ClickHouse原理

1. ClickHouse的优点：

从用户角度来说，ClickHouse就是实现了“多快好省，独立”。

- 快：提供了极致的查询性能
- 多：支持分布式集群模式，支持高吞吐写入能力
- 省：以极低的成本存储海量数据
- 好：提供完善SQL支持，上手十分简单；提供json、map、array等灵活数据类型适配业务快速变化；同时支持近似计算、概率数据结构等应对海量数据处理。
- 独立：独立于Hadoop技术栈

- **完备的管理功能**
  - DDL（数据定义语言）：可以动态地创建、修改或删除数据库、表和视图，而无须重启服务。
  - DML（数据操作语言）：可以动态查询、插入、修改或删除数据。
  - 权限控制：可以按照用户粒度设置数据库或者表的操作权限，保障数据的安全性。
  - 数据备份与恢复：提供了数据备份导出与导入恢复机制，满足生产环境的要求。
  - 分布式管理：提供集群模式，能够自动管理多个数据库节点。

- **列式存储与数据压缩**

  列式存储和数据压缩能够减少数据扫描范围和数据传输时的大小，让查询变得更快。按列存储与按行存储相比，可以有效减少查询时所需扫描的数据量。数据中的重复项越多，则压缩率越高；压缩率越高，则数据体量越小；而数据体量越小，则数据在网络中的传输越快，对网络带宽和磁盘IO的压力也就越小。

既然如此，那怎样的数据最可能具备重复的特性呢？答案是属于同一个列字段的数据，因为它们拥有相同的数据类型和现实语义，重复项的可能性自然就更高。

ClickHouse就是一款使用列式存储的数据库，数据按列进行组织，属于同一列的数据会被保存在一起，列与列之间也会由不同的文件分别保存（这里主要指MergeTree表引擎）。数据默认使用LZ4算法压缩，在Yandex.Metrica的生产环境中，数据总体的压缩比可以达到8:1（未压缩前17PB，压缩后2PB）。列式存储除了降低IO和存储的压力之外，还为向量化执行做好了铺垫。

为什么要使用压缩数据块?

ClickHouse服务为了节省磁盘空间，会使用高性能压缩算法对存储的数据进行压缩。默认启用的是lz4（lz4 fast compression）压缩算法。

ClickHouse是列存的数据，每一列的数据都会单独存放到bin文件中。这些.bin文件，最终承载着数据的物理存储。

压缩规则及流程

MergeTree在数据具体的写入过程中，会依照索引粒度（默认情况下，每次取8192行），按批次获取数据并进行处理。如果把一批数据的未压缩大小设为size，则整个写入过程遵循以下规则：

- 单个批次数据size<64KB ： 如果单个批次数据小于64KB，则继续获取下一批数据，直至累积到size>=64KB时，生成下一个压缩数据块。
- 单个批次数据64KB<=size<=1MB ： 如果单个批次数据大小恰好在64KB与1MB之间，则直接生成下一个压缩数据块。
- 单个批次数据size>1MB ： 如果单个批次数据直接超过1MB，则首先按照1MB大小截断并生成下一个压缩数据块。剩余数据继续依照上述规则执行。

此时，会出现一个批次数据生成多个压缩数据块的情况

通过上面的信息，我们可以知道一个.bin文件是由1至多个压缩数据块组成的，每个压缩块大小在64KB～1MB之间。多个压缩数据块之间，按照写入顺序首尾相接，紧密地排列在一起。

虽然数据被压缩后能够有效减少数据大小，降低存储空间并加速数据传输效率，但数据的压缩和解压动作，其本身也会带来额外的性能损耗。所以需要控制被压缩数据的大小，以求在性能损耗和压缩率之间寻求一种平衡。

> 在具体读取某一列数据时（.bin文件），首先需要将压缩数据加载到内存并解压，这样才能进行后续的数据处理。通过压缩数据块，可以在不读取整个.bin文件的情况下将读取粒度降低到压缩数据块级别，从而进一步缩小数据读取的范围。

- **向量化执行引擎**

  向量化执行，可以简单地看作一项消除程序中循环的优化。这里用一个形象的例子比喻。为了制作n杯果汁，非向量化执行的方式是用1台榨汁机重复循环制作n次，而向量化执行的方式是用n台榨汁机只执行1次。

