Numerical simulation system for parallel computing

R. A. Eskin[1], I. L. Artemieva[2]

[1]Institute of Applied Mathematics, Far-Eastern Branch of the Russian Academy of Sciences, 7 Radio Street, Vladivostok, Russia
[2] Far Eastern Federal University, 8 Sukhanova Street, Vladivostok, Russia

aginor@inbox.ru, iartemeva@mail.ru

Abstract. *Mathematic modeling is used in many fields of science and technology to create a mathematical model that is a base to computer model of a physical process. Computer modeling of real processes requires large scaled calculations, therefore solution of calculation acceleration problem is an urgent problem. The conception of parallel program system for numerical simulation with support for extendable set of numerical solution methods is proposed. Key part of the system is language processor to translate high-level problem specification into low-level source code for multiprocessor computer. The architecture of language processor and the idea of applicable solution method search algorithm are described. The target standards for output source code are MPI and OpenCL.*

Keywords

Parallel computing, GPGPU, MPI, OpenCL, numerical simulation.

1 Introduction

Nowadays modeling is used in many fields, such as atmosphere pollution analysis, weather forecast, space research and etc. To obtain results of mathematical modeling by a computer it is necessary to transform a mathematical model into a computer model that is a computer program using a numerical methods to calculate.

Computer modelling requires large scaled computations. For example, one experiment with global atmosphere model requires execution of about 10^{16}-10^{17} arithmetic operations with floating point [1]. Assuming that during research many experiments are required, it is evident that computing acceleration is actual problem.

It is obvious that now parallel computing is the only way to improve computer system performance. Further speed improvement only at the expense of photolithography process improvement is becoming impossible. Three major physical barriers are the reason of this: light, temperature and quantum [7]. Besides usual multiprocessor systems, usage of graphics processors allows us to obtain significant results (for example, up to 115 times performance improvement comparing with single processor in [3]). Other advantages are low power consumption, low cost, high availability, and rapid performance growth. On the other hand, program creation for GPU is even more complicated than for traditional multiprocessor systems.

To obtain full performance advantage of parallel computer systems, significant modifications of sequential program are needed, that require special skills (like detail understanding of platform architecture, etc). Program adaptation for parallel computing is laborious and complicated process, distracting resources from researches directly.

Imperative languages complicate the process of parallelization because of thorough dependency analysis necessity. Declarative approach in turn has proved itself useful in such languages as NORMA[5] and REPRO-S[6]. The main idea of declarative languages is to minimize the programming difficulties for a person, who is more proficient in subject area than in low level programming languages like C++.

Though modern declarative languages allow to reduce effort on model creation, they usually oblige to convert mathematical model to predefined representation (like finite-difference scheme) or limit range of solution methods or possible problems to solve. On the other hand, computer algebra systems allow to specify model in mathematic terms. But they don't allow to achieve high performance because of using symbolic computation and limited support of parallel methods. So, next evolution step may be a system, which allow to specify problem in mathematical model terms and which can take full advantage of parallel computing for extensible range of solution methods.

2 System conception

System core is language processor, which aim is to convert high-level model into source code, written in low-level programming language for parallel system. Translation process consists of three steps represented in figure 1. First step is lexical and syntactic analysis of high level model. The result of this step is the first internal representation model. Second step is problem analysis and selection of appropriate solution scheme from conversion bank. The selection of solution scheme depends on problem properties (like dimension). Second step output is the second internal representation model. Second internal representation model is a numerical scheme, written in abstract programming language, which contains minimum expressive tools for describing a parallel program. Third step is code generation for target parallel platform, basing on the second internal representation model.

