持久对象框架中基于多级访问模式的对象预取技术研究与实现*)

安静斌 贾 焰 王志英 韩伟红
(国防科技大学计算机学院网络与信息安全研究所 长沙 410073)

摘 要 在基于关系数据库和对象关系映射的持久对象框架中,对象之间通常通过对象引用和各种集合属性将对象相互关联起来,组合成更为复杂的复合对象。应用程序对这些复合对象的访问则是通过使用这些属性逐个访问成员对象来完成。这种在多个成员对象之间的导航操作导致了客户端和后端数据库系统之间的获取操作大幅度增加,从而导致严重的性能问题。对象预取技术根据某种策略,将应用程序可能访问到的对象成组或批量地预先从数据库中装载到客户端,从而减少了应用程序对后端数据库系统进行查询的次数。本文对现有各种对象预取技术并对其进行分析分类,在此基础上,提出了一种基于多级访问模式的对象预取技术。最后,介绍了该算法在软件构件平台 StarCCM 的持久对象框架中的实现。

关键词 持久对象框架,软件构件平台,多级访问模式,预取

Research and Implementation of Multi-level Access Pattern-based Prefetching in Persistent Object Framework

AN Jing-Bin JIA Yan WANG Zhi-Ying HAN Wei-Hong
(School of Computer Science, National University of Defence Technology, Changsha 410073)

Abstract In the persistent object framework based on RDBMS and O/R mapping, objects connect each other via references and collection attributes, and applications navigate from one to another in these objects. These navigations dramatically increased the fetch operations between client and back-end database server, and brought serious performance problem. Object prefetching techniques load objects that the application maybe access successively from database to client-side cache by group or in batch based on some kind of predication policy. This paper classifies various object prefetching policy, and promots a new kind of prefecthing algorithm based on multi-level access pattern. Furthermore, a prototype implementation of this algorithm in the persistent object framework designed for our software component platform StarCCM.

Keywords Persistent object framework, Software component platform, Mmulti-level access pattern, Prefetching

1 引言

在目前的大型企业应用系统中,面向对象的程序设计语言、基于构件的软件设计开发方法以及大型关系型数据库系统都占据了技术的主流地位。但由于程序语言所使用的对象模型和数据库系统所采用的关系模型之间存在着"阻抗失配"问题,各种试图解决这一问题的方法一直在不断出现和改进,以适应新的应用环境和提供更高的效率。其中持久对象系统就是目前软件构件平台上使用的一种面向构件系统的对象持久化解决方案。

但是,在这种基于关系数据库和对象关系映射的持久对象系统中,对象之间通常通过对象引用和各种集合属性将对象相互关联起来,组合成更为复杂的复合对象。应用程序对这些复合对象的访问则是通过使用这些属性逐个访问成员对象来完成。这种在多个成员对象之间的导航(navigation)操作导致了客户端和后端数据库系统之间的获取(fetch)操作大幅度增加,从而导致严重的性能问题。通过在客户端增加一个缓存(cache)可以很好地减少这种获取操作。基于这一思想,在以往的对象关系映射系统、对象关系数据库系统以及面向对象数据库系统中,研究者们提出了很多对象预取(prefetching)技术。这些预取算法根据某种策略,将应用程序可能访问到的对象成组或批量的,预先通过查询语句从数据库中装载到客户端存储器中,从而减少了应用程序对后端数据库系统进行查询的次数。下面首先介绍现有各种对象预取技术并对其进行分析分类,在此基础上,提出了一种基于多级访问模式的对象预取技术。最后,介绍了该算法在软件构件系统 StarCCM[1] 的持久对象框架中的实现。

2 对象预取算法

实现持久对象的一个方法是将对象映射到关系数据库系统中。这种方法有两个主要的好处:一是给已有的关系数据库提供持久对象视图,另一方面它给关系数据库系统的客户提供了一条途径,让他们可以不需要采用新的数据库引擎就可以建立一个新的面向对象的数据库系统,从而避免了对数

