Register-Aware Optimizations for Parallel Spare Matrix-Matrix Multiplication

第六组：隋东方 王浩 王丹宁
指导老师：龚春叶、甘新标、杨博

2019/4/15

目录

01 Introduction

SpGEMM（General sparse matrix-matrix multiplication ）一般稀疏矩阵矩阵乘法

应用场景：代数多重网格方法、最短路径算法、广度优先搜索算法、马尔可夫聚类算法等

C=A·B　　c=a·B（向量矩阵乘）

存储：片上共享内存、全局内存、线程寄存器

基于寄存器感知的SpGEMM算法：

1.Reg-sort

2.Reg-Merge

3.Reg-Hash

02 Backgoround

2.1 Spare Matrix

存储格式CSR(Compressed Spare Row)

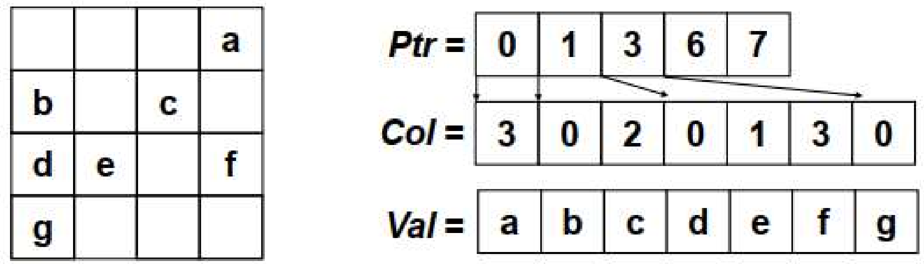


Fig. 1: A sparse matrix and its CSR format.

Ptr：记录每一行非零元个数的偏移量　　矩阵第i行非零元个数：Ptr[i+1]-Ptr[i]
Col：记录非零元的列索引号
Val：存储相应非零元的值

02 Backgoround

2.2 SpGEMM and Spare Accumulator

Matrix C=A·B　　　　vector c=a·B

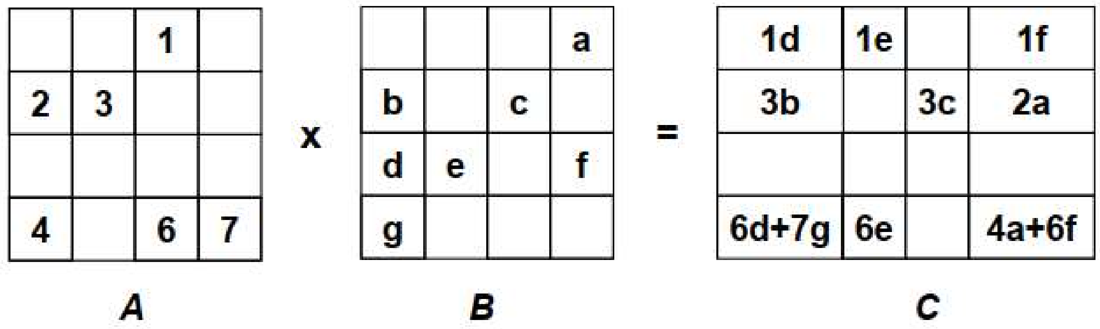


Fig. 2: An illustration of the SpGEMM operation.

