谷国太,孙陆鹏,张红艳,等. 一种基于 GPU 的枚举排序算法及其并行化[J]. 河南理工大学学报(自然科学版),2020,39(6):139-143. doi:10.16186/j.cnki.1673-9787.2020.6.20

GU G T,SUN L P,ZHANG H Y,et al. An enumeration sorting algorithm based on GPU and it's parallelization[J]. Journal of Henan Polytechnic University(Natural Science),2020,39(6):139-143. doi:10.16186/j.cnki.1673-9787.2020.6.20

一种基于 GPU 的枚举排序算法及其并行化

谷国太[1],孙陆鹏[2],张红艳[2],肖汉[2]

(1. 河南省新闻出版学校,河南 郑州 450044;2. 郑州师范学院 信息科学与技术学院,河南 郑州 450044)

摘要: 针对枚举排序算法在处理大规模数据时存在运算量大、计算时间长、计算效率低等问题,提出一种利用 GPU 并行运算提升大规模数据处理速度的方法。在 CUDA 下对枚举排序算法进行串-并行分析,分别从细粒度与粗粒度角度进行优化,根据 CPU 与 GPU 的结构特点优化排序数据的读取和存储方式,内核采用一个 GPU 线程对应一次比较操作的计算方法,以充分利用 GPU 计算能力。实验结果表明,当排序数据规模大于 40 000 时,在 GPU 上的运算速度比在 CPU 上快 3 倍左右,并且随着数据规模的不断增大,加速比越来越大。研究结果对于提升大规模数值计算效率具有重要的意义。

关 键 词: 枚举排序;图形处理器;并行计算;数据处理;性能优化

中图分类号: TP311 **文献标志码:** A **文章编号:** 1673-9787(2020)6-139-5

An enumeration sorting algorithm based on GPU and it's parallelization

GU Guotai[1],SUN Lupeng[2],ZHANG Hongyan[2],XIAO Han[2]

(1. *Henan Press and Publishing School,Zhengzhou* 450044,*Henan,China*;2. *School of Information Science and Technology,Zhengzhou Normal University,Zhengzhou* 450044,*Henan,China*)

Abstract: In order to solve the problems of large amount of computation,long computing time and low computational efficiency when enumeration sorting is used to deal with large amounts of data,a method of using GPU parallel operation to improve the processing speed was proposed. Under the condition of CUDA,the enumeration sorting algorithm was analyzed in series-parallel,and optimized from the point of view of fine-grained and coarse-grained respectively. According to the structural characteristics of CPU and GPU,the reading and storage mode of data to be sorted was optimized. The kernel used a GPU thread corresponding to a experimental. The results showed that when the size of the dataset to be sorted was larger than 40000,the operation speed on GPU is about 3 times faster than that on CPU. And with the continuous increase of the data scale,the speedup was getting larger and larger. This study was of practical significance for large-scale numerical calculation.

Key words: enumeration sorting; graphic processing unit; parallel computing; data processing; performance optimization

---

收稿日期: 2020-01-16;**修回日期:** 2020-03-26

基金项目: 国家自然科学基金资助项目(61572444);河南省高等学校重点科研项目(16A520031)

第一作者简介: 谷国太(1971—),男,河南尉氏人,高级讲师,主要从事计算机应用方面的教学和研究工作。E-mail:ggt_32777@126.com

通讯作者简介: 肖汉(1970—),男,湖北武汉人,博士后,教授,主要从事大规模并行算法研究与设计、遥感大数据并行处理等方面的教学和研究工作。E-mail:xiaohan70@163.com

0 引 言

在数据处理的过程中,排序是其中非常重要的一个环节,而且排序在各种系统或应用软件中都有着非常广泛的应用,在处理数据序列的各种应用中也发挥着重要的作用[1-2]。枚举排序是通过把目标元素与其他元素进行比较,最终确定其在整个元素序列中的位置。随着社会信息化的快速发展,无论在科学研究、工业生产、娱乐,还是在社会民生、环境与商业等领域,数据量都呈现爆炸式增长,这就对计算速度提出了更高的要求,提高枚举排序的运算速度有着非常重要的意义[3-4]。

传统的枚举并行排序算法是在 CPU 上运行,虽然多核 CPU 可以提高运算速度,但是受到空间、电力、冷却等因素限制,多核系统目前面临着核数超过 16 个以后性能无法随内核数线性扩展以及并行软件限制等问题[5]。在基于图形处理器(graphic processing unit,GPU)的并行计算中,GPU 解决了上述多核 CPU 在运算上的不足,同时降低了成本和功耗,为解决越来越复杂的图形和数据计算问题提供了新的方案。可以预见,在今后的发展中,GPU 处理器的地位越来越重要,并将会成为提高运算速度的重要突破口[6-8]。

