An $O(\sqrt{n})$-Worst-Case-Time Solution to the Granularity Problem*

A. Pietracaprina ** *** and F.P. Preparata***

Abstract. In this paper we deal with the granularity problem, that is, the problem of implementing a shared memory in a distributed system where $n$ processors are connected to $n$ memory modules through a complete network (Module Parallel Computer). We present a memory organization scheme where $m \in O(n^2)$ variables, each replicated into $2c - 1$ copies (for constant $c$), are evenly distributed among the $n$ modules, so that a suitable access protocol allows any set of at most $u$ distinct read/write operations to be performed by the processors in $O(\sqrt{n})$ parallel steps in the worst case. The well known strategy based on multiple copies is needed to avoid the worst-case $O(u)$-time, since only a majority of the copies of each variable need be accessed for any operation. The memory organization scheme can be extended to deal with $m \in O(n^3)$ variables attaining an $O(n^{2/3})$-time complexity in the worst case.

Key Words. Algorithms and Data Structures, Theory of Parallel and Distributed Computing, P-RAM Simulation.

1 Introduction

A central problem in the design and implementation of a parallel computer is represented by the organization of data that must be available to the system's processors. More specifically, given $n$ processors, $n$ memory modules and $m \gg n$ data items (variables), a memory organization scheme is sought to distribute the data among the modules so that any subset of $n' \leq n$ variables can be efficiently accessed by the processors. The problem, often referred to as the granularity problem, has received considerable attention in the literature. The survey by [Kuc77] cites a number of works dealing with particular instances of the problem. More recently, the granularity problem has become a major obstacle in the development of efficient P-RAM simulation schemes [MV84, UW87, AHMP87, KU88, LPP88, Her89, LPP90, Her90, Ran91].

In order to focus on the essence of the problem, most authors assume that processors and memory modules are connected by a complete bipartite graph, and that, in a synchronous mode of operations, each memory module can fulfill at most one access (read/write) request per time unit (Module Parallel Computer (MPC)). With this strategy, the time to complete a parallel access to a set of variables is proportional to the maximum number of requests addressed to a single module. Mehlhorn

---

* This paper was partially supported by NFS Grant CCR-91-96152.
** Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61801
*** Department of Computer Science, Brown University, Providence, RI 02912
and Vishkin [MV84] first pointed out the difficulty of finding a deterministic memory organization that allows a parallel access to $n$ variables in time significantly less than the trivial $O(n)$ bound. They introduced the idea of replicating each variable into several copies so that a read operation needs to access only one (the most convenient) copy. For $m \in O(n^2)$, they present a memory organization scheme that uses $c$ copies per variable and allows a set of read requests to be satisfied in time $O(cn^{1-1/c})$. However, to execute write operations all the copies of the variables must be accessed, so requiring $O(cn)$ time.

Later, Upfal and Widgerson [UW87] refined the strategy based on multiple copies introducing to the realm of P-RAM simulation the majority concept previously developed for data bases [Gif79]: given $2c-1$ copies per variable, only $c$ of them need to be accessed by any (read or write) operation. They also proposed a new memory organization scheme, based on a bipartite graph with particular expansion properties, which, when $m$ is polynomial in $n$, attains an $O(\log n(\log \log n)^2)$ time bound. Unfortunately, no explicit construction for such a graph is known. With a probabilistic argument, the authors prove its existence, and prove that a random graph will have the desired expansion properties, with high probability. However, as we will show in this paper, no efficient way is known to test a graph for these properties.

All deterministic schemes subsequently developed for solving the granularity problem [AHMP87, Her89, LPP90, Her90] follow the ideas in [UW87] and are non-constructively based on the existence of similar expander graphs. (It must be pointed out that most of the cited works also considered the distribution of shared data on more realistic models where processors and memory modules are interconnected by bounded degree networks.) Using randomization, the granularity problem appears to be of easier solution as the results in [MV84, KU88, LPP88, Ran91] show. A common feature of these works is the use of a class of universal hash functions to distribute the variables among the memory modules. Moreover, randomized schemes do not need to replicate the variables since the worst case is avoided in the probabilistic sense.

In this paper we present a simple deterministic memory organization scheme for the MPC that distributes $m \in \Theta(n^2)$ variables among the $n$ memory modules so that any $n$ read/write operations can be performed in parallel in $O(\sqrt{n})$ worst-case-time. As in [UW87], each variable is represented by $2c-1$ copies only $c$ of which need to be accessed in a read/write operation. The assignment of the copies to the memory modules is governed by the structure of Balanced Incomplete Block Design, a well known incidence structure, which guarantees that for any two variables there exists at most one module containing copies of both. The construction of such a graph is possible by using simple group theory; moreover, the physical addresses of the copies can be efficiently determined by the processors. This scheme can be extended to accommodate $m \in \Theta(n^3)$ variables. In this case, a set of $n$ memory requests can be satisfied in $O(n^{2/3})$ worst-case-time.

It is important to underscore that in all the previous works a major problem was represented not only by the explicit construction of the memory organizations, but also by the storage of an adequate description of the memory map, these two aspects being closely interdependent. In fact, with the exception of [Her89, Her90], where the author is explicitly concerned with implementation issues, all other works assume that each processor has, in some form of local storage, a complete description