- **关系模型与SQL查询**

  ClickHouse完全使用SQL作为查询语言（支持GROUP BY、ORDER BY、JOIN、IN等大部分标准SQL）。在SQL解析方面，ClickHouse是大小写敏感的，这意味着SELECT a 和 SELECT A所代表的语义是不同的。

  关系模型相比文档和键值对等其他模型，拥有更好的描述能力，也能够更加清晰地表述实体间的关系。ClickHouse使用了关系模型，所以将构建在传统关系型数据库或数据仓库之上的系统迁移到ClickHouse的成本会变得更低，可以直接沿用之前的经验成果。

- **多样化的表引擎**

  因为Yandex.Metrica的最初架构是基于MySQL实现的，所以在ClickHouse的设计中，能够察觉到一些MySQL的影子，表引擎的设计就是其中之一。与MySQL类似，ClickHouse也将存储部分进行了抽象，把存储引擎作为一层独立的接口。

- **多线程与分布式**

  ClickHouse在数据存取方面，既支持分区（纵向扩展，利用多线程原理），也支持分片（横向扩展，利用分布式原理），可以说是将多线程和分布式的技术应用到了极致。

- **多主架构**

  HDFS、Spark、HBase和Elasticsearch这类分布式系统，都采用了Master-Slave主从架构，由一个管控节点作为Leader统筹全局。而ClickHouse则采用Multi-Master多主架构，集群中的每个节点角色对等，客户端访问任意一个节点都能得到相同的效果。

- **在线查询**

  ClickHouse经常会被拿来与其他的分析型数据库作对比，比如Vertica、SparkSQL、Hive和Elasticsearch等，它与这些数据库确实存在许多相似之处。例如，它们都可以支撑海量数据的查询场景，都拥有分布式架构，都支持列存、数据分片、计算下推等特性。这其实也侧面说明了ClickHouse在设计上确实吸取了各路奇技淫巧。与其他数据库相比，ClickHouse也拥有明显的优势。例如，Vertica这类商用软件价格高昂；SparkSQL与Hive这类系统无法保障90%的查询在1秒内返回，在大数据量下的复杂查询可能会需要分钟级的响应时间；而Elasticsearch这类搜索引擎在处理亿级数据聚合查询时则显得捉襟见肘。

正如ClickHouse的"广告词"所言，其他的开源系统太慢，商用的系统太贵，只有Clickouse在成本与性能之间做到了良好平衡，即又快又开源。ClickHouse当之无愧地阐释了"在线"二字的含义，即便是在复杂查询的场景下，它也能够做到极快响应，且无须对数据进行任何预处理加工。

- **数据分片与分布式查询**

  数据分片是将数据进行横向切分，这是一种在面对海量数据的场景下，解决存储和查询瓶颈的有效手段，是一种分治思想的体现。ClickHouse支持分片，而分片则依赖集群。每个集群由1到多个分片组成，而每个分片则对应了ClickHouse的1个服务节点。分片的数量上限取决于节点数量（1个分片只能对应1个服务节点）。

  ClickHouse并不像其他分布式系统那样，拥有高度自动化的分片功能。ClickHouse提供了本地表（Local Table）与分布式表（Distributed Table）的概念。一张本地表等同于一份数据的分片。而分布式表本身不存储任何数据，它是本地表的访问代理，其作用类似分库中间件。借助分布式表，能够代理访问多个数据分片，从而实现分布式查询。

  这种设计类似数据库的分库和分表，十分灵活。例如在业务系统上线的初期，数据体量并不高，此时数据表并不需要多个分片。所以使用单个节点的本地表（单个数据分片）即可满足业务需求，待到业务增长、数据量增大的时候，再通过新增数据分片的方式分流数据，并通过分布式表实现分布式查询。

2.ClickHouse 的缺点

- 没有完整的事务支持
- 缺少高频率、低延迟的修改或删除已存在数据的能力，仅能用于批量删除或修改数据。
- 不支持Transaction：想快就别想Transaction
- 聚合结果必须小于一台机器的内存大小：不是大问题
- 缺少完整的Update/Delete操作
- 支持有限操作系统
- 不支持高并发，官方建议qps为100

