动态 Web 技术架构及其性能分析①

Dynamic Web Architectures and their Performance Analysis

覃雄派 （中国人民大学 信息学院 计算机系 北京 100872）

摘　要：本文对三项动态 Web 技术进行了性能研究，包括 PHP、Java Servlet、以及 EJB。用 TPC－W 负载对三种技术架构进行了性能测试。实验结果显示，对于中小规模的动态 Web 系统，PHP 技术表现出良好的性能，Java Servlet 技术在性能上比 PHP 稍微差一些，但是当数据库加锁冲突加剧的时候，可以利用 Java 的同步机制减少这种冲突，提高系统吞吐量，另外，当 Web 服务器成为系统瓶颈时，可以把 Servlet 执行引擎迁移到其它机器，通过负载均衡达到提高性能的目的。EJB 技术在所有的测试中，都获得最差的性能，但是通过不同版本的 EJB 服务器的性能比较，我们发现新版本的 EJB 容器在数据库存取上做了大量优化，获得了可观的性能提高，EJB 技术具有丰富的基础服务，其架构灵活，逐渐成为企业计算的现实选择。

关键词：动态 Web 应用　系统架构　性能分析　PHP　Java Servlet　EJB

1　前言

Web 技术已经由静止 Web 发展到动态 Web 阶段，并且正在向交互的、社区的、和智能的 Web 阶段迈进。Web 的内容不仅仅是静止的 HTML 页面和图像文件，更多的 Web 页面以动态方式生成。动态 Web 内容的生成需要 Web 服务器、动态内容生成器、数据库系统的支持，其基本工作原理是，Web 服务器提供静态的内容，并且把动态内容的请求转交给动态内容生成器，而动态内容生成器根据用户的请求，执行商业过程，从数据库系统查询相关的信息，按照特定的格式把查询结果组装到页面里，把页面返回给 Web 服务器，由之返回给用户。

2　技术架构比较

动态 Web 系统的核心模块是动态 Web 内容生成器，所有的商业逻辑以及页面的组装工作，由之完成。本文通过实验，比较 PHP、Java Servlet、EJB 等三项具有代表性的动态 Web 技术的性能。

PHP[1] 是对 HTML 标记语言进行扩展的脚本语言，可以直接嵌入到 HTML 网页中。PHP 的执行程序作为 Web 服务器的动态模块，由 Web 服务器加载和运行，用以解释执行 PHP 脚本语言。当 Web 服务器接收到用户的页面请求，并且发现页面里面包含 PHP 脚本时，则解释执行该 PHP 脚本，并且把执行结果组装到页面里，返回给用户。PHP 脚本可以执行一般的商业逻辑，此外，可以通过嵌入 SQL 语言的方式，对数据库进行存取。PHP 模块在 Web 服务器的进程空间里运行，Web 服务器和 PHP 模块的信息交换通过内存的共享实现，不需要进行进程间的通讯，减少了运行开销。PHP 技术的主要缺点是，由于 SQL 语句嵌入在脚本中，PHP 系统的开发和维护成本因为客户表现层和商业逻辑的互相缠绕而变的困难。

Java Servlet 一般是 HTTP Servlet[2] [3]，是一种特殊的 Java 类，Servlet 执行引擎根据 Web 服务器的请求，动态地装载 Servlet Class，并且在 Java 虚拟机里运行。Servlet 可以通过标准的 JDBC 接口，查询数据库系统，把信息查询出来，组装成 HTML 页面，由 Web 服务器返回给用户。可以把 Servlet 引擎部署到不同的机器上，达到负载平衡的效果，提高系统的吞吐量。Servlet 的优点是，可以充分地利用 Java 语言的特性，比如通过 JDBC 进行数据库的查询，还可以利用 Java 语言的同步机制，协调用户请求之间的冲突，减少数据库系统的

① 基金项目 国家自然科学基金项目(60496325)，国家自然科学基金项目(60503038)

加锁冲突,提高系统的吞吐能力。

EJB(Enterprise Java Bean)技术是面向企业计算的中间件软件平台。EJB 服务器提供了企业计算所需要的基本服务,包括数据库服务(JDBC)、事务处理服务(JTA)、消息服务(JMS)、名字服务(JNDI)、和基础设施管理服务(JMX)等。EJB 服务器提供一个 EJB 容器,负责进行 EJB 的生命周期管理、会话管理、数据库连接池管理、持久化工作、事务管理、以及包括认证和授权等的安全管理工作。EJB 有三种类型,包括实体 Bean、会话 Bean、以及消息处理 Bean。基于 EJB 技术的动态页面生成过程是,客户发出请求给 HTTP 服务器,HTTP 服务器通过 AJP 协议与 Servlet 引擎进行通讯,Servlet 引擎启动某个 Servlet 进行服务,这个 Servlet 则通过 RMI 协议调用 EJB 容器里的会话 Bean 和实体 Bean,通过会话 Bean 执行商业逻辑,通过实体 Bean 存取数据库,查询结果组装成 HTML 页面,返回给用户。

