Research Journal of Applied Sciences, Engineering and Technology 7(21): 4415-4422, 2014
DOI:10.19026/rjaset.7.818
ISSN: 2040-7459; e-ISSN: 2040-7467

Submitted: August 16, 2013 Accepted: September 02, 2013 Published: June 05, 2014

Research Article

Comparison among Performance Measures for Parallel Matrix Multiplication Algorithms

Halil Snopce and Azir Aliu
SEEU, CST Faculty, Ilindenska 335, Tetovo, Macedonia

Abstract: In this study we analyze how to make a proper selection for the given matrix-matrix multiplication operation and to decide which is the best suitable algorithm that generates a high throughput with a minimum time, a comparison analysis and a performance evaluation for some algorithms is carried out using the identical performance parameters.

Keywords: Algorithms for parallel matrix multiplication, linear transformation and nonlinear transformation, performance parameter measures, Processor Elements (PEs), systolic array

INTRODUCTION

Most of the parallel algorithms for matrix multiplication use matrix decomposition that is based on the number of processors available. This includes the systolic algorithm (Choi *et al*., 1992), Cannon's algorithm (Alpatov *et al*., 1997), Fox's and Otto's Algorithm (Agarwal *et al*., 1995), PUMMA (Parallel Universal Matrix Multiplication) (Choi *et al*., 1994), SUMMA (Scalable Universal Matrix Multiplication) (Cannon, 1969) and DIMMA (Distribution Independent Matrix Multiplication) (Chtchelkanova *et al*., 1995). The standard method for $n\times n$ matrix multiplication uses $O(n^3)$ operations (multiplications). Most of the parallel algorithms are parallelization of this method. For instance, matrix multiplication can be performed in $O(n)$ time on a mesh with wraparound connections and $n\times n$ processors (Cannon, 1969); in $O(\log n)$ time on a three dimensional mesh of trees with n^3 processors (Leighton, 1992); in $O(\log n)$ time on a hypercube or shuffle exchange network with n^3 processors (Dekel *et al*., 1983). On the other hand, all implementations have a cost, i.e., time processor product of at least $O(n^3)$. Therefore, the aim is to develop highly parallel algorithms that have the cost less than $O(n^3)$.

Extensive work has been done for constructing algorithms that require $O(n^\beta)$ operations, such that $\beta<3$. The research world was stunned with $O(n^{2.81})$ algorithm for matrix multiplication (Strassen, 1969). A recursive application gives an algorithm that runs in time (Strassen, 1969). Following Strassen's work, a series of studies published among the years 1970^s and 80s Achieved better upper bounds on the exponent. Currently, the best matrix multiplication algorithm takes $O(n^\beta)$ operations, where $\beta<2.375477$ (Coppersmith and Winograd, 1990). However, these algorithms are highly sophisticated even for sequential execution; parallelization of these methods are only of theoretical interest and far from practical (Leighton, 1992; Robinson, 2005; Coppersmith and Winograd, 1990). Therefore, from practical point of view, parallelization of standard matrix multiplication algorithm with $O(n^3)$ multiplications is more appropriate. The objective of this study is to investigate which parallel method is more appropriate and how to optimize the existing methods in order to obtain more efficient parallelization.

METHODOLOGY

Parallel computing paradigms: Parallel computing process depends on how the processors are connected to the memory. There are two different ways how this can be done: shared memory system and distributed memory.

In a shared memory system, a single address space exists; within it, to every memory location is given a unique address and the data stored in the memory are accessible to each processor. In such a system the processor P_i reads the data written by the processor P and it is necessary to use synchronization.

In a distributed memory system, each processor can only access to its own memory (local memory). The connection between processors is done by the so called "high-speed communication network" and they exchange information using send and receive operations.

MPI (Message Passing Interface) is useful for a distributed memory systems since it provides a widely used standard of message passing program. In MPI, data is distributed among processors, where no data is shared and data is communicated by message passing.

Corresponding Author: Halil Snopce, SEEU, CST Faculty, Ilindenska 335, Tetovo, Macedonia

Matrix multiplication algorithms as a nested loop algorithms are suitable for such kind of systems.

Systolic array for matrix multiplication: A systolic array is a computing network possessing with the features of synchrony (data are rhythmically computed and passed through the network), modularity and locality, special locality (a locally communicative interconnection structure), pipeline ability, repeatability (usually the repetition of one single type of cell and interconnection occurs). By the systolic arrays it can be achieve a high parallelism. Systolic arrays are very suitable for nested-loop algorithms; such is the problem of parallel solution of matrix-matrix multiplication (Bekakos, 2001; Snopce and Elmazi, 2008a; Gusev and Evans, 1994).