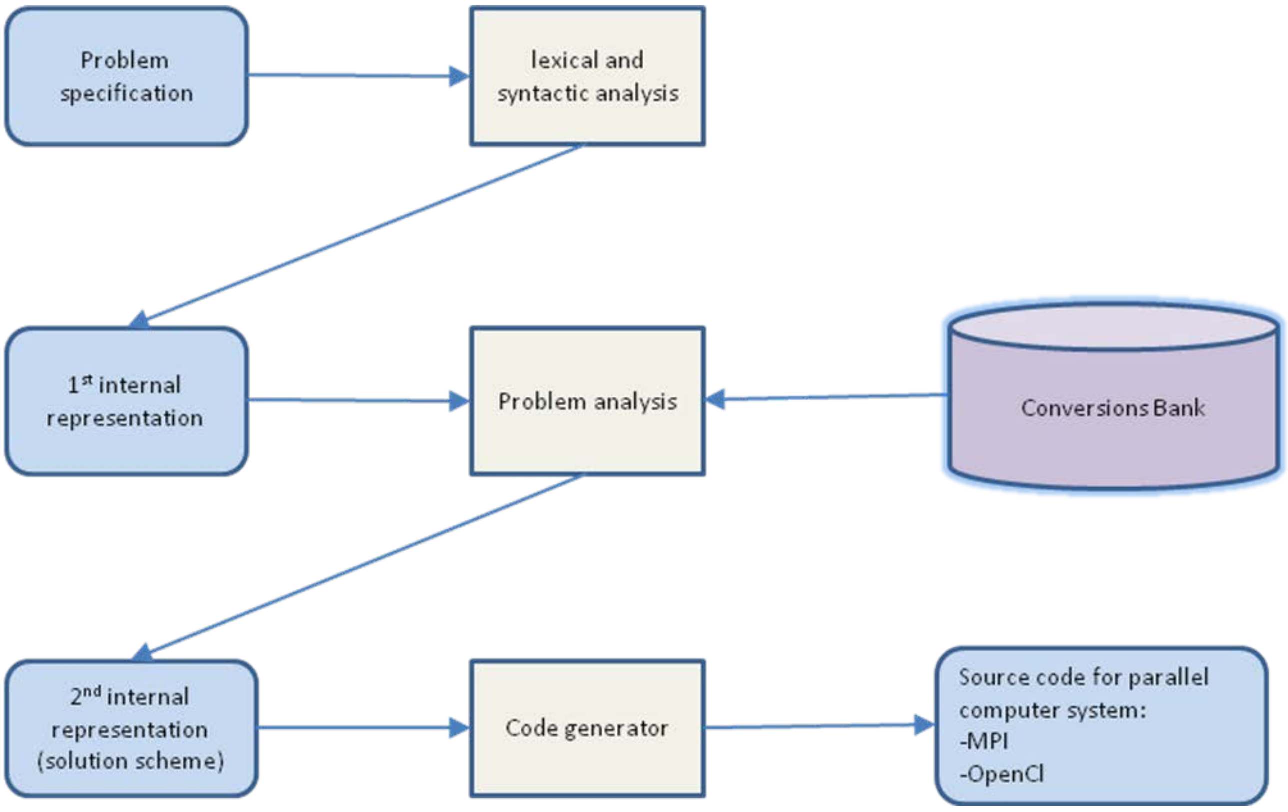
Problem specification
lexical and syntactic analysis
1st internal representation
Problem analysis
Conversions Bank
2nd internal representation (solution scheme)
Code generator
Source code for parallel computer system:
-MPI
-OpenCl

Fig. 1. Language processor architecture

Support for extendable set of numerical solution methods is planned. It will be achieved through the conversion bank (mentioned above). Efficiency of each specific solution method depends on the problem, which is currently being solved, and the opportunities of available computing system. The aim of conversion is to put in correspondence the most effective solution method to the problem. Fox example, to solve high-dimensional problem on parallel system with limited inter process communication, most effective will one of Monte Carlo methods [2][4]. Conversion consists of context condition and a set of transformations. A problem can have multiple methods of solution, each method matches one transformation. Context condition represents a problem scheme, it is defined in terms of abstract syntax as a set of formulas, their conjunction after scheme interpretation defines the applicability of conversion. Transformation represents problem solution scheme, it is defined in abstract programming language, which is simple enough to ensure effective translation into any specific low-level programming language for parallel system, and, on the other hand, it provides enough tools to express any parallel numerical scheme. Some names of variables and functions may be abstract symbols, that will be concretized after interpretation of context condition (abstract names of transformation are a subset of abstract names of context condition). Selection of transformation from a set of transformation is done basing on properties of current problem.

