**[性能——工具篇](http://wiki.primeton.com/pages/viewpage.action?pageId=12878492)**

     相信我，最好的工具是自己的双手和跳跃的思维。

aix，hp-ux，windows，linux，loadrunner，jmeter，rpt，was，weblogic，tomcat，db2，oracle，mysql。

     你看，要面对的环境，工具，中间件是如此的多。

     每个都需要熟悉，需要精通，还需要知道应用开发，应用架构，企业IT架构，这真是一个不可能完成的任务。

     那么，怎么来做呢？怎么来面对一个完全陌生的环境，来做性能测试和调优呢？当然，我继续按照我的想法写，您按照您的想法读，觉得合理，就彩一个，觉得不合理，就当看个笑话，绝对没有误导您的意思。

     我们做的是JAVA，当前几乎所有的应用都会是运行在JVM之上，如果要测试的是C++，C，请主动略过。有些有servlet的接入，有些没有。

     性能测试相对简单，录制脚本，写脚本，执行，生成报告，没做过的同志，找个入门的文章，看看，学学，包括参数化，基本就问题不大了。就算是用JAVA协议来做测试，也是吭哧吭哧写一些模拟调用的代码，结束。

     好吧，看到这里，你会发现，我重点想讲的，肯定不是loadrunner，RPT，jmeter，ab，在整个测试过程中，性能测试执行 的占比并不会太大，当然，如果是作为第三方测试，来专做测试，不管其他的，不在此处讨论之列。我们要做的，肯定是通过性能测试发现问题，并在性能测试的过 程中解决问题，让应用的性能达到最佳。

     好了，那我们经常遇见的问题还是在理念篇里说的三个问题，IO，CPU，内存。

1. 如果交易响应慢，CPU使用率不高，那看看内存，是不是满了，一直在full GC？IO是不是满了，IO拥堵导致交易响应慢？
2. 跑着跑着就OOM了，如何来界定问题的所在？

     我们一个问题一个问题的来说工具：

**先说一个整体的工具，nmon。**

     这是aix上常用的工具，能查看到当前服务器的资源使用情况，有同学把这个工具搬到了linux上，那在linux上，可以拷贝一些相关的工具上去，就可以使用nmon了。  
     同样，具体怎么用，自己google，每个人都有每个人的用法，我就打开看看资源使用情况。

