

基于 Java 应用程序的可移植缓存模型

黄学雨¹, 张茂新²

(1 江西理工大学 应用科学学院, 江西 赣州 341000; 2 江西理工大学 信息工程学院, 江西 赣州 341000)

摘 要: 为了研究 Java 应用程序与缓存系统的相互独立性, 结合 Java 应用程序的缓存特征, 提出了一种基于 Java 的可移植缓存模型 PCM-JA. 借助 Java 技术的平台无关性, PCM-JA 型可以移植到不同的平台. 并且为 Java 应用程序提供了一个通用的方法来创建、访问、更新和删除 Cache 中的条目, 介绍了该模型的五个核心模块及其功能实现, 研究了具体的缓存策略, 结合现有缓存算法给出了相应的算法设计. 实验结果表明, 提出的缓存模型具有较好的性能.

关键词: Java; Cache; 可移植性; 缓存模型; 缓存策略

中图分类号: TP393

文献标识码: A

文章编号: 1000-7180(2014)09-0101-04

A Portable Cache Model for Java Application

HUANG Xue-yu¹, ZHANG Mao-xin²

(1 College of Applied Science, Jiangxi University of Science and Technology, Ganzhou 341000, China;

2 School of Information Engineering, Jiangxi University of Science and Technology, Ganzhou 341000, China)

Abstract: To research the Mutual independence between Java application and cache systems, Combined with the characteristics of the cache for Java applications, PCM-JA, a portable Cache-model is proposed. With the Platform-independence of Java technology, PCM-JA can be moved to different platforms. A common way to create, access, update and remove entries from Cache is offered by this model. Five core modules and their function realization is discussed. A specific Cache-strategy is studied, with the existing caching algorithm, the related algorithm is designed. Experimental results shows that the proposed PCM-J model have a good performance.

Key words: Java; Cache; Portable; Cache-model; Cache-strategy

1 引言

Java 由于它的简单、面向对象、分布式、可移植等特征使得 Java 程序应用于不同环境与开发平台. 目前, 开发者们仅仅是在应用程序性能不满足要求时才使用缓冲池技术或相关的缓存插件来应付, 这些缓冲池或者缓存插件都是和缓存系统绑定在一起的, 若切换到其他的平台, 这些临时的缓存方法在兼容性、配置以及系统的安全性上很难达到要求, 并且不能达到复用的效果.

文献[1]中提出了一种基于面向编程思想的对动态 Web 内容进行缓存的智能框架, 它通过对前端

Servlet 引擎和后端 JDBC 接口进行横切拦截, 为应用程序装入缓存及其管理体系. 这种缓存方案为解决缓存问题带来了新的思路, 但是, 它是一种侵入式的缓存方法, 而且在切点的检测、捕获和方面重新织入等等都配置繁琐, 安全性也很难保证. 文献[2]中介绍了一种将 OSCache、EhCache 和 Hibernate 整合在一起的缓存解决方案. 这种方案虽然简单可行, 但是它建立在其他缓存插件的基础上, 可复用性差, 当缓存环境变化时, 需要重新搭建框架.

针对现有应用程序缓存机制的缺陷, 本文提出了一种基于 Java 的可移植缓存模型 PCM-JA (Portable Cache Model for Java Application). 此模型在

收稿日期: 2013-10-23; 修回日期: 2013-12-17

基金项目: 国家自然科学基金项目“基于合作演化的组织学习动态绩效研究”(71061008)

应用程序与缓存系统之间植入透明的缓存管理体系,不仅为应用程序提供了一个通用的方法来管理缓存,而且很好地利用了 Java 技术的平台无关性,可移植到不同的应用程序和平台运行。

2 PCM-JA 模型总体架构

PCM-JA 模型如图 1 所示,它由缓存接口、缓存管理器、Cache、条目、截止策略五个核心模块组成。其中缓存接口主要负责与应用程序和缓存系统进行交互,它很好地封装了缓存模型的内部结构,对于应用程序和缓存系统是透明的,应用程序只需要调用这个模型的缓存接口就可以获得模型的使用。缓存管理器是模型的核心,它统一管理调度各个 Cache,可以使 Cache 的使用效率更高。在这样的架构中,各个模块都有自己的从属关系,分工明确,便于配置与使用,各模块的具体功能如下。

