计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2016.20148246
53(2):362-373,2016

阵列众核处理器上的高效归并排序算法

石 嵩 李宏亮 朱 巍
(江南计算技术研究所 江苏无锡 214083)
(shisong1990@163.com)

Efficient Merge Sort Algorithms on Array-Based Manycore Architectures

Shi Song, Li Hongliang, and Zhu Wei
(*Jiangnan Institute of Computing Technology*, *Wuxi*, *Jiangsu* 214083)

Abstract Sorting is one of the most fundamental problems in the field of computer science. With the rapid development of manycore processors, it shows great importance to design efficient sort algorithms on manycore architecture. An important trend of manycore processors is array-based manycore architecture. This paper presents two efficient merge sort algorithms on array-based manycore architectures. Our algorithms achieve high performance by using DMA (direct memory access) multi-buffering strategy to improve the memory accessing efficiency, deeply balanced merging strategy to keep load-balancing between cores, SIMD (single instruction multiple data) merging method to exploit fine-grained parallelism and on-chip swap merging strategy to reuse on-chip data. Experimental results on DFMC (deeply-fused many-core) prototype system achieve a sorting rate of 647 MKeys/s (million keys per second), and the sorting efficiency (sorting rate/peak performance) is 3.3x higher than state-of-the-art GPU merge sort on GTX580, and 2.7x higher than the fastest merge sort on Intel Xeon Phi. Additionally, this paper establishes an analytical model to characterize the performance of our algorithms. Based on the analytical model, we analyze the influence of the main structural parameters to the performance of the algorithms, which will contribute to the study of many-core architecture.

Key words array-based manycore; merge sort; sort network; SIMD; SPMD; on-chip communication

摘 要 排序是计算机科学中最基本的问题之一,随着众核处理器结构的不断发展,设计众核结构上的高效排序算法具有重要意义.众核处理器的一个重要方向是阵列众核处理器,根据阵列众核处理器的结构特点,提出了2种面向阵列众核结构的高效归并排序算法,通过利用DMA(direct memory access)多缓冲机制提高访存效率、深度平衡归并策略保持众多核心之间的负载均衡、SIMD(single instruction multiple data)归并方法提高归并计算效率以及片上交换归并策略提高片上数据重用率,大幅度提高了阵列众核处理器的排序性能.在异构融合阵列众核处理器DFMC(deeply-fused many-core)原型系统的实验结果表明,算法排序速度达647 MKeys/s(million keys per second),其排序效率(排序速度/峰值性能)是NVIDIA GPU上最快的归并排序算法(GTX580平台)的3.3倍,是Intel Xeon Phi上最快的归并排序算法的2.7倍.最后,建立了阵列众核处理器上归并排序算法的性能分析模型,利用该模型分析了主要结构参数与算法性能的关系,对阵列众核处理器的研究有一定的指导意义.

收稿日期:2014-11-14;**修回日期**:2015-07-21
基金项目:国家"八六三"高技术研究发展计划基金项目(2014AA01A301);"核高基"国家科技重大专项基金项目(2013zx0102-8001-001-001)
This work was supported by the National High Technology Research and Development Program of China (863 Program) (2014AA01A301) and the National Science and Technology Major Projects of Hegaoji (2013zx0102-8001-001-001).

关键词　阵列众核;归并排序;排序网络;单指令多数据流;单程序多数据流;片上通信

中图法分类号　TP302

排序是计算机科学及算法研究中最基本、最重要的研究问题之一[1],是数据库、图运算、科学计算以及大数据等诸多重要应用的基础,排序效率对这些应用程序的性能有重要的影响,在不同计算平台和环境上不断提高排序的性能,具有重要的现实意义.

近年来,众核处理器在学术界和工业界得到了广泛关注和蓬勃发展,已在多个领域得到了广泛应用.当前主要的众核处理器和众核结构包括 NVIDIA 的 GPU 系列、AMD 的 GPU/APU 系列、Intel 的 MIC(many integrated core)和 SCC(single-chip cloud computer)构架,以及 Tilera[2],PACS-G[3],epiphany[4],MPPA[5],Godson-T[6],DFMC[7]等.其中 Tilera,PACS-G,epiphany,MPPA,Godson-T,DFMC 等属于阵列众核处理器,它们的共同特点是以阵列方式组织众多计算核心,具有可扩展性好、实现代价低的优点,是众核处理器发展的重要方向.

随着这些新型众核处理器的不断涌现,需要重新设计适应特定体系结构的排序算法,才能最大限度地发挥出处理器的运算潜力.国内外学者在 GPU 和 Intel Xeon Phi(属于 MIC 结构)上开展了大量研究工作,并取得了很好的效果[8-11],但阵列众核结构上排序算法的研究尚较少.

尽管阵列众核处理器提供了强大的计算能力,但是要发挥好性能需要用户利用更多的底层硬件特征.针对排序算法,需要解决 3 个关键的问题:1)访存问题,阵列众核中核心数目多,单个核心所能分得的带宽资源少,导致存储墙问题严重;2)众多核心之间的负载均衡问题;3)开发 SIMD 级并行性的问题,虽然阵列众核的计算核心通常提供 SIMD 指令以增加并行性,但是以 SIMD 方式实现排序并不容易.

本文首先建立了阵列众核处理器结构的共性抽象模型,然后设计了面向阵列众核结构的高效归并排序算法.算法通过利用 DMA 多缓冲机制提高访存效率、深度平衡归并策略保持核心间的负载均衡、SIMD 归并方法开发 SIMD 并行性以及片上交换归并策略重用片上数据,有效提升了排序性能.算法在 DFMC 原型系统上的实验结果优于 NVIDIA 众核以及 Intel Xeon Phi 众核上的基于比较的排序算法.最后,本文建立了算法性能分析模型,并用该模型讨论和分析了阵列众核结构与算法性能的关系,为阵列众核结构的研究提供了参考.

1　相关工作

排序问题由来已久,经过半个多世纪的发展,串行排序算法的研究已经非常成熟且许多算法如快速排序、基数排序等已接近理论最优.近些年来,研究人员主要将精力集中在结构相关特别是新型体系结构如 GPU 和 MIC 等众核结构上的排序算法的优化上,取得了许多进展.这些排序算法主要分为 2 类:基数排序和基于比较的排序.

基数排序是最为高效的排序之一,对于 d 位的键值,其时间复杂度是 $O(dN)$,文献[8,12-14]分别提出和实现了 GPU 上的基数排序算法,其中 Merrill 等人[8]的基数排序被集成到了 Thrust 库中,是当前最快的基数排序算法.Satish 等人[10]实现了 MIC 结构上的基数排序算法,有很好的性能.然而,基数排序的时间开销随着键值长度的增加而增长,而且,基数排序不是基于比较的排序,通用性不足.

基于比较的排序是一类通用的排序方法,包括双调排序、快速排序、采样排序(sample sort)和归并排序等,其时间复杂度的下界是 $O(N\log N)$.

双调排序是由 Batcher[15]于 20 世纪 60 年代提出的一种排序网络算法,采用分治法归并序列,时间复杂度是 $O(N\log^2 N)$,虽然不是渐进最优,但是由于其并行性好,且具有访存连续、对界的优点,所以在 GPU 上有很好的运用.文献[16-17]在 GPU 上实现了双调排序,Greb 等人[18]设计实现了 GPU 上自适应双调排序(adaptive bitonic sort)算法,将时间复杂度从 $O(N\log^2 N)$降低到 $O(N\log N)$.双调排序较高的时间复杂度限制了其在数据量很大的排序中的应用.

