Cassandra 集群核心配置和概念梳理

无锋剑客2019-07-10 14:05:37

https://blog.51cto.com/michaelkang/2418916

Cassandra是一款分布式的结构化数据存储方案(NoSql数据库)，存储结构比Key-Value数据库（像Redis）更丰富，但是比Document数据库（如Mongodb）
适合做数据分析或数据仓库这类需要迅速查找且数据量大的应用.

Cassandra 集群特性比较丰富，考虑场景也比较多，如果想用好集群，集群本很多概念都要能够了解，下面对相关概念进行简介；

与关系数据库相关概念：

1. keyspace -> table -> column，对应关系型数据库 database -> table -> column

集群主要配置：

1. cluster_name:集群名，同一集群的多个节点，集群名要一致；
2. seeds: 种子节点，集群中的全部机器的ip，以逗号隔开；
3. storage_port: Cassandra服务器与服务器之间连接的端口号，一般不需要修改，但要保证此端口上没有防火墙；
4. listen_address: Cassandra集群中服务器与服务器之间相互通信的地址。如果留空，将默认使用服务器的机器名；
5. native_transport_port: 默认的CQL本地服务端口，本地的cql客户端与服务器交互的端口；

默认Cassandra使用端口

1. 7000作为集群通信端口（如果开启了SSL就是7001端口）。
2. 9042端口用于native协议的客户端连接（如cqlsh）。
3. 7199端口用于JMX，
4. 9160端口用于废弃的Thrift接口

集群主要配置文件目录

1. data_file_directories: 数据文件存放的目录，一个或多个
2. commitlog_directory: 提交信息的日志文件存放的目录
3. saved_caches_directory: 缓存存放的目录

集群相关概念：

Data Center

1. 有多个Rack组成的逻辑集合。比如同一栋楼里互相连接的机器

Rack

1. 一个逻辑集合，有多个彼此临近node的组成。比如一个机架上的所有物理机器。

Gossip and Failure Detection

1. Gossip是一种p2p协议,用于failure detection,跟踪其他节点的状态，每秒运行一次。
2. 运用Phi Accrual Failure Detection实现failure detection
3. 计算出一个结果level of suspicion,表示节点失败的可能性。
4. 具有灵活性,同时也避免了传统heartbeat的不可靠。应为可能只是短暂的网络拥塞,尤其是在公有云上。

Snitches

1. snitch定义了集群中每个节点相对其他节点的邻近度，以此来确定从哪个节点读取和写入。
2. 一般采用手动定义的模式,在cassandra.yaml配置为endpoint_snitch: GossipingPropertyFileSnitch
3. 同时在cassandra-rackdc.properties配置当前节点的dc和rack，比如

Rings and Tokens

Cassandra表示由集群管理的数据作为一个环。环中的每个节点被分配一个或多个由token描述的数据范围，确定在环中的位置。
token是用于标识每个分区的64位整数ID,范围是-2^63 -- 2^63-1
通过hash算法计算partition key的hash值，以此确定存放在哪个节点

Virtual Nodes

virtual node的概念简称vnode,原先的token范围被缩减为多个更小的token范围。
每个节点包含多个token范围。默认每个节点产生256个token范围（通过num_tokens调节），也就是256个vnode。在2.0以后默认开启。
在性能差的节点上,可以适当减少num_tokens的值。

Partitioners

partitioners决定数据存放在哪个vnode上。它是一个hash函数，计算每行的partition key的hash值。
代码在org.apache.cassandra.dht包里,目前主要用Murmur3Partitioner、DHT即为distributed hash table。

Replication Strategies

第一份复制存在对应的vnode中。其他复制的存放位置由replica strategy(或叫replica placement strategy)决定
主要有两种策略：
SimpleStrategy
将副本放置在环上的连续节点处，从分区器指示的节点开始。
NetworkTopologyStrategy
允许为每个数据中心指定不同的复制因子。在数据中心内，它将副本分配给不同的rack，以最大限度地提高可用性

