2013 International Nuclear Atlantic Conference - INAC 2013
Recife, PE, Brazil, November 24-29, 2013
ASSOCIAÇÃO BRASILEIRA DE ENERGIA NUCLEAR - ABEN
ISBN: **978-85-99141-05-2**

CLUSTER IMPLEMENTATION FOR PARALLEL COMPUTATION WITHIN MATLAB SOFTWARE ENVIRONMENT

Antonio O. de Santana[1], Carlos C. Dantas[2], Luiz G. da R. Charamba[3], Wilson F.de Souza Neto[4], Silvio B. Melo[5], Emerson A. de O. Lima[6]

[1,2]Departamento de Energia Nuclear – CTG - Universidade Federal de Pernambuco
Av. Prof. Luiz Freire, 1000 – CDU
50740-540 – Recife - PE
mailto:aos@ufpe.br
ccd@ufpe.br

[3,4,5]Centro de Informática – CJN - Universidade Federal de Pernambuco
Av. jornalista Anibal Fernandes, S/N –Cidade Universitária
50740-560 – Recife - PE
sbm@cin.ufpe.br

[6]Departamento de Matemática – Universidade de Pernambuco
Rua Benfica, 455 - Madalena
50720-001 – Recife – PE
emathematics@gmail.com

ABSTRACT

A cluster for parallel computation with MATLAB software the COCGT – Cluster for Optimizing Computing in Gamma ray Transmission methods, is implemented. The implementation correspond to creation of a local net of computers, facilities and configurations of software, as well as the accomplishment of cluster tests for determine and optimizing of performance in the data processing.. The COCGT implementation was required by data computation from gamma transmission measurements applied to fluid dynamic and tomography reconstruction in a FCC-Fluid Catalytic Cracking cold pilot unity, and simulation data as well. As an initial test the determination of SVD – Singular Values Decomposition - of random matrix with dimension (n , n), n=1000, using the Girco's law modified, revealed that COCGT was faster in comparison to the literature [1] cluster, which is similar and operates at the same conditions. Solution of a system of linear equations provided a new test for the COCGT performance by processing a square matrix with n=10000, computing time was 27 s and for square matrix with n=12000, computation time was 45 s. For determination of the cluster behavior in relation to "parfor" (parallel for-loop) and "spmd" (single program multiple data), two codes were used containing those two commands and the same problem: determination of SVD of a square matrix with n= 1000. The execution of codes by means of COCGT proved: 1) for the code with "parfor", the performance improved with the labs number from 1 to 8 labs; 2) for the code "spmd", just 1 lab (core) was enough to process and give results in less than 1 s. In similar situation, with the difference that now the SVD will be determined from square matrix with n= 1500, for code with "parfor", and n=7000, for code with "spmd". That results take to conclusions: 1) for the code with "parfor", the behavior was the same already described above; 2) for code with "spmd", the same besides having produced a larger performance, it supports a larger work load

1. INTRODUCTION

Large and complex software generating great amount of data are typical of activities, such as: artificial intelligence, molecular physics, meteorological forecast, researches of DNA, exploration of petroleum, etc. Such activities demand the use of machines of multiple processors and of super-computers supplied by great companies. These solutions offer a great performance, but they have a high cost and low scalability. Before the presented difficulties, Donald Becker and Thomas Sterling, working in the NASA (1994), developed a prototype of a system composed of several interlinked PC as a local net. That experience, "cluster" call, produced excellent results, being obtained high processing levels, equivalent to the super-computers of the time, besides a bass cost in relation to those super-computers. A cluster can be defined as being a group of machines (nodes), usually without the need of peripheral, interlinked through a net; this machines work together, changing information amongst them to solve a certain task. Due to the high performance and low cost, the cluster of personal computers is quite used by the scientific communities, which have typical tasks that demand high processing power. In this context, the function of the cluster can be summarized as: given a complex problem and identified as being parallel, a server (master) should be responsible for the division of this problem in several pieces to be processed parallel in “nodes” slave (machines dedicated to the process). The solution found by each "node" is sent for the server such that the complete solution of the problem can be remounted.