The simplest systolic arrays are the chain and the ring, which in fact are one dimensional array. In the $m \times n$ two dimensional systolic array the nodes are defined by integer points in a rectangle $\{(i, j), 0 \le i \le m, 0 \le j \le n\}$. A node (i, j) is connected with nodes $(i \pm 1, j)$ and $(i, j \pm 1)$, if they are in the rectangle. If we add diagonal edges $((i, j), (i \pm 1, j \mp 1))$ to the two dimensional array, then we get the so called hexagonal systolic array. By adding the edges $((0, j), (m\text{-}1, j))$ and $((i, 0), (i, n\text{-}1))$ to an $m \times n$ array, we get a two dimensional torus. There are given some models of systolic arrays in Fig. 1.

The simplest case of systolic array which can be used for parallel matrix multiplication is similar as the array shown in the Fig. 1c. The systolic array is quadratic 2-D such that the elements of a matrix A are fed into the array by the rows and the elements of the matrix B are fed into the array by the columns. The array consists of $n \times n$ PEs. Each PE indexed by i and j, in each step k, adds a partial product $a_{ik}\ b_{kj}$ to the accumulated sum for every c_{ij}. Hence, we have a movement in two directions. The elements of the resulting matrix C are stationary. In Fig. 2 are shown just the first two cycles of such quadratic systolic array. The matrices A and B are taken to be quadratic of order 3.

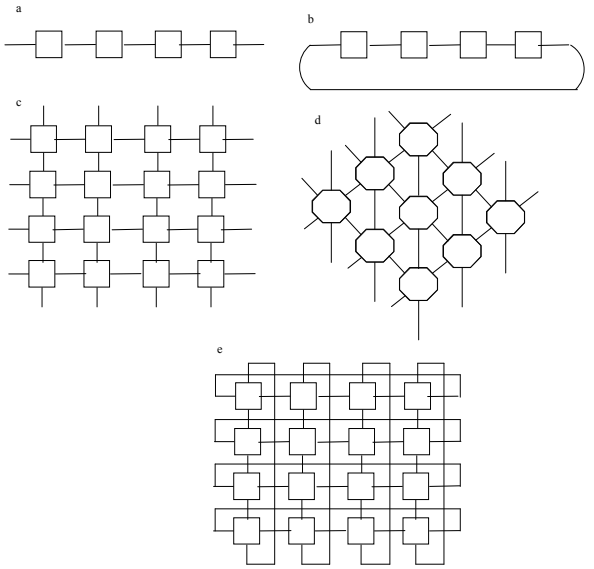

Fig. 1: Typical models of systolic arrays, (a) linear systolic array, (b) 1-D ring, (c) 2-D square array, (d) 2-D hexagonal array, (e) 2-D torus

Why systolic array? The MPI technique needs two kinds of time to complete the multiplication process, t_c and t_f, where t_c represents the time it takes to communicate one data between processors and t_f is the time needed to multiply or add elements of two matrices. It is assumed that matrices are of type $n \times n$. The other assumption is that the number of processors is p. Each processor holds n^2/p elements and it was assumed that n^2/p is set to a new variable m^2. The number of arithmetic operations units will be

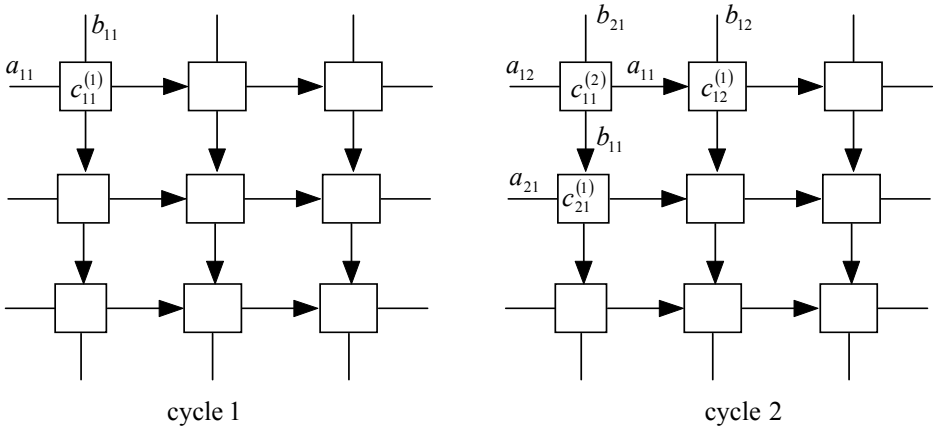

Fig. 2: The first two cycles (7 in total) for the 2-D systolic computation of 3×3 matrix product

