文章编号:1673 - 0062(2013)03 - 0061 - 05

多核构架下基于 OpenMP 的 Huffman 压缩算法并行设计

胡 荣,唐琨皓,黄 樱

(湖南工学院 计算机与信息科学学院,湖南 衡阳 421002)

摘 要:本文是对多核程序设计的一种探索,在 OpenMP 模型下以赫夫曼算法为基础设计并行压缩程序. 首先对传统的串行程序进行分析,使应用程序开发人员了解程序行为、发现性能瓶颈、明确优化方向. 再用 OpenMP 的基本结构进行并行程序的设计之后,借助开发工具对并行程序进行优化和调试,得到改进方案. 然后在双核处理器上分别运行并行程序与串行程序,将两者进行性能上的比较,实验结果证明性能得到很大程度地提高.

关键词:OpenMP 模型;并行程序设计;多核;多线程

中图分类号:TP311 **文献标识码**:B

Design of Parallel Huffman Compress Algorithm Based on OpenMP and Multi-core Architecture

HU Rong, TANG Kun-hao, HUANG Ying

(College of Computer and Information Science, Hunan Institute of Technology, Hengyang, Hunan 421002, China)

Abstract: This paper is an exploration of the multi-core programming under such background. A parallel Huffman compress algorithm is developed based on OpenMP programming model. The traditional serial-execution procedures were analyzed, so the application developers can understand the operations in the code, find performance bottlenecks, and work towards the correct directions for code parallelization and optimization. The code based on OpenMP was further optimized and debugged with the help from various developing tools. An experiment platform was built, which is used to evaluate and compare the code performance between the parallelized version and the serial-execution version. The experiment results show that the performance is improved to a large extend.

key words: the model of OpenMP; parallel programming; multi-core; multithread

收稿日期:2013 - 04 - 25

基金项目:湖南省科技厅科研基金资助项目(2011FJ3108;2013GK3036);湖南省教育厅科学研究基金资助项目(12C0653)

作者简介:胡 荣(1978 -),女,湖南衡阳人,湖南工学院计算机与信息科学学院讲师,硕士. 主要研究方向:并行计算,信号处理.

0 引 言

在过去的五十年中,CPU 时钟频率基本上是按照摩尔定律发展的. 但在现有工艺下,提高 CPU 性能的方法等在摩尔定律限制下已经难有很大的进展. 近年来新型芯片性能的提升则主要从超线程、多核和缓存等方面入手,其中最为突出的就是多核技术. 也正是由于多核技术的发展和普及,引领软件研发发生基础性变化. 为了充分利用多核处理器,程序要求同时做更多事情. 大量学者对此进行了研究,并提出了大量的技术和方法,如改进传统锁机制的可编程性事务存储技术[1]、基于对全局地址空间模型进行分割的编程语言[2]以及对现有编程语言 OpenMP[3]的扩展. 其中文献[4]提出了一种改进的指导调度策略,并加以实现,其调度性能要优于其他策略. 文献[5]则从二值化图像细化处理的并行算法上进行了研究,提出了一种改进的 Zhang 并行细化算法. 这些技术方法从不同的层面推动应用软件中并行执行能力的加强,从而使得众核处理器系统使用效率得到提升的目的.

目前在高性能计算领域常用的并行程序的编程模式分为基于消息传递的并行程序(MPI /PVM 程序)和基于共享主存的并行程序设计(OpenMP 程序). OpenMP 是一种共享内存编程的工业标准,具有良好的易用性且可以实现增量并行等特点,从而成为并行程序设计的主流模型之一[6].

本文则将利用 OpenMP 模型设计 Huffman 压缩算法. 通过设计学习多核体系相关理论和并行程序设计的基本结构,探讨影响并行程序性能以及多核处理器利用的一些因素.

1 OpenMP 模型

OpenMP 多线程编程是针对共享内存储器的方法,采用 fork-join 的并行执行模型[7],主要用于指挥多线程、共享内存储器并行. 拥有很好的程序移植性,且支持多样化的语言编程,最初主要应用于共享内存储器的多处理的体系结构设计的并行化编程,它使得应用程序在对称多处理器或多核处理器系统上并行执行时,获得大幅度地性能提高[8],这与采用消息传递并行化编程的模型有很大的不同. 对编程人员来说是让同一个内存设备被多个处理器共享.

