

Fast Sparse Matrix Vector Multiplication with Highly-Compressed Sparse Format

Yusuke Nagasaka, Akira Nukada and Satoshi Matsuoka
Tokyo Institute of Technology

Introduction and Motivation

Many scientific problems involve computation on large sparse matrices. These matrices are stored in special sparse formats to reduce memory usage; various such formats have been proposed for specific architectures and matrix types. Sparse matrix vector multiplication (SpMV) is the most time consuming part in many solvers and the particular choice of sparse matrix format can largely affect the performance of SpMV. **Performance can be largely affected by the total amount of memory access including random memory access to input vector.** To reduce memory access we devise new sparse matrix format which requires less memory usage.

Proposal

Adaptive Multi-level Blocking (AMB) format

1st Level blocking : Segment the matrix along columns

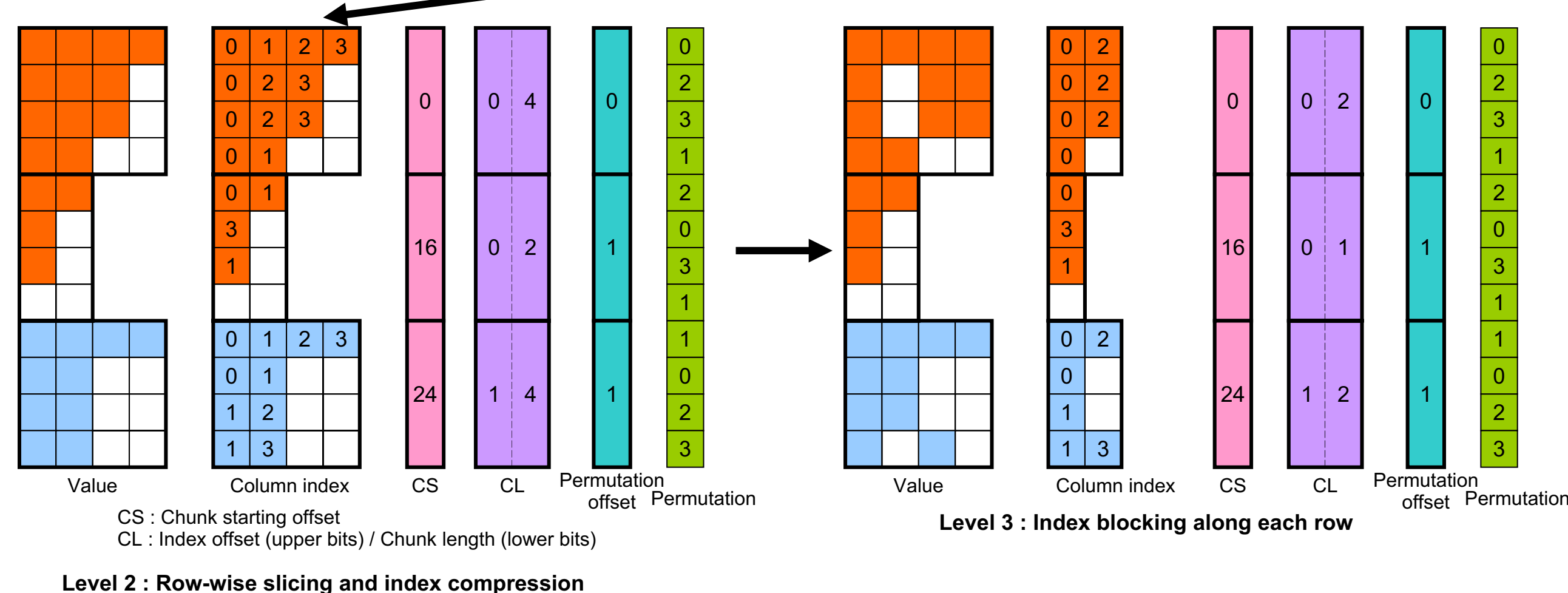
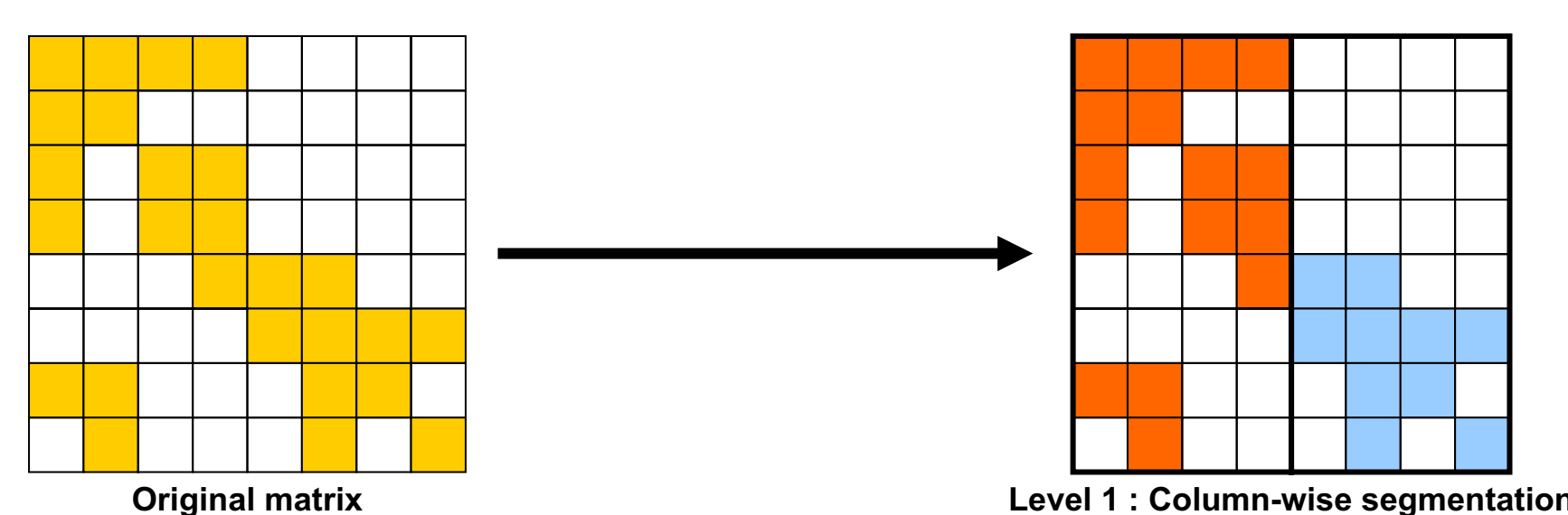
- Locality of the memory access to input vector is improved
- Segmentation width is limited to 64k columns

