websphere性能优化:WebSphere Application Server 的应用程序的性能测试规划

疯狂代码 http://www.crazycoder.cn/ ĵ:
http:/www.crazycoder.cn/SoftwareTesting/Article35368.html

在本文中Alexandre Polozoff 提供对基于 WebSphere Application Server 应用进行性能测试规划其中有有关规划、建立性能环境、进行实际测试和测量应用特征信息性能测试是确定应用在各种负载情景中(JVM、连接池等)最佳设置唯方式每个应用是区别且在区别条件下行为也区别这意味着所有应用在生产环境中被实现前必须经历性能测试活动

引言

protocol(规划)词被定义为“科学或医学实验、处理或过程详细规划”本文提供对基于 WebSphere® Application Server 应用进行性能测试规划其中有有关规划、建立性能环境、进行实际测试和测量应用特征信息

性能测试是确定应用在各种负载情景中(JVM、连接池等)最佳设置唯方式每个应用是区别且在区别条件下行为也区别这意味着所有应用在生产环境中被实现前必须经历性能测试活动

性能测试环境

在理想情况下性能测试环境应在每个细节(从服务器防火墙和后端资源数量到网络电缆规格)上完全模仿生产环境然而由于大容量生产环境大小和规模这是不实际包括最少两台或 3台在物理上分离 WebSphere Application Server 机器更小环境是更典型性能测试基本配置

图 1. 基本性能环境

er" height="213" alt="基本性能环境" src="http://www.crazycoder.cn/WebFiles/20092/127e6ee4-3513-475e-bcd7-08185b0ca2ed.g" width="400" />

如图 1 所示分布式空间中基本性能环境有两台连接到远程数据库并由台远程 HTTP 服务器驱动在物理上分离 WebSphere Application Server

如果 HTTP 服务器在应用生产环境中是远程那么最好使性能环境中 HTTP 服务器也是远程每台 WebSphere Application Server 在自己节点上独立运行运行和测试无关其他应用将引入针对本地 CPU、内存和磁盘资源竞争这不仅会影响测试结果而且这些应用和环境资源交互是难以估量

这两台 WebSphere Application Server 至少应该有相同机器和 OS 级别配置常见包括台或另台应用服务器使用区别 OS 补丁或修订包级别或区别内存配置这将导致不致结果和 / 或行为作为题外话请您定要确保每台服务器 TCP/IP 堆栈设置完全相同尤其是 NIC 卡双工设置

在理想情况下虽然应用数据不必驻留在和管理资源库相同数据库服务器上但是管理资源库所有应用数据和数据库将驻留在台远程机器上如果启用了 HTTP 会话持久性请确保会话表和其他数据库分离且被标记为 VOLATILE

这并不是说把完全区别配置用于性能环境是完全不可接受在当今商业环境中HTTP 服务器常常驻留在每个本地 WebSphere Application Server 节点上然而其中台 WebSphere Application Server 还作为管理资源库数据库从而肩负双重责任是不太合乎需要这些配置特征和其他配置特征违反了“应用不应该参和本地资源竞争”约束性能环境中这些不平衡可能使最终结果不准确在许多时候确实如此然而虽然折衷性能测试环境肯定不是更好选择但是拥有几乎任何类型性能测试环境都要比没有环境好

专用服务器环境

显然测试环境应尽可能地和生产环境相似任何差异(任何方面差异)将引入不确定性如果您缩小测试环境那么您必须扩大结果以取得生产环境中近似数字类似地如果 HTTP 服务器在生产中是独立但在测试中被包括在应用服务器节点上那么取得性能结果也不能准确地反映生产中结果配置测试环境是作出选择和让步以获取可能取得最精确数据过程

理想性能测试环境由专用服务器机器和连接它们专用网络组成如果基于 WebSphere Application Server 应用服务器也作为其他无关应用或配置服务器在运行那么进行性能测试是困难您应该将对本地 CPU、内存和磁盘资源竞争减少到最小这点至关重要

也许最大困难是把后端资源专用于性能测试由于在有些安装操作策略和复制成本专用服务器环境并不总是可能这点可以解释为什么人们常常在“夜晚”(此时后端资源处在轻(或更轻)负载条件下)运行性能测试然而因特网世

界正在不断地使后端资源利用率达到极限和此同时跨国公司正向 24/7 运营方向发展

缺乏专用服务器环境可能使性能测试陷于困境非专用服务器环境中些有问题情景包括:

