基于 J2EE 的 Web 应用系统的性能优化方法研究

陈丽冰

(广东经济管理学院　广州 510260)

摘　要　应用软件的性能优化是计算机应用的重要问题。良好的性能是企业应用系统的重要质量指标之一。本文以一个基于 J2EE 的 ERP 系统的开发实践为背景，从减少网络通信量，使用缓存技术和优化数据访问这三个方面讨论了基于 J2EE 的 Web 应用系统的性能优化的一系列方法，并提供了这些方法的实际应用效果的测试数据。

关键词　J2EE，性能优化，RMI，JDBC

Research on Performance Optimization Approaches of J2EE-based Web Application Systems

CHEN Li-Bing

(Guangdong Institute of Business Administration, Guangzhou 510260)

Abstract　Performance optimization of application software is an important issue of computer application, and high-performance is one of the most important quality attributes of enterprise application systems. On the basis of the development practice of a J2EE-based web system, this paper discusses a series of approaches to optimize the performance of J2EE-based ERP applications from three aspects, including reduction of network communication, use of cache technologies and optimization of data access, then shows the applied effect of these approaches with testing data.

Keywords　J2EE, Performance optimization, RMI, JDBC

1　引言

J2EE(Java 2 Platform, Enterprise Edition)是目前最复杂、最成熟、最有影响力的企业应用模型。作为一种分布式计算的体系结构，J2EE 在事务管理、持久性、安全性、组件的多实例管理等方面提供了强大的支持，然而，在 J2EE 的实际应用中，尤其是基于 Web 的应用系统中，性能方面并不尽如人意。这主要有两方面的原因：首先，J2EE 的分布式计算环境给系统的性能带来了许多新的问题和挑战；其次，如今的系统开发更注重良好的体系结构，易于扩展和维护等特性，而往往忽视了系统的性能问题。但是，较快的响应速度，良好的系统性能仍然是应用系统的重要目标。许多研究者对 J2EE 系统的性能问题进行了研究[1~3]，但这些研究主要集中在数据库访问的性能优化上，并没有在整体层面上考虑系统的性能优化策略。针对这一问题，本文以一个基于 J2EE 的 ERP 应用系统的开发实践为背景，在保持系统的良好体系结构和易于扩展、维护的基础上，从减少网络通信量，使用缓存技术和优化数据访问三个方面讨论了基于 J2EE 的 Web 应用系统的性能优化的一系列方法，并对这些方法的使用效果作了测试比较。

2　优化方法

J2EE 应用系统的性能优化并不是一件容易的事，必须在充分了解 J2EE 的体系结构及其特点的基础上，从设计层面上对性能问题进行认真考虑，才能提出有效的优化方法。下面，将从三个方面对基于 J2EE 的 Web 应用系统的性能优化方法进行讨论。

2.1　尽量减少网络的通信量

2.1.1　*避免客户端和服务器之间的不必要的往返*　在基于 Web 的应用中，客户机浏览器是通过 HTTP 协议和服务器通信的。HTTP 是一种无状态的协议，其工作原理是：客户端发出一个请求，服务端收到请求后进行处理，并把响应信息发回给客户端。因此，客户端的每次请求将会引起客户端和服务器间的一次往返操作。频繁的往返势必对系统的响应速度造成严重的影响。

为避免不必要的往返，通常我们只需在向服务器查询或更新数据时才触发客户端、服务器间的往返。能在客户端执行的数据操作应尽量用客户端脚本(如 Javascript)来实现。例如，对用户输入数据的校验，应该尽量在客户端进行校验，再将数据提交给服务器。

2.1.2　*减少远程调用的次数*　在 J2EE 的分布式对象模型中，一个系统的对象可以调用运行在另一个系统中的对象的方法，被调用的对象称为远程对象。远程对象用一个远程接口来描述，由远程接口暴露远程对象的方法。远程方法调用(Remote Method Invocation，RMI)是指通过远程对象的远程接口，调用远程对象的方法的过程[4]。RMI 使开发者可以像操作本地对象一样调用远程对象的方法。

在 J2EE 中，RMI 的原理大致如下：调用者通过 JNDI 服务查找要访问的远程对象，从而获得一个 stub 对象的引用，该 stub 对象实现了和远程对象相同的接口。当调用者调用 stub 对象的方法时，stub 对象对方法的调用参数进行编排(Marshaling)并通过网络传给服务器的 skeleton 对象，skeleton 对象对方法参数进行反编排(Unmarshaling)，然后调用实际的远程对象的方法，远程对象将方法调用的返回值传给 skeleton 对象，skeleton 对象将其编排后传给 stub 对象，stub 对象进行反编排后传给调用者。

