使用 OSCache 提高 JSP 系统的响应性能和稳定性

Improving the JSP Application's Response Performance and Stability with OSCache

覃 华　苏一丹　（广西大学计算机与信息工程学院　530004）

摘要：JSP 是一种主流的 B/S 开发技术，但它也存在着二层架构的种种缺点。本文提出了使用 OSCache 高速缓冲器优化 JSP 应用系统的方案，解决 JSP 系统在高并发网络访问下系统响应慢、稳定性差等问题，最后通过实验证明方案是可行的。

关键词：JSP　OSCache　JSTL 标记

1　引言

相对于传统的 ASP、PHP 技术，JSP 具有运行速度快、系统开销小、跨平台等优点，使得它成为中小型电子商务系统的最佳选择。从本质上说，JSP 仍属于 B/S 二层架构，所以在出现高并发访问时，系统往往会出现系统响应慢、系统资源消耗严重、濒临崩溃等现象。因此，为了提高 JSP 系统性能，不同的 JSP 服务器采用了不同的优化方法，如 weblogic8 通过自带的 HTTP 缓冲技术提高 JSP 的性能。本文的目标是探讨一种通用的、与 JSP 服务器无关的、可移植的解决方案。

首先来探讨 JSP 的一般工作原理。JSP 是动态网页，它的一般执行过程是：客户端发出访问请求；JSP 服务器查找相应的 JSP 网页，如果不存则返回访问出错信息，如果存在则判断此 JSP 网页是否已经编译成 servlet，如果没有编译则把 JSP 页面译成 JAVA 源代码，再编译成 servlet 字节代码，最后将 serlvet 调入内存中，交由 JVM 执行，网页内容被其中的 JAVA 代码打印返回到客户端。由此可见，通过 JVM 执行 servlet 产生 JSP 输出这正是 JSP 性能上的一个瓶颈，所以当并发网络访问量大时，因 JVM 运行负荷过重而使得 JSP 系统的响应速度变慢，最终引发系统不稳定。

在实际应用中，在某一段时间内，JSP 网页上有相当一部分内容是不变的（如：静态网页部分等），可以把这些不容易变动的信息暂存在缓冲区中，客户端需要这些信息时，直接从缓冲区中读出返回，而不需要通过 JVM 运行 servelt 产生，这样既能提高系统响应速度，又减轻 JVM 的负担。在本文中，我们提出使用 OSCache 高速缓存框架对 JSP 网页中不容易变动的信息进行缓冲的方案。

OSCache 由 OpenSymphony 公司设计的一个高速缓存框架，它提供了一组 JSTL 标记，可用于对 JSP 页面内容有选择地进行快速存缓冲。OSCache 能够在任何 JSP1.1 兼容的服务器上运行，它不仅能够为所有用户缓冲现有 JSP 代码块，而且能够以单用户为单位进行缓冲。OSCache 还包含一些提高可伸缩性的高级特性，比如：缓冲到磁盘、可编程的缓冲刷新、异常控制等等。

2　OSCache 的安装与配置

OSCache 是一个开放源代码的软件，可免费使用，下载后将其中的 oscache.jar、logging.jar 文件复制到 JSP 网页的 WEB－INF/lib 文件夹下；把系统配置文件 oscache.properties、标记库文件 taglib.tld 复制到 JSP 网页的 WEB－INF/classes 文件夹下。如果使用的 JAVA 环境是 JDK1.3，则需要将 commons－collections.jar 复制到 JSP 网页的 WEB－INF/lib 文件夹下，通过它来提高 collection 的处理速度；JDK1.4 不用复制此文件，因为 JDK1.4 的 collection 已经作了优化。

配置标记库。用记事本打开 WEB－INF/web.xml 文件，将 OSCache 的标记登记到系统中，在 web.xml 中增加以下代码：

```
<taglib>
  <taglib－uri>oscache</taglib－uri>
  <taglib－location>/WEB－INF/classes/taglib.
```

tld < /taglib – location >

< /taglib >

配置 OSCache 的工作参数。Oscache. properties **中定义 OSCache 工作参数有**:

(1) cache. memory = true | false。**这个参数定义是否允许用内存作缓冲。如果不允许,则用硬盘作缓冲器。默认值是使用内存缓冲**。

(2) cache. capacity = n。**其中 n 是一个整数,这个参数用于定义缓冲器中最多能容纳多少个对象。默认值是不限**。

(3) cache. algorithm = XXX。**这个参数是选择缓冲算法** XXX,**可用的算法有**:

com. opensymphony. oscache. base. algorithm. LRUCache – "**最近常用**"**算法,如果设置了** cache. capacity **属性后,这个算法就成为默认值**。

com. opensymphony. oscache. base. algorithm. FIFOCache – "**先进先出**"**算法**。

com. opensymphony. oscache. base. algorithm. UnlimitedCache – "**不限缓存**"**算法。添加到缓存中的内容永远不会被丢弃。如果** cache. capacity **属性没有设置,则这个算法是默认值**。

(4) cache. unlimited. disk = true | false。**这个属性定义:硬盘缓存是否能被当作无限大来使用,默认值是** false。

(5) cache. path = c: \cache。**这个参数定义** cache **文件在硬盘上的存储位置,例如** c: \cache **文件夹。注意文件夹的分隔符是**"\",**而不是**"\"。

3 OSCache 的主要标记

JSP **网页主要包括静态网页和动态部分(数据库访问部分),其中不常变动的信息可以用** OSCache **缓存在内存中,下次被访问时可立即从缓冲区中返回到客户端,而不必再由** JVM **执行** servlet **代码返回**。OSCache **提供了一套** JSP **标记库供对** JSP **网页进行缓存,这套标记符合** JSTL1. 0 **规范。主要的标记描述如下**:

3.1 < cache > < /cache > 标记

这是 OSCache **的主要标记。标记体中的内容在第一次被访问时写入缓冲池中,下一次访问时可立即返回客户端。每次这个标记被执行时,它会判断缓冲池中的内容是否已经陈旧,如果已经陈旧则会再执行一次标记体代码,将最新的内容刷新到池中。因此,** OSCache **即可以缓存静态内容,也可以缓存动态产生的内容(如:不经常变动的数据库记录、动态产生的图片、**PDF **文件等)。这个标记带有属性主要有**:

(1) key: **为缓存的内容起一个唯一的** ID **标识符。**cache **标记能自动为被缓冲的对象起一个** ID **标记符,默认是用**"URI + **查询串**"**构作,但如果想在其他页面内引用这些** cache **对象时,则要人为的定义一个明确的** ID **名,以方便引用。**ID **名是字串型的**。

(2) scope: **定义** cache **的作用范围,可取** session、application。

(3) time: **定义缓冲的内容的过期时间,单位为秒,默认值是** 3600 **秒,如果取负数表示永远不过期**。

(4) duration: **定义缓冲内容的持续时间,时间格式采用** ISO – 8601 **格式。**Time **和** duration **属性只能选择其中的一个使用**。

(5) cron: **它是使用** cron **表达式决定缓冲内容什么时候过期。它允许定义缓冲内容在指定的某个时间、日期过期,而不是缓冲一段时间后过期**。

(6) refresh: **取值为** true **时,表示强行刷新缓存中的内容,在程序运行期可用此参数强行刷新缓存中的内容。默认值为** false。

(7) mode: **取值为** silent **时,被缓冲的内容不会发送到客户端,如果想把一段内容写入缓存中但又不想让它返回到客户端显示时使用此参数。默认值为空,即缓存的内容被返回客户端**。

(8) groups: **它的值是用逗号分隔的若干个缓存对象的** ID **名,表示这些对象是有关系的一组,当组中的一个对象过期时,整个组中的所有对象均被当过期处理**。

(9) language: **语言代号,采用** ISO – 639 **语言编码**。

(10) refreshpolicyclass: **给出一个完整的类名,这个类是用户自定义的,用来决定缓冲的内容是否过期,这个类必须继承** com. opensymphony. oscache. web. WebEntryRefreshPolicy。

(11) refreshpolicyparam: **这是配合** refreshpolicyclass **属性用的,用于给** refreshpolicyclass **提供一些必要的参数**。

3.2 < flush/ > 标记

这个标记是个空标记,只有属性没有标记体。它可在程序运行期强行刷新缓冲中的内容。关键的属性

有：

（1）scope：定义被刷新的对象的范围。可取的值有 application、session、null。Null 表示刷新现有的所有缓存对象。

（2）key：缓存的 ID 名，它是配合 scope 一起使用，用于把指定的某个缓存刷新。本属性单独使用是无效的。

（3）group：组名，配合 scope 一起使用，表示将指定组名中的所有缓存同时刷新。

（4）pattern：缓存名的通配符，配合 scope 一起使用，即把符合通配符的缓存刷新。

（5）language：语言编号。

4　性能测试实验

实验环境是 WIN2000server，数据库系统 MS－SQL SERVER2000，JSP 服务器是 TOMCAT－4.1.24，JAVA 环境为 JDK1.3，Cisco 交换机局域网，100MB 网卡，DELL－GX260 机器 5 台（512MB 内存，P4－2GHzCPU）。