On the other hand, a limit of clock frequency around of 4 GHz for the current processors was appraised for the next years. This limit is due to the super-heating of the processors, having the need of a special and appropriate cooling. However, until the moment, didn't appear a project totally viable and that can be marketed. While this, now a strong tendency exists among the manufacturers of processors of increasing the number of cores (nuclei) for processor. However, the addition of cores doesn't mean increase in computing power. However, the improvements in performance, offered by the new hardware multi-cores, allow the obtaining of more potent clusters, generating a quite favorable environment for the parallel computation [1].

The objective of the present work is the implementation of a cluster for parallel computation in MATLAB environment. That implementation was required by data computation from gamma transmission measurements applied to fluid dynamic and tomography reconstruction in a FCC - Fluid Catalytic Cracking - cold pilot unity, and simulation data as well [2]. This implicates in the accomplishment of facilities and configurations (software and hardware), as well as the accomplishment of tests to determine and to improve the performance of the cluster in data processing.

2. METODOLOGY

2.1. Brief Description of Parallel Computation in MATLAB Environment

In general, a local net of computers PC (nodes calls) in a defined lay-out initially should be created. Then, software and services are distributed, installed and configured in agreement with the architecture of the system (hardware and software). The diagram of blocks contained in the Figure 2.1 gives a general vision of the parallel computation in MATLAB environment. The tasks and jobs are created by the client that sends them to the server, which through his scheduler (or "Job Manager"), manages the execution of the jobs in agreement with the procedure: 1) the tasks are sent for her execution in the workers; 2) the reception of the results of the workers is accomplished; 3) the results are sent to the client.

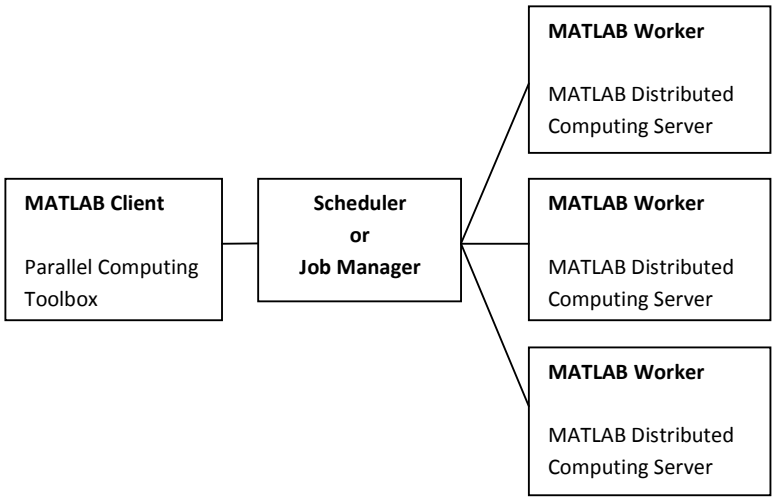

Figure 2.1 - Basic configuration of the parallel computation with MATLAB

2.2. Speedup and Efficiency

2.2.1. Speedup

"Speedup" corresponds to the metric for evaluation of performance of a system of parallel computation. With base in values of speedup, the number of machines to be used in the cluster is determined. Two modalities exist to define the "speedup" specifically: "absolute speedup" and "relative speedup". The "absolute speedup" is defined as the total time elapsed in the execution of the sequential algorithm with only 1 processor, divided by the total time elapsed in the execution of the parallel algorithm with "P" processors. This relationship is valid when both architectures (sequential and parallel) possess the same problem size. On the other hand, the relative speedup is defined as the relationship among the time of execution of

a parallel algorithm, with 1 (one) processor, and the time of execution due to the same algorithm, with "P" processors. The reason to use relative speedup is that the performance of parallel algorithms varies with the number of available processors in a cluster. Then, being compared the same algorithm with several numbers of processors, it is possible to verify with more precision the degradation of parallelism use; this doesn't happen being used the absolute speedup. The equation for the relative speedup is expressed as:

$$\boldsymbol{S = \frac{T(1)}{T(p)}} \quad \textbf{(1)}$$

where S is the speedup, T (1) is the time spend in the execution of the parallel algorithm with a single processor and T (P) is the time spend in the execution of the same algorithm with "P" processors. In systems parallel ideals, the "speedup" is equal to the number of processors ("P"). Already in practice, his value is smaller than "P".