由此可见，一次 RMI 操作会创建一个 stub 对象和一个 skeleton 对象，还需要两次编排和反编排操作，这显然是个昂贵的操作。而在 J2EE 的实际应用中，RMI 主要用于访问封装应用程序业务逻辑的核心组件——EJB 上。因此，对 EJB 组件的远程调用次数是影响系统性能的一个关键因素。要减少对 EJB 的远程调用次数，可以从以下 4 个方面着手。

陈丽冰　硕士生，主要研究方向：数据库。

①用 Session Facade 封装对相关数据的操作。在 J2EE 的实际应用中，经常会遇到要对多个相关数据项目进行一系列的操作，而每个数据项目封装在各自的 Entity Bean 中的情况。例如，在 ERP 应用中，操作一张入库单需要访问入库单信息，库存信息，仓库信息和物料信息等等，通常这些信息都封装在独立的 Entity Bean 中。在处理这类情况时，如果让客户程序直接与这些 Entity Bean 交互，在其上进行一系列的操作，那么即使这些 Entity Bean 在同一个服务器上，客户程序也要进行多次远程调用才能完成操作，如图 1 所示。这样的设计方式显然会造成系统性能的低下。为此，我们可以采用 Session Façade 的设计模式，在客户程序和 Entity Bean 之间引入一个叫做 Session Facade 的 Session Bean，由 Session Facade 封装对多个 Entity Bean 的一系列操作，并对外提供统一的接口，客户程序通过该 Session Facade 的接口进行操作。这样一来，客户程序只需一次远程调用就可以完成操作，从而可以大大提高性能，如图 2 所示。

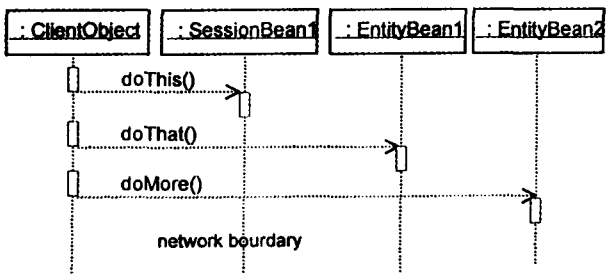


图 1　客户程序直接和 Entity Beans 交互

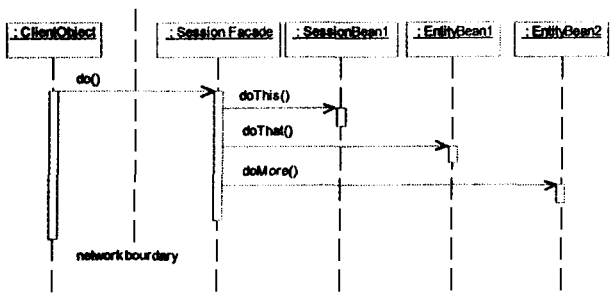


图 2　客户程序通过 Session Façade 与 Entity Beans 交互

②以 ValueObject 的形式封装，传递 Entity Bean 的数据。Entity Bean 中有多个属性，而这些属性通常是一起使用的。例如客户信息包括姓名、年龄、电话、地址等，访问客户信息时往往会需要所有这些属性。如果让客户程序将这些属性逐一从 Entity Bean 中取出，就会出现多次远程调用。为解决这个问题，我们可以把 Enity Bean 的属性封装在一个 Value Object 中，函数调用参数和返回值尽量采用 Value Object 的形式，这样也可以明显减少远程调用的次数，从而改善性能。

③利用本地接口提高性能。本地接口是在 EJB2.0 中引入的新特性。通过本地接口访问 EJB，参数以引用(Reference)的方式传递，并不需要像远程调用那样的昂贵操作。在实际应用中，特别是在基于构件的系统的开发中，通常以上文提及的 Session Façade 的形式提供构件的对外接口，而在 Session Façade 内部，实现业务逻辑的组件和封装业务数据的组件在逻辑上是紧耦合的，不需要位置透明性，所以它们之间的交互可以通过本地接口来实现，从而可以获得性能的改善。

④采用数组传递参数来改善批量操作的性能。在实际应用中，经常会遇到批量操作的情况。例如，在 ERP 应用中，计算一个月的材料消耗情况时，往往会产生成百上千个结果记录。如果将结果记录逐个地传给 EJB 组件，则需要进行多次远程调用，如下面的程序片断 1,2 所示。

程序片断 1：Session Façade 中，生成材料消耗结果记录项的接口方法定义

```
public void createMaterialSpentItem(MaterialSpentItem item);
```

程序片断 2：批量生成材料消耗结果记录的程序段

```
façadeHome= getRemoteFaçadeHome();//取得 Session Façade Bean 的远程 Home 接口
façade=façadeHome. create();//获得 Session Façade Bean 的组件接口
List result=calculateMaterialSpent();//获得材料消耗汇总的结果
for( int i=0;i<result. size();i++) {
//逐一将结果记录传给 EJB 组件，执行远程调用，次数与循环次数相等
façade. createMaterialSpentItem((MaterialSpentItem)result. get(i));
//……(1)
}
```

