Communication Pre-evaluation in HPF

Pierre Boulet¹ and Xavier Redon²

¹ LIP, École Normale Supérieure de Lyon, 46, allée d'Italie, F-69364 Lyon cedex 07, France
² LIFL, Université des Sciences et Technologies de Lille, Bâtiment M3, Cité Scientifique, F-59655 Villeneuve d'Ascq cedex, France

Abstract. Parallel computers are difficult to program efficiently. We believe that a good way to help programmers write efficient programs is to provide them with tools that show them how their programs behave on a parallel computer. Data distribution is the major performance factor of data-parallel programs and so automatic data layout for High Performance Fortran programs has been studied by many researchers recently. The communication volume induced by a data distribution is a good estimator of the efficiency of this data distribution.

We present here a symbolic method to compute the communication volume generated by a given data distribution during the program writing phase (before compilation). We stay machine-independent to assure portability. Our goal is to help the programmer understand the data movements its program generates and thus find a good data distribution. Our method is based on parametric polyhedral computations. It can be applied to a large class of regular codes.

1 Introduction

Parallel computing has become the solution of choice for heavy scientific computing programs. Unfortunately parallel computers require considerable knowledge and programming skills to exploit their full potential. A major mean to reduce the programming complexity is to use high-level languages. High Performance Fortran (HPF) is such a language. It follows the data-parallel programming paradigm where the computations are directed by the data. Indeed, the computations occur on the processor that "owns" the data being written. So the data distribution is a very important efficiency factor when programming with a data-parallel language.

Our goal is to help the programmer find a good data-distribution. We want a tool that is executed when the program is being written. Work on such a tool has started at the LIIFL with HPF-builder [6], a tool that interactively displays the arrays as they are distributed and aligned by the HPF directives. This tool also allows to change the data distribution graphically.

We propose here a new step in the development of a tool that helps the programmer understand how data move in its program. Given a data-distribution, we are able to compute the volume of the communications generated by a program. We use symbolic computation tools to stay free of the problem size. All
this is done at the language level, thus retaining portability and machine independence. This tool is intended as a mean to write a reasonably efficient program that can be tuned for a particular parallel machine with profiling systems later.

The sequel of this paper is organized as follows. In Sect. 2 we briefly review related works, then in Sect. 3 we describe the problem we consider and present its modelization in Sect. 4. We then detail the tools used to solve our problem along with an example in Sect. 5. And we finally conclude in Sect. 6.

2 Related Work

Many researchers [1, 9, 10, 13-15] have studied automatic data distribution. Estimating communication costs has been the key factor to determine the quality of a data distribution. Most of the previous works [12, 7, 11] have studied compile-time estimation of these communication costs. Indeed, they use the fact that most program parameters are known at that time and in many cases, these studies also use machine (and compiler) dependent data. Our work differs from previous work by the techniques used and the stage of program development we focus on: the program writing phase. We also use exact parameterized methods and we stay compiler and machine independent, we work at the language level.

Description of the Problem

3.1 General Remarks

The problem we study in this paper is the evaluation of the communication volume in a HPF program before compilation. The goal is to help the programmer understand the communications generated by his program and find a good data distribution (or a more efficient way to code his algorithm).

The communications we consider are at the PROCESSORS level: as in HPF, we use an abstract target machine. We stay at the language level, thus allowing to find a data distribution that is well suited to the problem, and thus retaining portability. Our aim is not to find the best data distribution for a given machine, but a good one for any machine (and compiler).

In the future, if some compiler optimization techniques such as overlap areas to vectorize communications are used by most compilers, we could adapt our evaluation techniques to these optimizations. In a first step, though, we remain at the language level to validate our approach. We believe indeed that a good data distribution at the language level should not be a bad one for any compiler, regardless of the compilation techniques used.

3.2 Modelization

We consider that the only communication generation statements are storage statements, we do not take into account I/O statements. For each of these storage