

基于面向方面和运行时反射技术的并行框架

张 杨^{1*}, 张冬雯¹, 王一拙²

(1. 河北科技大学 信息科学与工程学院, 石家庄 050000; 2. 北京理工大学 计算机学院, 北京 100081)

(* 通信作者电子邮箱 zhangyang@hebut.edu.cn)

摘 要:针对使用并行库 JOMP 的程序在性能方面存在的不足, 提出一个可以分离并行逻辑和功能逻辑的并行框架。该框架对程序中需要并行处理的部分进行标记, 采用面向方面和运行时反射技术实现被标记部分的处理, 其中面向方面技术用于实现并行逻辑的分离和编织, 运行时反射技术用于获取运行时被标记部分的相关信息, 以并行库 (waxberry) 的方式实现了该并行框架。使用基准测试程序 JGF 套件中的三个测试程序对并行库进行了测试, 实验结果表明, 应用该并行库的程序可以获得较好的性能。

关键词:并行编程; 面向方面编程; 反射; 并行框架; 并行库

中图分类号: TP311.1 **文献标志码:** A

Parallel framework based on aspect-oriented programming and run-time reflection

ZHANG Yang^{1*}, ZHANG Dongwen¹, WANG Yizhuo²

(1. School of Information Science and Engineering, Hebei University of Science and Technology, Shijiazhuang Hebei 050000, China;

2. School of Computer, Beijing Institute of Technology, Beijing 100081, China)

Abstract: JOMP that is the OpenMP-like implementation in Java needs to be optimized, so a parallel framework, which can separate parallel logic and logic function, was proposed. The parallel framework was implemented by a parallel library named waxberry, and the parts which need to be processed parallelly were annotated and executed by using Aspect-Oriented Programming (AOP) and run-time reflection. AOP was used to separate parallel parts with core ones, and to weave them together. Run-time reflection was used to obtain the related information during the parallel execution. The library waxberry was evaluated using Java Grande Forum (JGF) benchmarks on a quad-core processor. The experimental results show that the waxberry can obtain good performance.

Key words: parallel programming; aspect-oriented programming; reflection; parallel framework; parallel library

0 引言

并行框架和编程规范可以对并行程序设计过程进行规范和约束, 有利于简化并行程序的开发过程和提高并行程序设计的效率。

并行程序的开发是一项非常复杂的工作^[1-2], 并行相关技术已经在很多领域得到了应用^[3]。XJava^[4]探索了流的并行程序设计模型, 文献[5]探索了 CUDA 上的并行编程模型, 并且出现了专为计算机模拟的并行编程模型^[6]。文献[7]探讨了多核平台上并行编程模型的表达策略。

OpenMP^[8]是共享内存的并行编程规范, 通过向程序中添加预编译指令, 可以实现程序的并行化。JOMP^[9]是面向 Java 语言的 OpenMP 规范的实现, 通过向 Java 源程序中加入 JOMP 并行指令, 在 JOMP 编译器的辅助下可以实现 Java 程序的并行执行。使用 JGF (Java Grande Forum) 基准测试程序^[10]对 JOMP 的性能进行了测试, 给出了程序分别使用 JOMP 和使用线程的运行结果, 如图 1 所示, 这里选择了基准测试程序 JGF 套件中的加密程序 Crypt、超松弛迭代程序 SOR 和稀疏矩阵乘法程序 SparseMatMul, 此测试中三个程序的数据规模均为

SizeA, 这些程序的测试结果均在 4 线程情况下完成。从图 1 可看出, 三个程序使用 JOMP 的运行时间均比使用线程的执行时间多, 这表明 JOMP 在提升程序性能方面还需要作进一步的工作, 然而, 由于 JOMP 的代码没有公开, 因此很难定位产生这种现象的原因。

