

## 多核机群上通信高效的整数序列并行排序方法

柯琦<sup>1,2</sup>, 钟诚<sup>1\*</sup>, 陈清媛<sup>1</sup>, 陆向艳<sup>1</sup>

(1. 广西大学 计算机与电子信息学院, 南宁 530004; 2. 广西财经学院 信息与统计学院, 南宁 530003)

(\* 通信作者电子邮箱 chzhong@gxu.edu.cn)

**摘要:** 建立一个适用于整数序列排序的数据分配模型, 在多核计算节点组成的异构机群上设计通信高效的整数序列并行算法。所提出的数据分配模型依据机群中各节点不同的计算能力、通信速率和存储容量, 动态计算出调度分配给各节点的数据块的大小以平衡各个节点的负载。所设计的并行排序算法利用整数序列的特性, 主节点采取两轮分发数据与接收结果的方法, 从节点运用分桶打包方式返回有序的整数子序列给主节点, 主节点采用桶映射方法将各个有序子序列直接整合成最终有序序列, 以减少需要耗费较多通信时间的数据归并操作。分析与实验测试结果表明, 给出的多核机群上的整数序列并行排序算法高效, 具有良好的可扩展性。

**关键词:** 整数排序; 并行算法; 多核机群; 数据分配

**中图分类号:** TP338.6; TP301.6 **文献标志码:** A

### Communication-efficient parallel sorting integers sequence on multi-core cluster

KE Qi<sup>1,2</sup>, ZHONG Cheng<sup>1\*</sup>, CHEN Qingyuan<sup>1</sup>, LU Xiangyan<sup>1</sup>

(1. Computer and Electronic Information College, Guangxi University, Nanning Guangxi 530004, China;

2. School of Information and Statistics, Guangxi University for Finance and Economics, Nanning Guangxi 530003, China)

**Abstract:** A data distribution strategy and a communication-efficient parallel algorithm for sorting integers sequence were proposed on the heterogeneous cluster with multi-core machines. The presented data distribution model properly utilized different computation speed, communication rate and memory capacity of each computing node to dynamically compute the size of the data block to be assigned to each node to balance the loads among nodes. In the proposed parallel sorting algorithm, making use of the characteristic of integers sequence, master node distributed the data blocks to the slave nodes and received the sorted subsequences with two-round mode, each slave node returned its sorted subsequence to master node by bucket-packing method, and master node linked its received sorted subsequences to form directly a final sorted sequence by the bucket mapping in order to reduce the data merge operations with large communication cost. The analysis and experimental results on the heterogeneous cluster with multi-core machines show that the presented parallel sorting integers sequence algorithm is efficient and scalable.

**Key words:** integers sorting; parallel algorithm; multi-core cluster; data distribution

## 0 引言

在多核系统上研究设计存储、通信高效的并行排序算法向人们提出了新的课题。文献[1]采取单指令流多数据流(Single Instruction stream Multiple Data stream, SIMD)指令、线程级并行和消除非对准存储访问的方法,设计了一种多核处理器上高效的并行排序算法。文献[2]在 Cell 机器上实现了基数排序并行算法,该算法能够有效地利用了每个处理核心的能力。文献[3]采取负载均衡方法在 GPU 处理器上实现了快速排序算法。文献[4]研究了流体系结构 GPU 机器上并行排序的优化。基于划分-归并方法,文献[5]提出了一个在多核处理器上运行的缓存感知的并行排序算法。基于 Cell/B. E. 机器,文献[6]采用流水线方式组织归并过程以减少存储访问时间、优化了片上流水线归并排序的执行性能。利用 SIMD 指令,文献[7]分别实现了多核 CPU 系统和 GPU 系统上的带宽无关的并行排序算法。针对大规模 Multisets 数据序列中不同关键字较少的特殊情形,文献[8]采取动态平衡核

心负载的方法,设计了多核机器上存储高效的 Multisets 排序并行算法;进一步地提出了节点间的非周期多轮调度策略,确定出调度轮数和每轮调度任务大小,实现了 Multisets 数据序列在多核异构机群上的快速排序。文献[9]采取分块方法将整数序列划分分配到多核机器的二级缓存和一级缓存,并利用 SIMD 指令和线程绑定技术来加速整数序列的并行排序过程。文献[10]研究了多核机器上的线程级并行抽样排序方法,测试分析了不同抽样方法、抽样规模对多核多线程并行排序性能的影响。考虑返回信息的通信代价和节点存储容量有限的情形,采取两轮探测策略,文献[11]给出了一个多核处理器系统参数未知异构机群系统的可分负载多轮调度方法。整型数据排序问题在现实应用中经常出现。针对节点具有不同的计算能力、通信速率和存储容量的多核机群,提出适用于整数序列排序的通信高效的并行算法的研究工作成果极少。