Sengupta 等人[12]首先实现了 GPU 上的快速排序算法,随后 Cederman 等人[19-20]进一步做了改进.采样排序是快速排序的一种推广,它一次将输入划分成多个部分而非快速排序中的 2 部分,然后独立地排序各个部分.Leischner 等人[21]在 GPU 上实现了采样排序.尽管快速排序的时间复杂度是 $O(N\log N)$,但是其并行程序实现复杂,且性能表现不突出.

归并排序的时间复杂度是 $O(N\log N)$,由于并行性好且时间复杂度低,是 GPU 上广泛使用的排序算法之一[9,14,22-23],这些工作主要针对 GPU 的结

构特点在访存、同步、通信上对算法做优化,使归并排序算法更适合 GPU 结构. 其中 Davidson 等人[9]的归并排序是当前 GPU 上最快的归并排序算法. Tian 等人[11]设计了 Intel Xeon Phi 众核上的归并排序算法,是目前最快的归并排序算法.

由于基数排序的通用性不够,而基于比较的其他算法如双调排序、快速排序等具有时间复杂度高或者并行化困难且并行程序效率不高的缺点,所以本文选取归并排序进行研究.

2 阵列众核处理器结构

阵列众核结构最主要的特征是多个计算核心以阵列方式(mesh)组织,可以分为同构阵列众核和异构阵列众核 2 类,其中同构阵列众核全部由计算核心阵列构成,异构阵列众核在计算核心阵列外还有额外的管理核心,后者可视为前者的一般化情形,本文抽象的阵列众核结构采用后者,但是所设计的算法对于 2 种结构均适用.

阵列众核处理器的整体结构如图 1 所示,包括管理核心 MPE(management processing element)、计算核心阵列 CPA(computing processing array)、存控和系统接口,通过片上网络 NoC 互连.

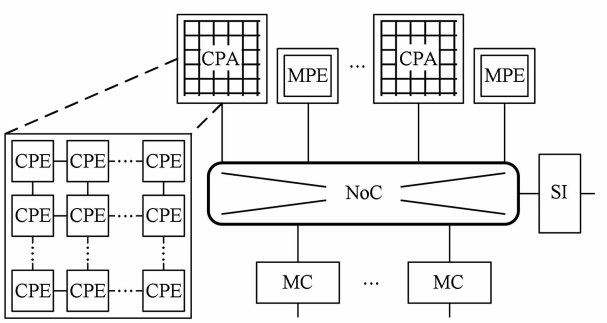


Fig. 1 The array-based manycore architecture.

图 1 阵列众核结构

其中 MPE 是功能完整的通用核心,负责提供一些标准服务如 IO 代理和消息代理等,也适合运行程序的串行部分. CPA 由若干计算核心 CPE (computing processing elements)以阵列方式组织构成,阵列内的 CPE 可通过高速的显式或者隐式的片上通信(如消息)交换数据. CPE 是功能简单的运算核心,支持 SIMD 指令以开发细粒度的并行性、提高峰值性能,用以加速大规模的并行任务.

阵列众核处理器中 CPA 的数据存储构架如图 2 所示. CPE 内具有片上存储器 SPM(scratched pad memory),或 CPA 中共享片上存储器(本文叙述采用前者). SPM 可配置为数据 Cache 结构,也可配置为由软件显式管理的局部存储器,配置为 Cache 结构可以减轻软件编程的负担,但是由于需要保持众多核心之间的 Cache 一致性,效率比作为局部存储器低. 本文将 SPM 视为局部存储器,SPM 与主存之间通过软件显式管理的 DMA 传输数据.

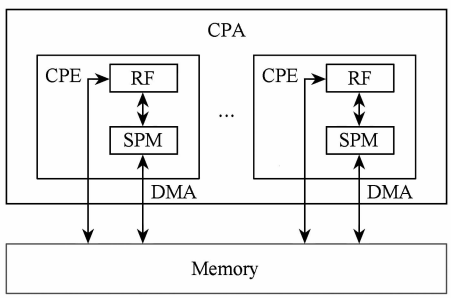


Fig. 2 The data memory hierarchy of CPA.

图 2 CPA 的数据存储构架

从阵列众核结构可以看出,阵列众核提供了 2 种粒度的并行性,核心或阵列之间的粗粒度线程级并行性 SPMD(single program, multiple data)和核心内的细粒度并行性 SIMD,因此设计阵列众核上的并行算法要充分利用这 2 种并行性才能充分发挥处理器的性能. 另外,阵列众核结构提供的阵列内高速片上通信使算法可以通过交换核心间的数据而减少访存操作.

3 阵列众核处理器上的归并排序算法

归并是指将 2 个有序序列合并成 1 个新的有序序列. 归并排序是基于归并操作的排序,它采用分治的策略先将待排序序列分成若干小的子序列,将各个子序列排序后,利用归并操作将各子序列逐步归并成有序序列. 对于长度为 N 的序列,串行算法的时间复杂度是 $O(N\log N)$,空间复杂度是 $O(N)$.

3.1 基本的并行归并排序算法

归并排序是一种自底向上的分治算法,并行归并排序算法将数据划分到各个 CPE,各 CPE 用串行归并排序算法对自己所持数据进行排序后,相互协作将各有序子序列归并成整体有序的序列. 基本并行归并排序的流程如下:

步骤 1. 将 N 个数据均匀划分到 C 个阵列;

步骤 2. 各阵列将数据均匀划分给阵列内的 P 个核心;

步骤 3. 各核心使用归并排序完成 $N/(C\times P)$ 个数据的排序;

步骤 4. 阵列内将各个有序子序列归并成 1 个有序序列;

步骤 5. C 个阵列协作将 C 个有序子序列归并成有序序列.

根据该流程本节首先实现一种基本的并行归并排序算法作为参照,该算法不使用 SIMD 与片上通信,没有显式管理访存,CPE 内的排序使用一般的串行归并排序算法,CPE 间的归并过程如算法 1 所示:

算法 1. 核间基本归并算法.

输入:序列 $A[0,1,\cdots,N-1]$,其中 $A\left[\frac{kN}{P},\cdots,\frac{(k+1)N}{P}-1\right]$ $(k=0,1,\cdots,P-1)$有序;

输出:有序序列 $B[0,1,\cdots,N-1]$, $\forall 0\leqslant i<j<N, B[i]\leqslant B[j]$.

① Copy $A[0,1,\cdots,N-1]$ to $B[0,1,\cdots,N-1]$;
② For $k=1$ To lb(P)
③　If $tid\%2^k==0$ then /* tid 是 CPE 号,即每次归并只有 $P/2^k$ 个 CPE 参与 */
④　　归并 $B[tid\times N/P,\cdots,(tid+2^k)\times N/P-1]$;
⑤　EndIf
⑥ EndFor

基本并行归并排序算法的优点是实现简单,但是该算法没有考虑阵列众核的结构特点,存在以下 3 个问题:1)没有使用 SPM 导致访存延迟大且存储带宽效率低;2)虽然使用了 SPMD,但是各核心间存在负载不均衡,分析算法 1 可知,随着归并级数递增 1 级,参加归并的 CPE 的数目减半;3)没有使用处理器提供的 SIMD 和片上通信对算法进行加速.

根据阵列众核的结构特点,本节接下来从不同层面解决和优化了这些问题和算法,在访存管理上设计了 DMA 多缓冲机制提高访存效率,在线程级并行层次设计了深度平衡归并策略保持核心间的负载均衡,在 SIMD 层次设计了 SIMD 归并方法提高归并计算速度,以及利用片上通信提高片上数据重用率,获得了很好的性能提升,最终实现了 2 种优化的归并排序算法 CPEB-MS(CPE-based merge sort)和 CPAB-MS(CPA-based merge sort). 其中 CPEB-MS 以 CPE 为单元,即各 CPE 先将自己所持数据排序,然后各 CPE 之间进行归并. CPAB-MS 则以 CPA 为单元,即 CPA 作为一个整体将自己所持数据排序,然后 CPA 间进行归并.