2.2.2. Efficiency

The efficiency is another measure that can be used in the study of parallel architectures, being derived from the speedup definition. The efficiency tells the problem size and the processors number requested to maintain the efficient system. A direct relationship exists among processors number, problem size and the efficiency. Therefore, if the problem size goes constant, increasing the processors number, the efficiency will be decreased because of the overhead increase caused by the number of processors. Already if the problem size be increased, staying constant the processors number, the efficiency will be increased (in scalable parallel systems) due to the low overhead that is insignificant in relationship to computation of the problem. On the other hand, if both the problem size and processors number be increased, the efficiency will be constant. This way, it can stay a good efficiency (or Speedup), increasing the number of processors proportionally to the size of the problem. It is very difficult to find the exact limit of this proportionality for each architecture since the problem can be associated to countless aspects of hardware and software. The efficiency is determined in agreement with the following equation:

$$\textbf{E=S / P} \quad \textbf{(2)}$$

where "S" is the speedup and "P", the number of processors. Therefore, the number of processors should be chosen from way to maximize the efficiency and the speedup of the architecture. In systems parallel ideals, the efficiency is equal to 1 (one). In practice, the efficiency varies between 0 (zero) and 1 (one).

2.3. Flops and Benchmarks

2.3.1. Flops

FLOPS (or flops) is a computation acronym that means FLoating point OPerations per Second. FLOPS can be defined as being an unit of measure used to evaluate the processing capacity and, therefore, the performance of a computer. This acronym is used in several fields of the science and engineering, where large use of calculations with flotation point is made, similar to the old expression "instructions per second". Since the computation devices have enormous processing capacity, it is recommended to use larger units than FLOPS, this is, their multiples. The multiples more used are: megaflops (MFLOPS), gigaflops (GFLOPS), teraflops (TFLOPS), petaflops (PFLOPS) and exaflops (EFLOPS) [3].

2.3.2. Benchmarks

The FLOPS is used as measure unit of flotation point if a benchmark is present in all computer of interest. A benchmark is a program that accomplishes tests in a certain machine with the objective of to measure or to foresee her performance and to emphasize the strong and weak points of her architecture. Benchmarks can be classified in agreement with the application class for which they are prepared as, for instance, scientific computation, net services, applications multimedia, processing of signs, etc. Among the benchmarks, Linpack is one of the more used by the scientific community. Her current version - HPL (High Performance Linpack) - it contains two groups of routines: one for decomposition of matrix and other to solve the system of resulting linear equations of the decomposition. Her application is mainly in machines that use software for calculations scientific and of engineering, because the operations more used in these types of applications are in point-flotation [4].

2.4. Experimental Procedure

2.4.1. Computer type

Two computers were used for the formation of the cluster (server e worker) and more one computer designated to be the client. Both computers possess the same hardware configuration, which is described below as:

CPU: Intel Colors i7 860, 2.8 GHz, 8 MB, LGA 1156
Plate Mother: Intel DP55WG
Memory (4 GB): 2x (DDR3, 2GB, 1333 MHz, KINGSTON, KVR 1333, D3N9/2GB)
Plate of Video: 1 GB, GF9400GT, PCI EXP, NVIDIA
Hard Disk (HD): 1000 GB, SATAN, 32 MB, 5400 RPM
Plate of Net: 10/100/1000 MB / s

2.4.2. Installation and configuration of the cluster (COCGT)

The computers, with the architecture described above, were installed and configured in agreement with the manuals of installation and configuration (version 4.3) supplied by MathWorks. These manuals are distributed in 4 (four) volumes (or apprenticeships), whose titles are:

Stage 1/4: Installing MATLAB Distributed Computing Server. 4.3 on to Windows Operating System Cluster

Stage 2/4: Configuring MATLAB Distributed Computing Server. 4.3 go Uses with Job Manager on Windows Operating Systems

Stage 3/4: Installing Parallel Computing Toolbox. 4.3 on Windows Operating Systems

Stage 4/4: Testing Your Installation of MATLAB Distributed Computing Server-4.3 with Job Manager

Initially, a local net was created being formed by three nodes (computers), two of them being designated as server and "worker", while the other "node" was designated as client (Figure 2.2). This way, the Cluster for Optimizing Computing in Gamma ray Transmission methods -- COCGT was created. Soon after, the installations and configurations of software and services were distributed agreement with the figure 2.2. The software MATLAB Distributed Computing Server (MDCS) was installed and configured in nodes Server and Worker, while the software Parallel Computing Toolbox (PCT) was installed and configured in the Client node. As for the Job Manager, the same is contained in the software MDCS, being activated during the installation and configuration of MDCS in COCGT. Those are the software more important of the COCGT system.

