DOI:10.7652/xjtuxb201411017

多核处理器中基于 MapReduce 的哈希划分优化

袁通，刘志镜，刘慧，王梓
（西安电子科技大学计算机学院，710071，西安）

摘要：针对传统的并行哈希划分算法不能高效地利用多核处理器的并行资源，且不能较好处理有倾斜的输入数据的问题，提出了一种在多核处理器中基于 MapReduce 的哈希划分算法，并且提出了存储结构优化、多步划分优化、数据倾斜优化 3 种优化策略。该算法将输入数据分成若干块后提交给各个线程并行处理，并选择合适的策略避免写冲突，使其能够高效地利用多核处理器的并行资源。文中提出的哈希表能够提高 cache 效率，从而提升算法的整体性能。引入 MapReduce 模型可使多步哈希划分在 Map 过程和 Reduce 过程中分别进行；数据倾斜优化策略能使算法适应有倾斜的输入数据，且具有较好的效果。实验结果表明：在多核处理器中，文中提出的算法能够适应各种分布的输入数据，并且使哈希划分的整体性能得到提升。

关键词：数据划分；哈希处理；多核处理器；MapReduce 模型
中图分类号：TP392 **文献标志码**：A **文章编号**：0253-987X(2014)11-0097-06

Hash Partitioning Optimizations Based on MapReduce for Chip Multiprocessors

YUAN Tong, LIU Zhijing, LIU Hui, WANG Zi
(School of Computer Science and Technology, Xidian University, Xi'an 710071, China)

Abstract: A hash partitioning method based on MapReduce framework and three efficient optimizations including storage structure optimization, multi-pass partitioning optimization and skew data optimization on chip multiprocessor (CMP) are proposed to address the problems that conventional hash partitioning method cannot take full advantage of CMP's parallel execution resources and properly process the skew input data. The input data are split into several units which are later processed by all threads simultaneously, and suitable strategy is adopted to avoid writing collision hence CMP's parallel execution resources could be fully unitized. The new hash table proposed in this paper can improve the overall performance by increasing the cache efficiency. The introduction of MapReduce framework makes it possible to multiply partition the data in Map phase and Reduce phase, respectively. In addition, the skew data optimization can make the proposed method suitable for processing various skew input data. Experiments have testified these advantages displayed by the proposed hash partitioning method.

Keywords: data partitioning; hashing; multicore processors; MapReduce framework

划分是数据库中的重要操作，同时也是其他数据库操作（如连接、聚集、排序等）的基本操作。划分是将一个较大的任务分成若干个较小的子任务，而处理若干个子任务所用的时间常常少于处理一个较

收稿日期：2014-04-09。 作者简介：袁通（1987—），男，博士生；刘志镜（通信作者），男，教授，博士生导师。 基金项目：国家科技支撑计划资助项目（2012BAH01B05）；陕西省科技统筹创新工程计划资助项目（2012KTZD-02-05-2）
网络出版时间：2014-09-02 网络出版地址：http://www.cnki.net/kcms/detail/61.1069.T.20140909.0908.004.html

大任务所用的时间，这是因为较小的任务能够高效地利用 cache 和内存。哈希划分是使用范围最广的划分算法。

当今的硬件发展十分迅速，CPU 拥有更多的核心，每个核心拥有更多的线程。最近，IBM 推出了新一代的 POWER 8 处理器，支持 12 核心 96 线程，共享 96 MB 的三级缓存，这说明多核 CPU 具有广阔的应用前景。面对新型的硬件架构，传统的并行哈希划分算法存在以下不足：①不能高效地利用多核处理器的并行资源，包括不能有效地降低内存存取延迟、减少 cache 丢失和 TLB 丢失等；②存储结构不能高效地支持多核并行读写；③不能较好地处理有倾斜的输入数据。

针对这些问题，本文提出一种多核处理器中基于 MapReduce 的哈希划分优化方法，用基于共享内存的 MapReduce 模型实现了哈希划分算法，并比较了 4 种避免线程冲突的策略。MapReduce 模型的使用可以使算法高效地利用多核并行资源。在此基础上，提出了存储结构优化、多步划分优化、数据倾斜优化 3 种有效的优化方法，以解决现有算法的不足。