国内外学者对枚举算法进行了大量的研究。He G H 等[9] 提出一种软输入软输出度量优先 MIMO 检测的算法和结构。在该算法中,混合枚举策略避免了全部枚举和排序,大大降低了计算复杂度;J. Davila 等[10] 提出一种算法,该算法解决了稀疏枚举排序问题,比以往的算法性能更好;孙斌[11] 利用 MPI 实现了并行枚举算法,将数据序列分成若干个小的子序列,分派给多个处理器同时进行处理排序,之后再将每个处理器的排序结果送往主节点进行整合并完成最终排序,实现了多进程并行计算,大大降低了运算时间;S. Rajasekaran[12] 提出一种合并排序(LMM)算法,证明在超立方体上可以得到比 Nassimi 和 Sahni 的经典算法更简单的稀疏枚举排序;王勇超[13] 利用 High Performance Linpack 基准测试方法对集群系统的运算速度和性能进行了研究和测试,进而深入探讨高性能计算的枚举排序算法;H. M. Alnuweiri[14] 提出一种新的 VLSI 最优排序器设计方法,它将轮换排序和枚举排序相结合,对 N 个数进行排序,降低了算法时间复杂度;A. E. Perez 等[15] 公开了一种符号枚举排序过程的方法及使用该方法的装置,允许准确和有效地计算度量以及确定低复杂度的星座符号的最佳搜索顺序;罗明山[16] 在 OpenMP 编程模型上实现了枚举并行排序,但是,多核处理器需要内核之间进行频繁的信息交换,实现核与核之间的同步需要大量的指令处理时间,这就需要在降低单个内核最高频率的基础上增加内核的数量。核数与主频相互牵制,并不能很好地解决问题。

目前国内外利用 CUDA 进行枚举排序并行算法的研究较少,大部分研究基于 CPU 传统的 MPI 或者 OpenMP 并行计算模型,虽然可以进行并行计算,但是加速效果不理想。为此,本文将 GPU 并行计算和枚举排序算法相结合,可以大大提高算法的运算能力和排序效率,着重研究如何将枚举排序算法在主从式异构计算模型下,利用 GPU 并行完成每个数据与其他数据大小比较以及统计记录值的计算,从而验证该结构下枚举排序并行算法对于大规模数据集数据处理的有效性。

1 CUDA 体系结构

GPU 由数量众多的逻辑运算部件 ALU 构成,这些逻辑运算部件又被划分为若干个流多处理器,而每个流多处理器中又包含若干个流处理器[17]。流多处理器是 GPU 结构中执行和调度的单元,包含了各自独立的控制单元,可以独立控制指令的分发和执行,流处理器是 GPU 最基本的计算单元[18]。

CUDA 编程模型采用 CPU 与 GPU 协同工作、各司其职的工作模式,如图 1 所示。CPU 的主要任务是逻辑分析运算和串行运算,GPU 的任务主要是并行计算,目的是把并行计算任务细化和线程化。CPU 与 GPU 都有自己相应的存储空间,且各自独立。kernel 函数不是一个完整的任务,而是整个任务中可以被并行执行的步骤[19-20]。

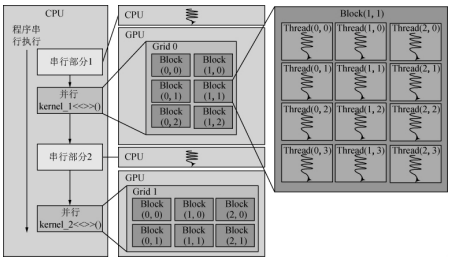

图 1 CUDA 编程模型

Fig. 1 CUDA programming model

2 枚举排序算法描述

枚举排序是一种简单排序算法。算法主要思

想为: 对每一个待排序的元素统计小于它的所有元素的个数,从而得到该元素最终处于序列中的位置。假定待排序的 n 个数存在 $a[1],\cdots,a[n]$ 中[21],首先将 $a[1]$ 与 $a[2],\cdots,a[n]$ 比较,记录比其小的数的个数,令其为 k,$a[1]$,存入有序的数组 $b[1],\cdots,b[n]$ 的 $b[k+1]$ 位置上; 其次将 $a[2]$ 与 $a[1],a[3],\cdots,a[n]$ 比较,记录比其小的数的个数,依此类推[22]。

```
算法: 枚举排序串行算法
输入: 无序数组 a[1],…,a[n]
输出: 有序数组 b[1],…,b[n]
Begin
for i = 1 to n do
(1) k = 1
(2) for j = 1 to n do
if a[i] > a[j] then
k = k + 1
end if
end for
(3) b[k] = a[i]
end for
End
```

