A tensor language prototype for high-performance computers

February 15 2022, by Steve Nadis

A new tensor language developed at MIT, with formally verified optimizations, could have benefits for high-performance computing. Credit: Massachusetts Institute of Technology

High-performance computing is needed for an ever-growing number of tasks—such as image processing or various deep learning applications on neural nets—where one must plow through immense piles of data, and do so reasonably quickly, or else it could take ridiculous amounts of time. It's widely believed that, in carrying out operations of this sort, there are unavoidable trade-offs between speed and reliability. If speed is the top priority, according to this view, then reliability will likely suffer, and vice versa.

However, a team of researchers, based mainly at MIT, is calling that notion into question, claiming that one can, in fact, have it all. With the new programming language, which they've written specifically for high-performance computing, says Amanda Liu, a second-year Ph.D. student at the MIT Computer Science and Artificial Intelligence Laboratory (CSAIL), "speed and correctness do not have to compete. Instead, they can go together, hand-in-hand, in the programs we write."

Liu—along with University of California at Berkeley postdoc Gilbert Louis Bernstein, MIT Associate Professor Adam Chlipala, and MIT Assistant Professor Jonathan Ragan-Kelley—described the potential of their recently developed creation, "A Tensor Language" (ATL), last month at the Principles of Programming Languages conference in Philadelphia.

"Everything in our language," Liu says, "is aimed at producing either a single number or a tensor." Tensors, in turn, are generalizations of vectors and matrices. Whereas vectors are one-dimensional objects (often represented by individual arrows) and matrices are familiar two-dimensional arrays of numbers, tensors are n-dimensional arrays, which could take the form of a 3x3x3 array, for instance, or something of even higher (or lower) dimensions.

The whole point of a computer algorithm or program is to initiate a particular computation. But there can be many different ways of writing that program—"a bewildering variety of different code realizations," as Liu and her coauthors wrote in their soon-to-be published conference paper—some considerably speedier than others. The primary rationale behind ATL is this, she explains: "Given that high-performance computing is so resource-intensive, you want to be able to modify, or rewrite, programs into an optimal form in order to speed things up. One often starts with a program that is easiest to write, but that may not be the fastest way to run it, so that further adjustments are still needed."

As an example, suppose an image is represented by a 100x100 array of numbers, each corresponding to a pixel, and you want to get an average value for these numbers. That could be done in a two-stage computation by first determining the average of each row and then getting the average of each column. ATL has an associated toolkit—what computer scientists call a "framework"—that might show how this two-step process could be converted into a faster one-step process.

"We can guarantee that this optimization is correct by using something called a proof assistant," Liu says. Toward this end, the team's new language builds upon an existing language, Coq, which contains a proof assistant. The proof assistant, in turn, has the inherent capacity to prove its assertions in a mathematically rigorous fashion.

Coq had another intrinsic feature that made it attractive to the MIT-based group: programs written in it, or adaptations of it, always terminate and cannot run forever on endless loops (as can happen with programs written in Java, for example). "We run a program to get a single answer—a number or a tensor," Liu maintains. "A program that never terminates would be useless to us, but termination is something we get for free by making use of Coq."

The ATL project combines two of the main research interests of Ragan-Kelley and Chlipala. Ragan-Kelley has long been concerned with the optimization of algorithms in the context of high-performance computing. Chlipala, meanwhile, has focused more on the formal (as in mathematically-based) verification of algorithmic optimizations. This represents their first collaboration. Bernstein and Liu were brought into the enterprise last year, and ATL is the result.

It now stands as the first, and so far the only, tensor language with formally verified optimizations. Liu cautions, however, that ATL is still just a prototype—albeit a promising one—that's been tested on a number of small programs. "One of our main goals, looking ahead, is to improve the scalability of ATL, so that it can be used for the larger programs we see in the real world," she says.

In the past, optimizations of these programs have typically been done by hand, on a much more ad hoc basis, which often involves trial and error, and sometimes a good deal of error. With ATL, Liu adds, "people will be able to follow a much more principled approach to rewriting these programs—and do so with greater ease and greater assurance of correctness."

The research was published in *Proceedings of the ACM on Programming Languages*.

More information: Amanda Liu et al, Verified tensor-program optimization via high-level scheduling rewrites, *Proceedings of the ACM on Programming Languages* (2022). DOI: 10.1145/3498717

Provided by Massachusetts Institute of Technology

Citation: A tensor language prototype for high-performance computers (2022, February 15) retrieved 4 May 2024 from

https://techxplore.com/news/2022-02-tensor-language-prototype-high-performance.html