Joint NCC & IIS Bull., Comp. Science, 12 (1999), 15–18

Support tools for supercomputing*

V.N. Kasyanov, V.A. Evstigneev, Ju.V. Malinina,
Ju.V. Birjukova, V.A. Markin, E.V. Haritonov, S.G. Tsikoza

The paper considers the problem of supercomputing support and Internet technologies. The PROGRESS project being developed at the A.P.Ershov Institute of Informatics Systems is discussed. The system is intended to support rapid prototyping of compilers for high-level languages (e.g., Fortran-77, Modula-2, Sisal-90) and for a family of architectures exploiting fine-grained parallelism. The next goal of the project is to develop an environment for investigation of optimizing and restructuring transformations of programs to be parallelized. The information subsystem TRANSFORM aimed at accumulating and processing knowledge about these transformations is considered. The subsystem SFP intended to support numerical high-performance computation on the base of the applicative language Sisal-90 is outlined.

1. Introduction

As the number of computers with parallel architectures and telecommunication networks increases, system programmers face numerous new problems concerned with their effective usage. Parallel programming (i.e., writing programs that exploit explicit parallelism) used for these purposes seems to be more complex than sequential. Conventional methods of programming based on explicit parallelism make it difficult to understand parallel code details. Explicit parallelism is usually considered as an extension of existing languages by means of parallel computation control.

Automatic parallelizing of programs is free of these disadvantages, but it has its own ones related to "conservatism" of compilers oriented on an "average" program in analysis and transformations of a program. In other words, while parallelizing a program, it is impossible to completely extract parallelism contained in it. Most of existing parallelizing compilers require some prompts from a programmer either before or during compilation, or they report segments that need analysis and reprogramming to be successfully parallelized. Thus it is necessary for an application programmer to be skilled in program transformation technique. Another approach concerns creation of appropriate optimizing compilers based on the so-called implicit parallel model. In this model an optimizing compiler is responsible for most control operations and actually constructs a parallel program under the input specification which, in particular, looks like an ordinary sequential program.

In the paper some methods and tools are discussed which support high-performance computation being developed at the A.P. Ershov Institute of Informatics Systems [1–5]. Investigations here reported are carried out within the PROGRESS project sponsored by the Russian Foundation for Basic Research and by the Russian Ministry of Higher Education. Our goal is to elaborate a framework to study optimizing and restructuring program transformations — the TRANSFORM system — and the SFP system for functional programming and support of high-performance computation.

2. PROGRESS project

The PROGRESS project is aimed to create a programming environment that supports:

- research of optimizing and restructuring transformations; design of new systems of transformations and implementation algorithms; search for efficient forms of an intermediate representation of programs, sets of transformations, context conditions and strategies of transformation application to various classes of programs and computers;

*Supported by the Russian Foundation for Basic Research under Grant 98-01-00748.

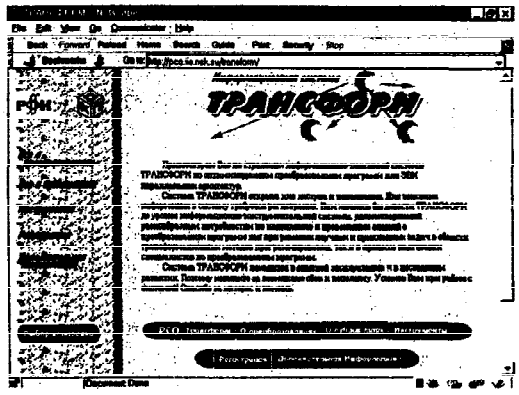

Figure 1. The information-retrieval system TRANSFORM: Russian Pages

- fast prototyping of optimizing compilers for various target architectures, such as VLIW, superscalar and multiprocessors with disrtibuted memory;
- training students in programming and optimizing compilation methods for parallel architectures.

To achieve these goals, the system is being developed as a tool for manipulation with programs; it includes means for step-by-step program transformation — from input text and upto either a text representation of the transformed program or an executable code to be run on the target architecture model.

Both imperative high-level languages (such as high-level FORTRAN-77, Modula-2, C++) and functional languages (such as SISAL) are considered as input languages.

The languages are supposed to be extended by program annotation means (formalized comments). By using them, a user can control the program transformation process (for example, by specifying additional information about properties of the program and the context of its execution), and the system can comment a residual program with information about the transformation process. It is possible to transform a sequential program into a parallel one (say, FORTRAN-77 program into the equivalent FORTRAN-90 one).

3. TRANSFORM Information Subsystem

The TRANSFORM subsystem is an important unit of the PROGRESS project. The subsystem stores information about optimizing and restructuring transformations for parallel computer programs and is interesting in itself. The experimental version of the TRANSFORM subsystem oriented to Internet environment (see Fig. 1) is now accessible at http://pcosrv.iis.nsk.su/transform/.

