Automatic Configuration of Parallel Programs for Processor Networks

L. Hluchý, D.Dobrovodský, M. Dobrucký

[#] 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 mapping algorithm for distributed memory, parallel message passing systems using the objective function to evaluate the optimality of the mapping of a task graph onto a processor graph. Our optimization method is compared with other ones, some of them issuing from artificial intelligence or operations research. In our experiments randomly generated tasks and processor graphs are used.*

Keywords: *static and dynamic mapping, multicomputer, scheduling, load balancing, smoothing.*

1 Introduction

Optimal planning of parallel program execution in distributed environment of a multicomputer on the basis of message passing solves the response speed. Static and dynamic mappings can be available and effective way of solution. The theory of optimal mapping 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.

For static mapping a parallel program can be represented by a task graph $S(V,E)$, vertices $V=\{1,2...q\}$ represent the program tasks and edges E represent mutual communication. The topology of available technical resources can be represented by processor interconnection graph $H(C,E_p)$, vertices of which, $C=\{1,2,...p\}$, represent processors, and edges E_p represent communication channels or links. We assume the homogenous structure of the multicomputer system. The communication cost depends on message size and on distance between source and destination. Interprocessor distance d_{ij} between processors i and j is given by routing rules, i.e. as minimal number of hops from processor i into the processor j. Clearly, if $d_{ij}=0$ then $i=j$. Hardware graph detection includes methods for system inspecting and diagnostics (worm programs, seeking connections and node parameters in the structure). Many optimization methods have been created to solve the mapping problem, [1 - 9]. Their basic principles are common with the method used in other areas of research.

2 The Cost (Objective) Function Definition

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 and H, mapping M, and routing RA, the general form of the cost function CF can be built as a combination of three parts - so called **vertex cost function** F_{vertex} , **edge cost function** F_{edge} and the **routing cost function** F_{rout} :

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

[#] *Funded by Commission of the European Communities DG12 under contract number CIPA-CT93-0251 and Grant Agency for Science of Slovak Academy of Sciences*

Variable *t* is duration of the iteration step of the mapping algorithm. *Fvertex(M,t)* expresses the effect of computation loads so that individual physical processors would be loaded in a uniform way. *Fedge(M,t)* expresses the effect of communication volume on mapping so that mapping should ensure the minimum of external (interprocessor) communications. *Frout(M,t,RA)* expresses the effect of imbalance of physical communication links. Our contribution describes the solution of two phase cost function optimization: load balancing with communication minimization and then traffic load balancing.

3 The Diffusion Method for Static Mapping

The principle of our mapping approach can be represented by the following procedure [10]. In the starting condition all tasks are located at the root node of the multicomputer network. The tasks are transferred by centrifugal force *f1* (following from the requirement of uniform processor load), against which centripetal force *f2* 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:

$$\Delta CF(M) = \alpha(t) \ . f1(t) + g(t).f2(t)$$

where *t* is the duration of iteration step. The total number of iterations must be sufficient for location stabilisation. This number is obtained from the experiments and is a trade-off between the time consumption and the result accuracy. We estimate this number from *q* - the number of tasks to be mapped. Standard- nominal value was estimated as $^{7q}/_{2}$.

The function *g(t)* that we use was obtained from programming experiments in the transputer environment for some applications. Our experiments aimed at looking for suitable face of function *g(t)* (linear, nonlinear, increasing and decreasing intervals,...) and the number of iterations. The function *g(t)* is suggested to have the following properties: During the first *n* iterations it can be kept near zero or increases slowly from zero. The reason is that all tasks are located at the same processor at the start of mapping, and there is a need to keep the communication force low to enable task distribution. In the *n*-th iteration step *g(t)* reaches the value where the influence of *f1* and *f2* is comparable. Then *g(t)* decreases to zero, and thus the influence of communications becomes weaker. The algorithm can be implemented in the distributed environment as well.

