文章编号: 1006-2475(2008)11-0134-03

基于 J2EE的高校教学管理系统的性能优化设计

王 ,李翠华

(厦门大学信息科学与技术学院, ,福建 厦门 361005)

摘要:随着 J2EE开发平台的普及和发展,基于 J2EE的教学管理信息系统在高校中的应用也日趋广泛。但是,这一应用也面临着在高负载和高并发条件下的性能问题的困扰。本文以基于 J2EE的高校教学管理系统的开发实践为背景,提出了系统性能上的优化设计方法,如使用服务器集群、负载均衡和数据缓存等技术,在实际的应用中有效地提高了系统性能,使系统达到了高负载和高并发的需求,最后提供了这些设计应用效果的测试数据。

关键词: J2EE; 集群;负载均衡; 性能优化
中图分类号: TP39 **文献标识码**: A

Performance Optimization Design of Education Management System Based on J2EE

WANG Kun, LI Cui-hua

(College of Computer Science and Technology, Xiamen University, Xiamen 361005, China)

Abstract: With the popularization and development of J2EE platform, more and more universities apply this technology to develop their Education Management System. Meanwhile, it also meets some performance problems in the situation of high load and high concurrent. On the basis of the development practice of a J2EE-based Education Management System, discusses a series of performance optimization designs, such as using cluster, load balancing and data cache technologies. A realistic example shows that these designs improve the performance effectively and the system meets the demand of high load and high concurrent. Then shows the applied effect of these designs with testing data.

Key words: J2EE; cluster; load balancing; performance optimization

0 引 言

建设高水平的大学, 教学管理信息系统建设是非常重要的一个环节。而 J2EE(Java 2 Platform, Enterprise Edition)是目前得到广泛应用的最复杂、最成熟的企业应用模型。作为一种分布式计算的体系结构,它在事务管理、持久性、安全性、构件管理等方面提供了强大支持[1],因此利用 J2EE开发教学管理信息系统有着许多优势,也是目前的一种趋势。但如果使用不当,所开发的系统性能也可能会比较差,无法满足大量并发用户访问的需要,导致系统不可用。因此对其进行性能优化设计是开发过程中的一个必要环节。本文以基于 J2EE的教学管理系统开发为背景,从整体层面上对系统进行优化设计,如采用应用服务器和图片服务器分离,服务器集群和负载均衡,以及数据缓存等一系列优化方法。通过优化设计,有效地提高了系统性能。

1 性能优化设计

1.1 应用服务器与图片服务器分离

对于 Web服务器来说,图片是最消耗资源的,将图片与页面进行分离,这是大型网站都会采用的策

收稿日期: 2008-09-25
基金项目:国家 863计划资助项目(2006AA01Z129);国家重点基础研究发展计划(973计划)
作者简介:王 (1984-),男,福建闽侯人,厦门大学信息科学与技术学院硕士研究生,研究方向:视频与图像处理;李翠华,教授,博士生导师,研究方向:视频与图像处理。

略,他们都有独立的图片服务器。这样的架构可以降低提供页面访问请求的服务器系统压力,并且可以保证系统不会因为图片问题而崩溃。在系统中,由于 Apache在处理静态页面和资源文件 (图片、css等)上的优势,则使用 Apache作为图片服务器。而对动态页面的请求则由 Apache转发给应用服务器 Tomcat来处理。这样应用服务器就只须处理动态页面的请求,缓解了其在大并发量下的压力,提高了系统在大并发量下的性能,取得了较好的效果。

1.2 服务器集群与负载均衡

解决高负载和大并发量请求的一个有效的方法就是采用服务器集群和负载均衡。负载均衡的思路下多台服务器组成服务器集群,每台服务器都可以单独对外提供服务。然后通过某种负载分担技术,将外部发送来的请求均匀分配到集群中的某一台服务器上,而接收到请求的服务器都能独立回应客户机的请求[2],因此服务器集群和负载均衡技术是建立一个高负载系统的关键性技术。在教学管理系统中采用了 Apache +Mod_jk + Tomcat的反向代理负载均衡方式。系统共使用三台服务器,一台安装 Apache作为负载均衡器,另外两台安装 Tomcat作为应用服务器集群,并进行负载均衡的配置。配置完成后,当客户请求到达负载均衡器时,负载均衡器就会根据配置的应用服务器集群的信息以及配置的负载均衡策略,将请求转发到后台的应用服务器集群的一个节点上,如图 1所示。在反向代理负载均衡方式中, Apache能够自动检测到停止掉的 Tomcat,然后不再发请求过去[2]。通过负载均衡,以及可以在群集中配置更多的应用服务器来分担用户的请求,从而能显著地提高系统最大并发用户数,同时自动平衡集群内每台服务器的负载,保证了高负载的情况下系统能快速地做出响应。并且,在群集中的一台服务器不可用时,服务不会中断,保证了系统的稳定性。

