# MongoDb PK Cassandra

## 1 对比基准：

1.1 存储空间

1.2 运维

1.3 Object Model

1.4 Secondary Indexes

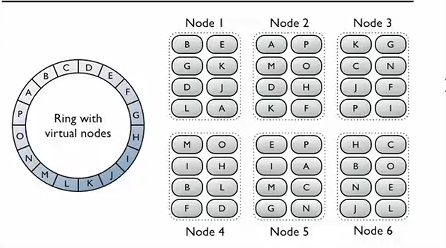
1.5 Scalability

1.6 High avialability

1.7 Query Language

## 2 Cassandra 关键概念

Cassandra 采用的是一致性hash 算法把节点映射到hash 环上 每个节点分别负责Hash环中的一部分数据，这种模式和Master/slave redis/Mongodb的分片集群有实质不同



采用无中心化设计，各个节点的地位都一样，每个节点都同步集群中的配置信息 通信协议使用Gossip 连接到其中一台节点，即可获取集群的所有配置

### 2.1数据一致性

CommitLog : 记录客户端提交过来的记录及操作，数据记录持久化到磁盘，方便没有写入磁盘的数据恢复数据，持久化完成的CommitLog 会被自动清楚

Memtable : 数据在CommitLog 记录后会被写入Memtable(内存中)

SSTable : 当内存中的数据达到设置的值会被刷新到磁盘 持久化到SSTable 中,数据是追加到文件末尾，一个列族一个SSTable 文件

数据一致性: 通过设置当多少个副本节点写入成功后通知客户端执行成功，多个副本读取最新数据 （Cassandra 是个无中心设计 ，数据一致性是通过用户在客户端设置）

### 2.2 数据读写原理

写:Cassandra写数据,首先会将请求写入Commit Log以确保数据不会丢失，然后再写入内存中的Memtable，超过内存容量后再将内存中的数据刷到磁盘的SSTable，并定期异步对SSTable做数据合并(Compaction)以减少数据读取时的查询时间。因为写入操作只涉及到顺序写入和内存操作，因此有非常高的写入性能

读: 1 先从RowCache 中判断是否有数据, 有直接返回

2.1 从内存Memtable 获取数据

2.2 通过Bloom Filter 算法判断key 存在那个节点那个磁盘位置SSTable。查询对应的SSTable 的KeyCache 中是否存在当前key 取到数据

3 把MemTable 内存数据+SSTable 数据整合返回client

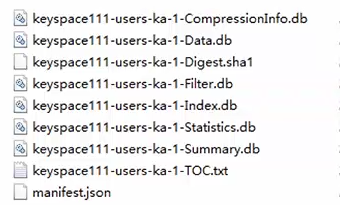
注释：

RowCache 缓存最近作为热点数据

KeyCaChe 中缓存了最近查询的row key在SSTable中的位置 SSTable 有个专门的Index.db 存储当前SSTable 的索引信息

### 2.3 物理存储

SSTa le 数据存储在磁盘上的文件 不同的文件在读写起着重要作用



\*-Data.db 是SSTable 的 数据文件

\*-Index.db 索引文件 保存Rowkey 在Data 的偏移文件

\*-Filter.db 是采用Bloom Filter 生成的Rowkey 映射文件 快速定位rowkey 在那个SSTable

\*-CompressionInfo.db 数据压缩存储的信息

\*-Summary.db 记录了Index.db中 索引信息的范围

\*-Statistics.db 记录了整个SSTable 的全部信息

### 2.4 分区器，分区策略

分区器用来计算Partition key 的hash 值 决定数据如何在集群内数据的分发和复制

Murmur3Partitioner 默认比RandomPartitioner 更快更好

RandomPartitioner 基于Md5 Hash 算法

ByteOrderPartitioner 根据Partitioner key 的Bytes 保证分区有序 但是容易造成数据严重偏移，负载不均衡，热点问题

SimpleStrategy 单一数据中心设置

NetWorkStrategy 多数据中心设置

### 2.5 Snitch 探测器

探测器决定从哪个数据中心和机架读取写入数据 检测网络拓扑

Dynamic Snitching 自动监控读请求 发现延迟转发给性能好的节点 默认开启

Simple Snitching 不考虑数据中心机架

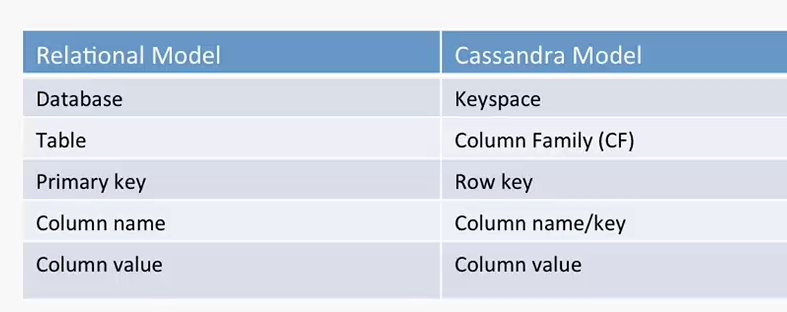
GossipingPropertyFileSnitching 多数据中心配置 生产环境设置