3 测试基准

为了对 PHP、Java Servlet、EJB 等三类动态 Web 技术架构进行性能分析,我们选择基于 TPC – W 标准的在线书店系统[4] 作为测试的基准。

在线书店系统的数据库包括 8 个数据表,包括 Customer、Address、Orders、Order_Line、Credit_Info、Items、Author、以及 Countries 等。Orders、Order_Line、Credit_Info 三张表保存定单的详细信息。Items 表保存每类书籍的信息,而 Authors 表保存作者的信息。客户的信息保存在 Customers 表和 Address 表中。TPC – W 标准规定了 14 个不同的事务,包括 6 个只读事务和 8 个更新事务。只读事务包括 :回到书店主页、查看新产品信息、查看畅销书列表、查看书籍的详细信息以及两个关键字搜索事务。更新事务包括 :注册用户、更新购物框、两个购买类事务和定单的查询和显示、以及两个管理类事务。TPC – W 对负载进行了分类,其区别在于只读事务和更新事务的比例关系。Browsing 负载类型,只读事务占 95% ,Shopping 负载类型,只读事务占 80% ,Order 负载类型,只读事务占 50% 。为了找出系统可能存在的性能瓶颈,获得系统优化的方案,我们增加一种负载类型 Short Query Dominated,以模拟短查询密集的应用场景。

4 实验与结果

4.1 实验环境

实验的硬件包括 4 台 PC 服务器,其基本配置是:Pentium – 4 2. 8GHz CPU,1GB 的内存,160GB 的 Seagate 7200.10 硬盘。客户端软件运行在 8 台普通的 PC 上。所有机器通过 100M 快速以太交换网络进行连接。我们使用 Apache V1.3.22 作为 Web 服务器,PHP 模块的版本是 4.0.6。Web 服务器的进程数设置为 512,以同时支持大量用户的请求。Servlet 执行引擎采用的是 Jakata Tomcat V5.0.25,在 Sun JDK V1.4 上运行。EJB 服务器采用的是 Jboss3.2.6(前三个实验)和 Jboss V4.0(最后一个实验)。数据库系统采用的是 My SQL V5.0.27,Java Servlet 和 EJB 通过 MM My SQL V2.04 Type4 JDBC 驱动程序进行数据库操作。我们往数据库添加了 10,000 本书籍(Items)和 288,000 个客户(Customers),数据库的大小为 350MB,图像文件存放在文件系统中,占用 183MB 的磁盘空间。

4.2 应用逻辑的实现

在 PHP 和 Java Servlet 实现方案中,为了保证性能比较的公平性,我们采用同样的 SQL 语句。而在 EJB 实现方案中,我们把显示逻辑和商业逻辑切分开,Java Servlet 在系统里的功能是调用 EJB,取得查询结果,生成 HTML。整个系统的架构符合 Session Fa ? ade Pattern[5] 设计模式,主要的商业逻辑通过无状态会话 Bean 来实现,这些无状态会话 Bean 存取实体 Bean,通过实体 Bean 来存取数据库系统。

4.3 配置模式与测试方法

系统的配置方式主要有 WebPHP – DB、WebServlet – DB、Web – Servlet – DB、Web – Serlet – EJB – DB 等几种模式,在所有的配置中,数据库系统运行在单独的服务器上。(1)在 WebPHP – DB 配置中,PHP 模块和 Web 服务器在一个服务器上运行。(2)针对 Java Serlet 技术,我们以两种配置类型来运行测试,分别是 WebServlet – DB 和 Web – Servlet – DB,在 WebServlet – DB 中,Servlet 引擎和 Web 服务器在一个机器上,而在 Web – Servlet – DB 中,Web 服务器和 Servlet 引擎分别运行在不同的机器上。(3)对于 EJB 技术,我们使用四台服务器分别运行 Web 服务器、Servlet 引擎、EJB 容器、和数

