链滴

全局唯一主键生成策略

作者：oldcaptain
原文链接：https://ld246.com/article/1411197094156
来源网站：链滴

flickr这一方案的整体思想是：建立两台以上的数据库ID生成服务器，每个服务器都有一张记录表当前ID的Sequence表，但是Sequence中ID增长的步长是服务器的数量，起始值依次错开，这样当于把ID的生成散列到了每个服务器节点上。例如：如果我们设置两台数据库ID生成服务器，那么就一台的Sequence表的ID起始值为1,每次增长步长为2,另一台的Sequence表的ID起始值为2,每次增长长也为2，那么结果就是奇数的ID都将从第一台服务器上生成，偶数的ID都从第二台服务器上生成，样就将生成ID的压力均匀分散到两台服务器上，同时配合应用程序的控制，当一个服务器失效后，系能自动切换到另一个服务器上获取ID，从而保证了系统的容错。

关于这个方案，有几细节这里再说明一下：
1. flickr的数据库ID生成服务器是专用服务器，服务器上只有一个数据库，数据库中表都用于生成Sequence的，这也是因为auto-increment-offset和auto-increment-increment这两个数库变量是数据库实例级别的变量。
2. flickr的方案中表格中的stub字段只是一<code个char(1) NOT NULL存根字段，并非表名，因此，一般来说，一个Sequence表只有一条纪，可以同时为多张表生成ID，如果需要表的ID是有连续的，需要为该表单独建立<code>Seuence表</code>。</code>
3. 方案使用了mysql的LAST_INSERT_ID()函数，这也决定了Sequence表只能有一条记录。
4. 使用REPLACE INTO插入数据，这是很讨巧的作法，主要是希望利用mysql自身的机制生成ID,不是因为这样简单，更是因为我们需要ID按照我们设定的方式(初值和步长)来生成。
5. SELECT LAST_INSERT_ID()必须要于REPLACE INTO语句在同一个数据库连接下才能得到刚插入的新ID，否则返回的值总是0
6. 该方案中Sequence表使用的是MyISAM</strog>引擎，以获取更高的性能，注意：MyISAM引擎使用的是表级别的锁，MyISAM对表的写是串行的，因此不必担心在并发时两次读取会得到同一个ID(另外，应该程序也不需要步，每个请求的线程都会得到一个新的connection,不存在需要同步的共享资源)。经过实际对比测试使用一样的Sequence表进行ID生成，MyISAM引擎要比InnoDB表现高出很多！
7. 可使用纯JDBC实现对Sequence表的操作，以便获得更高的效率，实验表明，即使只使用Sprig JDBC性能也不及纯JDBC来得快！

实现该方案，应用程序同样需要做一些处理，主要是两方面的工作：

1. 自动均衡数据库ID生成服务器的访问
2. 确保在某个数据库ID生成服务器失效的况下，能将请求转发到其他服务器上执行。