共享网络资源:该组织内部网共享性能环境这可能造成时有时无性能结果这和测试时网络利用率直接有关例如如果有人开始大量利用网络网络备份那么这将增加响应时间甚至完全拒绝性能环境中组件网络连接在这种情况下网络嗅探器可被用来帮助了解网络带宽利用率并确定什么时候负面响应时间不是由于应用造成如果您在这种环境中进行性能测试那么您将遇到很多挫折这种环境效率也很低
共享后端资源:多个应用试图访问相同或相关数据这也可能导致比正常情况更慢响应时间而且如果在测试中没有直接监视后端资源工具帮助来了解利用图这是难以识别情况此外其他应用可能在更改后端资源上数据这将增加重复测试困难甚至不可能进行

运行多个被测试应用 WebSphere Application Server 意味着应用和 HTTP 服务器都受到多个测试额外压力这是个完全合理可运行情景它进行了多个应用性能系统集成测试

WebSphere Application Server 环境基本配置

WebSphere Application Server 环境些基本配置常常被误解或被地配置

分层“门”
种配置是限制从因特网到 WebSphere Application Server 并最终到后端资源请求限制请求原因是针对应用最佳性能特征来配置它负载曲线中有点表示响应时间增加直接原因是应用处理请求数量当您使用这里描述规划来确定了这点后您就可以“限制”应用处理请求数量结果在应用能够处理下个请求前在 HTTP 服务器中应使进入请求排队您可以通过配置基础结构中 3个区别点来实现限制这些点可控制流向下层请求流动和容量

图 2. 限制请求

er" height="259" alt="限制请求" src="http://www.crazycoder.cn/WebFiles/20092/808a5ee1-4fcf-428c-99ad-e622b71c49c6.g" width="475" />

图 2 显示了基础结构中主要组件从左边开始请求从因特网进入 HTTP 服务器然后请求被发送到 WebSphere Application Server在这里应用连接到后端资源(例如数据库)在图中每个箭头(共 3个)中有个配置点这个点可限制进入应用并接着进入后端资源入站请求限制进入环境请求可以对最大工作负载调优并提供良好用户体验

限制规划
这里描述限制规划是基于推荐 WebSphere Application Server 最佳实战、摘自红皮书且被编排过信息和在大容量客户位置所获得经验

根据基础结构中 3个配置点这个规划有 3个步骤:

1. HTTP 服务器并发请求最大个数

WebSphere Application Server 支持 Web 服务器都提供定义可接受并发请求(而不是并发用户)最大个数功能在这里用户和请求区别被指出包括几个图像个 HTML 页面将导致来自个用户多个请求

在 Apache 世界中最大并发请求“门”完全由 MaxClients 设置来控制(在 iPlanet 上相应设置是 ThrottleRequests)对本地静态内容请求由 Web 服务器来处理对应用请求被转发给 WebSphere 插件 WebSphere 插件再把请求向外发送给应用

Web 服务器既处理对本地静态内容请求也处理对应用请求为了确定 MaxClients 值您需要分析发送给浏览器内容和应用服务器中 servlet 引擎线程最大个数对于任何使用台 Web 服务器和台应用服务器应用用于设置值般原则是:

maxClient = imageContentPerPage*maxServletEngineThreads

其中 imageContentPerPage 表示 HTML 响应中图像平均(或最大)个数maxServletEngineThreads 表示为应用服务器定义 servlet 引擎线程最大个数

这个公式外推法是基于 WebSphere Application Server 环境例如请考虑有 3台 Web 服务器为两台应用服务器(它们可以是相同 ServerGroup 中克隆)提供输入情景那么公式将变成:

maxClient = imageContentPerPage*maxServletEngineThreads*numberApplicationClones/numWebServers

如果 maxClient 值太小那么负载测试客户机将遇到连接这是 Web 服务器上可用侦听器太少在这种情况下应使 maxClient 值变大

2. servlet 引擎线程最大个数

您可以通过分析性能测试结果来确定 servlet 引擎线程最大个数不存在用于设置这个最大个数有效“通用”值在通常情况下缺省值 25 对于大容量应用来说太小了

在设置 servlet 引擎线程个数最大值时请记住最大 JVM 堆大小设置数量每个 servlet 引擎线程被分配给它自己堆栈该堆栈将消耗 JVM 中内存由于每个 servlet 线程在负载条件下处理请求您还需考虑在该活动期间创建对象个数请确保 JVM 最大堆大小设置足够大以支持增大 servlet 引擎线程个数并避免内存不足条件另外别忘了您很可能需要启用生成垃圾收集(请参阅 WebSphere Application Server InfoCenter Performance Tuning Guide中定义)

