INTERNATIONAL JOURNAL OF
NUMERICAL ANALYSIS AND MODELING, *SERIES B*
Volume 3, Number 2, Pages 185–191

SPARSE MATRIX-VECTOR MULTIPLICATION ON NVIDIA GPU

HUI LIU, SONG YU, ZHANGXIN CHEN, BEN HSIEH AND LEI SHAO

Abstract. In this paper, we present our work on developing a new matrix format and a new sparse matrix-vector multiplication algorithm. The matrix format is HEC, which is a hybrid format. This matrix format is efficient for sparse matrix-vector multiplication and is friendly to preconditioner. Numerical experiments show that our sparse matrix-vector multiplication algorithm is efficient on GPU.

Key words. sparse matrix-vector multiplication, GPU, HEC, parallel algorithm

1. Introduction

Sparse matrix-vector multiplication (SPMV) arises in numerous computational areas, such as eigenvalue problems and the solution of large-scale sparse linear system. Krylov subspace solvers [3, 4] and algebraic multigrid solvers [1, 2, 4] are general methods for these linear systems. For these linear solvers, the sparse matrix-vector multiplication operations control the total running time. It's critical to design efficient sparse matrix-vector multiplication algorithms.

NVIDIA Tesla GPUs are powerful in floating point calculation [13, 14]. Take the NVIDIA Telsa C2050 as an example. This GPU has a peak performance of 1030GFlops on single precision calculation, while the latest CPUs have about 100GFlops [11]. GPUs are much faster than CPUs. Nowadays, GPU computing has been popular in various scientific computing applications due to its superiority over conventional CPU. GPUs have been applied to FFT [13], Krylov subspace solvers [11, 12, 6], algebraic multigrid solvers [1] and reservoir simulation [11, 6].

In this paper, we focus on the sparse matrix-vector multiplication on GPU. Baskaran et al. developed a SPMV algorithm for CSR (Compressed Sparse Row) format matrix, where each row was calculated by a half warp [5]. Li et al. designed a SPMV algorithm for the JAD (Jagged Diagonal) format matrix [12, 11]. Bell and Garland investigated different kinds of matrix formats and SPMV algorithms in [9, 10], in which the HYB format matrix and the corresponding SPMV algorithm was also designed. Here a similar idea to HYB is applied, and we develop a new matrix format, HEC, which is hybrid of ELL format matrix [8] and CSR format matrix. A new sparse matrix-vector multiplication algorithm is also developed. This SPMV algorithm is efficient and the new matrix format is friendly to preconditioners, such as ILU(k), ILUT and domain decomposition preconditioners [7]. Authors investigated vector algorithm and serial algorithm for CSR format in [9, 10]. The memory access pattern for serial algorithm isn't coalesced, so this SPMV kernel is alway the worst. For the vector version, when the matrix is very dense, each warp has sufficient tasks to complete and the memory access is regular, its performance can be better than others. Memory access for ELL format is always coalesced, but it may consume too much memory even if one row is too long. HYB and HEC

Received by the editors January 1, 2012 and, in revised form, March 22, 2012.
2000 *Mathematics Subject Classification.* 65F30, 65F50, 65Y05, 65Y10, 68W10.
This research was supported by NSERC/AIEE/Foundation CMG and AITF Chairs.

are hybrid formats, and most non-zeros values are stored in ELL part. Memory access for these two formats are coalesced and memory usage is moderate. The HYB format and HEC format are general formats for numerical linear algebra. The algorithm difference between HYB and HEC formats is how to calculate the COO part and CSR part. Authors in [9, 10] used reduction operations, and in this case the communication may be complicated. In this paper, we use one thread for each row in CSR part and therefore there isn't any communication. From the numerical experiments we can see that our algorithm is better in most cases.

The layout is as follows. In §2, the new matrix format and the SPMV algorithm are introduced. In §3, numerical experiments are performed to test our SPMV algorithm and other SPMV algorithms.

2. Sparse Matrix-Vector Multiplication

In this section, we will investigate commonly used matrix formats and our new matrix format, HEC, is introduced. Then the sparse matrix-vector multiplication algorithm for GPU is proposed.