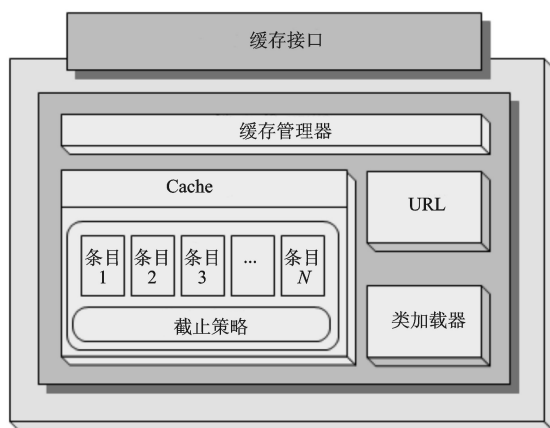


图 1 PCM-JA 模型总体架构图

2.1 缓存接口

缓存接口提供了一个创建、获得、配置、管理和控制一个或多个缓存管理器的机制。开发者通过缓存接口与缓存管理器交互,继而与 Cache 进行交互。

缓存接口提供的具体方法如下:

- (1) 一个获得默认的缓存管理器实例方法;
- (2) 一个创建缓存管理器实例的方法,这个实例由特定的 URI(Uniform Resource Identifier 统一资源标识符)唯一标识识别;
- (3) 一个由类装载机管理缓存管理器实例的方法;
- (4) 一个关闭和释放缓存管理器的方法;
- (5) 一个查询缓存管理器性能的方法。

2.2 缓存管理器

缓存管理器是整个缓存模型的管理中心,开发者通过缓存管理器来管理具体的缓存。

缓存管理器具有以下功能:

- (1) 创建、配置和查询一个唯一标识的 Cache;
- (2) 关闭 Cache 使得它不再被管理;
- (3) 销毁一个 Cache,包括它包含的所有条目;
- (4) 规定一个 Cache 将要使用的类加载器,这个类加载器用来加载应用程序的类;
- (5) 关闭缓存管理器和它当前所管理的所有 Cache;
- (6) 获得缓存接口对于缓存管理器的配置信息;
- (7) 查询缓存接口所支持的功能。

2.3 Cache

Cache 在应用程序的设计中,它通常用来描述一种开发者使用的单独的内存和低时间延迟的数据结构,来暂时存储应用程序后续步骤要用到的数据或者信息,以减少应用程序频繁访问数据库服务器的开销。

Cache 是整个缓存模型的数据处理中心,它主要进行缓存数据的读取以及截至策略的执行。

Cache 同时是一个映射类的数据结构,它提供映射的方法来访问、更新、删除缓存条目。为应用程序的数据提供暂时的存储空间以便与应用程序进行高速的数据交互。它将每个条目以键值对的形式存储以便于查询,并且允许使用 Java 范型来提高应用程序的类型安全。

下面是一个使用 Cache 的简单例子:

```
String cacheName = "cacheName";
CachePort port = Cache.getCacheingPort();
CacheManager manager = port.getCacheManager();
cache = manager.getCache
(cacheName,Integer.class,Date.class);
Date value1 = new Date();
Integer key = 1;
Cache.put(key,value1);
Date value2 = cache.get(key);
```

2.4 缓存条目与条目处理器

缓存条目就是以键值对存储在 Cache 中的数据,缓存中的条目是由条目处理器来处理的。条目处理器使应用程序可以检查缓存条目的值、计算新的值、更新条目或者自动地返回其他的值。条目处理器配置的具体程序如下:

```
CachingPort port = Caching.getCacheingPort();
CacheManager Manager = provider.getCacheManag-
er();
MutableConfiguration < String,Integer >
configuration =
```

```

new MutableConfiguration<String,Integer>()
etTypes(String.class, Integer.class);
Cache<String,Integer> cache = manager.configureCache(
"cache",configuration);
String key = "counter";
Cache.put(key,1);

```

2.5 截止策略模块

当 cache 中的条目没有被命中并且容量已经满了时,就需要在 cache 中踢出一个旧的条目,加入一个新的条目,而到底应该踢出哪个条目,就由缓存替代策略决定. 鉴于此,作者提出了一种简单的替代策略,即截止策略.