2nd Level blocking : Reducing memory usage with row-wise slice

- Each sub-matrix is converted to SELL-C- σ [1] (σ =sorting scope is also limited to 64k)
- Any Chunk which does not have non-zero elements is removed and **column index and permutation are represented in 16-bit integer (unsigned short)** to reduce memory usage.
- Offset of column index is stored to upper bits of cl (chunk length) array.
- In SpMV computation, the output vector is initialized and the result of each row is added using *atomicAdd* (in case of single precision)

3rd Level blocking : The column index is compressed by blocking along each row

- Block size : 1, 2, 3, 4, 5, 6, 7, 8
- This blocking largely reduces memory access.
- **Requires more zero filled for value array**



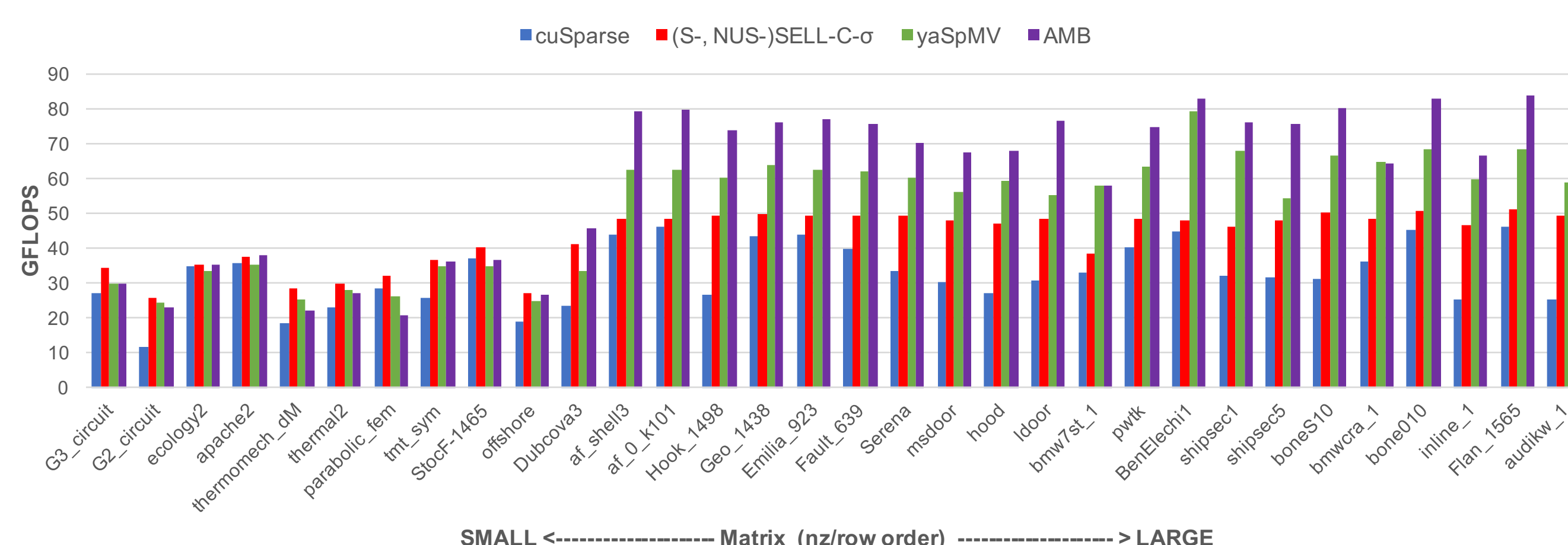
Performance Evaluation

- NVIDIA Tesla K20X GPU (ECC off) on TSUBAME KFC
- Matrix data is taken from the University of Florida Sparse Matrix Collection. 40 selected matrices are positive definite and row size is no less than 128k.
- Compared to cuSparse (CSR, HYBRID, BSR), SELL-C- σ , Segmented-SELL-C- σ , NUS-SELL-C- σ and yaSpMV [2] (BCCOO, BCCOO+)
- All SpMV computation is in single precision

FLOPS Performance

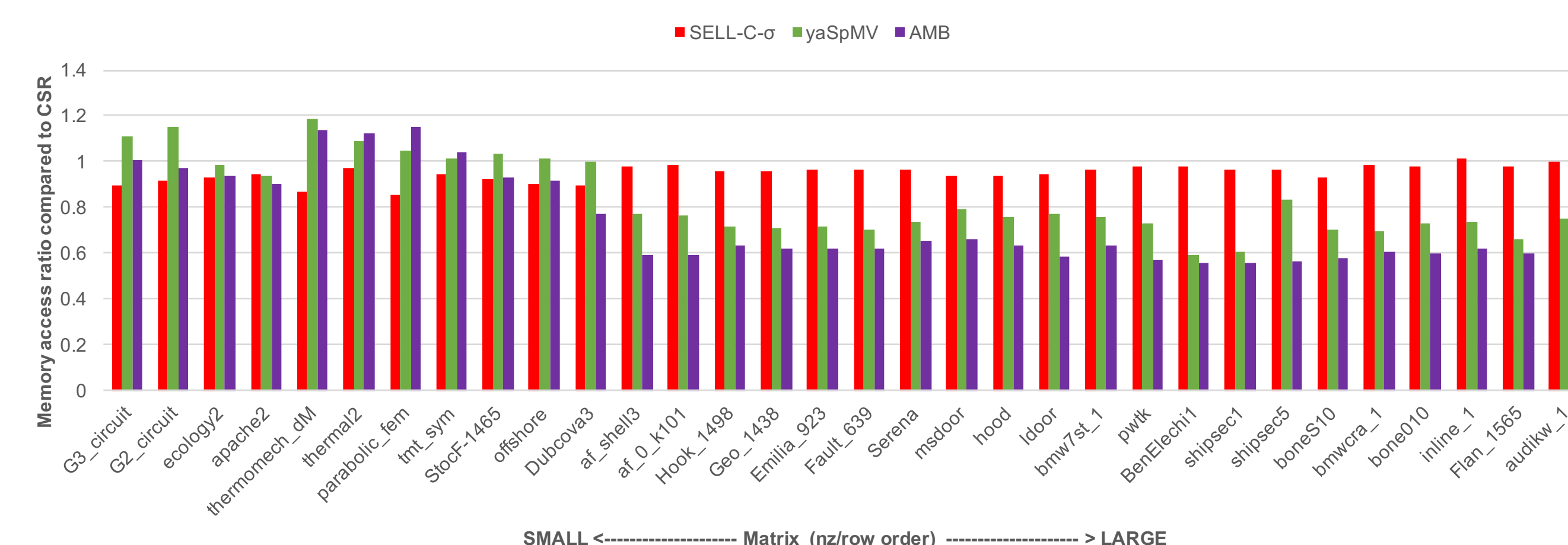
Speedup of AMB format is **x1.39** on maximum and **x1.13** on average compared to existing sparse format and library