随着多核处理器的发展和普及,对于提高多核的利用率,OpenMP 为程序设计者提供了一个重要途径. 开发人员只需要认真考虑哪些代码应该以多线程方式执行,以及如何重构算法以便在多核处理器上获得更好的性能等问题. 当使用 OpenMP 于那些最耗时且能够并发执行的部分时,将极大地提高程序的性能.

2 Huffman 压缩算法串行设计

Huffman(赫夫曼)编码是一种常用的编码方式,而且也是一种无损的压缩算法,常用来压缩文本和程序文件. 为了保证串行程序的正确性,以确保之后并行化的顺利进行,要对串行程序进行测试. 对于用此方法压缩的文件可以用此方法解压回原文件. 完成之后对程序的压缩效果进行了调试,采用此方法设计的程序能对一般的文本文件有较好的压缩能力,对其它格式文件可以进行压缩但不一定能有压缩效果,得出表 1 的结果.

表 1 串行程序压缩结果

Table 1 The compression results of serial program

文件格式	原文件大小/Byte	能否打开	压缩文件大小/Byte	压缩比/%	解压文件能否打开
DOC	234 580	可以	164 675	29.8	可以
TXT	79 341	可以	59 902	24.5	可以
PDF	750 452	可以	737 694	1.70	可以
BMP	1 023 275	可以	152 467	85.1	可以
MP3	2 467 085	可以	1 776 301	0.28	可以
PPT	2 056 752	可以	2 042 971	0.67	可以

根据此结果,基本可以确定该方法对各文件有一定的压缩能力并且可以还原为原文件. 通过此测试,可以认为此程序可以正常工作,对文件的操作基本没有异常,并尽可能多的考虑各种情况,

如文件大小,格式,以及路径等. 在这些情况下,程序是能够正确运行的.

3　Huffman 压缩算法并行设计

3.1　影响并行程序的因素分析

多核处理器现今已越来越广泛地应用到普通用户的各个层面,尤其是客户端要使用多核处理器,而大多数运行于客户端的软件要想发挥出多核的并行性优势一般没有服务器和那些可以进行大型并行化计算的特定领域简单.

如何去衡量并行程序设计所带来的性能效益. 通常,假设把任务划分为若干个不同的子任务,再一起处理这些子任务,则对于大幅度提高性能有很大帮助. 量化衡量并行程序性能的指标就是用最优串行算法的执行时间除以并行程序的执行时间所得到的比值,这就是加速比(speedup),加速比能够描述对程序并行化之后所获得的性能收益[9].

在加速比定义的基础上,当涉及处理核数量和并行比例时,又会考虑多核应用程序所能够获得的性能提升.

$$\text{加速比} = \frac{1}{S + (1 - S)/n} \tag{1}$$

公式(1)中,S 表示执行程序中串行部分的比例,n 表示处理器核的数量.

如果将 n 设为 ∞,且假定最优串行算法的运行时间为 1,则公式(1)就变成了以下的公式. 它指出了串行部分运行时间为 S 的应用系统程序并行化之后能得到的加速比上限为:

$$\text{加速比} = \frac{1}{S} \tag{2}$$

从公式(1)可以看出,要提高并行应用程序的执行效率,可行的方法是提高处理器核的数量并提高程序并行化的比例. 在机器的执行核一定的情况下,提高应用程序的执行效率的途径就是尽量提高并行化程度,这就为软件设计带来了机遇和挑战.

实际上执行效率是有多方面因素决定的,在程序编写完之后,在执行效率达不到预期目标的时候,就要考虑执行期可能产生影响的因素. 在多核机器的执行核一定并且并行比例已定的情况下,在编写 OpenMP 应用程序时一般还考虑其他的一些使效率降低的因素. 如:

1) OpenMP 获得应用程序多线程并行化的能力不是凭空而来的,而是需要一定的程序库的支持. 在运行时的程序库对程序并行加速的同时需要运行并行库的本身,因此 OpenMP 本身的执行会抵消一些并行带来的效率提高.