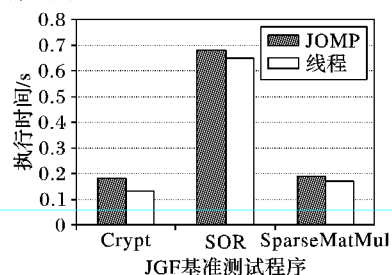


图 1 使用 JOMP 和使用线程的 JGF 测试程序性能比较

针对使用 JOMP 的程序在性能方面存在的不足, 本文提出了一个并行框架, 该框架将程序中并行逻辑作为横切关注点, 实现了并行逻辑和程序功能逻辑之间的分离。依据该并行框架, 本文实现了一个并行库 waxberry。使用 JGF 基准测试程序套件中的三个程序对 waxberry 并行库进行了测试, 验证该框架的有效性。

收稿日期: 2014-06-05; 修回日期: 2014-08-15。 基金项目: 河北省自然科学基金资助项目 (F2012208018)。

作者简介: 张杨 (1980 -), 男, 河北抚宁人, 讲师, 博士, CCF 会员, 主要研究方向: 软件重构、并行程序设计; 张冬雯 (1964 -), 女, 河北石家庄人, 教授, 博士, 主要研究方向: 软件工程; 王一拙 (1978 -), 男, 陕西西安人, 讲师, 博士, 主要研究方向: 并行编程模型。

1 并行框架

1.1 框架概述

并行性作为一种典型的横切关注点^[11],可以从程序的核心功能中分离出来,这对于提高程序的模块化程度是有好处的。对于核心功能部分,通过标记表明需要并行处理的部分;对于并行逻辑部分,使用相关的并行化方法实现并行处理。在框架中,需要并行处理部分的标记任务是由程序设计人员完成,被标记的部分将被自动抽取,并封装为若干个任务,这些任务将被放入一个任务队列中等待执行。

在运行时刻任务封装的过程中,运行时反射用于获得任务的相关信息,例如,在对一个对象的方法进行封装时,需要获得对象名、方法名和方法参数等信息,这些都可以通过反射获得。

每一个任务对应一个组,任务可以分为组内任务和组间任务,组内任务之间不存在相互依赖关系,适于并行处理。组和组之间存在运行时间先后的依赖关系,只有当前一个组的任务完成之后才进行下一个组的任务处理。

在并行处理过程中,该并行框架创建并维护了一个线程池,主要负责线程的创建、线程障碍和同步操作等。该线程池将不断从任务队列中取出需要处理的任务,组与组之间的执行先后关系通过线程障碍操作进行处理。

并行逻辑和核心功能逻辑之间通过编织组合到一起,形成最终的并行程序。并行框架如图 2 所示。

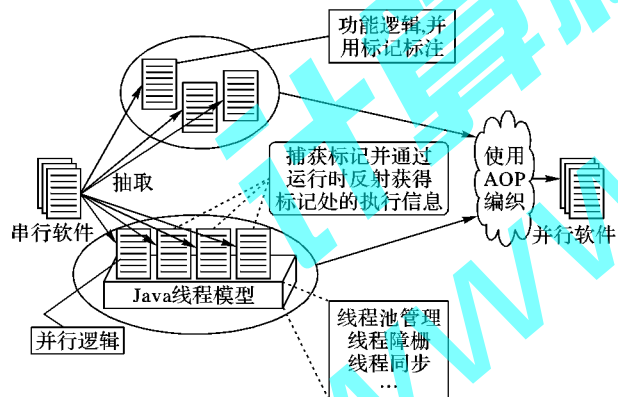


图 2 并行框架

1.2 框架实现

依据该并行框架,本文设计并实现了一个并行库,命名为 waxberry。

为了对程序中需要并行处理的部分进行标记, waxberry 首先定义了若干标记,这些标记通过 Java 语言中的注释接口来实现。

定义的标记主要有三个,分别是 @Parallel、@WaitForFinishing、@CleanAfterExecution,以 @Parallel 为例,定义的代码如下:

```
1) @Target(ElementType.METHOD)
2) @Retention(RetentionPolicy.RUNTIME)
3) public @interface Parallel {
4) String group() default "";
}
```

标记 @Parallel 主要用于对程序中需要并行处理的对象方法进行标记,其中 group(行 4)用于指明该任务所属的分

组。对于那些需要并行但又没有用方法实现的程序段,可以通过 Eclipse 开发工具中菜单项“refactoring”中的“Extract Method...”功能实现对对象方法的重构。@WaitForFinishing 和 @CleanAfterExecution 定义的形式和 @Parallel 的定义相同,不再赘述。