3 截止策略

缓存策略主要考虑三个方面的因素,即缓存内容、何时缓存和缓存替换.

不同系统的数据访问模式不尽相同,同一种缓存策略难以在各种数据访问模式下均取得满意性能^[3-5]. 由于本文重点在于缓存模型的设计,因此提出了一种简单的缓存策略,即截止策略.

3.1 截止策略算法基本思想

截止时间定义为 cache 存储的条目持续存在于 cache 的时间. 在这段时间内,条目可以被访问、更新和删除,一旦一个条目的截止时间过期,则称这个条目被截止. 此时,被截止的条目还存在于 cache,但是不能再被访问、更新和删除.

3.2 截止策略的算法设计

在 Cache 中的条目有 5 种状态,即创建、修改、访问、接触和永久存在,其中永久存在是默认的状态,它表示 Cache 中的条目永不被截止.

截止策略的截止时间由截止策略的配置和 Cache 操作共同决定. 以下截止策略的方法是基于缓存操作来定义的合适的持续时间的:

getExpiryForCreatedEntry() 当一个条目被创建后的截止时间;

getExpiryForAccessedEntry() 当一个条目被访问后新的持截止时间;

getExpiryForModifiedEntry() 当一个条目被修改后新的截止时间.

当以上方法被调用,截止策略将会返回下面之一的结果:

- (1) 一个和 cache 等价配置的截止时间;
- (2) Duration.ZERO 表明条目现在将考虑被截

止了;

getExpiryForAccessedEntry() 和 getExpiryForModifiedEntry

可能会返回空值,这表明截止策略应该使得缓存条目的截止时间不变.

示例程序:

```

Publicclass UsernameExpiryPolicy implements
ExpiryPolicy<String, Username>
{
@Override
Public Duration getExpiryForCreatedEntry (Entry<
string extends String, username extends Username>
entry) {
return THIRTY_MINUTES;
}
//新建的缓存条目返回一个配置的截止时间,缓存条目的
//截止时间将会重新开始计算
@Override
public Duration getExpiryForAccessedEntry (Entry<
string extends String, username extends Username>
entry) {
return null;
}
//返回一个空值,表示截止时间不变
@Override
public Duration getExpiryForModifiedEntry (Entry<
string extends String, username extends Username>
entry) {
Username u = entry.getValue();
return c.isClosed() ? ZERO : THIRTY_MINUTES;
}
//根据 u 所获得的值进行判断.
}
//当 username 关闭了,用户可以立即截止它. 此外,用户
//还可以给它另外的 30 分钟,但是必须要把 Username 立即
//放回 Cache 中.
}

```

4 缓存模型的设计

4.1 缓存模型 PCM-JA 在 Web 系统中的工作原理

缓存模型 PCM-JA 在 web 访问中的架构图如图 2 所示. 客户端应用程序发出一个请求后,首先是由缓存模型接收请求并进行相应的处理. 缓存模型与缓存系统交互后,将需要发送的信息传递给服务器,服务器处理请求,响应一个结果传给缓存模型,缓存模型与缓存系统交互,一起组成响应返回给客

户端.

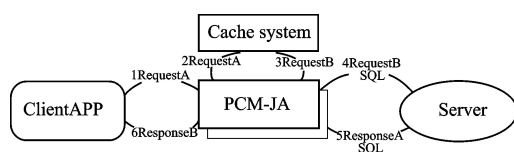


图2 PCM-JA模型在Web访问中的架构图

4.2 缓存模型的配置

本文应用提出的缓存模型创建了一个默认的缓存管理器. 缓存管理器配置了截止策略, 并给出了cache的实例“exampleCache”的配置过程. 这个Cache有一个String类型的键和一个整型的值, 然后还配置了一个小时的截止时间, 最后给定有一个输入和输出.

