

基于多核多线程的排序算法优化和实现

钱晓捷¹, 李秀芳²

(郑州大学 信息工程学院, 河南 郑州 450001)

摘 要: 根据各多线程技术和多核特点, 提出了归并排序基于多线程的改进方法, 并采用各排序算法组合的方法, 使用 WIN32 API 多线程编程方法实现了新的算法. 实验结果表明该算法比传统算法效率有很大提升.

关键词: 多核; 归并排序; 多线程; 并行排序算法; 算法优化

中图分类号: TP31

文献标识码: A

文章编号: 1000-7180(2011)01-0116-04

Optimization and Implementation of Sorting Algorithm Based on Multi-core and Multi-thread

QIAN Xiao-jie, LI Xiu-fang

(School of Information Engineering, Zhengzhou University, Zhengzhou 450001, China)

Abstract: According to multi-threading technology and the treats of multi-core architecture, an optimization of merge sort was proposed. Then combined with other sorting algorithms, it was implemented by WIN32 API multi-threaded programming methods. The experimental results showed that the new combined algorithm is more effective than the traditional algorithm.

Key words: multi-core; merge sort; multi-thread; parallel sorting algorithm; algorithm optimization

1 引言

由于具有优良的性价比, 多核处理器已经成为通用处理器市场的主流. 多核的优势在于“降低功耗”, 解决了以往靠主频提高带来的棘手问题; 计算性能更强, 能够满足用户同时进行多任务处理和多任务计算环境的要求.

由于处理器体系架构的根本性变化, 要想充分利用这些处理器, 软件设计就更具挑战性. 多核在传统的科学计算和服务器软件方面的应用并不困难, 因为原有的软件是并行的, 具有天生的并行性. 但是在桌面软件方面, 原有软件大部分是串行的. 此情况下, 急需新的并行编程模型来进行并行软件的开发.

多线程是并行编程模型中的一种. 在多核平台上, 各线程都在相互独立的执行核上并行运行的. 为了能够充分地利用多核 CPU 的性能, 软件开发必

须根据多核的特点有效的利用多线程技术.

2 多核与多线程技术

2.1 多核与多线程

在多任务系统下, 会频繁的产生进程(process)间切换, 进程间切换必须进行内存管理组态的切换, 而这种切换会严重影响执行效能. 为了避免此种切换的效率损耗, 先后出现了这两种技术:

(1) 多线程技术, 为了尽量不进行内存管理组态的切换, 避免切换的效能损耗, 采用线程(Thread), 即轻量级进程, 作为基本调度和分派的单位. 线程是属于进程的, 线程运行在进程空间内, 同一进程所产生的线程共享同一内存空间. 其本身基本上不拥有系统资源, 只拥有一点在运行中必不可少的信息(如程序计数器、一组寄存器和栈), 这样就大大减少了线程间切换的代价. 在单核时代, 程序

是单线程运行,由于 Intel 引入了超线程的技术,使我们还在单核时代就领略了将会在多核平台下成为主流的多线程编程技术。

(2) 多核技术,就是将多个完整的计算引擎(内核)集成在一个处理器芯片中,提供了多个可并行执行的独立处理单元.将要执行的各个任务分配给各个内核,每个内核执行一个任务,如此就可以加快整体的执行效率。

多线程和多核的共同之处就在于它们都是定位于解决计算工作中多任务的协同性^[1]。

2.2 基于多核的多线程划分

多线程程序设计的关键就是挖掘应用程序的并行性,将一个程序分解成多个可以同时执行的部分,然后由不同的线程在不同的内核上执行.将应用程序划分成多个独立的任务,并确定这些任务之间的相互依赖关系的过程就成为分解 (decomposition)^[2].分解的方法有:任务分解、数据分解、数据流分解,如表 1 所示.分解的目的是使得各个处理器都能分配到平衡的工作负载,从而提高运算的并行性。

