# 参考

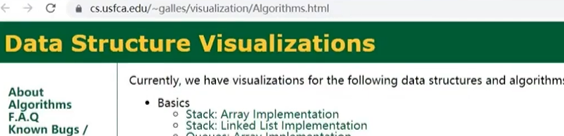
B站讲的最好的Java架构师教程全集（2021最新版）

<https://www.bilibili.com/video/BV1e44y1y7aE?p=14&spm_id_from=pageDriver>

# 数据库

## （B+树）千万级数据表如何用索引快速查找

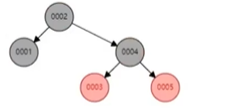
**如下的数据结构展示网站**



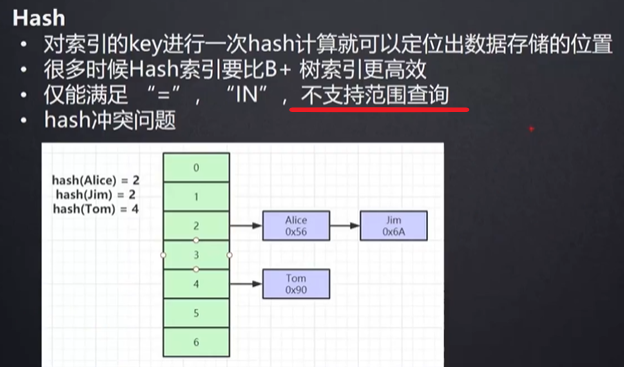
**索引可选数据结构：**

二叉树（可能退化成链表）

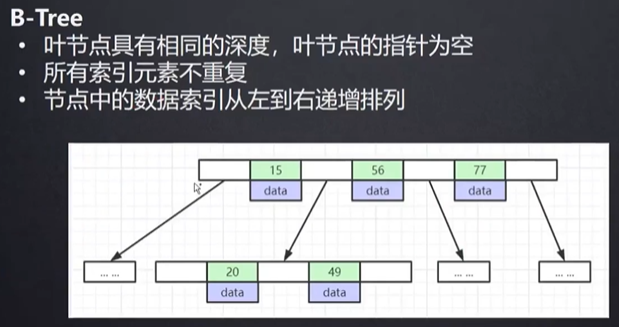
红黑树（二叉平衡树）：数据规模大时，查找复杂度增高



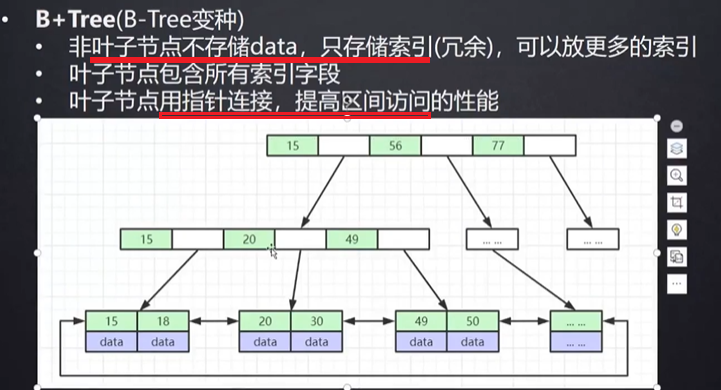
hash表：不支持范围查找，hash冲突问题



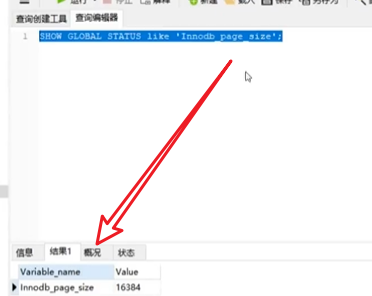
B-Tree：



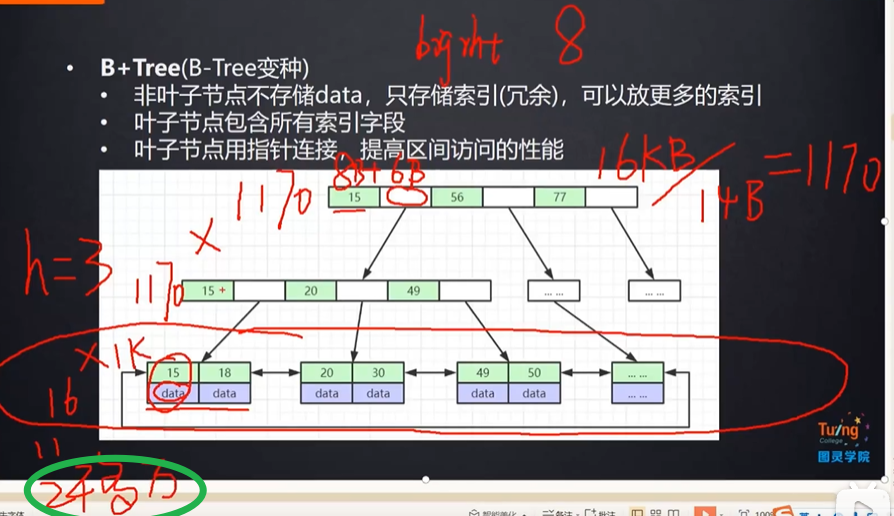
**B+树：**千万级或更大只需查找四次



**mysql默认给每个B+树节点数据结构大小16KB**



根据如下计算：1张表可存储1170\*1170\* 16 = 2kw数据 （假设每个数据元素大小占用1K，则叶子节点只有16个元素）



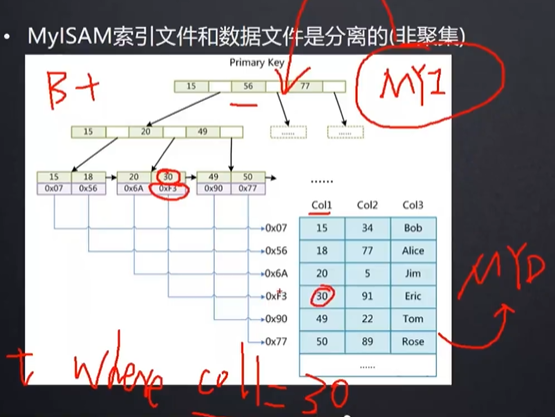
**为提升查找效率，mysql会提前将非叶子节点数据加载到内存中，仅一次磁盘IO查找。**