串行的枚举排序算法描述如下。

步骤一　随机产生一个数组序列 a,包含 n 个待排序数据。

步骤二　对于待排序数组元素 $a[i]$,从 $j=1$ 开始依次比较 $a[i]$ 和 $a[j]$,并记录比 $a[i]$ 小的数的个数,记录在数组 $b[k+1]$ 位置上[23]。

步骤三　若 i 小于数组的个数 n,则循环执行上述步骤二到步骤三,直至待排序数组 a 中的数据全部排序,否则执行步骤四。

步骤四　完成排序工作[24]。

串行秩排序算法中比较操作共 $n*(n-1)$ 次,时间复杂度为 $O(n^2)$。

3　GPU 加速的枚举排序并行算法的分析与设计

3.1　枚举排序并行算法设计

在算法并行特征分析的基础上可以看出,每个待排序数的排序都是独立排序。结合 GPU 高效并行的特点,假设对一个长为 n 的输入序列使用 n 个线程进行排序,只需分配每个线程负责完成对其中一个待排序元素在有序序列中的定位和存储,使整个待排序序列的排序并行实现,接着将所有存储的信息集中到主机中即可。CUDA 枚举排序并行算法描述如下。

```
算法: 枚举排序并行算法
输入: 无序数组 a[1],…,a[n]
输出: 有序数组 a[1],…,b[n]
Begin
(1) host send a[1],…,a[n] to device
(2) for all T_i where 1≤i≤n  par-do
(2.1) k = 1
(2.2) for j = 1 to n do
if(a[i] ≥a[j]) and(i > j) then
k = k + 1
end if
end for
b[k] = a[k]
end for
(3) device send b[1],…,b[n] to host
End
```

在枚举排序并行算法中使用了 n 个线程,由于每个线程定位一个元素,所以步骤(2)的时间复杂度为 $O(1)$; 步骤(3)中设备完成的数组元素重定位操作的时间复杂度为 $O(n)$; 所以总的时间复杂度为 $O(n)$。由此可见,枚举排序并行算法时间复杂度 $O(n)<O(n^2)$ 串行算法时间复杂度。

3.2　枚举排序算法并行化方案

(1) 为待排序序列 a 和有序序列 b 分配设备存储器空间。

cudaMalloc((void**)&a,(n+5)* sizeof(double))

cudaMalloc((void**)&b,(n+5)* sizeof(double))

(2) 把主机端的数据传递到设备端。

cudaMemcpy(a,A,(n+5)* sizdof(double),cudaMemcpyHostToDevice)

(3) 定义 kernel 配置。

线程块分配的线程数:

dim3 threads(256)

网格分配的线程块数:

dim3 blocks((n+threads.x-1)/threads.x)

通过实验寻找最合适的线程分配数,如表 1 所示。综合比较可以看出,当线程为 128 或 256 时,运算速度较快。

(4) 发射 kernel 进行并行计算。

enume<<<blocks,threads>>>(a,b,n)

(5) 将已排序数据从设备端传输到主机端进行输出。

cudaMemcpy(B,b,(n+5)* sizeof(double),

cudaMemcpyDeviceToHost)

表 1 不同线程数量运行时间对比

Tab. 1 Running time comparisons of different number of threads

线程数	数据规模/个					
	100	1 000	2 000	5 000	10 000	20 000
32	0.588	0.906	1.091	1.718	2.939	7.519
64	0.571	0.829	1.508	1.698	2.819	5.408
128	0.565	0.775	1.007	1.621	2.749	5.139
256	0.556	0.816	1.030	1.581	2.698	5.097
512	0.554	0.796	1.020	1.633	2.703	5.132
1 024	0.552	0.805	1.022	1.622	2.731	5.176

