Web 应用服务器中实体 BEAN 的性能优化策略*)

杨 波 张文博 范国闯 陈宁江
(中国科学院软件研究所软件工程技术中心 北京100080)

摘 要 Web 应用服务器是 Web 计算环境下产生的新型中间件,为创建、部署、运行、集成和管理事务性 Web 应用提供一个跨平台运行环境。如何快速地响应大规模并发客户的请求,提供高可用性等特性是 Web 应用服务器需要解决的重点问题之一。本文采用 ECperf 作为性能测试基准,对影响实体 Bean 组件性能的瓶颈进行研究,给出若干优化策略,包括使用由容器管理实体 Bean 的持久化方式,选用更高级别的提交方式,声明只读的实体 Bean 等,并通过试验对优化的性能进行分析和比较。这些优化策略已应用到中科院软件所自主研制的 WebFrame 应用服务器中,取得了良好的效果。

关键词 J2EE,ECperf,性能,实体 Bean,Web 应用服务器

The Policies of Performance Optimization of Entity Beans in Web Application Server

YANG Bo ZHANG Wen-Bo FAN Guo-Chuang CHEN Ning-Jiang
(Technology Center of Software Engineering, Institute of Software, Chinese Academy of Sciences, Beijing 10008)

Abstract Web application server is a new type of middleware which is emerged under the Web computation environment. It provides a cross-platform runtime environment to create, deploy, run, integrate and manage the transactional Web applications. How to quickly response concurrent requests from large-scale users and improve the availability is an important problem of Web application server. This paper uses ECperf as the performance benchmark to research the performance bottleneck of entity Beans and presents several optimizing policies, such as using container to manage the persistence of entity Beans, choosing appropriate commit option and declaring the read-only entity Beans. Test results under different policies are analyzed and compared to find the impact on performance of entity Bean. All the optimizing policies are applied to the WebFrame application server which was developed by the Institute of software, Chinese academy of sciences, and achieves great effect.

Keywords J2EE, ECperf, Performance, Entity Bean, Web application server

1 引言

Web 应用服务器是 Web 计算环境下产生的新型中间件,为创建、部署、运行、集成和管理事务性 Web 应用提供一个跨平台运行环境。基于 J2EE 的应用服务器是目前研发的热点。EJB 规范是 J2EE 的核心技术之一,它定义了用于构建大规模企业应用的一种分布式组件结构,EJB 组件是包含业务逻辑的高度可重用的组件。实体 Bean(Entity Bean)是 EJB 的一种主要类型,是应用服务器用来管理数据库的有效手段。由于实体 Bean 涉及大量的外部资源访问,因此它的性能对于包含它的应用系统具有重要影响。实体 Bean 的不良的设计和使用会成为 J2EE 应用的性能瓶颈。WebFrame 是中科院软件所自主开发的 Web 应用服务器,在我们对 WebFrame 的测试过程中,发现实体 Bean 是影响应用性能测试结果的关键因素,而这些性能瓶颈并非来自应用服务器本身的设计。为此,本文对 Web 应用服务器中影响实体 Bean 组件性能的因素进行研究,给出若干优化策略,并通过试验对优化的性能进行分析和比较。

本文的第2小节和第3小节分别介绍与实体 Bean 和 ECperf 测试相关的背景;第4小节提出和分析了实体 Bean 的优化策略;第5小节介绍了这些优化策略在 ECperf 测试中的测试,并对测试结果进行分析。