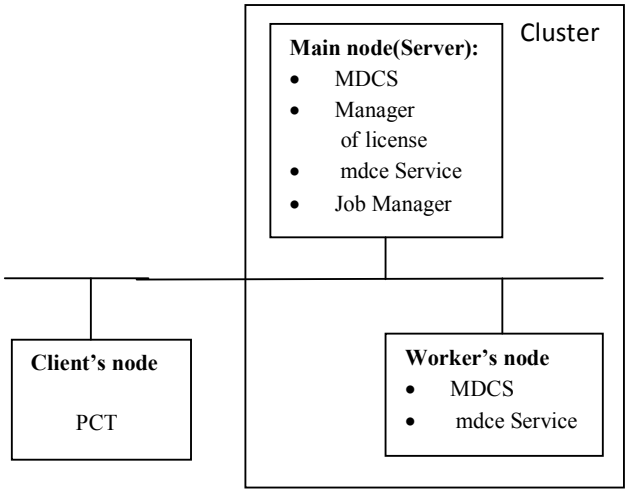

Figure 2.2 – Block diagram of the COCGT

2.4.3. Testes with the cluster

i) Test 1: Determination of singular values with application Girko's law

The Girko's law establishes that the eigenvalues of a random matrix N for N, whose elements are obtained of a normal distribution, tend to lie inside a circle of radius sqrt(N) for N large 'values. This law can be represented by the following MATLAB code [1]:

```
N = 1000
Plot (eig (randn (N)) / sqrt (N), '.');
```

The execution of the code above generates the Figure 2.3 below. Each point represents 1 (one) eigenvalue. It is observed that most of the eigenvalues lives inside in a circle of ray equal to 1 (one) and are centered in the origin of the axes, an indication that, for the Girco's law, the value of a eigenvalue doesn't exceed the square root of the size of the matrix. The Girko's law can be modified to be applied in the determination of singular values [1]. For the application of this law in the determination of SVD (Singular Values Decomposition), through MATLAB, the values of "max (svd (randn (N)))" relative to the arbitrary values of the variable "N" are fixed. The application of the modified Girko's law reduces the time of computation, allowing the numeric estimate of the singular values for any normal and random matrix [1].

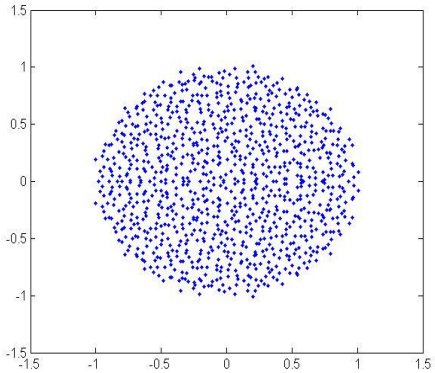

Figure 2.3. Eigenvalues of random and square matrix of size 1000, normalized for 1/sqrt (1000); the elements of the matrix are drawn from normal distribution

The parallel code of the Figure 2.4 generates a random matrix starting from normal distribution and her SVD is determined, besides registering the time elapsed in the loop execution. The "parfor-loop" of the code below has as basic concept the same of "for-loop" (MATLAB pattern). The difference among them is in the procedure, for which the body of the loop (loop declarations) is executed. With the "parfor-loop", the execution is accomplished in parallel and with "for-loop", the execution is made in the sequential way [5],[6].

```
y = zeros (1000, 1);
tic
parfor n=1:1000
    y (n) = max (svd (radn (n)));
end
toc
plot (y)
```

Figure 2.4.- Parallel code in MATLAB for SVD determination of normal and random matrix