2.1. Matrix format. The ELL format was introduced in ELLPACK [8], which is shown in Figure 1. From the figure we can find that this matrix contains two matrices. The left matrix is for the column indices and the right matrix is for the non-zero values. The row length of these two matrices is the same. It's clear that the memory access pattern for the ELL format matrix is regular, and therefore, the performance is high. The disadvantage of the ELL matrix is that even if one row has too many non-zero values, all rows should have the same length. In this case, it's a huge waste of memory space, which is limited on current GPUs.

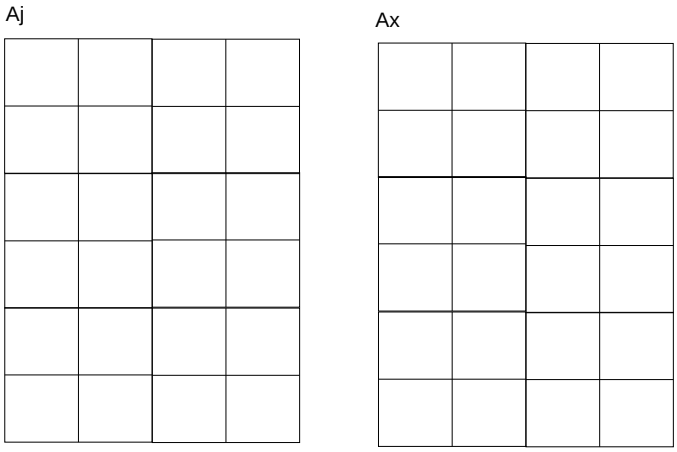

Figure 1: ELL matrix format.

The COO (Coordinate) format is shown in Figure 2. The matrix has three arrays, which are for row indices, column indices and non-zero values. The three arrays has the same length, which is the number of non-zero values. Data access pattern for this matrix is regular. The shortcoming is that when one row is split into different threads on GPU, we need to apply reduction operations to obtain final result. This matrix has good average performance, but may not have the best peak performance.

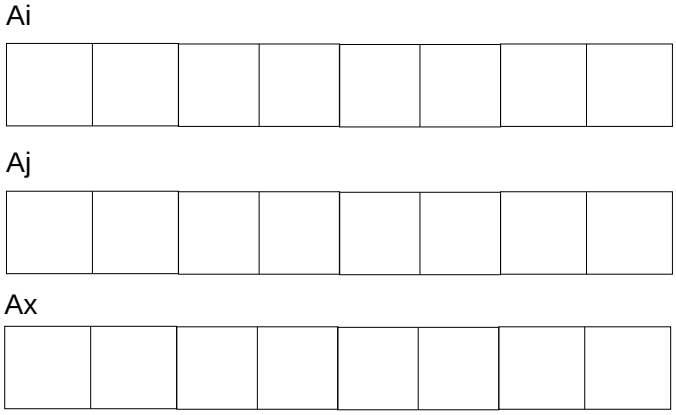

Figure 2: COO matrix format.

The authors [9, 10] designed a hybrid matrix format, HYB (Hybrid of ELL and COO). The HYB format matrix contains two submatrices, which is ELL matrix and COO matrix. The regular part of a given matrix is stored in the ELL part and the irregular part is stored in the COO part. This matrix has good average performance and peak performance.

The CSR (Compressed Sparse Row) format is another general matrix format, which is shown in Figure 3. This matrix contains three arrays. For a given matrix whose row dimension is n, the length of the array, Ap, which is used for offset of each row, is $n + 1$, and the length of the rest two arrays, which are for column indices and non-zero values, is the number of non-zero values. The CSR format is the most commonly used format for computational sciences.

