Merge-based Sparse Matrix-Vector Multiplication (SpMV) using the CSR Storage Format

Duane Merrill

NVIDIA Corp.
dumerrill@nvidia.com

Michael Garland

NVIDIA Corp.
mgarland@nvidia.com

Abstract

We present a perfectly balanced, “merge-based” parallel method for computing sparse matrix-vector products (SpMV). Our algorithm operates directly upon the Compressed Sparse Row (CSR) sparse matrix format, a predominant in-memory representation for general-purpose sparse linear algebra computations. Our *CsrMV* performs an equitable multi-partitioning of the input dataset, ensuring that no single thread can be overwhelmed by assignment to (a) arbitrarily-long rows or (b) an arbitrarily-large number of zero-length rows. This parallel decomposition requires neither offline preprocessing nor specialized/ancillary data formats. We evaluate our method on both CPU and GPU microarchitecture across an enormous corpus of diverse real world matrix datasets. We show that traditional *CsrMV* methods are inconsistent performers subject to order-of-magnitude slowdowns, whereas the performance response of our method is substantially impervious to row-length heterogeneity.

Categories and Subject Descriptors F.2.1 [Numerical Algorithms and Problems]: Computations on matrices; D.1.3 [Programming Techniques]: Concurrent programming

Keywords SpMV, segmented reduction, sparse matrix, sparse graph, parallel decomposition, merge, merge-path, GPU

1. Introduction

High performance algorithms for sparse linear algebra are commonplace within many application domains, including computational science, graph analytics, and machine learning. The sparse matrix-vector product (SpMV) operation is of particular importance for solving sparse linear systems, eigenvalue systems, and similar problems. In its simplest form, a SpMV operation computes $y = Ax$ where the matrix A is sparse and vectors x and y are dense.

SpMV usage is typically repetitive, often to the point of dominating overall application runtime. The primary design concerns for parallel SpMV performance are (1) uniform processor utilization and (2) efficient use of memory bandwidth [4]. Achieving these goals requires complementary strategies for parallel decomposition and matrix storage formatting, and has remained an active field of research.

Figure 1. The three-array compressed-sparse-row (CSR) format for representing non-zero values within sparse matrix datasets.

However, improving SpMV performance with innovative matrix formatting comes at significant practical cost. Applications predominantly rely upon the general-purpose Compressed Sparse Row (CSR) format for in-memory representation (as illustrated in Figure 1). As a result, SpMV implementations that require alternative formatting ultimately burden the application with both (1) processing time for format conversion and (2) duplicated matrix storage space (because the original CSR matrix is likely required by other routines and cannot be discarded). In contrast, our “merge-based” method of *CsrMV* parallelization delivers high performance without format conversion or augmentation.

Although the CSR format is amenable to straightforward SpMV parallelization via row-wise assignment among individual threads for privatized row-reductions, contemporary parallel *CsrMV* implementations are subject to data-dependent performance degradation arising from irregular row lengths and/or wide aspect ratios. Alternatively, a “nonzero-splitting” strategy can be employed in which parallel processors are assigned regularized sections of the non-zero data (i.e., the values and column_indices arrays). As each processor consumes its section of non-zeros, it must track its progress through the row_offsets array. Imbalance is still possible because some processors may consume arbitrarily more row offsets than others. Many heuristics have been developed to ameliorate these issues. Williams et al. survey a wide range of optimization techniques for multicore CPUs [5], and Filippone et al. survey more than sixty GPU-specific variations [2].

However, neither general strategy has proven to be a panacea. Poor performance can be exacerbated by (1) wider parallel processors that are increasingly susceptible to underutilization, (2) scale-free datasets in which a small minority of rows are many orders of magnitude longer than others, and (3) datasets having an abundant number of zero-length rows corresponding to “leaf” vertices in directed graphs. For both CPU and GPU microarchitecture, we show that these *CsrMV* strategies commonly suffer performance deficiencies of one or more orders of magnitude among similarly-sized matrices.

To eliminate workload imbalance altogether, our method adapts the fine-grained *MergePath* decomposition [3] for merging two sorted lists, allowing our implementations to distribute equal shares

This research was developed, in part, with funding from the Defense Advanced Research Projects Agency (DARPA). The views, opinions, and/or findings contained in this article/presentation are those of the author(s)/presenter(s) and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government. Distribution Statement A (Approved for Public Release, Distribution Unlimited)

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to post on servers, or to redistribute to lists, contact the Owner/Author. Request permissions from permissions@acm.org or Publications Dept., ACM, Inc., fax +1 (212) 869-0481. Copyright 2016 held by Owner/Author. Publication Rights Licensed to ACM.

