A COMPARISON OF THE PERFORMANCE OF THREE PARALLEL ARCHITECTURES FOR MATRIX-VECTOR MULTIPLICATION

Bruno CODENOTTI and **Chiara PUGLISI**

Istituto di Elaborazione del CNR
Via S.Maria 46, 56100-Pisa (Italia)

ABSTRACT

We present the parallel implementation of matrix-vector multiplication on a binary tree of a small number of processor, whose leaves are connected to local memories, each containing one column of the matrix. The performance attained can be favourably compared with the one of the mesh of trees and the linear array, each formed by the same number of processors.

KEY WORDS. Binary Tree, Mesh of Trees, Linear Array, Matrix-Vector Multiplication, Parallel Computation.

1. INTRODUCTION

Parallel algorithms for the solution of numerical problems have been developed since the sixties, although no parallel architectures had been constructed at that time (see, for example [2,5,6]). Today research on parallel algorithms can go from design and analysis to implementation.

This paper presents the description of several ways to carry out matrix-vector multiplication using parallel machines. Other work in the field has been performed in [3,4]. We study the performance of the linear array (see fig.1) and the mesh of trees (see fig.2), and we propose a column-oriented arrangement of the components of the matrix on a binary tree architecture (fig.3). The performance of this method can be favourably compared to the one of the known designs. All the architectures here considered have a fixed number of processors (i.e. independent of the size of the problem). The case of high parallelism has been analyzed in [1].

Matrix-vector multiplication plays a central role in numerical linear algebra, since it is the basic step of many algorithms. This problem can be efficiently parallelized, since the naive fan-in algorithm attains the logarithmic lower bound to the computation time. On the other hand, the problem arises of distributing data among the local memories of the processors. In fact, note that matrix-vector multiplication consists of a set of scalar products, of the type $\langle A_i , b \rangle$, where A_i denotes the i-th column of the matrix, b is the vector, and $\langle , \rangle$ denotes the scalar product. It is easy to see that one of the primary concerns of the problem is to

distribute the elements of vector b among different modules. The solution adopted in this paper consists of a column-oriented allocation of the components of the matrix among the leaves of the binary tree.

The computational model here considered is a slight modification of a theoretical model widely used in the literature [2]. Namely, we assume that

i) each processor can perform one arithmetic operation in a unit of time;
ii) no penalties occur for accessing memory (memory access is performed by a **get** statement);
iii) one interprocessor communication (**send** or **receive**) takes one unit of time;
iv) the computation is carried out on a given network of processors.

The measure of complexity we consider is the computation time T (i.e. the number of parallel steps).

This paper has the primary goal of giving some criteria to choose a convenient setting for a configurable network of few processors, in order to compute matrix-vector product. Throughout the paper, all logarithms are to the base 2, and we will assume n/p to be an integer. In the general case, all formulas hold by substituting n/p with $\lceil n/p \rceil$, where $\lceil x \rceil$ denotes the upper integer part of the real number x.

2. MAIN RESULTS

In this section, we study the time-performance of three parallel architectures for the computation of matrix by vector products. We assume that the number of processors p of each architecture is independent of the order n of the matrix.

Lemma 1. A linear array of **p** processors (denoted by P_i, i=1,2,...,p) can perform **nxn** matrix by **n**-vector multiplication in time

$$T_p^{LA}(n) = 4n^2/p - 2n^2/p^2 + 3n - 5n/p + p - 1,$$

where $3 \leq p \leq n/2$.

Proof. Let A be an nxn matrix, and b be an n-vector. Matrix A is partitioned into p^2 blocks of size (n/p)x(n/p). Each processor receives n/p rows of A. The i-th processor P_i receives a set of n/p rows (from index (i-1)n/p + 1 to index in/p), and processes them an (n/p)x(n/p) block at a time. In fig.4, partitioning of matrix A, and data-flow into the linear array are illustrated.

