Optimizing the SpMV kernel on long-vector accelerators

Constantino Gomez*†, Filippo Mantovani*†, Erich Focht‡, Marc Casas*†
*Barcelona Supercomputing Center, Barcelona, Spain
†Universitat Politècnica de Catalunya, Barcelona, Spain
‡NEC HPC Europe, Stuttgart , Germany
E-mail: {constantino.gomez, marc.casas, filippo.mantovani,}@bsc.es; erich.focht@emea.nec.com

Keywords—SpMV, NEC Vector Engine, Long-Vector Architectures, Performance Optimization

I. Extended Abstract

Sparse Matrix-Vector multiplication (SpMV) is an essential kernel for parallel numerical applications. SpMV displays sparse and irregular data accesses, which complicate its vectorization. Such difficulties make SpMV to frequently experiment non-optimal results when run on long vector ISAs exploiting SIMD parallelism. In this context, the development of new optimizations becomes fundamental to enable high performance SpMV executions on emerging long vector architectures. In our work, we improve the state-of-the-art SELL-*C*-σ sparse matrix format by proposing several new optimizations for SpMV. We target aggressive long vector architectures like the NEC Vector Engine. By combining several optimizations, we obtain an average 12% improvement over SELL-*C*-σ considering a heterogeneous set of 24 matrices. Our optimizations boost performance in long vector architectures since they expose a high degree of SIMD parallelism.

A. Background

Many different approaches have been proposed to efficiently store sparse matrices and efficiently run SpMV. One of the most common approaches, Compressed Sparse-Row (CSR), efficiently stores sparse matrices and enables simple stride-1 memory access patterns on A and y. However, accesses on x are highly irregular. Other approaches aim to mitigate the drawbacks of CSR by enlarging its storage requirements to increase the locality on x. SELL-*C*-σ [1] and ELLPACK Sparse Block [2] make use of row sorting and column blocking to improve both storage requirements and locality on x. Our work demonstrates that, although some of these approaches are very good abstractions to represent and manipulate sparse matrices, there are many unexploited opportunities to improve their performance on long vector architectures. For that, we implement, evaluate, and discuss the performance impact of several SpMV optimizations on the VE.

B. Optimizations

We revisit and adapt some optimizations previously proposed in the literature extending them with new approaches targeting long vector architectures.

In detail, we explore: *i)* the adequate sorting strategy based on the trade-off between performance and preprocessing overhead as the σ parameter increases; *ii)* the use of task-based parallelism and the impact of the task granularity in the scaling performance of SELL-*C*-σ; and *iii)* the impact of column blocking in matrices to improve locality on vector x.

In addition, our proposals to accelerate SpMV on long vector architectures are: *i)* the use of cache allocation to improve the reuse of x and deprioritization of store dependencies; *ii)* divergence flow control adapting the length of vector operations to avoid loading and computing *zero-padded* elements; *iii)* enabling loop unrolling in SELL-*C*-σ using partial loop fusion; *iv)* efficient computation of gather and scatter addresses with special instructions.

TABLE I: Optimizations applied on each of the implementations evaluated in our work.

Optimization	SELLCS	SELLCS DFC	SELLCS U8-DFC	SELLCS U8-NC	SELLCS U8-NC-DFC
Sorting strategies	•	•	•	•	•
Task-Based Parallelism	•	•	•	•	•
Column Blocking					
Cache Allocation & Store relaxation pol.				•	•
Divergent Flow Control		•	•		•
Loop unrolling			•	•	•
Efficient gather/scatter address computation	•	•	•	•	•

C. Results

The test-bench for our experiments is the NEC Vector Engine 10B. Figure 1 shows GFLOP/s performance results for a wide set of matrices. We evaluate six different implementations of the SpMV kernel: *NLC, SELLCS, SELLCS-DFC, SELLCS-U8-DFC, SELLCS-U8-NC* and *SELLCS-U8-NC-DFC*. The *NLC* category represents results obtained with the math library developed by NEC which is particularly tailored for the VE.

The improvements added by the three main optimizations visualized vary across the different matrices, as its effectiveness depends on specific matrix layout characteristics like size, sparsity or shape.