3. ClickHouse存储层

列式存储

与行存将每一行的数据连续存储不同，列存将每一列的数据连续存储。

相比于行式存储，列式存储在分析场景下有着许多优良的特性。

1）如前所述，分析场景中往往需要读大量行但是少数几个列。在行存模式下，数据按行连续存储，所有列的数据都存储在一个block中，不参与计算的列在IO时也要全部读出，读取操作被严重放大。而列存模式下，只需要读取参与计算的列即可，极大的减低了IO cost，加速了查询。

2）同一列中的数据属于同一类型，压缩效果显著。列存往往有着高达十倍甚至更高的压缩比，节省了大量的存储空间，降低了存储成本。

3）更高的压缩比意味着更小的data size，从磁盘中读取相应数据耗时更短。

4）自由的压缩算法选择。不同列的数据具有不同的数据类型，适用的压缩算法也就不尽相同。可以针对不同列类型，选择最合适的压缩算法。

5）高压缩比，意味着同等大小的内存能够存放更多数据，系统cache效果更好。

数据有序存储

ClickHouse支持在建表时，指定将数据按照某些列进行sort by。排序后，保证了相同sort key的数据在磁盘上连续存储，且有序摆放。在进行等值、范围查询时，where条件命中的数据都紧密存储在一个或若干个连续的Block中，而不是分散的存储在任意多个Block， 大幅减少需要IO的block数量。另外，连续IO也能够充分利用操作系统page cache的预取能力，减少page fault。

主键索引

ClickHouse支持主键索引，它将每列数据按照index granularity（默认8192行）进行划分，每个index granularity的开头第一行被称为一个mark行。主键索引存储该mark行对应的primary key的值。对于where条件中含有primary key的查询，通过对主键索引进行二分查找，能够直接定位到对应的index granularity，避免了全表扫描从而加速查询。

但是值得注意的是：ClickHouse的主键索引与MySQL等数据库不同，它并不用于去重，即便primary key相同的行，也可以同时存在于数据库中。要想实现去重效果，需要结合具体的表引擎ReplacingMergeTree、CollapsingMergeTree、VersionedCollapsingMergeTree实现。

数据插入、更新、删除

Clickhouse是个分析型数据库。这种场景下，数据一般是不变的，因此Clickhouse对update、delete的支持是比较弱的，实际上并不支持标准的update、delete操作。

标准SQL的更新、删除操作是同步的，即客户端要等服务端反回执行结果（通常是int值）；而Clickhouse的update、delete是通过异步方式实现的，当执行update语句时，服务端立即反回，但是实际上此时数据还没变，而是排队等着

Hbase随机读写，但是Hbase的update操作不是真的update，它的实际操作是insert一条新的数据，打上不同的timestamp，而老的数据会在有效期之后自动删除。而Clickhouse干脆就不支持update和delete。

数据Sharding

ClickHouse支持单机模式，也支持分布式集群模式。在分布式模式下，ClickHouse会将数据分为多个分片，并且分布到不同节点上。不同的分片策略在应对不同的SQL Pattern时，各有优势。ClickHouse提供了丰富的sharding策略，让业务可以根据实际需求选用。

1） random随机分片：写入数据会被随机分发到分布式集群中的某个节点上。

2） constant固定分片：写入数据会被分发到固定一个节点上。

3） column value分片：按照某一列的值进行hash分片。

4） 自定义表达式分片：指定任意合法表达式，根据表达式被计算后的值进行hash分片。

数据分片，让ClickHouse可以充分利用整个集群的大规模并行计算能力，快速返回查询结果。

另外，sharding机制使得ClickHouse可以横向线性拓展，构建大规模分布式集群，从而具备处理海量数据的能力。

数据Partitioning

ClickHouse支持PARTITION BY子句，在建表时可以指定按照任意合法表达式进行数据分区操作，比如通过toYYYYMM()将数据按月进行分区、toMonday()将数据按照周几进行分区、对Enum类型的列直接每种取值作为一个分区等。

数据Partition在ClickHouse中主要有两方面应用：

1）在partition key上进行分区裁剪，只查询必要的数据。灵活的partition expression设置，使得可以根据SQL Pattern进行分区设置，最大化的贴合业务特点。