2 实体 Bean 的有关概念

2.1 实体 Bean 的生命周期

EJB 组件驻留于应用服务器中的 EJB 容器,EJB 容器为 EJB 组件提供一个实时的运行环境,管理 EJB 的生命周期。EJB 规范[1]规定了实体 Bean 的三种状态:"不存在"、"被缓冲"和"预备"。各个状态之间的转换如图1所示。EJB 容器调用实体 Bean 的 setEntityContext 方法使之进入"被缓冲"状态,此时 Bean 没有和任何标识相关联,即在缓冲池中的所有实例都是相同的。当执行实体 Bean 的 find 方法时,容器从缓冲池中取出某个实例来给客户提供服务,此时实体 Bean 进入"预备"状态,它已和某个实体 Bean 的信息相关联,用户可以调用它的业务方法。ejbLoad 和 ejbStore 方法用来同步实体 Bean 和它的底层存储的状态。通常在事务开始时调用 ejbLoad 方法来获取该实体 Bean 代表的底层存储介质的最新状态,在事务结束时调用 ejbStore 方法把更新后的 Bean 的状态写回存

*) Supported by the National High Technology Development 863 Program of China under Grant No. 2001AA113010; 2001AA414020; 2001AA414310(国家高技术研究发展计划863); the National Grand Fundamental Research 973 Program of China under Grant No 2002CB312005 (国家重点基础研究发展计划973).杨 波 硕士研究生,主要研究方向为网络分布计算,软件工程技术。

储介质。

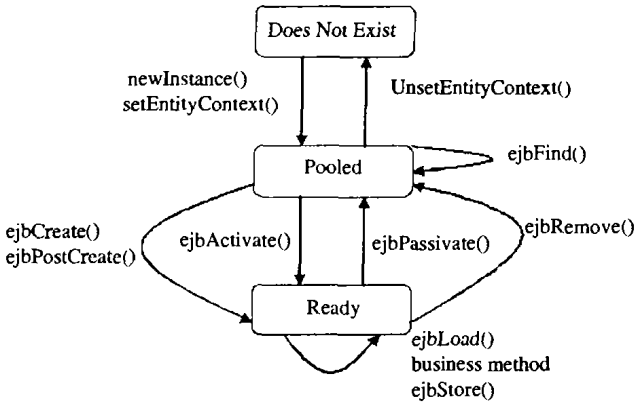


图1 实体 Bean 的生命周期

2.2 实体 Bean 的生命周期管理与持久化

根据 EJB 规范，EJB 容器为 EJB 组件提供组件池和生命周期管理，对于实体 Bean，EJB 容器会维护一定数量的处于“被缓冲”状态的实体 Bean 实例。EJB 容器管理这个组件池的大小、如何分配其中的实例给客户使用以及这些实例的生命周期。

EJB 规范规定 EJB 容器在事务之间缓冲实体 Bean 代表的数据，可使用三种提交方式来规定 EJB 容器的缓冲行为。这三种提交方式是：方式 A，EJB 容器在事务之间缓冲实体 Bean 的标识和数据；方式 B，EJB 容器在事务之间只缓冲 Bean 的标识而不缓冲数据；方式 C，EJB 既不缓冲数据也不缓冲 Bean 的标识。实现三种提交方式所体现的缓冲策略是几乎每个应用服务器都会考虑的问题。

EJB 规范规定了实体 Bean 的两种持久化方式，一种是由 Bean 的提供者负责编写数据访问的代码，称为 Bean-Managed Persistence(BMP)；另一种是由容器自动生成数据访问的代码，称为 Container-Managed Persistence(CMP)。这两种持久化方式各有利弊，BMP 方式使用比较繁琐，可移植性较差，但是可以处理复杂的 Bean 之间的关系；而使用 CMP 可以使 Bean 的开发人员更关注业务逻辑，由专家来完成系统的持久化服务，由于 Bean 的代码里不包含与特定数据库相关的信息，可移植性也更好。

3 ECperf 性能测试基准

本文使用 ECperf[3,6]作为测试案例和工具对 WebFrame 应用服务器进行系统测试和性能测试。ECperf 是由 Sun 公司联合多家软件生产商共同开发的 Web 应用服务器的测试基准，它主要衡量应用服务器中 EJB 容器的性能和可扩展性问题。Ecperf 实现了一个模拟现实世界生产状况的一个电子商务系统。它以产品制造、供应链管理、订单/库存管理作为测试用业务，使用“及时制造”(Just In Time)模型，这些业务都是具有普遍性的工业领域分布式问题。ECperf 使用了诸多基本的中间件服务，如名字服务、分布式事务、缓冲、对象持久化和资源管理池等。

