Distributed Static Mapping and Dynamic Load Balancing Tools under PVM

L. Hluchý, M. Dobrucký, D. Dobrovodský

Institute of Computer Systems, Slovak Academy of Sciences
Dúbravská cesta 9, 842 37 Bratislava, SLOVAKIA
e-mail: upsyhluc@savba.sk

Abstract: *This paper describes the static and dynamic task allocation tools in PVM environment for distributed memory parallel systems. For the static mapping the objective function is used to evaluate the optimality of the allocation of a task graph onto a processor graph. Together with our optimization method also augmented simulated annealing and heuristic move exchange methods in the distributed form are implemented. For dynamic task allocation the semidistributed approach was designed based on the division of processor network topology into independent and symmetric spheres. Distributed static mapping (DSM) and dynamic load balancing (DLB) tools are controlled by user window interface. DSM and DLB tools are integrated together with software monitor (PG_PVM) in GRAPNEL environment.*

Optimal planning of parallel program execution in a distributed memory parallel computer (DMPC) solves the response speed. Optimal allocation comes out from the assumption that the program execution time depends upon uniform load of the processors and upon interprocessor communication minimization. In this paper, our attention is concentrated on the diffusion method for static mapping and on the semidistributed approach for the dynamic task allocation. In our distributed static mapping tool also augmented simulated annealing and heuristic move exchange methods are implemented [1], [2]. To specify an appropriate optimization goal it is necessary to create a cost function, which provides a realistic evaluation of the communication and computation overhead. For the given graphs *S* (task graph) and *H* (hardware graph), mapping M, the general form of the cost function *CF* is proposed as the combination of two parts - so called **vertex cost function** F_{vertex} and **edge cost function** F_{edge}:

$$CF(M,t) = F_{vertex}(M,t) + F_{edge}(M,t)$$

where variable *t* is duration of the iteration step of the mapping algorithm. $F_{vertex}(M,t)$ expresses the effect of computation loads so that individual physical processors would be loaded in a uniform way. $F_{edge}(M,t)$ expresses the effect of communication volume on allocation so that allocation should ensure the minimum of external (interprocessor) communications.

The principle of our mapping approach [3] can be represented by the following procedure: In the starting condition all tasks are located on the root node of the DMPC. The tasks are transferred by centrifugal force f_1 (following from the requirement of uniform processor load), against which centripetal force f_2 is acting, trying to keep the processes with mutual communication as close to each other as possible. This way, uniform load is obtained and this method can be considered as an improvement of the "pure" load balancing method. The resulting sum of forces is a vector with components for each task stored in a node and directed to each communication link. A task and a direction with maximum (positive) value is chosen. The following relation is the definition of changes in the cost function, where $\alpha(t)$=const. and function $g(t)$ hides the weight coefficient (where *t* is the duration of iteration step):

$$\Delta CF(M) = \alpha(t).\, f_1(t) + g(t).\, f_2(t)$$

For dynamic load balancing we have chosen the compromise between centralized and distributed approach [4]. The set of processors is divided into independent symmetric regions, called spheres. In each sphere, a processor equidistant from all other processors is selected as the scheduler for that sphere. Cumulative load information for each sphere is exchanged among the independent schedulers. Using knowledge about load of local nodes and information from other schedulers, the scheduler can place a requested task on the appropriate local node or transfer it into a less loaded sphere. Dividing the set of nodes into spheres considers granularity of the tasks. In the problems with small granules, the average lifetime of the process is short. Tasks arise and extinct more frequently than those in the problem with big granules. So the sphere served by one scheduler should be smaller in a way to avoid overload of the central node.

Block scheme in Fig.1 represents the integration of GRAPNEL visual programming tool with DSM & DLB tools and with software monitor PG_PVM (the measurement of load and communication costs). The details about the GRAPNEL, GRP2C, GRP file blocks are described in [5], [6], [7]. The PVM loader represents the process which starts an application according to the options (with or without Dynamic Load Balancing (DLB) and applies mapping vector. In the case if DLB is chosen, PVM loader at first starts DLB main process and sends to it the mapping vector generated in DSM tool. After starting the DLB process, PVM loader starts the application main process.

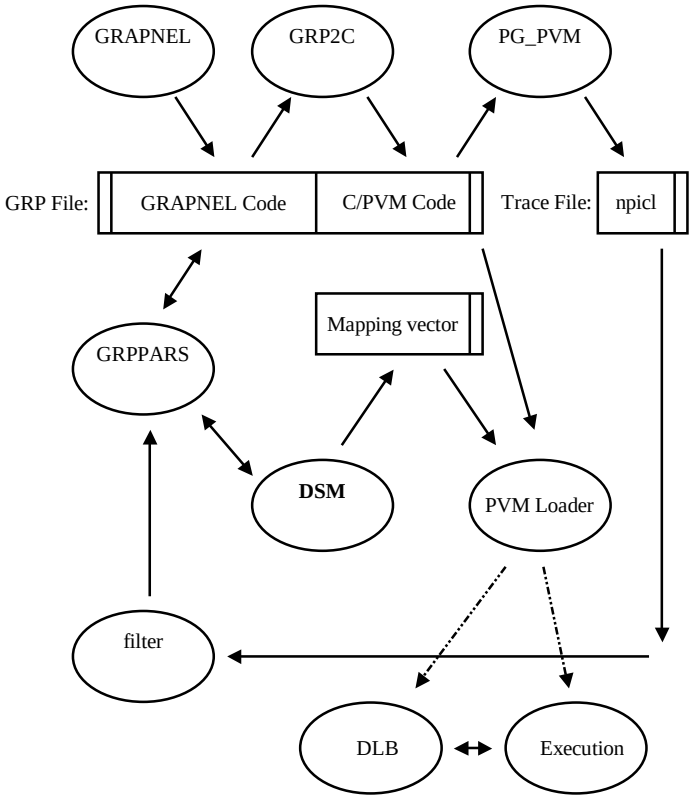

Fig.1. Block scheme of the integration DSM & DLB tools with visual programming tool

References

[1] KVASNICKA V., POSPÍCHAL J., BISKUPIC S.: Task Allocation Problem Solved by Augmented Simulated Annealing. Central European Journal for Operation Research and Economics, 1994.

[2] SELVAKUMAR S., MURTHY S.R.C.: An efficient heuristic algorithm for allocation parallel programs onto multicomputers. Microprocessing and Microprogramming, Vol.36, 1992/93, pp. 83-92.

[3] HLUCHÝ L., DOBRUCKÝ M., DUDÁK M.: Solving Method for Optimal Load Balancing and Communication Minimisation. In: Plander I. (Ed.): Proceedings of the Sixth International Conference on Artificial Intelligence and Information - Control Systems of Robots, World Scientific 1994, pp.297-302.

[4] AHMAD I., GHAFOOR A.: Semi-Distributed Load Balancing for Massively Parallel Multicomputer Systems. IEEE Trans. on Software Engineering, Vol. 17, No. 10, October 1991, pp. 987-1004

[5] High Perfomance Computing Tools for Industry, Contract N^{o} CP-93-5383, Progress Report No1, April 1995.

[6] High Perfomance Computing Tools for Industry, Contract N^{o} CP-93-5383, Progress Report No2, October 1995.

[7] High Perfomance Computing Tools for Industry, Contract N^{o} CP-93-5383, Progress Report No3, April 1996.