This same parallel code (Figure 2.4) was used by the literature [1] for determination of the parameters of performance: Execution Time, Speedup and Efficiency in function of the labs number. The data of the mentioned literature were obtained in a system similar to our cluster (COCGT). Therefore, for the comparison of results, the parameters above also were determined in function of the labs number. In the COCGT, the operation conditions of 1, 2, 3 and 4 labs were used. The labs were fixed by the parallel command "MATLAB pool". The determination of the times of execution, for each situation (for each lab number), made possible the determination of the values of "speedup" (relative) and of the "efficiency".

ii) Test 2: Evaluation of the cluster with the application of the benchmark Linpack HPL

The objective of that test was the performance evaluation of the COCGT through of the benchmark "HPCCLINPACK", an implementation for MATLAB of the benchmark "Global HPCC HPL", which is own Linpack in her version HPL (High Performance Linpack). That benchmark proposes the resolution of a system of linear equations obtained of a real matrix "A" of size (n , n) and of a column vector "B" with size "n", both obtained random way. Therefore, the time spend in the resolution of the equation A*X = B, whose solution is given by $X = (A^{-1}) * B$, is measured. That time corresponds to the metric of performance of the used system. The program HPCCLINPACK contained in MATLAB and used to solve this problem is presented in Figure 2.5.

```
m=input('which is the size of the matrix?');% "m" is the dimension of a
square and random matrix
matlabpool open local 4; % Fixe the labs number
spmd
% Create a distributed matrix in the 2d block cyclic distribution and a
distributed column vector 1d
A=codistributed.randn(m,m,codistributor2dbc);
b=codistributed.rand(m,1);
% Time of solution of the linear system
tic;
x=A\b;
t=toc;
%Need to convert to a 1d distribution for the cheking cod below
A=redistribute(A, codistributor1d);
```

```
% Compute scaled residuals
r1=norm(A*x-b,Inf)/(eps*norm(A,1)*m);
r2=norm(A*x-b,Inf)/(eps*norm(A,1)*norm(x,1));
r3=norm(A*x-b,Inf)/(eps*norm(A,Inf)*norm(x,Inf)*m);
if max([r1,r2,r3])>16
    error('Failed the HPC HPL benchmark');
end
end
% Performance in GFLOPS;
TamDados=8*m^2/(1024^3)
perf=((2/3)*m^3+(3/2)*m^2)/max([t{:}])/1.e9
tmax=max([t{:}])
matlabpool close
```

Figure 2.5. Code of the benchmark HPCCLINPACK used in present work.

In that tests, squares matrix and columns vectors with dimensions equal to 10000 and 12000 were used. Through the measures of processing time, speedup and efficiency, in relation to the labs (cores) number and the matrix dimensions above were fixed.

iii) Test 3: Behavior of the COCGT in relation to the use of "parfor-loop" and "spmd"

The objective of that test was the verification of the COCGT behavior in relation to the use of "parfor-loop" and "spmd" in different codes, but with the same “problem” (calculations) and same “problem size” (amount of calculations) Here, it is opportune a definition of "spmd” (Single Program Multiple Dates)": It is an environment of MATLAB parallel programming that allows the creation of a code block (spmd-end) that is executed in all of the available labs (workers) in the cluster. For the test, the following codes were used (Figure 2.6):

Code a:

```
matlabpool nlab
Y=zeros (1000,1);
tic
parfor n=1:1000
   Y (n) = max (svd (randn (n)));
end
toc
SVmax=max (Y)
matlabpool close
```

Code b:

```
matlabpool nlab
D=distributed.randn (1000);
tic
spmd
   m1=max (svd (D));
end
toc
SVmax=m1
matlabpool close
```

Figure 2.6. Codes used in verification of the COCGT behavior in relation to the use of "parfor" and "spmd

The two codes above have different structures: The “code a” of the Figure 2.6 is a loop that uses the command "parfor", while the “code b” uses the "spmd block” (spmd -.end). The “problem” and “problem size” are the same for both codes, being that the “problem” is the calculation of the "maximum singular value" and “problem size” correspond to the dimension (n, n) of the random matrix, being in this case n=1000. The calculated value of “maximum singular value” should be the same in the two codes. The metric used for the evaluation of the performance was the time consumed in the execution of each code. Two batteries of tests were accomplished: one with variation of 1 up to 4 labs for the “code b" and the other with variation of 1 up to 8 labs for the “code a", fastened through the MATLAB command "matlabpool."