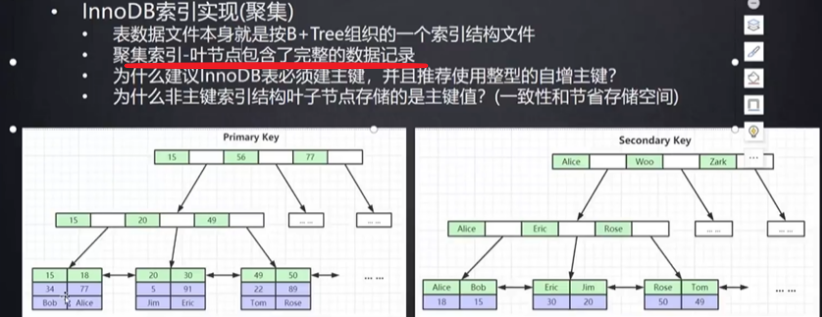
**叶子结点仅保存在磁盘上，不可能一次都给加载到内存中。**

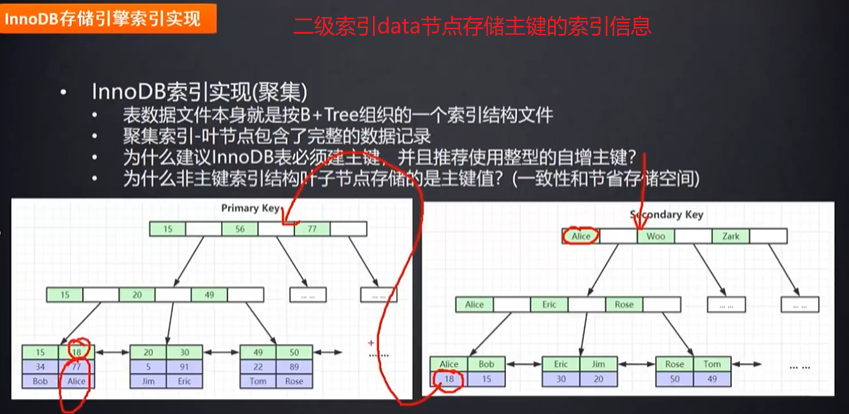
## 如何基于B+树精准建立高性能索引

**知识点：聚集索引InnoDB与非聚集索引MyISAM**









## 聚集索引、覆盖索引和索引下推是什么

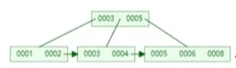
**问：为什么建议InnoDB表必须建主键，且推荐使用整型的自增主键**

答：

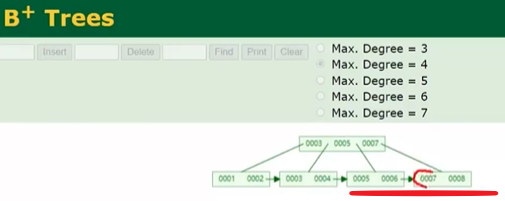
1、InnoDB要求必须通过B+树来存储数据，如果不指定，则自动选择数据均唯一的某一列当做主键。如果找不到，则使用默认创建的rowid来作为主键存储。

