ADAPTIVITY AND COMPRESSION: A RECIPE FOR SPARSE MATRIX VECTOR MULTIPLICATION ON GPUS

Marco Maggioni and Tanya Berger-Wolf, Department of Computer Science

UIC UNIVERSITY OF ILLINOIS AT CHICAGO

SPMV KERNEL

The Sparse Matrix-Vector multiplication (SpMV) is one of the most important computational kernels and holds a fundamental role in countless scientific and engineering applications. Calculating $\mathbf{y} = \mathbf{Ax}$ essentially reduces to many independent dot products such as the following

$$y_i = \sum_{\forall j : a_{ij} \in \mathbf{A_i}} a_{ij} x_j \qquad (1)$$

where $\mathbf{A_i}$ is a row of large sparse matrix $\mathbf{A}$, a_{ij} is a nonzero value, and x_j is an element of the dense vector $\mathbf{x}$.

CHALLENGES

The SpMV is intrinsically parallelizable but matrix sparsity and irregularity lead to the following challenges for optimization :

- Unbalanced workload as well as poor cache locality in accessing the dense $\mathbf{x}$.
- Low arithmetic intensity which categorizes the SpMV as bandwidth-limited.

The widespread importance of the SpMV leaded to a significant research effort to optimize its efficiency on GPUs. Each proposed sparse matrix formats can efficiently represent a particular structure (*e.g.* diagonal or blocked), with the aim of optimizing fine-grain parallelism, memory pattern efficiency, and memory footprint.

PROPOSAL

In this work, we propose instead a one-size-fits-all format that combines **adaptivity** and **compression** into an ELL-based data structure to obtain the best possible performance for the SpMV kernel on GPUs. The idea is to directly tackle the two fundamental challenges just mentioned. We cope with matrix irregularity and memory boundedness by creating **balanced warps** and using **delta compression** on the nonzero indices.

ELL-BASED SPARSE FORMATS

The sparse structure allows to conveniently represent a matrix by only its nonzero elements (including row and column indices).

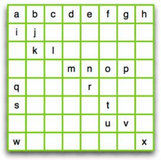


Figure 1: Sparse Matrix

The ELL sparse matrix format [1] is particularly well-suited to vector architectures (SIMT execution). Its basic idea is to compress a sparse $n \times m$ matrix using two dense $n \times k$ data structures (one for nonzero values and one for column indices), where k is the maximum number of nonzeros per row.

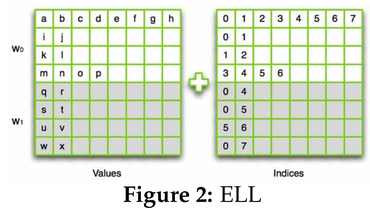


Figure 2: ELL

The SpMV computation is mapped as one thread (or warp lane) per row, so warps process nonzeros in a lockstep fashion. The ELL sparse format is intrinsically inefficient for irregular matrices, leading to zero-padding and waste of computation. The Warped ELL [2] format provides a more efficient data layout by slicing the matrix with warp granularity. Each slice is then stored with a local ELL structure with k depending by the longest row in the warp (as opposed to the entire matrix).

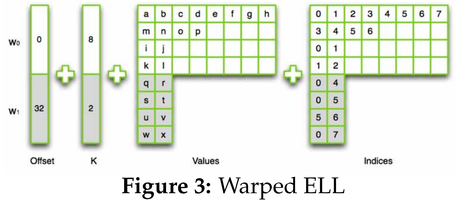


Figure 3: Warped ELL

The efficiency can be further improved by local row reordering, reducing the variability of nonzeros per row within warps without affecting the cache locality.

ADAPTIVITY

The AdELL sparse matrix format [3] has been designed to improve the efficiency in case of matrices with an irregular sparsity pattern. This format is based on the idea of creating warps well-suited for a vectorized execution. This is done by allocating an adaptive number t_i of working threads to each row i depending on the number of nonzeros nnz_i. A best-fit warp-balancing heuristic policy [3] takes care to assign more threads to heavyweight rows in order to provide a balanced and efficient computation. The AdELL format also exploits warp intrinsic synchronization and deals with very long rows by using multiple warps and atomic operations.

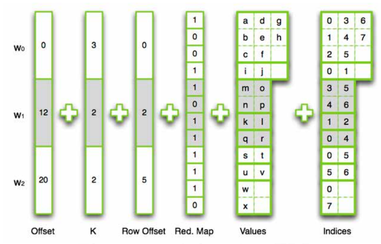


Figure 4: Adaptive ELL

COMPRESSION

We propose a warp-grained compression technique to reduce the storage associated with column indices for ELL-based formats. The idea is to use differential encoding between consecutive nonzeros since column distances may be often represented using smaller integer data types. This technique relies on thread independency to provide an execution well-suited for GPUs. In fact, during SpMV each thread processes its nonzeros and it is fully able to compute the current column index by summing the differential encoding with its previous column index. We embed this compression scheme into AdELL creating a novel sparse format called CoAdELL.

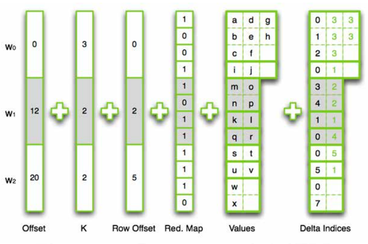
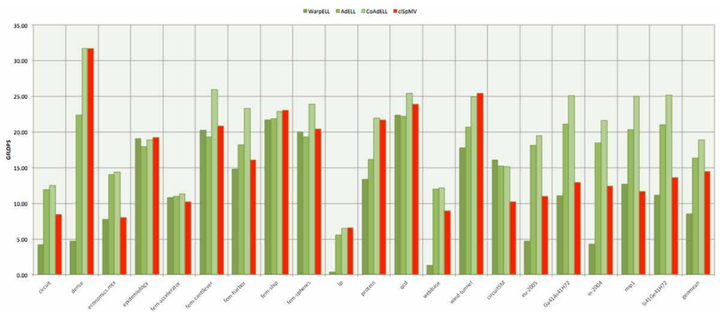


Figure 5: Compressed AdELL

EXPERIMENTAL RESULTS



We implemented the CoAdELL sparse format in CUDA and observed an overall 31% performance improvement over the state-of-the-art clSpMV framework [4] for double-precision calculation on a GTX580. We also observed a 40% improvement over the recent BRO-HYB format [5].

REFERENCES & ACKNOLEDGMENTS

[1] N. Bell and M. Garland, **"Implementing Sparse Matrix-Vector Multiplication on Throughput-Oriented Processors"**, SC, 2009
[2] M. Maggioni, T. Berger-Wolf, and J. Liang, **"GPU-based Steady-State Solution of the Chemical Master Equation"**, HiCOMB, 2013
[3] M. Maggioni and T. Berger-Wolf, **"AdELL: An Adaptive Warp-Balancing ELL Format for Efficient Sparse Matrix-Vector Multiplication on GPUs"**, ICPP, 2013
[4] B. Su and K. Keutzer, **"clSpMV: Across-platform OpenCL SpMV framework on GPUs"**, SC, 2012
[5] W. T. Tang et Al., **"Accelerating Sparse Matrix-Vector Multiplication on GPUs using Bit-Representation-Optimized Schemes**, SC, 2013

This work was supported by NSF grants IIS-106468.