Table 1: Theoretical results

Algorithm	*f*	c	q	S	E
Systolic algorithm	55080000	1440000	38.250	3.8250	0.9560
Cannon's algorithm	271800000	108720000	2.500	0.2500	0.0625
Fox's algorithm with square decomposition	162360000	270720000	0.599	0.0599	0.0150
Fox's algorithm with scattered decomposition	54360000	109440000	0.497	0.0497	0.0124
PUMMA	54360000	1620000	33.550	3.3550	0.8390
SUMMA	54360000	1800000	30.200	3.0200	0.7550
DIMMA	54360000	1800000	30.200	3.0200	0.7550

Table 2: Experimental results

p	Systolic	Cannon's	Fox 1	Fox 2	PUMMA	SUMMA	DUMMA
1	251.690	530.33	390.20	506.490	271.120	149.540	165.970
4	65.500	408.12	236.32	267.244	79.583	45.241	49.390

Table 3: Values of S and E

Algorithm	S	E
Systolic algorithm	3.832	0.96
Cannon's algorithm	1.299	0.32
Fox's algorithm with square decomposition	1.651	0.41
Fox's algorithm with scattered decomposition	1.895	0.47
PUMMA	3.407	0.85
SUMMA	3.305	0.83
DIMMA	3.360	0.84

denoted by *f*. The number of communication units will be denoted by *c*. The quotient $q = f/c$ will represent the average number of flops per communication access. The speedup is $S = q. (t_f /t_c)$ (the definitions of some special performance characteristics are given in the next subchapter). In order to apply the analytical variables, we need to make some assumptions. Firstly we assume that the number of processors is $p = 4$. The dimension of the matrix is $n = 600$. The last assumption is that $t_f/t_c = 0.1$. Also we need to use the Efficiency formula $E = S/p$. In the Table 1 we record the results that we obtain for all algorithms, under the assumptions that we made (Alpatov *et al*., 1997; Agarwal *et al*., 1995; Choi *et al*., 1994; Alqadi *et al*., 2008).

From Table 1, after analyzing the theoretical results, one can conclude that the systolic algorithm will give the best performance (i.e., efficiency). The efficiency increases as the parameter S approaches the number of processors.

The experimental results (Alqadi and Amjad, 2005) for the execution time using 1 processor and using 4 processors are shown in the Table 2. The table is obtained by implementing each algorithm 100 times and then, ten fastest five ones are taken and averaged. On the other hand, taking into the consideration that $S = T(1)/T(p)$ (which means that speedup is equal to the time using 1 processor dividing with the time using *p* processors), we give the Table 3 with the values of *S* and *E*.

The analysis explained above has shown that the systolic algorithm may be considered as the best algorithm that produces a high efficiency. The theoretical analysis matches with the experimental results performed in Alqadi *et al*. (2008). This analysis is useful for making a conclusion and giving a recommendation for selecting the best algorithm among others which is more effective for parallel solving of some linear algebra problems, including the problem of matrix-matrix multiplication.

MATHEMATICAL TOOLS

Definition 1: The array size (Ω) is the number of PEs in the array.

Definition 2: The computation Time (*T*) is the sum of the time for the date input in the array-T_{in}, the time for the algorithm executing-T_{exe} and the time necessary for dates leaving the array-T_{out}, i.e.:

$$T = T_{in} + T_{exe} + T_{out} \tag{1}$$

Definition 3: The execution time, T_{exe}, is defined as:

$$T_{exe} = 1 + \max_{(t,x,y)\in P_{ind}} t - \min_{(t,x,y)\in P_{ind}} t_{out} \tag{2}$$

Theorem 1: Zhang *et al*. (1994): The execution time is given by the relation:

$$T_{exe} = 1 + \sum_{j=1}^{3} (N_j - 1) \cdot \left| \min_t t_{ij} \right| \tag{3}$$

Definition 4:
The Pipelining period (α): The time interval between two successive computations in a PE. If λ is a scheduling vector and *u* is a projection direction, then the pipelining period is given by the relation:

$$\alpha = \lambda^T u \tag{4}$$

In some places (for the three nested loops algorithms), for the pipeline period it is used the formula:

$$\alpha = \left| \frac{\det(T)}{\gcd(T_{11}, T_{12}, T_{13})} \right| \tag{5}$$

where, T is a transformation matrix and T_{1i}, $i = 1, 2, 3$ is (1, i) co-factor (minor) of matrix T and gcd (T_{11}, T_{12}, T_{13}) is the greatest common divisor of T_{11}, T_{12} and T_{13}.