**继续接着说问题：**

1. 交易响应慢
   1. 应用服务器CPU资源使用很高  那我们需要看看CPU在忙些啥，使用系统命令：  
      kill -3 PID。PID为JAVA进程的PID ，这个命令只是用来答应个线程堆栈。  
      别小看这句命令，如果你对系统足够的了解，对于平台，应用有足够的了解，你会在中间发现足够多的信息。
      1. JVM现在做的，是不是你希望它做的？  
             JVM的线程栈中打出来的是当前正在执行的线程，如果你对应用了解，你应该能知道JVM现在应该在做什么。
      2. JVM现在做的最多的，是不是合理的？  
             JVM现在做的最多的事情，是不是合理的？比如会是大量的IO等待socketreader在等待数据库返回，那数据库应该不太对劲。
      3. 里面有没有你不希望看到的线程，就是它现在不应该做的？  
             缓存是一次加载的，如果在后面执行的线程中，还陆续的看到加载缓存的，那是不是就不对？有些配置文件，本拟是一次加载的，为何会持续有运行？有些 jar包本来是在classloader里的，为什么还要每次执行多个地方查找？如果看到这些，恭喜你，你已经发现问题了。
      4. 有拥堵么？拥堵是因为啥？  
             除了hotspot的jdk，ibm，jrockit都会有很直白的告诉你，这个线程堵了，是被谁堵的，这个堵的线程正在干嘛？干嘛就会堵住别人？是因为CPU太忙不小心堵的，还是CPU不忙，故意堵的？
      5. CPU间歇性很忙，怎么处理？  
             nmon只能看到CPU的繁忙程度，而如果要看到CPU的一个曲线图，可以通过visualvm或者jdk自带的jconsole通过把界面映射 到本地的方式，查看CPU的规律，jconsole是个好工具，还有jhat，jmap，jstatd，jvisualvm等，均可以好好学习使用。如果 发现是full GC导致，可以调整GC策略等方式来完成，当然，也可以通过内存堆栈来分析内存内容来避免这些问题，如一次取1000条数据在内存 sort，然后完成后要gc，这个功能点一并发，full GC的频率就跟玩儿似的，刷刷的。还有通过jdk参数，添加GC log，也可以很有效的发现 full GC的频率。
   2. 应用服务器CPU资源使用率不高，我们也需要看看CPU在做什么。  
           照例用kill -3 PID来打线程栈。  
           有一种情况是CPU完全不忙的，他就在那里等着，这种可能的原因是压力不大，比如并发没有上去啊，网络带宽太小，表单都提交不过来等，这是个问 题，但是这个问题在用nmon就能发现网络的使用情况，配合ftp等工具就可以定位是否网络存在瓶颈等。此处不做赘述。主要说的，是另外一种情况，就是压 力在也过来了，网络也没瓶颈，交易响应就是很慢，应用服务器CPU使用率也不高，怎么办？  
           照例从线程栈中发现端倪，如前文所述，如果是大量的IO 等待socketreader返回， 那问题就比较明显了，远端没返回嘛，如果是调用远端服务的，可以单压服务看响应时间，如果是调数据库的，那可以马上去看数据库的情况了。  
           单压远端服务这事儿吧，不细说，我们来说说数据库。  
           我们常见的数据库问题，查询太慢，导致远端等着数据库返回数据。那怎么来优化数据库太慢这个破事呢？本拟此文不是细说数据库优化，所以只是蜻蜓点水的说一些常用的方式。  
           数据库很慢，那首先得界定慢在哪？是CPU顶不住，还是IO满了？  
           linux或aix通过nmon均可查看当前的资源使用情况。如果是windows，可以使用windows自带的工具perfmon来辅助查看。  
           知道是CPU或者IO的情况之后，通过数据库快照抓取，就能有针对性的分析占用资源的语句了，如何分析，不细说，google一大把。  
           还有一个工具，我们要常学着看数据库的执行计划，table scan，index rangescan，这些，学会看了，就简单很多。  
           优化的方式无非就是两种，减少IO开销或增大IO，减少CPU开销。  
           减少CPU开销，我们说要用简单的sql做复杂的业务，把逻辑运算使用java，不要在sql中做大量的逻辑运算，合理的运用笛卡尔积等等，我也不太懂。  
           减少IO开销，能走索引的别走全表扫描，能使用缓存的使用缓存，pga，sga合理配置等。  
           增大IO，在性能测试的过程中，往往数据库的IO不够用，而又会给多块盘，通过nmon查看IO使用，发现IO集中在单盘上，那接下来的事情，就是分散IO。  
           如果可以退到操作系统之前，可以通过raid的模式，如果不可以，可以通过重建数据库，通过逻辑卷的模式挂载多块磁盘来分散IO。
2. OOM问题  
        如果系统会OOM，系统一定是会OOM的。  
        继续分几种情况讲解这个问题。当然，最好用的工具还是JVM的参数。
   1. 参与交易的数据量过大，导致OOM  
           常见于各种初级人员开发的系统中，我们常说的需要的人员素质不用太高，其实我们说的是编码功底不用太高。结果大家一理解吧，嘿，那行，我找几个应届毕业生，培训一下了就开始干系统。  
           在这种系统中，往往会存在OOM的隐患。因为受限于本身的开发能力，然后一些本来算法稍微设计设计，SQL稍微优化优化就能搞定的问题，不会弄，用构件搞，简单。结果经常出现查询10万条记录到内存中进行排序。而本身经验的不足导致对于系统的预估不够。  
           对于这种点击某几个业务会OOM的情况，通过JVM监控，例如GC log，jvisualvm，jconsole等工具皆能看出端倪，剩下的工作就是对当前业务代码的走查，看是否查询分页等。  
           这些问题相对比较粗，容易重现和定位，也容易解决。
   2. 系统中各种缓存的使用  
           这种问题相对就比较高阶了，缓存一般都在代码中写，而缓存的添加又是缓慢添加的。对线上来说，可能要运行一周才会出现一次OOM，这种问题，就比较难搞，第一是重现，第二是定位，第三才是解决。  
           **如何重现？**  
           既然系统是会OOM的，那么我把分配给JVM的内存压下来，就应该比较容易重现。对于调整JVM内存的事情，很多人会自欺欺人的说，重现内存 小，OOM，线上内存大，就不会OOM。请注意的是，OOM本身与内存的大小没有直接的关系，如果会OOM，一定会OOM，只是时间早晚的问题。之前碰到 过一个系统，运行一周增加8M，很稳定上升，系统非配的是4G内存，算一下，得运行多久才会OOM，但是它一定会OOM。  
           **如何定位？**  
           hotspot和jrockit均可以通过MAT，通过调整JVM的参数，每隔一段时间打印内存栈。通过工具比较一下两个内存栈，能很轻易的发现内存的突变部分的数据结构，便于定位问题。  
           IBM的JDK需要通过IBM的HA的分析工具，也可以隔一段时间打个内存堆栈。只是各JVM参数，各位都可以google出来。  
           **如何解决？**  
           找到问题的所在之后，下面我们要做的，是解决问题。  
           对于缓存来说，主要的解决思路只有一种，为什么没有移除？  
           重点关注异常处理和finnally 的处理即可。

     工具提到了一些，当然，我认为，最好的工具还是自己的思维。

To be continued……