1 相关工作

对于在不同应用中的划分操作，人们已经进行了大量的研究，这些研究主要是针对数据库操作的。在连接操作[1]和聚集操作[2]中，划分能够明显地提升操作性能。Balkesen 等人的研究表明，在整个连接操作中划分操作占用了大部分时间[3]。Manegold 等人证明了当划分数量较大时，多步划分比单步划分效果要好，且多步划分中第一步的划分数量等于第二步的划分数量时效果最好[4]。这是因为多步划分避免了大量的 cache 丢失和 TLB 丢失，且内存压力较小。但是，他们提出的 Radix-cluster 划分并不支持多核并行处理。Cieslewicz 等人提出了在多核处理器中并行划分的方法[5]，但局限性是存储结构事先需知道每个划分中的元组数量，且其算法只在处理均匀分布的输入数据时有较好的效果，而多核负载均衡性并不理想。Lisa 等人提出用硬件加速器来提升划分性能[6]，但这种方法受限于硬件系统。

MapReduce[7] 自 2004 年问世以来，已经证明其在处理海量数据时有巨大的优势。MapReduce 最初是针对机群提出的，其性能经常受硬盘 IO 和网络 IO 的制约，而基于共享内存的 MapReduce[8] 则避免了这些瓶颈，适合处理内存数据。所以，数据库管理系统（DBMS）可以利用基于共享内存的 MapReduce 来优化划分算法。基于共享内存的 MapReduce 模型适用于多核处理器，处理器的每一个线程被当作一个计算节点，计算节点之间的通信是以共享内存的方式完成的。

2 基于 MapReduce 的哈希划分

哈希划分可以通过一个 MapReduce 任务来实现，该 MapReduce 任务只实现 Map 任务即可。在 Map 任务中，根据键值对中键值的哈希值将该键值对写入结果区域。由于多线程并行地将键值对写入结果区域，所以可能会产生线程之间的冲突。为了避免写冲突，常采用以下 4 种策略。

（1）加锁策略。所有线程共享一个键值对存储结构，每一个划分区域是一个连续的存储空间；线程需要加锁后才能将键值对写入相应的划分区域，写入完成后需要解锁以便其他线程继续写入该区域。该策略的优点是内存消耗较小，且不会随着线程数量的增加而增加；缺点是频繁的加锁、解锁操作影响性能。

（2）无锁策略。每个线程有一个独立的键值对存储结构，由于每个线程只将数据写入自己的存储结构中，所以避免了频繁的加锁解锁操作。该策略的优点是避免了加锁解锁操作；缺点是需要额外的操作将所有存储结构进行合并，且内存消耗随着线程数量的增加而增加。

（3）两次遍历策略。该策略的思想来源于文献[9]，即每个线程 2 次遍历分配给它的输入键值对。如图 1 所示，在一个用 4 线程将数据哈希划分为 4 份的任务中，第一次遍历时每个线程计算出属于不同划分的键值对的数量，根据这些数据可以计算出每个线程将不同的键值对写入存储结构的位置

$$R[i]=\sum_{j=0}^{i}K[j] \tag{1}$$

式中：$R[i]$为划分 i 在最终存储结构中的起始位置；$K[j]$为划分 j 中键值对的数量；$i,j=t+pn$，其中 t 为线程编号（本例中 $t=0,1,2,3$），p 为划分编号（本例中 $p=0,1,2,3$），n 为线程总数（本例中 $n=4$）。

在第二次遍历时，每个线程根据上一步的计算结果将键值对并行写入中间键存储结构中。该策略的优点是将最终的划分结果连续存储，提高了程序的局部性，而缺点是需要 2 次遍历输入数据。

（4）并行缓存策略[10]。该策略类似于加锁策略，

不同之处是在加锁策略中每写入一个中间键值对都需要加锁解锁，而在并行缓存策略中每个线程有大小一定的独立存储空间，将键值对写入独立存储空间时不需要进行加锁解锁操作，但当该存储空间耗尽时，需要通过加锁解锁操作获得新的存储空间。