### 2.6 二级索引

Cassandra 实现模式：在某一列上创建索引 在存储上重新创建一个列族 rowkey 为列的值，列族下的列为原先的rowkey 通过得到原先行的rowkey 定位数据

### 2.7 Cql Cassandra Query Language

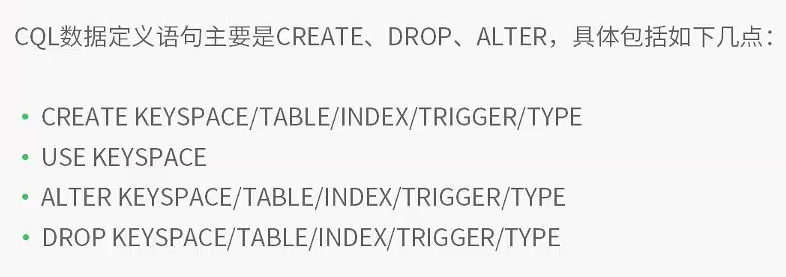
支持Mysql 语法 相识度达到90%



支持的数据类型



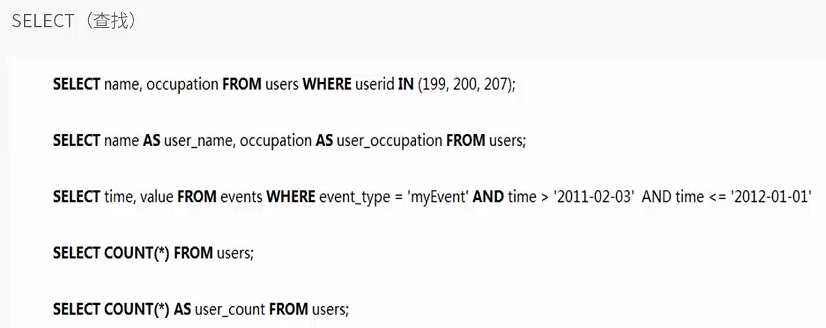
**支持List Map**



Insert into table() values ()

Delete /update ........

Select column from table/column family where (where-clause) order by( column asc/desc ) limit (allowing filtering)



Select 查询限制：

唯一可用的函数Count

可以使用limit 限制条数

出现在Where 条件后面的列名必须是Primary Key 指定的第一个列名 或者使用Create Index 建立了索引的列名 如果不家里索引 可以在结尾加上Allow Filtering

Primary key 指定的第一个列名 也叫做Parititon Key 对数据在节点分布有重要作用

不支持Join 子查询

### 2. 8 Cassandra 总结

1 面向列的多为hash 稀疏表

2 Cassandra分布式存储，无中心节点，避免单点失败，可扩展性良好

3 暂无多列索引，需逐列分别建立索引

4 动态添加删除节点对集群IO 流量 不会造成 影响 ，只会对添加删除节点附近的节点数据造成影响（Hash 环）。不会因为数据重新分区发生数据迁移 cassandra数据移动是个缓慢的过程

5 自动故障转移 不需要特殊情况时加班熬夜修复，本身自动修复

6 低运维成本，数据增长只需要添加机器，线性扩展方便。

7 写入性能大约读性能 有可以在任何地方任何时间集中读或写任何数据。并且不会有任何单点失败

8 Cql 查询语句 不支持聚合，类似Mysql 查询语句 但是在查询中有一定的限制

## 3 MongoDb 总结

1面向集合存储，容易存储对象类型的数据。

2 支持查询。MongoDB 支持丰富的查询操作，MongoDB 几乎支持SQL中的大部分查询

3强大的聚合工具。MongoDB 除了提供丰富的查询功能外，还提供强大的聚合工具，如count、group 等，支持使用MapReduce 完成复杂的聚合任务

4 强一致性,所以您的应用程序可以立即看到更新你的数据

5 支持完全索引，可以在任意属性上建立索引，包含内部对象。MongoDB的索引和RDBMS 的索引基本一样，可以在指定属性、内部对象上创建索引以提高查询的速度。

6 文件存储格式为BSON（JSON 的一种扩展）。BSON 是对二进制格式的JSON 的简称，BSON 支持文档和数组的嵌套。

7 自动处理分片。MongoDB 支持集群自动切分数据，对数据进行分片可以使集群存储更多的数据，实现更大的负载，也能保证存储的负载均衡

## 4 基准 PK

### 4.1存储空间

|  |  |
| --- | --- |
| MongoDB | Cassandra(设计模型类似Hbase 列/列族/时间戳) |
| 1 实际存储的事Binary Json 对象  2 面向集合存储，容易存储对象  3 集合类似RDBMS 中的表，一个集合中可以存储无限多的文档 | 1 存储数据为 Byte[]  2 逻辑试图为多维的稀疏表(每一行可以有不同的结构)  3 物理模型按照RowKey排序的 顺序存储在磁盘(不是随机写 顺序写入) |

