UUID和雪花算法

0. 背景

上周和Leader聊天的时候，问起来，UUID和雪花算法哪个是有序的来着？一时发现自己只知道这两个是生成唯一标识的算法，不知道内在原理，尴尬了。

所以赶紧课后来补补课！

1. 为什么需要唯一ID

在复杂分布式系统中，往往需要对大量的数据和消息进行唯一标识，如每一次请求，不管是生成订单，还是退款，都需要对这一次请求生成唯一标识（唯一ID），唯一ID可以作为业务对象的唯一标识符。

同时，随着业务发展，单机数据库已无法满足需求，需要进行分库分表。在分库分表后，数据分散在不同节点，数据库自增主键已无法保证全局唯一性，此时需要生成全局唯一的ID。

2. 唯一ID的要求

1、**全局唯一性**：不能出现重复的ID号，作为业务对象的唯一标识是最基本要求

2、**趋势递增**：有利于使用数据库的聚集索引提高查询效率，同时满足事务版本号、增量消息、排序等特殊需求

3、**信息安全**：ID不应该是连续的，否则恶意用户可以轻易扒取信息

4、**高可用性**：ID生成系统需要高可用，避免单点故障影响整个业务系统

5、**长度适中**：ID不应过长，以免浪费存储空间和影响查询效率。

3. UUID

UUID是由一组32位数的16进制数字所构成，理论上，UUID的总数为`16^32 = 2^128`，约等于`3.4 x 10^38`。也就是说每纳秒产生`1M`个UUID，要花100亿年才会将所有UUID用完。

UUID的标准型式包含32个16进制数字，以连字号分为五段，形式为`8-4-4-4-12`的`32`个字符，如：`550e8400-e29b-41d4-a716-446655440000`。

UUID由以下几部分的组合：

* 当前日期和时间
* 时钟序列
* 全局唯一的IEEE机器识别号，如果有网卡，从网卡MAC地址获得，没有网卡以其他方式获得

3.1 优点

* **全局唯一性强**：UUID通过组合时间戳、随机数等方式生成,重复概率极低，可以确保全局唯一性
* **无中心化**：UUID可以在分布式系统的各个节点独立生成，不需要中心协调，适合分布式场景
* **隐私性好**：UUID不透明，不会像自增ID那样泄露信息

3.2 缺点

* **存储空间大**：UUID通常占用16字节存储空间，比自增整数ID要大很多
* **查询效率低**：UUID无序且较长，在建立索引和进行范围查询时效率较低
* **可读性差**：UUID是一串随机字符，不直观，不利于开发调试
* **生成复杂**：需要额外的UUID生成算法，相对自增ID来说更复杂
* **信息不安全**：基于MAC地址生成UUID的算法可能会造成MAC地址泄露，这个漏洞曾被用于寻找梅丽莎病毒的制作者位置

4. 雪花算法
=======

雪花算法是Twitter开源的一种生成分布式唯一ID的算法，结合了时间戳、机器ID和序列号等元素。

第1位占用1-bit，其值始终是0，可看做是符号位不使用。

第2位开始的41位是时间戳，41-bit位可表示2^41个数，每个数代表毫秒，那么雪花算法可用的时间年限是`(1L<<41)/(1000L360024*365)=69` 年的时间。

中间的10-bit位可表示机器数，即2^10 = 1024台机器，但是一般情况下我们不会部署这么台机器。如果我们对IDC（互联网数据中心）有需求，还可以将 10-bit 分 5-bit 给 IDC，分5-bit给工作机器。这样就可以表示32个IDC，每个IDC下可以有32台机器，具体的划分可以根据自身需求定义。

最后12-bit位是自增序列，可表示2^12 = 4096个数。

4.1 优点

* **全局唯一性**：通过组合时间戳、机器编号和序列号等方式，可以确保生成的ID全局唯一。

* **高性能**：生成ID的过程很快，不需要访问数据库等操作，每秒可生成百万个不重复ID

* **趋势递增**：基于时间戳生成，ID趋势递增，有利于按时间排序

* **无中心化**：各节点可独立生成ID，不需要中心协调，适合分布式场景

* **无第三方依赖**：算法简单，在内存中进行，不依赖第三方库

4.2 缺点

* **依赖机器时钟**：如果时钟回拨，可能会生成重复ID，需要进行时钟同步

* **信息不安全**：ID中包含时间戳和机器编号等信息。可能会暴露系统敏感信息

4.3 解决时钟问题

美团的Leaf-snowflake方案中引入了zookeeper，对每个服务节点（用于发放唯一ID）进行编号。每次当服务节点启动时，需要将当前服务节点的系统时间与先前的时间进行比较。如果超出了阈值，则认为当前服务节点出现了系统时间回拨的问题，该服务节点启动失败并告警。

5. 常见面试题：为什么 MySQL 主键不推荐使用 UUID

1、**性能问题**：UUID 相比自增 ID（例如雪花算法生成的ID）长度更长，导致索引占用更多的存储空间，同时在Innodb中，非聚簇索引还都要包含主键的值，因此每个索引占的空间也都会增大。

2、**数据存储问题**：UUID 生成的ID是随机的，不是按顺序递增的，会导致插入新数据时可能需要频繁地调整索引，并且需要做页分裂来容纳新数据，增加了数据库的I/O操作，影响数据库性能。

3、**复制和分片问题**：使用 UUID 作为主键时，数据的复制和分片可能会面临一些挑战，因为UUID是随机生成的，不同服务器上的UUID可能会冲突或不均匀分布。

原文链接: https://juejin.cn/post/7363635167301025819