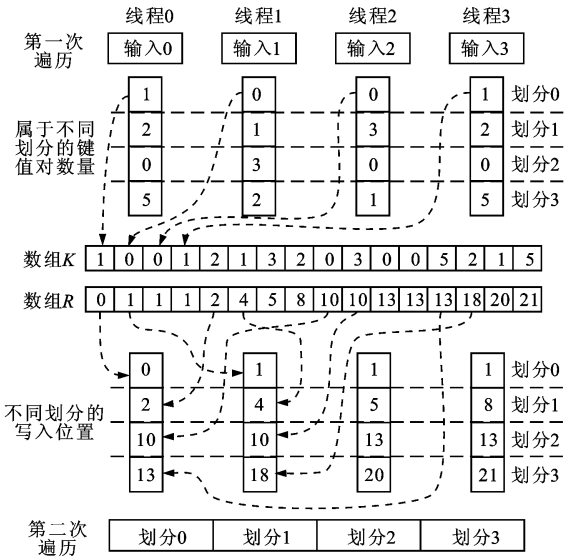

图 1　两次遍历策略

3　哈希划分优化方法

在多核处理器中每个线程被看作一个计算节点，利用 MapReduce 思想可以优化哈希划分。本节主要从存储结构优化、多步划分优化和数据倾斜优化这 3 个方面来优化哈希划分。

3.1　存储结构优化

哈希表的结构直接影响整个算法的效果。传统的哈希表用一个 vector 容器或一个数组来存储某一划分中的键值对，如图 2 所示。

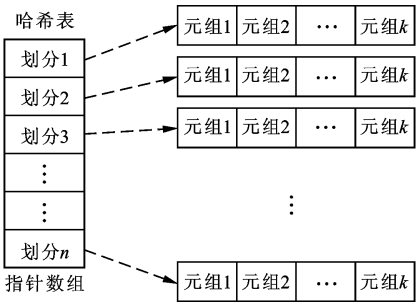

图 2　传统的哈希表结构

如果用一个 vector 容器来存储某一个划分中的键值对，当已存元组的数量很大时，元组的存储效率会明显降低。如果用一个数组存储某一个划分中的键值对，元组的存储效率较高且效率不会随着存储元组数量的增加而降低，但初始化一个容量较大的数组所需的开销却比较可观。

为此，在加锁策略和无锁策略中，为了提高 cache 的效率，我们采用一种优化的哈希表存储结构。不同于传统的哈希表，优化的哈希表用一个连续的数组表示，数组的每一位表示一个哈希桶，每一个哈希桶集中存储某一个划分中的键值对，如图 3 所示。每一个哈希桶由 2 个指针和一段连续的存储空间组成，连续的存储空间存储属于该划分的元组，free 指针指向该连续空间中下一个空闲元组的位置，当此哈希桶溢出时，next 指针指向另外的哈希桶。此外，每一个哈希桶的大小等于 CPU 中 cache line 的大小，这样在存取过程中可以避免大量的 cache 丢失。这样的设计既保证了元组存取效率，又降低了初始化的开销。

图 3　改进的哈希表结构

3.2　多步哈希划分优化

可以利用基于共享内存的 MapReduce 模型来优化多步哈希划分，用 Map 过程进行第一次划分，用 Reduce 过程进行第二次划分。

图 4 表示了利用基于共享内存的 MapReduce 模型优化两步哈希划分的流程，图中 p_1 表示第一次划分的数量，p_2 表示第二次划分的数量，所以最终产生 $p_1 p_2$ 个划分结果。

首先将输入数据分割成若干块，每一块数据交给一个 Map 线程进行第一次哈希划分，产生 p_1 个划分结果。Map 线程之间的写冲突问题可以用第 2 节中的 4 种策略之一解决。由于首先将输入数据分割成大小相同的块，所以第一次划分中各个线程处理的数据量大致相同，有良好的负载均衡。

