Cassandra

- Cassanddra

Cassanddra

Terminología Cassandra

- Node: Equipo donde se almacenan los datos.
- Data Center: Una colección de nodos. Puede ser físico o virtual
- Cluster: Un cluster contiene uno o más DataCenter.
- Commit Log: Es un mecanismo de recuperación de fallos. Todos los datos se escriben primero en el registro de confirmación (archivo) para mayor durabilidad. Después de que todos sus datos se hayan enviado a SSTables, se pueden archivar, eliminar o reciclar.
- Table: Colección de columnas ordenadas por filas.
- SSTable: Una tabla de cadenas ordenadas (SSTable) es un archivo de datos inmutables en el que Cassandra escribe los archivos de forma periódica. Los SSTables se agregan solo y se almacenan en el disco secuencialmente y se mantienen para cada tabla de Cassandra.
- Gossip: Gossip es un protocolo de comunicación de igual a igual en el que los nodos intercambian periódicamente información de estado sobre ellos mismos y sobre otros nodos que conocen. El proceso de chismes se ejecuta cada segundo e intercambia mensajes de estado con hasta tres nodos más en el clúster.
- Bloom Filter: son algoritmos rápidos, no deterministas, para probar si un elemento es miembro de un conjunto. Se accede a los filtros Bloom después de cada consulta.

Como escribe Cassandra los datos

Los procesos de escritura de Cassandra siguen unos pasos muy estrictos:

1. Escritura de la operación de inserción de datos en el commit log.
2. Escritura de los datos en memtable
3. Flushing de los datos desde memtable.
4. Escritura en los discos en SSTable.

Configuraciones recomendadas

Máquina Virtual de JAVA

Sincronización de los relojes internos

Configuraciones TCP

Durante intervalos de poco tráfico, el servidor puede cerrar las conexiones tanto entre nodos locales como en nodos externos, esto lo hace por el tiempo de espera de conexión inactiva que esté configurado. Para solucionarlo:

```
sysctl -w \
  net.ipv4.tcp_keepalive_time=60 \
  net.ipv4.tcp_keepalive_probes=3 \
  net.ipv4.tcp_keepalive_intvl=10
```

Esta configuración mantiene viva la conexión durante 60 segundos con 3 intentos y con 10 segundos entre cada intento.

Si debemos manejar miles de conexiones simultáneas que usará la base de datos, debemos cambiar esta configuración en el kernel:

```
sysctl -w \
net.core.rmem_max=16777216 \
  net.core.wmem_max=16777216 \
  net.core.rmem_default=16777216 \
  net.core.wmem_default=16777216 \
  net.core.optmem_max=40960 \
  net.ipv4.tcp_rmem=4096 87380 16777216 \
  net.ipv4.tcp_wmem=4096 65536 16777216
```

Hay que asegurarse que los cambios persisten después de reiniciar

Desactivar CPU Frequency Scaling

Los sistemas linux incluyen una característica llamada CPU frequency scaling o CPU speed scaling. Esto permite a los servidores puedan funcionar a velocidades de reloj más baja cuando la demanda o carga sea baja.

Desafortunadamente, este comportamiento tiene un efecto perjudicial en los servidores que ejecutan Cassandra porque el rendimiento se puede limitar a un ritmo menor. Para garantizar un rendimiento óptimo, reconfiguramos todas las CPU para usar el regulador de rendimiento que bloquea la frecuencia al máximo. Este gobernador no cambiará las frecuencias, por lo tanto no habrá ahorro de energía y los servidores siempre se ejecutarán al máximo rendimiento, para configurar el gobernador:

```
for CPUFREQ in /sys/devices/system/cpu/cpu*/cpufreq/scaling_governor
do
  [ -f $CPUFREQ ] || continue
  echo -n performance > $CPUFREQ
done
```

Optimize SSD's

La mayoría de las configuraciones SSD en Linux no son óptimas.

1. Asegurate que el SysFs rotational flag está en false.
2. Aplicamos la misma marca de rotación para cualquier dispositivo de bloque creado desde el almacenamiento SSD, como mdarrays.
3. Establecer el scheduler IO en fecha límite o noop.
    1. El programador de noop es la opción correcta cuando el dispositivo de bloque de destino es una matriz de SSD detrás de un controlador IO de gama alta que realiza la optimización de IO.
    2. El programador de la fecha límite optimiza las solicitudes para minimizar la latencia de IO. En caso de duda, utilice el programador de plazos.
4. Establecer el readhead con valor de 8k.