以A最后一行为例：
a30=4，
B矩阵的第0行b0={a}，c33=4a

a32=6，
B矩阵的第2行b2={d，e，f},
得到c30=6d，c31=6e，c33=6f

a33=7，B矩阵的第3行b3={g}，
c30=7g

累加得：c30=6d+7g，c31=6e，
c33=4a+6f

02 Backgoround

2.3 three implementations of Spare Accumulator

c33=4a   c30=6d，c31=6e，c33=6f   c30=7g

2.3.1Sort-based

{4a}，{6d,6e,6f}，{7g}

根据列索引号排序：{6d，7g，6e，4a，6f}

索引号相同的累加：{6d+7g，6e，4a+6f}

2.3.2merge-based

{4a}，{6d,6e,6f}，{7g}

寻找最小的列索引，迭代合并{6d+7g}，{6e}，{4a+6f}

2.3.3hash-based

利用hash函数快速定位位置，然后对相同列索引号的值加和

03 Methodology

3.1 Reg-sort： Register-Aware Sort-based Spare Accumulator

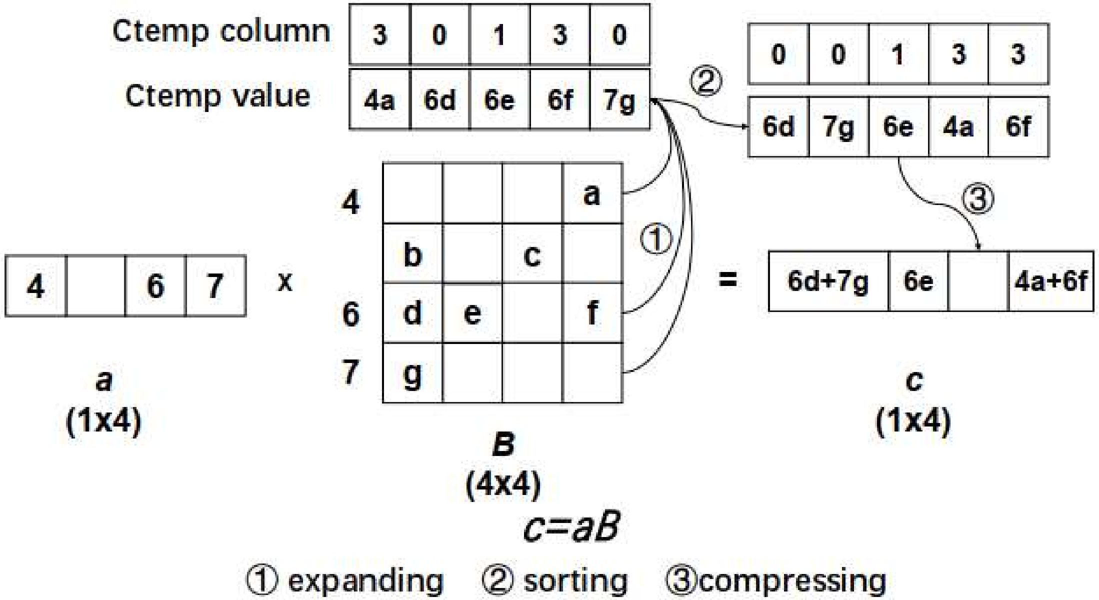


Fig. 3: Original sort implementation.