为改善性能，可以对程序片断 1,2 进行调整，增加接受集合作为参数的接口方法，如下面的程序片断 3,4,5 所示，其中的粗体部分是修改了的内容。

程序片断 3：改变后的接口方法定义

```
public void createMaterialSpentItem(MaterialSpentItem item);
public void createMaterialSpentItem(List items);
```

程序片断 4：新增加的接口方法的实现

```
public void createMaterialSpentItem(List items){
    for (int i=0;i<items. size();i++){
    //此处调用的是本地方法
    createMaterialSpentItem((MaterialSpentItem)items. get(i));
    }
}
```

程序片断 5：改变后的批量生成材料消耗结果记录的程序段

```
façadeHome= getRemoteFaçadeHome();//取得 Session Façade Bean 的远程 Home 接口
façade=facadeHome. create();//获得 Session Façade Bean 的组件接口
List result=calculateMaterialSpent();//获得材料消耗汇总的结果
//将整批材料消耗结果记录项作为参数一次性传输，只需要一次远程调用
façade. createMaterialSpentItem(result); //……(2)
```

从以上的程序片断可以看出，在修改前的程序片断 1,2 中，远程调用发生在(1)处，调用次数取决于结果记录项的个数；而在改变接口后的程序片断 3,4,5 中，远程调用发生在(2)处，只需一次远程调用。由此可见，通过增加接受集合作为参数的接口方法，可以大大减少远程调用的次数，从而提高系统的效率。

2.2　充分使用缓存技术

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。在 J2EE 的应用中，缓存技术主要可用于以下方面：

① 对各种资源的引用进行缓存。在 J2EE 应用中，经常要用到多种资源，如 DataSource、EJB 的 home 接口、数据库连接池、JMS 连接池等等。通常，获得这些资源的引用是通过 JNDI 查找来实现的。JNDI 查找是一个比较耗时的操作。因此，对于要多次使用的资源应当进行缓存并进行统一管理，以避免不必要的 JNDI 查找操作。

②对被频繁访问的业务数据进行缓存。在应用系统中，通常会有一些业务数据会被多个用户频繁访问。如 ERP 系统中的供应商信息、客户信息、部门信息、物料信息、用户权限信息等。如果每次访问这些数据时都通过 EJB 或 JDBC 去取，那将会引起远程调用，访问数据库等既耗时又耗资源的操作，给系统造成较大的负担。因此，如何对这些业务数据进行有效的缓存，成了改善系统性能的又一关键所在。

对业务数据的缓存和对资源的缓存并不一样。业务数据有其自身的特殊性。首先，不可能对每种业务数据都进行缓存，缓存需要额外的内存开销，对那些很少被访问的数据没必要进行缓存；其次，对那些数量庞大的业务数据只能进行部分

缓存，如 ERP 应用中的物料清单，通常有几千甚至几十万项，全部缓存显然是不切实际的，只能对经常的那部分进行缓存；再次，被缓存的数据所对应的数据库中的数据有可能被更新，必须要有统一的机制来保持缓存中的数据和数据库中的数据的同步。

在我们开发的 ERP 系统中，引入了一个叫 CacheProxy 的中间件来对业务数据的缓存进行统一管理。CacheProxy 的原理类似于操作系统中的高速缓存的实现原理，每种需要缓存的数据都有自己的缓存池，可以根据实际需要设定缓存池的最大记录数，采用 LRU(最近最少使用)的淘汰策略进行管理，当缓存的数据所对应的数据库中的记录被更新时，CacheProxy 会收到通知并刷新缓存中的相应数据。

2.3 优化数据访问的设计

对数据的访问速度在很大程度上影响应用系统的性能，提高访问数据库的速度的方法有很多，在 J2EE 中，除了传统的改善数据库的结构，优化 SQL 语句等方法之外，主要可以从以下方面进行优化。

①使用连接池。建立数据库的物理连接时，DBMS 需要为其分配多种资源，反之，释放连接时，DBMS 要释放掉这些资源。分配和释放资源都是耗时的操作，因此，反复建立/释放数据库连接会对系统的效率造成不良影响。在 J2EE 中，JDBC2.0 及其后续版本中采用了连接池技术。在连接池中维护多个活动的数据库物理连接，当系统需要进行数据访问操作时，从池中取一个获得连接，操作完毕后并不直接释放连接，而只是把连接放回池中，以便以后使用。这样就可以大大减少建立/释放数据库物理连接的次数，从而获得更好的性能。