本文研究异构计算节点组成的多核机群系统上的整数序列并行排序问题,主要贡献是:提出了将数据分配调度给多核机群中各节点的方法,使得各节点接收到的数据量能够适应

收稿日期:2012-09-24;修回日期:2012-11-08。 基金项目:国家自然科学基金资助项目(60963001)。

**作者简介:** 柯琦(1985-),女,广西恭城人,助教,硕士,主要研究方向:并行分布计算; 钟诚(1964-),男,广西桂平人,教授,博士生导师,博士,主要研究方向:并行分布计算、可信计算与软件; 陈清媛(1987-),女,贵州金沙人,硕士研究生,主要研究方向:可信计算与软件; 陆向艳(1973-),女,广西南宁人,副教授,主要研究方向:并行分布计算。

其计算能力、通信速率和存储容量,平衡各个节点的计算负载,以达到所有节点能够同时完成子序列排序之目的;利用整数序列和多级缓存特性,巧妙采取映射和桶排序方法,使得节点内线程级并行排序存储高效,节点间尽量减少需要通信量较大的数据归并操作,实现并行排序过程的通信高效。

## 1 整数序列分配模型及其并行排序

### 1.1 整数序列分配模型

待排序整数序列记为  $S = \{S_0, S_1, \dots, S_{N-1}\}$ ,  $0 \leq S_j \leq m$ ,  $0 \leq j \leq N-1$ ;多核机群共有  $d$  个节点,  $D_0$  为主节点,  $D_1 \sim D_{d-1}$  为从节点;分配调度给节点  $D_i$  ( $0 \leq i \leq d-1$ ) 排序处理的整数子序列有  $task_i$  个整数,  $D_0$  与  $D_i$  之间的通信链路记为  $L_i$ , 通信延迟为  $Tlat_i$ , 通信速率为  $g_i$ , 在链路  $L_i$  上  $D_0$  向  $D_i$  传输  $task_i$  个整数所用时间是  $Tlat_i + g_i \times task_i$ ; 从节点  $D_i$  返回结果给  $D_0$  时的通信延迟记为  $Tlat'_i$ , 通信速率为  $g'_i$ ,  $D_i$  返回规模为  $task_i$  的子序列有序结果给  $D_0$  时所需的通信开销为  $Tlat'_i + g'_i \times task_i$ ;  $D_i$  的计算延迟记为  $Jlat_i$ , 调用文献[9]给出的线程级并行算法排序处理一个整数所需时间为  $w_i$ ,  $D_i$  排序处理完毕规模为  $task_i$  的子序列所需时间为  $Jlat_i + w_i \times task_i$  ( $0 \leq i \leq d-1$ ); 主节点  $D_0$  调用多核线程级并行计算前缀和算法<sup>[12]</sup> 计算一个前缀和所需时间为  $wp_0$ , 计算  $bucknum$  个整数前缀和所需时间为  $wp_0 \times bucknum$ 。

设多核机群采用半双工通信方式传输数据, 支持计算与通信重叠。  $D_0$  将规模为  $task_i$  的整数分发给  $D_i$ 。多核机群上调度分发数据、返回有序子序列及整理排序结果的过程如图1所示。

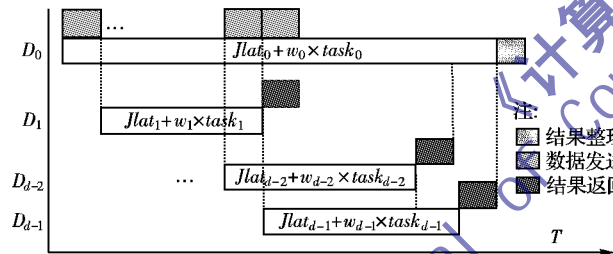


图1 多核异构机群上分发数据、返回有序子序列及整理排序结果过程

调度分配给多核机群中各个节点的整数子序列的规模必须适应其计算、通信和存储能力,以使得所有节点尽可能同时完成其子序列的排序。为此,必须研究建立适用于整数序列并行排序的多核机群上的数据分配模型。