程序如下:

```
//首先获得缓存接口
CachingPort cachingPort =
Caching.getCachingPort();
//然后从缓存接口中获得缓存管理器, 并进行配置
CacheManager cacheManager =
cachingPort.getCacheManager();
MutableConfiguration<String, Integer> config =
new MutableConfiguration<String, Integer>();
config.setStoreByValue(false);
.setTypes(String.class, Integer.class);
//配置截止策略, 截止时间为一个小时
.setExpiryPolicyFactory
(AccessedExpiryPolicy.factoryOf(ONE_HOUR));
.setStatisticsEnabled(true);
//缓存管理器配置 Cache
cacheManager.configureCache("exampleCache", config);
Cache<String, Integer> cache =
cacheManager.getCache("exampleCache",
String.class, Integer.class);
String key = "key";
Integer value1 = 1;
//值键为“key”, 值为“1”
cache.put("key", value1);
Integer value2 = cache.get(key);
```

5 仿真实验与分析

5.1 实验设置

本文为一个基于Java EE的在线选课系统植入缓存模型PCM-JA, 并使用JMeter进行性能测试. 在测试中定义了10种不同的动态交互请求, 其中约有80%的请求为只读请求. 测试的服务器为 Sugon

I620r-G, Web 服务器为 IIS 6.0, 应用服务器为 ClassFish 4.0, 数据库为 MySQL 5.6.10, 运行平台为 NetBeans 7.3.1.

5.2 实验结果与分析

为了验证 PCM-JA 的性能, 本文主要测试在线选课系统使用 PCM-JA 前后的平均响应时间以及缓存命中率.

使用 PCM-JA 前后, 系统的平均响应时间如图3所示. 随着并发访问数不断增加, 未使用 PCM-JA 前时, 系统响应时间上升较快, 而使用 PCM-JA 的系统响应时间平稳得多, 在并发数达到1100时, 使用 PCM-JA 的系统响应时间减少了56%左右. 可见, 在并发用户较多的Web应用系统中, 使用 PCM-JA 可以很好减少响应时间.

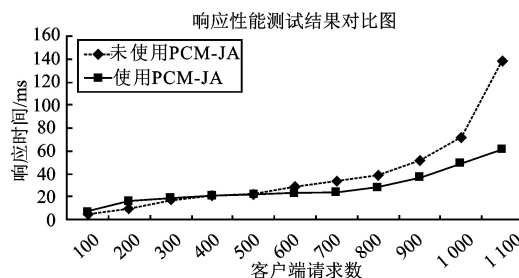


图3 平均响应时间

使用 PCM-JA 模型时的缓存命中率如图4所示. 当缓存大小大于640 kB时, 缓存命中率开始大幅度提升, 当缓存大小达到10 MB时, 缓存命中率为41.2%, 达到了较高的水平.

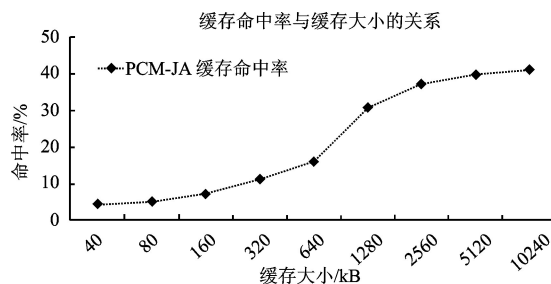


图4 使用 PCM-JA 时的缓存命中率

6 结束语

本文针对目前Java应用程序的缓存问题, 提出了一种基于Java的可移植缓存模型PCM-JA, 并给出了模型的架构图和相应的算法设计. 本模型为Java应用程序提供了一个通用的方法来操作Cache中的数据, 它将应用程序和缓存系统分开, 可以快速、简便的为其他应用程序搭建合适的缓存平台, 解

(下转第108页)