您实际使用最大值应完全由负载和强度测试期间应用监视结果来确定请使用应用监视器来确定是否所有 servlet 引擎线程被建立然后相应地调整最大值并再次运行测试请记住如果您更改了 servlet 引擎线程最大个数那么您还应对 Web 服务器上 MaxClients 参数作相应更改

3. 最大连接池大小

链中最后个“门”是应用访问数据源连接池大小有关连接池文档(请参阅 参考资料)指出应用在执行它们事务时应该短暂地维持数据库连接这将使几个数据库连接高效管理和共享成为可能广泛接受数据源连接最大值是 40即使在大容量安装中也是 40典型应用在 10 至 20 的间

在这里您实际使用最大值完全由负载和强度测试期间应用监视结果来确定应用监视应确定池中被利用数据源连接个数如果最大值设置是 20 且所有 20 个数据源连接在负载下被完全利用那么请把最大值设置增加到 30请再次运行测试、再次分析扩大连接池利用情况并再次确定连接池是否被完全利用请根据需要再次调整最大值

为了发现所需连接真正最大个数作实验是必要当打开连接存在时形式为影响应用性能内存使用和网络利用关联开销也存在虽然确定所需连接个数不是准确科学但是通过反复实验和应用监视您可以找到可能找到最佳值

使期望测试结果相关

测试结果应该显示负载测试客户机端所看到数字和应用服务器上数字的间有某些相关性例如如果台应用服务器3个克隆被配置成每台服务器 25 个 servlet 引擎线程并且注意到 servlet 响应时间小于秒那么您应期望看到至

少每秒 75 或更多个请求通过应用服务器您还应期望在客户机端看不到 8 秒响应时间结果必须同时考虑用于静态数据 HTTP 连接数量请确保测试结果和您在应用服务器配置中看到东西相匹配如果结果不匹配请确保负载测试客户机没有在运行时发出太多热量(100% CPU)、出现内存不够或遇到某些网络瓶颈

负载测试客户机

性能测试环境客户机端对性能测试结果有重大影响传统负载测试工具作为代理在几台客户机机器上运行您需要不止台客户机机器来生成有代表性容量负载这是 CPU 和内存限制了同时收集准确结果客户机所能真实代表用户数量

图 3. 负载测试客户机使容量流向服务器环境

er" height="331" alt="客户机使容量流向服务器"
src="http://www.crazycoder.cn/WebFiles/20092/0ccbead1-b409-4d20-be24-696a967be63b.g"
width="439" />

在图 3 中我们 3台负载测试客户机正把负载应用到性能测试环境中服务器请注意负载测试客户机没有驻留在应用服务器上客户机应该总是位于和应用服务器区别机器上

专用客户机

负载测试客户机机器应被完全专用于负载测试任务竞争本地 CPU、内存和磁盘资源其他应用不能共享这些机器这种对本地资源竞争确实会影响被测量响应可靠性客户机机器应保持在相同路由器和网络配置上并尽可能地(在网络连接意义上)接近专用服务器环境

还有负载测试客户机必须能够生成装入涉及应用所需正确类型请求例如基于对 EJB RMI 访问应用区别于测试对 JSP 或 servlet 框架基于 HTTP 请求

测试基线和致性

专用负载测试机器在区别性能测试运行的间提供了致性在运行于 WebSphere Application Server 应用性能测试规划第步中有步是记录组结果基线仅当整个测试情景可被致地再次产生时(也就是说您总是可以再次运行基线并获取相同结果)基线才是有关偶然交换负载测试机器或它们配置将增加完成“基线”任务困难(甚至无法完成该任务)此外这至少会使性能数据分析含糊不清