2、整型：方便比较

3、自增原因：不会出现新增元素导致对已有节点的分裂问题，如下图如果插入7场景

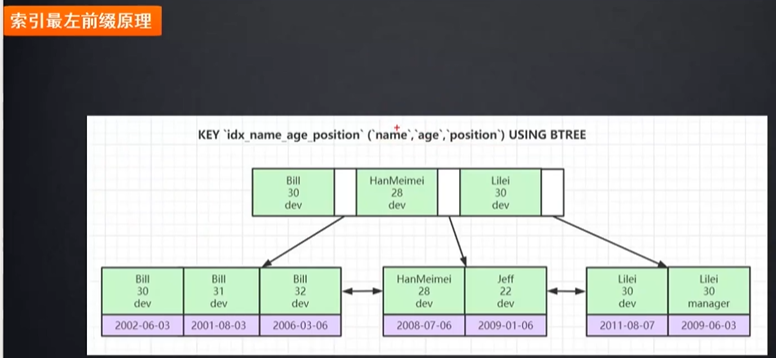


插入7 后，已有右侧节点重新分裂

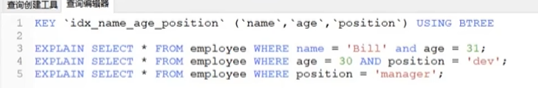


## 联合索引底层存储原理（索引最左前缀原理）

按照索引字段顺序挨个比较，相同则比较后续字段（见左下角）



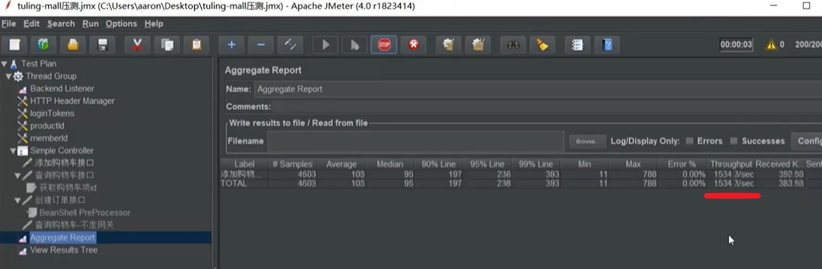
如下哪个会走联合索引：仅第1个，按照索引定义顺序



为何第2和第3不能按照联合索引查找：以第2个age=30为例，在上面截图中可以看到age并不是有序的（存在多个30的值）。

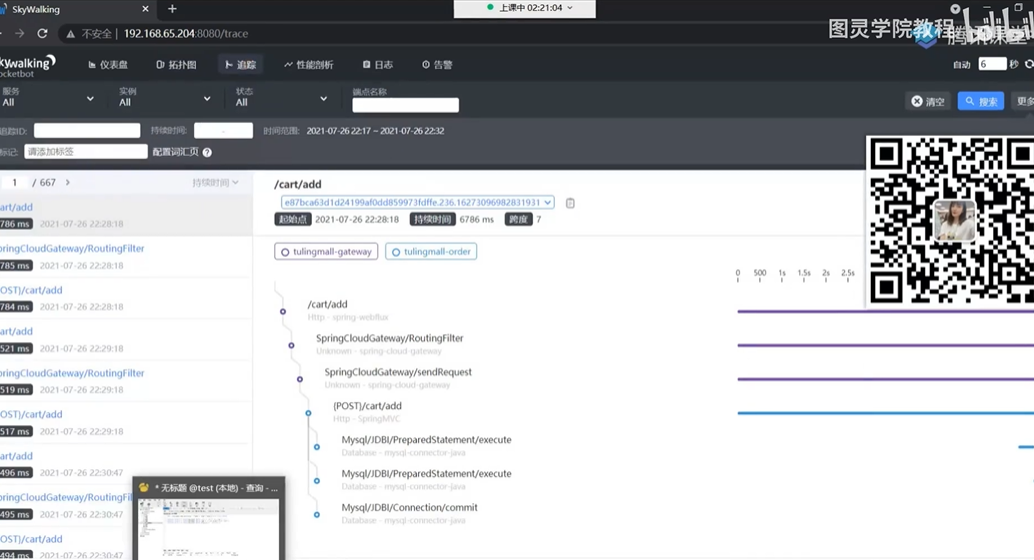
