Title: Affine Loop Optimization Based on Modulo Unrolling in Chapel

Advisor: Dr. Rajeev Barua

This work presents modulo unrolling without unrolling (modulo unrolling WU), a method for message aggregation for parallel loops in message passing programs that use affine array accesses in Chapel, a Partitioned Global Address Space (PGAS) parallel programming language. Messages incur a non-trivial run time overhead, a significant component of which is independent of the size of the message. Therefore, aggregating messages improves performance. Our optimization for message aggregation is based on a technique known as modulo unrolling, pioneered by Barua [1] whose purpose was to ensure a statically predictable single tile number for each memory reference for tiled architectures, such as the MIT Raw Machine [2]. Modulo unrolling WU applies to data that is distributed in a cyclic or block-cyclic manner. In this paper, we adapt the aforementioned modulo unrolling technique to the difficult problem of efficiently compiling PGAS languages to message passing architectures. When applied to loops and data distributed cyclically or block-cyclically, modulo unrolling WU can decide when to aggregate messages thereby reducing the overall message count and runtime for a particular loop. Compared to other methods, modulo unrolling WU greatly simplifies the complex problem of automatic code generation of message passing code. It also results in substantial performance improvements in both runtime and communication compared to the non-optimized Chapel compiler.

To implement this optimization in Chapel, we modify the Cyclic distribution module's follower iterator and the Block Cyclic distribution module's leader and follower iterators, as opposed to creating a traditional compiler transformation. Results were collected that compare the performance of Chapel programs optimized with modulo unrolling WU and Chapel programs using the existing Chapel data distributions. Data collected on a ten-locale cluster show that on average, modulo unrolling WU used with Chapel's Cyclic distribution results in 64 percent fewer messages and a 36 percent decrease in runtime for our suite of benchmarks. Similarly, modulo unrolling WU used with Chapel's Block Cyclic distribution results in 72 percent fewer messages and a 53 percent decrease in runtime. Finally, the results from three different scaling experiments suggest that the greatest improvements from modulo unrolling WU occur when parallel follower iterator chunks of work contain the greatest number of data elements.