To evaluate the time-performance of the linear array, we proceed as follows. We count the first steps of P_1, before P_2 starts performing its computation, then we evaluate all the computation time spent in P_2, finally we consider the time elapsed between the end of the computation in P_2 and

the completion of the whole algorithm, as shown by the following schema.

Processor	Computation	Time
P_1	0. **get** A_{11} , b_1	0
	1. $x_1=A_{11}b_1$	$(n/p)(2n/p-1)$
	2. **send** b_1	n/p
	.	
	.	
	.	
P_2	0. **get** A_{21} , **receive** b_1	n/p
	1. $x_2=A_{21}b_1$	$(n/p)(2n/p-1)$
	2. **send** b_1	n/p
	3. for i:=2 to p	
	3.0. **get** A_{2i} , **receive** b_i	n/p
	3.1. $x_2=x_2+A_{2i}b_i$	$2n^2/p^2$
	3.2. **send** b_i	n/p
P_i, i=3,...,p-1		$(p-3)(2n^2/p^2+n/p+1)$
P_p		$2n^2/p^2+1$

The overall running time can be evaluated by adding steps 1 and 2 of P_1, all the computation performed by P_2 (except that step 0 of P_2, which is partially overlapped by step 2 of P_1), and the further steps needed after P_2 ends its computation. The above mentioned overlapping results in n/p + 1 steps instead of 2n/p steps.

The thesis readily follows. □

Lemma 2. A mesh of trees of **p** processors can perform **nxn** matrix by **n**-vector multiplication in time

$$T_p^{MT}(n) = 6n^2/p + n/\sqrt{p/3} + 5\log\sqrt{p/3} - 2,$$

where $8 \le p \le 3n^2-2n$.

Proof. Let $p=3k^2-2k$. Let A be an nxn matrix, and b be an n-vector. Matrix A is partitioned into k^2 blocks of size (n/k)x(n/k). Each processor of the mesh is provided with one block of A. The computation can be performed processing sequentially the rows within each block. n/k elements of vector b are distributed in the column-tree, then the scalar product of two vectors of size n/k is computed by the processor of the mesh, and finally the partial sums for the computation of one element of the product are accumulated in the row-tree.

The number of steps is given according to the schema:

n/k steps are spent by the root of the column-tree to fan-out n/k components of vector b;

2(log k - 1) steps are used to fan-out the n/k-th entry of vector b in the column-tree;

$2n^2/k^2 + 1$ steps are executed by the leaf, 1 step to receive data, (2(n/k)-1)(n/k) to compute matrix-vector product, and n/k to send data;

3(log k - 1) steps are executed by the processors of the row-tree, except that the root and the leaves. 2(log k -1) of these steps are dedicated

to communicate data, and log k - 1 to compute sums;

2 steps have to be spent in the root: 1 sum and 1 receive.

From these results, the thesis follows. □

Lemma 3. A binary tree of **p** processors can perform **nxn** matrix by **n**-vector multiplication in time

$$T_p^{BT}(n) = 4n^2/(p+1) + 3\log(p+1) - 4,$$

where $3 \le p \le 2n-1$.

Proof. Let p=2k-1. Let $A=(a_{ij})$ be an nxn matrix, and $b=(b_i)$ be an n-vector. Matrix A is partitioned into k groups of n/k columns. Each of the k leaves is provided with n/k columns of the matrix, and with the corresponding n/k components of the vector. Each leaf processes its columns, a row at a time.

The computation time can be evaluated according to the formulas:

Leaf i: Compute $f(s) = \sum_{j=(n/k)(i-1)+1}^{(n/k)i} a_{sj} b_j$, send f(s), s=1,2,...n.

These steps take $2n^2/k$ units of time. After the leaves carry out their computation, internal nodes have to perform $3(\log(n/k)-1)+2$ more steps.

The thesis follows. □

These lemmas allow proving the following proposition.

Proposition. Under the hypotheses and using the notations of Lemmas 1,2,3, we have:

$$T_p^{AB}(n) < T_p^{LA}(n) < T_p^{MT}(n),$$

for any $8 \leq p \leq n/2$.