## 如何使用mysql查询计划定位慢SQL问题

### 使用JMeter压测

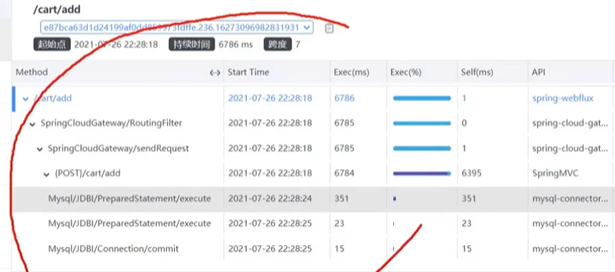


### Skywalking调用链分析

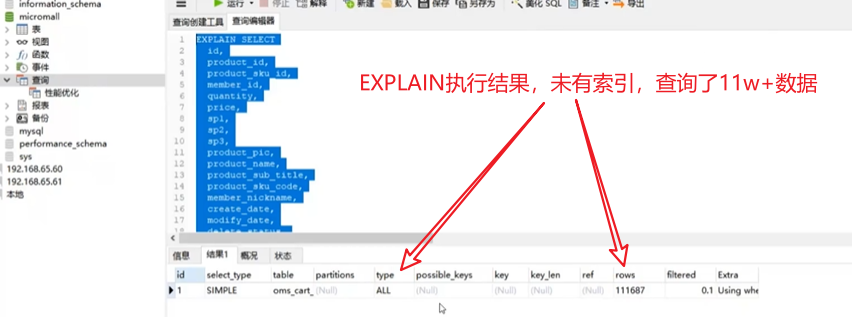
分析整个调用链的内部处理耗时



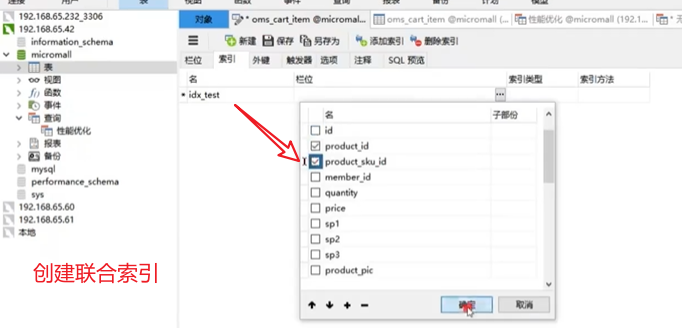
分析出第1个mysql执行300ms+，问题点



获取具体的mysql语句，通过EXPLAIN执行，发现没有创建索引，导致效率问题

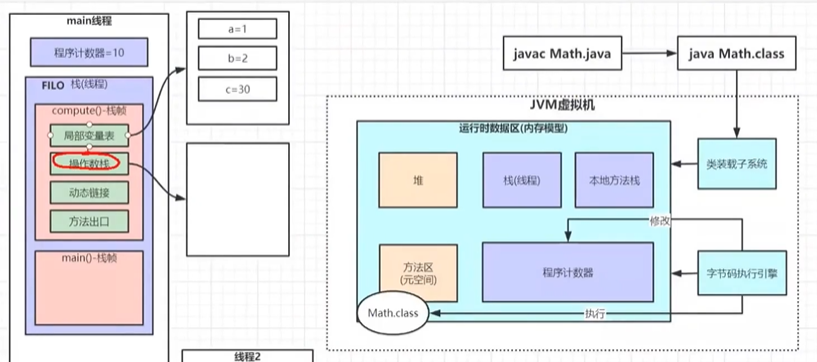


处理结果：增加联合索引

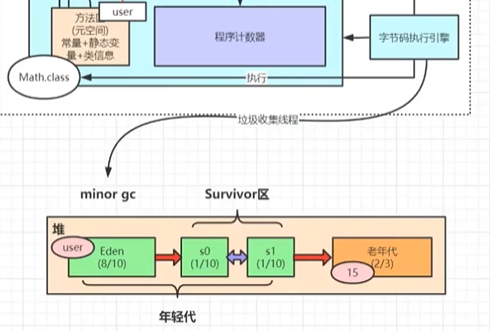


# JVM

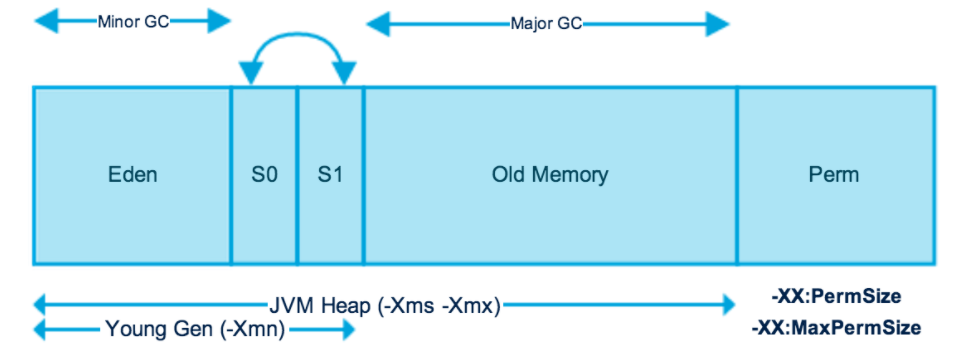