Ejemplo en disco Sda

```
echo deadline > /sys/block/sda/queue/scheduler
#OR...
#echo noop > /sys/block/sda/queue/scheduler
touch /var/lock/subsys/local
echo 0 > /sys/class/block/sda/queue/rotational
echo 8 > /sys/class/block/sda/queue/read_ahead_kb
```

Use de optimum / RAID on SSD

Establecer el readhead con valor de 8k.

Disable zone_reclaim_mode

El kernel de Linux puede ser inconsistente en habilitar/deshabilitar la zone_reclaim_mode. Esto nos puede traer los siguientes problemas de rendimiento: A

- Enormes picos aleatorios de CPU.
- Programas colgados indefinidamente, aparentemente sin hacer nada.
- Los síntomas aparecen y desaparecen de repente.
- Después de un reinicio los síntomas generalmente no se muestran de nuevo durante un tiempo.

```
Desactivar la zone_reclaim_mode:
echo 0 > /proc/sys/vm/zone_reclaim_mode
```

Set user resource limits

Aunque los límites pueden ser temporales DataStax recomienda hacer los siguientes cambios permanentes: A

```
Package and Installer-Services:
/etc/security/limits.d/cassandra.conf
<cassandra_user> - memlock unlimited
        <cassandra_user> - nofile 100000
        <cassandra_user> - nproc 32768
        <cassandra_user> - as unlimited
```

```
Tarball and Installer-No Services:
  <cassandra_user> - memlock unlimited
        <cassandra_user> - nofile 100000
        <cassandra_user> - nproc 32768
        <cassandra_user> - as unlimited
```

Run DataStax como root? Algunas distribuciones como ubuntu necesitan los pasar los límites que establecimos para el usuario cassandra a root.

```
        root - memlock unlimited
        root - nofile 100000
        root - nproc 32768
        root - as unlimited
```

Para RHEL 6.x-based systems, establecer también el nrpoc limits:

```
/etc/security/limits.d/90-nproc.conf
cassandra_user - nproc 32768
```

Para todas las instalaciones:

```
cassandra_user - nproc 32768
vm.max_map_count = 1048575
```

Para instalaciones de Debian o Ubuntu el módulo pam_limits.so no esta activado por defecto:

```
/etc/pam.d/su
session    required   pam_limits.so
sysctl -p para guardar los cambiar y activarlos. Después reiniciar el sistema para comprobar que los cambios sigue
```

Disable Swap

Si no se desactiva el intercambio por completo, se puede ver reducido considerablemente el rendimiento. Debido a que la bd tiene múltiples réplicas y conmutación por error transparente, es preferible que un réplica se elimine inmediatamente cuando la memoria está baja en lugar de ir a la Swap. Esto permite que el tráfico se redirija inmediatamente a una réplica en funcionamiento en lugar de continuar intentado acceder a la réplica que tiene una latencia alta debido a encontrarse en la memoria swap. Mientras más memoria Swap mayor reducción de rendimiento.

```
swapoff -all (Cambio no permanente).
Eliminar partición swap del /etc/fstab (Cambio permanente).
```

Esta opción no nos convence mucho ya que dejaríamos al sistema sin memoria Swap. Para que Cassandra no utilice tanto o demasiado la memoria swap vamos a reducir el porcentaje swapinnes, diciéndole al sistema que utilice menos memoria Swap.

Para ver la configuración actual:

```
cat /proc/sys/vm/swappiness
```

Para cambiar:

```
/etc/sysctl.conf
vm.swapinness = 10
```

Check Java Hugespaces setting

Cuando linux usa Hugepages, el núcleo intenta asignar memoria en grandes porciones (2MB en lugar de 4K). Esto puede mejorar el rendimiento al reducir el ńumero de páginas que la CPU debe asignar. Sin embargo algunas aplicaciones todavía asignan memoria basada en 4K y esto puede causar problemas de rendimiento cuando linux intenta desfragmentar páginas de 2MB y sean de 4K. A

Para solucionarlo desactivamos el hugepages transparent.

```
echo never | sudo tee /sys/kernel/mm/transparent_hugepage/defrag
```