4 实验与分析

4.1 实验运算平台

对枚举排序并行算法的计算效率进行测试，测试平台如下。

CPU 为 Intel Core i5，主频为 2.50 GHz，内存为 4 GB。GPU 为 NVIDIA GTX 580，GPU 型号为 GF119，属于 Fermi 构架。内置 512 个处理器核心，最高处理频率 1 544 MHz。操作系统为 Microsoft Window7 64 位，仿真实验软件为 MATLAB2013b，GPU 的应用程序编程接口为 CUDA7.0，开发环境为 Microsoft Visual Studio 2015。

4.2 实验结果和性能分析

4.2.1 实验数据

步骤一 排序数据规模为 100 ~ 40 000 时，对基于 CPU 的枚举排序串行算法的运算时间计时。

步骤二 排序数据规模为 100 ~ 40 000 时，对基于 GPU 的枚举排序并行算法的运算时间计时。

步骤三 基于 CPU 的枚举排序串行实现和基于 GPU 的枚举排序并行实现运算时间如表 2 所示。

4.2.2 加速性能分析

由图 2 可以看出，当数据规模达到 20 000 时，基于 GPU 的枚举排序相比基于 CPU 的枚举排序才呈现加速效果，当数组的数值达到 40 000 时，基于 GPU 的枚举排序时间是基于 CPU 的枚举排序时间的 1/4 左右。

在枚举并行排序的过程中，如果排序数据规模比较小，CPU 与 GPU 相比，在运算速度上有很大的优势。这是因为在 GPU 计算的过程中，数据在内存和全局存储器之间来回传输，数据交互带来的时间延迟是不能被忽略的，消耗了很多时间。

表 2 枚举排序串并行性能对比

Tab. 2 Performance comparisons of serial algorithm and parallel algorithm for enumeration sorting

数据规模/个	串行运行时间/s	并行运行时间/s	加速比
100	0.000	0.552	-
1 000	0.000	0.562	-
5 000	0.156	0.615	0.250
10 000	0.609	0.741	0.800
20 000	2.406	1.245	1.900
30 000	5.375	2.020	2.700
40 000	9.860	3.080	3.200

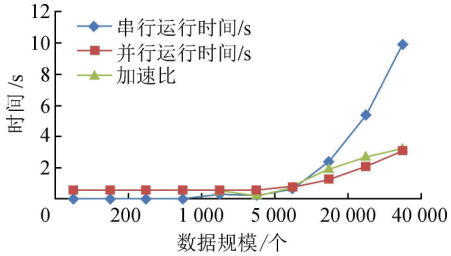

图 2 不同数据规模的枚举排序 GPU 加速效率趋势图

Fig. 2 Trend chart of GPU acceleration efficiency for enumeration sorting of different data scale

通过实验数据可以看出，如果数据规模太小，使用 CUDA 并不能带来处理速度提升。因为，对于一些数据规模非常小的系统而言，串行算法运算时间非常短，甚至能够在 ms 级内完成。但是在使用基于 CUDA 计算时，由于 GPU 系统存在启动时间问题，在 GPU 上的执行时间相对就较长，而随着数据规模的不断增大，GPU 运算时间增加的非常慢，而 CPU 运算时间则呈现迅速增长趋势，GPU 强大的浮点运算能力就显现出来了。

4.2.3 系统性能瓶颈分析

在基于 GPU 的枚举排序并行算法中需要对存储器进行读写操作。对于 n 个待排序数据，需要读取 n 次存储器，写入 n 次存储器。设待排序数据集合数据为 20 000，每个数据值分配存储空间大小是 4 字节。所以，存储器存取数据总量约为 1.6 GB。除以 kernel 实际执行时间 0.013 s，得到的带宽数值约为 123.08 GB/s，这已经接近 NVIDIA GTX 580 显示存储器的 192.38 GB/s 带宽了。因而，分析可得，基于 GPU 的枚举排序并行算法的效率受限于全局存储器带宽。

5 结 论

(1) 提出利用 GPU 并行运算提升大规模数据

处理速度的方法是可行，且效果显著。

(2) 通过枚举排序并行算法，在不同数据规模下，寻找每个线程块分配的最佳线程数，开展枚举排序并行算法分析，确定设计方案。

(3) 通过枚举排序串并行算法实验分析发现，当数据规模达到 40 000 时，并行枚举排序算法时间是串行算法运行时间的 1/4 左右，随着数据规模的不断增大，加速比会越来越大。

(4) 本文所提枚举排序并行算法是一重并行，将来研究中会继续优化并行算法，使枚举排序算法充分利用 GPU 的高效并行性。同时，目前主流的异构众核协处理器除了 GPU 以外，还有 Intel 的基于 MIC 架构众核协处理器 Xeon Phi，辅助计算的能力也非常强大。下一步计划完成基于 CPU + MIC 的枚举排序并行算法。

参考文献：

[1] AVIS D，JORDAN C. Mts: A light framework for parallelizing tree search codes [J]. Optimization Methods & Software，2019. DOI: 10.1080/10556788.2019.1692344.

