AN EFFICIENT MAPPING STRATEGY FOR PARALLEL PROGRAMMING

J.L. ORTEGA-ARJONA† and H. BENITEZ-PEREZ‡

† *Departamento de Matemáticas, Facultad de Ciencias, UNAM, México. jloa@ciencias.unam.mx*
‡ *Departamento de Ingeniería en Sistemas Computacionales y Automatización, IIMAS, UNAM, México. hector@uxdea4.iimas.unam.mx*

***Abstract*— Obtaining an effective execution of a parallel system requires that the mapping of the processes (of the parallel software) on the processors (of the parallel hardware) is efficiently performed. Hence, this paper presents an efficient mapping strategy based on optimizing communications between processes as well as load balancing process distribution onto an arbitrary processor network. Such a mapping strategy is developed as a parallel program, based on the simultaneous execution of local, independent processes. This fact contrasts with many other approaches for solving the mapping problem, like simulated annealing, heuristic search, and others, which require a centralized control for the mapping. In this paper, it is shown that the present mapping strategy is efficient enough when applied to two different mapping problems. Based upon an experimental setup, it is possible to review this mapping strategy following the related impact.**

***Keywords* — Parallel programming, mapping problem, mapping strategy.**

I. INTRODUCTION – MAPPING AND THE MAPPING PROBLEM

Mapping and the mapping problem are commonly defined as follows (Bokhari, 1981): "Suppose a problem made up of several modules that execute in parallel is to be solved on an incompletely connected array. When assigning modules to processors, pairs of modules that communicate with each other should be placed, as far as possible, on processors that are directly connected. We call the assignment of modules to processors a mapping and the problem of maximizing the number of pairs of communicating modules that fall on pairs of directly connected processors the mapping problem."

In topological terms, the mapping problem refers to find whether a graph that represents the system of communicating processes can be mapped onto a graph representing the processor network, so that neighboring processes are allocated on neighboring processors. Nevertheless, even though this seems simple enough, the mapping problem is known to be NP-complete (Bokhari, 1981). Therefore, it seems useless trying to propose exact algorithms for solving the mapping problem. Instead, only mapping strategies that are able to efficiently obtain suboptimal solutions have been proposed.

Let us consider that a parallel program is defined as the specification of a set of processes executing simultaneously, and communicating among themselves to achieve a common objective (Hoare, 1978). Based on this definition, a parallel software program can be represented in the form of a graph, in which each process is a vertex, and each communication between any two processes is an edge. In a similar way, a parallel hardware or processor network can be represented as a graph, in which now each processor is a vertex and each interconnection between any two processors is an edge. The mapping problem, hence, reduces to embedding the software graph into the hardware graph.

Nevertheless, since mapping should provide a certain distribution of the processes onto the processors so the most efficient execution is obtained; two optimization issues have to be considered:

- Load balancing. The processes have to be mapped onto the processors so the processing load caused by all processes is fairly distributed over all processors. In such a situation, the parallel system is considered to be balanced.
- Communication optimization. The communications between any two processes should be distributed as evenly as possible over all connections between processors. When this is the case, it is said that communications are optimal.

Regarding the second issue above, optimizing communications means that neighboring processes should be mapped onto neighboring processors. Otherwise, for any communication between any two processes, more than one connection has to be used. This produces a communication overhead due to communication re-emission, increasing overall execution time of the parallel program. Moreover, communication re-emission also causes load on the intermediate processors, since processing in needed for routing a communication until it reaches its final destination, and also tends to increase overall execution time of the parallel system. Nevertheless, it is commonly not possible to map every neighboring process onto neighboring processors. Thus, the processes should be allocated so that the overall communication costs are kept as minimal as possible.

A. The Mapping Problem: A Formal Description

For the current purposes, and for obtaining a more formal definition of the mapping problem, let us assume the following features of the parallel system:

- The parallel hardware platform has a distributed memory organization. Each processor has its own local, private memory.