Definition 5: The geometric area (g_a) of a two-dimensional systolic array is the area of the smallest convex polygon which bounds the PEs in the (x, y) - plane. The geometric area is given by the formula:

$$g_a = (N_1 - 1)(N_2 - 1)|T_{13}| + (N_1 - 1)(N_3 - 1)|T_{12}| + (N_2 - 1)(N_3 - 1)|T_{11}| \quad (6)$$

Definition 6: The Speedup (S) of a systolic array is the ratio of the processing time in the SA to the processing time in a single processor (T_1), i.e.:

$$S = \frac{T_1}{T} \quad (7)$$

Definition 7: The Efficiency (E) is defined as the ratio of the speedup to the number of PEs in the array i.e.:

$$E = \frac{S}{\Omega} = \frac{T_1}{Tx\Omega} \quad (8)$$

Theorem 2: Bekakos (2001) and Zhang *et al*. (1994): The number of processors on SHSA array is (the array where we have no using the linear transformation):

$$\Omega = 3N^2 - 3N + 1 \quad (9)$$

Theorem 3 (Snopce and Elmazi, 2008b): The number of processing elements in 2-dimensional systolic array for the algorithm of matrix-matrix multiplication for which is used the projection direction $u = [1\ 1\ 1]^T$, could be reduced and given with $\Omega = N^2$.

Theorem 4 (Gusev and Evans, 1994): The number of PEs for the systolic array which is constructed by using the nonlinear transformation the number of PEs is given by the relation:

$$\Omega = N \cdot \left\lfloor \frac{3N-1}{2} \right\rfloor \quad (10)$$

Let $P_{ind} = \{(i, j, k)\ /1 \leq i, j, k \leq N\}$ be the index space of the used and computed data for matrix multiplication. The purpose is to implement the mapping $(i, j, k) \xrightarrow{T} (t, x, y)$, using an appropriate form for the matrix T. There are different options of such matrix T (Bekakos, 2001).

Definition 8: The linear transformation matrix T is the matrix:

$$T = \begin{bmatrix} T_1 \\ S \end{bmatrix} = \begin{bmatrix} t_{11} & t_{12} & t_{13} \\ t_{21} & t_{22} & t_{23} \\ t_{31} & t_{32} & t_{33} \end{bmatrix}$$

where, $T_1 = [t_{11}, t_{12}, t_{13}]$ is the scheduling vector (time schedule of the computations; in the case of matrix multiplication this is always [1 1 1]) and:

$$S = \begin{bmatrix} T_2 \\ T_3 \end{bmatrix} = \begin{bmatrix} t_{21} & t_{22} & t_{23} \\ t_{31} & t_{32} & t_{33} \end{bmatrix}$$

is a transformation which maps P_{ind} into a 2-dimensional systolic array.

Definition 9: A matrix T is associated to the projection direction $u = [u_1\ u_2\ u_3]^T$ (there are some possible allowable projection vectors, Bekakos (2001), so that the following conditions must be satisfied:

- $\det T \neq 0$
- $T_2 u = 0$; $T_3 u = 0$
- The entries have to be from the set {-1, 0, 1}

Definition 10: The transformation matrix T maps the index point $(i, j, k) \in P_{ind}$ into the point $(t, x, y) \in T$. P_{ind}, where, P_{ind} is the set of index points such that:

$$t = T_1 [i\ \ j\ \ k]^T = [1\ \ 1\ \ 1] \begin{bmatrix} i \\ j \\ k \end{bmatrix} = i + j + k$$
$$[x\ \ y]^T = S[i\ \ j\ \ k]^T = \begin{bmatrix} t_{21} & t_{22} & t_{23} \\ t_{31} & t_{32} & t_{33} \end{bmatrix} \begin{bmatrix} i \\ j \\ k \end{bmatrix} \quad (11)$$

In this case t is the time where calculations are performed and (x, y) are the coordinates of processor's elements on the 2-dimensional systolic array.

Definition 11: For the algorithm of matrix-matrix multiplication the transformation which is used for obtaining the new index space is defined with $H = T \circ L_1$ where,

$$L_1 = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

If one uses definition 9 and 11, the linear transformation matrix is transformed to a new matrix of the form:

$$H = \begin{bmatrix} t_{11} + t_{12} + t_{13} & t_{12} & t_{13} \\ 0 & t_{22} & t_{23} \\ 0 & t_{32} & t_{33} \end{bmatrix}$$

This happens because from the case 2) of definition 9, it is easy to conclude that $t_{21} + t_{22} + t_{23} = 0$ and $t_{31} + t_{32} + t_{33} = 0$.