03 Methodology

3.1 Reg-sort： Register-Aware Sort-based Spare Accumulator

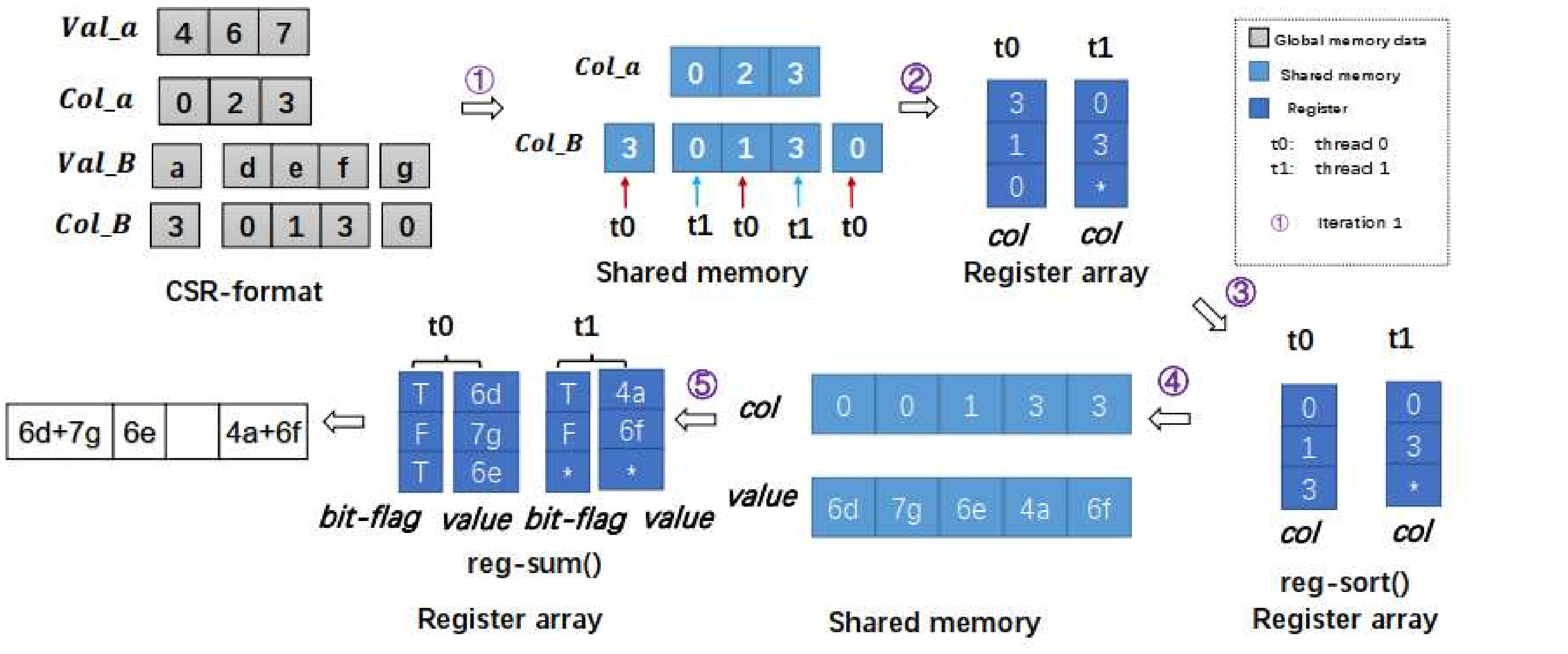


Fig. 4: Our reg-sort method.

03 Methodology

3.2 Reg-merge：Register-Aware Merge-based Spare Accumulator

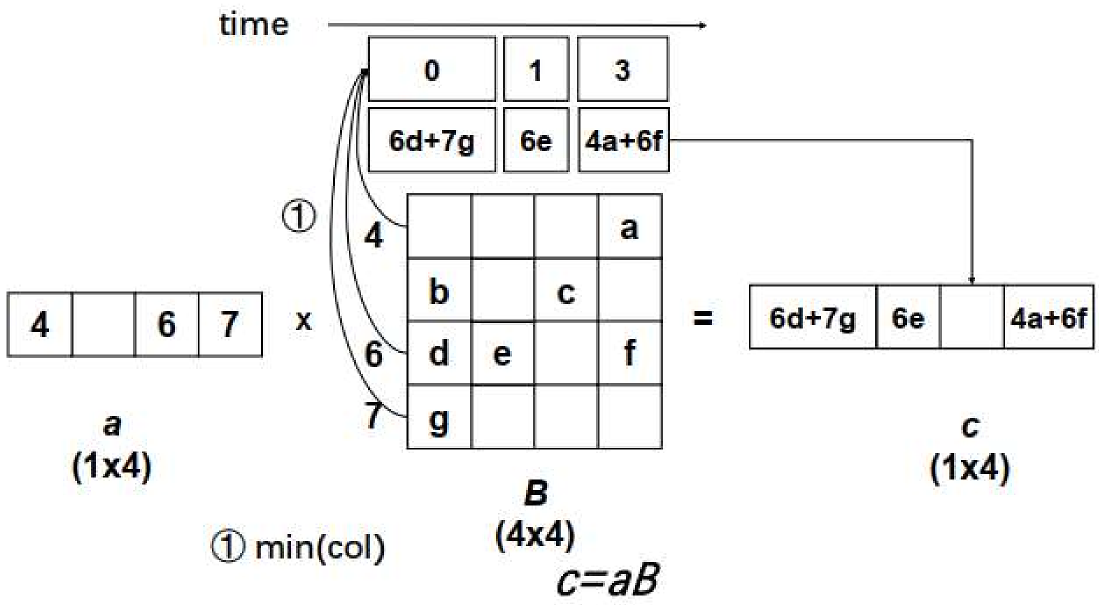


Fig. 5: Original merge implementation.

