Parallel-In-Space Implementation of Transient Stability Analysis on a Linux Cluster With Infiniband

Gürkan Soykan
Department of Electrical Engineering
École Polytechnique de Montréal
Montreal, Quebec Canada
Email: gurkan.soykan@polymtl.ca

Alexander J. Flueck
Electrical and Computer Engineering
Illinois Institute of Technology
Chicago, Illinois 60616
Email: flueck@iit.edu

Hasan Dağ
Information Technologies Department
Kadir Has University
Istanbul, Turkey
Email: hasan.dag@khas.edu.tr

Abstract— **On-line transient stability analysis is an inevitable way to provide real time power system security and control. Parallel computing is one of the most viable ways to perform on-line transient stability analysis. This paper presents the performance results of a parallel-in-space algorithm based on a multilevel partitioning scheme on an Infiniband cluster system. The algorithm decreases the transient stability simulation time using METIS for partitioning in conjunction with the linearized update solution process of the Very Dishonest Newton Method when solving the differential-algebraic systems of equations [1]. Two real power systems, a 3493-bus system with 844 generators and a 7935-bus system with 2135 generators, are tested on a Linux-cluster system with 16 nodes. Each node has 2 multi-core processors and is connected to each other by Infiniband network. The properties of a test system have a large impact on the performance of the parallel algorithm used, since it affects communication duration.**

I. Introduction

Transient stability analysis is used to determine the dynamic behaviour of an electric power system after a large disturbance. In terms of power system planning and power system operations, this analysis is a vital step for the operations performed on Energy Management centers. In this analysis, a differential algebraic model is used to describe the electric power system. The model consists of both differential and algebraic equations (DAEs). They are shown as

$$\dot{X} = f(X,V) \tag{1}$$

$$0 = g(X,V). \tag{2}$$

Equation (1) describes the dynamic devices in the power system and equation (2) indicates the algebraic equations of synchronous machines and power system network. So, X is called as a vector of state variables and V is called as a vector of algebraic variables.

In the solution of transient stability analysis, the differential equations are discretized first by one of the numerical integration methods, such as trapezoidal integration method. Then the integration equations and algebraic equations together are solved simultaneously at each time step. Newton's method is used to solve non-linear algebraic equations. In many production grade programs, LU factorization is implemented to solve the sparse set of linear equations resulted in Newton's method. Very Dishonest Newton Method (VDHN) has been preferred in order to achieve faster solution in those programs [2].

In power system operation, on-line transient stability analysis is an unavoidable way to perform real-time power system security and control in energy management system. Parallel computing and fast numerical methods are used for obtaining on-line transient stability analysis [3]. Therefore, several partitioning strategies have been developed to be able to achieve parallel processing for transient stability analysis.

In the literature, the parallel algorithms used for transient stability analysis are classified as parallel-in-space and parallel-in-time [4]. Parallel-in-space algorithms break the original system down into subsystems among processors and each time step computation is performed in parallel. Parallel-in-time algorithms solve multiple time steps simultaneously. In order to increase the effect of parallelism, parallel-in-space and parallel-in-time algorithms can be used together.

Because of widespread usage of distributed computing systems, several studies dealing with the parallel-in-space implementation of power system transient stability analysis have focused on a distributed approach. A different partitioning scheme based on the factorization path tree was applied to the parallelization of a VDHN version of the simulation on a distributed memory system, IBM-SP2 parallel computer [4]. The implementation results for a 92-generator, 616-bus system were presented. A generalization of the subtree-to-subcube mapping approach was used in the parallel-in space approach [5]. A 150-generator, 1007-bus and 450-generator, 3021-bus power systems were tested on IBM-SP2 parallel computer. The size of power system was increased so as to achieve better parallel performance in [5]. The distribution of the machine equations could be a bottleneck depending on the power system network. A parallel computation algorithm for network calculation in the transient stability calculation was proposed in [6]. This algorithm is based on the processing tree [7]. A multilevel power network partition scheme based on

978-1-4673-2308-6/12/$31.00 ©2012 IEEE

power network regional characteristics was utilized on both distributed and shared memory systems simultaneously [8]. Some optimization schemes were implemented in order to decrease computation and communication time of the proposed solution method. The largest test system used had 248 generators and 2115 buses. Different partitioning schemes based on the graph partition algorithm were implemented for parallel simulation of power system dynamics. The properties of the power system were regarded in the partitioning strategy [9], [10]. Another spatial parallel algorithm based on geographical location was applied on a PC cluster [11]. 173-generator, 1923-bus system was used to analyze the performance of the proposed algorithm. A METIS based multi-level partitioning algorithm was proposed to divide computational loads among processors [1].

