Chapel With Polyhedral Transformation Using Autotuning

Tuowen Zhao(presenting), Mary Hall
School of Computing
University of Utah
Salt Lake City, UT
{*ztuowen,mhall*}@*cs.utah.edu*

***Keywords*-Chapel; polyhedral transformation; autotuning; tiling; stencil**

The Chapel language is designed to increase programmer productivity in developing scalable parallel applications by separating the specification of the computation from that of how the data is mapped to a global, possibly distributed, address space. The data mapping is specified using domain maps, which can be implemented completely generally by user specifications, invoke a library of standard mappings, or rely on compiler code generation. Computations iterate over domains, abstracting a set of points within a domain map.

The simplest form of Chapel domains resemble sequential loop iterations in standard C. Thus, we observe that for a class of domains, those with *affine* loop bounds where bounds are linear functions of loop index variables, the same loop transformations available in polyhedral transformation and code generation frameworks could be applied to Chapel programs. Such frameworks represent and manipulate loop iteration spaces to apply transformations; their strength is the ability to compose transformations to apply a complex sequence of transformations to a code. Prior work has demonstrated the benefit of manually applying loop transformations to Chapel programs[1][2]. To apply polyhedral transformations, we use a polyhedral loop transformation tool CHiLL[3]. It enables us to transform code for complex loop nests in C. Locality and parallelization optimzations available in CHiLL can be be used to reduce vertical communication in programs by optimizing the utilization of bandwidth though the memory hierarchy. In this talk, we consider how to integrate CHiLL as a polyhedral framework with the Chapel compiler implementation.

A general decision algorithm for which transformations to apply and how to select optimization parameter values such as tile size or loop unroll factors is difficult and dependent on features of an architecture such as capacity of cache and register file. Thus, many researchers employ *autotuning* to derive the best set of optimizations and optimization parameter values for a given application targeting a specific architecture. In this talk, we combine polyhedral transformations with autotuning to explore a variety of optimization strategies for a computation, which allows the compiler to generate a search space of different optimization strategies and evaluate these strategies using model-guided empirical search [3]. Autotuning increases programmer productivity through a systematic exploration of a large collection of implementations that would be onerous for a programmer to write.

Our initial experiment demonstrates a performance benefits achievable when a collection of polyhedral transformations applied to Chapel domains (distribution and multiple levels of tiling) are combined with autotuning. Also, we explored the possibilities of using CHiLL to generate more complex computations for Chapel, especially stencil computations.

This talk includes the following topics: (1) it describes how to map Chapel domains to iteration spaces so that polyhedral transformations can be composed to optimize Chapel programs; (2) it demonstrates with a simple example the benefits of applying such transformations in conjunction with autotuning; and, (3) it describes and discusses how to generate stencil computations using this method in Chapel.

References

[1] I. J. Bertolacci, C. Olschanowsky, B. Harshbarger, B. L. Chamberlain, D. G. Wonnacott, and M. M. Strout, "Parameterized diamond tiling for stencil computations with chapel parallel iterators," in *Proceedings of the 29th ACM on International Conference on Supercomputing*, ser. ICS '15. New York, NY, USA: ACM, 2015, pp. 197–206.

[2] A. Sharma, D. Smith, J. Koehler, R. Barua, and M. Ferguson, "Affine loop optimization based on modulo unrolling in chapel," in *Proceedings of the 8th International Conference on Partitioned Global Address Space Programming Models*. ACM, 2014, p. 13.

[3] C. Chen, J. Chame, and M. Hall, "Chill: A framework for composing high-level loop transformations," Citeseer, Tech. Rep., 2008.