RESULTS AND DISCUSSION

The advantage of using the linear transformation in designing the systolic array for matrix multiplication: If one uses theorem 2, then it is possible to find the number of PEs in the corresponding systolic array. For example, if N = 4 then Ω = 37 (Fig. 3). In the case of the systolic array constructed using the properties of theorem 3, the number of processors (Fig. 4) is $\Omega = 4^2 = 16$. In Fig. 3 and 4 these two arrays are represented. The second array has less number of PEs and it is one of the advantages of using the linear transformation in constructing the systolic arrays. So, we can conclude how the number of processors on the array can be reduced using the linear transformation. For N = 4 is 16 vis-à-vis 37 without using the transformation. On the Table 4 we give the comparison for number of PE for different values of N. This information is taken from Bekakos (2001).

For the pipeline period, in the case of the array in Fig. 3, using relation (4) we get:

$$\alpha = \lambda^T u = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = 3 \tag{12}$$

Table 4: Comparison for number of PEs

N	Without using L	By using L
5	61	25
10	271	100
50	7351	2500
100	29701	10000

This means that every computation in each processor is performed after the time interval of three units (after every third step).

If one uses the relation (5), then the same result will be obtained. To confirm this, we give the transformation matrix used in this case:

$$T = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & -1 \\ 0 & 1 & -1 \end{bmatrix} \tag{13}$$

For this matrix we have the following results: det $(T) = 3$, $T_{11} = 1$, $T_{12} = -1$, $T_{13} = 1$. Thus, using the relation (5), the same result as in (12) will be obtained.

If this formula is used for the systolic array which is constructed by using the linear transformation matrix (the array in Fig. 4), we get:

$$\alpha = \left| \frac{\det(H)}{\gcd(H_{11}, H_{12}, H_{13})} \right| = \left| \frac{H_{11}}{H_{11}} \right| = 1$$

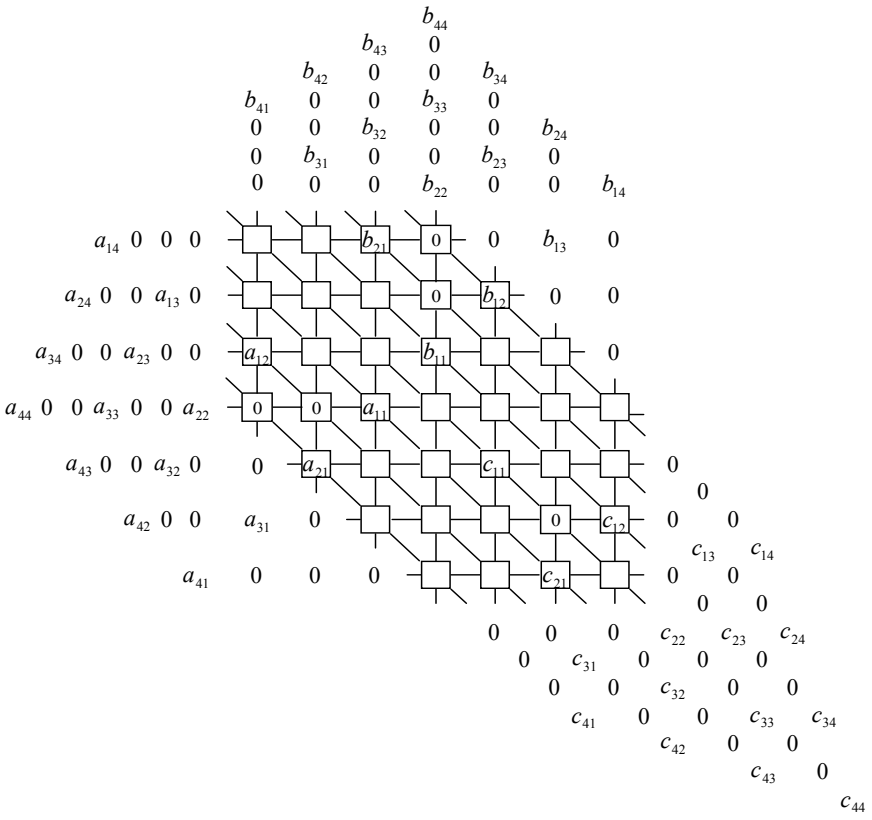

Fig. 3: The SHSA array for N = 4 (without using the linear mapping)