据库系统。(4)为了考察 Java 同步机制对性能的影响,我们设计了两种衍生的配置:WebServlet – DB(Sync)和 Web – Servlet – DB(Sync),分别对应 WebServlet – DB 和 Web – Servlet – DB,区别在于前者使用 Java 同步机制进行用户访问的协调,减少数据库的加锁冲突。

测试过程分为三个阶段,分别是预热阶段(Ramp Up Phase)、度量阶段(Measurement Phase)和关闭阶段(Ramp down Phase)。在预热阶段,需要往数据库导入初始数据集,把系统运行到一个稳定的吞吐量水平。接着进入度量阶段,我们度量的性能指标是每分钟交互数(Number of Interactions Per Minute)。关闭阶段保证正在运行的请求执行完毕。三个阶段的运行时间分别是 1 分钟、20 分种和 1 分种。实验过程中,我们用 Sys Stat 实用程序收集不同机器的 CPU、内存、网络、和磁盘的负载状况。

4.4 实验结果与讨论

Shopping 负载类型是电子商务系统最典型的负载,其实验结果如图 1 所示。从图 1 可以看出,WebPHP – DB 的峰值吞吐量为 530 Ipm(Interactions per Minute),当并发的客户数量继续上升,负载加重,系统的吞吐量开始下降,其原因是数据库的加锁冲突开始加剧。WebServlet – DB、以及 Web – Servlet – DB 的系统吞吐量和 WebPHP – DB 大致一样,这是因为这三种配置中,对数据库的查询是一样的。Web – Servlet – DB 把 Servlet 引擎转移到另外一台服务器上运行,但是没有带来性能的提高,其原因在于,数据库加锁冲突成为系统的瓶颈,所以独立的 Servlet 引擎并不能够提高系统性能。

WebServlet – DB(Sync)、以及 Web – Servlet – DB(Sync)两个配置都利用 Java 的同步机制减少数据库的加锁冲突,数据库系统的性能得到了发挥,整个系统的吞吐量有了大幅度的提高,其峰值吞吐量分别是 667 Ipm 和 669 Ipm。

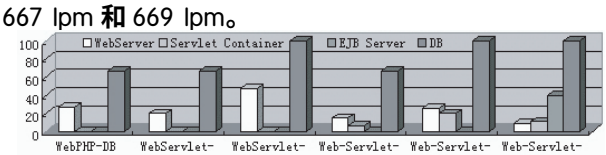

图 2　服务器的 CPU 利用率(Shopping 负载)

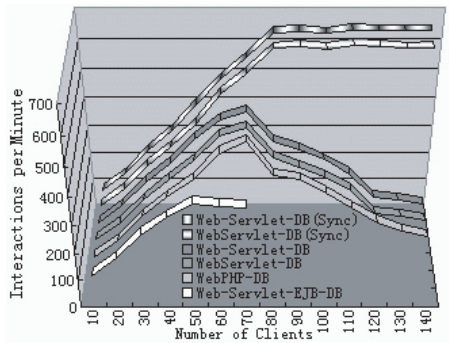

图 1　不同技术架构的吞吐量(Shopping 负载)

Web – Servlet – EJB – DB 配置的系统性能是最低的,这是由于 EJB 容器需要运行很多的短事务,来维护 Bean 的状态,导致系统的总体吞吐量下降。

图 2 是不同配置下系统运行过程中各个服务器的 CPU 平均利用率。从图 2 可以看出,WebPHP – DB、WebServlet – DB、Web – Servlet – DB 三个配置中,数据库服务器的 CPU 利用率在 60 – 70% 左右,而 Web 服务器的 CPU 利用率就更低了,这是因为数据库系统的加锁冲突,导致事务在等待锁的授予,CPU 没有足够的工作可以做。而 WebServlet – DB(Sync)、以及 Web – Servlet – DB(Sync)两个配置中,数据库服务器的 CPU 利用率将近 98%,这是因为 Servlet 利用 Java 同步机制,负担了用户之间的数据访问协调问题,数据库的加锁冲突大幅度减少,CPU 被充分地利用起来。Web – Servlet – EJB – DB 配置中,数据库服务器的 CPU 利用率也达到 100% 左右,但是其性能却是最低的。我们的分析是,为了维护内存里 Bean 的状态和数据库里数据行的同步,EJB 容器执行了大量的短更新事务,比如要修改一个实体的若干个属性,则执行多个 SQL 更新语句,每个 SQL 语句只负责更新某个实体的一个属性。Ordering 负载类型下,各个测试配置的系统吞吐量,其相对关系和 Shopping 负载类型下的表现是类似的。