接下来，p_1 个划分结果产生 p_1 个 Reduce 任务，每一个 Reduce 任务交给一个 Reduce 线程处理，进行第二次哈希划分，得到最终 $p_1 p_2$ 个划分结果。这里需要注意 2 个问题：①p_1 个 Reduce 任务的大小可能不一致，这将导致某些 Reduce 线程处理较多的数据，出现负载不均衡问题，从而导致总体处理时间的增加；②由于最终 $p_1 p_2$ 个划分结果中的某一个划分只可能来自第一次哈希划分的 p_1 个划分结果中的某一个划分，所以在第二次哈希划分中，

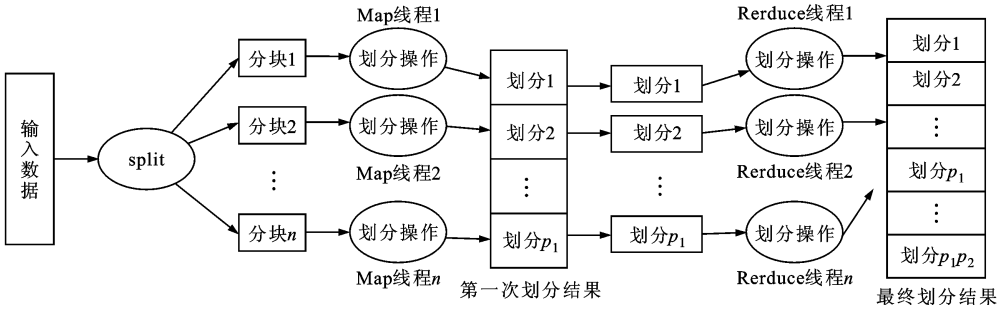

图 4 利用基于共享内存的 MapReduce 模型优化两步哈希划分

Reduce 线程之间不会出现写冲突。

3.3 数据倾斜优化

在多步哈希划分中，由于第一次划分产生的 Reduce 任务大小可能不一致，所以可能造成 Reduce 过程中的负载不均衡，导致总体处理时间增加。为此，提出以下优化方法，具体步骤如下。

(1)设定一个阈值 T，用于比较第一次哈希划分的 Reduce 任务的大小。

(2)Reduce 线程在处理 Reduce 任务之前，先比较 Reduce 任务的大小和阈值 T，如果 Reduce 任务小于 T，则 Reduce 线程处理该任务；如果 Reduce 任务大于 T，则将该任务加入队列 D 中，并不进行处理。至此，小于 T 的任务都已被处理，而大于 T 的任务都没有被处理，且都保存在 D 中。

(3)将队列 D 中的每一个任务平均分为 n 份(n 为 Reduce 线程的数量)，将 n 份任务交给 n 个 Reduce 线程并行处理。

(4)将队列 D 中的每一个任务处理完后，得到最终的划分结果。

上述优化方法利用多核运算能力解决了多步哈希划分中 Reduce 任务负载不均衡的问题。通常将阈值 T 设定为 $2C/p_1$，其中 C 表示最初输入数据的大小，p_1 表示第一次划分产生的划分结果的数量，也就是 Reduce 任务的数量。

4 实验与分析

4.1 实验环境

利用 C++语言在 Linux 系统中实现了基于 MapReduce 的哈希划分及其优化方法。本文的实验环境基于新型英特尔 Sandy Bridge 架构的 Xeon 8 核处理器(E5-2670 2.6 GHz)，每核包含 2 个线程。具体配置如下：核数为 8；线程数为 16；一级缓存大小为每核 32 KB；二级缓存大小为每核 256 KB；三级缓存大小为共享 20 MB；内存大小为 4×8 GB DDR3 内存(1 600 Hz)；cache line 大小为 64 B；快表(TLB)大小为每核 64 个。

本实验采用了 2 种数据集：一种是均匀分布的数据集(用 A 表示)；另一种是有数据倾斜的数据集(用 B 表示)。2 种数据集均含有 16×2^{20} 条元组，其中每条元组的大小为 16 B，含有 8 B 的编号和 8 B 的划分值。这种元组结构常应用在列存储数据库之中。对于有数据倾斜的数据集，采用 Zipf 指数来衡量倾斜程度。详细的数据集属性见表 1。