In this paper, the METIS-based partitioning strategy given in [1] is tested on a Linux cluster. The hardware properties of cluster are very up-to-date. Large scale power system data is used to show the performance of the partitioning strategy. The results indicate the efficiency of the parallel implementation is increased by the up-to-date computer system. The parallel performance for the proposed algorithm depends on mainly two factors: the size of the problem and the communication performance of the network.

II. Transient Stability Analysis

Transient stability analysis is concerned with the effects on generator synchronism due to a large disturbance such as loss of a large load or loss of generators. A differential algebraic model is used to describe the power system in this simulation method. Dynamic devices are modeled as a set of differential equations and the power system network is modeled as a set of algebraic equations. The differential-algebraic equations are

$$\dot{X} = f(X,V,u), \qquad X(0) = X_0 \tag{3}$$

$$I_1(X,V) = \mathbf{Y_N}V, \qquad V(0) = V_0 \tag{4}$$

where $\mathbf{Y_N}$ is the $N \times N$ bus admittance matrix of the network, N is the number of buses in the system and I_1 is the injected current vector. Equation (3) can be discretized by trapezoidal integration method. Then, equations (5) and (6) are written.

$$F(X_{n+1},V_{n+1}) \triangleq X_{n+1} - X_n - \frac{h}{2}[f(X_{n+1},V_{n+1}) + f(X_n,V_n)] = 0 \tag{5}$$

$$G(X_{n+1},V_{n+1}) \triangleq \mathbf{Y_N}V_{n+1} - I_1(X_{n+1},V_{n+1}) = 0 \tag{6}$$

where h is an integration time step. Using the rectangular coordinate representation of I_1, V and $\mathbf{Y_N}$ as $I_1{}^e$, V^e and $\mathbf{Y^e_N}$ in equations (5) and (6), one can obtain

$$F(X_{n+1},V^e_{n+1}) \triangleq X_{n+1} - X_n - \frac{h}{2}[f(X_{n+1},V^e_{n+1}) + f(X_n,V^e_n)] = 0 \tag{7}$$

$$H(X_{n+1},V^e_{n+1}) \triangleq \mathbf{Y^e_N}V^e_{n+1} - I_1{}^e(X_{n+1},V^e_{n+1}) = 0 \tag{8}$$

where $\mathbf{Y_N}$, $I_1{}^e$, V^e, $\mathbf{Y^e_N}$ and H are defined as

$$\mathbf{Y_N} = \mathbf{G_N} + \mathbf{jB_N},\ V^e = \begin{bmatrix} V^r \\ V^i \end{bmatrix}, I_1{}^e = \begin{bmatrix} I_1^r \\ I_1^i \end{bmatrix}$$

$$\mathbf{Y^e_N} = \begin{bmatrix} \mathbf{G_N} & -\mathbf{B_N} \\ \mathbf{B_N} & \mathbf{G_N} \end{bmatrix}, H(X,V) = \begin{bmatrix} Re(G(X,V)) \\ Im(G(X,V)) \end{bmatrix},$$

The linear equations are

$$\mathbf{J}^{(k)}_{n+1}\Delta\mathcal{X}^{(k)}_{n+1} = -\mathcal{F}_{n+1}{}^{(k)} \tag{9}$$

the unknown variables updated are

$$\mathcal{X}^{(k+1)}_{n+1} = \mathcal{X}^{(k)}_{n+1} + \Delta\mathcal{X}^{(k)}_{n+1} \tag{10}$$

where

$$\mathcal{F} = \begin{bmatrix} F \\ H \end{bmatrix}, \mathcal{X} = \begin{bmatrix} X \\ V \end{bmatrix}$$

$$\begin{bmatrix} F \\ H \end{bmatrix} = -\begin{bmatrix} \frac{\partial F}{\partial X} & \frac{\partial F}{\partial V^e} \\ \frac{\partial H}{\partial X} & \frac{\partial H}{\partial V^e} \end{bmatrix} \begin{bmatrix} \Delta X \\ \Delta V^e \end{bmatrix}$$

