Optimization of GPU Kernels for Sparse Matrix Computations in Hypre

Chaoyu Zhang
Department of Computer Science
Arkansas State University
Jonesboro, AR, USA
chaoyu.zhang@smail.astate.edu

Ruipeng Li (advisor)
Center for Applied Scientific Computing
Lawrence Livermore National Laboratory
Livermore, CA, USA
li50@llnl.gov

Hai Jiang (advisor)
Department of Computer Science
Arkansas State University
Jonesboro, AR, USA
hjiang@astate.edu

ABSTRACT

The acceleration of sparse matrix computations can significantly enhance the performance of algebraic multigrid (AMG) methods. In this work, we consider the GPU accelerations of the kernels of sparse matrix-vector multiplications (SpMV), sparse triangular matrix solves (SpTrSv) and sparse matrix-matrix multiplications (SpGEMM), which often represent major computational cost of AMG solvers. Existing kernels have been further optimized to fully take advantage of the CUDA and hardware support on Volta GPUs, which yielded significant performance improvement. The presented kernels have been put in HYPRE for solving large scale linear systems on HPC equipped with GPUs. The implementations of these kernels in Hypre and the optimization techniques will be discussed.

CCS CONCEPTS

• **Mathematics of computing → Mathematical software performance**.

KEYWORDS

GPU kernel optimizations, sparse matrix computations, algebraic multigrid

ACM Reference Format:
Chaoyu Zhang, Ruipeng Li (advisor), and Hai Jiang (advisor). 2019. Optimization of GPU Kernels for Sparse Matrix Computations in Hypre. In *Proceedings of SC '19: The International Conference for High Performance Computing, Networking, Storage, and Analysis (SC '19)*. ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The algebraic multigrid (AMG) solver in Hypre has recently been enabled to be executed on GPUs, the overall performance of which heavily depends on the throughput of a number of sparse matrix kernels. The computation in the form of sparse matrix-vector multiplications (SpMV) is repeatedly performed in each iteration of the solve phase of AMG, which also applies to sparse triangular matrix solves (SpTrSv) if Gauss-Seidel types of smoothers are used. On the other hand, sparse matrix-matrix multiplications (SpGEMM) are often found the most expensive computation in the setup phase of AMG to compute coarse-grid operators as Galerkin products.

SC '19, November 17–22, 2019, Denver, Colorado

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 SPARSE MATRIX-BY-VECTOR

Row-wise parallel SpMV algorithms [2, 3, 7] are adopted in this work where a group of threads works collectively on one row of the matrix which is assumed to be stored in the compressed sparse row (CSR) format. These algorithms have been shown to be efficient on GPUs. In this work, we improved the existing SpMV code available from [7] by applying the following optimizations: Automatically choosing the number of threads used for a row based on the average number of nonzeros in rows; Using warp shuffle instructions to perform the reductions, as opposed to using shared memory, which provides faster data exchanges in warps and memory access with lower latency and higher bandwidth; Applying the ideas of first-add-during-load to mitigate the idle-warp problem in the reduction, and algorithm cascading to combine sequential and parallel reduction.

3 SPARSE TRIANGULAR SOLVE

SpTrSv is traditionally deemed as a sequential computation in general. The level scheduling method [7, 11–13] exploits the parallelism in this computation by reorganizing the unknowns into levels, where the unknowns within each level can be computed in parallel. We optimized the implementation of the level scheduling method from [7] by similar skills used in SpMV and by grouping kernels with only one CUDA block into a single kernel to reduce the cost of kernel launches.

Finer level of parallelism can be achieved by a more aggressive element-based scheduling algorithm [6, 8], and thus the global synchronizations needed in the level scheduling method can be also avoided. This approach often yields much higher performance especially for the cases with a large number of levels. The algorithm of element-scheduling uses a counter scheme for the dependencies of each known and so enjoys a finer level of parallelism and a more aggressive scheduling, compared with the level-scheduling approach. The algorithm of DCSRV2 solver in cuSPARSE has not been published, so it is unknown. However, the down side of this algorithm is that it requires column access of the matrix. Therefore, the memory requirement is actually doubled.