Comandos útiles

Para encontrar el pid:

```
ps axu | grep cassandra
```

Para ver los puertos de escucha:

```
netstat -tulpn | grep -i listen | grep <pid>
```

Para conectar con la consola de cassandra

```
cqlsh ip_del_servidor -u cassandra -p cassandra
```

Insertar Datos para pruebas

```
cqlsh ip_nodo -u cassandra -p cassandra < file_inserts;
```

Refrescar una tabla

```
nodetool ip_nodo refresh adtech geosc
```

Rebuild (para replicar datacenters dentro de datacenters)

```
nodetool rebuild -- dc1 o dc2, o el que sea
nodetool abortrebuild // para abortar la sincro entre data centers.
```

Estado de la tabla, desglose de campos del comando.

```
nodetool tablestats adtech.geosc
```

Obtener snapshot de todas las colecciones.

```
nodetool snapshot.
```

Limpiar los snapshot de todas las colecciones.

```
nodetool clearsnapshot
```

Simple flush memtables en disco.

```
nodetool flush
```

Escribe todas las memtables de un nodo con SSTables en disco.

```
nodetool drain
Cassandra para de escuchar más conexiones de los clientes y otros nodos. Es necesario reiniciar Cassandra despu
```

Estado de cassandra utilizando la red

```
  nodetool netstats
```

Estado del backup

```
nodetool statusbackup
```

Ver estado de las particiones y del cluster

```
nodetool describing
nodetool describecluster
```

Info de un nodo

```
nodetool status
De aquÍ podemos sacar el identificador que por ejemplo necesitamos en el siguiente comando.
```

Quitar un nodo

```
nodetool removenode ID
```

Export

Export schemas

Para conocer los esquemas instalados

```
SELECT keyspace_name, table_name from system_schema.tables;
SELECT * FROM system_schema.tables;
```

Para exportar todo los schemas:

```
cqlsh ip server usuario -p passwd -e "DESCRIBE SCHEMA" > /your/path/schema.cql
```

Para importar con el archivo de exportación antes obtenido

```
cqlsh ip server -u cassandra -p cassandra -f schema_adtech.cql
```

Export data

Para exportar datos Cassandra es capaz de crear un snapshot de los datos actualizados en caliente.

El comando:

```
nodetool snapshop
```

Este comando genera una snapshot por cada colección que haya sido creada en Cassandra. Nos devuelve un número que es la carpeta donde está todo el contenido a exportar.

```
/var/lib/data/adtech/users-213434513042342/snapshot/155489393/
```

Dentro de la última carpeta tenemos los misma db que en:

```
/var/lib/data/adtech/users-213434513042342/
```

Para borrar los snapshot y evitar la carga de disco del servidor de producción:

```
nodetool clearsnapshop
```

Importante: Sin el schema los datos no se verán en ningún sitio, es importante tener el backup del schema importado.

Una vez ya recuperado el schema veremos que tenemos las mismas carpetas que en el servidor a exportar, solo nos faltará descargar los datos del Servidor de backup y descomprimirlos en la carpeta correspondiente.

Para poder apreciar los datos debemos lanzar el comando:

```
nodetools refresh <basedatos> <tabla>
```

Import

Como ya tenemos preparado un snapshot diario (guardamos una semana) de cada nodo de cassandra, en caso de querer recuperar una tabla de la colección adtech, la única que se generó y se está guardando, solo deberemos acceder a Store y obtener la copia de los datos que ya tenemos. Para recuperar los datos debemos seguir los siguientes pasos:

1. Para importar datos de un nodo debemos apagar el nodo donde queremos importar los datos

```
systemctl stop cassandra
```

1. Una vez apagado, debemos eliminar los datos anteriores:

```
Eliminar datos de:
/var/lib/cassandra/data/adtech/geosc-numeroID/*.db
```

1. Copiamos el interior del snapshot dentro de:

```
/var/lib/cassandra/data/adtech/geosc-numeroID/
```

1. Encendemos cassandra y refrescamos la tabla.

```
systemctl start cassandra
nodetool refresh adtech.geosc
```

1. Reiniciamos cassandra

```
systemctl restart cassandra
```

Para comprobar que todo ha ido bien:

```
Contamos las celdas donde cogimos el backup, realizamos estas sentencias tanto en el nodo donde exportamos lo
cqlsh nodo -u cassandra -p cassandra n
SELECT COUNT(*) FROM adtech.geosc;
```

Comando nodetool tablestats adtech geosc para obtener información de las tablas.