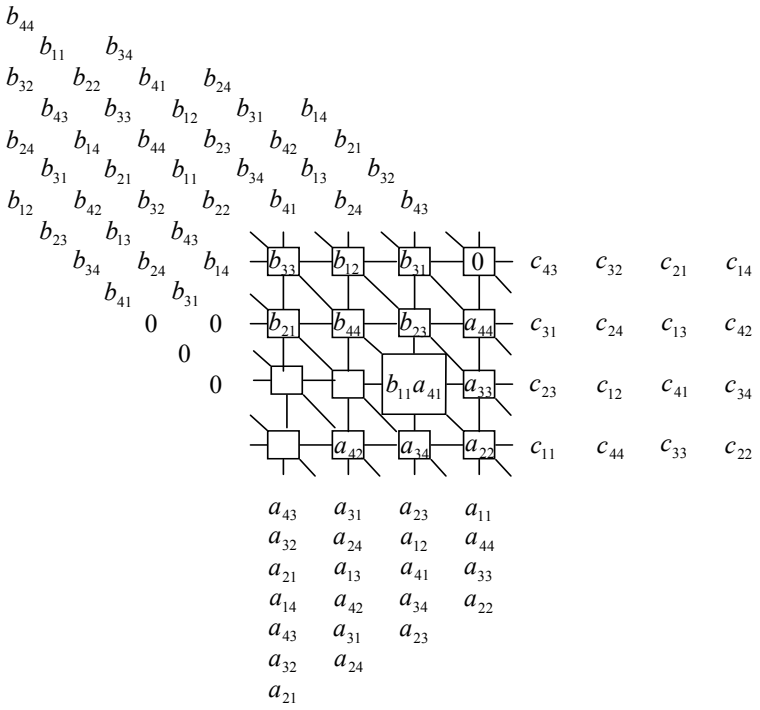

Fig. 4: Systolic array for N = 4 using the mapping L

This is due to the fact that for the matrix H the cofactor $H_{12} = 0$ and $H_{13} = 0$.

This means (the result $\alpha = 1$) that in the case of Fig. 4 the PEs perform in every step.

If k is an index point, then of course, max $k = (N_1, N_2, N_3)$ and min $k = (1, 1, 1)$. Using the relation (11) we have that:

$$\begin{aligned} &\max_{(t,x,y)\in P_{ind}} t = N_1 + N_2 + N_3 \\ &\min_{(t,x,y)\in P_{ind}} t = 1 + 1 + 1 = 3 \end{aligned} \tag{14}$$

From the relations (2) and (14) the execution time may be obtained:

$$\begin{aligned} &T_{exe} = 1 + N_1 + N_2 + N_3 - 3 = \\ &= N_1 + N_2 + N_3 - 2 \end{aligned} \tag{15}$$

If one uses the fact that if $N_1 = N_2 = N_3$, then $T_{exe} = 3N$-2. On the other hand $T_{in} = T_{out} =$ N-1, therefore the computation time using (1) will be:

$$T = 5N - 4 \tag{16}$$

In the case of the array in Fig. 4 the execution time will be ordered by using the theorem 1 which is associated by the relation (3):

$$\begin{aligned} &T_{exe} = 1 + (N_1 - 1)\cdot 0 + (N_2 - 1)\cdot 1 + (N_3 - 1)\cdot 1 \\ &= N_2 + N_3 - 1 \end{aligned} \tag{17}$$

If $N_2 = N_3$ then $T_{exe} = 2N$-1. In this case $T_{in} = N$-1 and $T_{out} = 0$, therefore the computational time is:

$$T = 3N - 2 \tag{18}$$

In the Fig. 5 the systolic array using nonlinear transformation is presented (Gusev and Evans, 1994). For this array (Fig. 5), we have that $T_{exe} = 3N$-1, $T_{in} = N$-1 and $T_{out} = 0$. Therefore the computation time is:

$$T = 4N - 2 \tag{19}$$

For the geometric area in the case of the array in Fig. 3 we have:

$$\begin{aligned} &g_a = (N_1 - 1)(N_2 - 1)|T_{13}| + (N_1 - 1)(N_3 - 1)|T_{12}| + (N_2 - 1)(N_3 - 1) \\ &|T_{11}| = (N_1 - 1)(N_2 - 1)\cdot 1 + (N_1 - 1)(N_3 - 1)\cdot 1 + (N_2 - 1) \\ &(N_3 - 1)\cdot 1 = N_1N_2 + N_1N_3 + N_2N_3 - 2(N_1 + N_2 + N_3) + 3 \end{aligned} \tag{20}$$