ECperf 模型包含客户域、生产域、供应域和公共域，各个域之间通过交互成为一个完整的系统。每个域包含一组典型的事务，称为 Ectrasaction。ECPerf 的四个域中都包含一组功能相近、相互关联的 EJB 组件，被部署到受测系统(SUT，System Under Test)中。此外，ECperf 还包含一个供应商模拟器组件，它负责同供应商交互，使用 Java Servlet 实现并被单独部署到一个 Web 容器中。供应域通过供应商模拟器来模拟发送和接收购货单。ECperf 应用中各个主要模块间的关系如图 2所示。

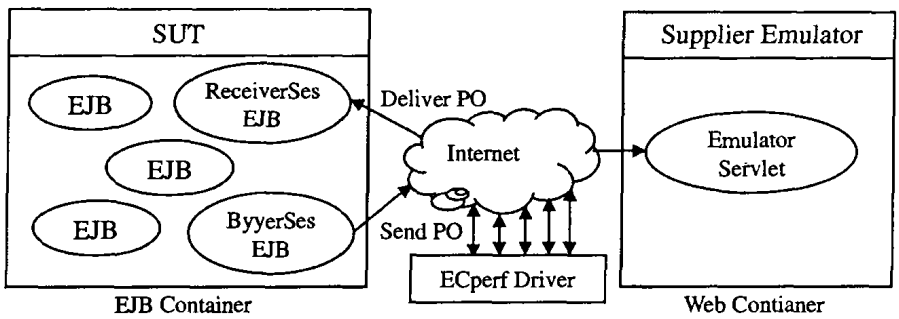


图2 ECperf 应用模型

ECperf 测试的吞吐量是指服务器在单位时间内完成的客户请求的数量，它由模拟订单应用和生产应用的客户所驱动。吞吐量同指定的客户的发射率(Ir，Injection Rate)紧密相关，发射率越高则表示模拟的客户数量越大，从而服务器处理请求的密度也越大。ECperf 测试基准的性能衡量指标是 BBops/min，它表示在每分钟内完成的基准业务操作(Benchmark Bussiness OperationS per minute)。有关 ECperf 更多信息请参阅 ECperf 规范[3]。

4 实体 Bean 的性能优化策略

4.1 ECperf 测试的性能瓶颈分析

经过最初的几次测试，我们发现 ECperf 测试在 WebFrame 上的表现非常差，随着测试发射率的提高，性能并不能如预期的那样大幅提高，甚至当测试压力进一步增大时，性能反而出现了下降。

·为了解决测试中出现的问题，首先采取消除资源访问限制的措施，包括设置足够的 Java 虚拟机的堆栈值以保障足够的内存、增加数据库连接池容量以避免数据库访问阻塞等，但是测试数据仍未有很大的提高。在多次测试过程中，我们发现受测系统和数据库服务器的 CPU 占用率都非常高，这表明了测试过程中进行了过多的数据访问。ECperf 测试中使用实体 Bean 来进行数据的持久化，当尝试增大应用服务器的组件池缓冲时，测试数据有了明显提高，这初步验证实体 Bean 是 ECperf 测试中的性能瓶颈。然后尝试将默认的提交方式由 B 改为 A，数据库服务器和测试服务器的负载明显降低，测试数据也有进一步的提高。由此，结合与实体 Bean 相关的两个重要系统服务——组件池管理和持久化的考虑，我们提出了相应的实体 Bean 优化策略。