2)一个应用程序的执行的过程中,有很多的同步点,线程只有在同步点进行同步之后才可以继续执行下面的代码.

3)相对于多线程应用系统程序而言,相对于串行运行程序的一个本质特点就是线程间可能存在的同步开销[10-11].

3.2　并行设计

3.2.1　发现热点代码

在分析并行化的可行性之前,采用 Intel Vtune 性能分析器来发现串行程序的热点代码,找到程序最耗时的部分,然后再利用并行程序的方法讨论这部分代码的并行化问题.

Intel Vtune 性能分析器是一种系统分析器,不仅可以提供进程/任务级的事件取样和调用图等所有可用信息,而且可以提供进程内线程级别的信息. Intel Vtune 性能分析器能够发现程序中的耗时部分,例如模块、函数、线程,甚至源码中某一行,而且不需要应用程序进行特殊的构造.

寻找热点代码是通过取样来实现的. 在建立 Sampling Wizard 项目后,对 Huffman 压缩串行程序进行分析,以 Instruction Retried(执行次数)排序得到的.

3.2.2　并行设计具体实现

根据得到的热点代码采用多线程,从而将任务分配给几个线程并行处理,就能提高应用程序的性能,会减少很多并行的开销. 根据压缩程序压缩数据的过程是边读数据边压缩数据的过程,由于读文件的串行天然属性,我们很难将其并行化. 因此想到数据分解:在压缩文件之前,先读取全部或一部分待压缩数据,然后利用将数据分解到多个线程来压缩,完成之后再将所得到的数据写到压缩文件. 这样,压缩速度将不受读写文件的影响,在多核处理器下能够提高程序的执行效率.

数据分解是将应用系统程序按照各任务所处理的数据来进行分解,而不是按照任务的天然属性来进行分解的一种方法. 总而言之,能够根据数据分解方式来进行分解的程序基本都有多个线程存在,这些线程分别对不同的数据对象执行相同的操作. 这种方法明显比只采用一个线程来执行所有的计算量要更加高效. 数据分解方式所能处理的问题规模随着处理器核数量的增加而增长. 这就意味着采用数据分解之后,在相同的时间内

能够完成更多的工作量.

而采用 OpenMP 技术的最主要的原因就是在多线程下可以提高应用系统程序的运行效率. 实际上运行效率是受多方面因素的影响,在完成编写后,并多次测试执行效果,如果执行效果无法达到预期的目标时,就要对运行期间所有对其性能产生影响的因素进行再次考虑.

3.3 实验结果分析

我们采用双核处理器进行实验测试,实际可在不同类型的多核处理器下运行,操作系统平台采用 Microsoft Vista home basic,编译平台为 Microsoft Visual Studio. Net 2005 及其以上,且编程语言为 C + +.

在实验过程中,采集了大量的数据,并将数据进行整理后利用公式(1)计算压缩每个文件为并行程序时的加速比. 具体结果见表 2. 从结果可知,加速比在 1 ~2 之间,有的甚至接近 2,考虑到实验的是双核的机器,可见并行程序很好的提高了多核处理器的利用效率.

表 2 压缩不同类型文件的执行效果

Table 2 Different types of file compression performance

文件格式	原文件大小/Byte	串行压缩时间	并行压缩时间	加速比/nt
DOC	2 541 568	64 902 079	43 280 383	1.499 573
RAR	2 929 322	106 303 939	69 319 544	1.533 535
BMP	1 255 386	10 083 781	5 740 530	1.756 594
PPT	2 248 706	98 066 708	52 982 434	1.850 929
ZIP	5 898 509	256 360 509	130 350 350	1.966 704
RMVB	228 685 077	9 355 172 792	5 436 501 202	1.720 808
WHT	371 803	12 897 829	7 394 876	1.744 158

表 2 中的数据是针对不同种类型的数据压缩得出的结果,在实验的过程中发现加速比与被压缩文件的大小有一定的关系.