基于合并的原始实现方法。
该方法有两个阶段：
第一个阶段：将矩阵A划分成小的子矩阵。若矩阵最大行的长度小于warp的大小32，该行就在共享内存中执行。

第二阶段：使用一个warp将子矩阵乘以第二个矩阵B。

03 Methodology

3.1 Reg-merge：Register-Aware Merge-based Spare Accumulator

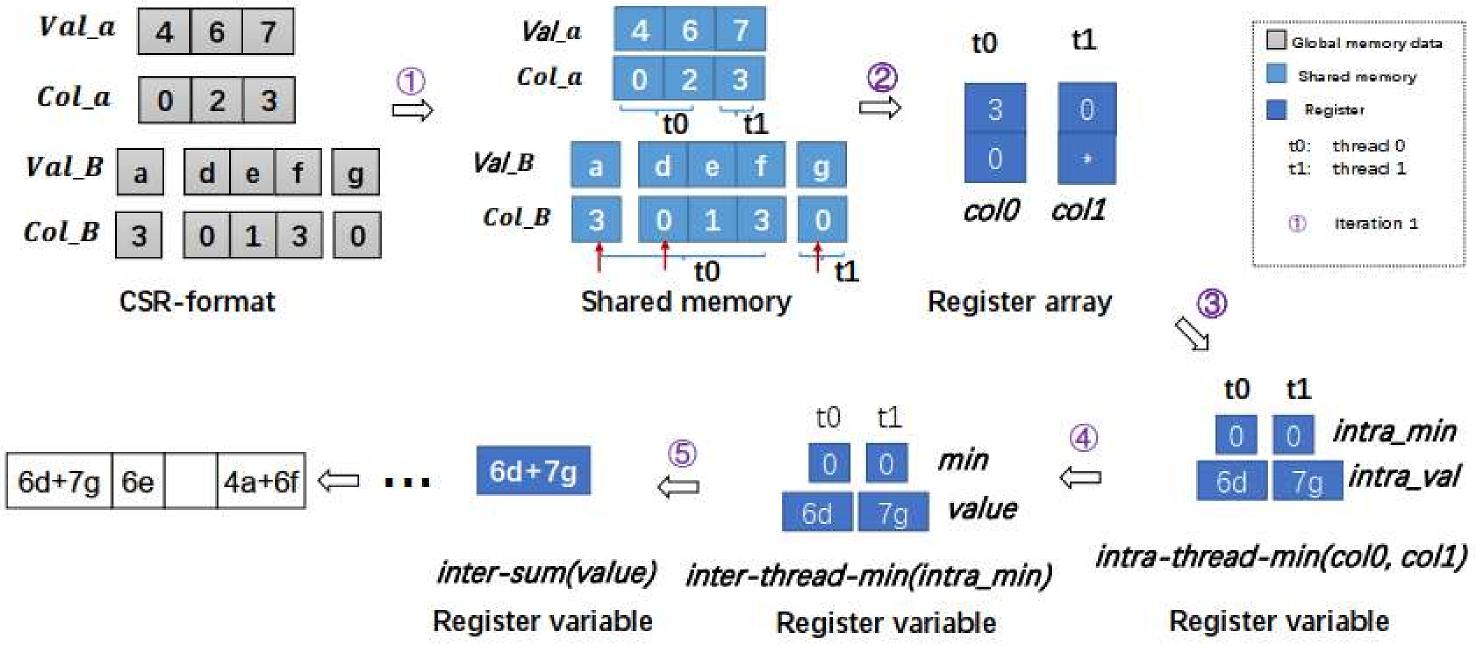


Fig. 6: Our reg-merge method.