iv) Test 4: The cluster behavior in relation to use of "parfor" and "spmd" in the determination of SVD on Square Matrix of Dimension above 1000

This section of tests had the objective of verifying the behavior of the COCGT in the determination of SVD with "parfor" and "spmd" in square and random matrix with dimensions above 1000 The codes of the Figure 2.7 – “code c” and “code d” – are the same used in test previous (Test 3), but now with matrix of dimensions equal to 1500 and 7000, where the "maximum singular value" will be evaluated with "parfor and “spmd” respectively.

Code c:

```
matlabpool nlab
Y=zeros (1500,1);
tic
parfor n=1:1500
   Y (n) = max (svd (randn (n)));
end
toc
SVmax=max (Y)
matlabpool close
```

Code d:

```
matlabpool nlab
      D=distributed.randn (7000);
      tic
      spmd
         m1=max (svd (D));
      end
      toc
      SVmax=m1
      matlabpool close
```

Figure 2.7. Codes used in the verification of the COCGT behavior in relation to the use of "parfor" and "spmd" on Square Matrix of Dimensions 1500 and 7000

3. RESULTS AND DISCUSSION

3.1. Results Obtained with the Test 1

The results obtained with the accomplishment of the Test 1 (item 2.4.3 - i), which was an application of the modified Girko’s law to determine SVD of matrix, are in the Tables 3.1 and 3.2. A comparison among these tables reveals that the times of execution of the parallel code

(Figure 2.4), contained in the Table 3.1, are larger in relation to the times than are contained in the Table 3.2. This means that our system was, for all of the labs numbers, faster than the system of the literature [1]. It can be that this difference is due to the difference among the architectures of computers involved in the tests. Our processor model has 4 (four) cores and the following specification:

Intel Cores i7 860 - 2,8 GHz. CACHE 8 MB. LGA 1156.

This model has two technologies of data processing. One of them is the technology "Turbo Boost" that is activated when the Operating System (OS) request the highest state of performance of the processor. Therefore her clock frequency is increased. The other

Table 3.1 – Determination of the performance of a parallel code in function of the amount of labs of the cluster – literature data [1]

Nº Labs (cores)	Processing Time (seconds)	Relative Speedup	Efficiency (%)
1	870.1	1.00	100
2	487.0	1.79	89
3	346.2	2.51	83
4	273.9	3.17	79

Table 3.2 – Determination of the performance of a parallel code in function of the amount of labs of the COCGT

Nº Labs (cores)	Processing Time (seconds)	Relative Speedup	Efficiency (%)
1	166.5±0,2	1.00	100
2	99.0±0,2	1.68	83
3	88.2±0,4	1.89	63
4	84.4±0,2	1.97	49

is denominated of multiple-segmentation or processing of multiple-tasks, presenting, a parallel architecture. This allows each nucleus of the processor to work with two tasks at the same time, what reduces the time of processing of the algorithm. In relation to the model of the literature, it was only informed that was a processor of 4 cores.

3.2. Results Obtained with the Test 2

The results obtained with the code used in the Test 2 (Figure 2.5) are contained in the Table 3.3 below. The analysis of this table reveals that: 1) it happens mistake for lack of memory when 1 or 2 labs are used in the processing of square and random matrix of dimensions corresponding to 10000 and 12000; 2) in the processing of the matrix with dimension equal to 10000, when it is used 3 and 4 labs, the times expenses are about 28 and 27 seconds, respectively; 3) in relation to processing of the matrix with dimension equal to 12000, being used the same amount of labs (3 and 4 labs), the times expenses are about 48 and 45 seconds, respectively. The times here obtained are small in if treating of resolution of linear equations systems starting from matrix with size relatively large. This denotes a good performance of the COCGT. The metric FLOPS, another important metric that evaluates the cluster performance, generated values in the interval from 24 to 26 GFLOPS in relation to the previously mentioned matrix.