The Jacobian has the following structure:

$$\mathbf{J} = \begin{bmatrix} \frac{\partial F}{\partial X} & \frac{\partial F}{\partial V^e} \\ \frac{\partial H}{\partial X} & \frac{\partial G}{\partial V^e} \end{bmatrix} = \begin{bmatrix} \mathbf{J_A} & \mathbf{J_B} \\ \mathbf{J_C} & \mathbf{J_D} \end{bmatrix}$$

$\mathbf{J_D}$ is written

$$\mathbf{J_D} = \begin{bmatrix} \mathbf{G_N} & -\mathbf{B_N} \\ \mathbf{B_N} & \mathbf{G_N} \end{bmatrix} - \begin{bmatrix} \frac{\partial I_1^r}{\partial V^r} & \frac{\partial I_1^r}{\partial V^i} \\ \frac{\partial I_1^i}{\partial V^r} & \frac{\partial I_1^i}{\partial V^i} \end{bmatrix}. \tag{11}$$

Equation (11) is composed of two parts denoted by $\mathbf{Y^e_N}$ and $\mathbf{Y^e_D}$ respectively

$$\mathbf{Y^e_N} = \begin{bmatrix} \mathbf{G_N} & -\mathbf{B_N} \\ \mathbf{B_N} & \mathbf{G_N} \end{bmatrix} \quad \textit{and} \quad \mathbf{Y^e_D} = -\begin{bmatrix} \frac{\partial I_1^r}{\partial V^r} & \frac{\partial I_1^r}{\partial V^i} \\ \frac{\partial I_1^i}{\partial V^r} & \frac{\partial I_1^i}{\partial V^i} \end{bmatrix}.$$

Then, equations (12) and (13) are obtained from (9)

$$\Delta X_{n+1} = -\mathbf{J_A}^{-1}\left[F_{n+1} + \mathbf{J_B}\Delta V^e_{n+1}\right]. \tag{12}$$

$$\mathbf{J_1}V^e_{n+1}{}^{(k+1)} = I_1^{e(k)} + (\mathbf{Y^e_D} - \mathbf{J_C}\mathbf{J_A^{-1}}\mathbf{J_B})V^e_{n+1}{}^{(k)} + \mathbf{J_C}\mathbf{J_A^{-1}}F_{n+1}{}^{(k)}. \tag{13}$$

$\mathbf{J_1}$ is defined as:

$$\mathbf{J_1} = \mathbf{J_D} - \mathbf{J_C}\mathbf{J_A^{-1}}\mathbf{J_B}. \tag{14}$$

State variables and algebraic variables are obtained from (12) and (13) in each time step. Equation (13) can be solved by using LU factorization, forward and backward substitution.

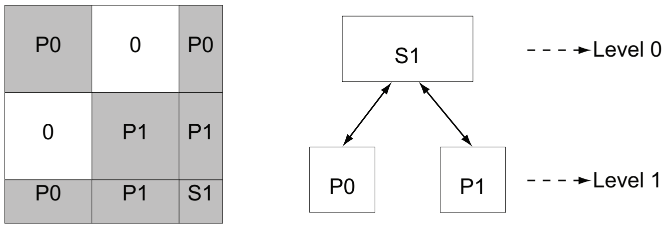

Fig. 1. Structure of reordered matrix and separator tree for two processors.

III. Parallel-In-Space Algorithm for transient stability analysis

Solution of the transient stability analysis in a serial environment is made up of in three main steps: data preparation, initialization and simulation. The most time consuming part of the analysis is the simulation part. METIS-based partitioning scheme is applied to the serial solution given the former section.