设  $T_i$  ( $0 \leq i \leq d-1$ ) 为节点  $D_i$  完成规模为  $task_i$  的整数子序列排序所需时间,  $T_{\max}$  为多核机群完成整数序列排序的时间, 则  $T_{\max} = \max\{T_i\}$ , 于是有:

$$\sum_{i=0}^{d-1} task_i = N; task_i \geq 0 \quad (1)$$

节点  $D_i$  排序完成分配给它的整数子序列的执行时间  $T_i$  应该在时间  $T_{\max}$  之前, 因此有:

$$T_0 = Jlat_0 + task_0 \times w_0 + wp_0 \times bucknum \leq T_{\max} \quad (2)$$

$$T_i = (Jlat_i + task_i \times w_i) + (Tlat'_i + g'_i \times task_i) \leq T_{\max} \quad (3)$$

当  $D_i$  ( $1 \leq i \leq d-2$ ) 排序完成其子序列所需时间等于  $D_0$  向所有  $D_j$  发送规模为  $task_j$  的子序列所需通信时间 ( $i+1 \leq j \leq d-1$ ) 与所有  $D_k$  返回规模为  $task_k$  的有序子序列给  $D_0$  所需的通信时间 ( $0 \leq k \leq i-1$ ) 之和时, 通信链路利用率达到最大:

$$\sum_{j=i+1}^{d-1} (Tlat_j + g_j \times task_j) + \sum_{k=0}^{i-1} (Tlat'_k + g'_k \times task_k) = Jlat_i + task_i \times w_i \quad (4)$$

分配给  $D_i$  的子序列之规模  $task_i$  必须小于  $D_i$  当前可用主存容量。设  $D_i$  主存当前可容纳  $buf_i$  个整数, 则有:

$$task_i \leq buf_i; 0 \leq i < d \quad (5)$$

上述式(1)~(5)构成了多核机群上整数序列并行排序的数据分配模型的线性约束条件, 通过求解出  $task_i$  以指导在多核机群上调度整数子序列分配给  $D_i$  的数据量, 然后进行并行排序。计算  $task_i$  的步骤如下: ① 测试出节点  $D_i$  的通信速率  $g_i$ 、通信延迟  $Tlat_i$ 、计算延迟  $Jlat_i$ 、节点执行多线程级并行算法排序处理一个整数所需时间  $w_i$ , 以及节点调用文献[9]的算法测试出排序等数据量的各节点执行的通信和计算时间  $Tx_i$ 。② 测试出的每个节点的执行时间  $Tx_i$  中必有一个最大值  $Tx_{\max}$ , 根据式(3), 令  $T_i = T_{\max}$ , 则可求出每个节点相对于速率最慢节点的节点权值  $q_i = \max_{j=1}^d Tx_j / Tx_i$ 。③ 通过求解由式(1)~(2)和式(4)~(5)组成的方程, 并根据权值  $q_i$ , 可求出分配给节点  $D_i$  的数据量  $task_i$  ( $0 \leq i \leq d-1$ )。

### 1.2 通信高效的整数序列并行排序算法

依据异构计算节点组成的多核机群和整数序列数据的特性, 为了减少并行排序过程中产生的通信代价, 让主节点执行两轮数据分发与有序子序列接收操作:

第一轮, 依据所建立的多核机群上整数序列的数据分配模型, 计算出调度分配给节点  $D_i$  ( $0 \leq i \leq d-1$ ) 的数据规模  $task_i$ , 然后主节点  $D_0$  分发数据块给从节点  $D_i$ ,  $D_i$  调用多核机器上存储高效的线程级并行排序算法<sup>[9]</sup> 排序整数子序列并将结果返回给  $D_0$ ;  $D_0$  按返回结果所属的桶号对应链接子序列数据, 从而得到桶与桶之间数据有序、桶内数据无序的整数序列;  $D_0$  调用多核机器上线程级并行计算前缀和算法<sup>[12]</sup> 以确定出第二轮将要分发给  $D_i$  的数据划分点(位置)。