4.2 组件池优化策略

·选用更高级别的提交方式。WebFrame 应用服务器实现了对实体 Bean 的两种缓冲服务。实例池(Instance Pool)用于缓冲处于"被缓冲"状态的实体 Bean,池中所有的实例都是没有标识的,可以从池中获取一个实例分配给任何一个实体 Bean 使用。对处于"预备"状态的实体 Bean,使用实例缓存(Instance Cache)来缓存这些带有数据的实体 Bean。根据每种提交方式实体 Bean 所需的数据库访问数量,可以认为各种提交方式的性能比较,从高到低依次为 A>B>C。

·配置足够的缓冲池容量。如在 B 提交方式下,Instance Pool 的最大值应取一个足够大的数目;在 A 提交方式下,Instance Pool 和 Instance Cache 的最大值都应该大到不会限制它们的缓冲容量。关于 Pool 和 Cache 容量与实体 Bean 性能的关系见文[4]。

4.3 持久化优化策略

· 使用 CMP 管理实体 Bean 的持久化。容器控制数据访问逻辑时提供了很多自动的优化措施,因此使用 CMP 来管理实体 Bean 的持久化,性能要优于 Bean 管理的持久化。使用 Bean 管理数据访问,容器对数据的管理参与得非常少,它只是负责调用一些规范规定的回调方法,如 ejbLoad、ejbStore 等。不管 Bean 的数据是否改变,容器都会调用这些方法来同步数据,这在很大程度上加重了服务器的负担。而由容器管理 Bean 的持久化,容器可以知道实体 Bean 的数据是否被修改,或是那些字段被修改,从而它可以采取相应的方法来同步数据访问。在装载数据的时候,BMP 在装载一个 Bean 的数据时,需要两个数据库操作:ejbFind 方法用来查找底层存储介质的相应记录并获得它们的主键;ejbLoad 方法从存储介质中读取 Bean 的数据。在装载 N 个 Bean 的数据时,使用 BMP 需要 $N+1$个操作,即1个 ejbFind 方法调用和 N 个 ejbLoad 操作;而使用 CMP,这 $N+1$个操作可以合并到一个操作中完成,这也很大程度上减轻了服务器的负担,从而使它的性能得到优化[5]。

·将使用过程中数据无需修改的实体 Bean 声明为只读。对只读的实体 Bean,EJB 容器只在第一次使用该 Bean 时调用它的 ejbLoad 方法装载数据,其后对 Bean 的使用都在 Cache 中进行,这样极大地减少了数据库的访问的数量。

5 实验与分析

5.1 实验设计与结果

为了验证上述优化策略,我们按照 ECperf 测试基准中的分布式要求设计了实验环境,包括受测系统(SUT)、数据库服务器、ECperf 供应商模拟器和客户模拟器等四个部分,如图3所示。

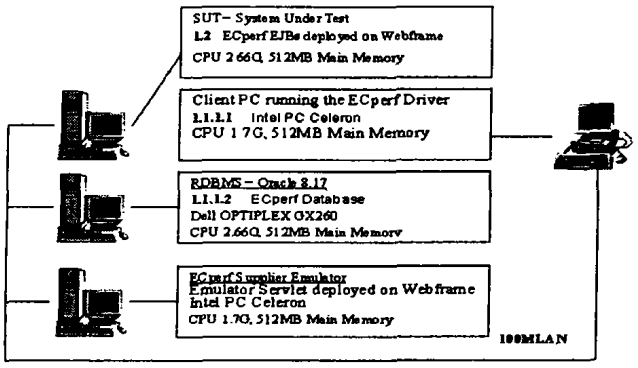


图3 ECperf 部署环境

ECperf 应用提供了分别支持两种实体 Bean 持久化方式(BMP 和 CMP)的版本。针对不同的持久化方式和提交方式,我们设计了六组实验,分别是 CMP-A、CMP-B、CMP-C、BMP-A、BMP-B 和 BMP-C,其中 A、B、C 分别代表3种提交方式。为了排除其它因素对实验结果的影响,在实验过程中依据如下的原则:

·使用方式 A 提交的实验,相应的实体 Bean 的 Pool 和 Cache 的容量保证足够大。

·使用方式 B 提交的实验,相应的实体 Bean 的 Pool 容量保证足够大。

·每次测试开始重新装载数据库。

·每次测试需重启服务器,清除上一次测试的 Pool 和 Cache 的内容。

·每次测试过程中用于"热身"时间不低于5分钟,以充分利用 Pool 和 Cache。

图4是六组实验总体实验结果。

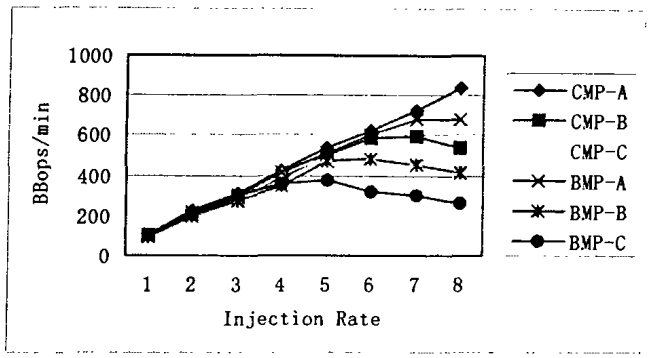


图4 不同测试条件下的 ECperf 测试结果

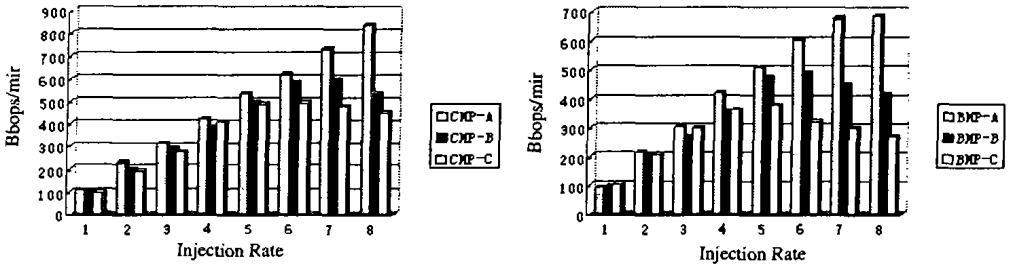
由于 ECperf 测试允许5%的误差,因此各组测试在 Ir<4 时,可以认为性能差异不大。随着测试压力的增大,各组测试的性能差异逐渐明显。总体的实验结果表明,CMP 方式确实比 BMP 方式的性能要优越,而组件池进行优化后应用服务器的性能也得到明显的改善。下面分类讨论不同场景下实体 Bean 的性能差异。

5.2 提交方式 A、B 和 C 的比较

从上面的测试结果可以看出,不同提交方式的性能比较为 A>B>C,为了更清楚地展现不同提交方式下实体 Bean 的性能差别,图5给出使用 CMP 和 BMP 的实体 Bean 的性能比较。

图5不同提交方式下的 ECperf 测试结果从图5中的两个图表可以看出,不管实体 Bean 使用 CMP 或是 BMP,提交方式 A 明显优于其它两种提交方式,而提交方式 B 要略优于提交方式 C。

三种提交方式的不同缓冲策略决定了实体 Bean 性能的较大差别。方式 A 是效率最高的,一旦一个实例为某个实体 Bean 服务后,它在整个生命周期内都能作为底层存储介质的一个"镜像",它可以避免几乎所有的昂贵的数据库操作。而方式 C 是效率最低的,它除了进行必要的数据同步操作外,还需要进行额外的 ejbActive 和 ejbPassivate 操作,这两个操作虽然不像 ejbStore 和 ejbLoad 等数据库操作那么耗时,但是在系统负担过重时,无疑会降低实体 Bean 的性能。如图5中所示,使用 BMP 的实体 Bean 在 Ir=5、使用 CMP 的实体 Bean 在 Ir=6时,C 提交方式的性能开始明显低于 B 提交方式。