The partitioning problem is the crucial point of a parallel algorithm. Multi-level partitioning algorithms are attractive for obtaining the most suitable partitioning for the problem. The partitioning of the graph corresponding to the $\mathbf{J_1}$ matrix should reduce the amount of communication in the solution of the Newton update. The multi-level partitioning algorithms consist of three phases. Firstly, the algorithm reduces the size of a given graph by collapsing vertices and edges, then partitions the coarsened graph and finally uncoarsens the partitioned graph to construct the given graph which is fully partitioned. METIS is an open-source software package using multi-level partitioning algorithms. This tool can be used either to partition irregular graphs or to compute fill-reducing orderings of sparse matrices. In the solution of Transient Stability Analysis, A METIS-based partitioning scheme was chosen to create a parallel algorithm because of two reasons. Since $\mathbf{A}$ matrix in $\mathbf{A}x = b$ linear system is a large sparse matrix, it should be reordered to minimize the fill-ins before the factorization. Secondly, it has been chosen to overcome the difficulty in load balancing and minimizing communication. The multi-level nested dissection algorithm from METIS creates an effective partitioning structure to distribute data to each of the available processors. Thus, this strategy handles both ordering and obtaining the partition information. The structure is similar to a binary tree.

Fig.1 shows the new structure of the matrix and the communication structure of the processors. Assuming that two processors are available, this structure is obtained. In Fig.1, S1, P0 and P1 are described as

- S1: Separator (a mutual part of both processors),
- P0: Partition of the first processor,
- P1: Partition of the second processor.

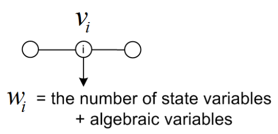

Fig. 2. The assignment of vertex weight.

There are two levels to show the task and communication pattern. The first level has a S1 sub-matrix, the second level is constructed from P0 and P1 sub-matrices. P0 and P1 are the children of S1 in this structure. The submatrices at the same level can be factorized simultaneously, but their parent can not be factorized until all its children are done and sent the update matrix to their parent. This is a bottom-up computation according to the separator tree for LU factorization. Also, the above structure can be used in the triangular solver that is forward and backward substitutions. The triangular solver moves in the separator tree once bottom-up and once top-down direction. Additional details about the structures for two processors and four processors can be found in [1].

In the parallel implementation, when the program is started, the master processor reads all the data from files. The new ordering information and the size of the partitions are obtained by using the multi-level nested dissection method. Then the master processor distributes the data to the other processors. In the remaining parts of the program, each processor continues the calculations by using its own partition information and some data from its neighbours. When equation (13) is solved, the communication will be done depending on the separator tree structure. Also the communication is necessary for the determination of the convergence of the Newton solution in each time step. The steps of parallel algorithm can be given as below:

1) Data preparation part of serial solution is done by master processor,
2) The partition structure and ordering information is obtained by master processor,
3) The distribution of the data among the processors is performed,
4) All processors compute the initial value of state variables based on a complex voltage snapshot,
5) The simulation part of Transient Stability Analysis is performed by all processors (Inter-processors communication is essential for this part).

In the proposed partitioning algorithm, a weighted graph in METIS is used to balance the computational load. The number of generators and the number of branches assigned to each processor has a big impact on the computational load. Because they affect the number of differential equation and the sparsity of the network matrix on each processor. Determining suitable vertex weights can help to create a balanced distribution. The number of state variables plus algebraic variables for any vertex are used as the vertex weight in the graph. It is given in Fig. 2. v_i shows the i. vertex and w_i presents the weight of v_i. Depending upon the structure of transient stability problem, the defined weight for each vertex as in Fig. 2 constructs the balanced distribution as possible in terms of computation. State variables are distributed to the processors with respect to obtained partitioning information. Thus, the communication among the processors are minimized in the simulation stage of transient stability solution. If equal number of generators distributed among processors does not guarantee the minimum communication requirement. Each generator has different communication complexity. In order to obtain a good load balancing and minimum communication, the weighted graph strategy is used.

IV. Test Results

All simulations were conducted on a Linux-cluster system that has 16 nodes. Each node has 2 processors that are Intel Xeon (E5540) 2.43 GHz. Each processor has 4 cores. So the number of cores on the system are 128. The amount of RAM for each core is 3 GB. The nodes are connected by Infiniband.