Table 3.3 – Performance of the COCGT in the resolution of systems of linear equations obtained of square and random matrix of dimensions 10.000 and 12.000

Matrix Dimension	Nº Labs (cores)	Processing Time (seconds)	Performance (GFLOPS)	Data Size (GB)
10000	01*	-	-	0.7451
	02*	-	-	
	03	27.82±0.05	24.47±0.21	
	04	26.89±0.18	25.60±0.70	
12000	01*	-	-	1.0729
	02*	-	-	
	03	47.60±0,37	23.97±0,04	
	04	44.97±1,53	24.80±0,17	

(*) . It was not possible to obtain the data for lack of memory in the computation of the problem with 1 and 2 labs (cores)

3.3 - Results Obtained with the Test 3

The behavior specifically depends on the computer/processor type, environment of parallel programming and algorithm type. In our case (COCGT), different behavior happened in relation to the times of processing among the two parallel codes presented in the Figure 2.6 (item 2.4.3-iii), both possessing the same problem with same size. In this case, the “same problem” is the determination of SVD of a square and random matrix, while the “same problem size” is the dimension of the matrix (n, n), that in the current case it is n=1000. For the code "parfor" (Figure 2.6 – code a), the time of processing decreased with the increment of labs number from 1 to 8 labs (Table 3.4). This means that the performance increased with the labs number; the time of processing declined of 169,55 seconds (obtained with 1 lab) for 43,20 seconds (obtained with 8 labs). For the code with "spmd” (Figure 2.6 - code b), in

agreement with the Table 3.5, were generated very small and close times of processing (< 1 s) with the use from 1 to 4 labs This means that the work load (problem size) for that code was very small; only 1 (one) lab was enough to process the code in less than 1 (one) second. Since the same hardware architecture (same cluster), the same programming environment (MATLAB), the same problem with same size were maintained, this difference is due to the two types of employed codes. It is very probable that the "code a" of the Figure 2.6 (with "parfor") is in a great relationship: "number of processors/work load". While the code "b" of the Figure 2.6 (with "spmd" block), needs an increase of size of the problem (increase of the work load) to reach a relationship "number of processors/work load" in way to produce a good performance. However, the executions of the two codes above produced the same result for the "maximum singular value" (about 63) of a matrix random of dimension n=1000 (Tables 3.4 and 3.5).

Table 3.4 - Behavior of the COCGT regarding the command "parfor" in the determination of SVD of a square and random matrix of dimension 1000.

Nº Labs (cores)	Processing Time (second)	Relative Speedup	Efficiency (%)	Maximum Singular Value
01	169.5529	1	100	63.2856
02	100.0698	1.69	84	63.2747
03	88.9894	1.91	64	63.4354
06	49.1429	3.45	57	63.2365
07	46.3321	3.66	52	63.3045
08	43.2354	3.92	49	63.1232

Table 3.5 – Behavior of the COCGT regarding the block of code "spmd -end" in the determination of SVD in a square and random matrix of dimension 1000

Nº Labs (cores)	Processing Time (second)	Relative Speedup	Efficiency (%)	Maximum Singular Value
01	0.9324	1	100	63.0787
02	0.7798	1.19	59	63.2471
03	0.7609	1.22	41	63.0142
04	0.8199	1.13	28	63.1278

3.4 - Results Obtained with the Test 4

The results regarding the Test 4 are in the Tables 3.6 and 3.7. The analysis of the data reveals that the determination of SVD of a square matrix of dimension n=1500, using the code with "parfor", was slower than the determination of SVD of a square matrix with n=7000, using the code with "spmd". The processing times generated were 883 - 524 s, for code with “parfor” and 237-144 s, for code with “spmd”´. The intervals of time here mentioned were obtained with the labs numbers varying of of 1 to 4 labs. Now, both codes above possess the same behavior: The times of processing decreased with the labs number varying of 1 up to 4. However; the code with “spmd” besides having larger performance (smaller times of processing), supports a larger work load (matrix size) in relation to the code with "parfor".