第二轮,  $D_0$  根据划分点将数据块分发给  $D_i$ ,  $D_i$  调用线程级并行排序算法对接收到的数据块排序, 采用分桶打包方式返回有序子序列给  $D_0$ ; 最后  $D_0$  接收有序子序列并装入最终的有序数组中。

**算法1** 多核机群上整数序列并行排序算法。

// 节点  $D_i$  使用的  $bucknum$  个数据桶为

//  $A_{i,0}, A_{i,1}, \dots, A_{i,bucknum-1}$ 。

Begin

- 1) 依据建立的多核机群上整数序列数据分配模型, 计算出调度分配给节点  $D_i$  的数据规模  $task_i$ , 然后  $D_0$  将规模为  $task_i$  的子序列分发给节点  $D_i$  ( $0 \leq i \leq d-1$ )。
- 2)  $D_i$  调用多核机器上整数序列线程级并行排序算法<sup>[9]</sup> 排序其接收的数据, 按照先进先出策略返回有序子序列给  $D_0$ 。
- 3)  $D_0$  将  $D_i$  返回的各桶数据  $A_{i,0} \sim A_{i,bucknum-1}$  ( $0 \leq i \leq d-1$ ) 按其所属的桶号分别链接到对应桶  $A_{0,0} \sim A_{0,bucknum-1}$  中, 得到桶间有序、桶内无序的整数序列, 即桶  $A_{0,j}$  中的元素小于等于桶  $A_{0,j+1}$  中的元素,  $0 \leq j \leq bucknum-2$ 。
- 4)  $D_0$  调用多核并行计算前缀和算法<sup>[12]</sup>, 计算下一轮分发给  $D_i$  的数据划分点, 并计算桶  $A_{i,j}$  中的数据存放在最终的有序数组中的起止位置,  $0 \leq j \leq bucknum-1$ ,  $0 \leq i \leq d-1$ 。
- 5)  $D_0$  按计算出的划分点将大小为  $task_i$  的数据块发送给  $D_i$  ( $0 \leq i \leq d-1$ )。
- 6)  $D_i$  调用多核机器上整数序列线程级并行排序算法<sup>[9]</sup> 对其接收的数据再次排序, 以分桶打包方式返回有序子序

列给  $D_0$  ( $0 \leq i \leq d-1$ )。

- 7)  $D_0$  合并有序子序列到对应的数据桶,根据4)中计算出的最终有序数组中起止位置进行装入,得到完整的有序整数序列结果。

End

**结论1** 若  $D_i$  ( $0 \leq i \leq d-1$ ) 处理规模为  $task_i$  的有序子序列全部属于  $D_0$  中的某一个桶,则直接返回给  $D_0$ ;若  $D_i$  处理规模为  $task_i$  的有序子序列属于  $D_0$  中的两个以上的桶时,则需要对桶打包操作,根据桶号对有序子序列数据打包之后再发送给  $D_0$ ,每个  $D_i$  最多只有3个不同的数据包。

**证明** 假设在最坏情形下  $D_i$  处理规模为  $task_i$  的有序子序列分别属于  $D_0$  中的  $y$  个连续桶  $A_{0,x}, A_{0,x+1}, \dots, A_{0,x+y-1}$  中的数据;并且假设  $D_{i-1}$  中已分得一部分桶  $A_{0,x}$  的数据,  $D_{i-1}$  已分得桶  $A_{0,x+y-1}$  剩下的数据。 $D_i$  对分得的桶  $A_{0,x}, A_{0,x+1}, \dots, A_{0,x+y-1}$  中的整数排序后,只需对桶  $A_{0,x}$  和  $A_{0,x+y-1}$  单独打包返回结果给  $D_0$ ,而将桶  $A_{0,x+1}, \dots, A_{0,x+y-2}$  中的数据一起封装打包返回给  $D_0$ 。这是因为算法第3)步之后,  $D_0$  得到了桶间有序桶内无序的整数序列。 $D_0$  接收  $D_i$  返回的桶  $A_{0,x}$  中的数据包之后,需要与  $D_{i-1}$  返回的桶  $A_{0,x}$  中的数据合并;而  $D_0$  接收  $D_i$  返回的桶  $A_{0,x+y-1}$  中的数据包后,需等待与  $D_{i+1}$  返回桶  $A_{0,x+y-1}$  中的剩下的数据进行合并。而桶  $A_{0,x+1}, \dots, A_{0,x+y-2}, A_{0,x+y-1}$  中的数据已经有序,无需与其他任何桶中数据进行合并,可直接存入最终有序数组对应位置。因此,每个从节点在最坏情形下最多返回3个不同的数据包。证毕。

