CIS 629: Project No.5 MasPar Programming

Sameer S. Shende

December 2, 1994

1. **Matrix Transpose** The matrix transpose problem was solved by two ways viz., by using xnet and by using the router network. The timings achieved indicate that it the router algorithm is an order of magnitude faster than the xnet one in this particular implementation. While implementing it was noticed that xnet[] takes singular integers as index while router[] can take plural values too. This was exploited and the entire transpose took just two instructions in the router case while it had to go through a loop in the xnet case. If xnet[] also supported the plural index (every processor getting data from processors different hops away) then we could have realised much better performance in that case.

   The file transpose.m contains the two routines for xnet and router transpose functions in transpose_matrix_xnet() and transpose_matrix_router() functions. The code listing and script file with a sample run are included. The matrix A is transposed by the two methods and is checked for correctness in the program.

   The timings for transposing the $64 * 64$ matrix are

Method for transpose	**Time in sec**
XNET	0.017643
ROUTER	0.002646

2. **Matrix Multiplication** The matrix multiplication problem was solved in O(n) algorithm (it took 3n -2 steps). The idea for the implementation was taken from Systolic array and mesh matrix multiplication algorithm. The aim was to keep 2 plural data items on each node (a and b matrices) and calculate c (another plural) by shifting the elements (each element is used n times exactly) and circulating these in the mesh. The code for this can be found in the function matrix_multiply() in the file multiply.m

   Also, a matrix multiplication algorithm from the MasPar manual was implemented to verify the correctness of the implementation described above. The two talied but the MasPar implementation was more efficient than the above implementation which forms the submission for this assignment.

   While programming on MasPar I realised that to get the algorithm to work efficiently, all processors must be running simulataneosly (and active most of the time). The condition loops should be avoided and the processor indices should be used for generating uniqueness among the processors.

   The reference for the algorithm implemented in matrix_multiply is "The design and Analysis of Parallel Algorithms" - Selim G Akl. Prentice Hall 1989. page 180 section

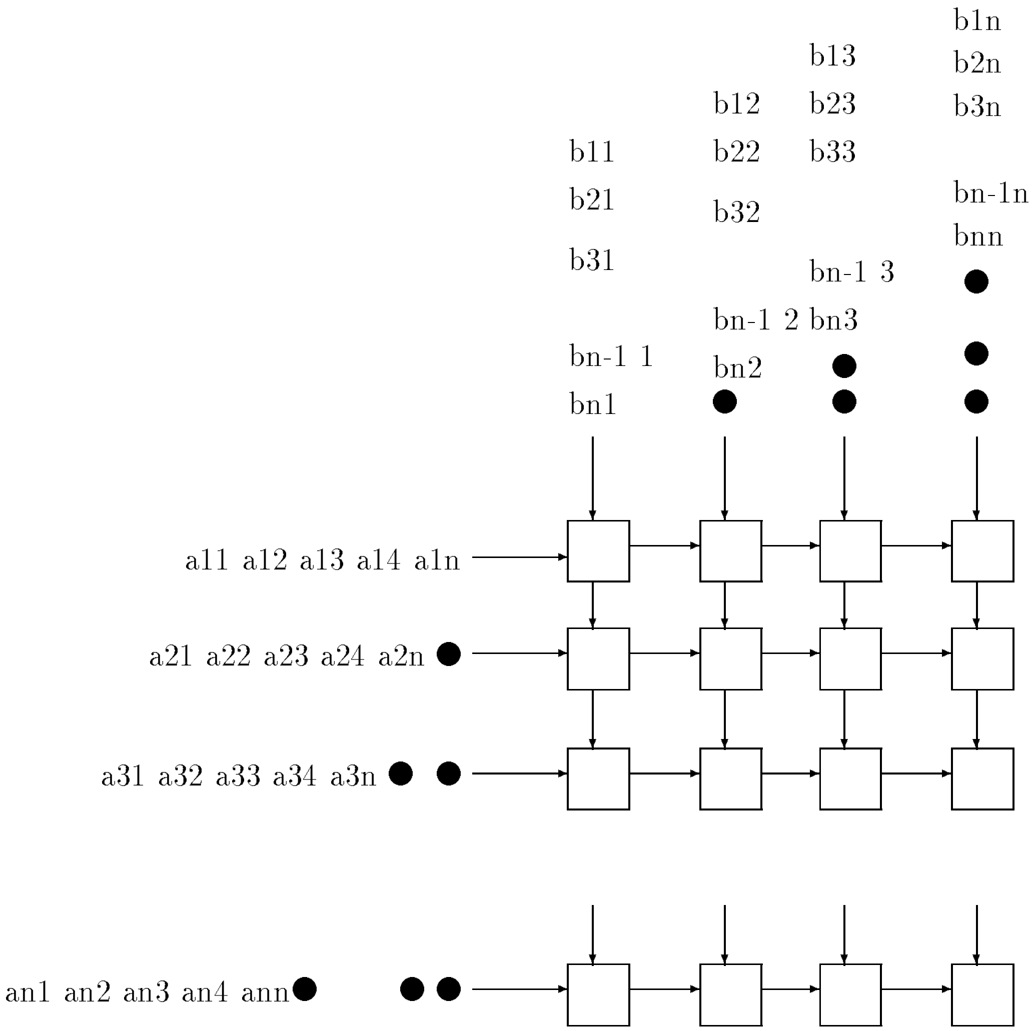

Figure 1: The movement of matrices a and b in Matrix Multiplication

7.3.1 "Mesh multiplication" This section describes how to implement the matrix multiplication on a mesh by staggering A and B. This implementation takes $3n - 2$ steps as it waits for the first n elements to arrive, then P(n,n) has to wait $n - 1$ steps for the first element to arrive and then has to wait for $n - 1$ steps for the last element to finish. This is not the most optimum style and can be improved further. This way using n^2 processors we are able to realise the sequential matrix multiplication algorithm which is $O(n^3)$ algorithm in $O(n)$ complexity.

The timings for matrix multiplication are

Matrix Size	**Time in sec**
$64 * 64$	0.036046