For the traffic load balancing (smoothing) as the second phase of cost function optimization we used simulated annealing. The algorithm consists of the following steps:

1. Creation of the shortest path table on the basis of the algorithm for the shortest paths. We used the matrix algorithm [11]. The input for this algorithm are hardware and software graphs.
2. Creation of the routing table which is at the begining initialized to the shortest path table.
3. There is a randomly chosen path and randomly chosen processor on it and then is randomly chosen neighbouring processor and this new state is evaluated under rule of simulated annealing. Cost function is represented by relation:

$$Frout(M,t,RA) = \gamma(t).\sum_{li=1}^{T}\left|AD_{li} - \overline{AD}\right|$$

AD_{li} means the amount of data transferred by link *li*, $\overline{AD}$ stands for average amount of data per link. The *RA* variable expresses the dependence of *Frout* upon the directing algorithm selected specifying the message delivery from the source to destination. The change following from this evaluation is written down in the routing table.

4 Experimental results

We have compared (Fig.1 - 3) diffusion method with methods given in [4] - iteration method (Boillat), [2] - heuristic move-exchange (HME) and [1] - simulated annealing (SA). Model calculations were performed for nonregular graphs and grids. Graphs were generated by random graph generator. The software graph size was limited by 160 nodes (tasks), hardware graph size was limited by 64 nodes (processors), connected in transputer-like network (each node max. four neighbours). The cost for path length was given as follows: 0 as the cost for communication on the same processor, 1 as the cost for communication between direct neighbours and *2n* as the cost for communication across n-1 intermediators.

The augmented simulated annealing method was specified by T_{max} = 0.5, T_{min} = 0.05, *decrease* = 0.8. Number of moves for each temperature was *3pq*, where *p* is the number of processors and *q* is the number of tasks to be mapped. The length *L* of queue for HME was 3, the number of random moves *R* was proportional to the number of mapped tasks (*2q*).

The number of iterations for Boillat and diffusion methods was selected as the results of the experiments. The influence of the number of iterations on the quality of solution for these two algorithms was compared. Results obtained for Boillat algorithm showed that saturation of solution quality can be achieved when the number of iteration is equal nominal value $NOM = {}^{7q}/_{2}$, where *q* is number of tasks to be mapped. For diffusion algorithm we can observe this saturation when the number of iteration equals ${}^{NOM}/_{4}$, so to obtain the comparable results it is sufficient to choose the number of iteration ${}^{NOM}/_{4}$ or *q*.

Fig.4 shows the comparison of traffic load balancing; *SM* means smoothing, *SP* means routing on the shortest paths.

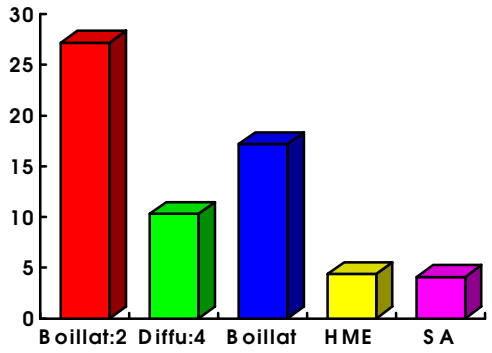

Figure 1: Comparison of load imbalances [%]