表 1 实验数据集属性

属性参数	数据集 A	数据集 B
键值对大小/B	8-8	8-8
元组数量	16×2^{20}	16×2^{20}
Zipf 指数	1	1.15

4.2 单步划分实验与分析

图 5 给出了 4 种不同写策略下单步哈希划分的实验结果。实验使用的是数据集 A，且采用了前述的优化方法。在加锁策略中，当 Hash Bits 较小时，每一个划分结果有较多的元组，频繁的加锁解锁操作会影响整体性能。随着 Hash Bits 的增加，每一个划分结果的元组数量减少，线程之间的冲突减少，整体性能提升。随着 Hash Bits 继续增加，cache 丢失和 TLB 丢失会影响程序的性能。在无锁策略中，因为没有加锁解锁操作，在 Hash Bits 较小时程序性能大大优于加锁策略的程序性能。由于程序需要许多额外的变量记录当前写入位置、划分大小等信息，而这些变量的数量随着线程数量的增加而增加，所以随着 Hash Bits 的增加，无锁策略承担的内存压力增加。再考虑到 cache 丢失和 TLB 丢失的影响，随着 Hash Bits 的增加，程序整体性能下降较为明显。这些分析同样适用于两次遍历策略和并行缓存策略。所以，当 Hash Bits 较大时，加锁策略优于其他 3 种策略，这说明 Hash Bits 较大时 cache 丢失和 TLB 丢失的影响大于加锁解锁操作的影响。需要说明的是，在两次遍历策略中程序的整体性能主

要受限于计算写入位置的操作。

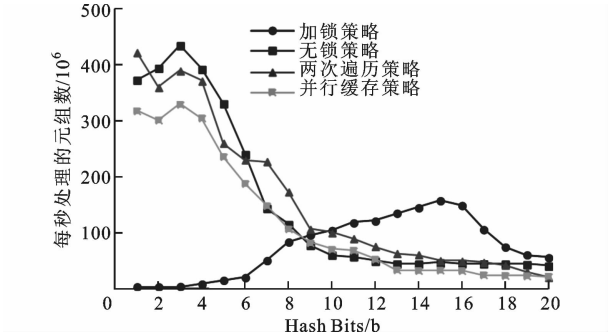

图 5　4 种不同写策略下的单步哈希划分实验结果

图 6 给出了本文算法与传统哈希划分算法的实验结果比较，图中传统哈希划分算法的结果来自文献[5]。实验使用的是数据集 A，在加锁策略和无锁策略下进行单步哈希划分。实验结果表明，由于采用了 MapReduce 模型和本文提出的优化方法，所以本文算法比传统哈希划分算法的效果更好。

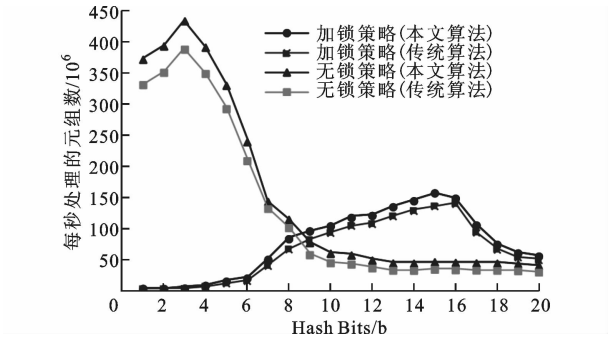

图 6　本文算法与传统哈希划分算法的比较

4.3　多步划分实验与分析

图 7 和图 8 分别给出了两次遍历策略下和无锁策略下单步和两步哈希划分的实验结果。该实验使用的是数据集 A，且在两步划分中第一次划分和第二次划分的数量分别为 $2^{\lceil b/2 \rceil}$ 和 $2^{\lfloor b/2 \rfloor}$。文献[4]表明，由于 cache 丢失和 TLB 丢失的原因，对于较大的划分数来说，多步划分比单步划分效果要好。实验结果表明，利用基于共享内存 MapReduce 模型的两步哈希划分的性能比其他两步哈希划分的性能[4-5]有明显提高，且两次遍历策略在 Hash Bits 较大时优于无锁遍历策略。这得益于 MapReduce 模型的高效性和良好的负载均衡性，且两次遍历策略中第一次划分的结果连续存放，读写时可以充分利用空间局部性原理。此外，利用 MapReduce 模型时仅第一次哈希划分存在线程写冲突问题，第二次哈希划分则不存在线程写冲突问题。

