN: 2306-2495 | RNPS: 2343
Pág. 47-59
http://publicaciones.uci.cu

i(2**127)/ número_de_nodos con i comenzando desde 0 hasta (número_de_nodos – 1)*

Esta fórmula permite que los datos almacenados en el sistema se distribuyan de forma equitativa sobre todos los nodos que comprende el clúster.

El valor 2 a la potencia de 127 representa el espacio de nombres DHT de 128 bits utilizado por Cassandra. La siguiente tabla muestra la asignación correcta de tokens para un clúster de 4 máquinas.

El total de datos administrados por el clúster es representado con un espacio circular o anillo. El anillo está dividido en intervalos iguales al número de nodos, con cada nodo siendo responsable de uno o más rangos del total de datos. Antes de que un nodo pueda unirse al anillo, debe tener asignado el token, el cual determina las posiciones del nodo en el anillo y el rango de datos del cual es responsable.

Los datos de las familias de columnas son particionados a través de nodos basándose en su identificador de fila. Para determinar el nodo donde estará la primera réplica, el anillo camina en el sentido de las agujas del reloj hasta que se localiza el nodo con un valor de token mayor que el del identificador de fila. Cada nodo es responsable de su región del anillo y la de su precursor. Con los nodos ordenados en orden simbólico, el último nodo es considerado como el predecesor del primer nodo.

Otro aspecto esencial dentro del proceso de clusterización de Cassandra es la replicación; que no es más que el proceso de almacenamiento de copias sobre múltiples nodos para asegurar en todo momento la disponibilidad y tolerancia a fallos de los datos. Cuando se crea un Keyspace en Cassandra (análogo a decir, un esquema en una base de datos relacional) se debe decidir la estrategia de colocación de réplicas, el cual determina el número de réplicas y cómo estas son distribuidas a través de los nodos en el clúster. La estrategia de replicación depende de un clúster debidamente configurado para ayudar a determinar la ubicación física de los nodos y su proximidad entre sí.

El número total de réplicas de todo el clúster hace referencia al factor de replicación. Un factor de replicación de 1 significa que sólo hay una copia de cada fila. Un factor de replicación de 2 significa dos copias de cada fila. Todas las réplicas son igualmente importantes, no hay réplica primaria o principal en términos de cómo leer y escribir el manejo de las peticiones.

Como una regla general, el factor de replicación no debe de exceder el número de nodos en el clúster. Sin embargo, es posible incrementar el factor de replicación y luego agregar el número deseado de nodos después.

- **Principio 3: Replicación de los datos**

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 11, No.12, Mes Diciembre, 2018
ISSN: 2306-2495 | RNPS: 2343
Pág. 47-59
http://publicaciones.uci.cu

Existen 2 estrategias básicas para replicar la información en Cassandra:

- SimpleStrategy: Coloca las réplicas de los datos a todos los nodos que están juntos en el mismo clúster.
- NetworkTopologyStrategy: Configura el número de réplicas por Centros de Datos. Es exclusivamente una estrategia de colocación de réplicas para clústers con múltiples Centros de Datos.

La presente investigación realizó mayor énfasis en la generación de un clúster con estrategia de colocación de réplica simple debido a que es la que más se adecua a las actuales condiciones tecnológicas que existen en el centro donde radica su autor. A continuación se muestra la sentencia a utilizar para el uso de la estrategia de colocación simple, definiéndose el factor de replicación a utilizar.

create keyspace tesis with placement_strategy ='org.apache.cassandra.locator.SimpleStrategy'
[{replication_factor: #(HAN et al. 2011)];

Resultados y discusión

Implementación de un clúster multinodos en Cassandra

Los valores predeterminados del fichero cassandra.yaml perteneciente al directorio /conf de la carpeta de instalación de Cassandra son muy buenos para levantar el sistema por primera vez y hacerlo correr en un solo nodo. Sin embargo, es inadecuado para su uso en un grupo de múltiples nodos. La configuración y los procesos que se indican en este epígrafe son la forma más sencilla para crear un clúster de varios nodos.