表 1 三种分解方法

分解方法	说明
任务分解	不同的程序行为采用不同的线程实现,常用于 GUI 应用程序
数据分解	多个线程对不同的数据块执行相同的操作,常用于音频图像处理和科学计算
数据流分解	一个线程的输出作为另一个线程的输入,尤其注意尽量消除启动和排空延迟

任务分解是一种能够简单实现并行执行的方法.能够对多个独立任务进行分类.数据分解也称为数据级并行,可采用多线程对不同的数据对象执行相同的操作.数据流分解根据任务间的数据流动来分解任务的.可根据不同的应用特点采用不同的分解方法。

3 基于多核多线程的排序算法

3.1 传统的几种主要排序算法

表 2 中的简单排序包括插入排序、起泡排序、简单选择排序.快速排序在所有同数量级 ($O(n\log n)$) 的排序方法中,平均性能最好^[3],但是,若数据基本有序时,快速排序出现最坏情况,其时间复杂度为 $O(n^2)$. 归并排序是将两个或两个以上的有序数列组合成一个新的有序数列.与快速排序和堆排序相比,

归并排序的最大特点是,它是一种稳定的排序方法。

表 2 几种传统排序算法

排序算法	时间复杂度	最坏情况
简单排序	$O(n^2)$	$O(n^2)$
快速排序	$O(n\log n)$	$O(n^2)$
归并排序	$O(n\log n)$	$O(n\log n)$

3.2 基于多核的并行排序算法设计

并行的排序算法采用分治再合并的思路.由于排序算法的对象是纯粹的数据集,我们采用数据分解的方法,将整个数据集分成几个数据块.为不同的数据块创建线程,采用平均性能最好的快速排序进行排序.形成多个有序数列.最后采用归并排序将多个有序数列进行合并.基于多核的并行排序算法主要分为分块快速排序部分和合并部分两部分.在合并部分采用了基于多线程优化了的归并排序^[4-5]。

线程划分时需考虑负载均衡问题,以充分利用每个线程.所以要将整个数据集进行平均分块.以下是划分线程的代码片段,使用 windows 线程库 API 进行实现。

```
for(int i=0;i<NUM_THREADS;i++){
    param[i] -> Array=array;
    param[i] -> First=i * NUMBER/NUM_THREADS;//①
    param[i] -> Last = (i + 1) * NUMBER/NUM_THREADS-1;//②
    handle[i] = CreateThread (NULL, 0, QS_Thread,
    (void *) param[i],0,NULL);//③
}
```

由变量 NUM_THREADS 控制要创建的线程个数.由语句 ① 和 ② 将数据块平分成 NUM_THREADS 个数据块.语句 ③ 创建线程.线程函数为 QS_Thread,参数是结构体对象 param[i].接下来需等待所有线程运行完并返回后进行合并.此时需阻塞等待各个线程函数返回:

```
WaitForMultipleObjects(NUM_TFHREADS,hThread,
true,INFINITE);
```

hThread 是创建线程时所创建的线程数组.第三个参数 true 表示要等待所有线程函数返回.第四个参数 INFINIT 表示若未等到所有的线程返回则一直等待。

4 归并算法的改进

合并部分采用了一个新的改进了的归并排序,它是基于多线程的.传统的归并排序在将两个有序

数据序列合并成一个新的有序数据序列时,从两个有序的数列的最小值开始比较.将较小值添加到新的数列,然后进行下一次比较.新的归并方法是建立两个线程,分别从两个有序数列的最小值边和最大值边,同时开始比较合并直至新数列的中间为止,完成整个数列的归并.假设线程 1 和线程 2 同时对数据集 A, B 进行合并,得数据集 C .数据集 A 有 m 个数据,数据集 B 有 n 个数据,那么数据集 C 就有 $m+n$ 个数据. $a_i (i=0,1,2,\dots,m-1), b_j (j=0,1,2,\dots,n-1)$ 和 $c_k (k=0,1,2,\dots,m+n-1)$ 分别表示 A, B, C 中的数据.以下为线程 1 和线程 2 的步骤描述.

