# 【分布式ID生成器】

## 类snowflake算法

snowflake是twitter开源的分布式ID生成算法，其核心思想为，一个long型的ID：

* 41bit作为毫秒数
* 10bit作为机器编号
* 12bit作为毫秒内序列号

算法单机每秒内理论上最多可以生成1000\*(2^12)，也就是400W的ID，完全能满足业务的需求。借鉴snowflake的思想，结合各公司的业务逻辑和并发量，可以实现自己的分布式ID生成算法。

## 举例方案

假设公司ID生成器服务的需求如下：

* 单机高峰并发量小于1W，预计未来5年单机高峰并发量小于10W
* 有2个机房，预计未来5年机房数量小于4个
* 每个机房机器数小于100台
* 目前有5个业务线有ID生成需求，预计未来业务线数量小于10个
* …

分析过程如下：

* 高位取从2017年1月1日到现在的毫秒数（假设系统ID生成器服务在这个时间之后上线），假设系统至少运行10年，那至少需要10年\*365天\*24小时\*3600秒\*1000毫秒=320\*10^9，差不多预留39bit给毫秒数
* 每秒的单机高峰并发量小于10W，即平均每毫秒的单机高峰并发量小于100，差不多预留7bit给每毫秒内序列号
* 5年内机房数小于4个，预留2bit给机房标识
* 每个机房小于100台机器，预留7bit给每个机房内的服务器标识
* 业务线小于10个，预留4bit给业务线标识

这样设计的64bit标识，可以保证：

* 每个业务线、每个机房、每个机器生成的ID都是不同的
* 同一个机器，每个毫秒内生成的ID都是不同的
* 同一个机器，同一个毫秒内，以序列号区区分保证生成的ID是不同的

将毫秒数放在最高位，保证生成的ID是趋势递增的



**缺点：**

* 由于“没有一个全局时钟”，每台服务器分配的ID是绝对递增的，但从全局看，生成的ID只是趋势递增的（有些服务器的时间早，有些服务器的时间晚）

## 改进方案

条件：

* 单机高峰并发量小于1W，预计未来5年单机高峰并发量小于10W
* 机器运行20年
* 最多不超过50个区域独立布署应用
* 每个区域的应用数不超过100个

## 实现

需要40位用于记录毫秒数

需要6位记录区域

需要7位记录毫秒内的序列号

需要7位记录应用编号

剩下4位进行保留