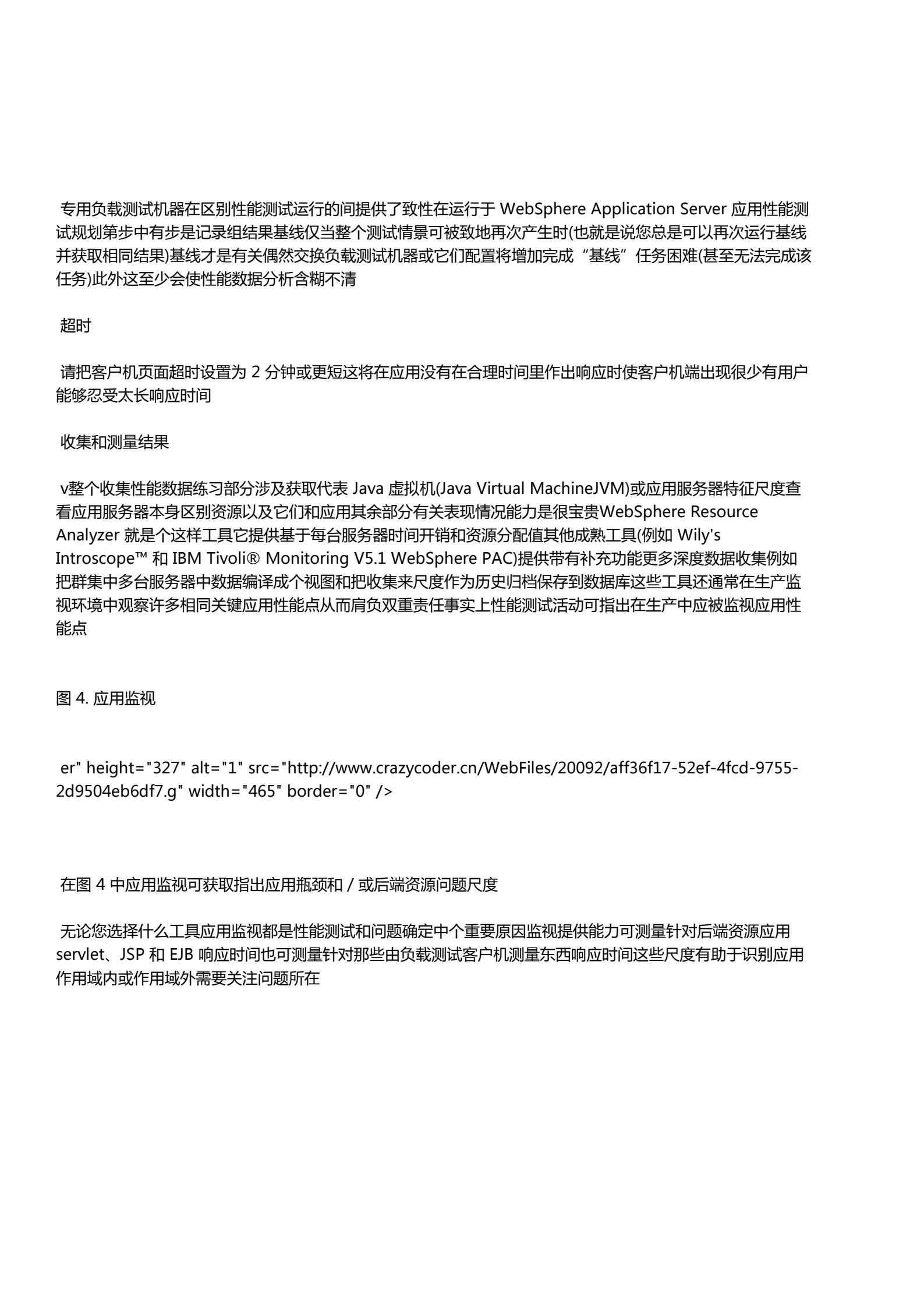
超时

请把客户机页面超时设置为 2 分钟或更短这将在应用没有在合理时间里作出响应时使客户机端出现很少有用户能够忍受太长响应时间

收集和测量结果

v整个收集性能数据练习部分涉及获取代表 Java 虚拟机(Java Virtual MachineJVM)或应用服务器特征尺度查看应用服务器本身区别资源以及它们和应用其余部分有关表现情况能力是很宝贵WebSphere Resource Analyzer 就是个这样工具它提供基于每台服务器时间开销和资源分配值其他成熟工具(例如 Wily's Introscope™ 和 IBM Tivoli® Monitoring V5.1 WebSphere PAC)提供带有补充功能更多深度数据收集例如把群集中多台服务器中数据编译成个视图和把收集来尺度作为历史归档保存到数据库这些工具还通常在生产监视环境中观察许多相同关键应用性能点从而肩负双重责任事实上性能测试活动可指出在生产中应被监视应用性能点

图 4. 应用监视



在图 4 中应用监视可获取指出应用瓶颈和 / 或后端资源问题尺度

无论您选择什么工具应用监视都是性能测试和问题确定中个重要原因监视提供能力可测量针对后端资源应用 servlet、JSP 和 EJB 响应时间也可测量针对那些由负载测试客户机测量东西响应时间这些尺度有助于识别应用作用域内或作用域外需要关注问题所在

历史归档

随着测试进行取得测量结果应该和以前运行结果相比较以确定性能是提高了还是下降了应用监视工具通常可以用种格式或另种格式来保存被收集数据更高端工具甚至可以把结果直接集成到数据库或其他数据存储库如果您使用工具不提供用可用格式来保存数据思路方法那么您可以把相关数据保存在简单电子表格中