(a) CMP (b) BMP

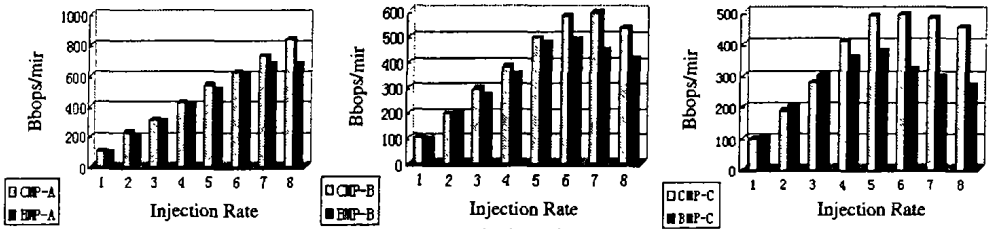
图5

除了性能方面的考虑外，决定实体 Bean 的提交方式还要考虑其它方面的问题。方式 A 只使用于单个服务器实例访问数据库时，这种排他性的访问使得 Bean 可以长时间缓冲数据而不会造成数据状态的不一致。如果在不能保证完全的排他性访问时，应使用 B 或 C 提交方式。

另一个需要考虑的问题是服务器的内存使用问题，服务器在进行各种缓冲服务时，需要占用大量的内存。就性能而言，提交方式 A>B>C 是建立在不限制组件池的两种缓冲的资源使用情况的基础之上。在实验过程中，使用提交方式 A 在发射率较高时，占用的内存是提交方式 C 的3～5倍，而使用提交方式 B 占用的内存是方式 C 的1～2倍。因此对可占用内存有限的服务器而言，选用提交方式 C，性能会比其它两种提交方式更稳定。

5.3 CMP 和 BMP 的比较

使用 CMP 的实体 Bean 在性能上总体要明显优于使用 BMP 的实体 Bean。但是这些优化措施的效果在不同的提交方式下表现也不尽相同。



(a) 提交方式 A (b) 提交方式 B (c) 提交方式 C

图6 CMP 和 BMP 的 ECperf 测试结果比较

图6中三组性能比较图分别展示了不同提交方式下，CMP 与 BMP 对实体 Bean 性能的影响。在 A 提交方式下，两者的性能差别直到 Ir=8时才开始显现。这一点在 B 提交方式下表现比较明显，因为使用 B 提交方式，EJB 容器在事务结束后不会缓冲实体 Bean 的数据，因此容器对数据库操作的优化较早的时候就表现出来(Ir=6)。而 C 提交方式下，EJB 容器不会保留 Bean 的标识，每次实体 Bean 的操作需要比 B 提交方式多出 ejbPassivate 和 ejbActivate 两个操作，这在系统负担很重的时候，加剧了使用 BMP 的实体 Bean 的性能恶化。

5.4 声明只读的实体 Bean

ECperf 应用中可以声明为只读的实体 Bean 的数目占到实体 Bean 总数的40%。使用这个优化策略可能会对性能的提高有很大帮助，为此专门设计了针对该优化策略的实验，比较使用 CMP 的实体 Bean 在声明只读前后在各种提交方式下的性能差别。

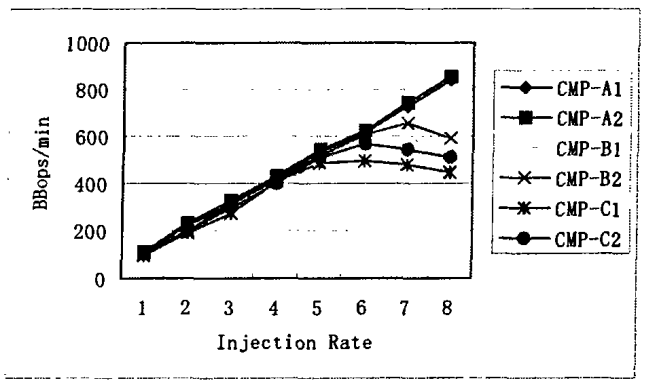


图7 把实体 bean 声明为只读后对实体 bean 在不同条件下的性能影响

图7表示在 ECperf 应用中所有可以声明只读的实体 Bean 都进行声明后，使用 CMP 的实体 Bean 在各种提交方式下的性能比较。图中后缀为1和2的线分别表示声明只读的实体 Bean 之前和声明只读的实体 Bean 之后的测试数据。