If one takes $N_1 = N_2 = N_3 = N$ then:

$$g_a = (N - 1)^2 \tag{21}$$

In the case of the array with nonlinear transformation, one can calculate the geometric area in a similar way as above:

$$g_a = (N_2 - 1)(N_3 - 1) + (N_2 - 1)(N_3 - 1)$$

Similarly for $N_1 = N_2 = N_3 = N$ we have:

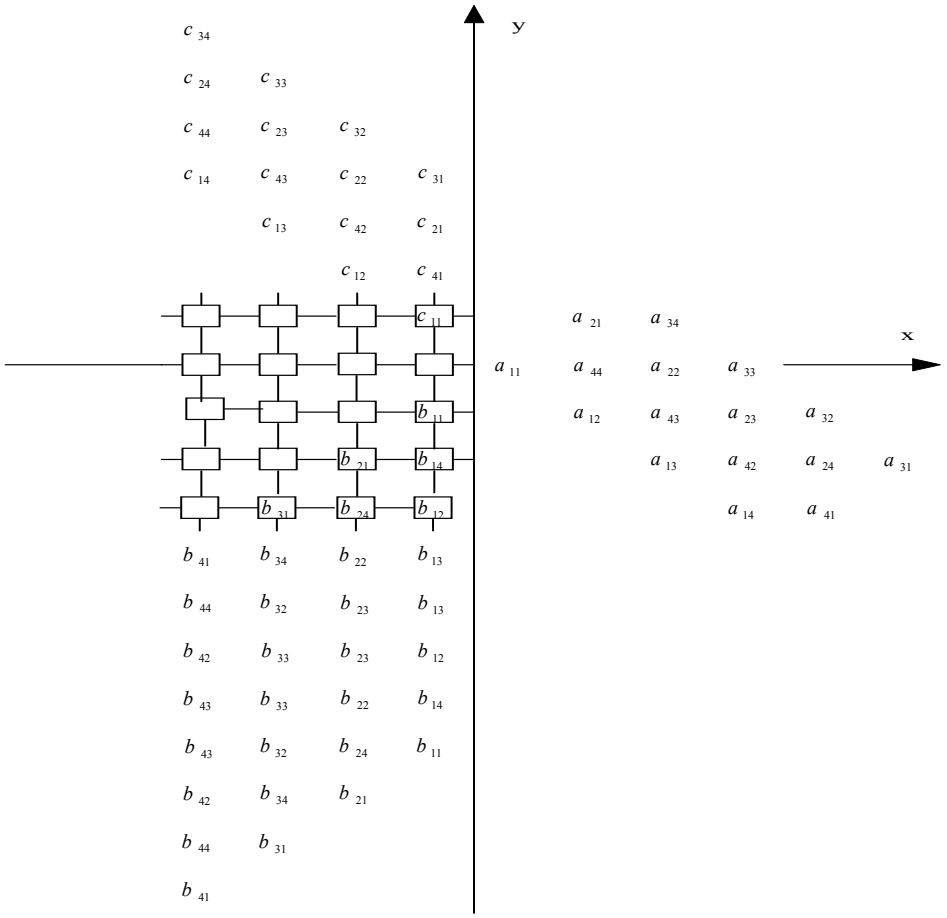

Fig. 5: Systolic array for N = 4 (with nonlinear mapping)

Table 5: Comparison of performance characteristics

	Without using L			By using L			By nonlinear transf.		
	N = 4	N = 10	N = 100	N = 4	N = 10	N = 100	N = 4	N = 10	N = 100
Ω	37	271	29701	16	100	10000	20	140	14900
T	16	46.0	496	10	28	298	14	38	398
S	4	21.7	2016	6.4	35.7	3355	4.5	26.3	2512.6
E	10.8%	8%	6.8%	40%	35.7%	33.5%	23%	19%	16.9%
g_a	27	243	29403	9	81	9801	18	162	19602

$$g_a = 2(N-1)^2 \tag{22}$$

Since the duration of matrix multiplication on a system with only one processor is $T_1 = N^3$, the speedup and efficiency in the case of the array in Fig. 3, using relations (7) and (8), will be respectively:

$$S = \frac{N^3}{5N-4} \tag{23}$$

$$E = \frac{N^3}{(5N-4)(3N^2-3N+1)}\left(\lim_{N\to\infty} E = \frac{1}{15} = 6.7\%\right) \tag{24}$$

