DOI: 10.13657/j.cnki.gxkxyxb.2005.s1.018

快速排序异步并行算法的多线程实现

Multi-thread Implement of Asynchronous Parallel Quick Sort Algorithm

石壬息,张锦雄,王 钧,罗晶晶,闫 铁,潘宏胜,冯 涛

Shi Renxi, Zhang Jinxiong, Wang Jun, Luo Jingjing, Yan Tie, Pan Hongsheng, Feng Tao

(广西大学计算机与电子信息学院,广西南宁 530004)

(School of Comp., Elec. and Info., Guangxi Univ., Nanning, Guangxi, 530004, China)

摘要:分析快速排序异步并行算法的排序过程,给出快速排序异步并行算法的多线程实现方式,指出算法多线程实现的特性。

关键词:快速排序 异步并行 多线程 算法

中图法分类号:TP311.1 文献标识码:A 文章编号:1002-7378(2005)S0-0053-02

Abstract The article, based on asynchronous parallel quick sort algorithm, analyzes the sorting process, gives the multi-thread implment, points out the executing feature.

Key words quick sort, asynchronous parallel, multi-thread, algorithm

Hoare提出的串行快速排序算法[1](Quick sort)是一种快速、有效、平均性能优异的排序方法,它利用分治策略,每次将当前长度为 n 的数据序列划分为两个长度几乎相同的序列分别进行排序。这种思想导致了问题求解过程固有并行特征的显式体现,所以说串行快速排序的并行化是快速排序方法本质的回归。

Win32环境中引入了线程的概念,一个进程至少有一个线程,即主线程,也可以有多个线程协同工作[2]。进程从主线程开始执行,进而可以创建一个或多个附加线程来执行该进程内的并发任务。基于线程的多任务允许一个程序的两个或多个部分同时执行,增加了程序的并行度,用户可以通过定义分离的执行线路来完成程序的各个子任务的执行,使编写出来的程序更高效。多处理器系统中,程序的多线程实现不仅可以极大地提高系统效率,而且能充分发挥并行执行的优势[3]。本文在分析快速排序异步并行算法的基础上,给出了算法的多线程实现方法,为并行计算的多线程编程作出有益的探索。

收稿日期:2005-09-10

作者简介:石壬息(1982-),男,广西桂林人,本科生。

1 快速排序异步并行算法分析

异步并行快速排序的基本思想[4]是基于分治策略的。首先生成一个线程,调用 k-选择算法在当前长度为 n 的数据序列中选出中值数据元素,并以此中值数据元素对原数据序列进行重排,重排后使小于中值数据元素的数据排在中值数据元素的前部,使大于中值数据元素的数据排在中值数据元素的后部;然后将中值数据元素存储到最终排序位置上;最后派生两个线程分别对中值数据元素的前部和后部这两个子序列进行上述排序操作。如此不断地进行下去,直到所有动态生成的子序列的长度小于等于 2 为止。当子序列的长度为 2 时,可做简单的比较排序,即可将原始数据序列排序完毕。

由此可见,快速排序过程可用一棵逻辑二叉线程树表示,树中每个结点代表一个线程,每个非叶子结点线程要做 3 件事情:找中值数据元素、重排原数据序列和派生两个线程;每个叶子结点线程只做简单的不多于 2 个数据的排序。

2 算法的多线程实现

应用程序启动一个进程,系统为该进程创建一个主线程。在主线程中首先完成待排序数据的输入,然后创建根线程,最后在排序完毕后负责输出已排

好序的数据。可见,主线程并不做排序工作,本文把主线程以外的线程称为排序线程。

每个排序线程首先判断其待处理的数据是否超过 2个,若不超过 2个则做不多于 2个数据的排序;若超过 2个,则首先找中值数据元素并重排所处理的数据,然后将中值数据元素存储到最终排序位置上,最后创建两个线程。

为了让主线程知道全部排序完毕,设置全局变量 count,用于存放待排序数据的总数。由于每个排序线程都确定 1个或 2个数据在排序后的位置,所以让排序 1个数据的排序线程对 count做减 1计数,让排序 2个数据的排序线程对 count做减 2计数。主线程在创建根线程后,就不断地检测 count值,当发现 count值为 0时,就可断定所有数据排序完毕。主线程的程序流程图如图 1所示,排序线程的程序流程图如图 2所示。

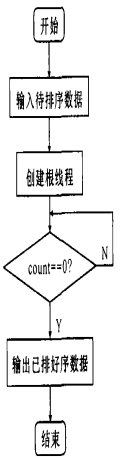

图 1 主线程流程

多线程编程实现时要注意以下几个问题:

(1)互斥地访问全局变量 count。排序线程对 count减法计数时应互斥地进行,调用函数 InterlockedDecrement((long*)& count)可实现对 count互斥地做减 1计数。

(2)处理好线程参数的传递。线程函数原型:DWORD WINAPI Thread (LPVOID param);Thread为线程函数名。

函数 CreateThread()用来创建线程,其原型如下:

```
HANDLE CreateThread
    LPSECURITY_ ATTRIBUTES
lpThreadAttributes, //安全属性指针
    DWORD  dwStackSize, //堆栈大小
    LPTHREAD_ START_ ROUTINE
lpStartAddress, //线程起始地址
    LPVOID  lpParameter, //传递给线程的参数
    DWORD  dwCreationFlags, //起始执行状态
    LPDWORD lpThreadId); //线程 ID指针
```

