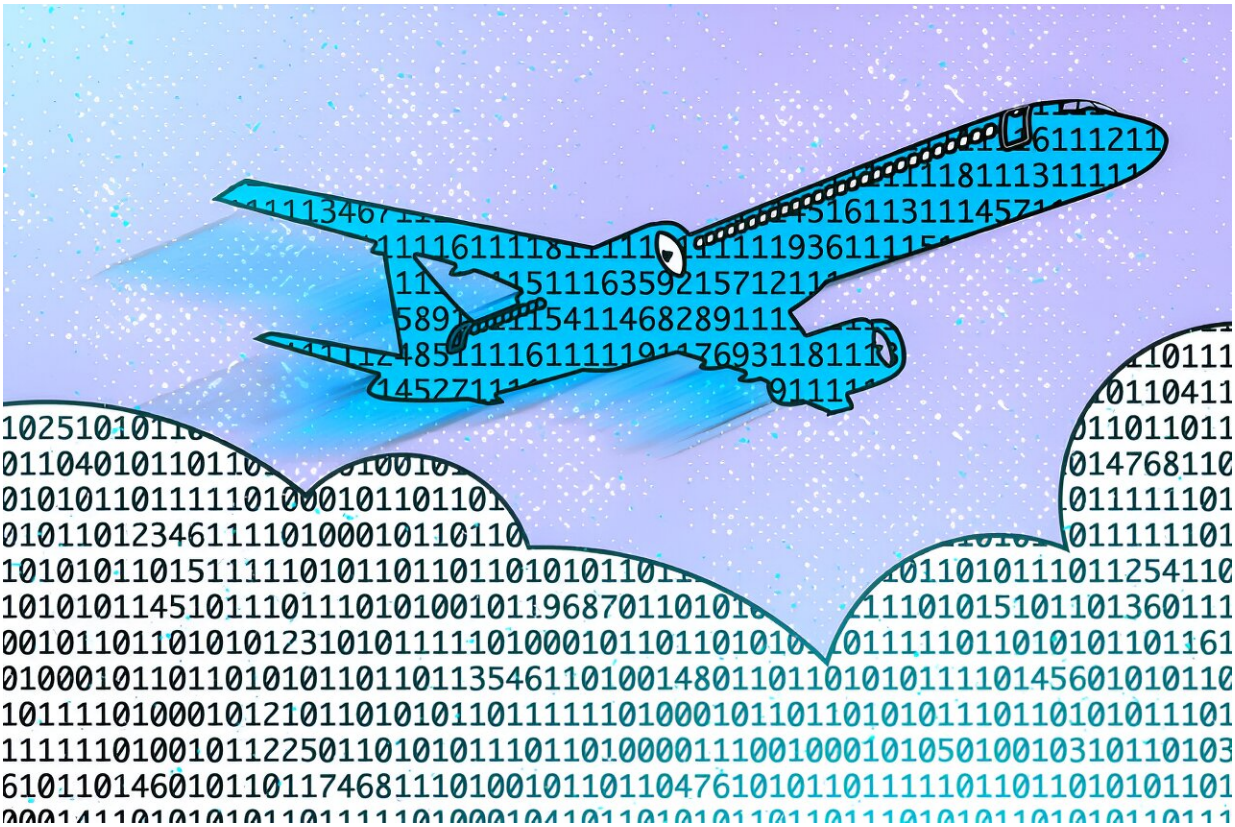
New techniques efficiently accelerate sparse tensors for massive AI models

October 30 2023, by Adam Zewe



Researchers from MIT and NVIDIA developed two complementary techniques that could dramatically boost the speed and performance of high-performance computing applications like graph analytics or generative AI. Both of the new methods seek to efficiently exploit sparsity—zero values—in the tensors. Credit: Image: Jose-Luis Olivares, MIT

Researchers from MIT and NVIDIA have developed two techniques that accelerate the processing of sparse tensors, a type of data structure that's used for high-performance computing tasks. The complementary techniques could result in significant improvements to the performance and energy-efficiency of systems like the massive machine-learning models that drive generative artificial intelligence.

Tensors are data structures used by machine-learning models. Both of the new methods seek to efficiently exploit what's known as sparsity—zero values—in the tensors. When processing these tensors, one can skip over the zeros and save on both computation and memory. For instance, anything multiplied by zero is zero, so it can skip that operation. And it can compress the tensor (zeros don't need to be stored) so a larger portion can be stored in on-chip memory.

However, there are several challenges to exploiting sparsity. Finding the nonzero values in a large tensor is no easy task. Existing approaches often limit the locations of nonzero values by enforcing a sparsity pattern to simplify the search, but this limits the variety of sparse tensors that can be processed efficiently.

Another challenge is that the number of nonzero values can vary in different regions of the tensor. This makes it difficult to determine how much space is required to store different regions in memory. To make sure the region fits, more space is often allocated than is needed, causing the storage buffer to be underutilized. This increases off-chip memory traffic, which requires extra computation.

The MIT and NVIDIA researchers crafted two solutions to address these problems. For one, they developed a technique that allows the hardware to efficiently find the nonzero values for a wider variety of sparsity patterns.

For the other solution, they created a method that can handle the case where the data do not fit in memory, which increases the utilization of the storage buffer and reduces off-chip memory traffic.

Both methods boost the performance and reduce the energy demands of hardware accelerators specifically designed to speed up the processing of sparse tensors. The papers have been posted to the *arXiv* preprint server.