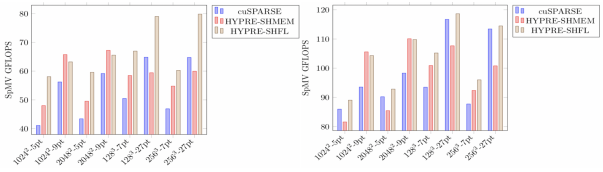

Figure 1: SpMV performance on an NVIDIA P-100 and a V-100 GPU

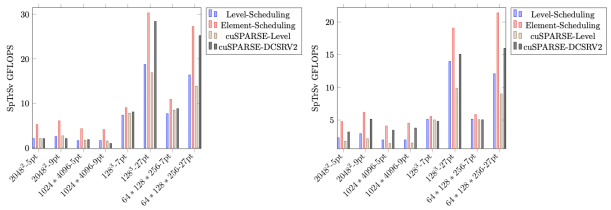

Figure 2: SpTrSv performance on an NVIDIA P-100 and a V-100 GPU

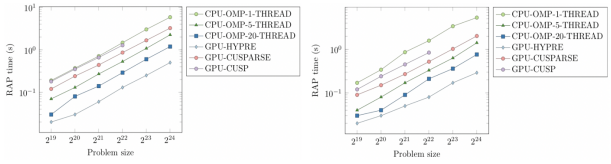

Figure 3: SpGEMM performance on an NVIDIA P-100 and a V-100 GPU

4 SPARSE MATRIX-BY-MATRIX

Efficiently computing the product of two sparse matrices on GPUs is known to be a much more difficult problem compared with the other two kernels considered in this work [1, 4, 5, 9, 10]. The difficulties mainly lie in developing GPU-appropriate *sparse accumulators*, the issue of dynamic memory allocation for the product, and the load-balancing issues among the rows. The memory is indeed very limited on the current GPUs. The out-of-memory issue has not been addressed in this work, which means we always assume the local problems can fit on a GPU. This is a reasonable assumption in the context of parallel AMG solvers, which is able to solve large-scale systems that can have billions of degrees of freedom. The global system is distributed across a large number of compute nodes with multiple GPUs on each node. As we showed, the size of the local problems can be up to several million, which is usually quite large for the settings of real numerical simulations. Our GPU implementation of SpGEMM is based on hash table as the sparse accumulator, which allows the most concurrency and the efficiency in memory use, and is able to exploit all the levels of parallelism available in the multiplication with a low memory requirement. Our hash-based SpGEMM algorithm is a single warp works on a row of the product matrix. Currently, a simple static scheduling approach is used, whereas a more complicated dynamic approach using a shared job queue to schedule warps is under the investigation.

5 NUMERICAL RESULTS

The implementations of the considered kernels in Hypre have been compared with the state-of-art libraries from NVIDIA. The experiment results demonstrated the efficiencies of our algorithms. Figure 1 shows the GFLOPS numbers of the SpMV kernels for Laplacian matrices, where `HYPRE_SHMEM` and `HYPRE_SHFL` are the two SpMV implementations in Hypre with shared memory and warp shuffle respectively. Figure 2 presents the GFLOPS numbers of a forward and a back Gauss-Seidel relaxations using the SpTrSv kernels with the level-scheduling and the element-scheduling methods. Finally, Figure 3 gives the timing results (in log-scale) of computing Galerkin products in AMG for 3-D 7-point Laplacians of sizes from 0.5 million to 16 million, where the SpGEMM algorithm was compared with the ones in CUSP and cuSPARSE and the CPU kernel in Hypre threaded with OpenMP. In Figure 3, The CPU used was IBM Power9, which has 40 cores. The CPU code was parallelized with OpenMP. The CPU SpGEMM for $C = AB$ partition the rows of C into equally sized row blocks and each thread computes each block in parallel. It is the size of the matrix R*A*P from the triple matrix product.According to experimental data, the authors propose algorithms that are up to 2 times faster than cuSPARSE.