## 深入字节码剖析Java虚拟机内存模型



## JVM垃圾回收机制详解



15：年轻代数据的老化时间，达到一定次数进入老年代；或年轻代大小超过一定大小，直接上升到老年代



永久代是Hotspot虚拟机特有的概念，是方法区的一种实现，别的JVM都没有这个东西。在Java 8中，永久代被彻底移除，取而代之的是另一块与堆不相连的本地内存——元空间。

**// 所以在Java8中必须使用专属的MetaSpace虚拟机参数指定元数据区，原有的PermSize参数不再生效**

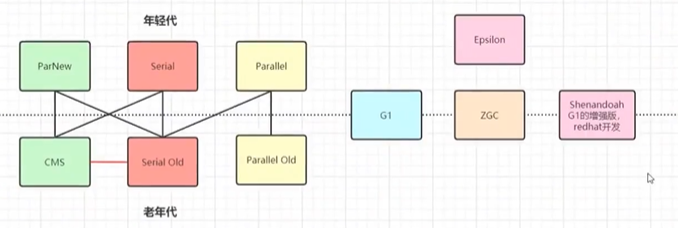
永久代或者“Perm Gen”包含了JVM需要的应用元数据，这些元数据描述了在应用里使用的类和方法。注意，永久代不是Java堆内存的一部分。永久代存放JVM运行时使用的类。永久代同样包含了Java SE库的类和方法。永久代的对象在full GC时进行垃圾收集。

STW: Stop the world

minor GC:

major GC:

full GC:



## 关键虚拟机参数

newRatio

## JVM调优工具

### 原始命令

jstat -gcutil

jstack

jmap

### VisualVM

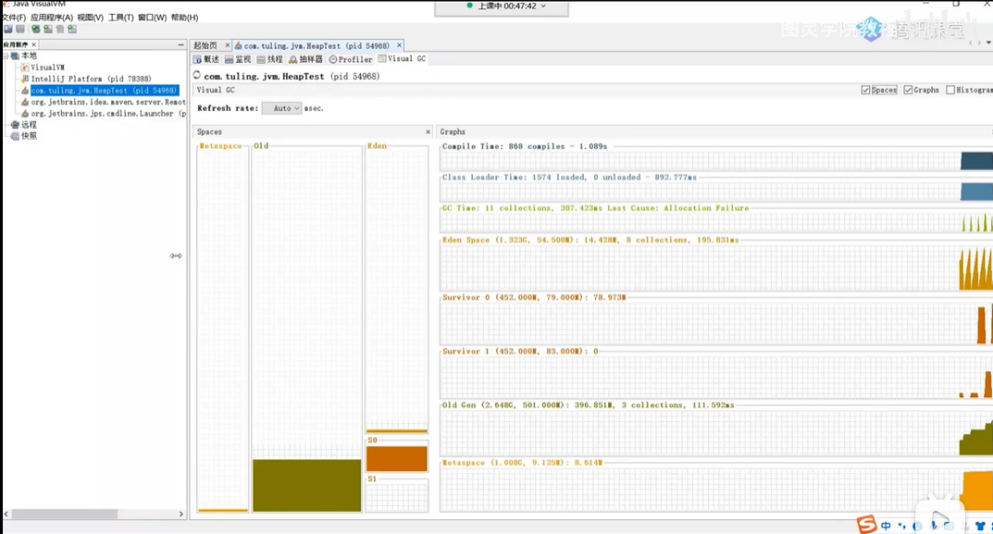
概述：整体结果，CPU、内存等信息

线程：查看线程执行繁忙情况状态，等待还是处理中

抽样器：分析CPU执行耗时



VisualGC卡片：需要安装插件，JVM内部各区域的使用情况



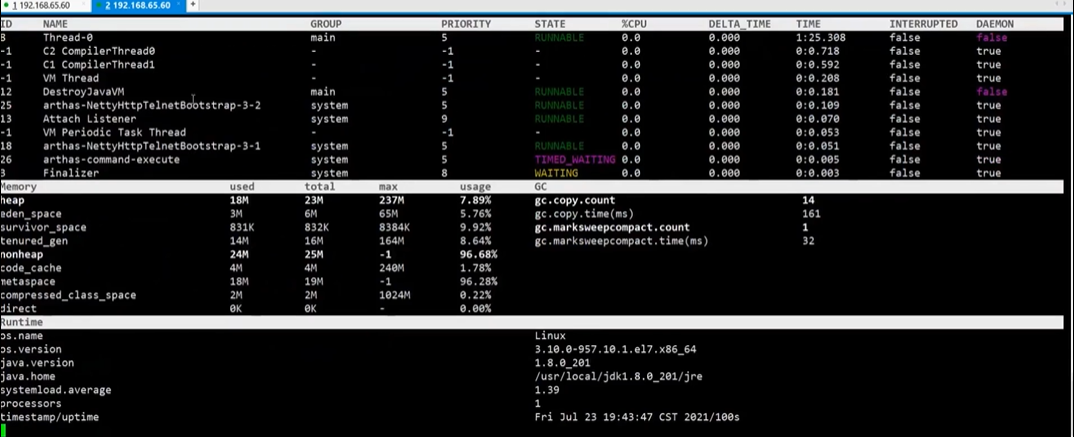
### 阿里内部JVM调优工具Arthas详解

**火焰图**：执行慢耗时调优



命令：

Dashboard：内存分配，线程情况

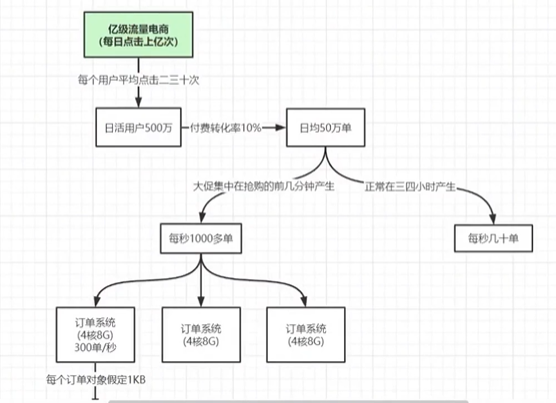


threads -n死锁