Two power systems were tested to observe the performance of the implementation. They are a 3493 bus system (with 844 generators, 6689 branches, 2565 loads) and a 7935 bus system (with 2135 generators, 13624 branches, 5718 loads). A three phase fault was used for the transient stability simulation. The fault occurred at 0.4 seconds and was cleared at 0.6 seconds. The total simulation time was 2 seconds. The integration time step was 0.01 seconds.

MPI [12], a standard communication library, was used as the message passing platform for the parallel computation in this implementation. No compiler optimization was used while measuring the processing time.

The performance of the parallel implementation is measured by speedup and efficiency. Speedup is a measure of how much speed gained w.r.t to time of the serial solution and is the ratio of the runtime of a serial solution to the parallel runtime. The following structure shows speedup (Spd) for p processors

$$Spd = \frac{T_s}{T(p)} \tag{15}$$

where T_s represents the execution time of a serial program and $T(p)$ is the execution time of the parallel transient stability program on p processors with the same test case. The value of Spd is between 0 to p. If it is equal to p, a program is said to have linear speedup. However, the value of speedup is less than p in most parallel solutions because of the parallel overhead. There are three main sources of parallel overhead: idle time, communication and extra computation [12]. The parallel runtime can be measured by using MPI timing routine, which is called as MPI_Wtime. In addition to speedup, efficiency (E) is a measure of process utilization. It is defined as:

$$E = \frac{Spd}{p} \tag{16}$$

The value of E is between 0 and 1. If it is less than 1, the program is exhibiting slowdown. If it is equal to 1, the amount of work done by parallel code is the same as the amount of work done by the serial code.

Time results of the parallel simulation for two different power systems are listed in Table I. By using (15) and (16), speedup and efficiency results are given in Table II, Table III, Fig. 3 and Fig. 4. According to Table III and Fig. 4, the larger power system produces higher efficiency, whereas the smaller system saturates earlier. Eventhough the system size is doubled, we do not get a double speedup as one might expect. This is due to the fact that the sparsity pattern of power system matrices do not change significantly with the system size. The cluster's properties of the test system is better than the other system used in [1] with regard to processor and network technology. Because of this difference, the performance of partitioning strategy can be seen better from the given results in here. Moreover, the speedup results for this algorithm are encouraging for large scale systems.

TABLE I
Time for parallel transient stability analysis (seconds of wallclock time).

The number of cores	3493 system	7935 system
1	9.03	13.67
2	5.04	6.97
4	2.80	3.55
8	1.80	2.10
16	1.20	1.43
32	0.84	0.99
64	0.92	0.77

TABLE II
Absolute Speedup for parallel transient stability analysis.

The number of cores	3493 system	7935 system
2	1.79	1.96
4	3.23	3.85
8	5.01	6.51
16	7.53	9.55
32	10.75	13.81
64	9.82	17.75

TABLE III
Efficiency for parallel transient stability analysis.

The number of cores	3493 system	7935 system
2	0.90	0.98
4	0.81	0.96
8	0.63	0.81
16	0.47	0.60
32	0.34	0.43
64	0.15	0.28

V. Conclusion

In this paper, a new partitioning scheme is used on a Linux cluster with Infiniband network. The test system properties as well as the hardware technology affect the performance of the parallel implementation. The performance of parallel-in-space implementation in this paper is better than the performance in [1]. The performance of this implementation is also increased by using larger power systems. The size of the power system and the technology of the computer system are the two main factors that impact the performance of the parallel

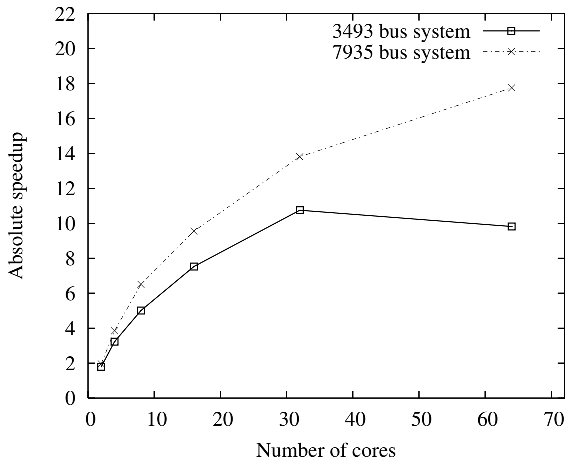

Fig. 3. Parallel speedup of Transient Stability Analysis.