Eliminar un nodo

1. Asegurarse de que el nodo esta UP/DOWN usando nodetool status (UN=up, DN=down):
2. Si el nodo esta UP, ejecutar nodetool decommission (en el nodo que se va a dar de baja)

```
Nota: El decommision no para el servicio de cassandra, hace falta pararlo una vez haya terminado. El proceso tard
```

1. Si el nodo está DOWN, ejecutar nodetool removenode <Host ID> (desde cualquiera de los demás nodos).
    1. Nota: el proceso tarda, según el tamaño del cluster de cassandra (ejecutar en un screen).
    2. Mucho ojo con el espacio en disco
2. Si el removenode falla, ejecutar nodetool assassinate <IP Address> (desde cualquiera de los demás nodos).

Añadir un DataCenter

- Con el “Cassandra” parado, “limpiar” posibles directorios de datos anteriores.

```
rm -rf /var/lib/cassandra/*  <- OJO! No eliminar el directorio, únicamente el contenido.
```

- Mientras se está “añadiendo” un nuevo Datacenter… es aconsejable que el cliente (los php’s del cliente), utilicen CONSISTENCY LOCAL_ONE, ó, LOCAL_QUORUM

- Asegurarse que en Datacenter que está “UP”, está como NetworkTopologyStrategy… sinó es así, aplicar el siguiente “ALTER”:

```
ALTER KEYSPACE "sample_ks" WITH REPLICATION =  { 'class' : 'NetworkTopologyStrategy', 'ExistingDC' : 3 };
```

Revisar ficheros de configuración (añadir nombres, ips, etc…):

```
/etc/cassandra/conf/cassandra.yaml
```

- Añadir en el fichero “cassandra.yaml” una “seed” (la IP) en cada “Datacenter”. Ej. seeds: "217.13.124.41,94.24.114.104" (1 nodo del Datacenter1 y otro nodo del Datacenter 2)
- Arrancar cada uno de los nuevos nodos del “nuevo Datacenter”
- Añadir el nuevo Datacenter en los “keyspaces” del “antiguo Datacenter”. (hacerlo solo en 1 nodo).

```
ALTER KEYSPACE "sample_ks" WITH REPLICATION =    {'class': 'NetworkTopologyStrategy', 'ExistingDC':3, 'NewDC
Ej:
ALTER KEYSPACE system_distributed WITH REPLICATION = { 'class' : 'NetworkTopologyStrategy', 'dc1' : 3, 'dc2' : 3
ALTER KEYSPACE system_traces WITH REPLICATION = { 'class' : 'NetworkTopologyStrategy', 'dc1' : 3, 'dc2' : 3 };
ALTER KEYSPACE adtech WITH REPLICATION = { 'class' : 'NetworkTopologyStrategy', 'dc1' : 3, 'dc2' : 3 };
ALTER KEYSPACE music WITH REPLICATION = { 'class' : 'NetworkTopologyStrategy', 'dc1' : 3, 'dc2' : 3 };
Opcional (usuarios "globales"?):
ALTER KEYSPACE system_auth WITH REPLICATION = { 'class' : 'NetworkTopologyStrategy', 'dc1' : 3, 'dc2' : 3 };
```

- Comprobar, desde consola del Cassandra que todo está OK:

```
SELECT * FROM system_schema.keyspaces;
```

- Hacer “sincronización inicial” en CADA UNO de los NODOS NUEVOS:

```
nodetool rebuild -- dc1
```

Añadir un Nodo

Tenemos varias variables a tener en cuenta a la hora de añadir un nodo a cassandra.

- Auto_bootstrap. Variable que solo hace falta en el primer arranque de cassandra. Importante borrar esta opción después de la primera arrancada.
- cluster_name. Nombre del cluster donde la vamos a añadir
- listen_address / broadcast_address. IP de escucha / Ip de comunicación con otros nodos.
- seed_provider. Sacar el nodo que queremos añadir de las semillas.

```
Importante: Estaría bien tener una copia de la configuración antes de levantar nada. Ya que allí vemos de forma rá
```

```
Importante: Mucho ojo con esta conf: cassandra-topology.properties
```