2）对partition进行TTL管理，淘汰过期的分区数据。

数据TTL

在分析场景中，数据的价值随着时间流逝而不断降低，多数业务出于成本考虑只会保留最近几个月的数据，ClickHouse通过TTL提供了数据生命周期管理的能力。

ClickHouse支持几种不同粒度的TTL：

1） 列级别TTL：当一列中的部分数据过期后，会被替换成默认值；当全列数据都过期后，会删除该列。

2）行级别TTL：当某一行过期后，会直接删除该行。

3）分区级别TTL：当分区过期后，会直接删除该分区。

主备同步

ClickHouse通过主备复制提供了高可用能力，主备架构下支持无缝升级等运维操作。而且相比于其他系统它的实现有着自己的特色：

1）默认配置下，任何副本都处于active模式，可以对外提供查询服务；

2）可以任意配置副本个数，副本数量可以从0个到任意多个；

3）不同shard可以配置不提供副本个数，用于解决单个shard的查询热点问题；

支持数据复制和数据完整性

ClickHouse 使用异步的多主复制技术。当数据被写入到任何一个可用副本后，系统在后台将数据分发给其他副本。

实时数据更新

ClcikHouse 数据是以增量的方式有序的存储在 MergeTree 中。因此，数据可以持续不断高效的写入到表中，并且写入的过程中不会存在任何加锁的行为。

支持近似计算

ClickHouse 提供各种在允许牺牲精度的情况下对查询进行加速的方法：

1. 用于近似计算的各类聚合函数，比如，近似估算distinct values、中位数，分位数等多种聚合函数；
2. 基于数据的部分样本进行近似查询，比如，建表DDL支持SAMPLE BY子句，支持对于数据进行抽样处理；
3. 不使用全部的聚合条件，通过随机选择有限个数据聚合条件进行聚合。

多核并行

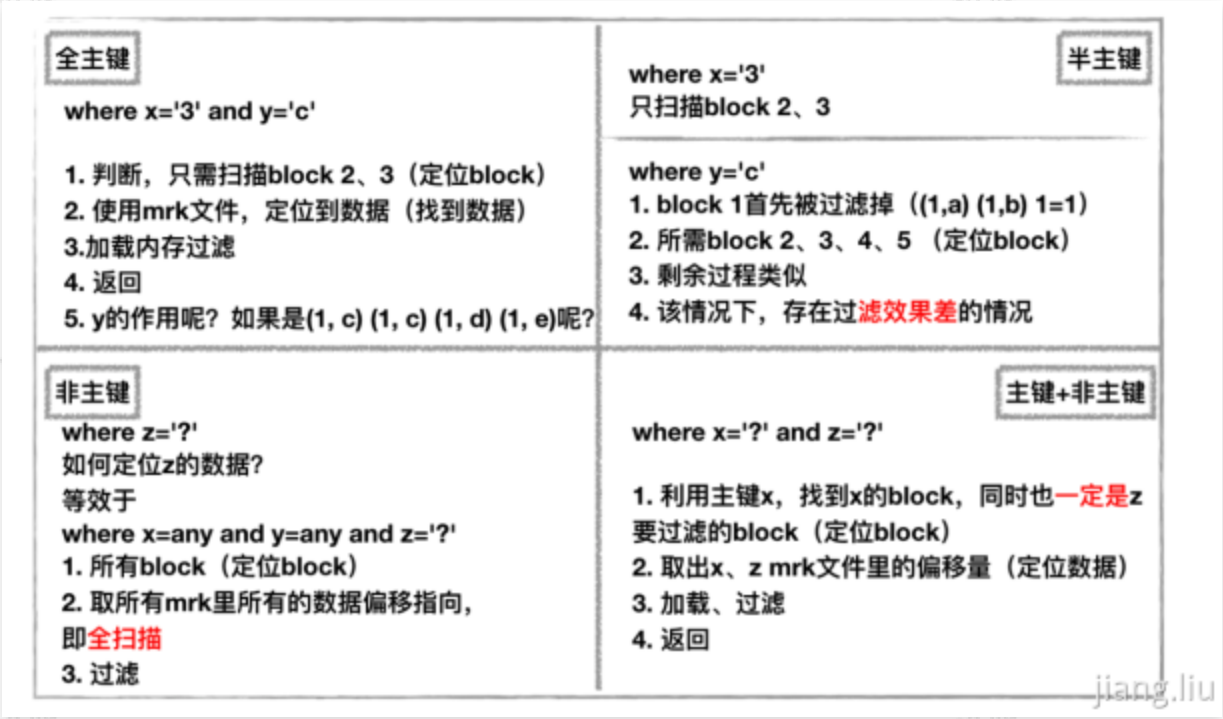
ClickHouse将数据划分为多个分片，每个分片再进一步划分为多个`index granularity`，然后通过多个CPU 核心分别处理其中的一部分来实现并行数据处理。在这种设计下，单条Query就能利用整机所有CPU。极致的并行处理能力，极大的降低了查询延时。