对于程序中被标识的对象方法,通过面向方面编程^[11](Aspect-Oriented Programming, AOP)技术进行处理。面向方面编程技术已经被广泛地应用于并行程序的设计中,文献[12]给出了面向方面技术的设计、语义和实现,并探讨了面向方面在并行编程上的应用。

首先,通过切点(Pointcut)来捕获这些需要处理的标记;然后,通过自定义的方面(Aspect)处理被标记的方法,从而实现了并行逻辑的分离。

在并行库 waxberry 中,针对不同的标记,定义了不同的切点,对于切点的定义如下:

```
pointcut parallelPointcut(): call(@Parallel * * . * (..));
```

该切点可以捕获所有被 @Parallel 标记的方法。其他两个标记的切点定义与 @Parallel 的切点定义类似。

切点处的执行动作通过通知(Advice)定义,例如,对于切点 parallelPointcut 的处理如下面代码段所示:

```
Object around(Parallel parallel): parallelPointcut() &&
@annotation(parallel) {
    parseParallelAnnotation(parallel);
    // 对 @paralle 标记进行解析
    // 获得当前方法的对象、方法名、参数等
    // 封装为任务
    // 在任务和组之间进行映射
}
```

在面向方面技术中,通常可以定义三类通知,分别是 before、around 和 after 通知。对不同的切点处理采用了不同的通知处理方法,例如,对切点 parallelPointcut 的处理使用了 around 通知,该通知主要完成对 @Parallel 的解析、将被标记的方法封装为任务、并将任务映射到所属的组。

在将方法封装为任务的过程中, waxberry 使用了运行时反射技术,通过反射获得被标记的对象方法的对象引用、方法名和方法参数等,这些运行时反射技术可以通过 AOP 中的 thisJoinPoint 的相关方法获得。

对 @WaitForFinishing 标记处理的方面中使用了 around 通知,主要用于对线程的创建、运行和线程障碍操作进行维护,线程障碍主要用于等待某一个组任务的完成。该 around 通知中主要使用了 Java 线程模型,根据被标记的方法是否有返回值,决定采用 Runnable 还是 Callable。在 waxberry 并行库中,根据可运行的最大线程数,将维护一个可以执行固定线程数的线程池,生成的 Runnable 或者 Callable 对象将放入线程池中执行,线程池维护线程的创建、运行和销毁的过程。

对 @CleanAfterExecution 标记处理的方面中定义了 before 通知和 after 通知。其中, before 通知主要用于获得程序运行的参数; after 通知用于在程序执行结束后进行一些必要的清理工作,例如关闭线程池、内存回收等。

1.3 waxberry 的优势

使用 waxberry 的优势在于:

- 1) 实现了同步逻辑和系统功能逻辑之间的分离。
- 2) 实现了类似于 OpenMP 的标记,在没有面向方面代码

的情况下,这些标记对于程序的执行没有影响。

3) 将串行程序转变为并行程序只需要添加若干标记,简单易行。这里需要指明的是, waxberry 可以将程序转换为多线程的并行程序,但是任务的划分还是由程序员完成。

2 实验与分析

2.1 实验环境

硬件:处理器为 Intel Q8200 四核处理器,主频 2.33 GHz,最大支持 4 线程,内存 4 GB。

软件:Linux 2.6.38, JDK 1.7.0_17。Eclipse 作为 waxberry 并行库的开发和测试环境, ajdt_2.2.3 插件作为面向方面编程的工具。

2.2 测试数据

选择 JGF 基准测试套件^[10]中的程序进行了测试,该测试套件包括了若干个应用测试程序,典型的应用程序包括加密 Crypt、超松弛迭代 SOR、稀疏矩阵乘 SparseMatMul 等。其中,加密 Crypt 和稀疏矩阵乘 SparseMatMul 这两个程序在并行化过程中对程序的数据大小根据线程数划分为若干个任务,多个任务并行处理;超松弛迭代 SOR 程序的并行化是对程序的迭代次数进行划分。

