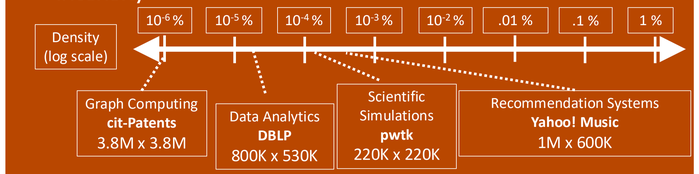
Tailors:
Accelerating Sparse Tensor Algebra by Overbooking Buffer Capacity

Zi Yu Xue*, Yannan Nellie Wu*, Joel S. Emer*^, Vivienne Sze*

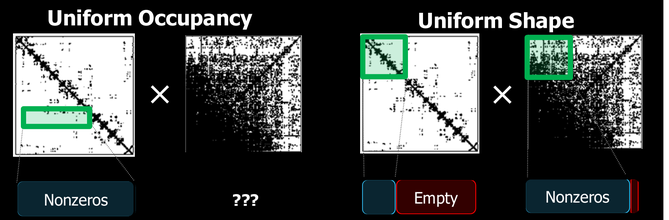
MIT*, NVIDIA^

Sparse Tensors are Large and Highly Sparse

- Tensor computation relies on tiling to improve data reuse and arithmetic intensity. **Larger tiles maximize data reuse.**
- Operations on sparse tensors, particularly multiple sparse operands, are especially challenging to tile effectively as they have further reduced arithmetic intensity



Current Tiling Approaches are Insufficient



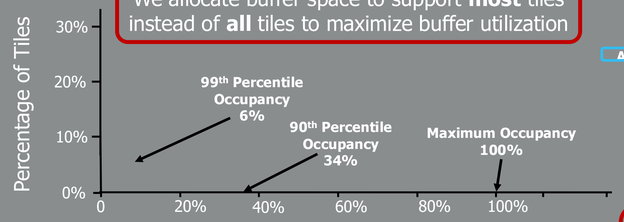
- Same number of nonzeros per tile
  ⟹ Ideal buffer utilization
- Varying coordinate range in second sparse operand
  ⟹ Hard to tile second operand

- All tiles must fit in buffer
  ⟹ Low buffer utilization
- Fixed coordinate ranges
  ⟹ Easy to tile both operands

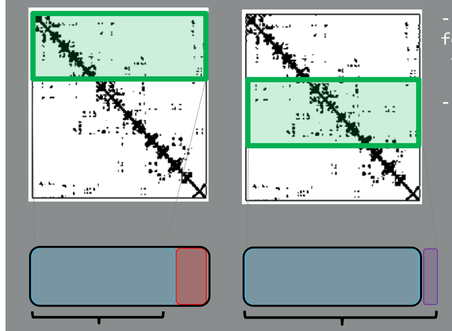
Opportunities for Overbooking

Tiles in Sparse Tensors Vary in Sparsity

We allocate buffer space to support **most** tiles instead of **all** tiles to maximize buffer utilization



Overbooking-based Coordinate-space Tiling



```
----- DRAM -----
for m1 in [0,2):
 for k1 in [0,2):
  for n1 in [0,2):
----- buffer -----
   for m0 in [0,4):
    for k0 in [0,4):
     for n0 in [0,4):
      Z[m1*4+m0,
        n1*4+n0] =
          A[m1*4+m0,
            k1*4+k0]*
          B[k1*4+k0,
            n1*4+n0]
```

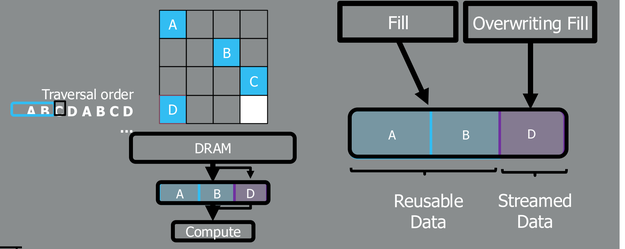
Larger tile size makes buffer more filled on average

Occasionally, tile occupancy will bump data ("**overbook**")

Example:
Larger M0 (larger A, Z tiles)
$\Longrightarrow \frac{M0^{new}}{M0}$ fewer loads of B tiles

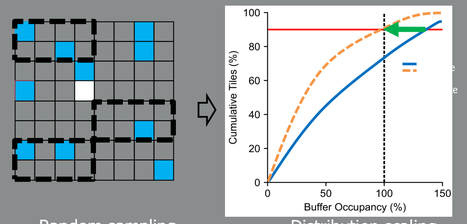
As long as the cost of recovering from bumped data is less than the benefit from larger tiles, overbooking is beneficial

Tailors: Tail Overbooked Buffers



When tile is overbooked, divide the buffer to stream data through without losing data reuse
Hardware support is **low cost** and **low complexity**, easy to integrate into existing accelerators

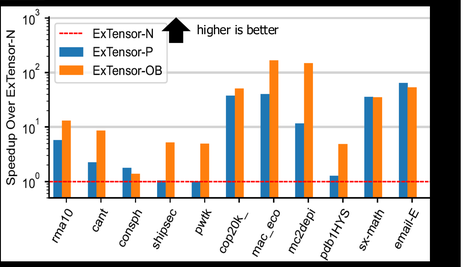
Swiftiles: Swift Tiling with Overbooking



Overbooking enables tile size *estimation* and do not need the exact tile size since tiles which do not fit entirely in buffer are still supported by the hardware

Minimize preprocessing by sampling tile occupancy distribution and scaling to buffer size

Results



52.7x speedup over ExTensor without tensor occupancy knowledge
2.3x speedup over ExTensor with perfect tensor occupancy knowledge

Conclusion

Tiling is key to improving data reuse and thus reducing memory traffic for sparse tensor algebra applications. We balance the tiling strategy's adaptability and efficiency by overbooking. We support overbooking in hardware with Tailors and speculatively tile with Swiftiles

Acknowledgements

MIT AI Hardware Program
NSERC PGS-D