Optimization methods of parallel execution of numerical programs in the LuNA fragmented programming system

Victor Malyshkin · Vladislav Perepelkin

Published online: 13 July 2011
© Springer Science+Business Media, LLC 2011

Abstract The organization of high-performance execution of a fragmented program has encountered with the problem of choosing of an acceptable way of its execution. The potentialities of optimizing the execution at the stages of fragmented program development, compilation and execution are considered. The methods and algorithms of such an optimization are proposed to be included into the LuNA fragmented programming language, compiler, generator and run-time system.

Keywords Parallel programming · Fragmented programming · High performance computing · Program execution optimization

1 Introductory definitions and relative works

The idea of data and algorithm fragmentation has been exploited in programming, at least, since the early 1970s [2–4, 7, 9, 12, 14, 15]. Different modifications of this approach were embodied in programming systems [3, 4, 14]. Many programming systems use the run-time systems for the organization of computation [1, 2, 5, 6, 8, 9, 12]. In [14], instead of a commonly used run-time system for organization of the program execution, a special hardware and operating system were developed. Our LuNA fragmented programming system project is oriented to the creation of a parallel numerical subroutine library.

A general model of a program in the above-mentioned systems can be described as computational model [15].
1.1 General model definition

Given:

- the finite set $X = \{x, y, \ldots, z\}$ of variables for representation of different computed values;
- the finite set $F = \{a, b, \ldots, c\}$ of functional symbols (operations, Fig. 1a), $m \geq 0$ is the number of input variables, $n \geq 0$ is the number of output variables;
- $\text{in}(a) = (x_1, \ldots, x_m)$ is a set of input variables, $\text{out}(a) = (y_1, \ldots, y_n)$ is a set of output variables (Fig. 1), if $i \neq j \Rightarrow y_i \neq y_j \& x_i \neq x_j$.

Model $C = (X, F)$ is called simple computational model (SCM). Operation $a \in F$ describes the possibility to compute the variables $\text{out}(a)$ from the variables $\text{in}(a)$, for example, with the use of a certain procedure. The model can be graphically depicted (Fig. 1).

Let $V \subseteq X$, $F \subseteq F$ be given. A set of functional terms $T(V, F)$ is defined as follows:

1. If $x \in V$, then $x$ is a term $t$, $t \in T(V, F)$; $\text{in}(t) = \{x\}$; $\text{out}(t) = \{x\}$.
2. Let $\{t^1, \ldots, t^s\} \subseteq T(V, F)$ and $a \in F$, $\text{in}(a) = (x_1, \ldots, x_s)$ be given. The term $t = a(t^1, \ldots, t^s)$ is included into $T(V, F)$ if $\forall i(x_i \in \text{out}(t^i))$, $\text{in}(t) \bigcup_{i=1}^{s} \text{in}(t^i)$, $\text{out}(t) = \text{out}(a)$. Here $t = a(t^1, \ldots, t^s)$ denotes that $t$ is the term $a(t^1, \ldots, t^s)$.

A term is depicted as a tree that contains both operations and variables of the term, see Fig. 2.

We say that a term $t$ computes a variable $y$ if $y \in \text{out}(t)$. A set of terms $T(V, F)$ defines all the variables of the SCM that can be computed from $V$ variables. A set of terms $T^W_V = \{t \in T(V, F) | \text{out}(t) \cap W \neq \emptyset\}$ computes all the variables from $W$ that can be computed from $V$ variables.