A Parallel Transient Stability Simulation for Power Systems

Jiwu Shu, Wei Xue, and Weimin Zheng

***Abstract*—As power systems continue to develop, online dynamic security analysis and real-time simulation using parallel computing are becoming increasingly important. This paper presents a novel multilevel partition scheme for parallel computing based on power network regional characteristics and describes the design and implementation of a hierarchical block bordered diagonal form (BBDF) algorithm for power network computation. Some optimization schemes are further proposed to reduce the computation and communication time and to improve the scalability of the program. The simulation results show that, for a large network with 2115 nodes, 2614 branches, 248 generators, and 544 loads, the proposed algorithms and schemes run ten times faster on a cluster system with eight CPUs than on a single CPU. Thus, they satisfy the real-time simulation requirement for large-scale power grids.**

***Index Terms*—Parallel computing, power system, real-time simulation, transient stability analysis.**

NOMENCLATURE

WR	Waveform relaxation.
BBDF	Block bordered diagonal form.

I. INTRODUCTION

POWER system transient stability analysis is an important tool in power system research. With the development of power system scale, the computation tasks are becoming increasingly heavier and more complex. Traditional sequential computation is inadequate for online dynamic security analysis and real-time simulation. Hence, the key to realizing the real-time dynamic simulation of large-scale power systems is to find new simulation algorithms and parallel software for use with high-performance computers. The recent development of high-performance computing technology, especially the wide application of cost-effective cluster systems, has made real-time simulation for large-scale power networks feasible.

Research on parallel algorithms and their application in transient stability analysis has been well developed in the last 15 years [1], [2]. Kron proposed the domain decomposition method to accomplish large-scale power system simulation, which was considered as the genesis of the research in this field [3]. This method was to partition the whole power network into blocks for simulation. It was an exciting idea and gave a new momentum to the research in this field. Up to now, there have been many research topics related to parallel transient stability algorithms and parallel simulation of power systems [4]–[13]. The research has developed in two directions: spatial parallelism and time parallelism. Spatial parallel algorithms, including the partition method and parallel factoring algorithm, take a time-domain integration method and break each time step computation down into subtasks among processors. As a coarse granularity parallel algorithm, the partition method is simple to apply and can achieve higher efficiency on distributed architecture, while the parallel factoring scheme is better on shared memory machines. In order to achieve better performance on more processors, solutions that use the simultaneous multiple time steps, such as the waveform relaxation (WR) method and the parallel-in-time Newton algorithm, have been introduced into parallel transient stability simulations. These time parallel algorithms enlarge the size of problems solved simultaneously and effectively improve the speed of simulations. However, it is difficult to achieve a large degree of parallelism while maintaining a high convergence rate. Another factor hindering efficient parallelism is that more invalid computations may be brought into the simulation when random events happen in the time window during computation. Recently, the wide use of scalable cluster systems, which achieve high performance at a low cost, has made parallel and distributed real-time transient stability analysis possible for large-scale power systems. The research on cluster-based parallel algorithms for transient stability simulations has become a new hot spot in the field [4], [11]–[13].

This paper presents a multilevel partition scheme for cluster systems based on power network regional characteristics. This partition scheme enhances the quality of the solutions and reduces the computation time of parallel transient stability analysis. The authors of this paper further propose an improved spatial parallel algorithm including a hierarchical block bordered diagonal form (BBDF) power network algorithm, which uses message-passing and shared-memory models simultaneously. In addition, some optimization schemes to reduce computation and communication time are also presented in this paper. The numerical results of three actual power networks show that the novel algorithms on a cluster system can run much faster than the real-time dynamic process, and simulations using these algorithms proved that they meet the requirement of online transient stability analysis for a future nationwide power grid in China.

Manuscript received March 8, 2004; revised July 22, 2005. This work was supported in part by the Intel Advanced Research Project, in part by the National Natural Science Foundation of China under Grant 60473101 and Grant 60433040, and in part by the National Key Basic Research and Development (973) Program of China under Grant 2004CB318205. Paper no. TPWRS-00119-2004.

The authors are with the Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China (e-mail: xuewei@tsinghua.edu.cn).