3.2　CPEB-MS 算法

CPEB-MS 的整体流程与基本算法相似,先将数据分配给各 CPE,CPE 先用归并排序将所持数据排序,然后 CPE 间协作完成归并. 该算法从 3 个层面——访存管理、线程级并行和 SIMD——进行了优化.

3.2.1　访存管理——DMA 多缓冲机制

基本归并排序算法的访存问题主要是由于 CPE 直接读写主存(使用 LD/ST 指令),一方面小数据量的访存操作造成存储带宽的利用率低,另一方面计算过程和访存过程不能实现并发执行,从而算法整体性能差.

DMA 多缓冲机制在 SPM 中开辟 2 个读缓冲 $bufA$,$bufB$ 和 1 个写缓冲 $bufC$,每个缓冲分为 2 份,构成双缓冲. 以 $bufA$ 为例,分为 $bufA_0$ 和 $bufA_1$,初始时使用 DMA 将数据装填到 $bufA_0$ 中,之后 CPE 使用 $bufA_0$ 中的数据进行计算,同时利用 DMA 将数据装填到 $bufA_1$ 中;当 CPE 计算完 $bufA_0$ 的数据后等待 $bufA_1$ 的装填结果,若已装填好,则 CPE 使用 $bufA_1$ 中的数据同时装填 $bufA_0$,持续这个过程直到完成所有数据的归并.

DMA 多缓冲的运用一方面使得主存的访问次数大为减少,提高了存储带宽的实际效率,CPE 的数据访问延迟大为减小;另一方面,多缓冲使得归并的访存时间和计算时间重叠,最大限度掩盖了访存延迟.

3.2.2　核心负载均衡——深度平衡归并策略

使用多线程主要需要考虑的 2 个问题是负载均衡和通信,归并算法需要的核间通信很少,因此主要要解决负载均衡的问题. 基本归并排序算法中的核间归并过程会造成 CPE 间负载不均衡的问题,即在步骤 3 中,归并级数每增加 1 级,参与归并的 CPE 就减少一半(如图 3(a)所示). 针对这一问题,设计了一种基于采样划分的深度平衡归并策略,保证在归并过程中,所有 CPE 都始终参与归并.

采样划分通过采样的方法将 2 个有序序列 A 和 B 各划分成 p 个不相交的块 $A_0,A_1,\cdots,A_{p-1}$ 和 $B_0,B_1,\cdots,B_{p-1}$,当 $i\neq j$ 时,A_i 和 B_j 不相交. 其中划分元的选取过程如算法 2 所示,根据 $p-1$ 个划分元,分别将 A 和 B 划分成 p 块(长度可能为 0),下标相同的块属于同一组,完成划分.

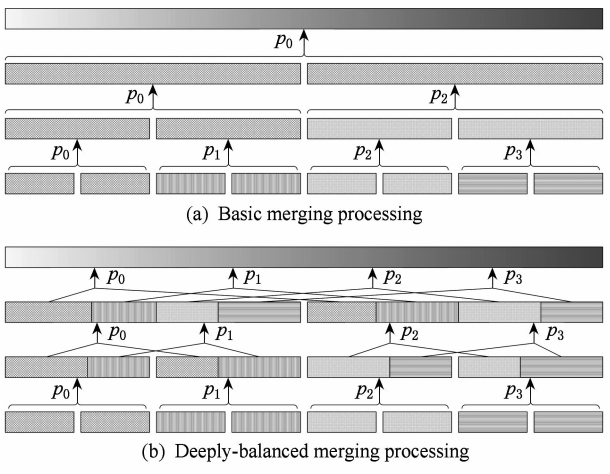


Fig. 3 The basic merging processing and the deeply-balanced merging processing(different textures are merging by different CPE).

图 3 基本归并与深度平衡归并示意图

算法 2. 采样选取划分元算法.

输入:序列 $A[0,1,\cdots,N-1]$, $B[0,1,\cdots,N-1]$各自有序;

输出:$p-1$ 个划分元 $c[1,2,\cdots,p-1]$.

① Choose sample $A[kN/p]$ and $B[kN/p](k=0,1,\cdots,p-1)$ to set S;

② Sort S;

③ For $k=1$ To $p-1$

④ $c[k]=(S[2k]+S[2k+1])/2$;

⑤ EndFor

可以证明采样划分后各个组的数据量是均匀分布(除第 1 组和最后 1 组有偏斜). 深度平衡归并策略如图 3(b)所示,在第 k 级(最底层为第 0 级),用采样划分将 2 个子序列划分成 2^k 组,分别分给 2^k 个 CPE,然后各 CPE 对分给自己的数据进行归并. 从图 3(b)可以看出,每一级归并的 CPE 数目不变且各 CPE 的任务量相当,保持了负载均衡.

3.2.3 使用 SIMD——SIMD 并行归并方法

传统归并过程中的循环体 1 次写回 1 个数据,SIMD 并行归并的目标是 1 次写回 K(SIMD 宽度)个数据,从而减少每个数据的时间摊销.

20 世纪 60 年代,Batcher 等人[15]提出了基于 Batcher 比较器的归并网络,如奇偶归并网络、双调归并网络,可在 $O(\log N)$的时间内完成归并. 图 4 显示了一个长度为 8 的双调归并网络,垂直线表示比较器,上(下)端输出较小(大)值,网络通过 $\mathrm{lb}(8)=3$ 级比较操作,可以将 2 个长度为 4 的有序序列 $A[0,1,2,3]$和 $B[0,1,2,3]$归并成 1 个有序序列.

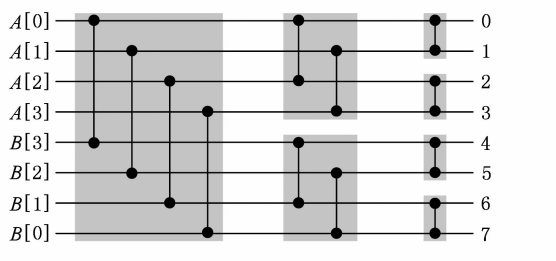


Fig. 4 Bitonic merge network.

图 4 双调归并网络

利用归并网络,可以实现 1 次输出多个数据的目标,长度为 $2K$ 的双调归并网络,每次可输出 K 个结果(较小的 K 个数据),另外的 K 个数据则继续和接下来读入的 K 个数据进行归并,重复该过程直到归并完所有数据. 归并网络可映射成 SIMD 指令实现,如图 4 中,A 和 B 分别放置在 2 个向量寄存器中,则每一级的 4 个比较器可以用 CPE 中 SIMD 指令集中的 1 条 vmax(取较大值)指令和 1 条 vmin(取较小值)指令实现,而在 1 次比较之后,2 个寄存器的值可通过混洗指令 vshuffle 混洗,以满足下一级比较的输入位置要求.

为了运用 SIMD 归并网络,需要先将输入序列排成长 K 有序的子序列. 该过程可通过基于排序网络和混洗操作的寄存器排序完成,如图 5 所示,将 K^2 个元素放置在 K 个寄存器中,首先用长度为 K 的奇偶归并排序网络将它们(如 $A[0,1,2,3]$)排序成 K 个长 K 有序的序列(如 a_0,a_1,a_2,a_3),如图 5(a)所示,然后通过 $K\,\mathrm{lb}(K)$次 vshuffle 操作将各有序序列混洗到单个寄存器中(图 5(b)).