Here are the basic design decisions on the information system:

- implementation of TRANSFORM on the base of freely distributed software (GNU);
- integration of hypertext facilities and database tools;
- development of TRANSFORM in terms of the client/server technology;
- usage of standard WWW-client applications as a user interface;
- usage of the SQL language for interaction between the DB and WWW-servers;
- usage of TCP/IP protocol as the system's transport layer.

The system is divided into three interacting functional components: data visualizator, database manager and data analysis tools. The first is to display the database structure, initial data and results

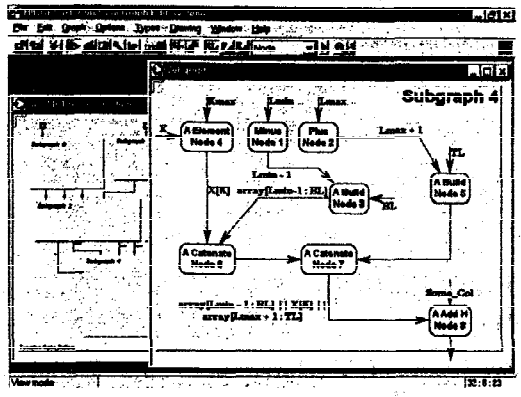

Figure 2. Visualization of an internal representation of Sisal-program

of analysis, i.e., it helps in navigating in the data and component space. The second component serves to control the database access, privileges, storage parameters and also to extract data according to some criterion. The third component helps a user to make conclusions, performs data processing, supports solutions and verifies hypotheses.

The TRANSFORM users can be divided into three groups:

- getting quantitative statistic by analysing data arrays contained in the database;
- extracting certain information by requests to the database;
- various visual representations of data that would help one to understand them better.

The System WWW-interface consists of WWW-client (Netscape, MS Internet Explorer) ↔ WWW-server (Apache) ↔ FastCGI-gateway (Lisp, C) ↔ database server (Postgres95) ↔ database.

Our future plans are to put TRANSFORM into experimental operation and to adjust it to various needs of knowledge accumulation and application. This will help us in solving scientific and applied problems in the area of tranformational programming methodology as well as in teaching program transformations to students.

4. The SFP Functional Programming System

Another component of the PROGRESS project is a functional programming system (SFP) based on the Sisal language aimed at support of high-performance computation.

The system is supposed to provide a programmer with a convenient environment that helps him to develop functional programs for various parallel architectures available via a network. This environment should provide means that allow one, first, to write and debug programs regardless of the target architecture and then to adapt the correct program to a certain target parallel architecture in order to achieve high efficiency of program execution on a supercomputer.

A program adaptation process consists of an optimizing cross-translation of a functional program being developed to the supercomputer's language (for example, C++). A programmer controls the process of program optimization and transformation by supplying additional information or by a direct control of these transformations. In particular, a programmer should be able to visually process a Sisal-program in terms of its internal representation (see Fig. 2).

The functional language SISAL (Steams and Iterations in a Single Assignment Language) is considered as an alternative to the FORTRAN language for supercomputers [6]. When compared with imperative languages (like FORTRAN), a functional language, such as Sisal, simplifies the programmer's work. He has only to specify the result of calculations and it is a compiler that is responsible for mapping an algorithm to a certain calculator architecture (including the instruction schedule,

data transfer, execution synchronization, memory management, etc.) In contrast to other functional languages, Sisal supports the data types and operators typically used in scientific calculations.

References

[1] V.A. Evstigneev, V.N. Kasyanov, *A program manipulation system for finegrained architectures*, Proc. EURO-PAR'95: Parallel Processing, Lect. Notes Comput. Sci., **966**, 1995, 719–722.

[2] V.A. Evstigneev, V.N. Kasyanov, *Optimizing transformations in optimizing compilers*, Programmirovanie, No 6, 1996, 12–26 (in Russian).

[3] V.N. Kasyanov, V.A. Evstigneev, L.V. Gorodniaja, Ju.V. Birjukova, Ju.V. Malinina, S.G. Tsikoza, T.A. Klimova, E.V. Haritonov, *Parallel processing: problems of training*, New informatics technology in the university education, 1997, 186–187 (in Russian).

[4] *Problems of designing of the effective and reliable programs*, (V. Kasyanov, ed.), Novosibirsk, 1995 (in Russian).

[5] *Optimizing compilation and constructing programs*, (V. Kasyanov, ed.), Novosibirsk, 1997, , (in Russian).

[6] J. Feo, P. Miller et al., *Sisal 90*, Proc. High Performance Functional Computing, Denver, Colorado, 1995, 35–47.