

文章编号:1001-9081(2005)02-0312-02

基于静、动态内容分离的 Web 应用性能优化技术的研究

赵 颢,魏慧琴

(北京交通大学 计算机与信息技术学院, 北京 100044)

(zhaoyang_163@tom.com)

摘 要:从目前 Web 领域的发展状况出发,简要介绍了基于 J2EE 架构的 Web 应用的体系结构。讨论了当前主流的将静态和动态内容都部署在应用程序服务器上的部署方式,并在此基础上对 Web 服务器和应用程序服务器之间划分文件的部署方式进行了研究,提出了通过静、动态分离来提高 Web 应用效率的新思路。最后以 IBM HTTP Server + WebSphere Application Server 为例对这两种方式的执行效率进行了分析和比较。

关键词:Web 应用;静态内容;动态内容;Web 服务器;应用程序服务器;Web 服务器插件

中图分类号: TP311 **文献标识码:** A

Research on performance improvement of Web applications based on separating static and dynamic content

ZHAO Yang, WEI Hui-qin

(Institute of Computer and Information Technology, Beijing Jiaotong University, Beijing 100044, China)

Abstract: According to the development of Web field, the construction of Web application based on J2EE was introduced. The popular approach of deploying the static and dynamic content on to Application Server was discussed, and the original approach of dividing files between the Web server and Application Server was researched, furthermore a new approach of performance improvement of Web application by separating static and dynamic content is advanced. Finally, using IBM HTTP Server and WebSphere Application Server, the efficiency comparison between the two approach was presented.

Key words: Web application; static content; dynamic content; Web Server; application server; Web server plugin

0 引言

在基于 B/S 的开发模式中,Web 应用非常普遍。作为 Web 应用的载体及提供应用服务的核心的应用程序服务器(Application Server),其性能的优劣直接决定整个应用的效率^[1]。因此,如何在有限的硬件条件下最大限度地提高 Application Server 的响应速度及吞吐能力,一直是一个热门的话题。

目前,普遍的做法是将整个应用程序部署到 Application Server 上,即静态访问和动态访问都由 Application Server 解释执行。虽然这种方法很简单,但是却是以巨大的性能损失为代价的。本文提出了将静态访问和动态访问进行分离的新的部署方法,以提高 Application Server 的响应速度及吞吐能力。

1 Web 应用组成

1.1 Web 服务器

Web Server(即 HTTP Server)提供 HTML、GIF、CSS files 等静态的内容,即 HTTP 文件服务。当 Web Server 接收到一个 HTTP 请求,会返回一个 HTTP 响应,例如送回一个 HTML 页面。为了处理一个请求,Web Server 可以响应一个静态页面或图片、进行页面跳转,或者把动态响应的产生委托给一些其

他的程序例如 CGI 脚本、JSP 脚本、servlet、ASP 脚本、服务器端 JavaScript,或者一些其他的服务器端技术,这些服务器端的程序通常产生一个 HTML 的响应让浏览器浏览。

其特点是:本身程序相对简单(大型的 HTTP Server 一般也就几十兆),功能相对单一,仅提供 HTTP 传输服务,即提供一个可以执行服务器端程序和返回(程序所产生的)响应的环境,因此,速度快,可提供非常高的服务器吞吐量,并且占用系统资源很少。

1.2 应用程序服务器

Application Server 解释执行 servlet 和 JSP 这些动态的内容,根据不同的客户请求按照 JSP 或 servlet 所定制的“规则”,组装成相应的满足客户不同需求的 HTML 页(也可以是 VXML 等各种 XML),返回给客户端。其间,根据需要调用 bean 或 EJB 完成所有的业务逻辑的处理,以及与后台 DataBase 或 EIS(企业信息系统)的交互。另外,现在大多数 Application Server 也包含了 Web Server 的功能,即可以把 Web Server 当作是 Application Server 的一个子集。例如,WebSphere Application Server 可以用自身的 File Serving 功能提供 HTTP 文件服务。

其特点是:本身程序相对复杂,能够完成商业逻辑层的各项功能,但占用系统资源较大。

收稿日期:2004-07-21;修订日期:2004-10-11

作者简介:赵颢(1979-),女,内蒙古人,硕士研究生,主要研究方向:计算机网络与数据库;魏慧琴(1965-),女,陕西人,副教授,主要研究方向:计算机网络与数据库。