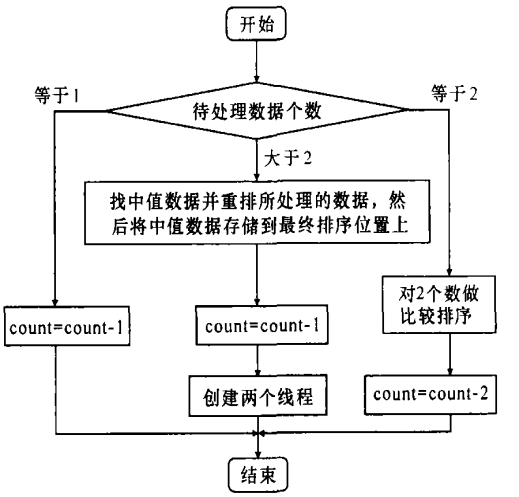

图 2 排序线程流程

安全属性指针选 NULL即可;dwStackSize也选 NULL,表示堆栈大小与主线程相同;dwCreationFlags有两种情况,0表示立即执行;CREATE_ SUSPENDED表示创建后被挂起,直到调用 ResumeThread();线程起始地址则以线程函数名即可;该函数如果操作成功则返回线程的句柄,失败则返回 NULL。注意到线程参数只有一个位置,当要传递许多不同类型的参数给线程时,应定义一个结构类型,此结构以需要传递的不同类型的参数为成员,在传递参数前需声明此结构的变量,并设置各成员的值,以变量名作为参数在创建线程时向线程传递参数。

(3)中值数据元素的查找及排序后的位置。若数组 a中下标从 p到 r这段数据为待处理的数据,记为 $a[p: r]$,那么其中的中值数据元素应是以 p为基点的第 $k(k = 1+\left[\frac{r-p}{2}\right])$小元素,中值数据元素找到后应存放最终位置的下标为$\left[\frac{p+r}{2}\right]$。

3 算法特性

本算法实现在运行过程中,呈现出以下几个方面的特性。

(1)逻辑二叉线程树是一棵平衡二叉树,非叶子结点线程只排序一个数据,叶子结点线程排序不多

(下转第 64页)

增加 DS体系结构的复杂性,而是在每个组播路由器的表中添加不同的 DSCP值,没有减弱 DiffServ的可扩展性。

5 结束语

区分服务和组播共存时产生的问题主要是由于 DiffServ体系结构的简化造成的,可以通过为单一传播和组播使用不同的 DS编码值,即在组播路由表的每个输出链路条目中加入一项 DSCP,并辅以一定管理机制加以解决,这种方案实现简单,同时又保持 DiffServ良好的可扩展性。总之,解决 DiffServ网络区支持组播存在的固有缺陷,并最终实现在 DiffServ网络区支持具有端到端 QoS保证的组播传输,是目前研究的难点问题。

参考文献:

[1] Shenker S, et al. Specification of Guaranteed Quality of Service, RFC 2212[M]. 1997.

[2] Blake S, et al. An Architecture for Differentiated Services, RFC 2475[M]. 1998.

[3] Bless R, Wehrle K. IP multicast in differentiated services (DS) networks[J]. Request for Comments, 2004 3754.

[4] Striegel A, Manimaran G. A scalable approach to diffserv multicasting [J]. Proc of International Conference on Communications, Helsinki, 2001, 6: 2327-2331.

[5] Striegel A, Manimaran G. A scalable protocol for member join/leave in diffserv multicast [J]. Proc of Local Computer Networks (LCN) Tampa Florida, 2001, 11: 395-404.

[6] 林 闯,单志广,任丰原.计算机网络的服务质量 [M]. 北京: 清华大学出版社, 2004.

(责任编辑: 黎贞崇)

(上接第 54页)

于 2个的数据。若待排序数据的总数为 n,平衡二叉树树高为 i,则有:

$$3\times 2^{i-1} \leqslant n \leqslant 3\times 2^{i} - 1$$

(2)线程树中在同一路径上的两个线程存在逻辑依赖关系,它们在计算上不可能重叠;即便是父子关系的两个线程,它们之间也没有实质性的计算重叠,不在同一路径上的两个线程则可异步并行执行。

(3)当待排序的数据很多时,算法将创建许多排序线程,线程创建将占用相当的时间开销。在逻辑并行度与物理并行度失配时,线程间的切换也将占用相当的时间开销。

4 结束语

多线程进行并行编程主要利用进程中线程的并发性来描述问题求解过程的并行性,用于完成子任务工作的各线程可并发执行。目前大多编程工具现均已引入线程机制,为并行计算的多线程实现提供了途径。本文介绍快速排序异步并行算法的多线程实现,探索并行计算的多线程编程方向,对今后的进一步研究有一定的参考价值。

参考文献:

[1] Clifford A. Shaffer. 数据结构与算法分析 (Java版) [M]. 张 铭, 刘晓丹 译. 北京: 电子工业出版社, 2002. 163-168.

[2] 任 哲,李益民,车进辉. MFC Windows应用程序设计 [M]. 北京: 清华大学出版社, 2004. 205-224.

[3] Cameron Hughes, Tracey Hughes. C++ 面向对象多线程编程 [M]. 周良忠 译. 北京: 人民邮电出版社, 2003. 80-83.

[4] 苏德富,钟 诚. 计算机算法设计与分析 [M]. 北京: 电子工业出版社, 2001. 128-130.

(责任编辑: 黎贞崇)