R-Stream: A Parametric High Level Compiler

Richard Lethin, Allen Leung, Benoit Meister and Eric Schweitz
{lethin, leunga, meister, schweitz}@reservoir.com
Reservoir Labs, Inc, New York, NY

R-Stream

R-Stream is a High Level Compiler being developed as part of the DARPA IPTO Polymorphous Computer Architecture (PCA) program. The compiler is targeted at the problem of mapping embedded signal/knowledge processing applications with high performance needs, as exemplified by the Lincoln Labs Integrated Radar Tracker (IRT) reference, to both polymorphous and commercial streaming hardware platforms. These polymorphous streaming architectures are exemplified by TRIPS (University of Texas) and MONARCH (ISI/Raytheon) projects. Other emerging commercial multi-core chips, such as IBM's Cell, are also instances of on-chip distributed memory multiprocessors that require explicit communication between tiled computation kernels.

The Morphware Forum is an initiative under the PCA program to develop standard mechanisms and tools for the programming of PCAs. Within this framework, R-Stream is an instance of the high-level compiler (HLC), a source-to-source restructuring tool in a phased compilation system.

More specifically, R-Stream is an ANSI C compiler aimed at producing high-level mapped code, i.e., C augmented with mapping instructions. This is exemplified by the Streaming Virtual Machine (SVM), which is a Morphware supported interface between a HLC and a target-specific low-level C compiler.

Internally, the version of R-Stream under development uses a polyhedral model of the program and architecture, drawn from academic work in advanced program representations. The polyhedral model represents affine loops [1], data references, schedules on parallel processors [2,3], data layout [4], computations and data distribution across multiple memories in a single unified mathematical framework: multidimensional parametric polyhedra. The approach subsumes many classic loop optimizations and allows to extract more parallelism and to better handle parameterized codes and architectures. In particular, it enables coordinated optimizations that traditionally must be made in a phased manner such as parallelization and data layout for locality.

Recently, we have been experimenting with our mapper technology on several kernel codes extracted from Ground Moving Target Indicator radar (GMTI, the front half of the IRT) and from Synthetic Aperture Rader (SAR), as well as basic linear algebra algorithms.

Compiler Structure

An XML-structured document describing the target machine is provided to R-Stream as an input parameter. R-Stream uses an industrial-strength front-end to parse sequential programs written in ANSI C with some extensions. Mappable regions, i.e., regions that may contain the appropriate parallelism, are then extracted and turned into polyhedral form.

This polyhedral information is then analyzed and manipulated in order to produce a mapping that includes the scheduling and grouping of operations into tasks, the placement of tasks and data onto different processors, the identification of needed values, the compaction of values needed in communication operations, and optimization of those communication operations.

Once mapped, responsibility is given to the code generation phase for producing the high-level output that can be consumed by either a native/low-level compiler or, potentially, a programmer who could use R-Stream as a parallelism and locality extraction tool.

Polyhedral Representation

The polyhedral representation models coding practices commonly found in high-performance applications in an abstract and uniform mathematical object, a polyhedron. Targeted codes consist of embedded static control loops that show some regularity. In Fig. 1, we show how iteration domains, array variable access patterns, and a schedule may be represented using polyhedra.

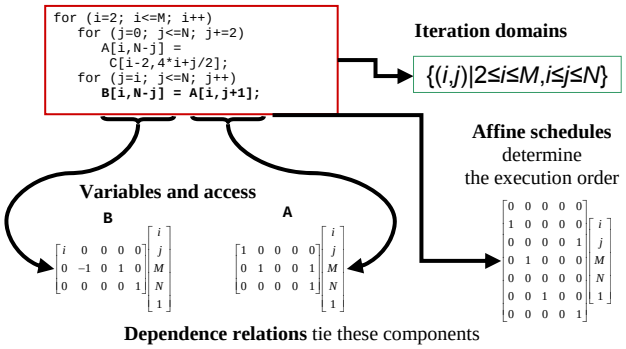

Figure 1: Polyhedral Model in a Nutshell

Polyhedra can straightforwardly model nested loops whose bounds are affine functions of the loop counters and other variables that are loop constants, as well as array elements accessed by such loops through multidimensional affine access functions. This framework, which already allows modeling many linear algebra and signaling processing kernels, can be extended to handle more complex cases [5]. Moreover, input code may be pre-processed to better fit the polyhedral model in cases when the original code is not directly amenable to being modeled. (See [6] for instance.) Reservoir Labs is also contributing to this field by

extending the applicability of these techniques with its own original research.