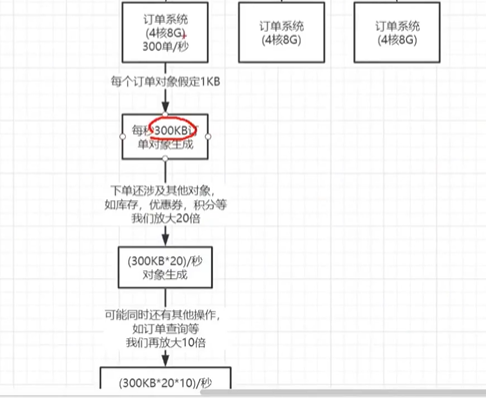
jad 完整类路径： 反编译代码

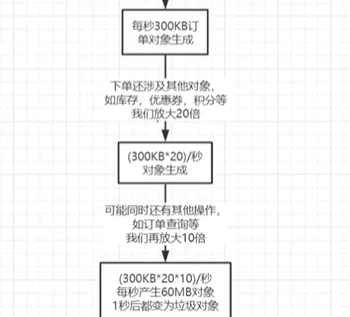
## 双十一亿级电商网站JVM参数调优实战

**问题：频繁FULL GC**

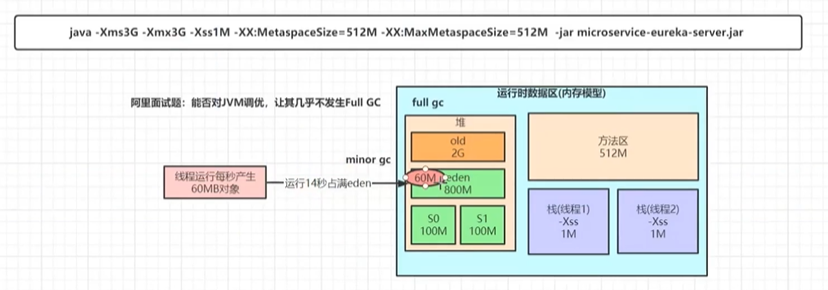


**具体评估实际的内存诉求**

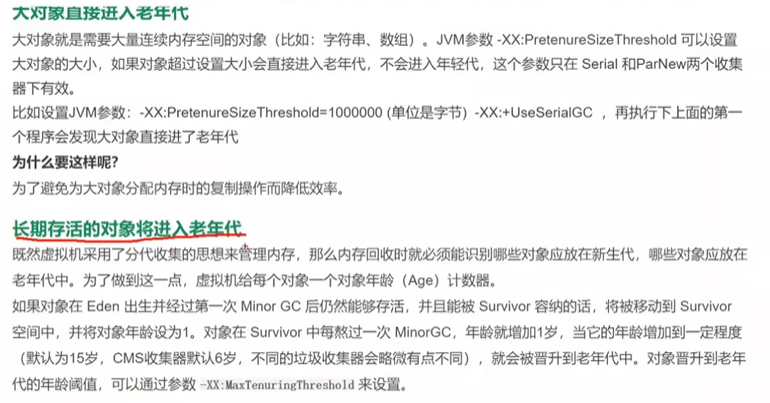


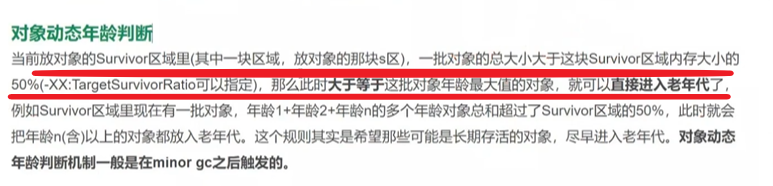


错误的配置：



**针对minor GC时，仍未完成的任务，其内存占用60m > 50% \* 100M,所以被直接放到老年代，导致老年代被快速占满，而导致频繁FULL GC**





解决方案：提升新生代大小