这些测试程序根据数据规模的大小分为三个测试点,分别为 SizeA、SizeB 和 SizeC。程序的某些测试点数据量过大,这有可能导致 Java 堆内存的溢出,对于那些数据量大的程序,测试时使用了 -Xmx 参数,并指明该参数的值为 -Xmx1024m,以防止堆内存的溢出。

2.3 实验结果

本节给出了 waxberry 并行库的性能测试结果,每一个测

试点都是在运行 5 次的基础上求平均值所得,所有的测试评测了程序的执行时间随线程数的变化情况,显然,执行时间越少越好。本文将 waxberry 的执行结果和 JOMP 的执行结果进行了对比。

加密程序 Crypt 的测试结果如图 3 所示。图 3(a) 是 Crypt 程序 SizeA 的测试结果, waxberry 的测试结果要明显优于 JOMP 的测试结果,在 4 线程情况下,程序的执行时间最少,此后,在 8 线程和 16 线程情况下,执行时间比 4 线程的执行时间略有增加,但 waxberry 的执行时间不如使用 JOMP 的执行时间增加明显,说明使用 waxberry 获得了较好的性能。在数据规模为 SizeB 和 SizeC 情况下,在小于或等于 4 线程情况下, waxberry 和 JOMP 的执行时间基本相同,当线程数大于 4 时, waxberry 的执行时间比 JOMP 的执行时间要略少,说明使用 waxberry 可以获得较好的性能。

超松弛迭代程序 SOR 的测试结果如图 4 所示。当数据规模为 SizeA 时(如图 4(a)所示),使用 waxberry 的执行时间在线程数不小于 4 时要优于使用 JOMP 的执行时间。当数据规模为 SizeB 时并且线程数为 4 时,使用 waxberry 的执行时间比使用 JOMP 的执行时间要少。当数据规模为 SizeC 时,二者的执行时间比较接近。

稀疏矩阵乘 SparseMatMul 的测试结果如图 5 所示,其中,数据规模为 SizeA 和 SizeB,线程数为 1 和 2 时, JOMP 和 waxberry 的运行时间基本相同,当线程数大于或等于 4 时,使用 waxberry 的程序性能要优于使用 JOMP 的程序性能。当数据规模为 SizeC 时,使用 JOMP 和使用 waxberry 都出现了随着线程数增多而执行时间下降的现象,出现这种现象可能的原因是 SizeC 本身数据量较大,随着线程数的增多,数据被划分