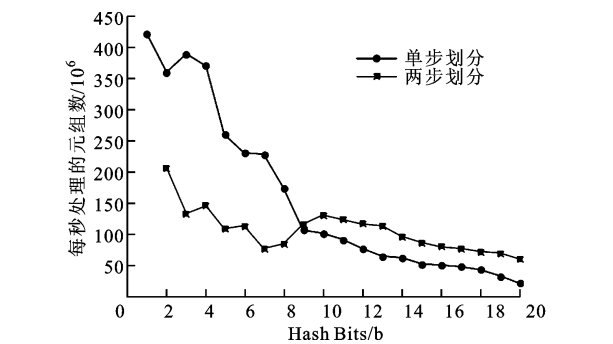

图 7　两次遍历策略下单步和两步划分的实验结果

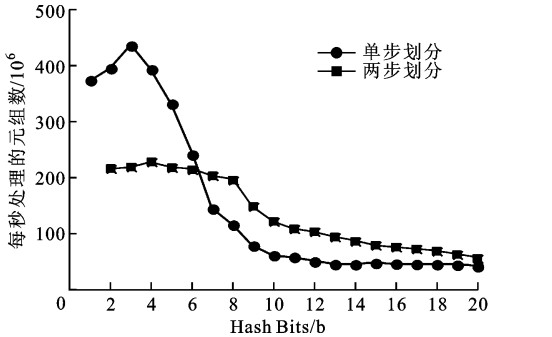

图 8　无锁策略下的单步和两步哈希划分实验结果

4.4　数据倾斜实验与分析

图 9 给出了两次遍历策略下使用数据倾斜优化和未使用数据倾斜优化的哈希划分实验结果，该实验使用的是数据集 B。实验结果表明，在多步划分处理有倾斜的输入数据时，使用本文提出的优化方法可明显提高划分性能。这是因为本文提出的优化方法避免了多个空闲线程等待一个工作线程的情况，因此在处理有倾斜的输入数据时可以有效提高整体划分性能。

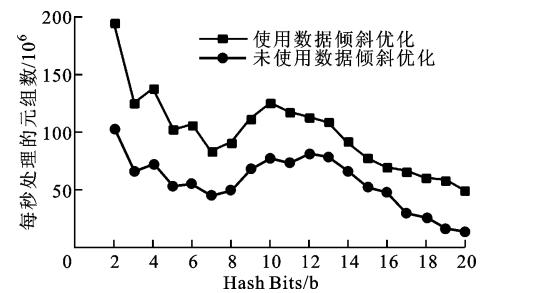

图 9　使用和未使用数据倾斜优化的两步划分实验结果

5　结　语

本文提出了一种多核处理器中基于 MapReduce 的哈希划分方法及其优化策略。结合基于共享内存的 MapReduce 模型和 4 种避免写冲突的策略，本文从存储结构、多步划分以及数据倾斜 3 个方面优化了多核并行哈希划分。实验结果表明，本文提出的方法能够适应多核处理器中各种分布的输入数据，并且使整体划分性能得到提升。

参考文献：

[1] 邓亚丹，景宁，熊伟. 基于共享 Cache 多核处理器的 Hash 连接优化 [J]. 软件学报，2010，21(6)：1220-1232.
DENG Yadan，JING Ning，XIONG Wei. Hash join query optimization based on shared-cache chip multiprocessor [J]. Journal of Software，2010，21(6)：1220-1232.

[2] YE Y，ROSS K，VESDAPUNT N. Scalable aggregation on multicore processors [C]// Proceedings of the Seventh International Workshop on Data Management on New Hardware. New York，USA：ACM，2011：1-9.