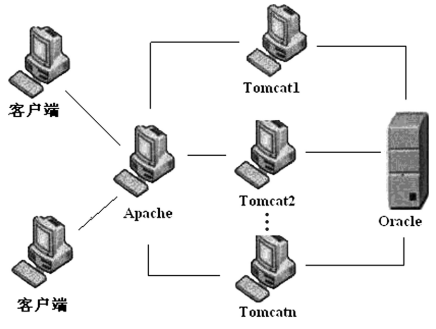

图 1　负载均衡示意图

1.3 数据库访问优化设计

对数据库的访问速度在很大程度上影响应用系统的性能,提高访问数据库速度的方法有很多,在系统中,主要在以下几方面进行优化。

1.3.1 数据库连接池技术

在应用程序中创建数据库连接是一项开销很大的操作,通常要耗费 0.5到 2秒的时间。为了有效解决频繁的数据连接和关闭带来的系统性能降低,有必要预先建立一定数量的数据库连接存放在缓存中构成数据库连接池,当需要访问数据库时,直接从连接池中取出连接,使用完后再把连接归还到连接池,而不是关闭。这样就不必在每次访问数据库时都重新建立连接,因此极大地节省了资源和时间[3]。在数据访问层,系统选用 Hibernate作为持久化方案,因为 Hibernate是免费、开源的,有丰富的文档和稳定的开发背景,并且提供了便利的池配置方案[4]。本系统采用了 Hibernate开发组优先推荐的第三方连接池产品 C3P0。

连接池的规模会极大地影响应用的性能。在优化测试过程中,曾经出现无法获取数据库连接的问题,针对本系统最大并发用户数较高的情况,将连接池规模调高,从而解决了这一问题。

1.3.2 使用缓存

缓存技术是减少对数据库直接访问的一种手段。缓存中的数据是数据库中数据的拷贝,应用程序在读数据的时候可以直接读取缓存中的数据。由于缓存通常以内存作为介质,而数据库是以硬盘作为介质的,从内存读取数据的速度远比从硬盘读取数据要快,因此缓存减少了直接读写数据库的次数,从而提高了数据的读写速度。合理有效地设计和使用缓存是优化应用系统性能的重要手段,这一点在基于 Web的、支持大量用户的系统中尤为明显[1]。

在 Hibernate中,实现了良好的缓存机制。由于在系统中,存在频繁的查询操作,因此对经常使用的查询语句使用查询缓存,可以减少数据库操作,提高查询性能。首先,在 Hibernate中配置第二级缓存,由于本系统使用了应用服务器集群,因此选择了支持群集环境的 JBossCache缓存插件作为 Hibernate的二级缓存[5]。JBossCache提供了 Reapplication式的缓冲,即如果集群中某个节点的数据发生改变,此节点会将发生改变的数据的最新版本复制到集群中的每个节点中以保持所有节点状态一致。

配置好缓存之后,就可以在系统中利用 Hibernate的 Query接口的 setCacheable()方法来实现查询缓存。以其中的一个查询为例,如要查询名字为“王五”的学生信息,程序如下:

```
String name = "王五";
Query query = session.createQuery("from student s where s.name =: name");
query.setString("name", name);
query.setCacheable(true);
```

这样就实现了利用查询缓存的机制来优化查询性能。

1.3.3 采用在查询语句中绑定参数的方式

Hibernate的参数绑定机制能够利用系统使用的 Oracle数据库所提供的预编译 SQL语句的功能,提高查询数据的性能。预编译即只需编译 SQL语句一次,把编译出来的可执行代码保存在缓存中,如果多次执行相同形式的 SQL语句,不需要重新编译,只要从缓存中获得执行代码即可[5]。以系统中的一个查询为例,如查询年级为 2002级的学生,程序如下:

```
Query query = session.createQuery("from student s where s.grade =: grade");
query.setString("grade","2002");
```

文献[6]中提到 JDBC执行查询时需要访问数据库的次数的计算公式是:访问数据库次数 = 结果集中的列数 ×语句执行的次数 ×2。因此,采用了预编译语句功能可以大大减少语句执行的次数,从而减少访问数据库次数。

1.3.4 SQL语句调优

若一个 SQL语句要执行很长一段时间,对整个资源也将一直占用,会降低程序的执行效率,因此应尽量使用优化过的 SQL语句。在系统中,根据以下一些规则对 SQL语句进行了调优:

(1)充分利用索引,避免相关子查询。嵌套层次越多,效率越低。如果必须使用,应该在子查询中过滤掉尽可能多的行。

(2)合理使用 EXISTS子句。避免在 where子句中使用 is null或 is not null。

(3)充分利用连接条件。在某些情况下,两个表之间有不止一个的连接条件,这是在 where子句中将连接条件完整地写上,可以大大提高查询速度。

(4)如果条件允许,可以适当增加冗余字段。虽然这种方式违背了第三范式,但它确实是提高查询性能最有效的手段之一。在适当的位置增加冗余字段,可以减少连接查询,提高了查询效率。

2 优化效果分析

使用 Loadrunner作为性能测试工具对系统在优化前后进行测试。选择学生登录教学管理系统并查询课程作为模拟场景,并模拟多个虚拟用户并发访问。在优化前,系统达到的最大并发虚拟用户数为 300,而优化后最大并发虚拟用户数可达到 400,并发用户数大大增加。这里要注意实际用户数一般能达到虚拟用户数的 2～3倍。图 2和图 3给出了优化前后的系统性能对比。从图中可以看出,经过优化设计后的系统响应时间增快,并且 Web服务器资源占用率有所降低。在实际使用中,系统运行稳定,没有出现服务器死机的现象,系统的并发用户数也达到了用户的要求。

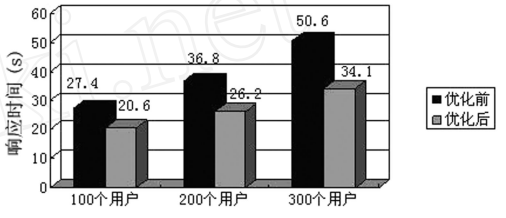

图 2 系统平均响应时间比较

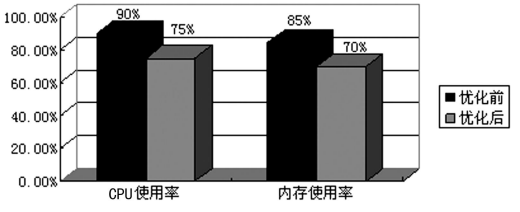

图 3 Web服务器资源使用情况比较
(300个虚拟用户)

3 结束语

通过以上的优化设计,使得系统达到了用户的需求,具有良好的性能。但是,除了以上提到的优化方法外,还可以在程序代码、数据库设计、应用服务器的配置等方面对系统进行优化。Oracle资深技术专家 Thomas Kyte说:80%的性能问题都出现在设计和实现级,而不是数据库级[7]。通过对应用的修改,常常能让系统性能呈数量级地增长。因此,在整个开发阶段,都应该把性能作为一个目标精心地设计,合理地构建,并且不断地调测,这样才能设计出具有良好性能的系统。

(下转第 142页)

(上接第 136页)

参考文献

[1] 陈丽冰.基于 J2EE的 Web应用系统的性能优化方法研究[J].计算机科学,2006,33(7):95-97.

[2] 宋海平,伍新华. 大型应用系统中数据库性能优化的研究[D]. 武汉:武汉理工大学,2007:39-42.

[3] 王新辉,杨海明,叶潇.基于 Hibernate对象持久化技术的优化策略分析与具体实现[J].开发研究与设计技术,2007,2(10):1075-1076.

[4] 宋海根,张亚东. 基于 J2EE税务征管系统持久层的 Hibernate解决方案[J].微计算机信息,2007,22(1-3):223.

[5] 孙卫琴. 精通 Hibernate:Java对象持久化技术详解[M].北京:电子工业出版社,2005.

[6] 赵强,乔新亮. J2EE应用开发(WebLogic + JBuilder)[M]. 北京:电子工业出版社,2003:96-108.

[7] Thomas Kyte. Expert Oracle Database Architecture 9i and 10g Programming Techniques and Solutions(Pap/Cdr edition)[M]. Apress, 2005:36-42.

[8] 刘金,徐苏,冯玉华. 基于 Hibernate的 J2EE数据持久层的设计与实现[J].计算机与现代化,2007(3):64-66.

[9] 苗晓辉. 基于 J2EE的数据持久化的研究与实现[J].计算机工程,2007(3):272-274.

[10] 成典勤,崔杜武. J2EE架构下数据库访问的性能优化[J].计算机应用研究,2006(4):64-66.

[11] 杨瑞. J2EE中提高数据库应用性能的方法[J]. 计算机系统应用,2003(11):60-62.

[12] 于晓慧. J2EE架构下数据库访问的性能优化研究[J]. 计算机应用研究,2005(4):90-92.