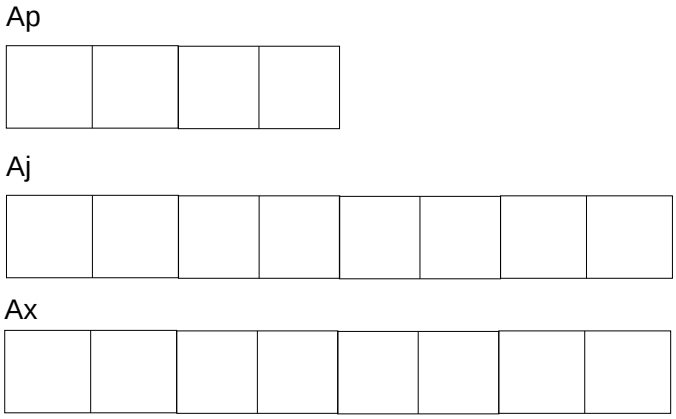

Figure 3: CSR matrix format.

In this paper, the similar idea as [9, 10] is applied. The matrix format we design is also hybrid, which is HEC, hybrid ELL and CSR. From its name, we know that an HEC matrix contains two submatrices: an ELL matrix and a CSR

matrix. One reason we design this format is that this format is friendly to ILU-like preconditioners. Take domain decomposition preconditioner [7] as an example, where a lower triangular linear system and an upper triangular linear system should be solved. For triangular problem, we need to know the diagonal element, elements before diagonal and elements after diagonal. With this HEC matrix, we always know where each row lies. Before we apply this kind of matrix format, we need to convert a given matrix to HEC format, the row length of the ELL matrix is computed first, which is obtained by solving a minimum problem [10]:

$$\text{Find } l \ (l \geq 0) \text{ such that } w(i) = i * n + p_r * nz(i) \text{ is minimized,} \tag{1}$$

where p_r is the relative performance of the ELL and CSR matrices and $nz(i)$ is the number of non-zeros in the CSR part when the row length of the ELL part is i. The minimum problem is easy to solve. Before we solve it, we should know p_r, which can be obtained by testing. A typical value for p_r is 20 [10].

2.2. SPMV algorithm. In [9, 10], the authors discussed the sparse matrix-vector multiplication algorithms for COO, CSR, ELL and HYB format matrices. In this section, we describe our algorithm only.

When a given matrix is stored in GPU, the ELL part of the HEC matrix is in column-major order and each column is aligned to ensure that the data access is coalesced [13, 14]. The CSR matrix is stored in row-major order. When the HEC matrix is used for ILU-related preconditioners, the HEC matrix is restricted that each row of the CSR matrix has at least one element if the matrix is a lower triangular matrix. In this case, the element is diagonal element [6].

With the HEC format, the algorithm for sparse matrix-vector multiplication on NVIDIA GPU is straightforward. For a given matrix A, whose row dimension is n, and vector x, a sparse matrix-vector multiplication operation $y = Ax$ can be completed in two steps, which is described in Algorithm 1. In the first step, the ELL matrix is calculated first and each thread is responsible for one row. Since the ELL matrix is stored in column-major order, the data access for the ELL matrix is well coalesced. Temporary results are stored in array y. In the second step, the CSR matrix is computed. As we do for the ELL matrix, here each thread is also responsible for one row. The results are added to the array y. Then we finish the SPMV operation $y = Ax$.

Algorithm 1 Sparse Matrix-Vector Multiplication, $y = Ax$

```
1: for i = 1: n do                    ▷ ELL, Use one GPU kernel to deal with this loop
2:     the i-th thread calculate the i-th row of ELL matrix;     ▷ Use one thread
3: end for
4:
5: for i = 1: n do                    ▷ CSR, Use one GPU kernel to deal with this loop
6:     the i-th thread calculate the i-th row of CSR matrix;     ▷ Use one thread
7: end for
```

The Algorithm 1 is easy to implement. In practice, we may need other BLAS 2 [3] operations, such as $y = \alpha Ax$, $y = \alpha Ax + \beta y$ and $z = \alpha Ax + \beta y$, where A is a matrix, x, y, z are vectors, α and β are real numbers. The algorithms and codes for these operation are similar to $y = Ax$.

3. Numerical results

In this section, numerical experiments are performed. The CPU we use is Intel Xeon X5570 and The GPUs we use are NVIDIA Tesla C2050/C2070. The operating system is Fedora 13 X86_64 with CUDA Toolkit 4.0 and GCC 4.4. All CPU codes are compiled with -O3 option. Fourteen matrices are employed, which are from the University of Florida sparse matrix collection [15] and a black oil simulator [16]. These matrices are listed in Table 1.