[3] BALKESEN C，TEUBNER J，ALONSO G，et al. Main-memory hash join on multi-core CPUs：tuning to the underlying hardware [C]// Proceedings of the 29th International Conference on Data Engineering. Piscataway，NJ，USA：IEEE，2013：362-373.

[4] MANEGOLD S，BONCZ P，KERSTEN M. Optimizing main-memory join on modern hardware [J]. IEEE Transactions on Knowledge and Data Engineering，2002，14(4)：709-730.

[5] CIESLEWICZ J，ROSS K. Data partitioning on chip multiprocessors [C]// Proceedings of the Fourth International Workshop on Data Management on New Hardware. New York，USA：ACM，2008：25-34.

[6] WU L，BARKER R，KIM M，et al. Navigating big data with high-throughput，energy-efficient data partitioning [C]// Proceedings of the 40th Annual International Symposium on Computer Architecture. New York，USA：ACM，2013：249-260.

[7] DEAN J，GHEMAWAT S. MapReduce：simplified data processing on large clusters [C]// Proceedings of the Sixth Symposium on Operating Systems Design and Implementation. Berkeley，USA：USENIX，2004：137-150.

[8] TALBOT J，YOO R，KOZYRAKIS C. Phoenix++：modular MapReduce for shared-memory systems [C]// Proceedings of the Second International Workshop on MapReduce and Its Applications. New York，USA：ACM，2011：9-16.

[9] FANG Wenbin，HE Bingsheng，LUO Qiong，et al. Mars：accelerating MapReduce with graphics processors [J]. IEEE Transactions on Parallel and Distributed Systems，2011，22(4)：608-620.

[10] CIESLEWICZ J，ROSS K，GIANNAKAKIS I. Parallel buffers for chip multiprocessors [C]// Proceedings of the Third International Workshop on Data Management on New Hardware. New York，USA：ACM，2007：1-10.

（编辑 葛赵青）

（上接第 57 页）

CHANG Shuping，WANG Yongsheng，WEI Yingsan，et al. Pressure fluctuation of unsteady flow in waterjet [J]. Journal of Jiangsu University：Natural Science Edition，2012，33(5)：522-527.

[9] 靳栓宝，王永生，常书平，等. 混流泵内流场压力脉动特性研究 [J]. 农业机械学报，2013，44(3)：64-68.
JIN Shuanbao，WANG Yongsheng，CHANG Shuping，et al. Pressure fluctuation of interior flow in mixed-flow pump [J]. Transactions of the Chinese Society for Agricultural Machinery，2013，44(3)：64-68.

[10] 靳栓宝，王永生，杨琼方. 基于数值试验的喷水推进轴流泵的一体化设计 [J]. 中国造船，2010，51(3)：39-45.
JIN Shuanbao，WANG Yongsheng，YANG Qiongfang. Integration design of water jet axial flow pump based on numerical experimentation [J]. Shipbuilding of China，2010，51(3)：39-45.

[11] 刘学强，伍贻兆，程克用. 用基于 M-SST 模型的 DES 数值模拟喷流流场 [J]. 力学学报，2004，36(4)：401-406.
LIU Xueqiang，WU Yizhao，CHENG Keyong. Computation of lateral turbulent jets using M-SST DES model [J]. Acta Mechanica Sinica，2004，36(4)：401-406.

[12] 魏应三，王永生. 喷水推进器进水流道不均匀度统一描述 [J]. 武汉理工大学学报，2009，31(8)：159-163.
WEI Yingsan，WANG Yongsheng. Unified description of out flow non-uniformity of waterjet duct [J]. Journal of Wuhan University of Technology，2009，31(8)：159-163.

[13] 黎义斌，李仁年，王秀勇，等. 低比转数混流泵压力脉动特性的数值模拟 [J]. 排灌机械工程学报，2013，31(3)：205-209.
LI Yibin，LI Rennian，WANG Xiuyong，et al. Numerical analysis of pressure fluctuation in low specific speed mixed-flow pump [J]. Journal of Drainage and Irrigation Machinery Engineering，2013，31(3)：205-209.

（编辑 苗凌）