学习情况

2014-4-11

周永强

六院八隊

Contents

说明

1 学习情况

这两周主要还是在上课，清明跑出去玩了两天，这篇报告是根据自己的课程《高级并行程序设计》来写的。上次师门报告的时候听师兄讲了一些工具的使用，有些是我知道的，有些是我不知道的但是却感兴趣的，于是这两周有相当一部分时间花在了工具的学习和使用上。

期间学习了 LaTeX 排版，以前只是知道有这个工具，但是没有用过，这次集中学习了一下，这篇报告就是用其排版而成，当然我不是从头做的，而是基于网上提供的一个模板。LaTeX 功能很多，我着重学习了比较常用的，也是自己当前需要使用的功能，包括：数学公式、伪代码书写、图片插入等。其次学习 gnuplot 绘图工具用于绘制常用数据分析图。本篇报告中的折线图就是用 gnuplot 绘制的。

结合课程需要，这两周在学习 MPI 编程，了解并行编程模型。目前对于并行编程我还处于入门阶段，后续我准备继续学习 MPI，以及 OpenMP，为以后的学习和工作打下基础，毕竟我当前并行方面的知识比较缺乏。下文中给出的是一个快速排序算法的实例，为了说明这两周的学习情况，由于时间不够用（主要是我懒），所以 OpenMP 版本程序没有写。

实验室那边这两周去的比较少，期间假期外出的缘故为一，其次则是课程这边也需要花不少时间。目前那边的情况是基本功能实现了，但是性能还跟不上去，达不到要求，主要是为并行程序设计把握的不好，这段时间我再想想办法改进。

快速排序

本节中先给出快速排序的串行化算法，然后对算法进行分析，引出并行的快速排序算法，并且分别使用 MPI 和 OpenMP 实现并行的快速排序，最后对算法的性能进行评估，给出评价和结果分析。

1 快速排序算法

快速排序是一种基本的排序算法，其时间复杂度为$O(n\log n)$，它的基本思想是：对于给定的无序序列$R[1,n]$，从中选取一个元素作为“标兵”（一般选取第一个元素或最后一个元素），以“标兵”为基准将序列划分为两个子序列$R[1,i-1]$和$R[i,n]$，且左边的无序子区中记录的所有关键字均小于等于基准的关键字，右边的无序子区中记录的所有关键字均大于等于基准的关键字，当子序列不空时，递归调用子上述过程，程序运

行结束时，序列有序。

快速排序依赖划分操作，划分算法给出如下。

Algorithm 1 快速排序算法

```
1: function quicksort(data, i, j)                    ▷ quicksort from i to j
2:     if i < j then
3:         r ← partition(data, i, j)
4:     end if
5:     quicksort(data, i, r − 1)
6:     quicksort(data, r + 1, j)
7: end function
```

Algorithm 2 partition算法

```
1: function partition(data, k, l)
2:     pivo ← data[l]
3:     i ← k − l
4:     for j ← k, l − 1 do
5:         if data[i] ≤ pivo then
6:             i ← i + 1
7:             exchange data[i] and data[j]
8:         end if
9:     end for
10:    exchange data[i + 1] and data[l]
11:    return i + 1
12: end function
```

快速排序算法并行化的一个简单思想是，对每次划分过后所得到的两个序列分别使用两个处理器完成递归排序。例如对一个长为n的序列，首先划分得到两个长为$n/2$的序列，将其交给两个处理器分别处理；而后进一步划分得到四个长为$n/4$的序列，再分别交给四个处理器处理；如此递归下去最终得到排序好的序列。

1.1 快速排序MPI实现

MPI 是 Message Passing Interface 的缩写，中译为消息传递接口，用于分布式内存系统的编程，在目前并行计算领域广为应用。MPI 并不是一门编程语言，而是与编程语言库的形式提供，开发人员可以根据自己问题的需要，应用 MPI 进行并行编程。
MPI 数据传送分为阻塞与非阻塞两种，通信方式有广播方式和点对点方式，快速排序中，由0号进程负责对数据进行初始化，然后将长度广播到其他的所用进程。限于篇幅，这里不再贴出源代码。

1.2 快速排序OpenMP实现

暂未实现

Algorithm 3 快速排序并行算法

```
function para_quicksort(data, i, j, m, id)
    if (j − i) ≤ k||m = 0 then
        P_id : call quicksort(data, i, j)
    else
        P_id r = partition(data, i, j)
        P_id : send data[r + 1, m − 1] to P_{id+2^{m−1}}
        para_quicksort(data, i, r − 1, m − 1, id)
        para_quicksort(data, r + 1, j, m − 1, id + 2^{m−1})
        P_id : send data[r + 1, m − 1] back to P_id
    end if
end function
```

1.3 实验结果与分析

程序的测试在我的个人笔记本上完成， CPU 芯片为 Intel T6600 双核处理器，图中给出的是单个进程在执行并行快速排序时所耗用的时间。测试中选用定长的1000000条数据进行排序，分别统计了$1, 2, 4, 8$个进程时在单个进程上排序所耗用的时间，这里没有算上通信所带来的开销。图中可以明确看出随着进程数的增加，单个进程上的并行快速排序时间在减少，而且当进程数增加到原来的两倍时，排序时间大约减为原来的一半。由此，除去通信的开销，并行程序的效率明显高于串行程序，当然这是以更高的硬件资源为代价的。

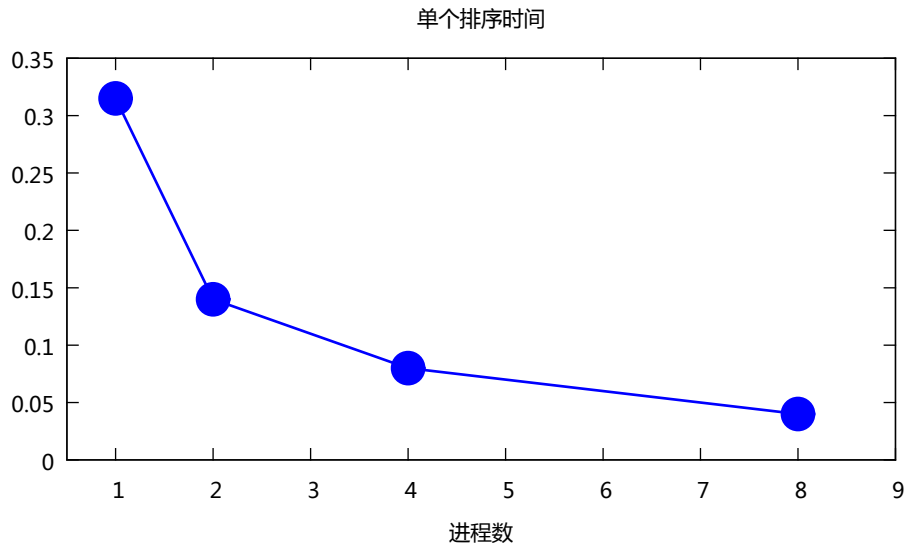

Figure 1: MPI结果分析图

Bibliography

[1] Leslie Lamport, LaTeX: A Document Preparation System. Addison Wesley, Massachusetts, 2nd Edition, 1994.

[2] 陈国良, 并行计算——结构算法编程. 高等教育出版社, 1999.