**结论2** 按照本文给出的数据分配模型调度分发整数数据给节点进行并行排序的算法所需的通信次数比传统的机群系统上并行归并排序算法所需的通信次数少,减少了大量通信开销。

**证明** 按照本文给出的并行排序方法,在排序过程中,主节点  $D_0$  与每个从节点  $D_i$  通信4次,总共需要  $4d$  次通信。而对于传统机群系统上的并行归并排序算法,它在第一次分发数据之后,即使是均匀分布的数据序列,任意两个节点  $D_i$  和  $D_j$  之间至少需要进行一次相互(双向)交换数据块,机群系统中共有  $d$  个节点,需要进行  $p_d^2$  次通信,这样使用传统归并算法进行并行排序至少需要  $d + p_d^2 = O(d^2)$  次通信。因此,本文给出的并行排序方法显著地减少了通信开销。

另一方面,本文的并行排序方法利用所建立的多核机群上的数据分配模型,计算出规模为  $task_i$  的子序列调度分配给节点  $D_i$  进行排序,可以使得各个节点的负载均衡;利用整数序列的特性,主节点进行两轮分发数据与接收有序子序列,从节点采用分桶打包方式返回有序子序列给主节点(根据结论1可知,每个从节点在最坏情形下最多返回3个不同的数据包),便于主节点把数据直接装入至最终的有序数组中,大大减少了数据归并操作(通信开销)。证毕。

因此,本文的并行排序方法实现了通信高效,优于传统的机群系统上的并行归并排序算法。

## 2 实验

实验采用100 Mb/s以太网将多核PC按星型拓扑结构构成一个异构机群系统,每个多核节点都具有自己独立的内存、一级和二级缓存、外存、IO外设,机群中多核节点处理器的基本信息如表1所示。运行的操作系统为RedHat Enterprise Linux 5。采用C语言、MPI和OpenMP混合编程。实验选用主存容量2 GB、二级缓存(L2 Cache)容量12 MB、一级缓存

(L1 Cache)容量4 \* 32 KB的一个Intel四核处理器作为主节点负责分发数据和接收结果。在实验之前,首先测试出各节点的  $g$ 、 $f_{lat}$ 、 $j_{lat}$  和  $w$ ,然后根据整数序列分配模型计算出分配给各节点的数据量  $task_i$ 。实验测试的数据规模分别为2000万、2500万、3000万、3500万、4000万和5000万个整数。

表1 异构机群中多核节点处理器的系统参数

处理器类型	主存 容量/GB	二级缓存 容量/MB	一级缓存 容量/KB
Xeon E5405 (4核, 2.0 GHz)	2	12	4 * 32
Athlon 64 X2 (2核, 2.3 GHz)	1	1	2 * 32
Pentium E5400 (2核, 2.7 GHz)	1	2	2 * 32

图2测试了对于不同数据规模的整数序列,当多核异构机群系统中运行的节点数目逐步增多时,多核异构机群上整数序列并行排序算法所需的运行时间。图3则给出了多核机群中运行节点数目与处理核心数目的不同组合对整数序列并行排序算法所需运行时间的影响。

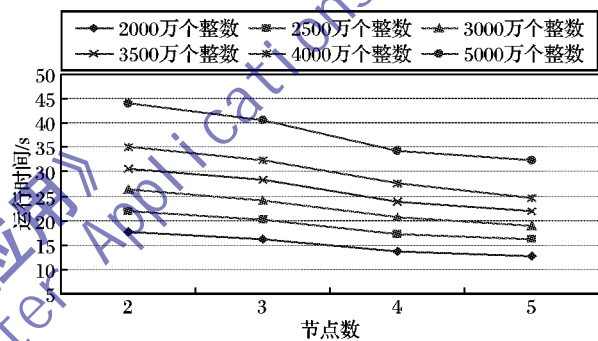


图2 运行节点数增加时本文算法的运行时间

从图2可看出,对于给出的6种规模的整数数据序列,随着多核机群系统中运行节点的增多,整数序列并行排序算法所需的运行时间是逐渐减少的。另一方面,从图2中也可以看出,当多核机群中运行的节点增多时,数据传输和排序过程中通信开销也相应有所增加,因此本文排序算法所需的运行时间下降趋势有所减缓。这说明了,在由异构节点组成的多核机群中设计实现并行排序算法时,尽量减少算法的通信代价是非常重要的。