"Typically, when you use more specialized or domain-specific hardware accelerators, you lose the flexibility that you would get from a more general-purpose processor, like a CPU. What stands out with these two works is that we show that you can still maintain flexibility and adaptability while being specialized and efficient," says Vivienne Sze, associate professor in the MIT Department of Electrical Engineering and Computer Science (EECS), a member of the Research Laboratory of Electronics (RLE), and co-senior author of papers on both advances.

Her co-authors include lead authors Yannan Nellie Wu Ph.D. '23 and Zi Yu Xue, an electrical engineering and computer science graduate student; and co-senior author Joel Emer, an MIT professor of the practice in computer science and electrical engineering and a member of the Computer Science and Artificial Intelligence Laboratory (CSAIL), as well as others at NVIDIA. Both papers will be presented at the IEEE/ACM International Symposium on Microarchitecture.

HighLight: Efficiently finding zero values

Sparsity can arise in the tensor for a variety of reasons. For example, researchers sometimes "prune" unnecessary pieces of the machine-learning models by replacing some values in the tensor with zeros, creating sparsity. The degree of sparsity (percentage of zeros) and the locations of the zeros can vary for different models.

To make it easier to find the remaining nonzero values in a model with billions of individual values, researchers often restrict the location of the nonzero values so they fall into a certain pattern. However, each hardware accelerator is typically designed to support one specific sparsity pattern, limiting its flexibility.

By contrast, the hardware accelerator the MIT researchers designed, called HighLight, can handle a wide variety of sparsity patterns and still perform well when running models that don't have any zero values.

They use a technique they call "hierarchical structured sparsity" to efficiently represent a wide variety of sparsity patterns that are composed of several simple sparsity patterns. This approach divides the values in a tensor into smaller blocks, where each block has its own simple, sparsity pattern (perhaps two zeros and two nonzeros in a block with four values).

Then, they combine the blocks into a hierarchy, where each collection of blocks also has its own simple, sparsity pattern (perhaps one zero block and three nonzero blocks in a level with four blocks). They continue combining blocks into larger levels, but the patterns remain simple at each step.

This simplicity enables HighLight to more efficiently find and skip zeros, so it can take full advantage of the opportunity to cut excess computation. On average, their accelerator design was about six times more energy-efficient than other approaches.

"In the end, the HighLight accelerator is able to efficiently accelerate dense models because it does not introduce a lot of overhead, and at the same time it is able to exploit workloads with different amounts of zero values based on hierarchical structured sparsity," Wu explains.

In the future, she and her collaborators want to apply hierarchical structured sparsity to more types of machine-learning models and different types of tensors in the models.

Tailors and Swiftiles: Effectively 'overbooking' to accelerate workloads

Researchers can also leverage sparsity to more efficiently move and process data on a computer chip.

Since the tensors are often larger than what can be stored in the memory buffer on chip, the chip only grabs and processes a chunk of the tensor at a time. The chunks are called tiles.

To maximize the utilization of that buffer and limit the number of times the chip must access off-chip memory, which often dominates energy consumption and limits processing speed, researchers seek to use the largest tile that will fit into the buffer.

But in a sparse tensor, many of the data values are zero, so an even larger tile can fit into the buffer than one might expect based on its capacity. Zero values don't need to be stored.

But the number of zero values can vary across different regions of the tensor, so they can also vary for each tile. This makes it difficult to determine a tile size that will fit in the buffer. As a result, existing approaches often conservatively assume there are no zeros and end up selecting a smaller tile, which results in wasted blank spaces in the buffer.

To address this uncertainty, the researchers propose the use of "overbooking" to allow them to increase the tile size, as well as a way to

tolerate it if the tile doesn't fit the buffer.

The same way an airline overbooks tickets for a flight, if all the passengers show up, the airline must compensate the ones who are bumped from the plane. But usually all the passengers don't show up.

In a sparse tensor, a tile size can be chosen such that usually the tiles will have enough zeros that most still fit into the buffer. But occasionally, a tile will have more nonzero values than will fit. In this case, those data are bumped out of the buffer.

The researchers enable the hardware to only re-fetch the bumped data without grabbing and processing the entire tile again. They modify the "tail end" of the buffer to handle this, hence the name of this technique, Tailors.

Then they also created an approach for finding the size for tiles that takes advantage of overbooking. This method, called Swiftiles, swiftly estimates the ideal tile size so that a specific percentage of tiles, set by the user, are overbooked. (The names "Tailors" and "Swiftiles" pay homage to Taylor Swift, whose recent Eras tour was fraught with overbooked presale codes for tickets).

Swiftiles reduces the number of times the hardware needs to check the tensor to identify an ideal tile size, saving on computation. The combination of Tailors and Swiftiles more than doubles the speed while requiring only half the energy demands of existing hardware accelerators which cannot handle overbooking.

"Swiftiles allows us to estimate how large these tiles need to be without requiring multiple iterations to refine the estimate. This only works because overbooking is supported. Even if you are off by a decent amount, you can still extract a fair bit of speedup because of the way the

non-zeros are distributed," Xue says.

In the future, the researchers want to apply the idea of overbooking to other aspects in computer architecture and also work to improve the process for estimating the optimal level of overbooking.

More information: Zi Yu Xue et al, Tailors: Accelerating Sparse Tensor Algebra by Overbooking Buffer Capacity, *arXiv* (2023). DOI: 10.48550/arxiv.2310.00192

Yannan Nellie Wu et al, HighLight: Efficient and Flexible DNN Acceleration with Hierarchical Structured Sparsity, *arXiv* (2023). DOI: 10.48550/arxiv.2305.12718

Provided by Massachusetts Institute of Technology