Digital Object Identifier 10.1109/TPWRS.2005.857266

II. Computing Model for Power System Transient Stability Analysis

To accomplish the task of power system transient stability computation, a set of differential algebraic equations (DAEs) have to be solved

$$\begin{aligned}\dot{\mathbf{X}} &= f(\mathbf{X},\mathbf{V}) = \mathbf{A}\mathbf{X} + \mathbf{B}u(\mathbf{X},\mathbf{V}) \\ 0 &= \mathbf{I} - Y(\mathbf{X})*\mathbf{V}. \end{aligned} \tag{1}$$

In (1), the first nonlinear differential equation group describes the dynamic characteristics of the power devices, and the second nonlinear equation group represents the restriction of the power network, where $\boldsymbol{X}$ is the state vector of individual dynamic devices, $\boldsymbol{I}$ is the vector of current injected from the devices into the network, $\boldsymbol{V}$ is the node voltage vector, $Y(\boldsymbol{X})$ is the complex sparse matrix, which is not constant with time, and u is the function of $\boldsymbol{X}$ and $\boldsymbol{V}$.

The most commonly used sequential algorithm for transient stability analysis is the interlaced alternating implicit approach (IAI) algorithm. The IAI algorithm uses a trapezoidal rule in the integration and solves differential equations and algebraic equations alternately and iteratively. It not only maintains the advantages of the implicit integration approach but also has modeling and computing flexibility.

At present, the time consumed for transient stability analysis increases super-linearly as the power system's size increases. The performance of sequential transient stability simulation is not adequate for real-time simulation of large-scale power grids. Its limitations become increasingly serious as the nationwide power grid continues to develop. Therefore, it is very important to study practical parallel algorithms and software.

III. Parallel Algorithm for Transient Stability Analysis

As is well known, a successful parallel algorithm has to accord with the characteristics of a parallel system. This brings more difficulties in the design and implementation of parallel algorithms. In addressing the high communication cost problem of cluster systems, the coarse granularity IAI algorithm proposed in [6] and [9] is regarded as the most effective of known parallel transient stability algorithms. In this paper, the complete solution to parallel transient stability analysis is studied, which involves an improved parallel transient stability algorithm based on the IAI algorithm. This algorithm makes a contribution to solving the bottleneck of the partition scheme, performs more optimizations in implementation on cluster systems, and achieves higher performance than the algorithm in [6] and [9]. Meanwhile, a new task-scheduling scheme is presented to address the conflict between load balancing and communication reduction and to ensure the validity of parallel computing.

A. Novel Multilevel Partition Scheme For Parallel Transient Stability Computation

The strategy for dispatching computation tasks to processors in the parallel transient stability algorithm focuses on how to separate the power network and devices into several subareas. To solve this problem, the following principles should be considered simultaneously:

- minimization of the connections between different subareas;
- maintaining an equal computational load for every subarea;

Since each power device's computation is independent from the other devices, the most difficult part of task scheduling is the power network decomposition.

On one hand, power systems are large-scale networks with connections related to geographical positions. The problem of network decomposition can be changed into a graph partition problem, which is well developed in the HPC field [14]. However, if a graph partition algorithm is simply used, more communication time will be introduced into the parallel simulation [15]. Meanwhile, more time will be consumed in the partition phase. On the other hand, because power systems are developed with connections of regional networks, the connections in the regional network are much tighter than those between regions. So partitioning based on this regional characteristic of power networks can reduce the communication cost in parallel simulations, but it may introduce more load imbalance [15].

This paper presents a multilevel network partition scheme to integrate the advantages of graph partition algorithms and the partition method based on power network regional characteristics. There are three phases in this new scheme: the coarsing phase, the multilevel partitioning phase, and the refining phase.

During the coarsing phase, the nodes in the same province or region are aggregated to successively decrease the size of the power network graph. The partition problem of power networks is formulated into a weighted graph partition problem, in which the weights of the vertex and edge represent the computation load of the subnetwork in the vertex and the amount of communication between subnetworks. Following network coarsing, the multilevel partition algorithm can identify the weak connections in a power network easily and effectively. The time consumed in the partition process is also reduced sharply.

In the multilevel partitioning phase, the derived small weighted graph is broken down into a specific number of subgraphs with a graph partition algorithm. The small derived graph limits the freedom in partitioning and may bring more load imbalance into the final solution. The multilevel scheme is, therefore, proposed to evaluate the quality of partition results, decompose the subarea with the maximum computation load, and partition the new graph again. This process continues to run until the optimal solution is found. The objective function used was

$$\min F(p) = \mathrm{Max}_{i=1,\ldots,p}(\mathrm{CompCost}_i) + \mathrm{CompCost}_B \tag{2}$$

in which $\mathrm{CompCost}_i (i = 1,\ldots,p)$ is the computation of the subarea i, and $\mathrm{CompCost}_B$ is the computation of the boundary system described below. In the evaluation function, the sum of $\mathrm{CompCost}_i$ and $\mathrm{CompCost}_B$ represents the total computation in the critical path and also the overall time consumed in the parallel simulation. It is noted that the more communication occurs between subtasks, the more computation is introduced into the boundary system. So the influence of communication has been taken into account by $\mathrm{CompCost}_B$.