时间开销

应用开发小组倾向于通过某种记录机制把性能时间开销测量构建到他们应用中在应用中记录性能测量结果作法应被尽力劝阻这将在应用中加入更多代码这些代码不仅消耗处理周期还必须被维护和测试最终固定在应用上并可由个命令来控制外部应用监视工具效率更高

监视 JVM

为了更好地理解 JVM 是如何运行您需要在性能测试活动中监视 JVM 几个重要地方些需要观察基本参数包括:

活动 servlet 引擎线程个数:这使您能够理解 JVM 中应用在任时刻可以提供最大工作量它也有助于确定用于生产环境所需最大设置
活动 ORB 线程个数:用于有 EJB 应用
可用和用去内存:有助于理解应用是如何利用内存和垃圾收集周期执行频率
servlet 响应时间:有助于比较应用服务器上观察到响应时间和负载测试客户机上测得响应时间的间异同通过防火墙和其他网络组件瓶颈可能带来被记录 servlet 响应时间所证实延迟

监视应用

应用监视是特定于每个应用您应该考虑监视访问后端资源重要思路方法为了理解序列化 / 反序列化对象思路方法对性能影响您必须观察这些思路方法

频率和思路方法执行持续时间是两个应该被获取性能数字频率需被监视以确定在区别性能运行期间思路方法被次数思路方法执行时间开销使您深入了解在某些任务上花费响应时间占全部响应时间百分比也有助于指出应用瓶颈开发者可以把这些信息用于进步代码重构以提高应用性能

监视后端资源

性能测试期间后端资源监视是性能测试活动至关重要部分任何应用性能和环境中最慢链路样快如果后端资源没有提供足够级别性能那么访问后端资源应用也不在最佳级别运行后端资源性能下降原因可能有很多您需要让每个特定资源管理员使用正确工具集和知识以帮助解决这种有关后端任何问题

监视网络资源

网络连接应用环境常见链路是网络本身参和测试活动网络资源也必须被监视它们数据必须被分析以确保最大吞吐量和效率这包括整个网络(包括参和测试任何防火墙、路由器、CSS 交换机、负载平衡器、反向代理等)您必须首先解决由于配置防火墙、反向代理或其他网络设备所造成任何性能问题否则被收集数据总是不准确和 / 或不致

需检查常见网络配置包括:

被设置为半双工而不是全双工吞吐量
在往返于相同组设备时使用区别跳点路由
安装用于代理而不是通行防火墙

应被监视网络资源:

CPU(如果适用话)
处于“建立”状态端口
连接吞吐量时间开销
带宽
设置性能期望

在性能测试开始前就从几个区别角度设置期望对于确保测试结果是有价值且测试活动本身是成功有很重要意义这些期望包括测试小组需要哪些区别输入和哪些输出将定义合理应用性能如果您没有提前设置那么您很难定义被测试应用性能是否是足够好

同样您必须设置性能测试活动预期持续时间由于测试和被修改参数数量较多性能测试需花去很长时间才能完成提前安排适量测试时间将使每个人受益

应用期望

为了确定应用性能是否是足够好您必须定义性能期望列出以下期望应该是任何测试计划第部分:

可接受 servlet 响应时间
可接受负载客户机响应时间由于网络开销、通过 Web 服务器和防火墙额外跳点等它区别于 servlet 响应时间
可接受每秒请求吞吐量
可接受后端资源响应时间
可接受每秒后端请求吞吐量
可接受网络开销(包括 Web 服务器、防火墙、反向代理、负载平衡器、CSS 等)

同样如果测试结果没有满足可接受结果标准那么您必须制定处理任何缺点计划缓解应用中性能瓶颈基本策略有两条:

扔掉更多有问题硬件这可能很贵
排除应用瓶颈这可能很费时

虽然预算限制常常是决定中个原因但是根据具体问题两种策略可能都是正确和可行任个解决方案都不便宜但是般来说排除应用瓶颈是更好策略故您应尽可能采用它解决应用中性能问题还将导致某种类型事后分析过程(即记录并传播从这些任务中学到知识)

性能测试活动总共持续时间

不幸是人们常常在开发周期结束前(把应用转移到生产环境前)安排几周性能测试这种思想问题是直到给应用施加负载时许多应用问题才会暴露出来而且只有在性能测试阶段中应用才被施加生产环境预期负载量

即使应用问题很少开发周期性能测试阶段可能需要几个月才能完成这取决于应用、环境和许多可变原因有更多问题和应用将花去多得多时间这是我强烈建议您尽早在性能测试环境中测试个原因在开发周期中越早开始测试您就能越早地检测到应用问题并正确地处理它如果您等到开发周期快结束时才开始负载测试那么您很可能遇到最坏性能测试情景

