Java多线程优化之偏向锁原理分析Java认证考试 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/644/2021_2022_Java_E5_A4_9A_E7_BA_BF_c104_644792.htm Java偏向锁(Biased Locking)是Java 6引入的一项多线程优化。它通过消除资源无竞争情况下的同步原语，进一步提高了程序的运行性能。 轻量级锁也是一种多线程优化，它与偏向锁的区别在于，轻量级锁是通过CAS来避免进入开销较大的互斥操作，而偏向锁是在无竞争场景下完全消除同步，连CAS也不执行（CAS本身仍旧是一种操作系统同步原语，始终要在JVM与OS之间来回，有一定的开销）。 所谓的无竞争场景，举个例子，就是单线程访问带同步的资源或方法。 偏向锁实现原理 偏向锁，顾名思义，它会偏向于第一个访问锁的线程，如果在接下来的运行过程中，该锁没有被其他的线程访问，则持有偏向锁的线程将永远不需要触发同步。 如果在运行过程中，遇到了其他线程抢占锁，则持有偏向锁的线程会被挂起，JVM会尝试消除它身上的偏向锁，将锁恢复到标准的轻量级锁。(偏向锁只能在单线程下起作用) 通过下图可以更直观的理解偏向锁：