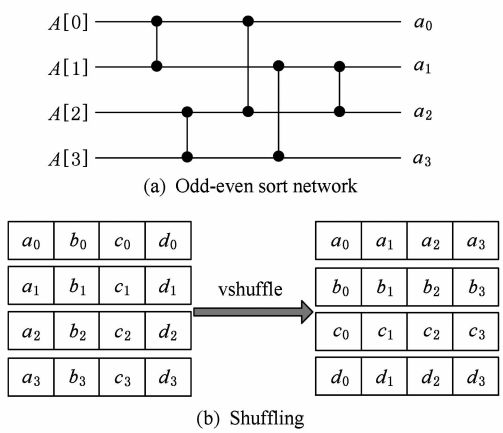


Fig. 5 In-register sort with SIMD.

图 5 SIMD 寄存器排序

3.2.4　CPEB-MS 算法流程

结合上述 3.2.1～3.2.3 小节的内容,总结出 CPEB-MS 算法整体流程如下:

步骤 1. 将 N 个数据均匀划分给各 CPE(C 个 CPA,每个 CPA 中 P 个 CPE,共 $C\times P$ 个 CPE),每个核心有 $N/(C\times P)$个数据;

步骤 2. 各 CPE 使用步骤 2.1～2.2 完成 $N/(C\times P)$个数据的排序:

2.1) 将所持数据分为 $N/(C\times P)/M$ 个组,每组 M 个数据(可以存放于 SPM 缓冲中),使用步骤 2.1.1～2.1.2 对每一组进行排序:

2.1.1) 用寄存器排序将序列排序成长 K(SIMD 宽度)有序的序列;

2.1.2) 使用 lb(M/K)次 SIMD 归并,逐步将长 K 有序的子序列归并成 1 条有序序列;

2.2) 各 CPE 使用 lb($N/(C\times P)/M$)次 SIMD 归并将自己所有的长为 M 的有序子序列归并成 1 个有序序列(读写数据使用 DMA 多缓冲技术);

步骤 3. CPA 内各 CPE 利用 lb(P)次深度平衡归并策略将 P 个有序子序列归并成 1 个有序序列(读写数据使用 DMA 多缓冲技术);

步骤 4. C 个 CPA 继续应用 lb(C)次深度平衡归并策略将 C 个有序子序列归并成整体有序序列(读写数据使用 DMA 多缓冲技术).

3.3　CPAB-MS 算法

CPEB-MS 中各 CPE 首先将所持数据排序成有序序列然后 CPE 间进行归并的流程没能利用阵列众核结构提供的片上通信支持来减少访存次数.因此本文继续设计了 CPAB-MS 算法,它将 CPA 视为一个整体,算法将数据分给各 CPA,各 CPA 先将所持数据排序,然后 CPA 之间进行归并.

3.3.1　使用片上通信——片上交换归并策略

在 CPEB-MS 算法流程中的步骤 2.1 中,各 CPE 排序 M 个元素后即将其写回主存,接着处理后 M 个元素.与此不同,片上交换归并的策略是,各 CPE 排序完 M 个元素后不直接写回主存,而是由阵列选出 $P-1$ 个划分元,各 CPE 根据划分元划分所持的 M 个元素,将各分块发送到其他对应的 CPE(数据交换),数据交换后,归并所接收的各分块,然后 CPA 内利用前缀和计算出各 CPE 所持数据的偏移位置,将其写回主存,得到长为 $P\times M$ 的有序序列.

其中,划分元的选取和深度平衡归并中的方法类似,不同的是,这里为了降低空间复杂度采用 2 级采样划分策略:行划分和列划分.首先是行划分,每个 CPE 选取 $P-1$ 个代表元发送到本行第 1 列的核心 $P_{x0}(x=0,1,\cdots,P-1)$,P_{x0} 对这些数据进行排序后选取 $P-1$ 个代表元发送给 P_{00},P_{00} 对其排序后选出 $P-1$ 个划分元广播给所有 CPE,完成划分元的选取.与直接采样划分策略相比,这里的 2 级划分策略的空间复杂度从 P^2 降低到了 $P^{3/2}$.

数据交换过程是一个全局通信操作,由于各 CPE 待交换的数据长度可能不一致,通信过程非常复杂.为了解决该问题,本文设计了基于握手的点对点通信策略来完成全局通信:共 $P-1$ 次点对点通信,每次通信在数据传输之前源节点先将子序列长度发送给目的节点(握手).第 $i\sqrt{P}+j$ 次通信中(如图 6 所示),节点 $P_{m,n}$ 发送第$(m\oplus i)\sqrt{P}+(n\oplus j)$块数据 $A_{m\oplus i,n\oplus j}$[①] 给 $P_{m\oplus i,n\oplus j}$,而从 $P_{m\ominus i,n\ominus j}$ 那里接收数据,如果这 2 个节点不同行且不同列,则 $P_{m,n}$ 与 $P_{m\oplus i,n\oplus j}(P_{m\ominus i,n\ominus j})$ 之间的通信需要 1 个转发节点 $P_{m\oplus i,n}(P_{m,n\ominus j})$转发,源节点先把数据发送到转发节点上,然后由转发节点发送至目的节点.该通信模式可以保证所有 CPE 能并发进行点对点通信,不会出现空闲或者死锁的状态.

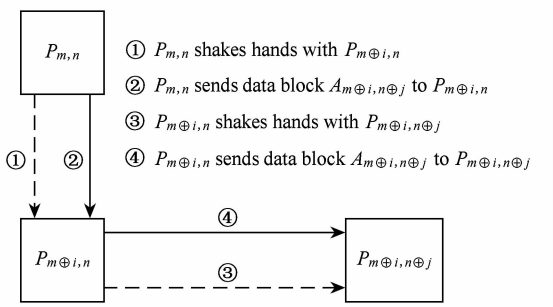


Fig. 6　Data communication processing.

图 6　数据发送过程示意图

3.3.2　CPAB-MS 算法流程

综合 3.2 节和 3.3.1 节所设计的 4 项优化技术,总结出 CPAB-MS 算法整体流程如下:

步骤 1. 将 N 个数据均匀划分到 C 个 CPA,每个阵列有 N/C 个数据;

步骤 2. 各 CPA 将所持数据分为$(N/C)/(P\times M)$个组,每组 $P\times M$ 个数据(可以存放于 CPA 内各 CPE 的 SPM 中),使用步骤 2.1～2.2 对每一组数据进行排序:

① 本文⊕/⊖是模$\sqrt{P}$的加减运算

2.1) 每个 CPE 读取 M 个数据到 SPM,使用步骤 2.1.1～2.1.2 完成 M 个数据的排序;

2.1.1) 用寄存器排序将序列排序成长 K (SIMD 宽度)有序的序列;

2.1.2) 使用 lb(M/K)次 SIMD 归并,逐步将长 K 有序的子序列归并成 1 条有序序列;

2.2) CPA 内各 CPE 通过片上交换归并策略将 $P\times M$ 个数据排成有序序列;

步骤 3. CPA 内各 CPE 利用 lb($N/(C\times P\times M)$)次深度平衡归并策略将 $N/(C\times P\times M)$个有序子序列归并成 1 个有序序列(读写数据使用 DMA 多缓冲技术);

步骤 4. C 个 CPA 继续应用 lb(C)次深度平衡归并策略将 C 个有序子序列归并成整体有序序列(读写数据使用 DMA 多缓冲技术).

4 实验结果与分析

4.1 实验环境

异构融合阵列众核处理器 DFMC 的结构包括 4 个 MPE,4 个 CPA 共 256 个 CPE,主频 1.0 GHz,支持 128 b(2 个 64 b 整数或浮点运算)的 SIMD 指令,每个 CPE 拥有 32 KB 局存,局存访问延迟 3 周期,局存与主存间支持多种 DMA 模式传输,CPA 内支持片上通信,单次主存访问延迟约 100 周期,主存带宽 102.4 GBps,芯片 32 位整数运算峰值性能 512GOps(giga operations per second). 软件配置是,MPE 运行在内核为 2.6.28 版本的 Linux 系统上,CPE 的系统是一个轻量级的定制系统,编程环境是定制的提供并行编程的 C 语言.