- **x2.91** on maximum and **x1.81** on average compared to cuSparse
- **x1.39** on maximum and **x1.16** on average compared to yaSpMV



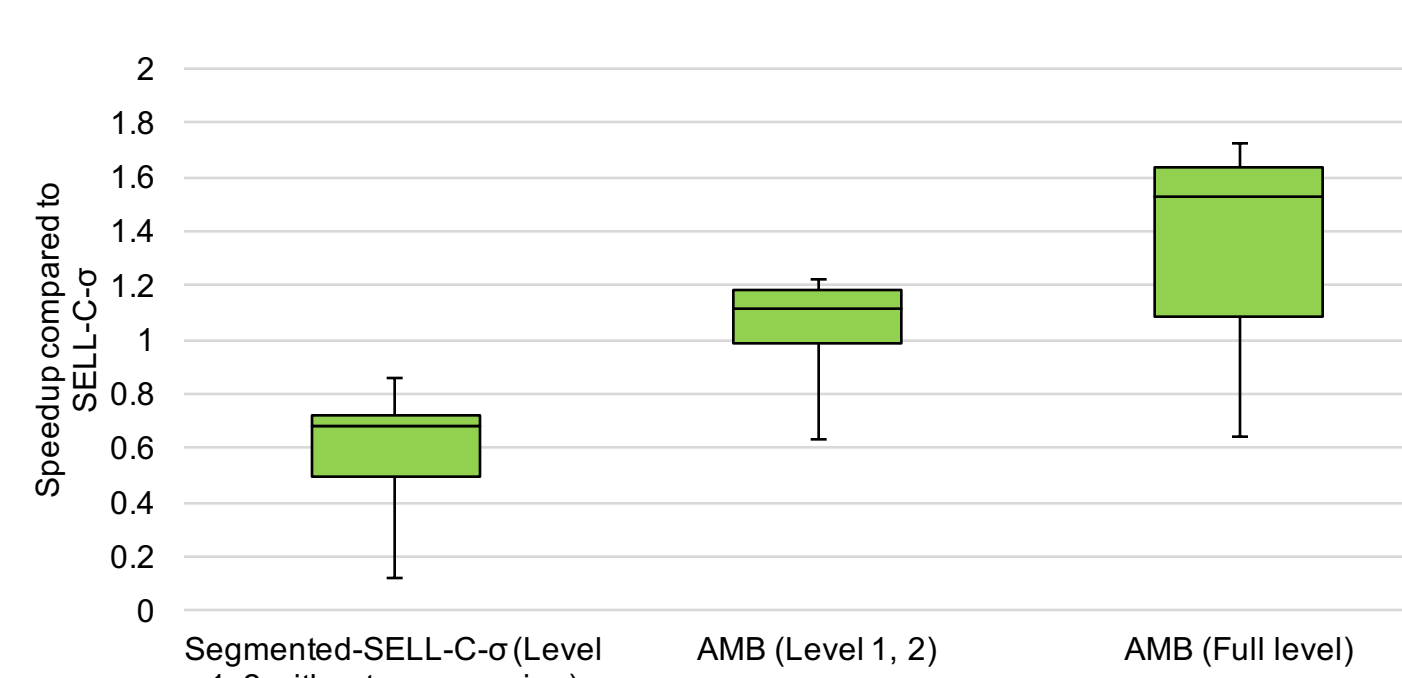
Memory Traffic

The traffic of GPU's global memory in SpMV computation is evaluated using nvprof. For most of the matrices, **the format which requires minimum traffic shows best SpMV performance.**



Performance Contribution

Speedups compared to SELL-C- σ show that **performance gains are the result of column index compression by using 16-bit integer and blocking.**



Conclusion

We proposed the novel sparse matrix formats which were designed for reducing the memory access in SpMV computation. AMB format achieves **x1.39** on maximum and **x1.13** on average performance gain compared to the existing fast SpMV library and other sparse matrix formats. We also found out that the performance of SpMV computation with AMB format strongly depends on the memory traffic.