Table 1: Matrices used for testing

Matrix	N	Non-zero	NNZ/N
human_gene1	22283	12345963	554.1
msc23052	23052	588933	25.5
TSOPF_FS_b300_c2	56814	4391071	77.3
cfd2	123440	1605669	13.0
ESOC	327062	6019939	18.4
cage13	445315	7479343	16.8
af_shell8	504855	9046865	17.9
parabolic_fem	525825	2100225	4.0
Emilia_923	923136	20964171	22.7
atmosmodd	1270432	8814880	6.9
Serena	1391349	32961525	23.7
Hook_1498.mtx	1498023	30436237	5.9
SPE10	2188851	29915573	13.7
cage15	5154859	99199551	19.2

3.1. Single Precision. Five algorithms are applied. In this section, the CSR(S) means a scalar algorithm for CSR format matrix. The CSR(V) means a vector algorithm for CSR format matrix. The ELL means algorithm for ELL format matrix. HYB and HEC mean algorithms for HYB and HEC format matrices respectively. Details for these algorithm can be found in [9, 10]. Single precision data type, which is float in C/C++ language, is applied. Only speedup is recorded in Table 2.

From Table 2 we can see that when the average number of non-zeros is large, the CSR(V) is better than CSR(S) and other algoirhtms, such as for matirx human_gene1, msc23052 and TSOPF_FS_b300_c2. For CSR(S), since the data access of isn't well coalesced, the performance for this algorithm isn't high. The only exception is matrix parabolic_fem, where the average number of non-zeros is low.

The performance of HYB and HEC is stabler than ELL, and higher than ELL in most cases. This means HYB and HEC are suitable for general sparse matrix-vector multiplication. From Table 2, we can find that HEC has better average performance than HYB.

3.2. Double Precision. Double precision is applied in this section and the same five algorithms are employed. The results are collected in Table 3.

From Table 3, we can see that the CSR(V) has better performance than other algorithms for the first three matrices, which have large average number of non-zeros. HYB and HEC always have better average performance, which also indicate that HYB and HEC are suitable for general sparse matrices. Our HEC format has better speedup than HYB format in most cases.

Table 2: Speedup of SPMV, Single Precision

Matrix	CSR(S)	CSR(V)	ELL	HYB	HEC
human_gene1	0.64	6.34	0.82	2.86	1.04
msc23052	0.66	5.19	1.59	2.96	2.55
TSOPF_FS_b300_c2	0.60	11.90	6.37	6.38	8.41
cfd2	1.62	3.61	9.25	8.61	13.77
ESOC	1.24	4.47	17.30	17.28	17.40
cage13	1.24	3.62	8.48	11.15	12.84
af_shell8	1.23	4.87	10.62	12.44	14.04
parabolic_fem	7.38	1.78	14.56	11.28	14.37
Emilia_923	0.91	5.12	8.30	11.01	11.42
atmosmodd	4.43	1.82	17.63	17.60	17.66
Serena	0.87	5.35	2.30	10.39	10.34
Hook_1498.mtx	0.99	4.79	6.59	10.37	10.51
SPE10	1.60	2.66	1.59	16.28	14.71
cage15	1.08	1.65	7.90	12.63	12.80

Table 3: Speedup of SPMV, Double Precision

Matrix	CSR(S)	CSR(V)	ELL	HYB	HEC
human_gene1	0.76	4.85	0.84	2.68	1.13
msc23052	0.71	3.63	1.55	2.29	2.50
TSOPF_FS_b300_c2	0.79	8.62	5.42	5.41	8.96
cfd2	1.41	2.26	8.06	6.86	11.61
ESOC	1.06	2.62	11.56	11.56	11.61
cage13	1.25	2.36	7.56	9.42	11.04
af_shell8	1.17	3.58	8.66	9.63	11.12
parabolic_fem	4.36	1.12	9.97	7.56	10.00
Emilia_923	0.97	3.69	6.64	8.36	8.65
atmosmodd	2.94	1.38	14.54	14.50	14.57
Serena	0.98	3.58	1.79	7.26	7.29
Hook_1498.mtx	0.94	3.48	5.39	7.89	8.01
SPE10	1.13	1.74	1.24	11.15	10.27
cage15.mtx	1.15	2.16	7.09	10.72	10.86

4. Conclusion

We have developed a new matrix format HEC, which is a hybrid format, and the corresponding SPMV algorithm for general sparse matrices is developed. The numerical experiments show that our SPMV algorithm is efficient and is suitable for general sparse matrices.