03 Methodology

3.2 Reg-hash： Register-Aware hash-based Spare Accumulator

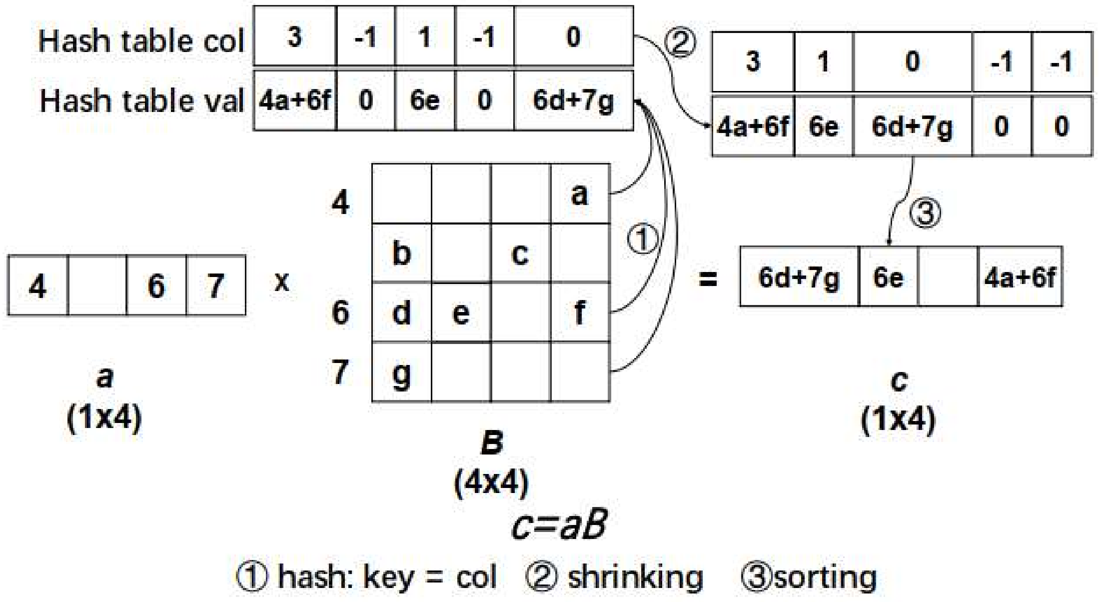


Fig. 7: Original hash implementation.