Consistency Levels

根据CAP理论,一致性，可用性和分区容忍性不可兼得，cassandra通过设置读写时最少响应节点的数量，实现了可调的一致性。
可选的一致性级别：ANY, ONE, TWO,THREE, QUORUM, ALL，其中QUORUM,ALL是强一致性。强一致性公式：R+W>N R:读复制数， W:写复制数，N:复制因子

Queries and Coordinator Nodes

可以连接任一节点来执行读写操作，被连接的节点叫做Coordinator Nodes,需要处理读写一致性。比如：写到多个节点，从多个节点读取

写操作执行过程

当执行一个写操作时，数据被直接写到commit log文件,并将设置commit log中的dirty flag为1。
然后将数据写到内存memtable,每个memtable对应一个表，当memtable的大小达到一个限值后会被写入磁盘SSTable,
然后将commit log中的dirty flag设为0

Caching

有三种cache：
key cache
缓存partiton keys到row index entries的映射，存在jvm heap
row cache
缓存常用的row,存在off heap
counter cache
提升counter性能

Hinted Hando

一种写入高可用特性，当写入请求发给coordinator是，replica节点可能因为种种原因不可用(网络、硬件等)，
此时coordinator会临时保存写请求,等到replica节点重新上线时再写入。默认保留两个小时

Tombstones

SStables文件是不可修改的。删除数据被当做一个update,会被更新为tombstone。在compact运行之前，它可以抑制原来的值。
设置中：Garbage Collection Grace Seconds(GCGraceSeconds),默认864,000,10天。

会清理超过这个时间的tombstones。当节点不可用时间超过这个这个时间，会被替换

Bloom Filters

是个快速的、非确定性算法，用于确定测试元素是否在集合中。以此降低不必要的磁盘读取。
可能得到一个false-positive结果。通过将数据集映射到bit array上,一种特殊的缓存。

Compaction

SSTables是不可变的，通过compaction。重新生成一个新的SSTable文件(此文件中不包含不需要的数据，比如被删除的数据)
三种策略：
SizeTieredCompactionStrategy (STCS)
默认的策略，写密集型
LeveledCompactionStrategy (LCS)
读密集型
DateTieredCompactionStrategy (DTCS)
用于基于时间或日期的数据

Anti-Entropy, Repair

assandra使用Anti-Entropy协议，这是一种用于修复复制集数据的gossip协议
有两种情况
read repair
读取时发现有不是最新的数据。此时开始修复
Anti-Entropy repair
通过nodetool手动运行修复

Merkle Trees

Merkle Trees来源于Ralph Merkle,也叫做hash tree,是一种二叉树。每个父节点是它直接子节点的hash值，用于减少网络I/O。

Staged Event-Driven Architecture (SEDA)

cassandra采用分阶段事件驱动架构,SEDA: An Architecture for Well-Conditioned, Scalable Internet Services
一个stage由事件队列、事件处理器和线程池组成
controller决定stage的调度和线程申请。主要代码在org.apache.cassandra.concurrent.StageManager
以下操作都是作为stage来执行的
Read (local reads)
Mutation (local writes)
Gossip
Request/response (interactions with other nodes)
Anti-entropy (nodetool repair)
Read repair
Migration (making schema changes)
Hinted handoff

System Keyspaces

system_traces
system_schema
keyspaces
tables
columns
存储kespace,table,column的定义
- materialized_views
存储可用的view
- functions
  用户定义函数
- types
  用户自定义类型
- triggers
  每个表的触发配置
- aggregates
  聚合定义

```
system_auth
system
local
peers
存储节点信息
available_ranges
range_xfers
存储token范围
materialized_ views_builds_in_progres
built_materialized_views
跟踪view的构建
paxos
存储paxos状态
batchlog
存储 atomic batch操作的状态
size_estimates
存储每个表的分区的估计数量,用于hadoop集成
```

参考文档:

https://www.2cto.com/database/201802/717564.html

https://blog.csdn.net/zhuwinmin/article/details/76066642

https://segmentfault.com/a/1190000015610357