Memory Traffic Estimation

The traffic of GPU's global memory in SpMV computation with AMB format can be estimated after format conversion.

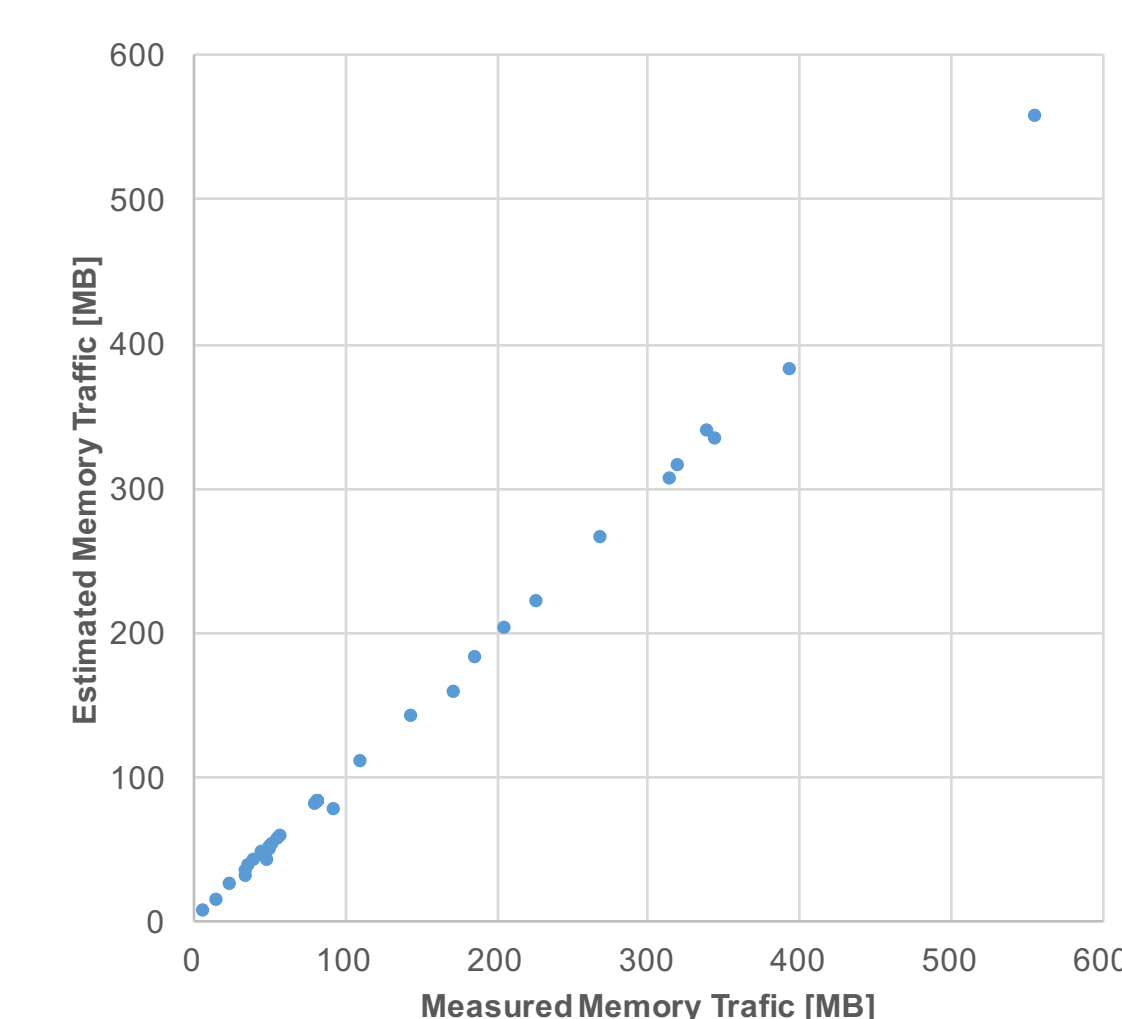
Variables

M : #row, N : #column, nz : #value elements after zero-fill
 b : block size, C : chunk size, nc : #chunk