*)本课题受国家自然科学基金项目"新一代网络中间件的体系结构、协议及实现机制"(No. 90104020)和国家 863 课题"网络环境的新一代中间件核心技术及运行平台"(No. 2004AA112020)的资助。安静斌 博士研究生,研究方向为分布计算与数据库;贾 焰 教授,博士生导师,研究方向为数据库系统和分布计算;王志英 教授,博士生导师,研究方向为计算机体系结构和信息安全;韩伟红 副教授,研究方向为分布计算与数据库。

据库管理过程的改变和与已有应用的互操作问题。因此,这种方法比对象关系数据库系统有更多的优势。但对象关系映射的一个主要不足就是性能问题,对于很多常用的应用场景来说,其性能都低于专门的面向对象数据库。

持久对象系统的一个重要特性就是将持久对象装入主存储器,使之成为使用应用环境对象模型的新的活动对象。如Java、C++、Smalltalk、COM等的对象。这减小了程序语言与数据库系统之间的阻抗失配,但同时对数据库的实现产生了更多性能上的挑战,尤其是在映射到关系数据库系统的时候。

一个主要的性能问题是应用程序的对象模型是可导航的(navigational)。即对象可以持有其他对象引用,或与其他对象有某种联系,应用程序可以循着这些关系从一个对象导航到另一个对象。

根据选取预取对象的不同策略,已有对象预取方法可以分为以下四类:

- 基于页的预取算法
- 基于用户设置的预取算法
- 基于上下文的预取算法
- 基于访问模式的预取算法

基于页的预取算法取回当前访问的对象所在的同一页中所有的对象。显然,这种方法只有当该页中的对象会被连续访问到时才有效。当应用程序对对象的访问是随机的时,这种方法就会失效。由于这种方法的有效性完全依赖于对象在页中的存放顺序,因此当应用程序不是按照对象的存放顺序进行访问时,这种预取策略就不能很好地工作。面向对象系统ORION[4]和ObjectStore[5]中就采用了这种预取策略。

基于用户设置的预取算法根据用户的设置来指导对象的预取。在一些商用的对象关系数据库系统使用这种预取策略,系统给用户提供了若干配置选项来设定预取策略。这种方法显然的缺点是它将责任推给了用户,加重了用户的负担,这与当前发展自调优的数据库管理系统的方向也不相符。

基于上下文的预取策略将所请求的对象的结构上下文(structure context)中所有对象预取到客户端。对象的结构上下文描述了一个结构,其中的对象构成了预取的上下文。例如对象查询所返回的结果、对象集合等。图1是一个基于上下文的预取策略的例子。我们假设对象O_1目前不在客户端缓存中,O_1的结构上下文就是根对象O_r的属性A的值。所以,当访问O_1时,O_1的结构上下文中的对象$O_1 \sim O_n$都会被预取到客户端。可以看出,当应用程序以广度优先方式(breadth-first-search, BFS)遍历时,这种预取策略是有效的。这种策略已经在商业书库库系统中得到了使用,相对于即时获取策略,这种预取策略最高可提高70%的性能[2]。基于上下文的预取策略在应用程序以深度优先方式(depth-first-search, DFS)遍历时会产生问题,如图2所示,当应用程序沿着O_r,O_1,O_{n+1},O_{n+2},…的顺序访问时,当该序列足够长时,就会占据所有客户端缓存空间,将预取的对象O_2,O_3,…,O_n替换出缓存。等到应用程序访问O_2时,又需要重新从服务端获取。此外,基于上下文的策略不能预取非集合属性所引用的对象。

基于访问模式的预取算法可以分为三类,分别是基于对象级访问模式的预取算法、基于页级访问模式的预取算法和基于类型级访问模式预取算法。三种方法都是在某个级别上,从应用程序以前的对象访问行为中获取一定访问模式信息,即某种规则,并以此为依据来决定预取哪些对象。在文[3]中,Curewitz,Krishman和Vitter在对最近的对象导航动作进行分析的基础上,使用压缩算法来指导页一级的预取。在文[6]中,Palmer和Zdonik提出了一种基于对象级访问模式的预取算法,根据对对象间引用的轨迹进行预先分析,形成访问模式,然后据此知道对象预取过程。在文[7]中,Han,Whang和Moon提出了一种新的基于类型级访问模式的预取算法。