基于hash的原始实现方法。该方法有3个阶段：

1.Hash操作

2.Hash表中有效值排在表头

3.根据列索引排序

03 Methodology

3.1 Reg-hash： Register-Aware hash-based Spare Accumulator

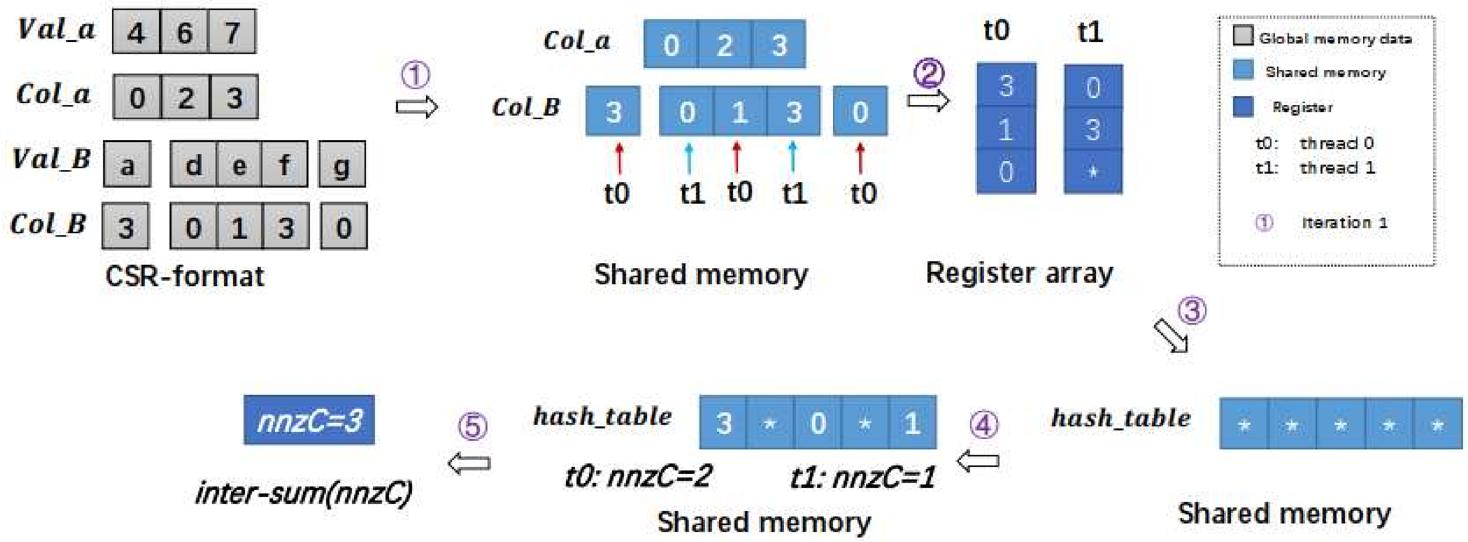


Fig. 8: Our reg-hash method.

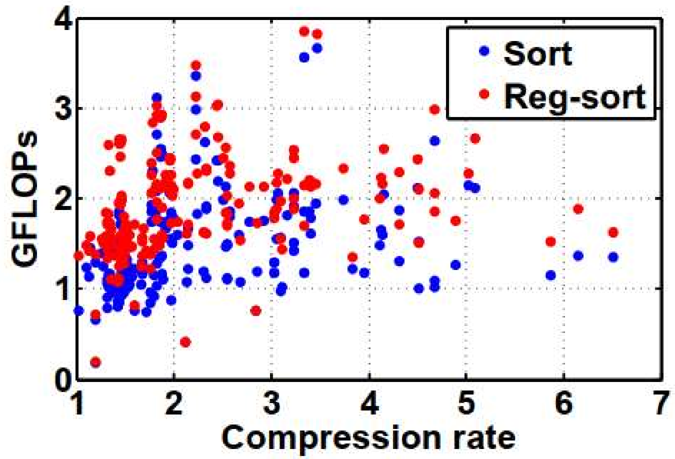
04 Evaluation & Conclusion

实现平台：Nvidia Pascal P100 GPU　　IntelXeon server
编译环境：CUDA V8.0　　Intel C/C++ v18
Benchmark：SuiteSparse Matrix Collection　　956个稀疏矩阵

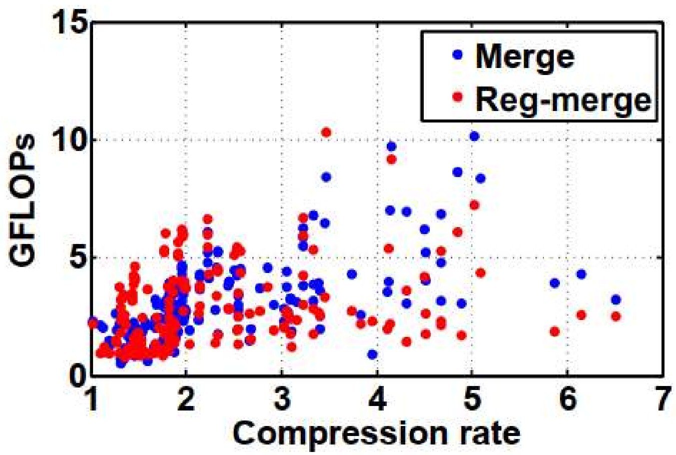
算法比较：bhSPARSE , RMerge and NSPARSE

04 Evaluation & Conclusion

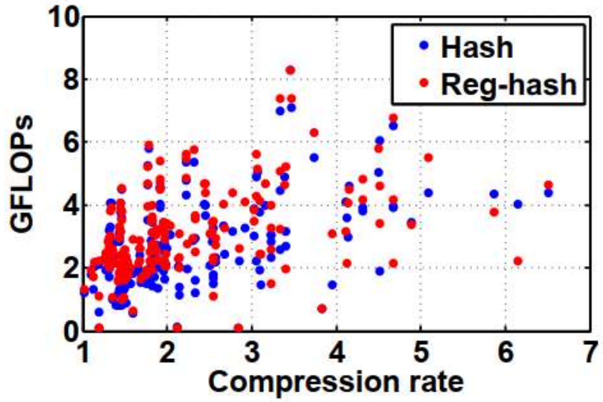
1.双精度浮点运算性能比较



(a) Overall performance.



