Review of [Lamport CACM 1974]
The Parallel Execution of DO Loops

Sanjay Rajopadhye, CSU CS, August 15 2013

Summary of the paper

This paper presents two algorithms—called the hyperplane method and the coordinate method—for automatically parallelizing a nested loop program. The algorithms are applicable to perfectly nested loops (i.e., any assignment statement is inside the innermost loop), where the assignment statements have a specific form, and where the loop bounds are either statically known constants or expressions involving the surrounding loop indices. The main technical contribution is the hyperplane method, while the coordinate method is a "simpler" method that can be implemented without performing complicated loop transformations, but is not always applicable. The hyperplane method is the more general one.

Who is the audience?

The audience is compiler writers seeking to automatically rewrite programs to use parallel machines such as the ILLIAC IV.

Importance and Context

Is it important/interesting? What was the context for the paper? Why should the audience care?

The paper was one of the earliest papers that addressed the problem of automatic parallelization. The problem was extremely important, and has acquired new urgency in the era of multicore programming. When the paper was written, parallel computers were a novelty and usually research prototypes rather than productions machines. The paper therefore provided some seminal results on automatic parallelization.

The paper has been a very influential paper in automatic parallelization. Google scholar lists over 700 citations. In my opinion, this number could have been much higher. FOr example, the Wolf-Lam 1991 PLDI paper that presented an algorithm

to combine scheduling and locality optimization (tiling) has more than twice as many citations, although Lamport's results are just as important.

Approach

What is the approach used to solve the problem?

The techniques that were use were drawn form linear algebra. To develop the hyperplane method, Lamport defines the class of programs for which the method can be applied, and then develops a *dependence analysis* for this class of programs—an analysis that identifies precisely the set of instances of loop iterations on which a given iteration depends. In general this set can be unmanageably large, so the paper proposes a compact representation. Next, an algorithm is developed that deduces a program transformation that generates a new set of loops (of the same depth) such that a certain number of outer loops are sequential while the inner ones can be executed concurrently since there are no dependences between the instances of these loops.

The dependence analysis presented in the paper represents dependences as a finite number of vectors whose elements are either constants or symbols like $+$ or $*$ which represents any (strictly or not) positive integer. This is true regardless of the size of the iteration space or the number of iterations that the program executes. This is the earliest paper where such compiler analysis is presented.

Next, the hyperplane method formulates the parallelization problem as identifying a "scheduling vector" π (the normal to a family of hyperplanes) such that the dot product of π and the dependence vectors is strictly positive. Lamport then provides a greedy algorithm that solves this problem and claims that the general problem can be formulated as an integer programming problem, but that a solution to that is not available.

Justification

How does the paper support or otherwise justify the conclusions it reaches?

The paper justifies its claims mathematically, by providing a proof of correctness of the proposed algorithms.

Strengths/Weaknesses

What are the strengths and weaknesses of the paper?

As stated above, the strengths of the paper are its seminal contributions to the theory of automatic parallelization. The hyperplane theorem has an elegant proof, and has served as the basis for most later work on scheduling, whether it was in the context of systolic array synthesis or for automatic parallelization.

The most important weakness of the paper in my opinion, is that it is very poorly written. Notation is not clearly explains,ed, and introduced on the fly. Elegant mathematical results are interspersed with rather grandiose claims (e.g., "It is possible to generalize our results to the case [... where] e^i is any linear function ...]" on page 5).

In my opinion, this has reduced the impact of this paper, as reflected inits citations.

Open Problems

What problems are explicitly or implicitly left as future research questions?

The problem of actually implementing the algorithms in a compiler have been left unresolved as well as a precise quantitive measure of the efficacy of the method.