本文在综合现有几种基于访问模式的算法的基础上,结合其优点,并根据软件构件平台对于对象持久化的需求提出了一种新的更高效的基于多级访问模式的预取算法。

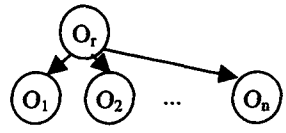

图1 基于上下文的预取策略

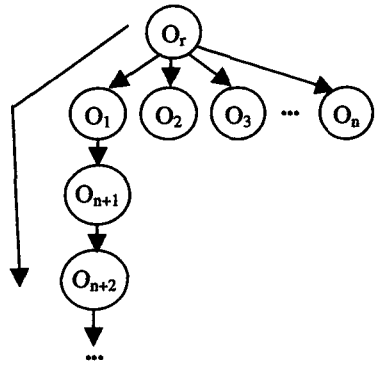

图2 基于上下文的预取策略在深度优先遍历史失效

3 基于多级访问模式的预取算法

本节重点介绍基于多级访问模式的预取算法,该算法在对象和类型两个级别根据应用程序访问持久对象的模式来实现对象的预取。下面给出算法的详细描述。

3.1 对象模型

为了描述本算法,我们综合了已有的几个对象模型(ODMG、CORBA、UML等)提出了一个简单的对象模型,简述如下:

1)每个持久对象的持久状态(persistent state)都由属性(attribute)构成。

2)对象的属性可以是一个标量、一个对象,或者一个集合。

3)每个标量(scalar)属性符合一个标量类型,它指出该属性的名称(name)和数据类型(type)。

4)每个对象有一个标量属性ObjectID,用来唯一标识该对象。

5)每个对象引用(object reference)属性指向另一个对象实体。

6)每个集合(set)属性包含一个集合对象,该集合包含标量值或者对象引用。

7)每个对象类型(object type)有一个名称和一个该对象类型可以包含的属性类型的集合。

8)类(class)是实现的一个或多个对象类型的代码实体。

3.2 相关概念

根对象集合(root set)是一组根对象的集合,应用程序获取这组对象并以这些对象为出发点导航到其他相关联的对象。根对象集合一般是对象查询所返回的结果集。

对象访问路径(object access path,OP)是指应用程序对持久对象的访问序列。

类型访问路径(type access path,TP)是指一个对象从其根对象导航到自身的所引用到的对象属性的序列。

对象访问模式图(object-level access pattern graph,OPG):根对象集合的对象访问模式图是一个有向图,图的结点是被访问的对象的 ObjectID,图的边则是被引用的属性。对象访问模式图用来获取和存贮对象的对象级访问模式。

类型访问模式图(type-level access pattern graph,TPG):根对象集合的类型访问模式图是一个有向图,图的结点是被访问的对象的类型,图的边则是被引用的属性。类型访问模式图用来获取和存贮对象的类型级访问模式。

3.3 访问模式获取算法

输入值:

OID_{cur}:所访问对象的 ObjectID

A:访问的属性的名称

算法步骤如下:

1. 如果对象 OID_{cur} 从根对象集合开始的访问路径 OP(OID_{cur}, A)不在当前 OPG 中,就将 OP(OID_{cur})添加到 OPG。

2. 如果对象 OID_{cur} 从根对象集合开始的类型访问路径 TP(OID_{cur}, A)不在当前 TPG 中,就将 TP(OID_{cur})添加到 TPG。

3.4 预取算法

输入值:

OID_{cur}:当前访问对象的 ObjectID

OPG:当前的对象访问模式图

TPG:当前的类型访问模式图

算法步骤如下:

1. 根据 TPG 中当前类型访问路径、属性类型和当前正在访问的对象的类型确定要预取的对象。

2. 预取后续对象,将其装入内存,并在 TPG Cache 中加入这些对象的引用。

3. 根据 OPG 中当前对象访问路径和当前正在访问的对象 ID 确定要预取的对象。

4. 预取后续对象,将其装入内存,并在 OPG Cache 中加入这些对象的引用。

步骤 2 中,根据 TPG 所确定的预取对象并不是直接通过访问数据库装入内存,而是要先执行对象一级的访问模式获取算法和预取算法。

4 基于多级访问模式的预取技术在软件构件平台持久框架中的实现

StarCCM 是我们实现的一个基于 OMG CORBA 构件模型的构件运行平台,其结构如图 3 所示。StarCCM 提供的构件类型分为三种:Entity,Service 和 Session 构件。其中 Entity 构件和 Session 构件需要将状态信息进行持久化。构件的持久化方式分为 CMP(Container-Managed Persistence)和 SMP(Self-Managed Persistence)两种。CMP 是指容器管理的持久化,在这种方式中,构件的实现不需要编写任何持久状态管理的代码。SMP 是指用户自管理的持久化方式,与 CMP 方式不同,用户需要提供构件持久状态数据的存储和装入代码,供容器在构件激活和去活时调用。StarCCM 中持久框架的体系结构如图 4 所示。其中 PSS 是 CORBA 的持久状态服务(Persistent State Service),容器通过 PSS 服务访问关系数据库实现对象到关系的映射。

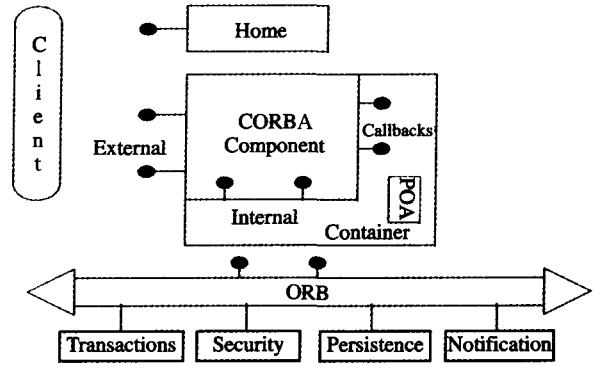

图 3 StarCCCM 构件运行平台体系结构

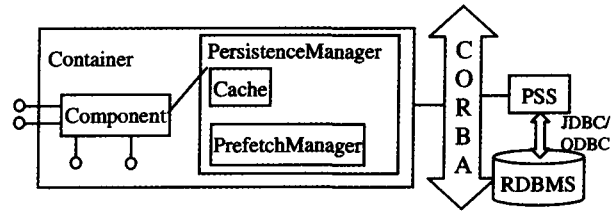

图 4 StarCCM 持久框架的体系结构

结束语 本文分析研究了现有各种持久对象预取技术的特点和优劣,在此基础上提出了一种同时基于类型一级和对象一级访问模式的预取算法。该算法结合了基于类型和对象两级访问模式信息的优势,更进一步提高了对象预取的命中率。最后本文介绍了我们在 StarCCM 构件平台的持久化框架中对该算法的实现,并分析了算法应用前后的持久对象访问的性能。

本文提出的算法本质上仍是基于访问模式的预取算法,因此不可避免地带有此类算法所共有的局限性。在将来的工作中,我们会尝试结合其他各类预取算法的优点,期望能够进一步提高持久对象的访问效率和持久框架的整体性能。

参 考 文 献

1 StarCCM: A CORBA Component Model implementation in C++ language. http://starccm.sourceforge.net, 2004

2 Bernstein P A, Pal S, Shutt D. Context-Based Prefetch for Implementing Objects on Relations. In: Proc. 25th Int'l Conf. Very Large Data Bases, 1999. 327~338

3 Curewitz K M, Krishnan P, Vitter J S. Practical Prefetching via Data Compression. In: Proc. ACM SIGMOD Int'l Conf. Management of Data, 1993. 257~266

4 Kim W, et al. Architecture of the ORION Next-Generation Database System. IEEE Trans. Knowledge and Data Eng, 1990,2(1)

5 Lamb C, et al. The ObjectStore System. Comm. ACM, 1991, 34(1):50~53

6 Palmer Z, Zdonik S B. Fido: A Cache That Learns to Fetch. In: Proc. 17th Int'l Conf. Very Large Data Bases, 1991. 255~264

7 Han W S, Whang K Y, Moon Y S. A Formal Framework for Prefetching Based on the Type-Level Access Pattern in Object-Relational DBMSs. IEEE Transactions on Knowledge and Data Engineering, 2005, 17(10)