# TLAB是什么？

## new对象与指针碰撞

new对象怎么就出问题了呢？  
java中我们要创建一个对象,用关键字new就可以了。但是，在我们日常中，有很多生命周期很短的对象。比如：

public void dome(){

User user=new user();

user.sayhi();

}

这种对象的作用域都不会逃逸出方法外，也就是说该对象的生命周期会随着方法的调用开始而开始，方法的调用结束而结束。  
假设JVM所有的对象都放在堆内存中(为什么用假设，因为JVM并不是这样)一旦方法结束，没有了指向该对象的引用，该对象就需要被GC回收，如果存在很多这样的情况，对GC来说压力山大呀。

那么什么又是指针碰撞呢？  
假设JVM虚拟机上，堆内存都是规整的。堆内存被一个指针一分为二。指针的左边都被塞满了对象，指针的右变是未使用的区域。每一次有新的对象创建，指针就会向右移动一个对象size的距离。这就被称为指针碰撞。



好，问题来了。**如果我们用多线程执行刚才的dome方法，一个线程正在给A对象分配内存，指针还没有来的及修改，同时为B对象分配内存的线程，仍引用这之前的指针指向**。这样就出现毛病了。  
(要注意的是，上面两种情况解决方案不止一个，TLAB是解决方法之一)

## TLAB的出现

TLAB的全称是Thread Local Allocation Buffer，即**线程本地分配缓存区**，这是一个线程专用的内存分配区域。

如果设置了虚拟机参数 -XX:UseTLAB，在线程初始化时，同时也会申请一块指定大小的内存，只给当前线程使用，这样每个线程都单独拥有一个空间，如果需要分配内存，就在自己的空间上分配，这样就不存在竞争的情况，可以大大提升分配效率。

TLAB空间的内存非常小，缺省情况下仅占有整个Eden空间的1%，也可以通过选项-XX:TLABWasteTargetPercent设置TLAB空间所占用Eden空间的百分比大小。

TLAB的本质其实是三个指针管理的区域：start，top 和 end，每个线程都会从Eden分配一块空间，例如说100KB，作为自己的TLAB，其中 start 和 end 是占位用的，标识出 eden 里被这个 TLAB 所管理的区域，卡住eden里的一块空间不让其它线程来这里分配。

**TLAB只是让每个线程有私有的分配指针，但底下存对象的内存空间还是给所有线程访问的，只是其它线程无法在这个区域分配而已**。从这一点看，它被翻译为 线程私有分配区 更为合理一点  
当一个TLAB用满（分配指针top撞上分配极限end了），就新申请一个TLAB，而在老TLAB里的对象还留在原地什么都不用管——它们无法感知自己是否是曾经从TLAB分配出来的，而只关心自己是在eden里分配的。

## TLAB的缺点

事务总不是完美的，TLAB也又自己的缺点。因为TLAB通常很小，所以放不下大对象。  
1，TLAB空间大小是固定的，但是这时候一个大对象，我TLAB剩余的空间已经容不下它了。(比如100kb的TLAB，来了个110KB的对象)  
2，TLAB空间还剩一点点没有用到，有点舍不得。(比如100kb的TLAB，装了80KB，又来了个30KB的对象)  
所以JVM开发人员做了以下处理，设置了最大浪费空间。  
当剩余的空间小于最大浪费空间，那该TLAB属于的线程在重新向Eden区申请一个TLAB空间。进行对象创建，还是空间不够，那你这个对象太大了，去Eden区直接创建吧！  
当剩余的空间大于最大浪费空间，那这个大对象请你直接去Eden区创建，我TLAB放不下没有使用完的空间。

当然，又回造成新的病垢。  
3，Eden空间够的时候，你再次申请TLAB没问题，我不够了，Heap的Eden区要开始GC，  
4，TLAB允许浪费空间，导致Eden区空间不连续，积少成多。以后还要人帮忙打理。