R-Stream is designed to use the polyhedral model to perform the following tasks in the mapper:

- Identifying and extracting parallelism
- Forming tasks
- Loop transformations
- Locality optimization
- SIMD optimizations
- Data layout transformations
- DMA generation
- Communication optimizations
- Data distribution and processor mapping

Mapping to Cell

In recent work, we have begun exploring the use of our mapper to restructure code for the IBM/Sony/Toshiba Cell processor. A block diagram of the Cell processor architecture is shown in Fig. 2.

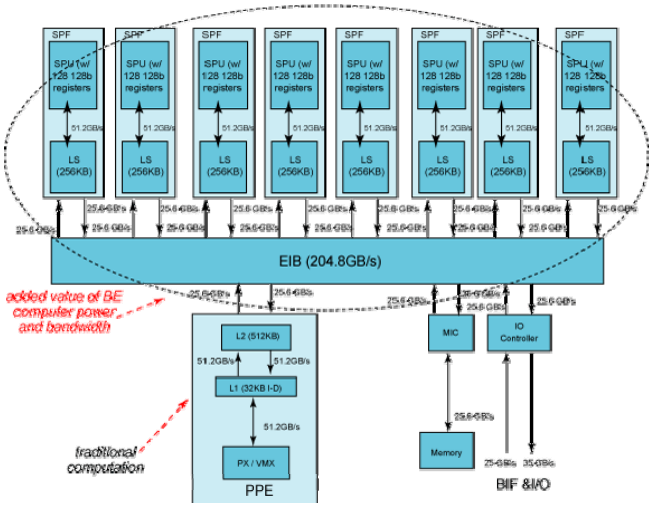

Figure 2: Cell processor (diagram is © of IBM)

The Cell processor is able to achieve a theoretical maximum of approximately 205GFLOPS. However, in order to obtain near this level of raw number crunching power requires some unique programming of the chip as a whole. For example, the 8 SPEs and the PPE may take advantage of a fairly coarse-level of parallelism; the SPEs may be used to exploit another level of parallelism; and finally, each individual SPE may internally exploit a very fine-grained SIMD parallelism on long data words.

Clearly, an advanced framework that can detect parallelism at many levels, optimize for locality, and produce parallel code holds productivity advantages over programmers that can be overwhelmed when managing this complexity by hand.

Recent Results

We are experimenting with our polyhedral mapper to produce code for the Cell processor. This code, which exposes parallelism and (DMA) communications, can then be hand-tuned or compiled by the Cell "low level" compilers (namely, gcc and xlc). This also allows us to study the interaction between R-Stream and the low-level compiler. The output of our code generator is shown for a matrix multiply kernel in Fig. 3.

Figure 3: Cell output code

Related Work

IBM researchers are working on the *Octopiler* [7], a compiler that targets the Cell processor and performs automatic parallelization. This compiler is similar to ours, in the sense that it also tries to extract parallelism. However, R-Stream differs from the Octopiler in that R-Stream is using a unique, higher-level kind of modeling to drive its automatic parallelization. Furthermore, while the Octopiler target is fixed (the Cell architecture exclusively), R-Stream is targeting a parametric parallel machine.

References

[1] C. Ancourt, F. Irigoin, "Scanning polyhedra with do loops", Proceedings of the third ACM SIGPLAN symposium on Principles and practice of parallel programming, 1991

[2] P. Feautrier, "Some efficient solutions to the affine scheduling problem", International Journal of Parallel Programming, 21(5), 1992

[3] A.W. Lim, M.S. Lam, "Maximizing parallelism and minimizing synchronization with affine partitions", Parallel Computing, 24(3-4), 1998

[4] V. Loechner, B. Meister and Ph. Clauss, "Precise data locality of nested loops", The Journal of Supercomputing, 21(1), 2002

[5] M. Geigl, "Parallelization of loop nests with general Bounds in the Polyhedron Model", M.Sc. thesis, Universisaet Passau,1997

[6] J. Sheldon, W. Lee, B. Greenwald, and S. Amarasinghe, *Proceedings of the '01 Languages and Compilers for Parallel Computing (LCPC)*, 2001.

[7] A. E. Eichenberger et al., "Using advanced compiler technology to exploit the performance of the Cell Broadband Engine architecture," IBM Systems Journal, 45(1), 2006.