Trabajando con el primer nodo:

Los valores por defecto del fichero cassandra.yaml tienen configuradas las direcciones de escucha (para la conexión entre nodos) y Thrift (acceso de los clientes) de forma local, como se muestra a continuación:

listen_address: locahost

rpc_address: localhost

A medida que la dirección de escucha sea utilizada para la comunicación dentro del clúster, debe ser cambiada a una dirección enrutable para que los otros nodos puedan llegar a ella. Por ejemplo, suponiendo que su ordenador tiene una interfaz Ethernet con la dirección 10.208.2.40, tendría que cambiar la dirección de escucha de esta manera:

listen_address: 10.208.2.40

La interfaz Thrift puede ser configurada usando una dirección especificada, al igual que la dirección de escucha, o el uso del comodín 0.0.0.0. Lo que hace Cassandra para escuchar a los clientes en todas las interfaces disponibles. Esta dirección se actualiza de la siguiente manera:

rpc_address: 10.208.2.40

Tal vez la máquina tenga una segunda tarjeta de red con IP 10.208.2.122 para así dividir el tráfico de la red interna del clúster del de Thrift para mejor rendimiento.

Configurándose de la siguiente manera:

rpc_address: 10.208.2.122

Si la entrada DNS de su servidor es correcta, es seguro usar un nombre de host en lugar de una dirección IP. Del mismo modo, la información de la instrucción seed debe cambiar su dirección. Por defecto estaba configurada de la siguiente manera:

seeds:
- 127.0.0.1

La nueva configuración sería:

seeds:
- 10.208.2.40

Una vez que estos cambios fueron realizados, simplemente se debe reiniciar Cassandra en ese nodo. Se debe usar la sentencia netstat (*netstat -ant | grep 7000)* para verificar que Cassandra está escuchando en la dirección correcta, apareciendo una línea como esta:

tcp 0 0 10.208.2.40.7000 *.* ESCUCHAR

Si netstat muestra que Cassandra todavía escucha en 127.0.0.1.7000, entonces o bien el proceso de Cassandra anterior no fue terminado correctamente o no se está editando el archivo cassandra.yaml.

Trabajando con el resto de los nodos:

Los otros nodos del clúster usarán el fichero cassandra.yaml casi idéntico al del primer nodo configurado. Por lo que se debe usar la configuración del primer nodo como la base de estos cambios en lugar de los valores por defecto de este fichero. El primer cambio es convertir en automático el bootstrapping, esto hará posible la unión del nodo al anillo e intentará tomar el control de una amplia gama de espacios.

auto_bootstrap: true

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 11, No.12, Mes Diciembre, 2018
ISSN: 2306-2495 | RNPS: 2343
Pág. 47-59
http://publicaciones.uci.cu

El segundo cambio es a la dirección de escucha, ya que no debe ser también el bucle de retorno y no puede ser el mismo que cualquier otro nodo. Suponiendo que el segundo nodo dispone de una interfaz Ethernet con la dirección 10.208.2.30, se establece su dirección de escuchar con:

listen_address: 10.208.2.30

Por último, debe de actualizar la dirección del Thrift para aceptar conexiones de clientes, como con el primer nodo, ya sea con una dirección específica o comodín:

rpc_address: 10.208.2.30

Se debe tener en cuenta el hecho de no realizar cambios en la configuración de la sección Seeds, que es la que los nuevos nodos utilizan para usar el primer nodo para el arranque. Una vez que se realicen estos cambios, inicie Casandra en el nuevo nodo y automáticamente se unirá el anillo.