Table 3.6 – Behavior of the COCGT in relation to the Application of "parfor" in the determination of SVD in a matrix of Dimensions 1.500 for 1.500

Nº Labs (cores)	Processing Time (second)	Relative Speedup	Efficiency (%)	Maximum Singular Value
01	883.2376	1	100	77.8250
02	592.3603	1.49	75	77.5470
03	542.3689	1.63	54	77.5533
04	524.5181	1.68	42	77.6263

Table 3.7 – Behavior of the COCGT in relation to the application of the "spmd" block for the determination of SVD in matrix of dimension equal to 7000

Nº Labs (cores)	Processing Time (second)	Relative Speedup	Efficiency (%)	Maximum Singular Value
01	236.5987	1	100	167.2956
02	162.7237	1.45	73	166.9973
03	147.4094	1.60	53	167.5415
04	144.0799	1.64	41	167.4436

4. C O N C L U S I O N S

The determination of SVD (Singular Values Decomposition) of a square and random matrix of dimension equal to 1000, through the Girko’s modified law, revealed that the COCGT generated, for 4 labs, a time of processing (84.4 s) about 3 times smaller than the time (273.9)

produced by the cluster of the literature [1]. This way, the COCGT were faster in relation to the other cluster, both using the same procedure and experimental conditions.

Low processing times were obtained in the resolution of systems of linear equations generated for square and random matrix with dimensions relatively high: 10000 and 12000. The times expenses in the resolution of those equations for 4 labs were: 26,9 ± 0,2 seconds, for the matrix dimension equal to 10000, and 45,0 ± 1,5 seconds, for the matrix dimension equal to 12000. This means an excellent performance of our system (COCGT). In terms of FLOPS, it was obtained a significant value of 25 GFLOPS for both the matrix.

The study of the COCGT behavior regarding applications of "parfor" and "spmd" in a same problem with same size revealed adverse results among those two commands of programming parallel with MATLAB. Therefore, the determination of the "maximum singular value" of a random and square matrix of dimension equal to1000 revealed that the code with "parfor" it produced results in that the time of processing decreased and the speedup increased with increase of the labs number in an interval from 1 to 8 labs; a time of 43.23 s using 8 labs was obtained. Already the code with "spmd", as discussed in the item 3.3, needs a larger size of problem (a large matrix size) to have a relationship "number of processors/matrix size" capable to produce a good performance. On the other hand, the values of the "maximum singular value", regarding "parfor" and "spmd", were constant and about 63 for each lab as it was of waiting, because it is the resolution of a same problem computational (with same size).

The determination of the behavior of COCGT, now with the "maximum singular value", being calculated from square and random matrix of dimensions equal to 1500 (using "parfor") and 7000 (using "spmd"), was done. The behavior with use of "parfor" was the same happened previously using a square matrix with size equal to 1000. However, the times of processing of the matrix with dimension equal to 7000, using "spmd", are smaller than the processing times of the matrix of dimension 1500, using "parfor" (see tables 3.6 and 3.7). Therefore, the code with "spmd", besides having a larger performance, supports larger work load (size of problem) in relation to the code with "parfor."

ACKNOWLEDGMENTS

The authors are grateful to PETROBRAS/CNPq for the technical/financial support

REFERENCES

1. P. Luszczek, Enhancing Multicore System Perfomance Using Parallel Computing with MATLAB, The MathWorks News&Notes, Technical Articles, (2008). www.mathworks.com/parallelcomputing/technicalliterature.htm

2. C.C. Dantas, S. B. Melo, E. F. Oliveira, F. P. M. Simões, M. G. dos Santos, V. A. dos Santos, Measurement of density distribution of cracking catalyst in experimental rise with sampling procedure, nuclear instrum. and Methods in physis, B 266 (2008), pp.841-848

3. "FLOPS", http://pt.wikipedia.org/wiki/FLOPS, Page posted in: May 27 of 2011

4. "Benchmark", http:/pt.wikippedia.org/wiki/Benchmark, Page posted in: February 2 1, 2011

5. E. Ellis, Solving Large-Scale Linear Algebra Problems Using SPMD and Distributed Arrays. The MathWorks News &Notes, Technical Articles, www.mathworks.com/parallelcomputing/technicalliterature.htm

6. C. Moler, Parallel MATLAB: Multiple Processors and Multiple Core, The MathWorks News &Notes, Technical Articles, (2007) http://www.mathworks.com/company/newsletters/technicalarticles.html