After the weighted graph partition, the results have to be applied to the original network. Further refinement and adjustment are performed to improve the quality of partition results.

The flow chart of the proposed multilevel partition scheme based on the regional characteristic of power networks is shown below.

Step 1) *Establish the multilevel model of the power network*
Establish the power network model at the grid level, province level, region level, station level, and so on.

Step 2) *Network coarsing*
2.1 Decide the reasonable level of network coarsing with partition number.
2.2 Reduce the network.
2.3 Compute the vertex weights and edge weights of the derived graph.

Step 3) *Graph partitioning*
3.1 Perform graph partitioning with the multilevel recursive bisection algorithm.
3.2 Evaluate the results of partitioning. If the result of the objective function is bigger than that of the previous partition, terminate the successive partition process and the previous result is the final one. Then go to Step 4); otherwise, go to Step 3.3).
3.3 Split the subarea with the maximum computation load, and go to Step 3.1).

Step 4) *Refine results*
4.1 Identify the isolated node sets by topo analysis.
4.2 Analyze the connections of the nodes in the boundary system.
4.2 Refine the network partition.

B. New Hierarchical BBDF Algorithm for Power Network Computation

According to the power system's characteristics, the corresponding differential equations for dynamic devices such as generators are only related to one of the network nodes. The differential part of (1) can be put into connected subareas and can be divided into the corresponding processors for simultaneous computation. Therefore, the parallel computation of linear power network equations is the key to the transient process simulation. Furthermore, new events occurring during the simulation result in a rapid increase in the computation of transient stability problems, including adjusting the admittance matrix, rebuilding factors, and renewing the node voltages.

Based on cluster systems, the network equations are reformed in the BBDF [16], as shown in (3). The following equations are for two subareas as previously described:

$$\begin{bmatrix} \mathbf{Y} & \mathbf{M}' \\ \mathbf{M} & \mathbf{Z} \end{bmatrix} \begin{bmatrix} \mathbf{U} \\ \mathbf{I} \end{bmatrix} = \begin{bmatrix} \mathbf{I}_p \\ 0 \end{bmatrix} \tag{3}$$

in which we have the equation shown at the bottom of the page, where the subscripts 1 and 2 represent the subarea number, and the subscripts p, n, and z represent the positive, negative, and zero sequence networks, respectively. $\mathbf{Y}_{1p}$, $\mathbf{Y}_{2p}$, $\mathbf{Y}_{1n}$, $\mathbf{Y}_{2n}$, $\mathbf{Y}_{1z}$, and $\mathbf{Y}_{2z}$ are the admittance matrices of three (positive, negative, and zero) sequence networks, respectively. $\boldsymbol{Z}_{CF}$ is the impedance matrix for cutting branches (the branches between different subareas) and fault branches and also the coefficient matrix of boundary equations in the BBDF computation. $\boldsymbol{M}$ and $\boldsymbol{M}$' are the associated matrices between $\boldsymbol{Y}$ and $\boldsymbol{Z}$, respectively.

According to the parallel scheme used in BBDF equations, the factor hindering parallelism is the solution of the boundary system, which is the sequential part in the whole parallel algorithm. With the increase of subareas, the time required to complete the boundary equations and the time spent on communication between processors increase sharply. This paper presents a hierarchical BBDF power network algorithm to enhance the computation efficiency of boundary equations. This algorithm introduces the BBDF parallel scheme into boundary equations recursively. The coefficient matrix of the boundary system is reordered by positive, negative, and zero sequence parts of cutting branches before fault branches, as described in

$$\begin{bmatrix} \mathbf{Y}_{Tp} & & & \mathbf{N}_{Tp} & & \\ & \mathbf{Y}_{Tn} & & & \mathbf{N}_{Tn} & \\ & & \mathbf{Y}_{Tz} & & & \mathbf{N}_{Tz} \\ \mathbf{N}_{pT} & & & \mathbf{Y}_F & & \\ & \mathbf{N}_{nT} & & & & \\ & & \mathbf{N}_{zT} & & & \end{bmatrix} \tag{4}$$