目前 DFMC 原型系统采用 FPGA 构建,时钟频率 2.6 MHz,如图 7 所示. 实验在该原型系统上进行,所得的实验结果根据真实的时钟频率进行校准.

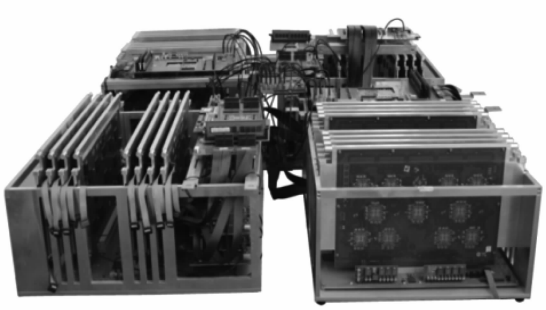


Fig. 7 DFMC FPGA prototype system.

图 7 FPGA 搭建的 DFMC 原型系统

4.2 优化效果

实验中待排序的数据是均匀分布的 32 位整数序列,数据量从 2^{20} 增长到 256×2^{20},实验算法包括基本并行归并、CPEB-MS 和 CPAB-MS 三种. 图 8 显示了 3 种算法的排序速率(序列长度与排序时间的比值),结果表明,与基本算法相比,CPEB-MS 和 CPAB-MS 的排序速度得到了巨大的提升,平均分别提升了 158 倍和 194 倍. CPAB-MS 的排序速度相对 CPEB-MS 平均提升了 23%,这是因为 CPAB-MS 通过片上通信重用片上数据减少了访存次数. 然而,CPAB-MS 中片上通信是有代价的,将在第 5,6 节进一步分析.

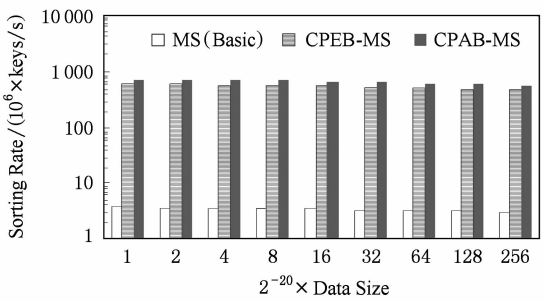


Fig. 8 Sorting rates of the three merge sort algorithms for varying sequence sizes.

图 8 3 种归并排序算法的排序速率

其中各项优化技术所提供的加速比如表 1 所示. 从表 1 可以看出,多缓冲和深度平衡归并所提供的加速比非常大,说明基本算法中由于访存低效和负载不均衡导致的性能问题非常严重. 与之相比,SIMD 归并和片上交换归并策略所提供的性能增益相对较小,这是因为通过前 2 项技术后,存储带宽的利用率已接近极限,此时,SIMD 计算所带来的性能提升受限于访存瓶颈,而片上交换归并所减少的访存次数受限于 CPA 内各 CPE 的 SPM 大小之和.

Table1 Speedups of All Optimization Techniques

表 1 各优化技术加速比

Optimization Strategy	Multi-Buffering	Deeply-Balanced Merging	SIMD Merging	On-Chip Swap Merging
Speedup	18.16	6.29	1.39	1.23

4.3 与其他平台排序算法的比较

实验选取了近几年来众核处理器上效率最高的几种比较型排序算法进行对比. 在 GPU 方面,选取了 NVIDIA GTX580 上的归并排序(GPU Merge)[9]、NVIDIA Tesla C1060 上的采样排序(GPU Sample)[21]、

CPU-GPU(Intel i7 980+GTX580)混合结构上的采样排序(Hybrid Sample)[24];在 Intel Xeon Phi 众核上,选取了归并排序(Xeon Phi Merge)[11]进行比较,实验数据均来源于公开发表的文献.表 2 显示了这些处理器的性能参数.

Table 2　The Parameters of Referred Processors

表 2　处理器参数

Processor	Performance/GOps	Bandwidth/GBps
DFMC	512	102.4
GTX580	622	192.4
NVIDIA Tesla C1060	311	102.4
Intel Xeon Phi	1 010	320

实验结果如图 9 所示,可以看出,在所有这些比较型排序中,DFMC 上的实验结果是最好的,如数据量为 16×2^{20} 时,DFMC Merge 的排序速度达到 647 MKeys/s,是 GPU Merge 的 2.7 倍,是 GPU Sample 的 5.5 倍,是 CPU-GPU 混合结构上 Hybrid Sample 的 2.4 倍;比 Xeon Phi 上归并排序排序快 36%.折算成同等峰值性能,DFMC 上的归并排序的排序效率(排序速率与峰值性能的比值)是 GTX580 归并排序的 3.3 倍,是 Xeon Phi 上归并排序的 2.7 倍.CPAB-MS 算法比其他处理器上比较型排序算法效率高的主要原因是 CPAB-MS 充分利用了阵列众核结构的硬件特征,最大限度地发挥了处理器的性能.

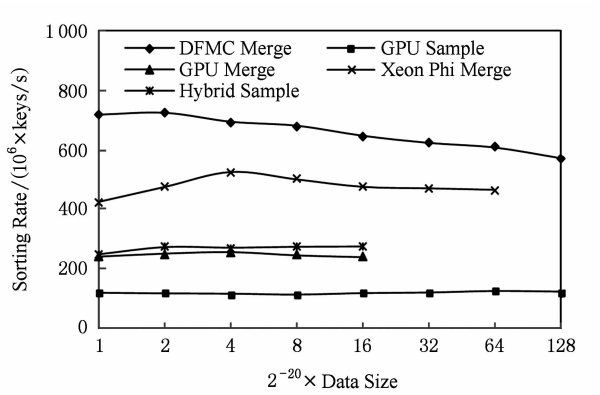


Fig. 9　Sorting performance of different methods on different platforms.

图 9　不同平台排序算法性能比较

5　算法性能模型

本节建立算法性能的分析模型,用以指导算法的优化以及分析阵列众核结构与算法性能之间的关系.

5.1　模型建立

阵列众核上并行归并程序的执行时间主要包括访存时间、计算时间和片上通信时间(针对 CPAB-MS).设 τ_m 是归并过程中单次单个元素的平均访存时间,τ_c 是归并过程中单次单个元素的平均计算时间,τ_s 是单个元素的平均寄存器排序时间,τ_i 是单个元素的平均片上通信时间.则 CPEB-MS 算法框架中步骤 2 的时间是

$$T_{cpe,step2}=\frac{N}{C\times P}\left[\tau_s+\tau_c\,\mathrm{lb}\left(\frac{M}{K}\right)+2\tau_m\right]+\frac{N}{C\times P}\left[\mathrm{lb}\left(\frac{N}{C\times P}\right)-\mathrm{lb}(M)\right]\times\max(2\tau_m,\tau_c),\quad(1)$$

其中 N,M,K 分别表示数据量、SPM 缓冲中可容纳的元素数目和 SIMD 宽度,C 表示阵列数,P 表示每个阵列中的核心数.$\max(2\tau_m,\tau_c)$是由于多缓冲使得计算时间和访存时间重叠,总的执行时间是这二者的最大值,τ_m 前的系数 2 说明读写 2 次操作.步骤 3 和步骤 4 的时间之和是

$$T_{cpe,step3-4}=\frac{N}{C\times P}\mathrm{lb}(P)\times\max(2\tau_m,\tau_c)+\frac{N}{C\times P}\mathrm{lb}(C)\times\max(2\tau_m,\tau_c).\quad(2)$$

CPEB-MS 的总执行时间是这 3 个步骤执行时间之和(步骤 1 时间可忽略不计):

$$T_{cpe}=\frac{N}{C\times P}\left[\tau_s+\tau_c\,\mathrm{lb}\left(\frac{M}{K}\right)+2\tau_m\right]+\frac{N}{C\times P}[\mathrm{lb}(N)-\mathrm{lb}(M)]\times\max(2\tau_m,\tau_c).\quad(3)$$

对于 CPAB-MS 算法,步骤 2 的执行时间是

$$T_{cpa,step2}=\frac{N}{C\times P}\left[\tau_s+\tau_c\,\mathrm{lb}\left(\frac{M}{K}\right)+\tau_m\right]+T_{alltoall}+\frac{N}{C\times P}[\tau_c\,\mathrm{lb}(P)+\tau_m],\quad(4)$$

其中 $T_{alltoall}$ 是片上通信时间,片上通信时间与数据分布有关,最好的情形,即已经完全有序,无需通信,$T_{alltoall}=O(1)$;最坏的情形,每次点对点通信的传输数据量达到 $N/(C\times P)$,$T_{alltoall}=P\times N/(C\times P)\times2\tau_i$;一般情形,平均时间是 $T_{alltoall,ave}=N/(C\times P)\times2\tau_i$.

