Load Balancing and Communication Bottlenecks in SpMV on the Cerebras WSE-3

Saho Orihara
ce255032@meiji.ac.jp
School of Science and Technology, Meiji University
Kawasaki-shi, Kanagawa, Japan

Kouki Sakai
ce255006@meiji.ac.jp
School of Science and Technology, Meiji University
Kawasaki-shi, Kanagawa, Japan

Reo Nagashima
ce265052@meiji.ac.jp
School of Science and Technology, Meiji University
Kawasaki-shi, Kanagawa, Japan

Takaaki Miyajima
takaaki.miyajima@cs.meiji.ac.jp
School of Science and Technology, Meiji University
Kawasaki-shi, Kanagawa, Japan

ABSTRACT

SpMV is a fundamental kernel in scientific computing, but its random memory access limits performance. The Cerebras CS-3, with 43 GB of on-chip SRAM, is a promising platform for this problem. That memory, however, is distributed across roughly 900,000 PEs, so how the non-zero elements are distributed determines performance. The simple 2D matrix partitioning used in our previous work left most cores underutilized because of skew in the non-zero elements [4]. We therefore introduce load balancing and evaluate both its effectiveness and the execution-time breakdown as the PE count grows. For Poisson3Da, load balancing achieved a 3.40x speedup, although the benefit diminishes as the PE count increases. For F1, inter-PE communication accounted for 71.2% of execution time. This work is a first step toward load balancing on the CS-3.

KEYWORDS

Cerebras CS-3, SpMV, Load Balancing, Communication Bottleneck

ACM Reference Format:
Saho Orihara, Kouki Sakai, Reo Nagashima, and Takaaki Miyajima. 2023. Load Balancing and Communication Bottlenecks in SpMV on the Cerebras WSE-3. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 2 pages. https://doi.org/XXXXXXX.XXXXXXX

1 CEREBRAS CS-3

The Cerebras CS-3 is built around the Wafer Scale Engine 3 (WSE-3), which uses an entire 300 mm wafer as a single chip, fabricated in a TSMC 5 nm process on 46,225 mm^2 of silicon. It integrates 893,064 Processing Elements (PEs) in a 762×1172 grid, and provides 43 GB of on-chip SRAM, 21 PB/s of memory bandwidth, and runs at 750 MHz [3]. Figure 1 illustrates the architecture of a single PE, consisting of a Compute Element, a 48 KB scratchpad memory, and a router that connects to its north, south, east, and west neighbors through a crossbar, forming a two-dimensional mesh.

Permission to make digital or hard copies of all or part 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, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
Conference'17, July 2017, Washington, DC, USA
© 2023 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/XXXXXXX.XXXXXXX

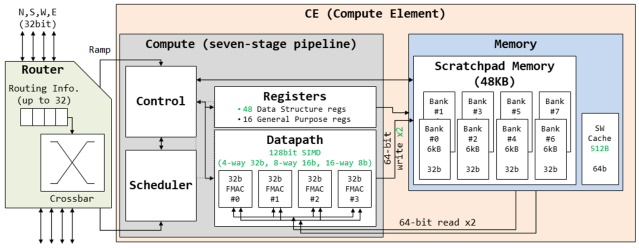


Figure 1: Architectural overview of a PE in the WSE-3.

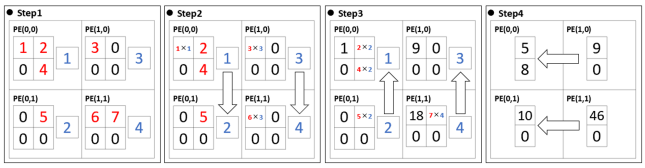


Figure 2: Computation algorithm for 2D SpMV on the CS-3.

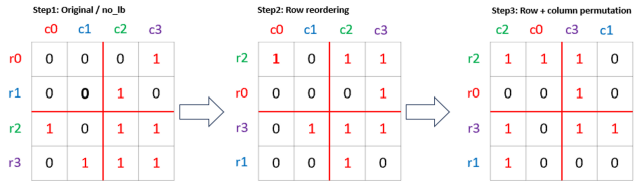


Figure 3: Load balancing algorithm based on greedy row/column permutation.

2 IMPLEMENTATION AND LOAD BALANCING OF SPMV

In our previous study on the CS-2 without load balancing, non-zero elements concentrated on a small number of PEs, which became the rate-limiting factor [4]. We therefore perform load balancing offline on the host CPU, and transfer the divided matrices and vectors to the CS-3. The SpMV computation, depicted in Figure 2, is identical to previous studies: the matrix and vector are mapped onto the PE grid, the vector is broadcast from the topmost PE, each PE computes its local SpMV, and results are aggregated leftward into the output

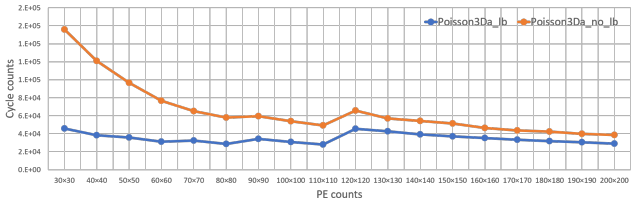


Figure 4: Processing cycle counts of Poisson3Da with and without load balancing.