图3的实验结果表明,对于规模相同的整数数据序列并行排序,当多核机群中运行的节点个数相同时,如果系统中参与并行排序的处理核心越多,那么排序算法所需的运行时间越短。这说明了,当多核机群中运行的节点规模固定时,通过增强每个节点的计算能力(增加核心数目),可以增强机群系统的处理能力,加速并行算法的执行。

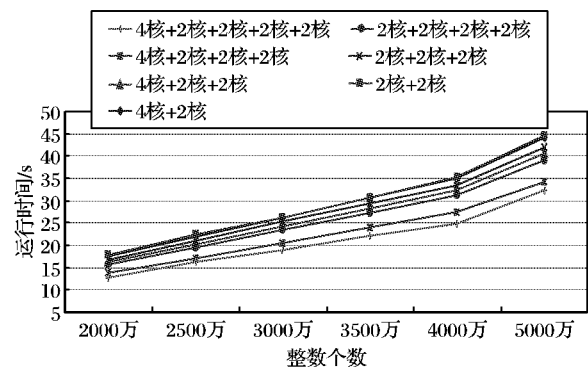


图3 运行不同组合数目的节点与核心时本文算法的运行时间

图4给出了当多核机群中运行节点逐步增多、相应地增



加数据规模时,本文的整数序列并行排序算法的等效率函数曲线,以考察算法的可扩展性。图5则给出了不考虑多核节点之间异构性、采用平均分配(均匀划分)<sup>[13]</sup>数据方式对整数序列进行并行排序所需的运行时间,以及基于本文给出的整数序列分配模型调度数据给节点进行并行排序所需的运行时间。

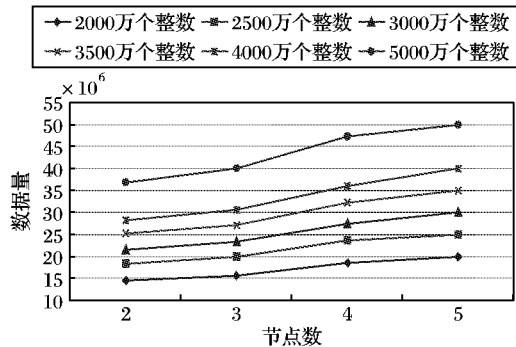


图4 异构机群上整数序列并行排序算法的等效率函数曲线

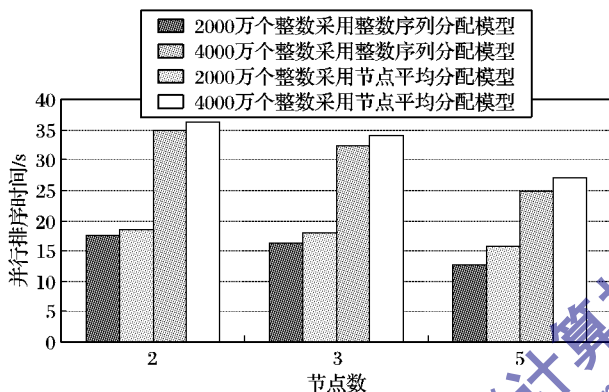


图5 基于平均分配数据模型与本文分配模型的并行排序时间

从图4可看出,为了维持整数序列并行排序算法的等效率,当多核机群中运行节点逐步增多时,需要增加的整数序列的数据规模与节点规模呈亚线性增长趋势,这说明本文给出的并行排序算法具有良好的可扩展性,它能够较好地利用了多核机群的计算能力。

从图5可看出,采用本文提出的针对异构机群系统设计的数据分配模型,调度分发数据给从节点进行并行排序整数序列所需的执行时间要比不考虑各节点之间的异构性而平均分配数据给从节点进行并行排序所需的执行时间少。这是因为,按照本文提出的数据分配模型将整数序列调度分配给异构机群系统中的各个多核节点,可以确保各个节点的负载平衡,从而使得各个节点完成局部排序所需的时间相同以及使得完成整数序列的完全排序所需的时间尽可能少。另一方面,从图5中也看到,采用两种分配数据方式进行并行排序整数序列所需的执行时间差距较小,这主要是因为本文进行实验的机群系统各节点是采用低速以太网连接的,系统的数据传输速率受到一定的限制。

