SMASH: Co-Designing Software Compression and Hardware-Accelerated Indexing for Efficient Sparse Matrix Operations

Konstantinos Kanellopoulos  Nandita Vijaykumar  Christina Giannoula  Roknoddin Azizi  Skanda Koppula  Nika Mansouri Ghiasi  Taha Shahroodi  Juan Gomez Luna  Onur Mutlu

1: Summary
- Many important workloads heavily involve Sparse Matrix Operations
- Extremely sparse matrices require compression to avoid storage and computational overheads
- Shortcomings of existing compression formats:
  - expensive discovery of the positions of non-zero elements
  - narrow applicability
- SMASH: hardware/software cooperative mechanism for efficient sparse matrix storage and computation
  - Software: Efficient compression scheme using a Hierarchy of Bitmaps
  - Hardware: Hardware unit interprets the Bitmap Hierarchy and accelerates indexing
- Performance improvement: 38% and 44% for SpMV and SpMM over the widely used CSR format
- SMASH is highly efficient, low-cost and widely applicable

2: Sparse Matrix Operations and Compression Formats

Sparse Matrix Operations are widespread today
- Recommender Systems
- Graph Analytics
- Neural Networks

Compressed Sparse Row Indexing overhead

Limitations of Existing Compression Formats
1. General formats optimize for storage → expensive discovery of the positions of non-zero elements
2. Specialized formats assume specific matrix structures and patterns (e.g., diagonals) → narrow applicability

3: SMASH

Hardware/Software cooperative mechanism: Explain SMASH
- Enables highly-efficient sparse matrix compression and computation
- General across a diverse set of sparse matrices and sparse matrix operations

4: Software Compression Scheme

Bitmap
- Encodes the presence/absence of a non-zero element in a block of the original matrix with a single bit
- Might contain high number of zero bits

Hierarchy of Bitmaps

5: Hardware Acceleration Unit

Bitmap Management Unit (BMU)
- Need for a cross-layer interface that enables software to control the BMU
- Communicate the parameters needed to calculate the index
- Query the BMU to retrieve the index of the next non-zero element

Enables SMASH to flexibly accelerate a diverse range of operations on any sparse matrix

6: Cross-Layer Interface

SMASH provides significant performance improvements over state-of-the-art formats

7: Use Case: SpMV

SMASH significantly reduces the number of executed instructions

8: Evaluation

Methodology
- Simulator: 2Sim Simulator
- Workloads:
  - Sparse Matrix Kernels: SpMV & SpMM from TACO
  - Graph Applications: PageRank & Betweenness Centrality from Lina
- Input dataset: 15 diverse sparse matrices & 4 graphs from the Sparse Suite Collection (4)
- Sparsity ranges from 0.01% to 8.79%

Performance Improvement using SMASH
- SMASH provides speedups regardless of the sparsity of the matrix

Hardware Overhead
- SMASH configuration:
  - Support for 4 matrices in the BMU
  - 256 bytes / SRAM buffer
  - 140 bytes for registers & counters
- 0.076% area overhead over an Intel Xeon CPU

Other Results in the Paper
- Compression ratio sensitivity analysis
- Distribution of non-zero elements
- Detailed results for SpMM
- Conversion from CSR to SMASH overhead
- Software-only approaches

Scan for full paper

SAFARI
ETH Zürich
Carnegie Mellon University