线程 1:

(1) 初始化 $i=0, j=0, k=0$. (从 A 和 B 的最小值部分开始比较).

(2) 比较 a_i 和 b_j .将较小者的值付给 c_k ,并将其索引(i 或者 j) 加 1,同时将 k 加 1.

(3) 判断 k 是否大于 $(m+n)/2$.若不大于,则重复步骤(2).否则,线程结束.

线程 2:

(1) 初始化 $i=m-1, j=n-1, k=m+n-1$. (从 A 和 B 的最大值部分开始比较).

(2) 比较 a_i 和 b_j ,将较大的值付给 c_k ,并将其索引(i 或者 j) 减 1,同时将 k 减 1.

(3) 判断 k 是否小于 $(m+n)/2$.若不小于,则重复步骤(2).否则,线程结束.

归并排序双线程行为描述图如图 1 所示.

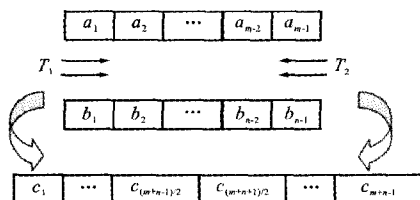


图 1 归并排序双线程行为描述图

两个线程(T_1 和 T_2)同时开始比较合并到数组 c 的中间为止,达到了两个线程负载均衡的目的.这样更加充分地利用了多线程,发挥了多核的潜力.同时,为了能够使用大规模并且重复率小的数据集,避免出现快速排序的最坏情况,采用以下随机数产生方式:

```
for (int i=0; i<NUMBER; i++){
    UINT temp=0;
    for (int num=0; num<uCount; num++){
        temp*=10;
        temp+=rand()%10;
    }
}
```

```
array[i]=temp;
```

```
}
```

NUMBER 是要产生的数据个数, uCOUNT 是根据 NUMBER 而定的要产生的数据的最大位数.例如要产生 1000000 个数据, digit_num 则设定为 6, 即产生的数字最大为六位数.数组 array 就是用来存放随机产生的数据的.

5 算法性能分析

5.1 实验内容及结果分析

为了测试比较组合后的并行排序算法的效率,在以下三个平台上做了实验:

实验平台 1: 单核平台, 惠普台式机, AMD Sempron™ Processor 3800, 2.21GHz, DDR2 内存, 896MB. 操作系统 Microsoft Windows XP.

实验平台 2: 双核平台, Dell vostro 220 双核台式机, Pentium Dual-Core CPU E5200, 2.50GHz. DDR2 内存, 2GB. 操作系统 Microsoft Windows XP.

实验平台 3: 双核平台, 组装机, Intel Core 2 Duo E7200 CPU, 2.52GHz, 3GB 内存. 操作系统 Microsoft Windows XP.

实验内容是在以上三个实验平台上, 随机产生 1 000 000 个整数, 分别测试串行算法和并行算法的效率. 其中串行算法采用平均性能最好的快速排序算法, 未使用多线程技术. 并行算法是优化组合后的排序算法, 采用多线程实现. 并行算法的实现分为 2 线程并行和 4 线程并行两种情况. 实验结果如表 3 所示.

表 3 实验结果 (s)

算法	线程	平台 1	平台 2	平台 3
串行算法	—	0.13496	0.11583	0.11387
并行算法	2 线程	0.13259	0.05674	0.05532
	4 线程	0.13931	0.06024	0.05923

对于串行算法, 在双核平台^[6] (平台 2 和平台 3) 上的性能比在单核平台上的性能提升并不大. 说明传统的串行算法并不能充分发挥多核的潜力. 对于并行算法, 无论是 2 线程还是 4 线程, 在双核平台上的性能比在单核平台上的性能提升了一倍多. 平台 2 和平台 3 的实验结果基本一致, 说明该并行算法是平台无关的, 在不同的双核上均可以提升算法的性能.