Reemplazo de nodos caídos

Para reemplazar un nodo del clúster que haya muerto –por fallas de hardware, por ejemplo– se debe añadir un nuevo nodo en su lugar mediante el parámetro *-Dcassandra.replace_token=<token>*, esta instrucción permite que el nuevo nodo añadido al anillo asuma la posición del token del nodo que ha muerto. Para poder reemplazar nodos de esta manera:

1. El token que se le asigna al nuevo nodo debe de corresponder con el token del nodo caído – si se trata de reemplazar un nodo asignándole el token de un nodo activo se lanzaría una excepción en el sistema–.
2. El token que se le asigna al nuevo nodo debe ser parte del anillo.
3. El nuevo nodo que se une al clúster no puede contener datos –el directorio de datos debe estar vacío si se desea realizar un reemplazo–.

Pasos para reemplazar un nodo caído:

1. Confirmar la existencia del nodo caído usando el comando nodetool ring sobre cualquiera de los otros nodos activos del clúster. La Figura 10 se muestra un ejemplo, en el cual se observa que el nodo con dirección IP 10.46.123.12 está en estado caído.

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 11, No.12, Mes Diciembre, 2018
ISSN: 2306-2495 | RNPS: 2343
Pág. 47-59
http://publicaciones.uci.cu

```
$ nodetool ring -h localhost

Address         DC          Rack        Status State   Load            Owns    Token
10.46.123.11    datacenter1 rack1       Up     Normal  179.58 KB       16.67%  0
10.46.123.12    datacenter1 rack1       Down   Normal  315.21 KB       16.67%  28356863910078205288614550619314017621
10.46.123.13    datacenter1 rack1       Up     Normal  267.71 KB       16.67%  56713727820156410577229101238628035242
10.46.123.14    datacenter1 rack1       Up     Normal  315.21 KB       16.67%  85070591730234615865843651857942052863
10.46.123.15    datacenter1 rack1       Up     Normal  292.36 KB       16.67%  113427455640312821154458202477256070485
10.46.123.16    datacenter1 rack1       Up     Normal  300.02 KB       16.67%  141784319550391026443072753096570088106
```

2. Preparar al nodo de reemplazo instalando Cassandra y configurando correctamente el fichero de configuración cassandra.yaml.

3. Iniciar Cassandra en el nuevo nodo haciendo uso de la propiedad *-Dcassandra.replace_token=<token>* asignándole el mismo token que hasta ese momento usaba el nodo caído. En el caso de la figura de ejemplo sería de esta forma:

$ cassandra -Dcassandra.replace_token= 28356863910078205288614550619314017621

4. El nuevo nodo iniciará en estado hibernado y comenzará a obtener los datos pertenecientes a la partición que ocupa dentro del clúster. Durante ese tiempo este nodo no aceptará escrituras y será visto como caído por los otros nodo en el clúster. Cuando la carga se haya completado, el nodo será visto como activo por los otros miembros del anillo.

5. Una vez que el nuevo nodo está activo, es recomendable usar el comando nodetool repair sobre cada keyspace para asegurarse de que el nodo es completamente consistente. Por ejemplo:

$ nodetool repair -h 10.46.123.12 keyspace_name -pr

El clúster multinodos implementado como muestra demostrativa está compuesto por 3 nodos, cada uno de ellos tiene almacenada una porción del volumen total de registros de la base de datos donde como se puede observar en la Figura, el mayor peso de la carga lo tiene el host con la dirección ip 10.208.2.115 con un 66,12% de total de registros del sistema.

```
root@Produccion:/opt/cassandra/bin# /opt/cassandra/bin/nodetool -host 10.208.2.40 -p 7199 ring
Address         DC          Rack        Status State   Load            Owns    Token
                                                                                152280516958630826558320640298900719891
10.208.2.40     datacenter1 rack1       Up     Normal  1,17 MB         20,76%  174574042883143471896907989945626391 35
10.208.2.12     datacenter1 rack1       Up     Normal  1,13 MB         13,13%  39789933473527090949080423105380590785
10.208.2.115    datacenter1 rack1       Up     Normal  3,56 MB         66,12%  152280516958630826558320640298900719891
```