步骤 3 和步骤 4 的执行时间之和是

$$T_{cpa,step3-4}=\frac{N}{C\times P}\mathrm{lb}\left(\frac{N}{C\times P\times M}\right)\times\max(2\tau_m,\tau_c)+\frac{N}{C\times P}\mathrm{lb}(C)\times\max(2\tau_m,\tau_c).\quad(5)$$

CPAB-MS 算法的总执行时间是这 3 个步骤执行时间之和:

$$T_{\text{cpa}}=\frac{N}{C\times P}\left[\tau_{\text{s}}+\tau_{\text{c}}\text{lb}\left(\frac{P\times M}{K}\right)+2\tau_{\text{m}}\right]+$$

$$T_{\text{alltoall}}+\frac{N}{C\times P}\text{lb}\left(\frac{N}{P\times M}\right)\times\max(2\tau_{\text{m}},\tau_{\text{c}}).\quad(6)$$

观察式(3)和式(6),发现二者均可表示成如下形式,t_{other}是其他与 τ_{m} 和 τ_{c} 无关的时间:

$$T=a\times 2\tau_{\text{m}}+b\times\tau_{\text{c}}+t_{\text{other}};\quad(7)$$

$$a+b=N/(C\times P)\times[\text{lb}(N)-\text{lb}(K)+1].\quad(8)$$

式(3)、式(6)~(8)可以作为阵列众核上归并排序算法的性能模型,通过调整系数,可以表示不同结构上的性能模型. 例如,若众核不支持 SIMD 指令,则 $K=1$;若众核具有共享片上存储,则 $T_{\text{alltoall}}=0$.

5.2 性能模型误差

我们在 DFMC 原型系统上检验了算法性能模型的准确度. 性能模型中的 $\tau_{\text{m}},\tau_{\text{c}},\tau_{\text{s}},\tau_{\text{i}}$ 可通过程序代码、存储带宽和时钟频率计算出来,其他系数可由结构参数得到,从而算出执行时间的估计值. 估计值和实验值的结果如图 10 所示. 其中 CPEB-MS 算法性能模型的平均误差在 5%以内,CPAB-MS 算法性能模型的平均误差会达到 11%,这是因为 CPAB-MS 中片上通信过程里有大量的同步操作,带来一些额外的不确定性时间.

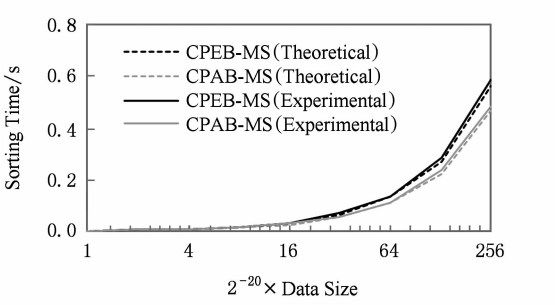


Fig. 10 Comparison between estimated values and experimental values.

图 10 实验值与理论值比较

从实验结果可以看出,算法性能模型的准确度高,可以用来分析不同结构参数下的算法性能.

6 进一步讨论

本节利用第 5 节建立的算法性能模型,对算法优化以及算法性能与众核结构之间的关系进行一些分析和探索.

在算法优化方面,可以分析出如下结论:

1) 阵列众核上归并排序算法的时间复杂度是 $O(N\text{ lb}(N)/(C\times P))$,降低并行归并排序时间的主要途径是降低 τ_{m} 和 τ_{c}. 本文设计的多缓冲策略和 SIMD 归并方法即是降低了 τ_{m} 和 τ_{c}.

2) 从式(8)可以看出 τ_{m} 和 τ_{c} 的系数之和是与算法无关的恒定值. 对于核数较多的众核结构,通常 τ_{m} 比 τ_{c} 大,因此提高排序性能的一种策略是减少 a 而增加 b 来降低整个执行时间,即以增加计算来减少访存时间,本文中 CPAB-MS 与 CPEB-MS 相比就是通过重用片上数据减小访存次数来提高性能.

3) CPAB-MS 的执行时间中 T_{alltoall} 受待排序序列数据分布的影响,最坏情形的通信开销大,性能将比 CPEB-MS 差,而一般情形时,其性能比 CPEB-MS 好.

阵列众核结构中的核心数目、存储带宽、SPM 大小以及 SIMD 宽度,与算法性能密切相关,可以依据算法性能模型进行定量分析.

6.1 核心数目与存储带宽

核心数目对排序性能的影响有 2 方面,一方面核心数目越多,计算能力越强,单位时间可完成的比较操作越多;另一方面,核心数目的增加会导致单个核心所分得的存储带宽减小,从而排序容易陷入访存瓶颈.

在第 4 节的阵列众核归并排序性能模型中,式(3)和式(6)中的 τ_{m} 是和存储带宽以及核心数目有关的变量,设带宽为 B(单位为 Bps),CPE 数目为 P,存储带宽利用率为 ρ,键值大小为 w(单位为 B),则

$$\tau_{\text{m}}=\frac{P\times w}{\rho\times B}.\quad(9)$$

以式(3)为例,当 N 很大时,执行时间主要是由式中第 2 项决定,记为 $T_{\text{cpe,second}}$. 而当 $2\tau_{\text{m}}>\tau_{\text{c}}$ 时,将式(9)代入,有

$$T_{\text{cpe,second}}=\frac{N}{C\times P}[\text{lb}(N)-\text{lb}(M)]\times 2\tau_{\text{m}}=$$

$$\frac{2w\times N}{\rho\times B\times C}[\text{lb}(N)-\text{lb}(M)],\quad(10)$$

由式(10)可看出 $T_{\text{cpe,second}}$ 与 P 无关,即使增加 P 也无法再减少 $T_{\text{cpe,second}}$,继而对 T_{cpe} 的影响非常有限. 因此,$2\tau_{\text{m}}=\tau_{\text{c}}$ 是一个临界条件,即

$$P=\frac{\rho\times\tau_{\text{c}}}{2w}B.\quad(11)$$

若 $2\tau_{\text{m}}<\tau_{\text{c}}$ 说明排序是计算受限的,而 $2\tau_{\text{m}}>\tau_{\text{c}}$ 则是访存受限. 当未来阵列众核处理器核心数目和存储带宽的扩展关系满足式(11)时,算法有很好的可扩展性.

根据当前 DFMC 的配置情况,可以算出 $P=42$ 是临界点,即当前 DFMC 结构为访存受限.单个 CPA 上使用不同 CPE 数目的 CPEB-MS 算法的实验结果如图 11 所示(数据量为 16×2^{20}).从图 11 可以看出临界 CPE 数在 32～64 之间,之后,执行时间随 CPE 数目增加的下降速度变缓慢.根据式(11),DFMC 单个 CPA 的带宽应当提高约 50%,才能最大限度发挥 64 个 CPE 的计算性能,此时,算法的排序速度将提高 25%.