总结： 写入相同的数据( Gps Location 表) Cassandra 存储空间少，机器内存占用相比占部分优势

**注释：之前给出的Mongo 测试文档和Cassandra 测试文档 测试环境不一样**

**Mongo 测试是在线下 16G 内存 8Core 实体机上测试**

**Cassandra 测试环境是在 8G内存 4Core 虚拟机上测试 这点需要注意**

### 4.2 运维

4.2.1 Cassandra

Cassandra 只要设计好Data model 使用DataStarX OpsCenter 可以动态添加机器删除机器，后台数据repair 相关操作 无需专门运维人员去维护。

最坏情况：数据某一节点宕机 Cassandra 会把写请求暂时转发给其他节点 节点恢复自动进行数据的迁移（Hinted off 特性）运维代价30%

4.2.1 Mongo

MongoDb 需要专门精通的维护 监控机器内存，节点故障，同时还要负责Collections 的一些设计或者调优 MongoDb 相比Casssandra 存在更多的内存碎片 运维代价 70%

最坏情况: 节点宕机 导致同一分片的master-slave 宕机，无法写入读取，节点恢复过程会有一段重新数据分发IO 流量高峰期

### 4.2.1 Object Model

Mongo : 基于Document 型 一个集合里面可以存放大量不同结构的数据 （类似于Cassandra 数据模型中的row）

Cassandra :1 是基于列存储 按照rowkey 排序的面向行的带索引的存储。一组包含名称值对的数据叫做行（Row），而每一组名称值对（Name/Value Pair）被称之为列（Column）

2 无结构的可以动态往表添加数据 不必考虑之前的设计模式(类似Mongo Doc 不同的结构)

3 有列族 列 超级列族 复合键 概念

**总结：需要一个丰富的数据模型,那么MongoDB更适合你，Cassandra 也可以**

### 4.2.2 Secondary Indexes

Mongo: 1 实现B+Tree 实现 每个集合最多可以创建64个索引。它支持能在RDBMS中找到的各种索引，升序、降序、唯一性、复合键索引

2支持复合索引

3 支持完全索引，包含内部对象

Cassandra : 1 索引实现是内部增加一个列族 由row- column 模式 变成 column -row

2 不支持复合索引 多个索引只能单独通过create index on table() 创建

**总结：如果应用程序需要二级索引,需要灵活的查询模型 MongoDB更适合你。**

### 4.2.3 Scalability

Mongo : 添加一个节点 需要添加一个replicate set 还需手动添加一个shared 并且move 相关数据

Cassandra : 只需添加一个节点，自动加入已经存在的集群，加入Ring 接受数据

**总结： 都可以接受一个看运维一个啥都不用干 ，如果要求写扩展性 ，推荐Cassandra**

### 4.2.4 High avialability

Mongo : 集群share 模式 每个分片存在一个 Master Slave Aribter 不存在单点问题 ，Master 宕机 Slave 会接管

Cassandra: 无中心，每个节点角色都一样。

**总结 ：看喜好**

### 4.2.5 Query Language

Mongo : 文档片段查询

Cassandra :提供了类似Sql 语言查询

总结：希望Sql 查询 Cassandra 更好选择

## 5 国外机构测试

**官方给的测试的对比：**

**http://www.datastax.com/nosql-databases/benchmarks-cassandra-vs-mongodb-vs-Hbase**

**https://academy.datastax.com/planet-cassandra//mongodb-to-cassandra-migration/**

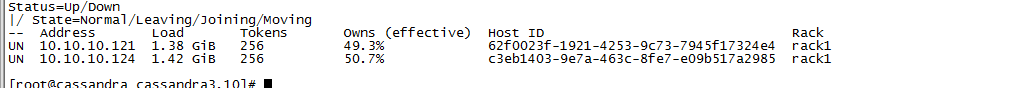
## 6 具体业务思考

Mysql T\_location 表 有两个索引 一个是F\_GPS\_TIME F\_VEHICLE\_ID

### 6.1 存储量比较

|  |  |  |
| --- | --- | --- |
|  | 500W | 5000W |
| Mongodb | 500M | 4.5G |
| Cassandra | 300M | 2.4G |





### 6.2 业务查询复杂度

MongoDb 满足多车 跨时间段查询 多条件查询

Cassandra 需要建立合适的Partition key Cluster Key 索引 费劲 有些功能还实现不了