应用接受标准

在任何认真软件Software开发活动后在性能测试活动开始前应用必须达到性能测试接受标准如果应用没有达到这些标准那么它不应该被性能测试环境所接受:

单元测试能力所有应用开发工作必须提供全面单元测试策略和附带可被执行单元测试代码以确定构建是完整和有效如果单元测试案例应用中或缺少单元测试代码而无法被成功完成那么应用不应该被性能测试所接受
低负载级别能力在单用户和 10 个用户负载级别上应用应该用去正常计算时间来达到合理性能和预先定义期望

如果应用无法在低负载级别正常运行那么它肯定无法在更高负载级别正常运行开始性能测试将会浪费时间
可用测试数据在性能测试期间执行应用所需数据必须被提供或被详细描述以使性能测试小组能够建立和生产环境尽可能接近副本测试数据必须是真实、致和完整

应用在以预期行为通过性能测试前绝不能被用于生产环境

测试规划

当性能测试环境被建立且在最佳状态运行时下步是制定测量被测试应用性能和特征详细测试计划下面提供信息可被用作应完成任务和应进行测量清单许多规划推荐还将指导您如何完成容量计划步骤这些步骤能使您了解必须定义多少 JVM 才能处理生产中全部预期用户负载

个别测试持续时间

般来说对于个别测试您应该只在最大负载级别上运行下面描述个别测试和配置点运行时间最多为 10 至 30 分钟第组测试运行应严格被用于在各种配置点和负载级别上收集数据在分析了结果并确定了应用定义预期最佳配置点后您就可以执行更长时间测试(持续时间为 12 小时或更长)以测量长期运行应用特征更长时间测试还提供在指定负载下功能和稳定性测试

单个 JVM 和多个 JVM 测量

性能测试有两个基本设置:单个 JVM 和多个 JVM单个 JVM 测试在单独运行时可介绍说明基本应用性能些测量值(例如每秒响应数量、servlet 响应时间和后端访问数量)可提供预期最大吞吐量这些都是最大值群集环境性能可能因后端资源限制而难以超过单个应用性能群集环境提供可伸缩性和故障转移在极少数情况下在多个 JVM 配置中可能出现在单个 JVM 配置中不可能出现应用问题

推荐:在所有测试中您必须既使用单个 JVM 配置也使用群集多个 JVM 配置

JVM 堆大小设置

您可以使用组从最小到最大值来调整 JVM 堆大小设置直到找到最佳设置servlet 引擎线程数量设置将在某种程度上决定最佳设置这是每个堆栈都使用 JVM 中内存

推荐:通过合理地增加 JVM 堆大小设置来对它进行调整以确定应用最佳内存设置请确保 JVM 堆大小设置和同台服务器上任何其他应用不超出机器物理内存限制请记住基本操作系统也有必须被考虑内存要求在所有操作系统环境中内存交换会对应用性能产生负面影响

生成垃圾收集

针对所有操作系统平台生成垃圾收集(generational garbage collection)概念在 JVM V1.3 中被引入般来说启用生成垃圾收集将使积极创建许多临时对象并运行有大量用户大容量应用受益然而可能是这种情况也可能不是这取决于应用是如何使用对象

请注意在性能测试期间出现主要垃圾收集周期数量并理解在这些周期中对应用性能含义将导致更小 JVM 大小物理内存限制常常使您看到在负载下更高收集率理解这里性能影响对于在生产环境中避免意外有重要意义和流行看法相反垃圾收集不会导致应用服务器丢失请求

推荐:对于每组 JVM 堆大小设置请在次测试运行中启用垃圾收集在另次中禁用它请您花足够时间来运行这些测试以确保至少执行几个垃圾收集周期从而可以测量应用行为请通过观察 JVM 如何利用可用内存和用去内存来监视垃圾收集周期

servlet 引擎线程池

servlet 引擎线程池大小决定了包含 Web 应用 JVM 次可执行工作量在 WebSphere Application Server V4.0 和更新版本中线程池定义了反映池大小限制最小值和最大值般来说大容量应用有较大线程池大小但是有些原因(例如应用瓶颈、synchronized 关键字使用和 / 或代码中其他限制)将阻碍大线程池高效利用

利用 servlet 引擎线程池应用往往是使用 servlet 和 JSP 应用这也包括基于 SOAP 应用直接和 EJB 交谈应用客户机不使用 servlet 线程但确使用 ORB 线程(请看下部分)