[2] KARTUN-GILES A P，KIM S. Counting k-hop paths in the random connection model [J]. IEEE Transactions on Wireless Communications，2018，17(5): 3201-3210.

[3] BONZOM V. Large N limits in tensor models: towards more universality classes of colored triangulations in dimension d > =2 [J]. Symmetry Integrability and Geometry-Methods and Applications，2016. DOI: 10.3842/SIGMA.2016.073.

[4] WANG C，LI X H，TIAN T，et al. Coordinated control of passive transition from grid-connected to islanded operation for three/single-phase hybrid multimicrogrids considering speed and smoothness [J]. IEEE Transactions on Industrial Electronics，2020，67(3): 1921-1931.

[5] CLAESSON A，GUDMUNDSSON B A. Enumerating permutations sortable by k passes through a pop-stack [J]. Advances in Applied Mathematics，2019，108(6): 79-96.

[6] FUJITA，S. Standardization of mark tables and usci-cf (unit subduced cycle indices with chirality fittingness) tables derived from different o-h-skeletons [J]. Match-Communications in Mathematical and in Computer Chemistry，2019，82(2): 327-373.

[7] WANG S M，MA S，DUAN W Y. Seakeeping optimization of trimaran outrigger layout based on NSGA-II [J]. Applied Ocean Research，2018，78(9): 110-122.

[8] MENG X，BRADLEY J，YUVAZ B，et al. Mllib: machine learning in apache spark [J]. Journal Machine Learning Research，2016，17(34): 1-7.

[9] HE G H，ZHANG X Y，LIANG Z J. Algorithm and architecture of an efficient MIMO detector with cross-level parallel tree-search [J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems，2020，28(2): 467-479.

[10] DAVILA J，RAJASEKARAN S. Randomized sorting on the POPS network [J]. International Journal of Foundations of Computer Science，2005，16(1): 105-116.

[11] 孙斌. 枚举排序方法在机群系统中的并行化实现 [J]. 湖北成人教育学院学报，2006，12(4): 57-61.
SUN B. Parallel implementation of enumerated sorting method in cluster system [J]. Journal of HuBei Adult Education Institute，2006，12(4): 57-61.

[12] RAJASEKARAN S. A framework for simple sorting algorithms on parallel disk systems [J]. Theory of Computing Systems，2001，34(2): 101-114.

[13] 王勇超. 高性能计算集群技术应用研究 [D]. 西安：西安理工大学，2007.
WANG Y C. The research and realization on HPC cluster technology [D]. Xi'an: Xi'an University of Technology，2007.

[14] ALNUWEIRI H M. A new class of optimal bounded-degree VLSI sorting networks [J]. IEEE Transactions on Computers，1993，42(6): 746-752.

[15] PEREZ A E，FETTWEIS G. Method for a symbol enumeration ordering procedure and an apparatus using the method [J]. IEEE Transactions on Computers，2016(12): 1-37.

[16] 罗明山. 基于枚举的并行排序与选择算法设计 [J]. 电脑知识与技术，2015，11(12): 88-90.
LUO M S. Design of parallel sorting and selection algorithm based on enumeration [J]. Computer Knowledge and Technology，2015，11(12): 88-90.

[17] RASCH A，BIGGE J，WRODARCZYK M，et al. dOCAL: high-level distributed programming with OpenCL and CUDA [J]. Journal of Supercomputing，2020，76(7): 5117-5138.

[18] CERVERA E. GPU-accelerated vision for robots: Improving system throughput using OpenCV and CUDA [J]. IEEE Robotics & Automation Magazine，2020，27(2): 151-158.

[19] BOZKURT F，COBAN O，GUNAY F B，et al. High performance twitter sentiment analysis using CUDA based distance kernel on GPUs [J]. Tehnicki Vjesnik-Technical Gazette，2019，26(5): 1218-1227.

(下转第 158 页)

[11] 张家铭,胡明鉴,侯国强,等. 基于大型直剪试验的强风化粗粒料剪胀特性试验研究[J]. 岩土力学,2013,34(s2):67-72.
ZHANG J M,HU M J,HOU G Q,et al. Dilatant behavior of intensely weathered coarse-grained soils in large-scale direct shear tests [J]. Rock and Soil Mechanics,2013,34(s2):67-72.