### 3 结语

针对整数序列并行排序问题,提出的数据分配模型使得数据的分配调度能够适应机群中各节点不同的计算能力、通信速率和存储容量,平衡了各个节点的负载;给出的并行排序方法利用了整数序列和多核结构多级缓存特性,每个节点执行存储高效的线程级并行排序算法排序其接收到的子序列;经过主节点采取两轮分发数据与接收有序子序列、从节点采

用分桶打包方式返回有序子序列给主节点之后,主节点可直接将各个有序子序列整合成最终有序的整数序列,从而减少了传统机群并行排序算法中需要耗费较多通信时间的数据归并操作,实现了并行排序过程的通信高效。随着 GPGPU 机器的出现,研究多核 CPU 和 GPU 混合异构系统高效并行算法是一个新的课题。我们下一步将研究多核 CPU 和 GPU 混合异构机群上高效的整数数据序列并行排序算法。

#### 参考文献:

- [1] INOUE H, MORIYAMA T, KOMATSU H, *et al.* AA-Sort: a new parallel sorting algorithm for multi-core SIMD processors [C]// Proceedings of the 16th International Conference on Parallel Architecture and Compilation Techniques. Washington, DC: IEEE Computer Society, 2007: 189-198.
- [2] RAMPRASAD N, BARUAH P K. Radix sort on the cell broadband engine [C]// HiPC2007: Proceedings of the 14th Annual IEEE International Conference on High Performance Computing. Piscataway, NJ: IEEE Press, 2007.
- [3] CEDERMAN D, TSIGAS P. On sorting and load balancing on GPU [J]. ACM SIGARCH Computer Architecture News, 2008, 36(5): 11-18.
- [4] GREB A, ZACHMANN G. GPU-ABISort: optimal parallel sorting on stream architectures [C]// Proceedings of the 20th International Parallel and Distributed Processing Symposium. Washington, DC: IEEE Computer Society, 2006: 25-29.
- [5] HAO S, DU Z, BADER D, *et al.* A partition-merge based cache-conscious parallel sorting algorithm for CMP with shared cache [C]// Proceedings of the 38th International Conference on Parallel Processing. Washington, DC: IEEE Computer Society, 2009: 396-403.
- [6] HULTÉN R, KESSLER C W, KELLER J. Optimized on-chip-pipelined mergesort on the Cell/B.E. [C]// Proceedings of the 16th International Euro-Par Conference on Parallel Processing, Part II. Berlin: Springer-Verlag, 2010: 187-198.
- [7] SATISH N, KIM C, CHHUGANI J, *et al.* Fast sort on CPUs and GPUs: a case for bandwidth oblivious SIMD sort [C]// Proceedings of 2010 ACM SIGMOD International Conference on Management of data. New York: ACM Press, 2010: 351-362.
- [8] ZHONG C, QU Z Y, YANG F, *et al.* Efficient and scalable thread-level parallel algorithms for sorting multisets on multi-core systems [J]. Journal of Computers, 2012, 7(1): 30-41.
- [9] ZHONG C, KE Q, LIU J, *et al.* Thread-level parallel algorithm for sorting integer sequence on multi-core computers [C]// Proceedings of the 4th International on Parallel Architectures, Algorithms and Programming. Washington, DC: IEEE Computer Society, 2011: 37-41.
- [10] ZHONG C, FENG P, YIN M X, *et al.* Sampling-based cache-efficient parallel sorting on multi-core systems [J]. Journal of Computational Information Systems, 2012, 8(8): 6713-6722.
- [11] ZHONG C, LI X, YANG F, *et al.* Scheduling divisible loads with return messages on multi-core heterogeneous clusters with unknown system parameters [J]. International Journal of Advancements in Computing Technology, 2012, 4(7): 110-120.
- [12] 柯琦, 钟诚, 李智, 等. 多核计算机上的最大和子序列并行算法 [C]// 计算机技术与应用进展 2010. 合肥: 中国科学技术大学出版社, 2010: 586-590.
- [13] 陈国良. 并行计算——结构·算法·编程 [M]. 修订版. 北京: 高等教育出版社, 2003: 140-141.