Macromodule technology

Victor P. Gergel, Alexey Sidnev
Computational Mathematics and Cybernetics department,
N.I. Lobachevsky State University,
Nizhny Novgorod, Russia
e-mail: alexey.sidnev@itlab.unn.ru

***Abstract*— A development of the parallel, optimal and portable software is a difficult problem. It consists of learning of libraries, programming techniques (such as optimization and paralleling), usage of the libraries and modification of written code. At present there is a set of the optimized libraries for the big number of tasks. But each library is unique, it demands studying and is optimized for specific software and hardware systems. It complicates a choice of the most suitable library (or several libraries) for usage and implementation in the developed software. In this paper, we propose approach for solving the specified problems. The main idea of the approach is the semantic description of programmed code.**

Keywords-macromodule technology; semantic description of code

I. INTRODUCTION

Recently software development has accepted mass character, being hard professional work. Development of parallel programs for high-efficiency computing systems is much more difficult. The main complexity of it consists of learning of libraries, programming techniques (such as optimization and paralleling), and usage of the libraries and modification of written code.

In general, the computer program represents sequence of operations which computer has to execute (Fig. 1).

```
Program Demo
  Action 1
  Action 2
  …
End_of_program
```

Figure 1. The common view of the program

In most cases, separate small operations can be merged in the larger units. In that case the modular representation of program looks like on Fig. 2.

Programs can consist of much number of units. It is possible to select some standard units which describing solutions of some typical tasks among them (an ordering of the data, search of the minimum or maximum value, etc.). These standard units can be developed, placed in libraries of standard units and used. In this case programs look like on Fig. 3.

```
Program Demo
  Unit 1
  Unit 2
  …
End_of_program
```

Figure 2. Modular representation of the program

The similar technology – usage of libraries – is one of the main in a software development. This approach has many advantages. Such as quality implementation of the standard units, essential decrease development coast. At present there is a set of libraries solving tasks of linear algebra (library LAPACK, ATLAS), many-dimensional multicriterion optimization, fast Fourier transform (library FFTW, MKL), etc. There are a number of problems at the library approach. First of all, the programmer has to know a lot of existing libraries. He must know the structure and the rules of usage of the library, because each library is unique. It is difficult to overcome such problems. It is hard to provide portability of programs (work of programs in various hardware and software conditions). In this case, upgrade of programs with replacement of used library units is usually required.

In this paper we propose the technology of macromodular software development which decreases complexity of programming.

```
Program Demo
Unit 1
  Call the unit from library
  Unit 2
  …
End_of_program
```

Figure 3. Modular representation of the program with usage of libraries

II. MACROMODULAR TECHNOLOGY

A. General Overview

The main idea of the macromodular approach is the semantic description of programmed code. It is performs description analysis and automatic assembly of optimal

application for target software and a hardware platform (portable, parallel or for the specialized processor). Thus the basis for assembly is debugged and optimized libraries. The software developer does not choice the most suitable library and does not write code to use that library. It is enough to make the macrodescription of a program code and to specify a target platform. Modification of the program under new software and a hardware platform is simplified. It is enough to rebuild the application.

First of all, programmer writes description of code. It consists of action specification, data storage format and so on. Description based on language syntax rules such as preprocessor directives. Preprocessor is a program that processes the code before it passes through the compiler. So it can transform program before actual compilation. After that the programmer selects target software and hardware platform and perform assembly of optimal application.

B. Usage Example

As an example we will consider a problem of matrix multiplication [1]. It is popular enough problem. It finds application in problems of a computer graphics, physicists, etc. Matrix multiplication is realized in many libraries (LAPACK, ATLAS, MKL). The example will be considered on the C language.

The implementations of square matrixes multiplication is presented on the Fig. 4. Matrixes A and B are initial matrixes. In matrix C the result of multiplication is stored.

Matrix multiplication is a simple enough task but it is necessary to know:

- Matrixes size.
- Type of matrix elements. It can be simple type (for example, integer type or float type) or complex type (for example, complex number).
- Data storage format. It can be dense representation (in the rows or columns) or sparse (Compressed Row Storage, Compressed Column Storage, Sparse Block Compressed Row Storage, etc) [2].

```
for(i = 0; i < n; i++)
   for(l = 0; l < n; l++)
     for(j = 0; j < n; j++)
       C[i*n+j] += A[i*n+l] * B[l*n+j];
```

Figure 4. Classic matrix multiplication

Macromodular technology uses preprocessor directives for tasks specification, such as matrix multiplication. We will use special directives, named pragmas. Pragmas are the preprocessor instructions, so they are processed before a compilation stage. It often controls actions of the compiler and linker. All unrecognized pragmas are ignored. For example, OpenMP parallel programming model is based on pragmas [3].

Directives of macromodular technology describe the block of code and have the following format: "#pragma mmt <action>(<parameters>)". <Action> is an operation which performs in the specified block, for example matrix multiplication. <Parameters> are input and output arguments of the computing function. Description of matrix multiplication is presented of Fig. 5.

All matrixes in the example are square, have type of elements "float" and have dense representation in memory (so-called style C).

C. Target Platforms

We consider CUDA (toolkit version 2.3) [4] and MKL (version 10.0.5.025) [5] libraries. MKL is a library of optimized and threaded math routines for science, engineering, and financial applications that require maximum performance. CUDA is a library for GPU allows developing the programs for nVidia video cards.