4. 基本原理

记录方式：每隔8192行数据，是1个block,主键会每隔8192，取一行主键列的数据，同时记录这是第几个block

查找过程：如果有索引，就通过索引定位到是哪个block，然后找到这个block对应的mrk文件,mrk文件里记录的是某个block的数据集，在整列bin文件的哪个物理偏移位,加载数据到内存，之后并行化过滤。

主键本身也符合最左原则,下面是查找图,所以查询时最好利用好主键条件：



> 分布式需要借助Zookeeper，ClickHouse大部分都是自运维的，如果我们要保证ClickHouse高可用首先要保证Zookeeper高可用

5.数据同步方式

使用ClickHouse内置的同步方式

只要设置internal_replication为false，那么我们不需要任何其他的配置即可实现数据复制和同步。其实现方式是：

- clickhouse按照权重将数据分成分片数量的等份
- 将对应份数据分别写入该分片中的所有备份中

> internal_replication=false时，往分布式表(注意分布式表只是本地表的view，是不存放任何实体数据的)里面写入数据时，表层面自动同步开启，数据会写入所有备份中（同属一个shard内的表数据相同），但是这个时候是不校验数据一致性的（比如说写入server1的时候成功了，但是写入server1的备份server2的时候有一些没有写入成功，那么这两个互为备份的表就不一致了），也就是说，有可能出现两个备份数据略微不一致的情况，虽然这种可能性很小，另外出现了不一致的时候表之间不会自动同步需要自己手动

使用zookeeper来守卫同步方式

此种同步方式是上一种的优化，需要`配合zookeeper和clickhouse的ReplicatedMergeTree表引擎来使用`，缺一不可。

实现逻辑如下：

- clickhouse会将数据拆成分片数量的等份，然后选择每个分片中的的某一个完好的备份写入数据（只写入一份数据）
- ReplicatedMergeTree会自动同步分片内部的各备份间的数据
- zookeeper会将各分片各备份建立索引id，不停的去检验各备份间数据的同一性（心跳模式）

> internal_replication=true时，如果没有zookeeper的配合如果不使用zookeeper，那么往分布式表写入数据时，是只写入一组备份中的（也就是说同一个shard内部只有一个表写入了数据，其他的表均不会写入数据，除非有一个宕机另外的作为补充下入，但是这个时候表之间的数据就不同了，需要人工手动统一（备份合并就是shard总体）。
>
> internal_replication=true时，使用ReplicatedMergeTree表引擎+zookeeper，这种方案是最笨重但是也是最稳重的方案。

6. 分布式表引擎

分布式表本身是不存储任何实体数据的，分布式表是实体表的镜像（view）。

```java
Distributed(cluster_1st, ods, access_log_replica, rand())
```