Browsing 负载类型下,各个测试配置的系统吞吐量,如图 3 所示。所有配置的吞吐量,都比 Shopping 负载类型低,这是因为只读事务比例占更高的比重,而只读事务相对来说,比更新事务要复杂,执行时间更长。在所有的配置中,数据库服务器的加锁冲突并不严重,因为大量的事务是只读型事务,所以即使利用 Java 的同步机制来改善数据库服务器的加锁冲突,也

没有收到预期的性能提升。除了 Web – Servlet – EJB – DB 配置之外,其它配置的系统吞吐量是大致相同的,而 Web – Servlet – EJB – DB 配置的性能仍然是最低的。图 4 是各个服务器在不同配置下的 CPU 利用率,可以看出,数据库服务器的 CPU 成为了系统的性能瓶颈。

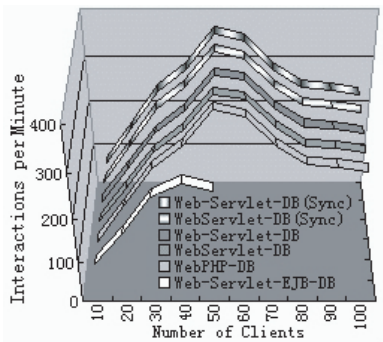

图 3　不同技术架构的吞吐量(Browsing 负载)

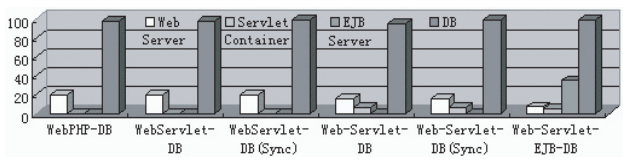

图 4　服务器的 CPU 利用率(Browsing 负载)

为了观察不同配置在大量短事务下的性能表现,我们在 TPC – W 三种负载类型之外,通过裁剪其事务混合比例,组成新的负载类型,称为 Short Query Dominated 负载类型,在这种负载类型中,查询事务占 80%,更新事务占 20%,所有的查询都是短查询,也就是查询的数据量比较小,执行时间短。在这种负载类型下,不同配置所表现的性能又有新的变化,如图 5 所示。为了比较不同 EJB 容器版本的性能,在本测试中,采用 Jboss4.0。

我们首先比较 WebPHP – DB、WebServlet – DB、Web – Servlet – DB 三种架构的系统吞吐量。WebPHP – DB 的峰值系统吞吐量为 995 Ipm,并发用户数为 110。WebServlet – DB 的峰值系统吞吐量为 763 Ipm,并发用户数为 70。三者当中,Web – Servlet – DB 的峰值吞吐量最高,达到 1033 Ipm,并发用户数为 120。图 6 显示不同架构下,各个服务器的 CPU 利用率。当动态页面生成器和 Web 服务器在同样一个机器上时

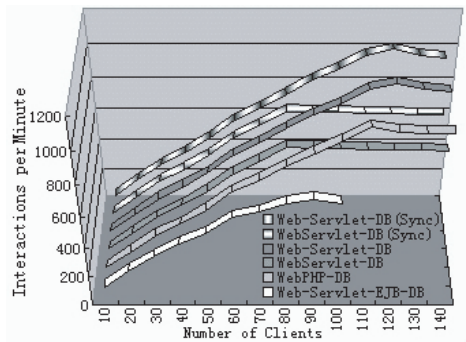

图 5　不同技术架构的吞吐量(Short Query Dominated)

(WebPHP – DB,WebServlet – DB),Web 服务器的 CPU 成为系统的性能瓶颈。在 WebPHP – DB 和 WebServlet – DB 的性能比较中,PHP 的性能要好,有 33% 的性能提升,这是因为 PHP 脚本的执行和 Web 服务器的通讯是进程内实现的,而 Servlet 和 Web 服务器需要通过进程间通讯进行数据交换。把 Servlet 迁移到不同的服务器上运行,减轻了 Web 服务器的负担,Web – Servlet – DB 能够获得三者中最高的系统性能。

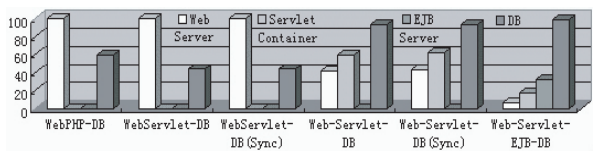

图 6　服务器的 CPU 利用率(Short Query Dominated)