Available set of libraries is the basis for select of the target platform. If target system is GPU oriented (has a powerful graphics card) it is more preferable to use CUDA. If target system is CPU oriented (has a powerful processor of Intel) it is more preferable to use MKL.

The operating time of matrix multiplication (type "float") on various platforms is presented on Fig. 6. The first platform has two quad cores Intel Xeon E5320 processors, but very slow integrated video card. The second platform has GPU GeForce 8800 GTS and dual core Intel Core 2 E6650 processor.

On the second platform, performance of matrix multiplication on the GPU is more effectively, than on the processor. First platform with two quad cores processors have shown the best result. On the first platform, GPU does not support general computation. So libraries have various effectiveness on different platforms.

D. Current Research

Development of a prototype of the system supporting macromodular technology is at present carried on. A development of the extension for Visual Studio 2008 [6] is now finished. It allows performing preprocessing of a program and a choosing of a target platform.

```
#pragma mmt mmult(A=Matrix<CStyle, float>(n, n), B=Matrix<CStyle,
float >(n, n), C=Matrix<CStyle, float >(n, n))
{
for(i = 0; i < n; i++)
   for(l = 0; l < n; l++)
     for(j = 0; j < n; j++)
       C[i*n+j] += A[i*n+l] * B[l*n+j];
}
```

Figure 5. Macromodule definition of matrix multiplication

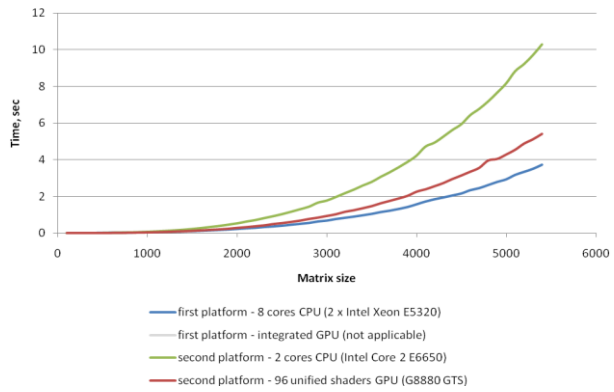

Figure 6. Compare CPU and GPU performance on different platform

III. SUMMARY

The main idea of the macromodular technology is the semantic description of programmed code. It is performs automatic assembly of optimal application for target software and a hardware platform. Thus the basis for assembly is debugged and optimized libraries. The software developer does not choice the most suitable library and does not write code to use that library.

The main ideas of macromodular technology of software development are:

- Standardization of rules of standard unit's usage (the unit name, assignment, the input data and received results). Standard implementation of units is not fixed, so it is defines abstract units. Definition of the abstract unit can be such as on Fig. 7 (on an example of the data ordering). Standardization of abstract units is regulated and it is reported to software developers.
- The extension of standard modules by the available units in libraries (for each unit of library the abstract unit is indicated). Generally, for one abstract unit there can be some implementations in various libraries of standard units (Fig. 8).
- Modular development of programs with usage of abstract units (thereby, the developer should know only definition of abstract units, instead of set of their various implementations).
- Automated choice of implementations of abstract units used in programs. Automation process include the analysis of libraries available in the environment and a choice of the best implementation of abstract units from available set, usage of concrete implementations of abstract units and construction of a final version of programs.

The main advantages of the offered technology are:

- Standardization of the standard units, selected in development process of programs.
- Essential decreasing of practical usage complexity of all set of various implementations of standard units.
- Considerable decreasing of software development complexity – at accumulation of sufficient size of the standardized descriptions of computer data processing.

Appreciable improvement of programs portability between various hardware-software platforms, the localized implementations of programs can have high working speed in the presence of the effective-developed programs.

Unit name: SortData
Purpose: Data ordering
Input data:
- Number of the ordering data
- Initial data

Results:
- Ordered data

Figure 7. Abstract unit definition of a data ordering

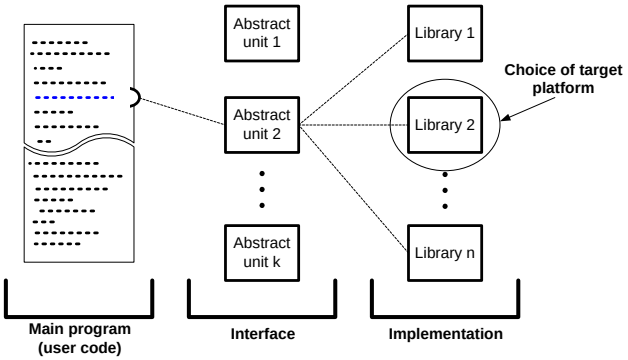

Figure 8. Common view of macromodule technology

REFERENCES

[1] Knuth, D.E. The Art of Computer Programming Volume 2: Seminumerical Algorithms. Addison-Wesley Professional; 3 edition. November 14, 1997. pp. 501.

[2] John R. Gilbert, Cleve Moler and Robert Schreiber. Sparse matrices in MATLAB: Design and Implementation. SIAM Journal on Matrix Analysis and Applications 13 (1), 1992, pp. 333–356.

[3] R. Chandra, R. Menon, L. Dagum, D. Kohr, D. Maydan, J. McDonald, Parallel Programming in OpenMP. Morgan Kaufmann, 2000.

[4] NVidia CUDA. Reference Manual. Version 2.3. July 2009.

[5] Intel® Math Kernel Library. Reference Manual. September 2007.

[6] Craig S., Marc Y., Brian J. Working with Microsoft® Visual Studio® 2005. Microsoft Press.