The Loop Parallelizer LooPo—Announcement

Martin Griebl and Christian Lengauer

FMI, Universität Passau, D–94030 Passau, Germany
{griebl,lengauer}@fmi.uni-passau.de, http://www.uni-passau.de/~lengauer

1 What is LooPo?

LooPo is a new loop parallelizing framework developed at the University of Passau to aid us in research on the space-time mapping of loop nests. LooPo is available on the Web [10] and uses only free software. It runs under SunOS 4.1.x, Solaris 2.x, and Linux.

LooPo is based on the polytope model [8]. Anybody interested in one special aspect of the parallelization can plug his/her own module into LooPo and obtai a customized source-to-source compilation environment. The central data structures (restricting the applicability of LooPo) are—according to the model—polyhedra and piecewise affine functions.

The next major version will also have the capability of parallelizing loop nests containing while loops [9].

2 The Structure of LooPo

LooPo consists of a number of modules, listed below, which transform the source program to an executable parallel target program. A front end provides the user with a graphical interface by which he/she can control LooPo. There is also a graphical displayer of index spaces and iteration dependence graphs of loop nests.

The input. LooPo accepts (possibly imperfect) loop nests in C, Fortran or a number of abstract loop notations, and declarations of functions, procedures and symbolic constants.

The inequality solvers. We considered several methods for parametric linear programming, which is the central mathematical problem of the polyhedron model [10]. The current implementation of LooPo is based on PIP and Fourier-Motzkin. *Fourier-Motzkin* is the standard doubly exponential method of polytope projection [1]. *PIP* is Feautrier's system for parametric integer programming [3].

The dependence analyzer. At present, LooPo features the dependence analysis method described by Banerjee [1]. We are working on the implementation of Feautrier's method [4], which allows potentially for more parallelism, due to a more precise analysis.

The schedulers. Presently, LooPo provides three automatic schedulers. The hyperplane method of Lamport [7] handles perfectly nested for loops with uniform dependences. The Feautrier scheduler [5] determines an optimal (concave) schedule for imperfectly nested for loops with affine dependences, at the cost of a longer computation time caused by parametric integer linear programming [3]. The scheduler of Darte/Vivien [2] is fast and produces reasonably good results for arbitrary loop programs with uniform and non-uniform dependences.

The allocator. Presently, LooPo provides only Feautrier's allocation method [6] which determines the placement of operations on virtual processors based on the owner computes rule.

The target code generator. The target code generation proceeds in three phases [11]: first, the statements of the source program are transformed individually. Then, these transformed statements in their respective index spaces are merged to a single target program. Finally, the target parse tree is translated to one of a variety of possible output languages, e.g., parallel C, and synchronization and communication is added if the user so desires.

Acknowledgements

We thank the LooPo team [10] and J.-F. Collard. This work received funds from DFG project RecuR and from the DAAD exchange program PROCOPE.

References

1. U. Banerjee. *Loop Transformations for Restructuring Compilers: The Foundations*. Kluwer, 1993.
2. A. Darte and F. Vivien. Automatic parallelization based on multi-dimensional scheduling. Technical Report 94-24, Laboratoire de l'Informatique du Parallélisme, Ecole Normale Supérieure de Lyon, September 1994.
3. P. Feautrier. Parametric integer programming. *Operations Research*, 22(3):243–268, 1988.
4. P. Feautrier. Dataflow analysis of array and scalar references. *Int. J. Parallel Programming*, 20(1):23–53, February 1991.
5. P. Feautrier. Some efficient solutions to the affine scheduling problem. Part II. Multidimensional time. *Int. J. Parallel Programming*, 21(6):389–420, October 1992.
6. P. Feautrier. Toward automatic distribution. *Parallel Processing Letters*, 4(3):233–244, 1994.
7. L. Lamport. The parallel execution of DO loops. *Comm. ACM*, 17(2):83–93, February 1974.
8. C. Lengauer. Loop parallelization in the polytope model. In E. Best, editor, *CONCUR'93*, Lecture Notes in Computer Science 715, pages 398–416. Springer-Verlag, 1993.
9. C. Lengauer and M. Griebl. On the parallelization of loop nests containing while loops. In N. N. Mirenkov, Q.-P. Gu, S. Peng, and S. Sedukhin, editors, *Proc. 1st Aizu Int. Symp. on Parallel Algorithm/Architecture Synthesis (pAs'95)*, pages 10–18. IEEE Computer Society Press, 1995.
10. LooPo. http://www.uni-passau.de/~loopo/.
11. S. Wetzel. Automatic code generation in the polytope model. Diplomarbeit, Fakultät für Mathematik und Informatik, Universität Passau, 1995.