由于整个负载都是由短事务构成,短查询占 80% 的比例,数据库系统的 CPU 利用率最高只达到了 65%,数据库系统也没有太多的加锁冲突,所以利用 Java 的同步机制协调用户访问,并不能带来性能的提升。从图 5 可以看出,WebServlet – DB(sync)和 Web-Servlet – DB、以及 Web – Servlet – DB (Sync) 和 Web – Servlet – DB 的性能大致相同。

Web – Servlet – EJB – DB 的系统吞吐量随着并发用户数量的增加而提升,达到 670Ipm 以后则开始下滑。从图 6,我们可以看出,EJB 服务器的 CPU 成为了系统的性能瓶颈,其利用率已经达到 100%。由于本测试采用了新版本 EJB 服务器 Jboss4.0,Jboss4.0 在数据库操作上做了大量的优化,性能得到了提升,通过比较图 5 和图 1 可以看出,EJB 配置的性能有了 81% 的提升。

5 相关工作

文献[6]比较了基于 PHP、Java Servlet、和 CGI 技术的 Web 应用程序的性能，其测试仅仅局限于数据的提取(只读事务)。本文的研究则基于 TPC－W 标准来进行，其负载更加具有代表性，和动态 Web 应用的实际负载更为接近。[6] 比较了 Java Servlet 的一种测试配置，得出 Java Servlet 比 PHP 和 CGI 效率更高的结论。我们则针对 Java Servlet 比较了负载均衡及利用 Java 同步机制等不同的配置，得出进一步的结论。Sun 公司的白皮书[7] 比较了 EJB、Java Servlet/JSP、PHP、CGI/Perl 等技术的功能，得出的结论是，Perl 或者 PHP 等技术，可以满足短期的目标，而 Java Servlet 则具有平台无关性、可移植性等优点，EJB 技术提供了最广泛的基础服务，系统架构最为灵活。本文是文献[7]的一个补充，在性能问题上对各个技术架构进行了比较。

6 总结

本文比较了动态 Web 的三种实现技术，分别是 PHP、Servlet、和 EJB。从编程和维护的容易性来比较，PHP 脚本、以及 SQL 语句和 HTML 语言混合在一起，更改和维护都相对困难。Java Servlet 有很好的开发工具如 Eclipse IDE，帮助进行开发、调试和部署，大大简化了开发的工作。随着 EJB3.0 技术的推出，采用标记开发方法，EJB 的开发变的更加容易，EJB 技术由于提供了完善的企业计算基础服务，用户只需要关心如何实现商业逻辑即可。从性能角度分析，PHP 技术在中小型动态 Web 网站中，具有相对的性能优势。Java Servlet 可以利用 Java 语言的同步机制减少数据库服务器的加锁冲突，获得更好的性能。当整个系统的性能瓶颈在 Web 服务器上的时候，可以通过把 Servlet 执行引擎迁移到其它机器上，达到负载均衡从而提高性能的效果。EJB 技术的性能在本文的研究中是最差的，究其原因，在于 EJB 容器的优化还没有被充分利用起来。厂商对新一代的 EJB 容器做了大量的性能优化，其性能获得了很大的提升，EJB 技术将获得更加广泛的使用。

参考文献

1 威利(澳)著，武欣译. PHP 和 MySQL Web 开发. 北京 机械工业出版社，2005. 13－35.

2 Jason Hunter，William Crawford. Java Servlet Programming. USA：O'Reilly，2001. 43－77.

3 霍尔(美国)著，赵学良译. Servlet 与 JSP 核心编程(第 2 版). 北京 清华大学出版社，2006. 55－67.

4 Harold W. Cain，Ravi Rajwar，Morris Marden，Mikko H. Lipasti. An Architectural Evaluation of Java TPC－W. In：H. Power ed. Proc. of the 7th Intl. Symposium on High－Performance Computer Architecture，Washington，DC，USA：IEEE Computer Society，2001. 229－235.

5 马林纳斯卡(美国)著，饶若南译. EJB 设计模式. 北京 机械工业出版社，2006. 152－195.

6 Amanda Wu，Haibo Wang，Dawn Wilkins. Performance Comparison of Alternative Solutions For Web－To－Database Applications. In：I. Banicescu ed. Proc. of the Southern Conference on Computing，University of Southern Mississippi：Academic Press，2000. 133－137.

7 Sun Microsystems. Comparing Methods For Server－Side Dynamic Content White Paper. USA：Sun Microsystems Press，2000. 51－57.