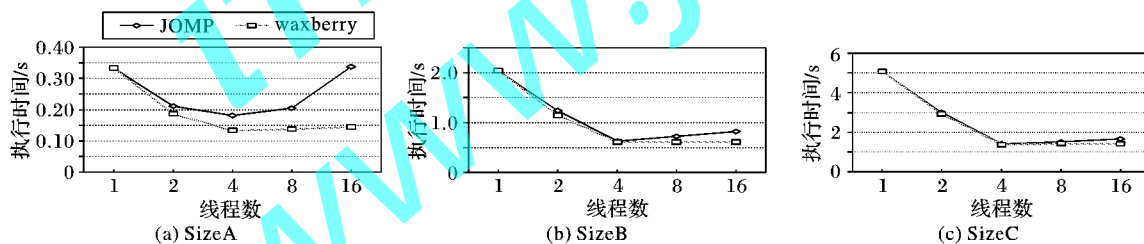


图3 Crypt 程序执行时间

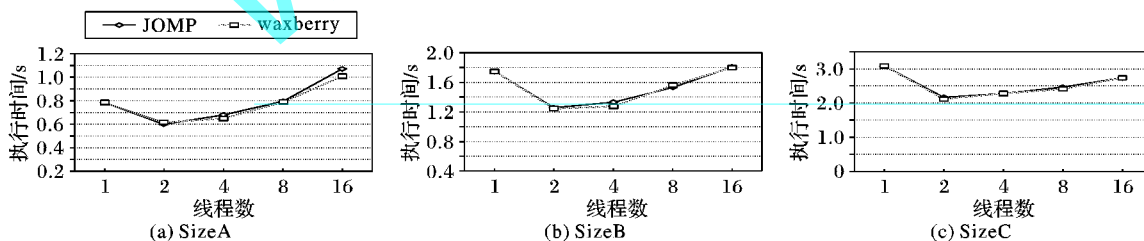


图4 SOR 程序的执行时间

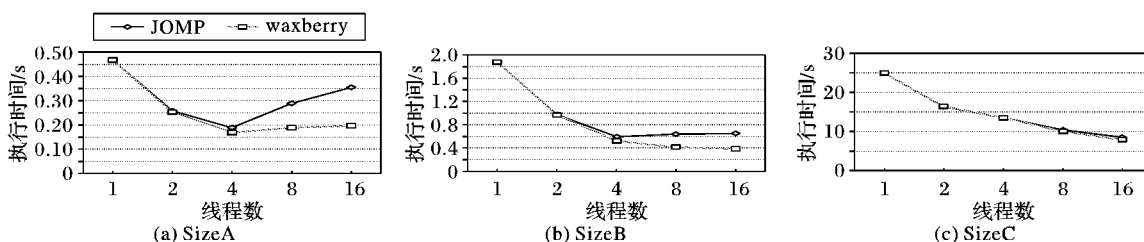


图5 SparseMatMul 程序的执行时间

得越细,这种数据量在多核处理器上可以较快地完成。

通过这三个程序的实验测试表明, waxberry 在执行上可以获得较好的性能,并且性能要优于 JOMP。

3 结语

本文针对 JOMP 在提升程序性能方面存在的不足,提出了一个并行框架,该框架可以分离并行程序中的并行逻辑,采用面向方面和运行时反射技术以并行库的方式实现了该并行框架。在实验中,使用 JGF 基准测试程序对并行库的性能进行了测试,并与 JOMP 进行了对比分析,实验结果表明本文设计的并行库在性能上要优于 JOMP。

参考文献:

