Parallel Programming Systems (Justin Wozniak)

Accelerator-equipped systems have the potential to address the needs of computation-hungry scientific applications, but are difficult to program. GPUs lack traditional programming constructs but offer a new set of capabilities and concepts - maps, streams, and filters - highly relevant to scientific processing but requiring specialized techniques. The availability of math libraries and application kernels for GPUs also makes them increasingly attractive. Intel MIC processors (e.g. Xeon Phi Knights Landing) require a programming model with work offloading which can be a difficult task. This work seeks to address how programming many-core computing processors and accelerators can be made easier and more productive through implicitly parallel languages. For decades, researchers have tried with limited success to use dataflow models to simplify programming through implicit parallelism, distribution, and fault recovery. Our prior work on the Swift parallel scripting language [**1**, **4**] has made significant advancement toward those goals, providing implicitly parallel functional dataflow for compositional programming to a growing community of scientists and engineers. Students will investigate how a dataflow-based, compositional software development model can ease developing accelerated scientific codes and broaden the applicability of high-performance computing technologies. This includes systems research questions into the partitioning and deployment of computational work on accelerator devices and many-core processors, integration questions in the development of scalable applications, and utility questions into the benefits and challenges faced by scientific use cases. Our objectives are to deliver a unified dataflow language using cross-layer techniques towards micro-function granularity measured in cycles; students will work to design, analyze, and implement components of a framework for task-based computing on many-core computing platforms, supporting fine-grained workloads portably across architectures. We plan to evaluate performance and usability on real applications to characterize the usability and performance envelope of the proposed programming model on many-core systems.

The PI have explored the efficient support of many-core architectures for parallel runtime systems such as Swift [**1**, **2**, **3**] with the GeMTC project [**6**] (supporting NVIDIA GPUs) and the XTask project [**5**] (supporting Intel Xeon Phi). The GeMTC and XTask frameworks have significant performance advantages over traditional threaded or MPI-based runtime systems, but work needs to be done to integrate high level languages such as Swift with the GeMTC/XTask runtime systems. Students with some background in compilers and computer systems, would be exposed to cutting edge research in parallel languages and efficient runtime systems, and how to optimize these systems to run on many-core architectures efficiently.

These projects related to improving parallel programming systems are quite ambitious and complex on their own, but with the right guidance from the mentor Wozniak and senior PhD students who have been working on topics for many years, we believe that students with a solid systems background will be productive on these topics over a 10-week program.

[1] Y. Zhao, M. Hategan, B. Clifford, I. Foster, G. von Laszewski, I. Raicu, T. Stef-Praun, M. Wilde. “Swift: Fast, Reliable, Loosely Coupled Parallel Computation,” IEEE Workshop on Scientific Workflows 2007.

[2] M. Wilde, I. Foster, K. Iskra, P. Beckman, Z. Zhang, A. Espinosa, M. Hategan, B. Clifford, I. Raicu. "Parallel Scripting for Applications at the Petascale and Beyond", IEEE Computer Nov. 2009 Special Issue on Extreme Scale Computing, 2009.

[3] Y. Zhao, I. Raicu, I. Foster, M. Hategan, V. Nefedova, M. Wilde. “Realizing Fast, Scalable and Reliable Scientific Computations in Grid Environments”, Grid Computing Research Progress, Nova Publisher 2008.

[4] Swift, A simple tool for fast, easy scripting on big machines. http://swift-lang.org/main/, 2017.

[5] J. Anderson, P. Nookala, I. Raicu. “Building a High-Level Interface for XTask Fine-grained Parallelism for the Modern Era”, under review at IEEE/ACM Supercomputing/SC 2017.

[6] S.J. Krieder, J.M. Wozniak, T. Armstrong, M. Wilde, D.S. Katz, B. Grimmer, I. Foster, I. Raicu. “Design and Evaluation of the GeMTC Framework for GPU-enabled Many-Task Computing”, ACM HPDC 2014.