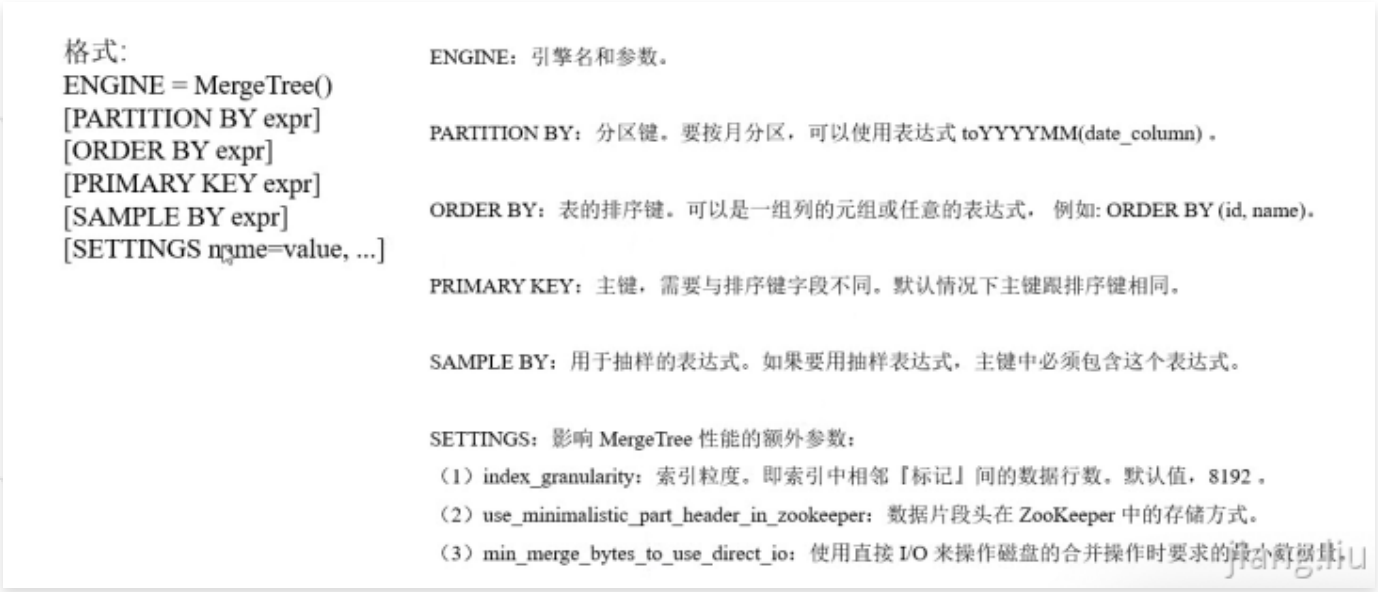
分布式表我划重点如下：

- 分布式表本身并不存储数据，它是本地表的马甲和镜像，只是提供了一个可以分布式访问数据的框架，查询分布式表的时候clickhouse会自动去查询对应的每个本地表中的数据
- 注意AS test.bank_replica，它表明了分布式表所对应的本地表（本地表是存储数据的）
- 可以配置Distributed表引擎中的最后一个参数来设置数据条目的分配方式
- 可以直接往分布式表中写数据，clickhouse会自动按照上一点所说的方式来分配数据和自平衡
- 也可以自己写算法，然后往本地表中写数据（当然这个就比较高级了）

7. mergetree（重点）

MergeTree 允许您依据主键和日期创建索引，并进行实时的数据更新操作。
会自动的合并多个分支，减少数据的存储。
特点：

1. 按照主键进行排序
2. 可以指定分区
3. 可以支持数据副本，保证数据安全性
4. 支持数据采样



索引粒度：每个索引有多行，默认值是8192。

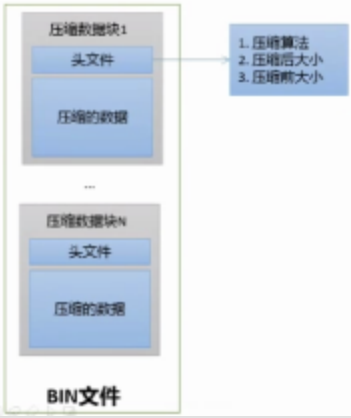
没有指定主键，默认情况下主键和排序键是一样的。

分区表是按照月来分区的，分区就是有节点了。

索引的粒度为3的时候就是每3条数据生成一条索引记录。主键就是索引。

数据存储

bin文件是真正存储数据的文件。
bin文件的数据是按照排序健排序后存储的。
一个bin文件由N个压缩数据块组成。
一个压缩数据块存储压缩前大小为64K~1M 字节的数据。



id（主键）	name	age
A04	铁蛋	15
A01	狗娃	23
A03	二狗	11
A07	狗剩	36
A02	翠花	21

A01	狗娃	23
A02	翠花	21
A03	二狗	11
A04	铁蛋	15
A07	狗剩	36
id.bin	name.bin	age.bin

jiang.liu