- [1] WILKINSON B, ALLEN M. Parallel programming[M]. Upper Saddle River: Prentice Hall, 1999.
 - [2] STAS N, MIHAI C, DANNY D, *et al.* Mining fine-grained code changes to detect unknown change patterns[C]// ICSE 2004: Proceedings of the 36th International Conference of Software Engineering. New York: ACM Press, 2014: 803 – 813.
 - [3] LIU X, YUE L, CHEN B, *et al.* Method of target recognition in remote sensing images based on parallel processing[J]. Journal of Computer Applications, 2007, 27(9): 2123 – 2125. (刘晓沐, 岳丽华, 陈博, 等. 遥感图像目标识别的并行处理方法[J]. 计算机应用, 2007, 27(9): 2123 – 2125.)
 - [4] OTTO F, PANKRATIUS V, TICHY W F. XJava: Exploiting parallelism with object-oriented stream programming[C]// Proceedings of the 15th International Euro-Par Conference, LNCS 5704. Berlin: Springer-Verlag, 2009: 875 – 886.
 - [5] KOSTOPOULOS S, GLOTSOS D, SIDIROPOULOS K. A pattern recognition system for prostate mass spectra discrimination based on the CUDA parallel programming model[J]. Journal of Physics: Conference Series, 2014, 490(1): 120 – 144.
 - [6] SMITH B. A new parallel programming model for computer simulation[EB/OL]. [2014-02-10]. <http://www.mcs.anl.gov/papers/P5135-0414.pdf>.
 - [7] NOBRE R, PINTO P, CARVALHO T, *et al.* On expressing strategies for directive-driven multicore programming models[C]// PARMA-DITAM 2014: Proceedings of the 2014 Workshop on Parallel Programming and Run-Time Management Techniques for Many-core Architectures and Design Tools and Architectures for Multicore Embedded Computing Platforms. New York: ACM Press, 2014: 7 – 14.
 - [8] DAGUM L, MENON R. OpenMP: an industry standard API for shared-memory programming[J]. Computational Science and Engineering, 1998, 5(1): 46 – 55.
 - [9] BULL J M, KAMBITES M E. JOMP — an OpenMP-like interface for Java[C]// Proceedings of the ACM Conference on Java Grande. New York: ACM Press, 2000: 53 – 61.
 - [10] SMITH L A, BULL J M, OBRIZALEK J A. Parallel Java Grande benchmark suite[C]// Proceedings of the 2006 ACM/IEEE Conference of Supercomputing. New York: ACM Press, 2006: 6 – 13.
 - [11] KICZALES G, LAMPING J, MENDHEKAR A, *et al.* Aspect-oriented programming[C]// Proceedings of the 1997 Europe Conference of Object-Oriented Programming, LNCS 1241. Berlin: Springer-Verlag, 1997: 220 – 242.
 - [12] TANTER E, FIGUEROA I, TABAREAU N. Execution levels for aspect-oriented programming: design, semantics, implementations and applications[J]. Science of Computer Programming, 2014, 80(2): 311 – 342.
-
- (上接第 3095 页)
- [2] LIU X, HAN J, ZHONG Y, *et al.* Implementing WebGIS on Hadoop: a case study of improving small file I/O performance on HDFS[C]// CLUSTER 2009: Proceedings of the 2009 IEEE International Conference on Cluster Computing and Workshops. Piscataway: IEEE Press, 2009: 1 – 8.
 - [3] JIANG L, LI B, SONG M. The optimization of HDFS based on small files[C]// IC-BNMT 2010: Proceedings of the 3rd IEEE International Conference on Broadband Network and Multimedia Technology. Piscataway: IEEE Press, 2010: 912 – 915.
 - [4] DONG B, ZHENG Q, TIAN F, *et al.* An optimized approach for storing and accessing small files on cloud storage[J]. Journal of Network and Computer Applications, 2012, 35(6): 1847 – 1862.
 - [5] GOHIL P, PANCHAL B. Efficient ways to improve the performance of HDFS for small files[J]. Computer Engineering and Intelligent Systems, 2014, 5(1): 45 – 49.
 - [6] CHEN J, WANG D, FU L, *et al.* An improved small file processing method for HDFS[J]. International Journal of Digital Content Technology and its Applications, 2012, 6(20): 296 – 304.
 - [7] HUA X, WU H, LI Z, *et al.* Enhancing throughput of the Hadoop distributed file system for interaction-intensive tasks[J]. Journal of Parallel and Distributed Computing, 2014, 74(8): 2770 – 2779.
 - [8] CHANDRASEKAR S, DAKSHINAMURTHY R, SESHAKUMAR P, *et al.* A novel indexing scheme for efficient handling of small files in Hadoop distributed file system[C]// ICCCI 2013: Proceedings of the 2013 International Conference on Computer Communication and Informatics. Piscataway: IEEE Press, 2013: 1 – 8.
 - [9] MACKEY G, SEHRISH S, WANG J. Improving metadata management for small files in HDFS[C]// CLUSTER 2009: Proceedings of the 2009 IEEE International Conference on Cluster Computing and Workshops. Piscataway: IEEE Press, 2009: 1 – 4.
 - [10] HAO Q, ZHU M, HAO J. New proxy cache replacement policy[J]. Journal of Computer Research and Development, 2002, 39(10): 1178 – 1185. (郝沁汾, 祝明发, 郝继升. 一种新的代理缓存替换策略[J]. 计算机研究与发展, 2002, 39(10): 1178 – 1185.)
 - [11] SHI L, MENG C, HAN Y. Web replacement policy based on prediction[J]. Journal of Computer Applications, 2007, 27(8): 1842 – 1845. (石磊, 孟彩霞, 韩英杰. 基于预测的 Web 缓存替换策略[J]. 计算机应用, 2007, 27(8): 1842 – 1845.)
 - [12] JIN Z, ZHANG G. Intelligent prefetch and cache techniques based on network performance[J]. Journal of Computer Research and Development, 2001, 38(8): 1000 – 1004. (金志刚, 张钢. 基于网络性能的智能 Web 加速技术——缓存与预取[J]. 计算机研究与发展, 2001, 38(8): 1000 – 1004.)