[12] 董启朋,卢正,詹永祥,等. 土石混合体原位试验的颗粒流数值模拟分析[J]. 上海交通大学学报,2013,47(9):1382-1388.
DONG Q P,LU Z,ZHAN Y X,et al. Particle flow modeling of soil-rock mixture in-situ tests [J]. Journal of Shanghai Jiaotong University,2013,47(9):1382-1388.

[13] 薛亚东,刘忠强,吴坚. 崩积混合体直剪试验与PFC2D数值模拟分析[J]. 岩土力学,2014,35(s2):587-592.
XUE Y D,LIU Z Q,WU J. Direct shear tests and PFC2D numerical simulation of colluvial mixture [J]. Rock and Soil Mechanics,2014,35(s2):587-592.

[14] 赵金凤,严颖,季顺迎. 基于离散元模型的土石混合体直剪试验分析[J]. 固体力学学报,2014,35(2):124-133.
ZHAO J F,YAN Y,JI S Y. Analysis of direct shear test of soil-rock mixture based on discrete element model [J]. Chinese Journal of Solid Mechanics,2014,35(2):124-133.

[15] 李振,邢义川. 干密度和细粒含量对砂卵石及碎石抗剪强度的影响[J]. 岩土力学,2006,27(12):2255-2260.
LI Z,XING Y C. Effects of dry density and percent fines on shearing strength of sandy cobble and broken stone [J]. Rock and Soil Mechanics,2006,27(12):2255-2260.

[16] 刘建锋,徐进,高春玉,等. 土石混合料干密度和粒度的强度效应研究[J]. 岩石力学与工程学报,2007,26(s1):3304-3310.
LIU J F,XU J,GAO C Y,et al. Study on strength effects of dry density and granularity on earth and rock mixture [J]. Chinese Journal of Rock Mechanics and Engineering,2007,26(s1):3304-3310.

[17] 梁维,李克钢,侯克鹏,等. 散体云母片岩粒度分形特征及其与抗剪强度参数的关系[J]. 岩土力学,2012,33(3):767-772.
LIANG W,LI K G,HOU K P,et al. Fractal characteristics of size distribution of loose micaschist and its relation to shear strength parameters [J]. Rock and Soil Mechanics,2012,33(3):767-772.

[18] 梁维. 曼家寨排土场散体岩石物料力学特性研究[D]. 昆明:昆明理工大学,2011.
LIANG W. Study on mechanical properties rockfill material in Manjiazhai waste dump [D]. Kunming: Kunming University of Science and Technology,2011.

[19] 杜俊. 松散岩土体物理力学性质及其工程应用研究[D]. 昆明:昆明理工大学,2010.
DU J. Study on mechanical properties and engineering applications of rockfill material [D]. Kunming: Kunming University of Science and Technology,2010.

[20] 杨忠平,李万坤,胡元鑫,等. 压实系数对粗粒土剪切特性的影响[J]. 地下空间与工程学报,2017,13(2):348-356.
YANG Z P,LI W K,HU Y X,et al. Effect of Compaction Coefficient on Shear Characteristics of Coarse Grained Soil [J]. Chinese Journal of Underground Space and Engineering,2017,13(2):348-356.

[21] MATSUOKA H. A microscopic study on the shear mechanism of granular materials [J]. Soils and Foundations,1974,14(1):29-43.

(责任编辑 郭倩倩)

(上接第 143 页)

[20] DENG Y Y,ZHAO X Y,LI T Z,et al. CUDA-based volume rendering and inspection for time-varying ultrasonic testing datasets [J]. Computing in Science & Engineering,2019,21(5):76-86.

[21] LEE C C,HERVE J M. Type synthesis of primitive schoenflies-motion generators [J]. Mechanism and Machine Theory,2009,44(10):1980-1997.

[22] CERBAI G,CLAESSON A,FERRARI L. Stack sorting with restricted stacks [J]. Journal of Combinatorial Theory Series A,2020. DOI: 10.1016/j.jcta.2020.105230.

[23] OTA T,MORITA H,MANADA A. Compression by substring enumeration using sorted contingency tables [J]. IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences,2020,E103A(6):829-835.

[24] DAMASCHKE P. The solution space of sorting with recurring comparison faults [J]. Theory of Computing Systems,2018,62(6):1427-1442.

(责任编辑 李文清)