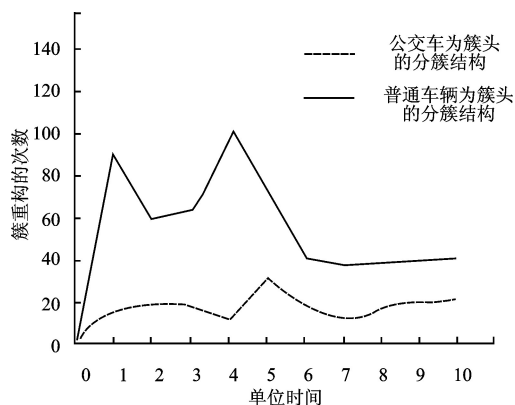


图1 簇重构次数

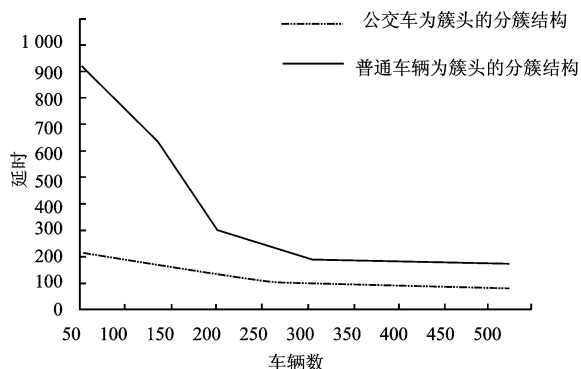


图2 延时

方法,重点研究了簇头的选举算法、簇成员的动态管理等问题,实验结果表明,本文的方法不仅在簇稳定性方面具有优势,也可以有效降低通信时延。

此外,本文提出的借助路边停靠车辆实现消息的缓存转发的策略,与使用路边基础设施的方法相比,既有效节约了部署成本,也能显著提高车辆在稀疏区域的通信质量、减少消息的丢失和时延。下一步研究的重点是高速公路环境下的组网技术和网络流通性问题。

参考文献:

- [1] 常促宇,向勇,史美林. 车载自组网的现状与发展[J]. 通信学报, 2007, 28(11): 116-126.
- [2] Vehicular Technology Conference Fall [C] // VTC 2010-Fall. Ottawa: IEEE, 2010: 1-5.
- [3] Fan, Peng, James Haran, John Dillenburg, et al. Traffic model for clustering algorithms in vehicular ad-hoc networks [C] // Proc of CCNC. [s. l.]: IEEE, 2006: 168-172.
- [4] J Eriksson, H Balakrishnan, SMadden. Cabernet: vehicular content delivery using wifi [C] // Proceeding Mobicom 08' Proceedings of the 14 th ACM international conference on Mobile computing and networking. USA, New York: ACM, 2008: 199 - 210.
- [5] Y Zhang, J Zhao, G Cao, Service scheduling of vehicle-roadside data access [J]. Mobile Networks and Applications, 2010, 15(1): 83 - 96.
- [6] Chen, Bin Bin, Mun Choon Chan. Mobtorrent: A framework for mobile internet access from vehicles [C] // INFOCOM. Rio de Janeiro: IEEE, 2009: 1404 - 1412.
- [7] N Liu, M Liu, W Lou, et al. Pva in vanets: Stopped cars are not silent [C] // INFOCOM. Shanghai: IEEE, 2011: 431 - 435.

作者简介:

沈明玉 男, (1962-), 博士, 副教授, 硕士生导师. 研究方向为网络与信息安全。

彭盼 女, (1989-), 硕士研究生. 研究方向为车载自组网的组网技术及相关技术。

赵培 女, (1986-), 硕士研究生. 研究方向为无线传感器网络数据存储与访问以及相关技术。

(上接第 104 页)

决了现有 Java 应用程序临时缓存的兼容性和安全性问题. 实验表明, PCM-JA 模型在多个性能指标上均具有较好的效果。

参考文献:

- [1] 邓磊, 陈志刚, 黄键, 等. 基于 AOP 的智能 Web 缓存框架 [J]. 计算机工程, 2008, 34 (22): 283-285.
- [2] 张艳, 石磊, 卫琳. Web 缓存优化模型研究 [J]. 计算机工程, 2009, 35 (8): 85-87.
- [3] 黄敏, 蔡志刚. 缓存替换算法研究综述 [J]. 计算机科学, 2006, 33 (B12): 191-193.

[4] 林永旺, 张大江, 钱华林. Web 缓存的一种新的替换算法 [J]. 软件学报, 2001, 12(11): 1710-1715.

[5] PAN Peng, LU Yansheng. Monitoring nearest neighbor queries with cache strategies [J]. Journal of zhejiang university science, 2007, 8(4): 529-537.

作者简介:

黄学雨 男, (1970-), 硕士, 副教授. 研究方向为企业信息系统、视觉工程技术。

张茂新 男, (1988-), 硕士研究生. 研究方向为企业信息系统。