Proof. Follows from Lemmas 1,2,3. □

REFERENCES

1. B.Codenotti and C.Puglisi, Matrix-Vector Multiplication: Algorithms and Architectures, submitted for publication (1988).

2. D.Heller, A Survey of Parallel Algorithms in Numerical Linear Algebra, SIAM Rev. 20, 740-777 (1978).

3. H.T.Kung and C.E.Leiserson, Systolic Arrays (for VLSI), in (Mead and Conway, Introduction to VLSI systems, Addison Wesley, 1979).

4. F.T.Leighton, New Lower Bound techniques for VLSI, Proc. IEEE FOCS, 1-12 (1981).

5. W.L. Miranker, A Survey of Parallelism in Numerical Analysis, SIAM Rev. 13, 524-547 (1971).

6. A.Sameh, Numerical Parallel Algorithms-A Survey, in High Speed Computer and Algorithm Organization, Academic Press, 207-228 (1977).

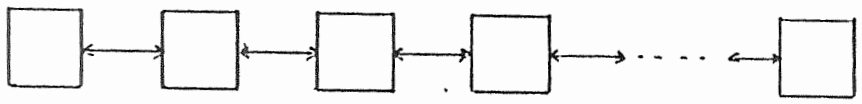


Fig. 1. A linear array.

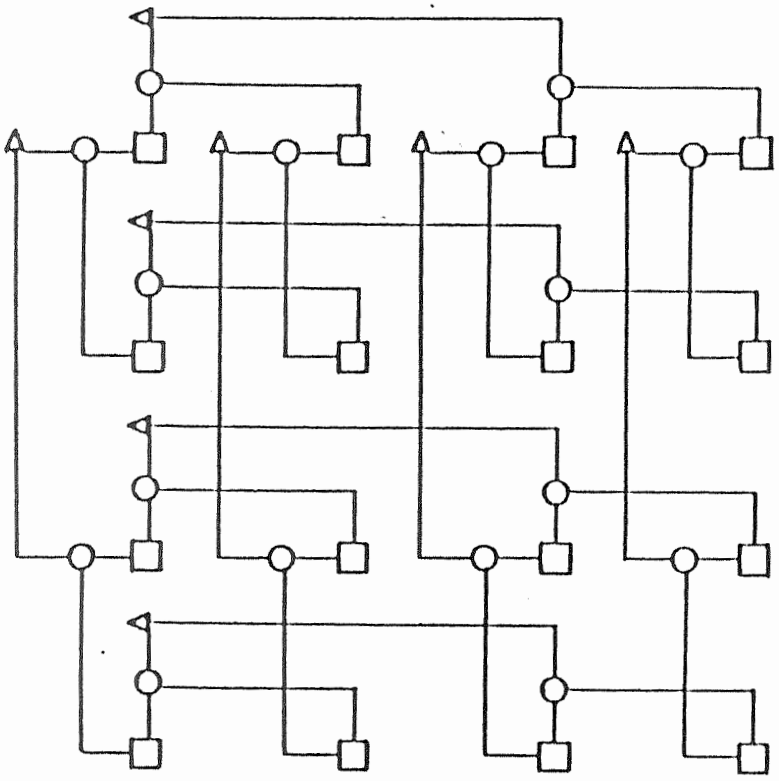


Fig.2. A mesh of trees.