为了使实验结论更为贴近实际,决定测试压缩同一种不同大小的同种类型的文件的加速比,选取的文件格式是 doc 格式的文件. 得出表 3 的结果.

表 3 压缩不同大小数据的执行效果

Table 3 The effects of the implementation of different size of data compression

原文件大小/Byte	串行压缩时间(性能计数器)	并行压缩时间(性能计数器)	加速比/nt
28 672	627 953	708 734	0.886 021
55 808	1 946 879	1 660 326	1.172 588
127 488	4 899 651	4 019 061	1.219 103
235 520	9 573 752	7 203 290	1.329 080
799 744	35 628 716	25 048 727	1.422 376
5 813 248	257 241 465	174 065 957	1.477 839
30 736 384	1 332 636 907	889 077 963	1.498 898
92 175 872	4 078 929 573	2 635 179 836	1.547 875

在表 3 中,显示的是压缩同一种不同大小的同种类型的文件的加速比,当文件过小时,加速比甚至小于 1,即并行程序的性能不如串行程序的性能. 当被压缩的文件大小逐渐增加时,加速比也随着增加,但不会超过极限 2(处理器核的数量). 产生这种情况的原因是:当被压缩文件过小时,使

用 OpenMP 进行并行化后带来的效率不足以抵消 OpenMP 本身引入的开销;并行程序的并行化部分主要在数据的压缩上,随着被压缩文件的增加,并行化比例将会提高,此时加速比就会上升,使用 OpenMP 进行并行化后带来的效率已经远远超过 OpenMP 本身引入的开销. 因此,OpenMP 并行化较适合于具有一定计算规模的程序,只有在这种情况下,才能使得并行化的性能提高远远超过 OpenMP 开销,从而提升总体性能.

4　结　论

本文在 OpenMP 模型下对 Huffman 压缩算法的并行设计进行了大量的研究和实验,并且通过实验采集了大量具体数据,评估出并行化后的性能加速比. 经实验结果得出并行后的加速比得到了很大提高,有些甚至接近处理器核的数量的结论,可见此并行设计很好的提高了多核处理器的利用效率.

参考文献:

[1] Abadi M, Birrell A, Harris T, et al. Semantics of transactional memory and automatic mutual exclusion[J]. ACM Trans. on Programming Languages and Systems, 2011, 33(1): 1-50.

[2] Loh E. The ideal HPC programming language[J]. Communications of the ACM, 2010, 53(7): 42-47.

[3] Bhattacharjee A, Contreras G, Martonosi M. Parallelization libraries: Characterizing and reducing overheads [J]. ACM Trans. on Architecture and Code Optimization, 2011, 8(1): 5-29.

[4] 刘胜飞,张云泉,孙相征. 一种改进的 OpenMP 指导调度策略研究[J]. 计算机研究与发展, 2010, 47(4): 687-694.

[5] 牟少敏,杜海洋,苏平,等. 一种改进的快速并行细化算法[J]. 微电子学与计算机, 2013, 30(1): 53-55.

[6] Chapman B, Jost G, Vander P R. Using OpenMP: portable shared memory parallel programming[M]. 1st ed. Oxford Cambridge: MIT Press, 2007.

[7] 徐小龙,罗克露. 椭圆曲线基点判断算法的多核并行化[J]. 计算机应用研究, 2010, 27(9): 3545-3548.

[8] 周伟明. 多核计算与程序设计[M]. 1 版. 武汉:华中科技大学出版社, 2009.

[9] Akhter S, Roberts J. Multi-core programming: increasing performance through software multi-threading[M]. Li Baofeng Translated. 1st ed. Beijing: Publishing House of Electronics Industry, 2007.

[10] 殷顺昌,赵克佳. 一种基于 POMP 的 OpenMP 程序负载均衡分析方法[J]. 计算机工程与应用, 2006, 42(35): 133-138.

[11] Maronju A, Benini L. Efficient OpenMP support and extensions for MPSoCs with explicitly managed memory hierarchy[C]//Proceedings of the Conference on Design, Automation and Test in Europe, Nice, France, 2009: 809-814.