As target interfaces for parallel systems MPI (Message Passing Interface) and OpenCL (Open Computing Language) are proposed. It will allow to effectively utilize computing capabilities of traditional multiprocessor systems and graphical accelerators.

3 Conversion matching

The innovation in the language processor architecture is the “Problem analysis” block. Its responsibility is to perform conversion matching – selecting the applicable conversion from conversion bank, and compose numerical solution scheme from this conversion and problem specification.

Selecting of applicable conversion from conversion bank is a task of enumeration of possibilities. Context condition of each conversion in a bank has to be interpreted and matched with task specification. If matching after interpretation is successful, then the conversion is applicable. Interpretation and matching is done using labeled parse trees of problem specification and context condition. Tree labeling contains information about dependencies of objects. For example, expression (2*t+x) is anonymous object depending on names t and x, and expression y(t) is named object, depending on t.

Interpretation and matching algorithm (not-formal description):

Input of algorithm are labeled parse tree of problem specification (treePS) and labeled parse tree of context condition (treeCC).

Output is substitution of context condition names into problem specification names, which defines the interpretation of context condition.

Algorithm steps:

1. find sub-tree of treePS, satisfying following conditions: 1. the sub-tree includes root 2. if non-terminal node belongs to this sub-tree, then all its neighbors belong to this sub-tree 3. sub-tree structure is matching treeCC. Designate this sub-tree as R.
2. find all possible substitutions of context condition names into problem specification names. Designate the set of all possible substitutions as P.
3. get substitution $p \in P$, compare treeCC after substitution p with R. If trees are equal, then algorithm succeeded, p is the output.
4. if there are no more elements in P, than algorithm failed.

Step 2 of algorithm can be reduced to the task of recognition of graph isomorphism (NP class problem). All names with dependencies from problem specification represent a directed graph (names are vertices, dependencies are arks). An isomorphic graph with vertices – names from context condition – will represent a substitution of context condition names into problem specification names.

Once the context condition interpretation is found, 2nd internal representation can be composed. It is done by applying the interpretation to transformation.

References

[1] V. V. Voevodin, Vl. V. Voevodin: Parallel computing, 608, 2002.
[2] M. E. Zhukovsky, S. V. Podolyako, R. V. Uskov: Simulation of electron transfer in material on hybrid computer systems. *Computational Methods and Programming*, 152-159, 2011.
[3] M. E. Zhukovsky, R. V. Uskov: About usage of graphical processors of video accelerators in application tasks. *Preprints IAM RAS. URL: http://library.keldysh.ru/preprint.asp?id=2010-2*, 23, 2010.
[4] S. A. Maidanov: General approaches for developing parallel Monte Carle methods. *High-performance parallel computing on cluster systems: Proceedings of International scientific-practical seminar*, 103-111, 2002.
[5] D. V. Osipov: Integrated environment for parallel programs creation on very high-level programming language NORMA. *High-performance parallel computing on cluster systems: Proceedings of seventh International scientific-practical seminar*, 272-277, 2007.
[6] M. B. Tyutyunnik: Development and research of parallel programming production system. *PhD thesis*, Vladivostiok. 158, 2010. (in Russian)
[7] O. A. Yufryakova: Barriers on unlimited scaling of parallel systems. *Research service in the Internet: exaflop future: Proceedings of the International Supercomputer Conference*, 267-270, 2011