推荐:请用区别最小和最大线程池大小来测试应用以确定哪些设置使应用完成尽可能多工作般来说为了使线程池大小更大您可以上调 JVM 堆大小设置但在开始测试时请使用最小内存设置请监视应用对 CPU 利用率旦应用对 CPU 利用率接近 80%您将遇到 CPU 极限必须为基本操作系统本身保留周期

别忘了 servlet 引擎线程池大小直接影响 Web 服务器上 MaxClient 设置请确保 Web 服务器有足够已定义侦听器以避免在负载测试客户机端出现“连接失败”如果确实出现了“连接失败”那么 Web 服务器侦听器最大数量设置太低了您必须增加它

ORB 线程池大小

运行于容器中 EJB 在被分配到 ORB 线程池线程中执行和 EJB 通信应用(这些应用基于远程思路方法(Remote Method InvocationRMI)和 servlet)必须使 ORB 线程池大小被配置

推荐:请调整 ORB 线程池大小请监视线程池和 EJB 活动 / 响应时间以确定最佳线程池大小

连接池

使用有连接池 JDBC 资源应用需要设置连接池大小广泛接受实战把池中最大连接数限制在 30-40 左右太多连接(连接数超过 40)不会给您带来什么好处您应该记住每个被建立连接消耗内存和网络资源

会话持久数据源般只需把最大值定义为 10 个连接绝不要使最大连接池大小大于 servlet 引擎线程数量

推荐:请调整连接池最大值和最小值请监视 servlet 响应时间和吞吐量(请求 / 秒)、JDBC 活动和内存利用情况以达到最佳性能

用户

您需要知道准确代表生产中应用行为性能测试活动用户类型和数量这点很重要区别组织用区别思路方法来测量用户和预期负载的间相关性有些组织通过每秒 CICS 事务数量(而不是用户数量)来测量利用率为了使用户负载和正确尺度相关您需要理解应用是如何利用后端资源这实际上是道数学练习题无论负载是如何被测量大家对最后结果和它们表示含义都应该有个共同理解

用户数量

用户数量定义了应用所受某个负载每个应用很可能有区别类型用户(例如浏览目录用户和买东西购物者)所以您必须使用各种情景(全部这些情景类似于平均、预期工作负载)来测试应用

有些应用在高用户负载时性能好于其他应用有瓶颈问题或过多同步应用常常表现为较长响应时间和较低 CPU 利用率

单用户负载测试常常被用来建立应用基线性能前面已讲过如果应用在单用户负载级别上表现不佳或崩溃那么继续在更高负载级别上进行应用性能测试是没用同样如果应用在 10 个用户级别上表现不佳那么我建议您终止测试

旦应用服务器上 CPU 利用率接近并达到 100% 分用户数量增加只会导致更长响应时间这应该被测量和记录它

明确定义了应用限制这也有助于容量规划和确定群集环境中所需应用克隆数量以支持预期负载

用户类型

个应用常常有几个类型用户例如个卖东西 Web 站点至少有两种类型用户:浏览者和购物者种类型用户和该站点交互但什么也不买另种用户执行额外功能(例如购物和信用卡验证)并不是所有用户都是购物者但所有用户至少都是浏览者定义参和测试用户类型是理解站点性能关键如果您预先更多地了解使用站点各种用户那么您所进行性能测试将更为逼真同样理解常被使用页面流动也是有益这就要求负载测试脚本开发者理解应用、用户类型和这些用户正常行为

推荐:请使用前面所有规划推荐在以下用户负载级别上测试应用:1101005008001500 等(是否合适取决于现实负载预期)请监视应用响应时间、吞吐量(请求 / 秒)、CPU 利用率、JVM 内存利用率、线程利用率和 JDBC 后端利用率

CPU 利用率

为了理解应用负载对被测试应用影响您需要记录应用服务器机器 CPU 利用率服务器目在于使它达到可能最高 CPU 利用率如果昂贵服务器在运行时 CPU 利用率仅为 20%那么它性价比不高在分布式环境中CPU 利用率必须在预期最大负载、硬件故障和已定义服务质量要求的间平衡

图 5. CPU 和响应时间
er" height="149" alt="CPU 和响应时间" src="http://www.crazycoder.cn/WebFiles/20092/942c8cf5-e079-4641-8247-5faa5a2532a0.g" width="199" />

图 5 中图表用蓝色显示了 CPU 利用率用红色显示了响应时间旦对 CPU 利用达到饱和增加负载只会增加响应时间您必须比较测量值和预先定义应用期望以确定在哪些负载级别上可接受响应时间才是可能如果应用有瓶颈或其他问题这个级别对应 CPU 利用率可能明显更低

