Compressed sparse tiles for memory-efficient unstructured and semi-structured sparsity

Mike Lasby[1], Max Zimmer[2], Sebastian Pokutta[2], Erik Schultheis[3,4]

Cooperation: [1]University of Calgary, [2]TU Berlin & Zuse Institute Berlin, [3]Aalto University, [3]IST Austria

Introduction

Setting: Storing the parameters of LLMs in GPU memory is challenging due to their size.
To remedy:

- *Quantization:* Store less bits per parameter.
- *Pruning:* Store fewer parameters by setting many to zero.

However: Unstructured sparsity is hard to exploit efficiently on modern hardware.

$\rightarrow$ Encoding the sparsity pattern introduces memory overhead.
$\rightarrow$ Enforcing structure in the sparsity is desirable, but leads to performance degradation.

Goal: Find a format that offers a good balance between memory efficiency, hardware acceleration and model performance.

Compressed Sparse Tiles

- *Coordinate format (COO):* Each non-pruned weight is identified by row index, column index, and weight value $\rightarrow$ weights are three times as expensive as in dense format
- *Compressed Sparse Row (CSR):* Performs localization of weights hierarchically - first, group together by coarse location (row), then by finer location (column).

Observations:

- The inner locator of CSR still needs 16 bits $\rightarrow$ reduce size of regions such that 8 bits are sufficient by replacing pointers to row starts with pointers to 256-tile starts
- **Problem:** We have traded less bits per weight for a larger number of pointers $\rightarrow$ inefficient at high sparsity, since we store more pointers than weights
- **Idea:** Group tiles into larger super-tiles (32 bit), use smaller-sized offsets to identify the within-super-tile position.

Overview of CS256:

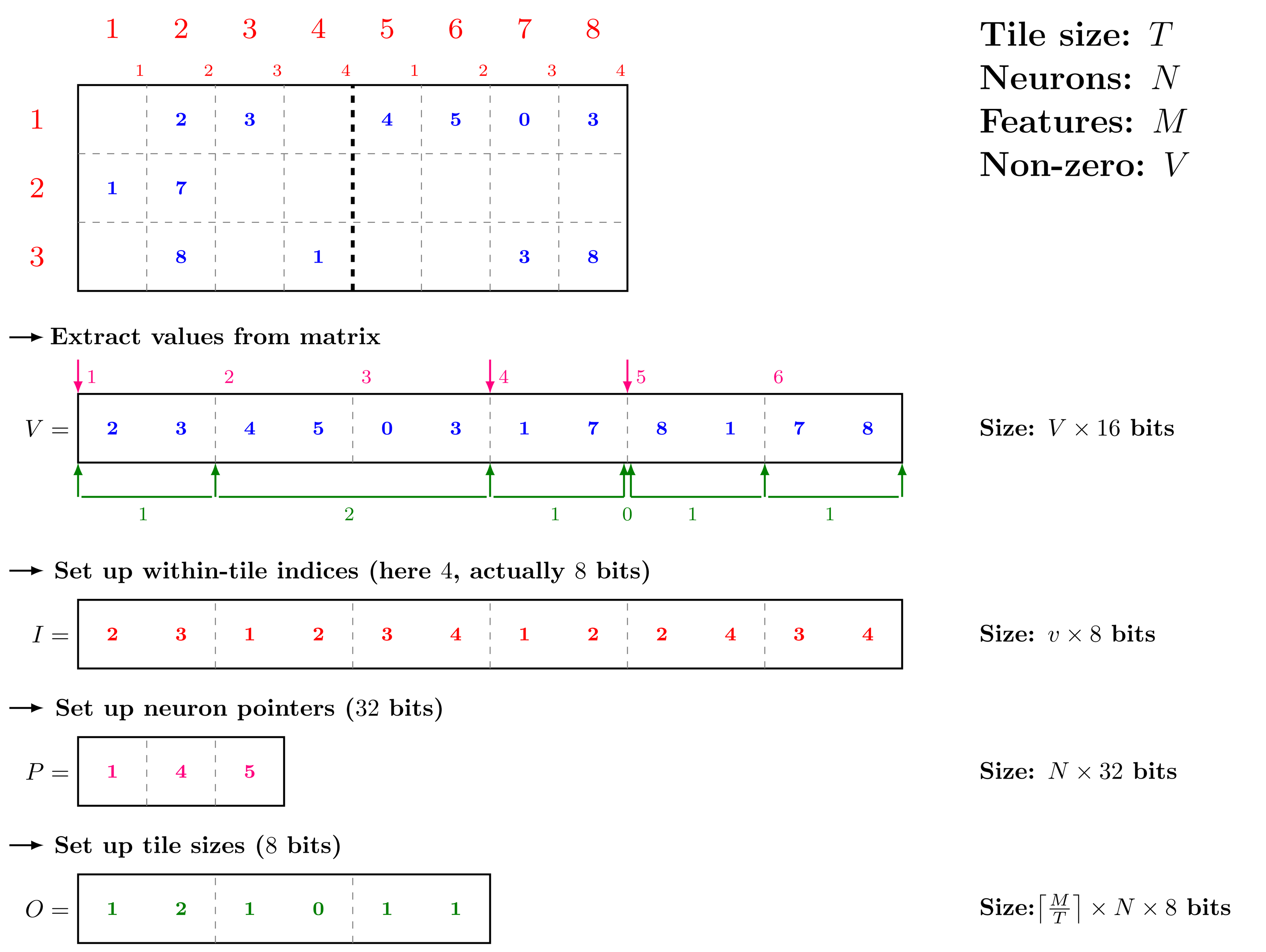

Figure 1: Visualization of the CS256 format, using a smaller tile size of 4 for illustration.

Tradeoffs of different formats

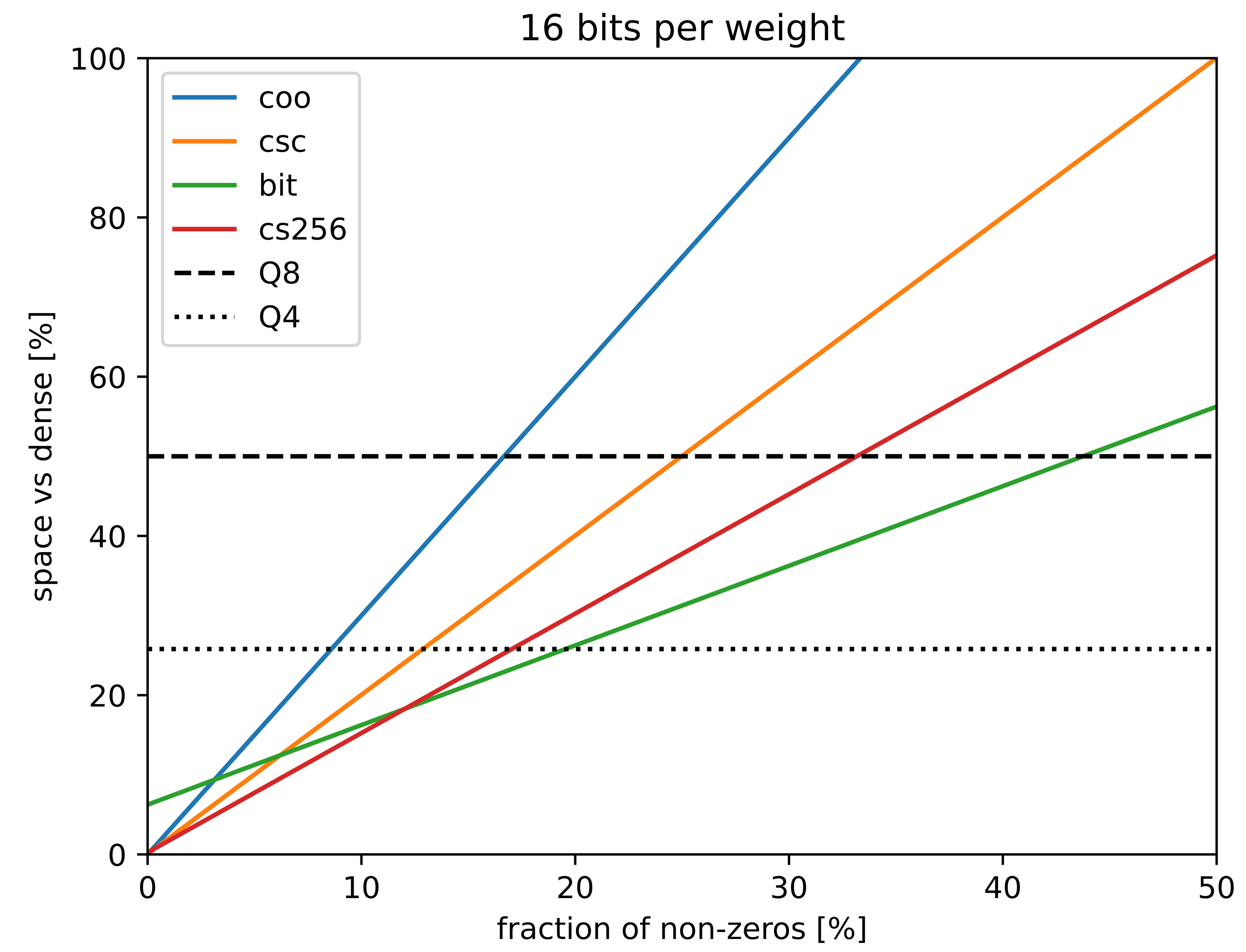

Figure 2: Compression ratio of several sparse storage formats at 16 bit weight precision, compared with 8 and 4 bit quantization (assuming effectively 4.127 bits per parameter (Dettmers et al., 2024)).