1.3 Web 服务器插件

负责 Application Server 与 Web Server 之间的通信和工作的协调。当客户端向 Web Server 发出一个 HTTP 请求时, Web Server Plugin 首先根据其配置将请求进行重定向,以决定是由 Web Server 或由 Application Server 处理。Web Server Plugin 查看该请求的 URL,首先查看是否由 Application Server 处理,当发现 Application Server 并不处理该 URL 的请求时,再转交给 Web Server 处理。Web Server Plugin 的配置文件是由 Application Server 的配置决定的,当 Application Server 的配置变化时,Plugin 的配置文件会随之而改变^[2]。

1.4 Browser

负责接收用户的输入及解析执行服务器传来的 HTML 显示给用户,从而完成客户与服务器端的交互。

2 传统的静态、动态不分离方式的缺点

通常 Application Server 的管理员会采取比较直接的方法,按照 J2EE 的架构,将整个应用程序部署到 Application Server 上^[3]。使用这种方法,应用程序的动态内容或运行时组件(servlet、JSP 和 EJB)和静态内容(HTML 文件和图形图像)都被部署到了 Application Server 上。如 1 图所示(假设 Web 服务器和 Application Server 被部署到不同的物理机器上)。

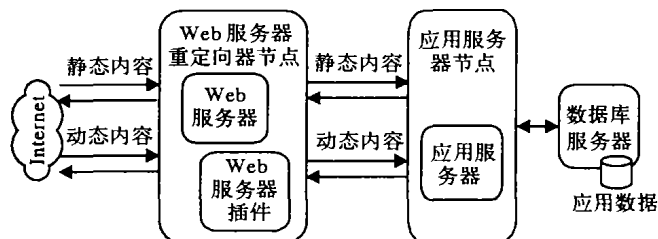


图1 静态、动态内容不分离的部署方式

使用这种拓扑结构构建的系统,静态文件作为逻辑 Web 应用程序的一部分,通过 Application Server 的静态服务功能执行^[4]。在 WebSphere Application Server 中,所有的 Web 应用程序都被打包成 WAR(Web Archive)文件来部署到 Application Server 上。一个 WAR 文件不仅仅包含 servlet 和 JSP 的 Java 元素,还包含静态内容(HTML 页面、GIF 文件和 JPEG 文件)。当客户端进行静态访问时,Web Server Plugin 根据 URL 和插件的配置文件 plugin-cfg.xml 中的信息,对每一个传入的 HTTP 请求进行检查,并将这个 HTTP 请求转发到 WebSphere Application Server 上。WebSphere Application Server 的缺省的文件服务(File Serving)行为解释执行该请求,并将结果返回到客户端。当一个针对包含在 WAR 文件中的文件的请求产生时,一个特殊的名为 file serving enabler 的“隐藏的”servlet 便被调用。这个 servlet 将相应的文件从 WAR 文件中适当的目录中取出,然后将它作为这个请求的响应返回。

这种方法的优点是简单、易用。这些文件只被保存在 Application Server 文件系统中,而不需要将它们保存在 Web Server 上。然而,这种简单性是以巨大的性能损失为代价的。对于所有由 Application Server 服务的文件都需要额外的网络跳树,如图 1 所示。并且使用这种方法处理文件需要 Application Server 做额外的处理工作,削弱了 Application Server 处理更多重量级的业务逻辑和处理更多事务的能力。

3 本文方法的部署方式与实现

在这种部署方式中,静态文件由 Web Server 解释执行。

其执行情况如图 2 所示。

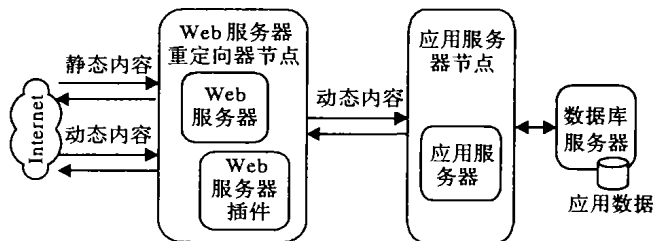


图2 静态、动态内容分离的部署方式

在 WebSphere Application Server 上实现动、静态访问分离的具体的操作步骤为:

1) 禁用 WebSphere Application Server 的文件服务功能。

禁用 Application Server 的文件服务功能后,只有 JSP 和 servlet URL 由 Application Server 提供服务。

2) 在 WebSphere Application Server 上安装更新的 EAR 文件。

3) 重新生成了插件文件。

禁用 Application Server 的文件服务功能后,Application Server 同时已经智能地重新生成了插件文件,以便让 Web Server 服务静态内容。插件文件将 servlet 和 JSP 的动态 URL 传递回到 Application Server。

4) 配置 Web 服务器来对静态 URL 加以识别。

5) 设置由 HTTP Server 来提供静态访问。

首先,将静态内容从 WAR 文件中取出再传递到 Web Server 中。其次,配置 Web Server 来对静态 URL 加以识别。在 IBM HTTP Server 中使用 Alias 伪指令来实现。最后重新启动 Web Server。

在这种部署方式中,Web Server 提供静态访问。如果停止了 Application Server,仍然可以从 Web Server 上访问到静态页面,但访问不到 Servlet 和 JSP 页面。并且,功能已在 Web Server 和 Application Server 之间分割开来,因此可以在根据站点中动态内容与静态内容的分割情况来变换分配在两者之间的功能总量^[5]。如果站点服务许多静态内容,那么从执行效率角度看,这样做是有用的。如果 Web Server 和 Application Server 执行相同的任务,那么添加更多的 Web Server 比添加更多的 Application Server 更合算。

4 两种方式的系统性能开销测试

为了对以上的方案进行测试,配置了三台服务器:

Server 1:

操作系统:Microsoft Windows 2000 Server 内存 1G

处理器:x86 Family 6 Model 8 Stepping 1 GenuineIntel ~ 596 Mhz

这台机器安装 IBM HTTP Server 1.3.26,其功能相当于分离的 Web 服务器。WebSphere Application Server 插件被配置在该服务器上以与 Server 2 进行互操作。

Server 2:

操作系统:Microsoft Windows 2000 Server

内存:512M

处理器 双 CPU

1) x86 Family 6 Model 7 Stepping 3 GenuineIntel ~ 548 Mhz

(下转第 316 页)

于 SessionGraphs 集合或 r_j 为符号“-”,则以 f_i 为顶点生成新的会话图 G ,并把 G 加入 SessionGraphs 集合;否则,则存在一条从 r_j 到 f_i 的有向弧。

2.2 会话识别图算法描述

输入:ExLF 日志文件。
输出:用户会话识别图(如图 1)。

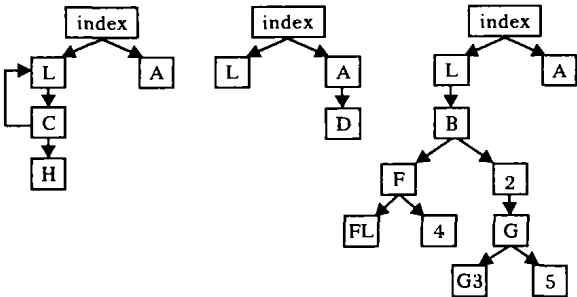


图 1 用户会话识别图

- 1) 对 Web 服务器的日志文件进行清洗、转化,生成 SessionTable 数据表;
- 2) 以 IP、Time、Agent 对 SessionTable 排序;
- 3) SessionGraphs = \emptyset , count = 0 及域值 T;
- 4) while not SessionTable.EOF() do
for each unique IP/Agent
if $t_j - t_{j-1} < T$
 f_i = SessionTable. RequestURL;
 r_j = SessionTable. ReferrerURL;
if r_j not exist in SessionGraphs or $r_j = '-'$
 $G = (f_i, \emptyset)$;
SessionGraphs = SessionGraphs $\cup \{G\}$;

```
count++;  
else if  $r_j$  in SessionGraphs and  $r_j$  在 SessionGraphs 中出现次数 > 1,  
 $r_k = f_i$  最接近网页;  
 $G.V = G.V \cup \{f_i\}$ ;  
 $G.E = G.E \cup \{r_k f_i\}$ ;  
else  
 $G.V = G.V \cup \{f_i\}$ ;  
 $G.E = G.E \cup \{r_j f_i\}$ ;  
End
```