由图7可以看出，将这些实体 Bean 全部声明为只读后，使用 CMP 的实体 Bean 在提交方式 B 和 C 下的性能有了较明显的提高。而由于提交方式 A 在事务之间本来就无需进行与数据库的同步操作，因此把实体 Bean 声明为只读的效果并不明显。

5.5 小结

在此对上述优化策略进行一个总结。使用容器管理实体 Bean 的持久化，性能总是优于同等条件下 Bean 管理的持久化，因此，应尽量使用容器来管理实体 Bean 的持久化；对可以声明为只读的实体 Bean，应该在部署过程中进行指定，这也可以有效地提高实体 Bean 的性能。

三种提交方式的选择则取决于应用的实际环境。如果可以确定服务器对数据源的独占式使用，并有足够的内存在对实体 Bean 进行有效的缓存，则使用 A 提交方式可以大幅地提高性能；如何选择提交方式 B 和 C 则主要需要考虑服务器可支配的内存情况，如果不能指定足够的缓存，则由于频繁的内存管理，提交方式 B 的性能会低于 C，这时选择提交方式 C 则是最理想的选择。

EJB 容器的行为也是对实体 Bean 进行性能优化的一个重要方面。声明容量足够大的 Cache 对 A 提交方式至关重要；声明足够大的 Pool 对 A 和 B 提交方式都很重要。

结束语 Web 应用服务器和 J2EE 应用的性能优化是一

个复杂的问题。本文采用 ECperf 作为性能测试基准，对实体 Bean 的性能优化策略进行了一些探讨。实验表明，在服务器资源充足的情况下，使用更高的提交方式可以使性能得到极大的提高；使用容器来管理实体 Bean 的持久化总会得到性能的优化。本文的优化策略和实验都取自中科院软件所自主研制的 WebFrame 应用服务器和 ECperf 测试。这些优化策略在其它的应用系统中也取得了明显的效果。

参 考 文 献

1 Sun Microsystems. Enterprise JavaBeans Specification, v2. 0. 2001. http://java. sun. com/products/ejb/docs. html

2 http://ecperf. theserverside. com/ecperf/

3 Sun Microsystems. ECperf specification. 2001. http://java. sun. com/j2ee/ECperf

4 Berbner P, Ran S. Entity Bean A, B, C's: Enterprise Java Beans Commit Options and Caching. In: proc. of IFIP/ACM Intl. Conf. on Distributed Systems Platforms - Middleware, 2001

5 Sucharitakul A. Seven Rules for Optimizing Entity Beans. Java Developer Connection - http://www. java. com/, 2001

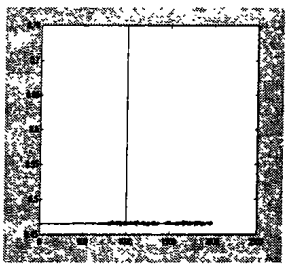
6 Deshpande S, Martin B, Subramanyam S. Eight Reasons ECperf is the Right Way to Evaluate J2EE Performance. The ServerSide. com J2EE Community, 2001. http://www. theserverside. com

(上接第117页)

与原始图像相比，嵌入水印后的图像在视觉上几乎没有什么改变，峰值信噪比 PSNR 的值为52. 2589，可见该算法的透明性较好。当水印没有受到攻击时，提取的水印 w^* 与原始水印 w 的相似度为1，说明可以完全地提取原水印。

