Bridging the Programmability Gap: From Laptops to Supercomputers

Engin Kayraklioglu

Computers are used in myriad disciplines for purposes including modeling, simulation and data analytics. However, increasing problem sizes and complexities necessitate using much computational power that is unattainable by a personal computer. Supercomputers can typically be used for such demanding tasks.

Programming supercomputers are notoriously difficult compared to personal computers. The most important characteristic that makes them hard to program is the distributed nature of the data and the processing elements. Supercomputers are sets of highly optimized (hardware- and software-wise) computers connected via a high-speed local network. As fast as the network may get, processors that access remote data always suffer from significant latencies compared to accessing local data. Therefore, programmers need to devote time to orchestrate data movement alongside the computation.

Many potential users of supercomputers, such as domain scientists and engineers, however, either do not have enough time to spend on optimizing their algorithms or lack enough experience. Moreover, in their use cases, computation is a tool rather than a purpose. Therefore, agility of computational development can enable such users to focus on more important tasks and quickly find solutions to complex science and engineering problems.

In this work, we aim to bridge the programmability gap between personal computers and supercomputers. Our approach is three-pronged:

- Modeling the data movement: We model the data movement as an engineering process instead of a monolithic optimization. We define subtasks and stakeholders (programmer, compiler and runtime system) involved in data movement and create an ontology to specify pieces of knowledge that need to be used by stakeholders to achieve optimal data movements.

- Designing a novel language feature: Using our model, we redefine the labor distribution for data movement to reduce programmer's burden. We design a language feature where programmers can specify access patterns easily, and language system can manage details of data movement.

- Implementing an assistive tool: We design a tool which can help programmers generate logs of accesses and present them in a user-friendly fashion. It augments the programmer capabilities to understand access patterns in their applications.

We show that the language feature we propose can achieve more than 100x speedup over unoptimized code and outperform laborious manual optimization. This speedup is achieved by adding very few lines of code to the unoptimized version. We believe that the insights we discovered can help design programming languages that reduce time to solution significantly.