3 结语

本文在介绍 ExLF 日志文件格式的基础上,阐述了会话识别图的概念,并给出了生成此会话识别图的启发式算法。我们将在以下相关领域进一步探讨和完善:

- 1) 在用户会话识别图基础上,研究会话图之间的相似性算法,可以对用户进行聚类分析。
- 2) 在用户会话识别图的基础上,以浏览页面的时间作为图权值,可以挖掘用户在一个网页上所花时间,来确定用户对感兴趣的页面。
- 3) 在用户会话识别图基础上,研究用户访问网页的进程,可以进一步挖掘用户行为模式。

参考文献:

- [1] 朱明. 数据挖掘[M]. 合肥: 中国科学技术大学出版社,2002.
- [2] SWEIGER M. 点击流数据仓库[M]. 陆昌辉,张光剑,陈佐,等译. 北京: 电子工业出版社,2004.
- [3] SLINOFF G. Web 数据挖掘[M]. 沈钧毅,宋擒豹,燕彩蓉,等译. 北京: 电子工业出版社,2004.
- [4] 史忠植. 知识发现[M]. 北京: 清华大学出版社,2002.

(上接第 313 页)

2) x86 Family 6 Model 7 Stepping 3 GenuineIntel ~548 Mhz
这台机器配置有 WebSphere Application Server 5.0.25。
Test Client:
操作系统:Microsoft Windows 2000 Server
内存:512M 处理器:x86 Family 15 Model 2 Stepping 4
GenuineIntel ~1804 Mhz

这台机器装有 Microsoft 的 Web Application Stress 负载测试工具,作为测试客户端运行测试。用单个线程单 Socket 对以下两个方案进行无间歇压力测试,测试的持续时间 3 分钟。

方案 1 由 Server 2 上的 WebSphere Application Server 服务静态内容,并且打开了 file serving enabler。

方案 2 由 Server 1 上的 IBM HTTP Server 服务静态内容,并将所有内容都传递回在 Server 上的 Application Server。

表 1 两个方案的测试结果

case	Number of test clients	Test length (h/min/sec)	Number of hits	Requests per second
1	1	0:03:00	293	1.63
2	1	0:03:00	441	2.45

尽管这些测试运行是最低限度的,但它们提供了一些有用的数据。首先,如表 1 所示,用分离的 Web Server 服务静态内容(这些静态内容来自与 Application Server 分离的机器)使性能有了明显的提高。如果 Application Server 服务静态内容,那么 Application Server 机器的 CPU 利用率比其他几个方案的都高。当 Application Server 不服务静态内容时,CPU 利

用率会较低并且在测试运行期间错误也较少。因此,Application Server 机器一般应执行紧急任务,最好不要让 Application Server 机器服务静态内容^[6]。

5 结语

本文提出了将 Web Server 独立配置,和 Application Server 的功能分离来提高性能的思想,并且通过测试已经取得了良好的效果。目前,这种思想越来越受到重视,越来越多的企业正在采用这种发式,IBM 的官方网站上也建议将动态、静态分离来提高性能,这说明这种新的部署方式有着十分广阔的前景。

参考文献:

- [1] 范国闯,钟华,黄涛. Web 应用服务器研究综述[J]. 软件学报,2003,14(10):1728-1739.
- [2] COCASSE S, KULKARMI M. IBM WebSphere Execution Team. Understanding the WebSphere Application Server Web sever plug_in [EB/DL]. <http://www-3.ibm.com/software/webservers/appserv/doc/lates/prereq.html>, 2004.
- [3] 刘慧,李玉忱苏鹏. 基于 J2EE 架构的分布式 Web 应用的研究[J]. 计算机应用研究,2003,20:(9):47-49.
- [4] 郝永芳,舒云,王德军. Web 应用环境中的系统可用性设计[J]. 计算机应用研究,2003,20:(6):123-126.
- [5] SADTLER C, HEYWARD J, IWAMOTO A, et al. IBM WebSphere Application Server V5.0 System Management and Configuration [EB/DL]. <http://www.ibm.com/redbooks>, 2004.
- [6] 姜昌华,朱敏,陈优广. Web 应用程序压力测试[J]. 计算机应用,2003,23:(10):75-77.