- value + col : $nz * (4 + 2 / b)$
- CS + CL + permutation : $nc * 4 * 2 + nc * 2 + nc * C * 2$
- Input vector : $N * 4$
- Output vector : $M * 4 + nc * C * 4 * 2$
- ➔ Total : $nz * (4 + 2 / b) + nc * 10 * (1 + C) + (M + N) * 4$

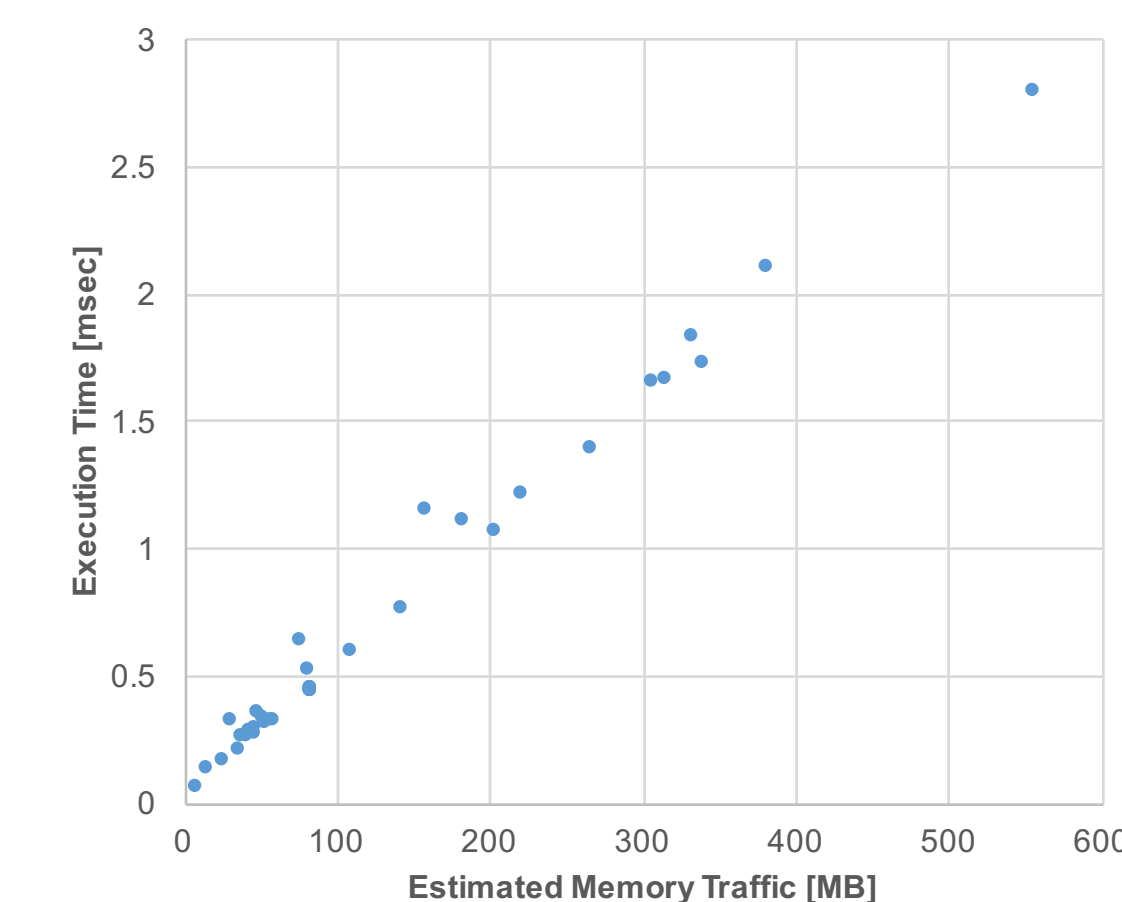
Comparison to Measured Memory Traffic

We evaluated 40 matrices using AMB format with best parameters. Result shows that **the estimation of memory traffic is highly accurate.**



Estimated Memory Traffic and Execution time

Basically **execution time depends on memory traffic.** **Load imbalance causes performance down** for some matrices.



Reference

- [1] Kreuzer et al., "A Unified Sparse Matrix Data Format for Modern Processors with Wide SIMD Units", CoRR, 2013
- [2] Schengen Yan et al. "yaSpMV : Yet Another SpMV Framework on GPUs", PPOPP'14