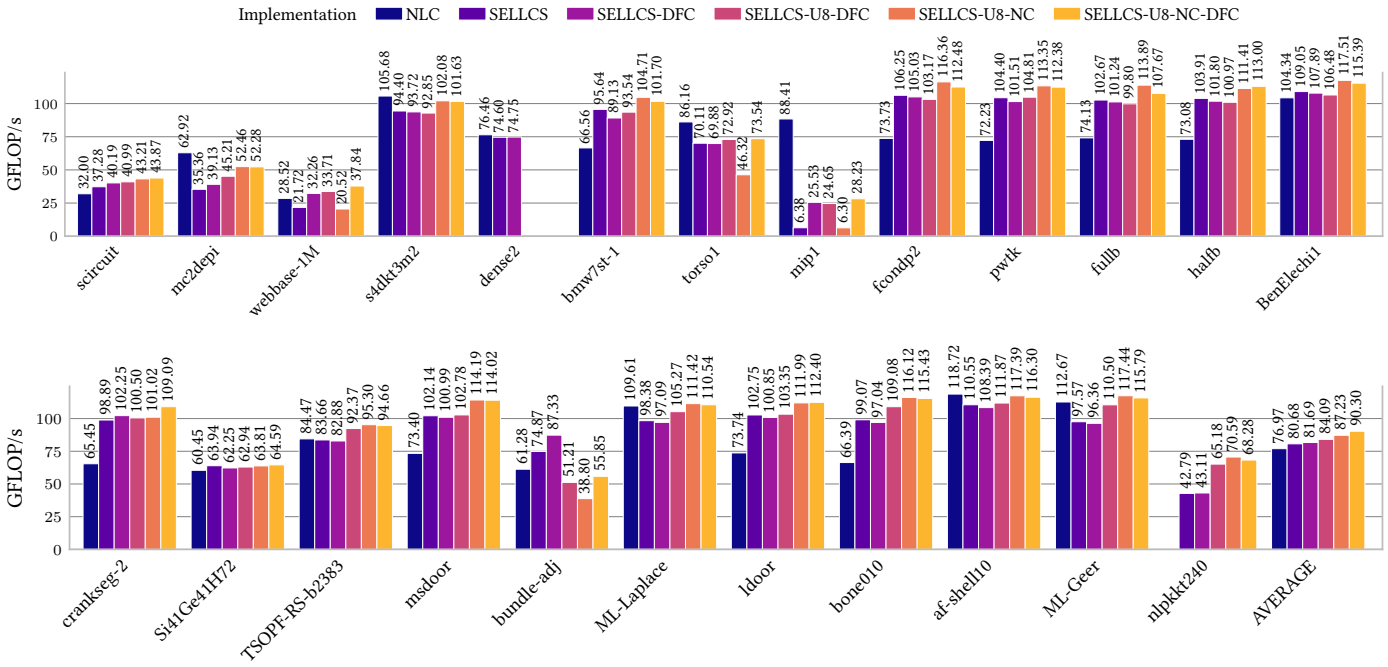

Fig. 1: Performance comparison of NLC vs our SELL-C-σ implementations for regular matrices.

In short, we can draw the following insight from Figure 1. The cache allocation and store relaxation policies obtain improvements ranging between 5% to 12%, in two thirds of the matrices, when using *SELLCS-U8-NC-DFC* compared to *SELLCS-U8-DFC*. Moreover, unrolling by 8 slices yields, in general, gives benefits ranging from 1% to 15%, with a favorable trend for bigger matrices. In the particular case of nlpkkt240 it brings a 51% performance increase. Finally, to understand the impact of the *DFC* optimization, we compare the performance of *SELLCS* with *SELLCS-DFC*. These two implementations only differ in the use of the *DFC* optimization. Only the second one includes it. On average, the overall performance gains of adapting each vector length instruction to the optimal size are almost negligible. However, it has a large impact in some scenarios. For example, when considering webbase-1M, which represents a website connectivity matrix and has a very low non-zero element density, *SELLCS-DFC* is 50% faster than *SELLCS*.

We obtain in average 90.3 GFLOPs across all matrices by enabling all optimizations, which constitutes a significant improvement of $\sim$12% and $\sim$17% compared to the baseline SELL-C-σ and NEC math library implementations, respectively. The significant performance increase that we obtain over the NEC proprietary software, which is specially tailored to SX-Aurora VE, demonstrates the relevance of our optimizations in long vector architectures.

D. Conclusion

In this work, we developed an implementation of SpMV for the SX-Aurora long vector architecture shows very competitive performance results which mostly overtake the highly optimized proprietary vendor implementation found in the NEC Library Collection. Additionally, we explore a set of optimizations targeting long-vector accelerators, and provide insight on how to exploit them in similar platforms.

II. Acknowledgment

This work has been published as a conference paper is published in the proceedings of the 26th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming[3].

References

[1] M. Kreutzer *et al.*, "A Unified Sparse Matrix Data Format for Efficient General Sparse Matrix-Vector Multiplication on Modern Processors with Wide SIMD Units," *SIAM Journal on Scientific Computing*, vol. 36, no. 5, pp. C401–C423, Jan. 2014.

[2] X. Liu *et al.*, "Efficient sparse matrix-vector multiplication on x86-based many-core processors," in *Proceedings of the 27th international ACM conference on International conference on supercomputing*. Eugene, Oregon, USA: Association for Computing Machinery, Jun. 2013, pp. 273–282.

[3] C. Gómez *et al.*, "Efficiently running spmv on long vector architectures," in *Proceedings of the 26th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, ser. PPoPP '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 292–303. [Online]. Available: https://doi.org/10.1145/3437801.3441592

Constantino Gómez is a last year Ph.D student at the Barcelona Supercomputing Center. He received the BSc and MSc degrees in Computer Science from the Universitat Politècnica de Catalunya (UPC) in 2014 and 2016. He has been involved as a researcher in the European Processor Initiative since the beginning. His research interests include long vector architectures and co-design for future massively parallel systems.