PPoPP '16, March 12–16, 2016, Barcelona, Spain
 Copyright © 2016 ACM 978-1-4503-4092-2/16/03...\$15.00
 DOI: <http://dx.doi.org/10.1145/http://dx.doi.org/10.1145/2851141.2851190>

Figure 2. Merge-based CsrMV decomposition for three parallel threads. Inputs are the CSR matrix A from Figure 1 and the vector $x = [1.0, 1.0, 1.0, 1.0]$. The output is computed as $y = [2.0, 0.0, 2.0, 4.0]$.

of the *total* CsrMV problem among parallel processing elements. The result is an even utilization of parallel processing elements to provide predictably-good performance response regardless of row-irregularity and skew.

As illustrated in Figure 2, we can visualize the CsrMV work list as a 2-dimensional decision path in which the `row_offsets` array is merged with the sequence of natural numbers, which indexes the non-zeros in the `column_indices` and `values` arrays. The decision path begins in the top-left corner and ends in the bottom-right. When traced sequentially, the path moves downward when consuming indices (accumulating matrix-vector dot-products within a given row) and rightward when consuming row offset sentinels (outputting row-aggregates). The path itself is linearly indexed by the grid *diagonals*, where diagonals are enumerated from top-left to bottom-right. The goal of the *MergePath* parallel decomposition is to use the diagonals to partition this decision path into equal-length sections (one section per processing element).

In this example, the path is being split into three sections of four work items each. The fundamental insight is that each grid coordinate (i, j) along the path can be found by independent binary searches along the *diagonal_k*, where $k = i + j$. Given the diagonal constraint, we simply find the first (i, j) where `row_offsets[i]` is greater than $j - 1$. In this example, the ninth of the CsrMV work item begins the third partition, and is located at $(row_3, nonzero_5)$.

Once a given thread’s starting coordinate (i, j) is found, the remainder of its section can be processed using a sequential CsrMV algorithm starting at row_i and $nonzero_j$. Finally, the partial sums from rows that span multiple threads can be aggregated in a subsequent “reduce-value-by-key” pass.

Assuming the number of processors p is a finite constant of the underlying machine (unrelated to the number of rows or non-zeros), the searching overhead does not asymptotically affect the total work complexity, which remains linear $O(N)$. Furthermore, the decision path can be partitioned hierarchically, trivially enabling parallelization across multi-scale systems (e.g., GPUs).

We evaluate our method across the approximately 4,400 non-trivial matrices of the entire Florida Sparse Matrix Collection [1]. Figure 3 and Figure 4 plot runtime vs. dataset size (nonzeros) as performed by the Intel MKL, the NVIDIA cuSPARSE, and our merge-based CsrMV implementations. These performance land-

Figure 3. Log-plot comparison of CPU CsrMV running times (dual-socket Intel e5-2695: 24-core, 48-thread).

Figure 4. Log-plot comparison of GPU CsrMV running times (NVIDIA K40).

scapes serve to highlight performance inconsistencies. The presence of significant outliers is readily apparent for the row-based parallelizations (MKL, cuSPARSE). In comparison, our merge-based CsrMV achieves a substantially more consistent performance response. Statistically, our runtimes are much less anti-correlated to row variation (-0.07 versus -0.16 on Intel e5-2695x2, and -0.017 versus -0.24 on K40). We demonstrate speedups of up to $16\times$ and $216\times$ on the CPUs and GPU, respectively. Furthermore, we demonstrate aggregate evaluation times that are $1.1\times$ and $2.9\times$ faster for the entire corpus on the CPUs and GPU, respectively.

References

- [1] T. Davis and Y. Hu. University of Florida Sparse Matrix Collection. URL <http://www.cise.ufl.edu/research/sparse/matrices/>.
- [2] S. Filippone, V. Cardellini, D. Barbieri, and A. Fanfarillo. Parallel tridiagonal equation solvers. *ACM Trans. Math. Softw.*
- [3] S. Odeh, O. Green, Z. Mwassi, O. Shmueli, and Y. Birk. Merge Path - Parallel Merging Made Simple. In *Proceedings of the International Parallel and Distributed Processing Symposium Workshops and PhD Forum*, pages 1611–1618, May 2012.
- [4] R. W. Vuduc. *Automatic Performance Tuning of Sparse Matrix Kernels*. PhD thesis, 2003. AAI3121741.
- [5] S. Williams, L. Oliker, R. Vuduc, J. Shalf, K. Yelick, and J. Demmel. Optimization of sparse matrix-vector multiplication on emerging multicore platforms. In *Proceedings of the 2007 ACM/IEEE Conference on Supercomputing*, SC ’07, pages 38:1–38:12, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-764-3. doi: 10.1145/1362622.1362674.