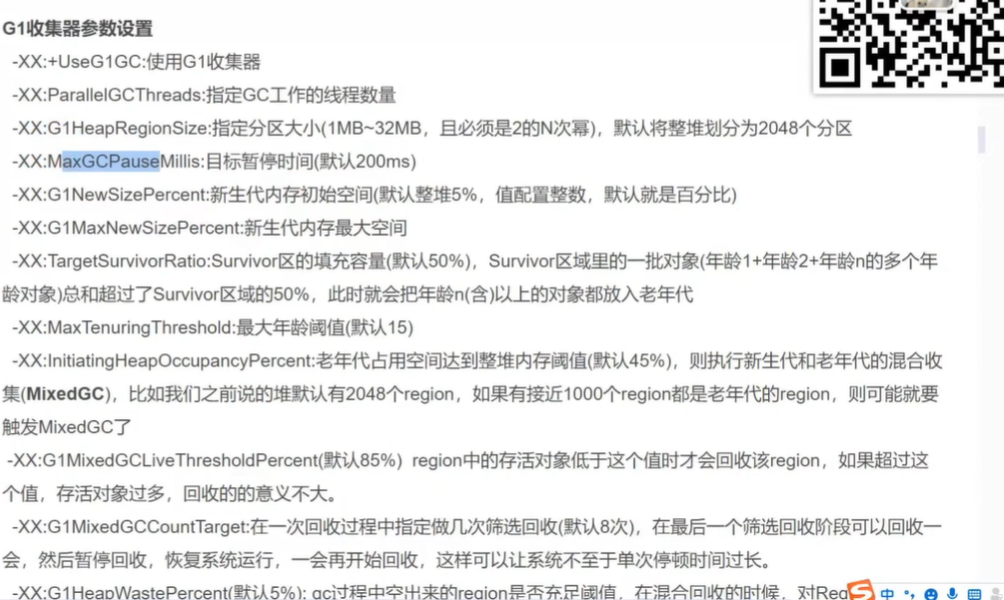
## 垃圾回收器CMS/G1/ZGC

<https://www.bilibili.com/video/BV1e44y1y7aE?p=18>

如何应对单机支撑十几万并发？

此时通常设置很大的堆内存，如果等到常规新生代满了后再回收通常会造成几秒的STW。CMS

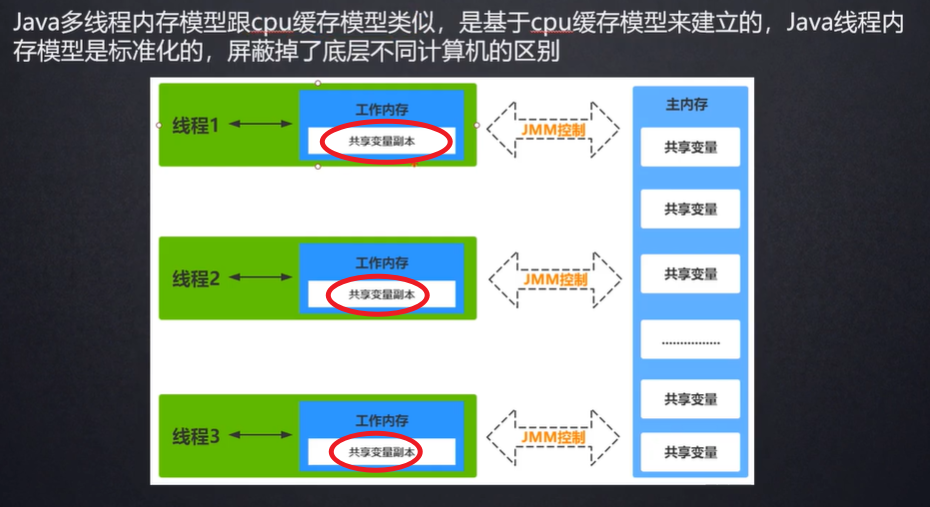
解决方法：边使用边回收，介绍每次STW的时间，分多次进行回收=》 G1回收



# JMM内存模型

## CPU多核并发缓存架构解析

**每个线程都保存了一份副本，需要配置volatile刷新可见性**



https://www.bilibili.com/video/BV1e44y1y7aE?p=39

# Netty

# IO/NIO

## Reactor模型