The same parameters in the case of using linear transformation matrix are:

$$S = \frac{N^3}{3N-2} \tag{25}$$

$$E = \frac{N}{(3N-2)}\left(\lim_{N\to\infty} E = \frac{1}{3} = 33.3\%\right) \tag{26}$$

And finally these parameters for the array where nonlinear transformation has been used are:

$$S = \frac{N^3}{4N-2} \tag{27}$$

$$E = \frac{N^2}{(4N-2)\cdot\left\lfloor\frac{3N-1}{2}\right\rfloor}\left(\lim_{N\to\infty} E = \frac{1}{6} = 16.7\%\right) \tag{28}$$

Using the results obtained by the relations (16-28), as well as theorems 1, 2 and 3, one can construct the corresponding table, where all the results can be compared. In Table 5 it is given a comparison of performance characteristics for some values of *N*.

CONCLUSION

In this study we make the comparison concerning some performance measures for parallel matrix multiplication. We emphasized the systolic approach as most efficient. We can conclude that using the identical performance parameters, for each parameter, the array which is constructed using linear transformation matrix has better performances. Especially for the efficiency when *N* tends to the infinity we have that it is approximately five times better than the array without using the linear transformation. From the Table 5 we can deduce the advantage of using the linear transformation. The results from the Table 5 are done for N = 4, N = 10 and N = 100. The table can be enlarged for other values of N as well. In all the cases, the results by using L are better.

REFERENCES

Agarwal, R.C., S.M. Balle, F.G. Gustavson, M. Joshi and P. Palkar, 1995. A 3-Dimensional approach to parallel matrix multiplication. IBM J. Res. Dev., 39(5): 1-8.

Alpatov, P., G. Baker, C. Edwards, J. Gunnels, G. Morrow, J. Overfelt, Robert Van de GEijn and J. Wu, 1997. Plapack: Parallel linear algebra package. Proceeding of the SIAM Parallel Processing Conference.

Alqadi, Z. and J. Amjad, 2005. Analysis of program methods used for optimizing matrix multiplication. J. Eng., 15(1): 73-78.

Alqadi, Z., J. Amjad and M. Ibrahiem, 2008. Performance analysis and evaluation of parallel matrix multiplication algorithms. World Appl. Sci. J., 5(2): 211-214.

Bekakos, M.P., 2001. Highly Parallel Computations-algorithms and Applications. Democritus University of Thrace, Greece, pp: 139-209.

Cannon, L.E., 1969. A cellular computer to implement the kalman filter algorithm. Ph.D. Thesis, Montana State University.

Choi, J., J.J. Dongarra and D.W. Walker, 1992. Level 3 BLAS for distributed memory concurrent computers. Proceeding of Environment and Tools for Parallel Scientific Computing Workshop.

Choi, J., J.J. Dongarra and D.W. Walker, 1994. PUMMA: Parallel Universal Matrix Multiplication Algorithms on distributed memory concurrent computers. Concurrency-Pract. Ex., 6(7): 543-570.

Chtchelkanova, A., J. Gunnels, G. Morrow, J. Overfelt and R. van de Geijn, 1995. Parallel implementation of BLAS: General techniques for level 3 BLAS, TR-95-40. Department of Computer Sciences, University of Texas, OCT.

Coppersmith, D. and S. Winograd, 1990. Matrix multiplication via arithmetic progressions. J. Symb. Comput., 9: 251-280.

Dekel, E., D. Nassimi and S. Sahni, 1983. Parallel matrix and graph algorithms. SIAM J. Comput., 10: 657-673.

Gusev, M. and D.J. Evans, 1994. A new matrix vector product systolic array. J. Parallel Distr. Com., 22: 346-349.

Leighton, T., 1992. Introduction to Parallel Algorithms and Architectures: Arrays-trees-Hypercubes. Morgan Kaufmann, San Mateo, California.

Robinson, S., 2005. Toward an optimal algorithm for matrix multiplication. SIAM News, 38(9).

Snopce, H. and L. Elmazi, 2008a. Reducing the number of processors elements in systolic arrays for matrix multiplication using linear transformation matrix. Int. J. Comput. Commun. Control, 3: 486-490.

Snopce, H. and L. Elmazi, 2008b. The importance of using linear transformation matrix in determining the number of PEs in systolic arrays. Proceeding of ITI 2008. Cavtat, Croatia, pp: 885-892.

Strassen, V., 1969. Gaussian elimination is not optimal. Numer. Math., 13: 354-356.

Zhang, C.N., J.H. Weston and F.Y., Yan, 1994. Determining object functions in systolic array designs. IEEE T. VLSI Syst., 2(3): 357-360.