Decentralized control of parallel computing[1]

Gennadi Malaschonok and Evgeni Ilchenko

Abstract. For one class of parallel computational algorithms, we describe the mechanism of control, which has no centralized management.

The problem of managing the computing process in the SPMD cluster, which has a shared memory is one of the major problems in the theory of parallel programming.

In this paper we consider one class of computational algorithms. The computational process of these algorithms is described by the tree graph. To get the result you can double-pass through the tree. First, you need to pass from the root to the leaves, and then in the opposite direction. It is assumed that the computational process can be described as a weighted tree. Natural weight of the edges represent the order of calculations in the subtrees, which defined by these edges.

For this class of computational algorithms, we describe the mechanism of control, which has no centralized management. Each node of the graph corresponds to a subtask. Each processor is assigned a subtask. This assignment occurs dynamically. We provide an automatic redistribution of vacated processors to solve problems in the other vertices of tree. This delivers a uniform load of processors in the cluster.

This problem was considered earlier in [1]. We describe a new, somewhat simplified version of the algorithm for this control scheme, assuming that all the outgoing edges of a vertex have the same weight and can be computed in parallel. However, it should be noted that this is the first version of the algorithm with decentralized management, which was implemented and experimentally investigated.

SPMD program is based on the two processing threads per processor. They are called the Computing thread and control thread.

For each thread has been developed corresponding class. And one more class, Task, has been developed for the algorithms of the tree nodes , which provides computational task.

[1]This work was partially supported by the Russian Foundation for Basic Research (grant No. 12-07-00755, 12-01-06020).

The mechanism of redistribution of the load based on the following fields, which are available on each processor in a control thread: List of Free Processors, List of Daughter Processors, List of numer of recent issues, which are referred to a child and the Number of the Parent Processor. The third list is only needed to resolve conflicts of bilateral asynchronous processors with its subsidiaries.

At the begining of calculations the root processor has List of Free Processors, which contains all the processes of the cluster, and it has the input data of the whole problem.

Control thread has three states:

0) the expectation of the problem;

1) calculation of its subtree, or the distribution of free processors;

2) completion of the work and data transmission to the parent node.

The process number 0 (root) starts in state 1, the others are in standby mode (the state 0).

Control thread performs the following steps.

1) Get the number of free processors from the parent node.

2) Identify the completion status of the child nodes.

3) Send the list of free processors to the child nodes.

4) Send data to the computing thread if it is free.

5) In the absence of child nodes generates a partition of an existing problem on the parts, creates the child nodes, and transfer them their subtasks and lists of free nodes.

6) Changes to the Mode 2 (shutdown) when completed calculations in child nodes and the computing thread is free.

Experiments.

Experiment on the multiplication of two matrices was carried out on 10 processors. Each processor can multiply several rows of the first matrix to the second matrix.

For a matrix size of 100x100 the distribution of rows turned out uneven: 20, 17, 18, 8, 7, 4, 4, 8, 10, 4.

For a matrix size of 1000x1000 the distribution of rows has turned more uniform:57, 51, 54, 42, 47, 39, 32, 43, 45, 48.

These results are in good agreement with the picture of improving the uniformity of loading with an increase in the size of the problem, which usually show a good parallel program.

References

[1] Malashonok G.I. Control of parallel computing process. Tambov University Reports. Natural and Technical Sciences. V. 14, part. 1, 2009. P. 269-274.

Gennadi Malaschonok and Evgeni Ilchenko
e-mail: malaschonok@gmail.com, ilchenkoea@gmail.com