Compilation Techniques for Partitioned Global Address Space Languages

Kathy Yelick

EECS Department, UC Berkeley
Computational Research Division, Lawrence Berkeley National Lab

Abstract. Partitioned global address space (PGAS) languages have emerged as a viable alternative to message passing programming models for large-scale parallel machines and clusters. They also offer an alternative to shared memory programming models (such as threads and OpenMP) and the possibility of a single programming model that will work well across a wide range of shared and distributed memory platforms. Although the major source of parallelism in these languages is managed by the application programmer, rather than being automatically discovered by a compiler, there are many opportunities for program analysis to detect programming errors and for performance optimizations from the compiler and runtime system. The three most mature PGAS languages (UPC, CAF and Titanium) offer a statically partitioned global address space with a static SPMD control model, while languages emerging from the DARPA HPCS program are more dynamic.

In this talk I will describe some of the analysis and optimizations techniques used in the Berkeley UPC and Titanium compilers, both of which source-to-source translators based on a common runtime system. Both compilers are publicly released and run on most serial, parallel, and cluster platforms. Building on the strong typing of the underlying Java language, the Titanium compiler includes several forms of type-based analyses for both error detection and to enable code transformations. The Berkeley UPC compiler extends the Open64 analysis framework on which it is built to handle the language features of UPC. Both compilers perform communication optimizations to overlap, aggregate, and schedule communication, as well as pointer localization, and other optimizations on parallelism constructs in the language. The HPCS languages can use some of the implementation techniques of the older PGAS languages, but offer new opportunities for expressiveness and suggest new open questions related to compiler and runtime support, especially as machines scale towards a petaflop.

G. Almási, C. Caşcaval, and P. Wu (Eds.): LCPC 2006, LNCS 4382, p. 1, 2007.