在单核平台 1 上, 2 线程的并行算法比串行算

法性能有略微提升,这是由于适当的多线程可以充分利用 CPU. 4 线程的并行算法的性能比串行算法稍有下降,说明过多的线程会导致频繁的线程切换,进而影响性能. 在双核平台上,2 线程的并行算法比串行算法性能提升了约一倍. 4 线程的并行算法比串行算法性能也提升了将近一倍,但不及 2 线程并行算法. 这说明只有将算法并行化才能充分利用多核的潜力. 同时,针对多核的多线程编程必须使用适当的线程个数.

在平台 2 上,对不同的数据规模测试了串行算法和 2 线程的并行算法的效率,结果如图 2 所示.

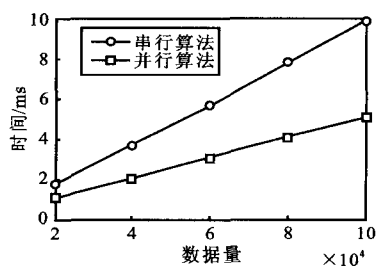


图 2 不同数据规模的算法性能比较

由图 2 可知随着数据规模的增大,并行算法的提升越来越明显. 说明这种并行排序算法适合较大规模的数据排序.

5.2 时间复杂度分析

并行算法主要分为两部分,分块快速排序部分和归并排序部分. 快速排序的平均时间复杂度为 $O(n \log n)$. 在双核系统上,有两个线程同时执行快速排序,快速排序部分的时间复杂度为 $O(n/2 \log(n/2))$. 在归并排序部分,并行算法(2 线程)把两个包含 $n/2$ 个数据的数据集进行合并,生成一个包含 n 个数据的有序数列. 总共需要执行 n 次比较,由于两个处理器并行执行,每个处理器执行了 $n/2$ 次比较. 合并排序部分的时间复杂度为 $O(n/2)$. 所以,整个并行算法的时间复杂度为 $O(n/2 \log(n/2)) + O(n/2)$.

6 结束语

归并排序经过基于多核采用多线程方法优化后的算法有很大的性能提升. 同时证实了以下几点结论:

- (1) 传统的串行算法并不能充分发挥多核的潜力,需要并行化软件来充分利用多核;
- (2) 即使是在单核平台上,适当的多线程可充分利用 CPU,提高软件性能;
- (3) 针对多核的多线程编程必须使用适当的线程个数,也不可线程数目过多于处理器核的数目.

参考文献:

- [1] Kevin D Kissell. 揭密多线程和多核技术[J]. 电子产品世界, 2006(14): 24-26.
- [2] Shameem Akhter, Jason Robertrts. 多核程序设计技术——通过软件多线程提升性能[M]. 北京: 电子工业出版社, 2007.
- [3] 严蔚敏, 吴伟民. 数据结构: c 语言版[M]. 北京: 清华大学出版社, 1997.
- [4] Herruzo Ezequiel, Ruiz Guillermo, Benavides J, et al. A new parallel sorting algorithm based on odd-even mergesort[C]// Distributed and Network-Based Processing, 15th EUROMICRO International Conference on. Parallel, 2007: 18-22.
- [5] Xie Hongwei, Xue Yafeng. An improved parallel sorting algorithm for odd sequence[C]// Advanced Computer Theory and Engineering, International Conference. Thailand: Phuket, 2008: 356-360.
- [6] 多核系列教材编写组. 多核程序设计[M]. 北京: 清华大学出版社, 2007.

作者简介:

钱晓捷 男, (1963-), 副教授, 硕士生导师, CCF 会员. 研究方向为计算机系统结构、微机应用.

李秀芳 女, (1986-), 硕士研究生. 研究方向为多核上的软件并行优化.