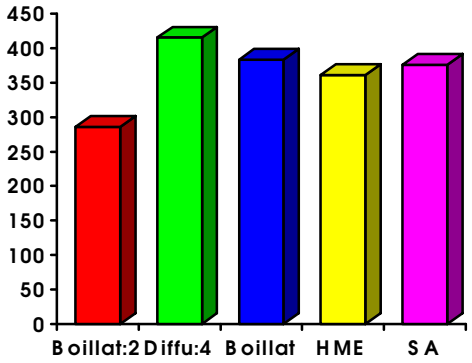

Figure 2: Comparison of communication costs

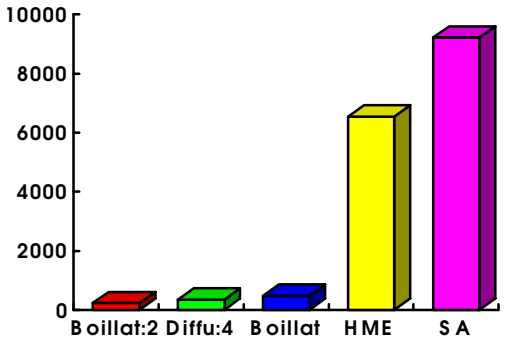

Figure 3: Comparison of time consumption [s]

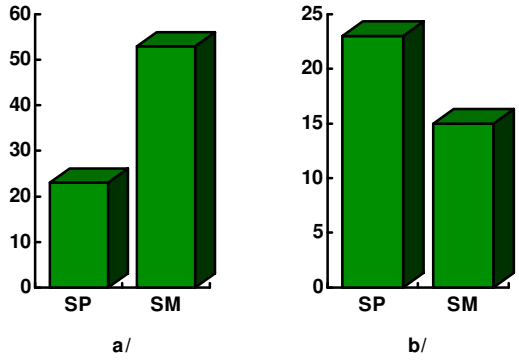

Figure 4: Influence of SM on the path length (a) and communication imbalance (b) [%]

5 Conclusion

Beside the algorithms given above, we have analyzed also other approaches: - modified algorithm [3] and genetic algorithms. Our aim was not to evaluate the best optimization strategy for static mapping only, but also to show which methods can be used in dynamic load-balancing. Further, our attention will be concentrated on the comparison of static traffic load-balancing with the using adaptive routing algorithms. Together with parallel programs monitoring we are implementing the integrated programming tool for automatic development of parallel programs in the PVM environment.

References

[1] KVASNIÇKA, V. - POSPÍCHAL, J. - BISKUPIÇ, S.: Task Allocation Problem Solved by Augmented Simulated Annealing. Central European Journal for Operation Research and Economics, in print.

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

[3] BOKHARI, S.H.: On the Mapping Problem. IEEE Transactions on Computers, Vol. C-30, 1981, No.3, pp. 207-214.

[4] BOILLAT, J.E. - ISELIN, N. - KROPF, P.G.: MARC: A Tool for Automatic Configuration of Parallel Programs. In: Welch, P. et. al. (Eds.): Transputing 91, IOS Press 1991, pp.311-329.

[5] van LAARHOVEN, P.M.J. - AARTS, E.H.L.: Simulated Annealing: Theory and Applications. Reidel, Dordrecht, 1987.

[6] BOLLINGER, S.W. - MIDKIFF, S.F.: Heuristic Technique for Processor and Link Assignment in Multicomputers. IEEE Transaction on Computers, Vol.40, 1991, No.3, pp. 325-333.

[7] BIANCHINI, R.P. - SHEN, J.P.: Interprocessor Traffic Scheduling Algorithm for Multiple-Processor Networks. IEEE Transactions on Computers, Vol.36, 1987, No.4, pp.396-409.

[8] LEE, S.Y. - AGARWAL, J.K.: A Mapping Strategy for Parallel Processing. IEEE Transactions on Computers, Vol.36, 1987, No.4, pp.433-442.

[9] COROYER, C. - LIU, Z.: Effectiveness of Heuristics and Simulated Annealing for the Scheduling of Concurrent Tasks-An Empirical Comparison. In: Bode, A.-Reeve, M.-Wolf, G. (Eds.): Proceedings PARLE'93. Springer 1993, pp.452-463.

[10] HLUCHÝ, L. - DOBRUCKÝ, M. - DUDÁK, M.: Solving Method for Optimal Load Balancing and Communication Minimization. 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.

[11] CHEN, W.K.: Theory of nets: flows in networks. John Wiley & Sons, Inc., 1990, 493 pp.