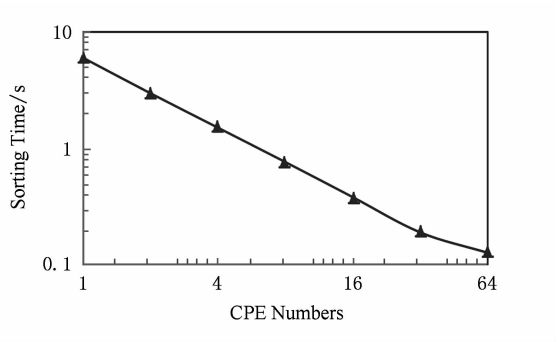


Fig. 11　Sorting time for varying number of CPEs.

图 11　排序时间与 CPE 数目关系

6.2　片上存储大小

根据算法性能模型式(6),可算出 DFMC 结构上排序时间与片上存储开辟的缓冲区大小的关系,如图 12 所示(数据量为 16×2^{20}).从图 12 可以看出,排序时间随缓冲区大小增长呈对数级下降,可见,通过增大 SPM 大小可以提升排序性能,但空间有限.

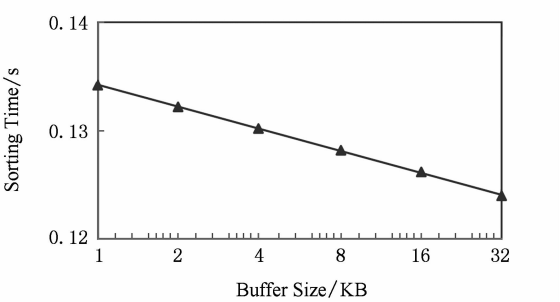


Fig. 12　Sorting time for varying buffer sizes.

图 12　排序时间随缓冲大小关系

6.3　SIMD 宽度

SIMD 归并算法中,对于宽度为 K 的 SIMD 指令,归并 K 个元素需要 1 个 $\mathrm{lb}(2K)$级的归并网络,每级网络需要 2 次比较操作和 2 次混洗操作,所以前面性能模型中的 τ_c 可表示成

$$\tau_c=\Theta\left(\frac{\mathrm{lb}(2K)}{K}\right),\tag{12}$$

其中 Θ 表示紧致界[25],由式(12)结合式(3)、式(6)可以看出,访存理想的情况下,排序时间会随着 K 的增加而呈式(12)下降.但实际的情况是,当随着 K 的增加,算法会达到访存受限,之后执行时间的下降空间有限.图 13 显示了 DFMC 结构上排序时间与 SIMD 宽度的关系,当前 DFMC 的 SIMD 宽度为 2,由于存储带宽的限制,SIMD 宽度的进一步加大所带来的性能提升有限.但是,如果存储带宽增加 1 倍,同时 SIMD 宽度增加 1 倍,将使得排序速度提升 71%.

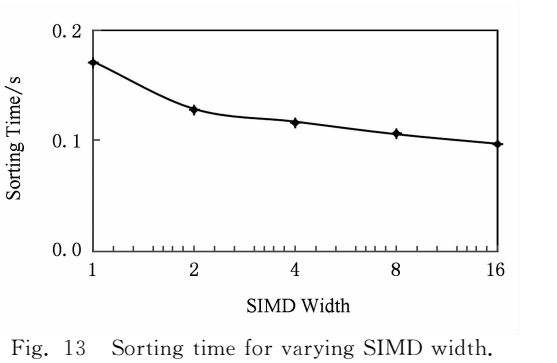


Fig. 13　Sorting time for varying SIMD width.

图 13　排序时间随 SIMD 宽度关系

7　总　　结

阵列众核处理器是众核处理器发展的重要方向,能够提供很高的峰值性能,但如何利用其多级并行性(SPMD 和 SIMD)和 SPM 的结构特点、充分发挥其峰值性能,对并行算法的设计提出了较高的要求.

本文通过分析阵列众核处理器的结构特点,设计了多种相应的优化策略,大幅度提高了阵列众核结构上的归并排序算法性能,实验结果优于其他众核处理器上基于比较的排序算法.本文的优化过程说明了,并行算法的设计要和特定的处理器结构紧密结合起来才能最大限度地发挥处理器的性能.本文所展示的阵列众核结构上归并排序算法的优化过程对于其他处理器上并行算法的设计和优化也有很好的借鉴意义.

本文最后分析了阵列众核结构与代表数据密集型应用的归并排序算法的性能之间的关系,相信这些分析对阵列众核结构的研究是一个有益的参考.

参　考　文　献

[1]　Zhong Cheng, Chen Guoliang. An optimal parallel sorting algorithm for multisets [J]. Journal of Computer Research and Development, 2003, 40(2): 336-341 (in Chinese)
(钟诚,陈国良. Multisets 排序的最优并行算法[J]. 计算机研究与发展, 2003, 40(2): 336-341)

[2] Ramey C. Tile-gx100 manycore processor: Acceleration interfaces and architecture [OL]. San Jose, CA: Tilera Corporation, 2011 [2014-10-25]. http://www.hotchips.org/wp-content/uploads/hc_archives/hc23/HC23.18.2-security/HC23.18.220-TILE-GX100-Ramey-Tilera-e.pdf

[3] Mitsuhisa S. Feasibility study on future HPC infrastructure [OL]. Tsukuba, Janpan: University of Tsukuba, 2014 [2014-10-25]. http://www.ccs.tsukuba.ac.jp/files/ex-review/FS-ccs-eval-2014.pdf

[4] Gwennup L. Adapteva: More flops, less watts [OL]. Mountain View, CA: The Linley Group, 2011 [2014-10-25]. http://www.adapteva.com/wp-content/uploads/2011/06/adapteva_mpr.pdf

[5] Dinechin B D, Ayrignac R, Beaucamps P E, et al. A clustered manycore processor architecture for embedded and accelerated applications [C] //Proc of the 17th IEEE Conf on High Performance Extreme Computing. Piscataway, NJ: IEEE, 2013: 1-6

[6] Fan D R, Yuan N, Zhang J C, et al. Godson-T: An efficient many-core architecture for parallel program executions [J]. Journal of Computer Science and Technology, 2009, 24(6): 1061-1073

[7] Zheng Fang, Zhang Kun, Wu Guiming, et al. Architecture techniques of many-core processor for energy-efficient in high performance computing [J]. Chinese Journal of Computers, 2014, 37(10): 2176-2186 (in Chinese)
(郑方, 张昆, 邬贵明, 等. 面向高性能计算的众核处理器结构级高能效技术[J]. 计算机学报, 2014, 37(10): 2176-2186)

[8] Merrill D G, Grimshaw A S. Revisiting sorting for GPGPU stream architectures [C] //Proc of the 19th Int Conf on Parallel Architectures and Compilation Techniques. New York: ACM, 2010: 545-546

[9] Davidson A, Tarjan D, Garland M, et al. Efficient parallel merge sort for fixed and variable length keys [C] //Proc of Innovative Parallel Computing. Piscataway, NJ: IEEE, 2012: 1-9

[10] Satish N, Kim C, Chhugani J, et al. Fast sort on CPUs, GPUs and Intel MIC architectures [OL]. Santa Clara, CA: Intel Labs, 2010 [2014-10-25]. http://www.intel.com/content/www/us/en/research/intel-labs-radix-sort-mic-report.html

[11] Tian X, Kamil R, Reiji S. Register level sort algorithm on multi-core SIMD processors [C] //Proc of the 3rd Workshop on Irregular Applications: Architecture and Algorithms. New York: ACM, 2013: No 9