实验的主要目的是测试使用 OSCache 后，JSP 系统的性能是否有所提高。设计两个相同的 JSP 网页，网页中包含图片、FLASH 动画等静态网页元素，网页复杂度与中小型门户网站相当。在两张 JSP 网页的 <body> </body> 中适当位置加入访问数据库的代码，读出 MS－SQL SERVER2000 样例数据库 pubs/titles 表的数据并显示在网页上，用来测试 OSCache 对动态数据的缓冲能力：

```
<%
  try { //连接数据库
    Class.forName("com.microsoft.jdbc.sqlserver.SQLServerDriver");
    Connection con = DriverManager.getConnection("jdbc:microsoft:sqlserver://127.0.0.1:1433;databasename=pubs","sa","");
    Statement st = con.createStatement();
    ResultSet rs;
    rs = st.executeQuery("select title,type,price from titles");
    //将查询结果输出到网页
    while(rs.next())
    {out.print(rs.getString(1));
     out.print(rs.getString(2));
     out.print(rs.getString(3));
     out.print("<br>");
    }}
  catch(Exception e)
  { out.print(e.getMessage()); }
%>
```

在第二张 JSP 网页中加入 OSCache 标记进行缓冲操作。首先在网页的 <head> </head> 中声明 OSCache 标记库，在本文中是：

```
<%@ taglib uri="cachetags" prefix="oscache" %>
```

然后使用将网页上的内容放入 cache 中，在本文中是：

```
<oscache:cache> <body> ……被缓冲的内容…… </body> </oscache:cache>
```

在 TOMCAT 中设置 JSP 页面请求超时时间是 5 分钟。

开始测试：首先在 IE 中分别访问上述的两张 JSP 网页，网页静态部分、数据库部分的内容都能正常访问、显示，此时也使第 2 张 JSP 网页完成了首次数据的缓存；然后打开 MS－SQL SERVER2000 查询分析器，输入一个事务的不完整的 SQL 语句：

```
begin tran
  update titles set price=1
```

执行以上语句，目的是将 titles 表的 price 字段值全部修改为 1，并使得 titles 表因等待事务提交而进入表上锁状态，此时会使其他访问此表的 SQL 语句产生无休止的等待，用以模拟 JSP 服务器及数据库服务器现处于高负荷“忙”运行状态。打开 IE，分别访问以上的两张 JSP 网页，第 1 张 JSP 网页只看到空白，静态网页部分也看不到，IE 状态栏显示“等待回应”的信息，5 分钟后出现“访问超时”的错误；而第 2 张 JSP 网页的静态部分很快就显示出来，但数据库访问部分内容仍是空白，5 分钟后数据库部分的信息显示出来，但结果集中 price 字段的取值是第 1 次访问的结果，而不是事务 SQL 语句中修改的“1”，说明此时显示的是第 1 次的“缓冲数据”。实验证实：使用 OSCache 后，JSP 系统的响应速度、稳定性、容错性有所提高，既使出现超时等错误，OSCache 系统有很好的容错能力，会自动使用最近的缓冲数据给客户端提供部分返回信息。

（下转第 60 页）

(上接第 67 页)

5 总结

针对 JSP 系统在高并发网络访问下表现出的不稳定等缺点,本文提出使用 OSCache 高速缓冲器对 JSP 系统进行性能优化的方案,利用它可缓冲静态、动态信息的优点来提高 JSP 系统的响应速度、稳定性、容错性,实验证实方案是可行性的。在项目中可根据实际需要作灵活使用,以提高 JSP 系统的整体性能。

参考文献

1 林上杰、林康斯,JSP2.0 技术手册,电子工业出版社,2004-6。

2 Phil Hanna 著,闻道工作室译,JSP 技术大全,机械工业出版社,2002-1。

3 Ben Forta 等著,章明、吴疆等译,JSP 应用程序开发指南,清华大学出版社,2001-6。