A continuación se muestra el listado de las conexiones activas, tanto entrantes como salientes de uno de los nodos en ejecución cuando el sistema Cassandra está funcionando en su totalidad para el clúster implementado.

```
root@Produccion:/opt/cassandra/bin# netstat -ant | grep 7000
tcp        0      0 10.208.2.40:7000        0.0.0.0:*               ESCUCHAR
tcp        0      0 10.208.2.40:7000        10.208.2.115:54879      ESTABLECIDO
tcp        0      0 10.208.2.40:33252       10.208.2.12:7000        ESTABLECIDO
tcp        0      0 10.208.2.40:42503       10.208.2.12:7000        ESTABLECIDO
tcp        0      0 10.208.2.40:39086       10.208.2.115:7000       ESTABLECIDO
tcp        0      0 10.208.2.40:7000        10.208.2.12:41212       ESTABLECIDO
tcp        0      0 10.208.2.40:42979       10.208.2.115:7000       ESTABLECIDO
```

//
http://web.archive.org/web/20120223143034/http://cassandra.apache.org/

Conclusiones

A partir de la implementación de la propuesta se logró un estudio exhaustivo sobre el impacto de la clusterización en bases de datos no relacionales resaltando las ventajas que brinda la implementación de un clúster en el sistema de bases de datos Cassandra con respecto a los sistemas de bases de datos relacionales, específicamente Postgres SQL.

Referencias

ABRAMOVA, V. and J. BERNARDINO. *NoSQL databases: MongoDB vs cassandra*. Proceedings of the international C* conference on computer science and software engineering, ACM, 2013. 14-22 p. 1450319769

CATTELL, R. Scalable SQL and NoSQL data stores *Acm Sigmod Record*, 2011, 39(4): 12-27.

CORNELIO, O. M. and J. G. GULÍN Modelo para la evaluación de habilidades profesionales en un Sistema de Laboratorios a Distancia *Revista Científica*, 2018, 3(33): 1.

HAN, J.; E. HAIHONG*, et al. Survey on NoSQL database*. 2011 6th international conference on pervasive computing and applications, IEEE, 2011. 363-366 p. 1457702088

HECHT, R. and S. JABLONSKI. *NoSQL evaluation: A use case oriented survey*. 2011 International Conference on Cloud and Service Computing, IEEE, 2011. 336-341 p. 1457716372

LEAVITT, N. Will NoSQL databases live up to their promise? *Computer*, 2010, 43(2): 12-14.

MAR-CORNELIO, O.; I. SANTANA-CHING*, et al.* Sistema de Laboratorios Remotos para la práctica de Ingeniería de Control *Revista Científica*, 2019, (36): 356-366.

MAR CORNELIO, O.; J. GULÍN GONZÁLEZ*, et al.* Sistema de Laboratorios a Distancia para la práctica de Control Automático *Revista Cubana de Ciencias Informáticas*, 2016, 10(4): 171-183.

MAR, O.; B. BRON*, et al.* Sistema para la auditoría y control de los Activos Fijos Tangibles. *Serie Científica de la Universidad de las Ciencias Informáticas*, 2016: 110-122.

MAR, O.; Y. Z. VÉLIZ*, et al.* Motor de inferencia decisional en sistema informático para la evaluación del desempeño *Revista Cubana de Ciencias Informáticas*, 2015, 9(4): 16-29.

MONIRUZZAMAN, A. and S. A. HOSSAIN Nosql database: New era of databases for big data analytics-classification,

characteristics and comparison *arXiv preprint arXiv:1307.0191*, 2013.

OKMAN, L.; N. GAL-OZ, *et al. Security issues in nosql databases*. 2011IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications, IEEE, 2011. 541-547 p. 145772135X

STRAUCH, C.; U.-L. S. SITES, *et al.* NoSQL databases *Lecture Notes, Stuttgart Media University*, 2011, 20.

TIWARI, S. *Professional NoSQL*. John Wiley & Sons, 2011. p. 1118167805

TUDORICA, B. G. and C. BUCUR. *A comparison between several NoSQL databases with comments and notes*. 2011 RoEduNet international conference 10th edition: Networking in education and research, IEEE, 2011. 1-5 p. 1457712350