|  |
| --- |
| **性能测试中如何定位性能瓶颈** |
| **地址：http://www.uml.org.cn/Test/20114152.asp** |
| 2011-4-15 来源：网络 |
|  |
| |  | | --- | | 性能测试的概念是什么，基本目的是什么，我想大家都基本清楚，不作详述，总之，性能测试只是测试过程中的一种方式，帮助我们的功能更好的运行，如果功能测试是可用，易用，满足需求、用户使用为目的，性能测试无非就是让这些目的更流畅。没有什么专业的概念，无非实现两个字：好用！  所以，性能测试这种测试方式在发生过程中，其中一个过渡性的工作，就是对执行过程中的问题，进行定位，对功能的定位，对负载的定位，最重要的，当然就是问题中说的“瓶颈”，接触性能测试不深，更非专家，自己的理解，瓶颈产生在以下几方面：   * 1、网络瓶颈，如带宽，流量等形成的网络环境 * 2、应用服务瓶颈，如中间件的基本配置，CACHE等 * 3、系统瓶颈，这个比较常用：应用服务器，数据库服务器以及客户机的CPU，内存，硬盘等配置 * 4、数据库瓶颈，以ORACLE为例，SYS中默认的一些参数设置 * 5、应用程序本身瓶颈，   针对**网络瓶颈**，现在冒似很少，不过也不是没有，首先想一下如果有网络的**阻塞，断网，带宽被其他资源占用，限速**等情况，应用程序或系统会是什么情况，针对WEB，无非是超时，HTTP400，500之类的错，针对一些客户端程序，可能也是超时，掉线，服务器下发的，需要服务器返回的信息获取不到还有一种更明显的情况，应该就是事务提交慢，如果封装事务的代码再不完善，一般造成的错误，无非就是数据提交不完整，或者因为网终原因+代码缺陷造成重复性提交。如此综合下来，肯定是考虑网络有瓶颈，然后考虑网络有问题时，怎样去优化，是需要**优化交互的一些代码，还是接口**之类的。  **应用服务的瓶颈**的定位，比较复杂，学习中，不过网上有很多资料可以参考的。一般像tomcat，weblogic之类的，有默认的设置，也有经过架构和维护人员进行试验调试的一些值，这些值一般可以满足程序发布的需要，不必进行太多的设置，可能我们认识的最基本的就是**JAVA\_OPTS**的设置，maxThreads，time\_out之类的参数我们做借助LR，Jemeter或webload之类的工具，执行性能测试，尤其是对应用服务造成了压力，如果应用服务有瓶颈，一般我们设置的**log4j.properties**，日志都会记录下来。然后**根据日志**，去进一步确定应用服务的问题  **系统瓶颈**，这个定位虽说比较复杂，但是有很多前辈的经验值参考，不作说明，相信用LR的同行，也可以从性能记数器中得出一些指标值，加上nagios，cacti，可以很明显的看出系统哪些资源够用，哪些资源明显不够用。不过，一般系统瓶颈的造成，是因为应用程序本身造成的。关于这点儿的分析和定位，就需要归入应用程序本身瓶颈分析和定位了。  现在基本所有的东东，都离不开**数据库**这个后台，数据库的瓶颈实在是不知道是什么概念，数据库管理员的工作，数据库管理员日常做的工作，可能就是有瓶颈定位的工作，比如：查询一下V$sys\_event，V$sysstat，v$syssql之类的表，比对一下日常正常情况下的监控数据，看一下有没有异常等。其他方面，我也不是太了解。  **应用程序瓶颈**，这个是测试过程中最需要去关注的，需要测试人员和开发人员配合执行，然后定位，我这儿做的大都是执行性的，比如会有脚本去运行，开发人员会结合**jprofiler**之类的工具，去看一下**堆遍历，线程剖析**的情况确定哪儿有问题。大致是这样，没有实际操作过  逐步细化分析，先可以监控一些常见衡量CPU，内存，磁盘的性能指标，进行综合分析，然后根据所测系统具体情况，进行初步问题定位，然后确定更详细的监控指标来分析。  **怀疑内存不足时：**  **方法1：**  【监控指标】：Memory Available MBytes ，Memory的Pages/sec， page read/sec， Page Faults/sec  【参考值】：  如果 Page Reads/Sec 比率持续保持为 5，表示可能内存不足。  Page/sec 推荐00-20（如果服务器没有足够的内存处理其工作负荷，此数值将一直很高。如果大于80，表示有问题）。  **方法2：根据Physical Disk 值分析性能瓶颈**  【监控指标】：Memory Available MBytes ，Pages read/sec，%Disk Time 和 Avg.Disk Queue Length  【参考值】：%Disk Time建议阈值90%  当内存不足时，有点进程会转移到硬盘上去运行，造成性能急剧下降，而且一个缺少内存的系统常常表现出很高的CPU利用率，因为它需要不断的扫描内存，将内存中的页面移到硬盘上。  **怀疑内存泄漏时**  【监控指标】：Memory Available MBytes ，Process\Private Bytes和Process\Working Set，PhysicalDisk/%Disk Time  【说明】：  Windows资源监控中，如果Process\Private Bytes计数器和Process\Working Set计数器的值在长时间内持续升高，同时Memory\Available bytes计数器的值持续降低，则很可能存在内存泄漏。内存泄漏应该通过一个**长时间**的，用来研究分析当所有内存都耗尽时，应用程序反应情况的测试来检验。  **CPU分析**  【监控指标】：  System %Processor Time CPU，Processor %Processor Time CPU  Processor%user time 和Processor%Privileged Time  system\Processor Queue Length  Context Switches/sec 和%Privileged Time  【参考值】：  System\%Total processor time不持续超过90%，如果服务器专用于SQL Server，可接受的最大上限是80-85% ，合理使用的范围在60%至70%。  Processor %Processor Time小于75%  system\Processor Queue Length值，小于CPU数量的总数+1  **CPU瓶颈问题**  1、System\%Total processor time如果该值持续超过90%，且伴随处理器阻塞，则说明整个系统面临着处理器方面的瓶颈.  注：在某些多CPU系统中，该数据虽然本身并不大，但CPU之间的负载状况极不均衡，此时也应该视作系统产生了处理器方面的瓶颈.  2、排除内存因素，如果Processor %Processor Time计数器的值比较大，而同时网卡和硬盘的值比较低，那么可以确定CPU 瓶颈。（内存不足时，有点进程会转移到硬盘上去运行，造成性能急剧下降，而且一个缺少内存的系统常常表现出很高的CPU利用率，因为它需要不断的扫描内存，将内存中的页面移到硬盘上。）  **造成高CPU使用率的原因：**  频繁执行程序，复杂运算操作，消耗CPU严重  数据库查询语句复杂，大量的 where 子句，order by， group by 排序等，CPU容易出现瓶颈  内存不足，IO磁盘问题使得CPU的开销增加  **磁盘I/O分析**  【监控指标】：PhysicalDisk/%Disk time，PhysicalDisk/%Idle Time，Physical Disk\ Avg.Disk Queue Length， Disk sec/Transfer  【参考值】：%Disk Time建议阈值90%  Windows资源监控中，如果% Disk Time和Avg.Disk Queue Length的值很高，而Page Reads/sec页面读取操作速率很低，则可能存在磁盘瓶径。  Processor%Privileged Time该参数值一直很高，且如果在 Physical Disk 计数器中，只有%Disk time 比较大，其他值都比较适中，硬盘可能会是瓶颈。若几个值都比较大， 那么硬盘不是瓶颈。若数值持续超过80%，则可能是内存泄露。如果 Physical Disk 计数器的值很高时该计数器的值（Processor%Privileged Time）也一直很高， 则考虑使用速度更快或效率更高的磁盘子系统。  Disk sec/Transfer 一般来说，该数值小于15ms为最好，介于15-30ms之间为良好，30-60ms之间为可以接受，超过60ms则需要考虑更换硬盘或是硬盘的RAID方式了.  Average Transaciton Response Time（事务平均响应时间）随着测试时间的变化，系统处理事务的速度开始逐渐变慢，这说明应用系统随着投产时间的变化，整体性能将会有下降的趋势  Transactions per Second（每秒通过事务数/TPS）当压力加大时，点击率/TPS曲线如果变化缓慢或者有平坦的趋势，很有可能是服务器开始出现瓶颈  Hits per Second（每秒点击次数）通过对查看“每秒点击次数”，可以判断系统是否稳定。系统点击率下降通常表明服务器的响应速度在变慢，需进一步分析，发现系统瓶颈所在。  Throughput（吞吐率）可以依据服务器的吞吐量来评估虚拟用户产生的负载量，以及看出服务器在流量方面的处理能力以及是否存在瓶颈。  Connections（连接数）当连接数到达稳定状态而事务响应时间迅速增大时，添加连接可以使性能得到极大提高（事务响应时间将降低）  Time to First Buffer Breakdown（Over Time）（第一次缓冲时间细分（随时间变化））可以使用该图确定场景或会话步骤运行期间服务器或网络出现问题的时间。  **碰到过的性能问题：**   * 1. 在高并发的情况下，产生的处理失败（比如：数据库连接池过低，服务器连接数超过上限，数据库锁控制考虑不足等） * 2. 内存泄露（比如：在长时间运行下，内存没有正常释放，发生宕机等） * 3. CPU使用偏离（比如：高并发导致CPU使用率过高） * 4. 日志打印过多，服务器无硬盘空间   **如何定位这些性能问题：**  1. **查看系统日志**，日志是定位问题的不二法宝，如果日志记录的全面，很容易通过日志发现问题。  比如，系统宕机时，系统日志打印了某方法执行时抛出out of memory的错误，我们就可以顺藤摸瓜，很快定位到导致内存溢出的问题在哪里。  2. 利用性能监控工具，比如：JAVA开发B/S结构的项目，可以通过JDK自带的Jconsole，或者JProfiler，来监控服务器性能，Jconsole可以远程监控服务器的CPU，内存，线程等状态，并绘制变化曲线图。  利用Spotlight可以监控数据库使用情况。  我们需要关注的性能点有：CPU负载，内存使用率，网络I/O等  3. 工具和日志只是手段，除此之外，还需要设计合理的性能测试场景  具体场景有：性能测试，负载测试，压力测试，稳定性测试，浪涌测试等  好的测试场景，能更加快速的发现瓶颈，定位瓶颈  4. 了解系统参数配置，可以进行后期的性能调优  除此以外，还想说个题外话，就是关于性能测试工具的使用问题  在刚开始用Loadrunner和JMeter的时候，做高并发测试时，都出现过没有把服务器压垮，这两个程序自己先倒下的情况。  如果遇到这个问题，可以通过远程调用多个客户端的服务，分散性能测试工具客户端的压力来解决。  说这个的目的是想说，做性能测试的时候，我们**一定要确保瓶颈不要发生在我们自己的测试脚本和测试工具上。** | |