Acknowledgements

The support of Department of Chemical and Petroleum Engineering, University of Calgary and Reservoir Simulation Group is gratefully acknowledged. The research is partly supported by NSERC/AIEE/Foundation CMG and AITF Chairs.

References

[1] G. Haase, M. Liebmann, C. C. Douglas and G. Plank, A Parallel Algebraic Multigrid Solver on Graphics Processing Units, HIGH PERFORMANCE COMPUTING AND APPLICATIONS, 2010, 38-47.

[2] Nathan Bell, Steven Dalton and Luke Olson, Exposing Fine-Grained Parallelism in Algebraic Multigrid Methods, NVIDIA Technical Report NVR-2011-002, June 2011

[3] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine and H. Van der Vorst , Templates for the solution of linear systems: building blocks for iterative methods, 2nd Edition, SIAM, 1994.

[4] Y. Saad, Iterative methods for sparse linear systems (2nd edition), SIAM, 2003.

[5] Muthu Manikandan Baskaran and Rajesh Bordawekar, Optimizeing Sparse Matrix-Vector Multiplication on GPUs, In Ninth SIAM Conference on Parallel Processing for Scientific Computing, 2008.

[6] H. Liu, S. Yu, Z. Chen, B. Hsieh and L. Shao, Parallel Preconditioners for Reservoir Simulation on GPU, SPE Latin American and Caribbean Petroleum Engineering Conference held in Mexico City, Mexico, 16-18 April 2012, SPE 152811-PP.

[7] X.-C. Cai and M. Sarkis, A restricted additive Schwarz preconditioner for general sparse linear systems, SIAM J. Sci. Comput., 21, 1999, pp. 792-797.

[8] R. Grimes, D. Kincaid, and D. Young, ITPACK 2.0 User's Guide, Technical Report CNA-150, Center for Numerical Analysis, University of Texas, August 1979.

[9] N. Bell and M. Garland, Efficient sparse matrix-vector multiplication on CUDA, NVIDIA Technical Report, NVR-2008-004, NVIDIA Corporation, 2008.

[10] N. and M. Garland, Implementing sparse matrix-vector multiplication on throughput-oriented processors, Proc. Supercomputing, November 2009, 1-11.

[11] H. Klie, H. Sudan, R. Li, and Y. Saad, Exploiting capabilities of many core platforms in reservoir simulation, SPE RSS Reservoir Simulation Symposium, 21-23 February 2011

[12] R. Li and Y. Saad, GPU-accelerated preconditioned iterative linear solvers, Technical Report umsi-2010-112, Minnesota Supercomputer Institute, University of Minnesota, Minneapolis, MN, 2010.

[13] NVIDIA Corporation, Nvidia CUDA Programming Guide (version 3.2), 2010.

[14] NVIDIA Corporation, CUDA C Best Practices Guide (version 3.2), 2010.

[15] T. A. Davis, University of Florida sparse matrix collection, NA digest, 1994.

[16] Z. Chen, G. Huan, and Y. Ma, Computational methods for multiphase flows in porous media, in the Computational Science and Engineering Series, Vol. 2, SIAM, Philadelphia, 2006.

Department of Chemical and Petroleum Engineering, University of Calgary, Calgary, AB, Canada

E-mail: hui.j.liu@ucalgary.ca, yusong0926@gmail.com, zhachen@ucalgary.ca, bhsieh@ucalgary.ca, jedyfun@gmail.com

URL: http://schulich.ucalgary.ca/chemical/JohnChen