[12] Sengupta S, Harris M, Zhang Yao, et al. Scan primitives for GPU computing [C] //Proc of the 22nd ACM SIGGRAPH/EUROGRAPHICS Symp on Graphics hardware. Aire-la-Ville, Switzerland: Eurographics Association, 2007: 97-106

[13] Satish N, Harris M, Garland M. Designing efficient sorting algorithms for manycore GPUs, NVR-2008-001 [R]. Santa Clara, CA: NVIDIA Corporation, 2008

[14] Satish N, Harris M, Garland M. Designing efficient sorting algorithms for manycore GPUs [C] //Proc of the 23rd IEEE Int Symp on Parallel & Distributed Processing. Piscataway, NJ: IEEE, 2009: 1-10

[15] Batcher K E. Sorting networks and their applications [C]// Proc of the Spring Joint Computer Conf. New York: ACM, 1968: 307-314

[16] Purcell T J, Donner C, Cammarano M, et al. Photon mapping on programmable graphics hardware [C] //Proc of the ACM SIGGRAPH/EUROGRAPHICS Conf on Graphics hardware. Aire-la-Ville, Switzerland: Eurographics Association, 2003: 41-50

[17] Peters H, Schulz-Hildebrandt O, Luttenberger N. Fast in-place, comparison-based sorting with CUDA: A study with bitonic sort [J]. Concurrency and Computation: Practice and Experience, 2011, 23(7): 681-693

[18] Greb A, Zachmann G. GPU-ABiSort: Optimal parallel sorting on stream architectures [C] //Proc of the 20th Int Conf on Parallel and Distributed Processing. Piscataway, NJ: IEEE, 2006: 45-54

[19] Cederman D, Tsigas P. A practical quicksort algorithm for graphics processors [G] //LNCS 5193: Proc of the 16th Annual European Symp on Algorithms. Berlin: Springer, 2008: 246-258

[20] Cederman D, Tsigas P. GPU-quicksort: A practical quicksort algorithm for graphics processors [J]. Journal of Experimental Algorithmics (JEA), 2009, 14(4): 1-22

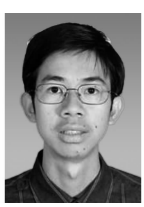
[21] Leischner N, Osipov V, Sanders P. GPU sample sort [C] // Proc of the 24th IEEE Int Symp on Parallel Distributed Processing. Piscataway, NJ: IEEE, 2010: 1-10

[22] Sintorn E, Assarsson U. Fast parallel GPU-sorting using a hybrid algorithm [J]. Journal of Parallel and Distributed Computing, 2008, 68(10): 1381-1388

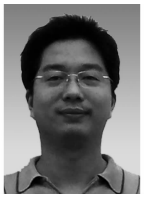
[23] Ye X, Fan D, Lin W, et al. High performance comparison-based sorting algorithm on many-core GPUs [C] //Proc of the 24th IEEE Int Symp on Parallel Distributed Processing. Piscataway, NJ: IEEE, 2010: 1-10

[24] Banerjee D S, Sakurikar P, Kothapalli K. Fast, scalable parallel comparison sort on hybrid multicore architectures [C] //Proc of the 27th IEEE Int Symp on Parallel and Distributed Processing Workshops & PhD Forum. Piscataway, NJ: IEEE, 2013: 1060-1069

[25] Cormen T H, Leiserson C E, Rivest R L, et al. Introduction to Algorithms [M]. Cambridge, MA: MIT Press, 2001



Shi Song, born in 1990. MSc. Member of China Computer Federation. His research interests include computer architecture, microprocessor design and high-performance computing.



Li Hongliang, born in 1975. PhD, associate professor. Member of China Computer Federation. His main research interests include computer architecture, microprocessor design and high-performance computing (hongliangli@263.net).

Zhu Wei, born in 1976. MSc, senior engineer. His main research interests include microprocessor design and integrated circuit verification (zw84611@sina.com).

2013 年《计算机研究与发展》高被引论文 TOP10

排名	论文信息
1	孟小峰，慈祥. 大数据管理:**概念、技术与挑战**[J]. 计算机研究与发展，2013，50(1)：146-169 Meng Xiaofeng and Ci Xiang . Big Data Management: Concepts, Techniques and Challenges [J]. Journal of Computer Research and Development, 2013, 50(1): 146-169
2	傅颖勋，罗圣美，舒继武. **安全云存储系统与关键技术综述**[J]. 计算机研究与发展，2013，50(1)：136-145 Fu Yingxun, Luo Shengmei, and Shu Jiwu. Survey of Secure Cloud Storage System and Key Technologies. Journal of Computer Research and Development, 2013, 50(1): 136-145
3	李建中，刘显敏. **大数据的一个重要方面:数据可用性**[J]. 计算机研究与发展，2013，50(6)：1147-1162 Li Jianzhong and Liu Xianmin. An Important Aspect of Big Data: Data Usability [J]. Journal of Computer Research and Development, 2013, 50(6): 1147-1162
4	刘大有，陈慧灵，齐红，杨博. **时空数据挖掘研究进展**[J]. 计算机研究与发展，2013，50(2)：225-239 Liu Dayou, Chen Huiling, Qi Hong, and Yang Bo. Advances in Spatiotemporal Data Mining. Journal of Computer Research and Development, 2013, 50(2): 225-239
5	廖彬，于炯，孙华，年梅. **基于存储结构重配置的分布式存储系统节能算法**[J]. 计算机研究与发展，2013，50(1)：3-18 Liao Bin, Yu Jiong, Sun Hua, and Nian Mei. Energy-Efficient Algorithms for Distributed Storage System Based on Data Storage Structure Reconfiguration [J]. Journal of Computer Research and Development, 2013, 50(1): 3-18
6	贾冬艳，张付志. **基于双重邻居选取策略的协同过滤推荐算法**[J]. 计算机研究与发展，2013，50(5)：1076-1084 Jia Dongyan and Zhang Fuzhi. A Collaborative Filtering Recommendation Algorithm Based on Double Neighbor Choosing Strategy [J]. Journal of Computer Research and Development, 2013, 50(5): 1076-1084
7	武建佳，赵伟. **WInternet:从物网到物联网**[J]. 计算机研究与发展，2013，50(6)：1127-1134 Wu Jianjia and Zhao Wei. WInternet: From Net of Things to Internet of Things [J]. Journal of Computer Research and Development, 2013, 50(6): 1127-1134
8	余凯，贾磊，陈雨强，徐伟. **深度学习的昨天、今天和明天**[J]. 计算机研究与发展，2013，50(9)：1799-1804 Yu Kai, Jia Lei, Chen Yuqiang, and Xu Wei. Deep Learning: Yesterday, Today, and Tomorrow [J]. Journal of Computer Research and Development, 2013, 50(9): 1799-1804
9	熊金波，姚志强，马建峰，李凤华，李琦. **基于行为的结构化文档多级访问控制**[J]. 计算机研究与发展，2013，50(7)：1399-1408 Xiong Jinbo, Yao Zhiqiang, Ma Jianfeng, Li Fenghua, and Li Qi. Action-Based Multilevel Access Control for Structured Document [J]. Journal of Computer Research and Development, 2013, 50(7): 1399-1408
10	张振海，李士宁，李志刚，陈昊. **一类基于信息熵的多标签特征选择算法**[J]. 计算机研究与发展，2013，50(6)：1177-1184 Zhang Zhenhai, Li Shining, Li Zhigang, and Chen Hao. Multi-Label Feature Selection Algorithm Based on Information Entropy [J]. Journal of Computer Research and Development, 2013, 50(6): 1177-1184

数据来源:中国知网;统计日期:2014-12-18