对嵌入水印后的图像进行剪切攻击。用检测器对2000个随机产生的水印进行响应，其中只有一个与原水印相匹配(第1000个)，在该处检测器的响应最大，而其它地方的响应比较而言小得多，这显示了嵌入的水印都能被正确地检测出来。图4～图6给出了剪切攻击示意图及其对应的水印检测图，可以发现在剪切达到75%的时候，仍能够正确地检测出水印的存在，这表明该算法对剪切攻击的鲁棒性非常好。

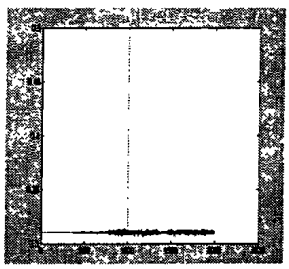
(a) 14. 1%的剪切图



(b) 水印的检测图

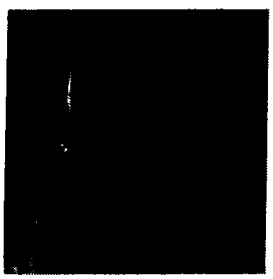
图4

(a) 50%的剪切图

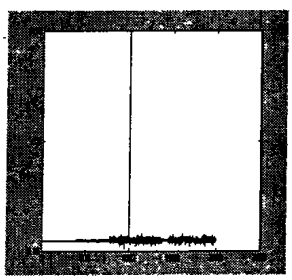


(b) 水印的检测图

图5



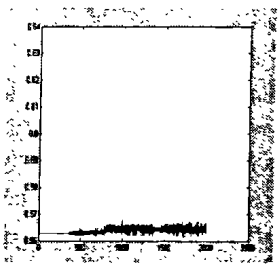
(a) 75%的剪切图



(b) 水印的检测图

图6

(a) 经滤波后的图像



(b) 水印的检测图

图7

对嵌入水印后的图像再进行滤波攻击，图7给出了经滤波后的图以及对应的水印检测图。

结束语 本文提出一种将混沌映射和人类视觉系统相结合的数字水印技术，混沌映射的应用提高了水印算法的安全性，同时结合了人类视觉的特性，保证了水印的不可见性。实验结果表明，本算法的隐蔽性较好，具有很强的抗剪切攻击能力。由于该水印算法是属于空域水印算法，因此算法的效率较好，实时性也较好；而且该算法在检测时采用了盲检测，不需要用到原始图像，因此实用性很强。

参 考 文 献

1 Swanson M D, Kobayashi M, Tewfik A H. Multimedia data-embedding and watermarking technologies [J]. Proc. of the IEEE, 1998, 86(6): 1064～1087

2 Petitcolas F A P, Anderson R J, Kuhn M G. Information hiding——A survey [J]. Proc. of the IEEE, 1999, 87(7): 1062～1078

3 Frank Hartung, Martin Kutter. Multimedia watermarking technologies [J]. Proc. of the IEEE, 1999, 87(7): 1079～1107

4 Voyatzis G, Pitas I. The use of watermarks in the protection of digital multimedia products [J]. Proc. of the IEEE, 1999, 87(7): 1197～1207

5 杨义先，钮心忻，等. 信息安全新技术. 北京：北京邮电大学出版社，2002

6 陈士华，陆君安. 混沌动力学初步. 武汉：武汉水利电力大学出版社，1998

7 钟桦，刘芳，等. 自嵌入空域易损水印技术. 计算机研究与发展，2003, 40 (6): 825～830

8 宿富林，马国强，等. 一种对图像剪切具有鲁棒性的数字水印算法. 电子与信息学报，2003, 25 (3): 295～299

9 王相生，甘俊人. 一种基于混沌的序列密码生成方法. 计算机学报，2002, 25(4): 351～355

10 卢宗庆，梅蕤蕤，等. 基于位平面的数字水印算法. 计算机工程与应用，2003, 39(6): 79～81

11 张军，王能超，等. 基于混沌影射和遗传算法的公开数字水印技术. 模式识别与人工智能，2002, 15(1): 42～47