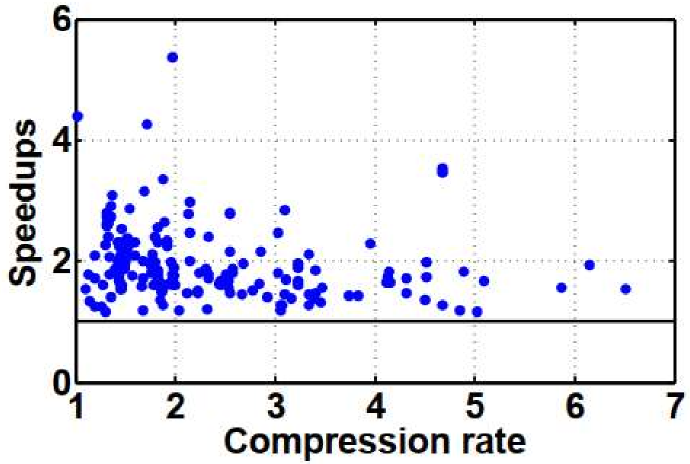
(a) Overall performance.



(a) Overall performance.

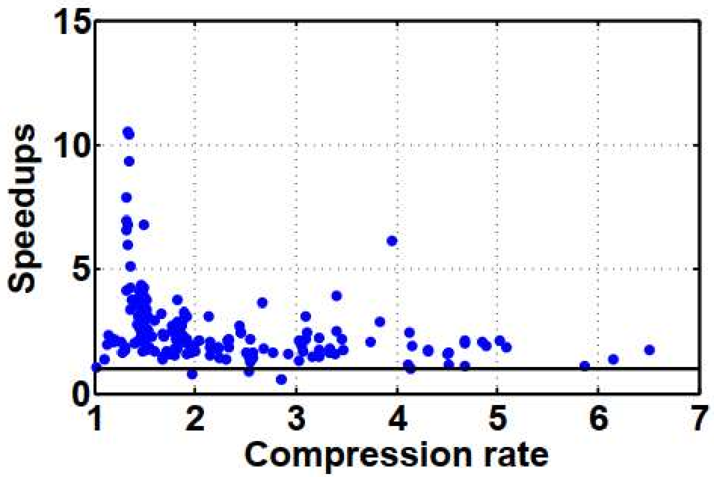
04 Evaluation & Conclusion

2.内核加速比的比较



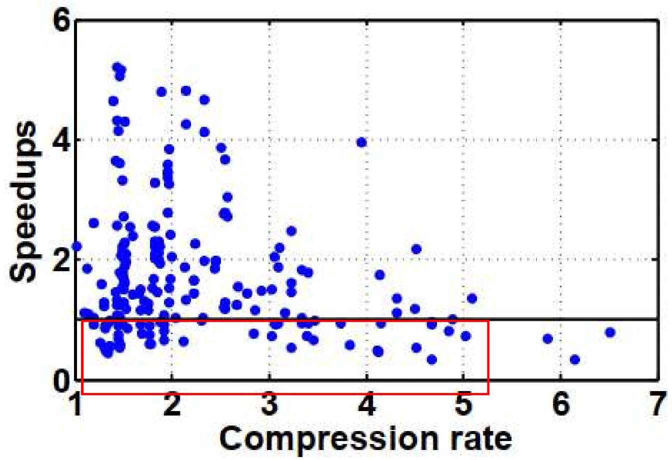
(b) Kernel speedups.

Reg-sort



(b) Kernel speedups.

Reg-Merge



(b) Kernel speedups.

Reg-Hash

04 Evaluation & Conclusion

3.总结

已有研究成果对基于寄存器感知的SpGEMM加速算法的优化不够深入。

本文利用**GPU**寄存器改进三种具有代表性的稀疏累加器的算法，充分利用寄存器在大规模并行架构上加速SpGEMM。

未来工作：

1.稀疏矩阵的特征与三种不同加速器之间的关系

2.利用SpGEMM实现数据重用

3.利用现有的自动调优技术在现实程序中执行算法

THANK YOU