请测量应用使服务器达到饱和状态时用户负载级别并记录它这些数字对容量规划演习有用

记录并评估结果

基于以上规划推荐各种测试执行涉及对应保留记录活动只有记录了数据才能进行分析保留记录是性能测试活动中重要最后步

表 1. 样本电子表格收集了基于规划推荐某次运行结果

日期 / 时间 2002 年 8 月 6 日 — 12:03:54
用户数量 100
测试持续时间(分钟) 10
Web 服务器设置
MaxClients 600
TTL(每个进程请求数总和) 10,000
WebSphere Application Server 设置
JVM 堆大小最小值 256
JVM 堆大小最大值 512
生成垃圾收集 有关 -XX 设置
servlet 引擎线程池大小最小值 10
servlet 引擎线程池大小最大值 200
ORB 线程池大小最小值 10
ORB 线程池大小最大值 200
Web 服务器测量值
CPU 利用率(在最大时测量) 40-45 %
平均吞吐量 0.45 秒
每秒请求数 44.85
负载客户机测量值
每秒请求数 44.85
请求数总和 7,348
成功完成请求数 7,342
超时请求数 4
连接失败请求数 2
第 1 个页面响应时间 0.35 秒
第 2 个页面响应时间 0.54 秒
第 3 个页面响应时间 1.33 秒
WebSphere Application Server 测量值
servlet 响应时间 0.28 秒

垃圾收集周期数量 2
CPU 利用率(在最大时测量) 33%
JDBC 每秒请求数 -- 查询 125
JDBC 响应时间 -- 查询 37 ms
JDBC 每秒请求数 -- 插入 35
JDBC 响应时间 -- 插入 125 ms
JDBC 每秒请求数 -- 更新 98
JDBC 响应时间 -- 更新 222 ms

基于本文中给出规划推荐组测试结果和表 1 中列出数据类似其中记录了配置各种元素和测得结果对于每个个别测试系列您都需要这样做组收集到数据依赖于被应用所利用后端资源不直接访问 JDBC 资源应用显然不会收集 JDBC 时间开销数据您应该针对后端资源来收集基本时间开销和频率数据

旦结果被收集并编译后它们可被生成各种图表以介绍说明应用性能结果收集和编译可能涉及些对数据手工处理没有个工具能够收集所有所需数据

图 6. 按用户数量来显示完成请求数
er" height="183" alt="按用户数量来显示完成请求数"
src="http://www.crazycoder.cn/WebFiles/20092/ec9cb9f7-3e11-4e92-ab11-b6a94457d19a.g"
width="297" />

图 7. 按页面和用户数量来显示响应时间

er" height="183" alt="按页面和用户数量来显示响应时间"
src="http://www.crazycoder.cn/WebFiles/20092/b145c848-d24d-44e3-8ce3-f51509c869f7.g"
width="298" />

图 6 和图 7 显示了在把测试结果合并成个电子表格后可被创建两个可能图表在图形格式中结果可以对地被并排放置以使其他人能够分析结果您应该提供介绍说明任何测量到或观察到异常文本当性能结果显著提高或下降时您更应该这样做

(别忘了手工维护电子表格或数据库中可能存在人犯下为了找出可能或表示您应该在数据被记录后审阅下数据)

数据分析涉及找出应用以下特征以确定各种最佳值:

servlet 响应时间和客户机响应时间
网络带宽
应用服务器和 Web 服务器 CPU 利用率
后端资源利用率
网络组件利用率

旦确定了最佳设置您必须考虑应用是应该在自己应用服务器上运行还是和其他应用起在应用服务器上运行您还必须考虑在相同物理机器上运行应用服务器数量很可能有物理资源限制最后对于群集环境您不仅必须评估应用服务器机器上物理限制还必须评估被应用所利用后端资源所有这些原因将适用于生产环境性能和容量规划阶段从而达到最佳性能

性能测试频繁程度

个常见误解是性能测试是次性工作如果您从不更改应用代码或机器配置那么性能测试是次性工作然而经常出现情况是应用代码被更改或更好机器配置被用于生产环境或这两种情况都发生了因此每当应用代码或机器配置被更改时您必须再次执行性能测试以确定可达到最佳性能新参数

结束语

性能测试规划是定义、理解和执行全面组合在开发周期中尽早地、尽可能多地测试是在把应用发布到生产环境前确定性能问题最佳方式专门负责性能测试小组可提供致测试并使个别开发小组不必再肩负性能测试责任般来说使您深入了解 JVM 和应用特征工具是物有所值它们使您快速确定问题并使性能测试活动尽可能快速和高效

2009-2-12 5:21:04
疯狂代码 http://www.crazycoder.cn/