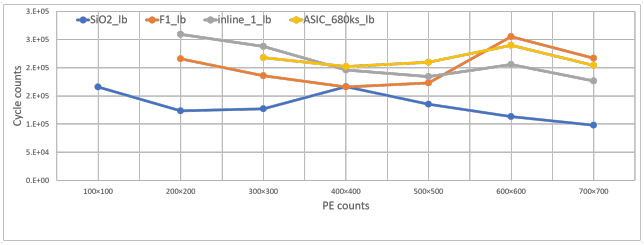


Figure 5: Strong scaling study of four matrices indicates there is a trade-off point.

vector. For load balancing, we implemented a greedy row- and column-permutation method in Python, following the approach used in Cerebras's official utility [1]. Rows and columns with a high number of non-zero elements (nnz) are iteratively assigned to lighter-loaded PEs until the maximum nnz per PE no longer improves, as shown in Figure 3. We evaluated the CS-3 on the ALCF AI Testbed (SDK v2.10.0) using five structurally diverse sparse matrices (FEM/structural, quantum chemistry, circuit simulation) from the SuiteSparse Matrix Collection [2], represented in FP32. We measured combinations of an $M \times M$ matrix and an $N \times N$ PE configuration. N was varied from 30 to 200 for the smaller Poisson3Da ($13{,}514^2$), and from 100 to 700 for the other matrices, within the executable range.

3 EVALUATION RESULTS

Load balancing effect: As shown in Figure 4, for Poisson3Da, load balancing achieved a 3.40x speedup over the non-load-balanced version (NLB) at the smallest PE configuration (30^2 PEs). This benefit diminished as the PE count increased, reflecting the relatively smaller impact of load imbalance at larger PE counts. For the other matrices, NLB exceeded the per-PE memory capacity for small PE configurations and could not be run.

Strong scaling: As shown in Figure 5, increasing the PE count did not monotonically reduce the cycle count. Instead, the cycle count peaked around $N \approx \sqrt{M}$ within the evaluated range. Table 1 shows the execution-time breakdown and reveals a trade-off: communication becomes dominant as the PE count increases. For the F1 matrix, communication reaches 71.2% of the total when using 600^2 PEs.

Bottleneck Analysis: In our implementation, the number of broadcasts reaches its maximum and the cycle count peaks around $N \approx \sqrt{M}$, where the vector chunk size—the number of elements sent by each PE in a single broadcast—reaches one. This can be explained by the over-partitioning of the vector, which results in excessive communication. A vector of length M is first partitioned across N PE columns and then across the N PEs within each column, giving a vector chunk size of approximately $\lceil M/N^2 \rceil$. Beyond $N \approx \sqrt{M}$, the chunk size cannot be reduced below one, while the number of vector elements assigned to each PE column, M/N, decreases as N increases, thereby reducing the number of broadcasts. For F1, $N \approx 586$, which closely matches the observed cycle-count peak.

Table 1: Execution-time breakdown for the F1 matrix.

N	Vector Chunk	Bcast Rounds	Compute [%]	Comm [%]	Reduction [%]	Sync [%]
200	9	191	35.0	29.3	6.2	29.5
300	4	287	21.4	49.6	8.4	20.6
400	3	287	18.3	55.7	8.9	17.1
500	2	344	16.2	60.6	8.4	14.9
600	1	573	12.1	71.2	8.2	8.5
700	1	492	12.4	69.6	9.4	8.6

4 SUMMARY

We introduced load balancing for SpMV on the Cerebras CS-3 and measured cycle counts with and without it. Load balancing achieves up to a 3.40x speedup, though its benefit diminishes as the PE count increases. We further show that as the PE count increases, there is a trade-off point where the bottleneck shifts from compute-bound to communication-bound. Future work will focus on designing an optimal communication scheme based on communication count and per-round data volume.

ACKNOWLEDGEMENT

This work was supported by Japan Science and Technology Agency (JST) as part of Adopting Sustainable Partnerships for Innovative Research Ecosystem (ASPIRE), Grant Number JPMJAP2341. This work was supported by JSPS KAKENHI Grant Number 24K14972. This research used resources of the Argonne Leadership Computing Facility, a U.S.Department of Energy (DOE) Office of Science user facility at Argonne National Laboratory and is based on research supported by the U.S. DOE Office of Science-Advanced Scientific Computing Research Program, under Contract No. DE-AC02-06CH11357.

REFERENCES

[1] Cerebras Systems. 2026. Hypersparse SpMV. Cerebras SDK Documentation, version 1.3.0. Accessed: 2026-08-05.
[2] Timothy A. Davis and Yifan Hu. 2011. The University of Florida Sparse Matrix Collection. *ACM Trans. Math. Softw.* 38, 1 (2011), 1–25. https://doi.org/10.1145/2049662.2049663
[3] Sean Lie. 2024. Wafer-Scale AI: GPU Impossible Performance. Presented at the 36th IEEE Hot Chips Symposium (Hot Chips 2024).
[4] Saho Orihara, Kouki Sakai, and Takaaki Miyajima. 2026. Towards Acceleration of Sparse Matrix-Vector Multiplication on the Cerebras CS-2. In *Poster Session, SCA/HPCAsia 2026: The International Conference on High Performance Computing in Asia-Pacific Region.* Osaka, Japan. Poster P-254.