②使用 PreparedStatement 对象和 executeBatch 方法。在 JDBC 中，通常使用 Statement 对象的 executeUpdate()方法进行数据更新。但在实际应用中，有时需要进行大量的数据更新操作(包括 INSERT，UPDATE 和 DELETE)。如果通过 executeUpdate()进行操作，则每次调用都会向 DBMS 发送 SQL 语句，而 DBMS 往往跟应用服务器处在不同的机器上，多次的网络通信显然是不合适的。为此，可以使用 addBatch()和 executeBatch()方法。这是 JDBC2.0 中新增加的方法。通过 addBatch()可以将多个更新数据的 SQL 语句加进一个批(batch)中，然后通过 executeBatch()一次性将整个批提交给 DBMS 执行。这样就可以减少应用服务器和 DBMS 的通信次数，加快数据访问的速度。

此外，如果要执行多个形式相同，只是参数值不同的 SQL 语句时，可以使用 PreparedStatement 对象，因为 DBMS 在执行 SQL 语句之前要对其进行编译。如果用 Statement 对象执行，即使语句的形式相同，每次操作前也需要进行编译[5]。而 PreparedStatement 可以一次性编译语句的形式，执行语句时只需设置相应的参数即可[2]。这样，一个语句只编译一次，就可以多次执行，从而可以提高数据访问的效率。

③合理使用事务。在分布式环境中，执行分布式事务需要对很多资源进行锁定。在 J2EE 中，应用服务器提供了方便、完善的分布式事务环境支持，但是，不适当地使用事务可能会造成系统的性能问题。在使用事务应注意以下两个方面：

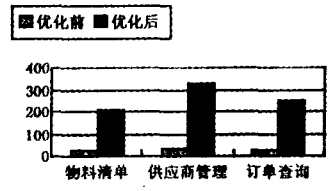
避免不必要的事务。不要把任何操作都放到事务环境中。一般而言，只有那些要对数据或外部资源进行更改的操作才需要使用事务。而像查询数据等不会修改数据的操作就没必要放到事务环境中。

对于执行时间较长，又不需要立即返回结果给用户的事务，应尽量采用异步的方式执行。例如，在 ERP 应用中，计算物料需求计划往往需要几十分钟或更长的时间。如果在系统繁忙时执行这样的事务，可能造成系统缓慢甚至因资源不足而崩溃。为此，在我们的 ERP 中，我们使用异步的方式处理该事务，通过 JMS 触发事务的执行，并指定适当的优先级，使系统在负荷较低的时候执行事务，而用户可以在适当的时候查看结果。

④直接通过 JDBC 实现查找。在 J2EE 应用中，通常都会把业务数据封装在 Entity Bean 中。然而，每个 Entity Bean 的实例都需要使用一定的内存资源，当 Entity Bean 的实例过多时，会造成系统内存资源的紧张，影响系统的响应速度。在实际应用中，如果通过 Entity Bean 来实现查询，则有可能在内存中产生很多 Entity Bean 实例。为避免这样的情况，应该直接通过 JDBC 实现查询，以减少内存的占用，提高系统的性能。

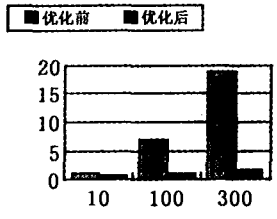
3 实际效果

以上提及的性能优化方法都已在我们的系统开发中得到应用，并取得了良好的效果。图 3 和图 4 是我们的系统中部分模块在优化前和优化后的页面性能(以 page/second 度量)和事务性能(以 transaction response time 度量)的效果对比。测试平台是 BEA WebLogic Server7.0。从图中可以看出，上述性能优化方法使系统的性能获得了约一个数量级的提高。



(横轴为样本模块，纵轴为每秒产生的页面数)

图 3 页面性能比较



(横轴为事务写入的记录数，纵轴为事务响应时间(秒))

图 4 事务性能比较

结束语 除了以上提到的优化方法以外，还可以通过优化数据库结构的设计，合理配置应用服务器所提供的性能优化选择，合理配置编译器选项等方法对系统进行优化。总而言之，要对基于 J2EE 的 Web 应用系统进行性能优化，必须充分考虑系统的各个方面和环节，进行详细周全的分析，这样才构建出性能良好的应用系统。

参 考 文 献

1 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
2 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
3 林建，董亚波，朱淼良. 基于 J2EE 的数据库访问技术设计模式研究[J]. 计算机工程与应用，2004，14：129～131
4 Sun Microsystems, Inc. JavaTM Remote Method Invocation Specification [S/OL]. 1999. http://java.sun.com/j2se/1.4.2/docs/guide/rmi/spec/rmiTOC.html, 2003-12-01/2003-12-23
5 Spell B 著. Java 专业编程指南[M]. 邱仲潘，等译. 北京：电子工业出版社，2001