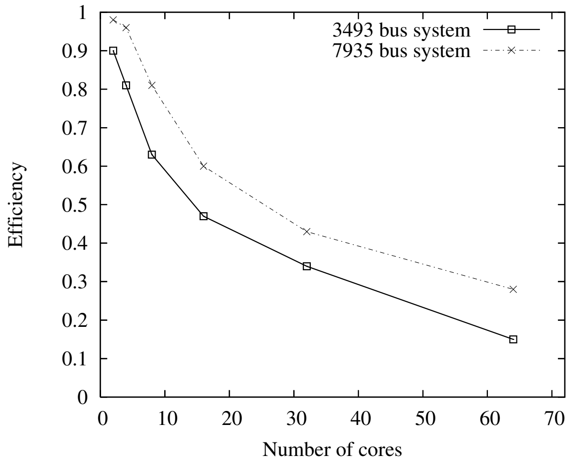

Fig. 4. Parallel efficiency of Transient Stability Analysis.

implementation. For the next step, the distributed and shared memory properties will be taken into account to increase the performance of the partitioning scheme.

REFERENCES

[1] G. Soykan, A. Flueck, and H. Dag, "Distributed memory parallel transient stability analysis on a pc cluster with ethernet," *International Review of Electrical Engineering(IREE)*, vol. 5, no. 3, pp. 1053–1060.

[2] J. Wu, A. Bose, J. Huang, A. Valette, and F. Lafrance, "Parallel implementation of power system transient stability analysis," *IEEE Transactions on Power Systems*, vol. 10, no. 3, pp. 1226–1233, August 1995.

[3] D. J. Tylavsky, A. Bose, F. L. Alvarado, R. Betancourt, K. Clements, G. Heydt, G. Huang, M. Ilic, M. L. Scala, M. Pai, C. Pottle, S. Talukdar, J. VanNess, and F. Wu, "Parallel processing in power system computation," *IEEE Transactions on Power Systems*, vol. 7, no. 2, pp. 629–637, May 1992.

[4] C. Hong and C. Shen, "Parallel transient stability analysis on distributed memory message passing multiprocessors," in *Proceedings of the 4th International Conference on Advance in Power System Control, Operation and Management*, Hong Kong, November 1997, pp. 304–309.

[5] ——, "Implementation of parallel algorithms for transient stability analysis on a message passing multicomputer," in *IEEE Power Engineering Society Winter Meeting*, vol. 2, January 2000, pp. 1410–1415.

[6] M. Nagata and N. Uchida, "Parallel processing of network calculations in order to speed up transient stability analysis," *Electrical Engineering in Japan*, vol. 135, no. 3, pp. 26–36, March 2001.

[7] W. F. Tinney, V. Brandwajn, and S. M. Chan, "Sparse vector methods," *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-104, no. 2, pp. 295–301, February 1985.

[8] J. Shu, W. Xue, and W. Zheng, "A parallel transient stability simulation for power systems," *IEEE Transactions on Power Systems*, vol. 20, no. 4, pp. 1709–1717, November 2005.

[9] C. Huang and M. Chen, *Distributed Network Computing on Transient Stability Analysis and Control*. Springer Berlin / Heidelberg, 2005, vol. 3758, pp. 737–742.

[10] W. Xue and S. Qi, "Multilevel task partition algorithms for parallel simulation of power system dynamics," in *Computational Science - ICCS 2007*, ser. Lecture Notes in Computer Science. Springer Berlin/Heidelberg, 2007, vol. 4487/2007, pp. 529–537.

[11] J. Ye, Z. Liu, and L. Zhu, "The implementation of a spatial parallel algorithm for transient stability simulation on pc cluster," in *2nd IEEE Conference Industrial Electronics and Applications*, Harbin, China, May 2007, pp. 1489–1492.

[12] P. S. Pacheco, *Parallel Programming with MPI*. Morgan Kaufmann Publishers, Inc, 1997.