To achieve 50% memory reduction:

- *Coordinate format (COO):* Requires sparsity of 83.5%.
- *Compressed Sparse Column (CSC):* Requires sparsity of 75%.
- *CS256:* Requires sparsity of $\approx$ 70%.

CS256:

- Offers better balance between space efficiency and hardware acceleration.
- Is essentially unstructured, but with less overhead than CSC.

Experimental Results

Setup:

- We extend SparseGPT to our format.
- Model: Llama-3.1-8B-Instruct
- Calibration dataset: 512 samples from UltraChat-200K
- Evaluation dataset: 25-shot ARC-C

Results:

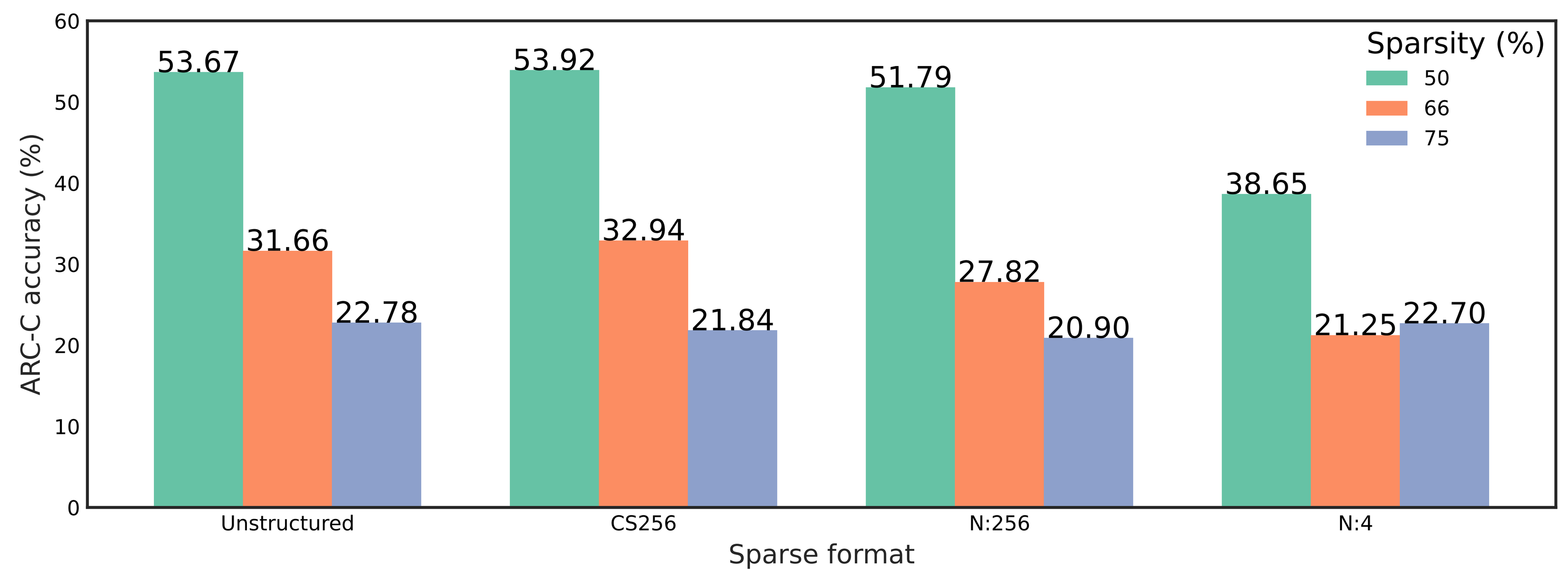

Figure 3: **ARC-C accuracy for various sparsities and formats.** CS256 matches unstructured sparsity. For N:M, $M = 256$ yields considerable improvements over the 1:4, 2:6, and 2:4 formats. As most questions in the ARC-C corpus have 4 choices, the 75% sparse models and 2:6 formats fail to exceed chance accuracy.