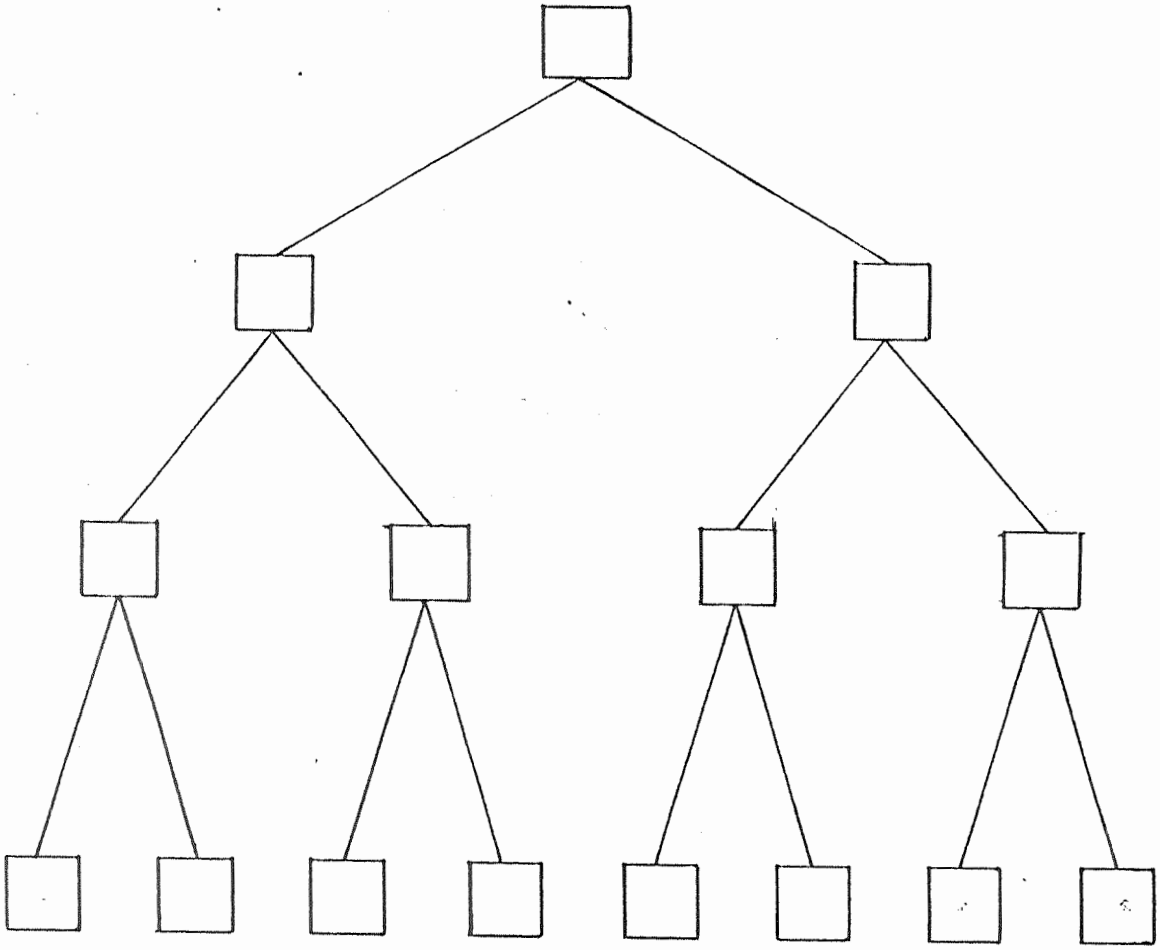
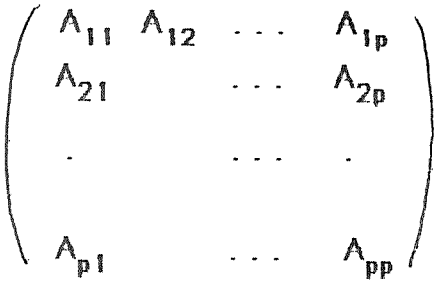
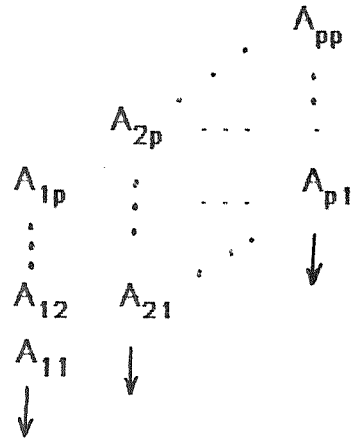


Fig.3. The binary tree.



a)



b)

Fig.4. a) Partitioning of matrix A;
b) Data Flow in the linear array.