Performance Accessibility Through Distributed Array Programming

Rohan Yadav

Stanford University

I research programming languages, with a focus on compilers and programming systems that enable the achievement of high performance on modern computing systems. The transition from single-processor machines to distributed and accelerated systems has made developing high performance software incredibly challenging, and often out of reach for end users building real applications that target these systems. My overarching research vision is to close this gap in *performance accessibility*. I aim to make advances in programming language techniques to create systems and libraries that enable everyday programmers to realize the peak performance available on modern distributed and accelerated computing systems.

A key component of distributed programming is the expression of parallel array computations at scale on both dense and sparse arrays. Dense and sparse array computations are key building blocks of many modern workloads, including applications in machine learning, data analytics, and scientific computing. Easy-to-use systems for distributed dense and sparse array programming that deliver high performance are a key component of my vision for performance accessible computing. I believe that it is possible to build high-level distributed array programming systems that provide high degrees of flexibility around data representation and computation while providing speed-of-light performance on clusters of CPUs and GPUs. This class of systems have the potential to make high performance distributed programming a reality for programmers without deep parallel programming expertise, directly enabling new science and data analysis to be performed.

Despite the potential benefits of distributed array programming systems, their development has been hindered by at least two major technical challenges that cut across the software stack. The first key challenge is in the development and optimization of distributed computational kernels within an array programming system, as these require the skills of an expert distributed programmer. The second key challenge is the prospect of building an easy-to-use system employing these kernels that is efficient, composable and automated.

I have spent the initial two years of my PhD at Stanford University addressing an aspect of the first key challenge, the development of distributed computational kernels. In particular, I focused on the description and optimization of dense and sparse tensor algebra, which are performance-defining array programming operations in modern workloads. Even specific instances of tensor algebra computations are difficult for experts to achieve peak performance on, and the full generality of tensor algebra magnifies this challenge. As a result, modern high-performance tensor algebra libraries contain a small but heavily optimized subset of operations, and end users must fit their desired operations into this set. To address these limitations, I developed a novel compiler called DISTAL [1] which enables high-level descriptions of both dense tensor algebra computations and the way that these computations should be mapped onto a distributed and accelerated machine. I then extended DISTAL to reason about sparse tensors [2], resulting in the first compiler that enables the efficient and distributed implementation of general dense and sparse tensor algebra programs.

The key idea of DISTAL is to separate the description of computation from the descriptions of how to map the computation and data onto the target machine through independent high-level languages. This separation enables the rapid iteration of optimization strategies, at both the distributed and kernel level, to occur without compromising the correctness of the resulting program. In contrast to the state of the art, many optimization decisions that would result in dramatic changes to source code are expressed as concise correctness preserving transformations in DISTAL. At the same time, DISTAL generates code that is competitive with kernels in hand-tuned systems such as ScaLAPACK, PETSc and Trilinos while supporting more expressions and data structures than each of these systems. My work on DISTAL has the potential to change the way that optimized distributed libraries are built by providing expert developers efficient ways to experiment with and discover new optimizations. My work also impacts end users by enabling the

development of more powerful tensor algebra libraries, and brings the possibility of writing high performance tensor algebra programs closer to the hands of end users.

Armed with techniques to develop high performance tensor algebra kernels, I am taking a first step towards developing a high-level distributed array programming library through an internship at NVIDIA Research with Michael Bauer and Michael Garland. We are developing a prototype library called `legate.sparse`, which is a distributed and accelerated drop-in replacement for the widely used `scipy.sparse` Python package for sparse matrix computations. `legate.sparse` enables programmers to scale programs written with `scipy.sparse` to clusters of GPUs, putting high performance in the hands of domain scientists. `legate.sparse` heavily outperforms `scipy.sparse` through parallel execution, delivers similar performance as `CuPy` on a single GPU, and is competitive with established distributed systems such as PETSc. I utilized DISTAL to implement a large percentage of the `scipy.sparse` interface, targeting clusters of both CPUs and GPUs within a 12-week period. Without the techniques I developed in DISTAL, such an effort would have taken much longer as each kernel would need to be individually developed and optimized for each architecture backend.

The initial success of `legate.sparse` paves the way for future work to generalize the static, matrix-based library to a fully featured array programming library that supports user-defined computations on arrays of any dimension and sparse data structure. Supporting this sort of generality has many technical challenges, yielding future research directions. One major challenge in supporting user-defined computations of tensors of any dimensions and sparse data structure requires utilizing DISTAL as a just-in-time compiler instead of as ahead-of-time compilers in `legate.sparse`. Systems that provide a pre-defined set of optimized kernels are inflexible and can incur significant performance penalties through data and computation reorganization to match the pre-defined set. Ideally, a flexible system would utilize DISTAL to generate kernels as needed for user-defined computations, generating fused and specialized code. The integration of DISTAL as a just-in-time compiler requires automatically finding good schedules without user input, while an expert programmer can provide a schedule in an ahead-of-time setup. This problem is more challenging than the standard auto-scheduling problem as the system must not only deliver high-performance kernels, but must also find schedules that compose well with existing data distributions in the machine. Another major technical challenge in building a general purpose sparse array programming system is the distributed and parallel assembly of sparse arrays as the result of computations. DISTAL can generate code that constructs sparse output arrays for a certain class of computations and data structures but must fall back to a generic approach that suffers in performance for many operations. In order to fully support user-defined new code generation techniques and abstractions must be developed to assemble sparse data structures in a distributed and parallel manner.

Looking beyond distributed array programming systems, I am broadly interested in investigating new ways of bringing high performance programming closer to the hands of everyday programmers. Future topics of interest include domain specific languages and frameworks for different application domains, such as those for simulations and matrix free computations, or more broad research in new parallel programming languages and systems.

References

[1] Rohan Yadav, Alex Aiken, and Fredrik Kjolstad. Distal: The distributed tensor algebra compiler. In *Proceedings of the 43rd ACM SIGPLAN International Conference on Programming Language Design and Implementation*, PLDI 2022, page 286–300, New York, NY, USA, 2022. Association for Computing Machinery.

[2] Rohan Yadav, Alex Aiken, and Fredrik Kjolstad. Spdistal: Compiling distributed sparse tensor computations. In *(To Appear) SC22: International Conference for High Performance Computing, Networking, Storage and Analysis*, 2022.