REFERENCES

[1] N. Bell, S. Dalton, and L. N. Olson. 2012. Exposing Fine-Grained Parallelism in Algebraic Multigrid Methods. *SIAM Journal on Scientific Computing* 34, 4 (2012), C123–C152. https://doi.org/10.1137/110838844 arXiv:https://doi.org/10.1137/110838844

[2] N. Bell and M. Garland. 2008. *Efficient Sparse Matrix-Vector Multiplication on CUDA.* NVIDIA Technical Report NVR-2008-004. NVIDIA Corporation.

[3] N. Bell and M. Garland. 2009. Implementing Sparse Matrix-vector Multiplication on Throughput-oriented Processors. In *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis (SC '09).* ACM, New York, NY, USA, Article 18, 11 pages. https://doi.org/10.1145/1654059.1654078

[4] M. Deveci, C. Trott, and S. Rajamanickam. 2017. Performance-portable sparse matrix-matrix multiplication for many-core architectures. In *2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW).* 693–702. https://doi.org/10.1109/IPDPSW.2017.8

[5] F. Gremse, K. Küpper, and U. Naumann. 2018. Memory-Efficient Sparse Matrix-Matrix Multiplication by Row Merging on Many-Core Architectures. *SIAM Journal on Scientific Computing* 40, 4 (2018), C429–C449. https://doi.org/10.1137/17M1121378 arXiv:https://doi.org/10.1137/17M1121378

[6] R. Li. 2017. On Parallel Solution of Sparse Triangular Linear Systems in CUDA. *CoRR* abs/1710.04985 (2017). arXiv:1710.04985 http://arxiv.org/abs/1710.04985

[7] R. Li and Y. Saad. 2013. GPU-accelerated preconditioned iterative linear solvers. *The Journal of Supercomputing* 63 (2013), 443–466. Issue 2. https://doi.org/10.1007/s11227-012-0825-3

[8] W. Liu, A. Li, J. D. Hogg, I. S. Duff, and B. Vinter. 2016. *A Synchronization-Free Algorithm for Parallel Sparse Triangular Solves.* Springer International Publishing, 617–630. https://doi.org/10.1007/978-3-319-43659-3_45

[9] W. Liu and B. Vinter. 2014. An Efficient GPU General Sparse Matrix-Matrix Multiplication for Irregular Data. In *Proceedings of the 2014 IEEE 28th International Parallel and Distributed Processing Symposium (IPDPS '14).* IEEE Computer Society, Washington, DC, USA, 370–381. https://doi.org/10.1109/IPDPS.2014.47

[10] Y. Nagasaka, S. Matsuoka, A. Azad, and A. Buluc. 2018. High-Performance Sparse Matrix-Matrix Products on Intel KNL and Multicore Architectures. In *Proceedings of the 47th International Conference on Parallel Processing Companion (ICPP '18).* ACM, New York, NY, USA, Article 34, 10 pages. https://doi.org/10.1145/3229710.3229720

[11] M. Naumov. 2011. Parallel solution of sparse triangular linear systems in the preconditioned iterative methods on the GPU. *NVIDIA Corp., Westford, MA, USA, Tech. Rep. NVR-2011* 1 (2011).

[12] A. Picciau, G. E. Inggs, J. Wickerson, E. C. Kerrigan, and G. A. Constantinides. 2016. Balancing Locality and Concurrency: Solving Sparse Triangular Systems on GPUs. In *2016 IEEE 23rd International Conference on High Performance Computing (HiPC).* 183–192. https://doi.org/10.1109/HiPC.2016.030

[13] B. Suchoski, C. Severn, M. Shantharam, and P. Raghavan. 2012. Adapting Sparse Triangular Solution to GPUs. In *2012 41st International Conference on Parallel Processing Workshops.* 140–148. https://doi.org/10.1109/ICPPW.2012.23