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}_{CF-1p} & \mathbf{M}_{CF-2p} & & & & \\ & & \mathbf{M}_{CF-1n} & \mathbf{M}_{CF-2n} & & \\ & & & & \mathbf{M}_{CF-1z} & \mathbf{M}_{CF-2z} \end{bmatrix}$$

$$\boldsymbol{Y} = \begin{bmatrix} \boldsymbol{Y}_{1p} & & & & & \\ & \boldsymbol{Y}_{2p} & & & & \\ & & \boldsymbol{Y}_{1n} & & & \\ & & & \boldsymbol{Y}_{2n} & & \\ & & & & \boldsymbol{Y}_{1z} & \\ & & & & & \boldsymbol{Y}_{2z} \end{bmatrix}, \quad \mathbf{U} = \begin{bmatrix} \mathbf{U}_{1p} \\ \mathbf{U}_{2p} \\ \mathbf{U}_{1n} \\ \mathbf{U}_{2n} \\ \mathbf{U}_{1z} \\ \mathbf{U}_{2z} \end{bmatrix}$$

$$\boldsymbol{I} = \begin{bmatrix} \boldsymbol{I}_{1p} \\ \boldsymbol{I}_{2p} \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \boldsymbol{M}' = \begin{bmatrix} \boldsymbol{M}_{1p-CF} & & & \\ \boldsymbol{M}_{2p-CF} & & & \\ & \boldsymbol{M}_{1n-CF} & & \\ & \boldsymbol{M}_{2n-CF} & & \\ & & \boldsymbol{M}_{1z-CF} & \\ & & \boldsymbol{M}_{2z-CF} & \end{bmatrix}$$

$$\mathbf{Z} = Z_{CF}, \quad I = I_{CF}$$

in which $\mathbf{Y}_{Tp}$, $\mathbf{Y}_{Tn}$, and $\mathbf{Y}_{Tz}$ are the three sequence node admittance matrices for cutting branches, $\mathbf{Y}_F$ is the admittance matrix for fault nodes, and $\boldsymbol{N}_{Tp}$, $\mathbf{N}_{Tn}$, $\mathbf{N}_{Tz}$, $\mathbf{N}_{pT}$, $\mathbf{N}_{nT}$, and $\mathbf{N}_{zT}$ are the associated matrices. Because some fault forms have invalid impedance matrices, an admittance matrix is adopted in (4).

Boundary equations are much smaller than network equations. Most of their computations focus on reforming the boundary equations and factoring the coefficient matrix when events occur. A dynamic multithread scheme is used to solve the boundary equations with nonsymmetric faults, which is more effective for clusters that consist of multiprocessor machines (SMP-Cluster). When this hierarchical network equation algorithm is applied to transient stability problems, it not only dramatically improves the efficiency and gains but also enhances the scalability of the program. At the same time, the boundary system computation scheme provides two other benefits. One is computation time reduction based on the symmetry of negative and zero parts in the boundary system, and the other is fewer conditions in the boundary matrix. This improves the precision and robustness of the algorithm.

C. Parallel Algorithm for Transient Stability Analysis

In the simulation, a large system represented by (1) is broken into N subsystems based on the partition scheme. N subsystems are, respectively, assigned to N processors for computation. For example, the kth subsystem is processed by the processor P_k $(k = 1, 2, \ldots, N)$. Each subsystem is calculated independently with the solutions of the boundary system. Then the to the boundary equations comprises the computation results for each subsystem. This process is repeated until convergence is reached, as described in the following flow chart.

Initialization
For TimeStep = 1,..., MaxTimeStep (Simulaton Loop)
For Iter = 0,..., MaxIters (Simulation Loop)
Parallel Simulation by Processor $P_1 \ldots P_N$ [e.g., in P_k]
Include new events in subsystem k
Solve the differential equations of subsystem k with the trapezoidal rule

$$\dot{X}_k = f(X_k, V_k) = AX_k + Bu(X_k, V_k).$$

Compute the current injected into subsystem k

$$I_k(X_k, V_k).$$

Check local convergence of subsystem k

$$\left\| I_{kt} - I_{k(t-1)} \right\| < \varepsilon_I, t - \text{iteration number}.$$

Solve the vector and matrix corrected in subsystem k

$$\begin{cases} \Delta Y_{km} = \left(M_{CF-km} Y_{km}^{-1} M_{km-CF} \right)^{-1}, & m = p, n, z \\ \Delta I_{kp} = \Delta Y_{kp} M_{CF-kp} Y_{kp}^{-1} I_{kp} \end{cases}$$

Communication of collection between processors
Include new events in the boundary system
Global convergence checking
If convergence is reached, then break the iteration loop
Solve the boundary system (positive sequence related part)

$$\begin{bmatrix} Y_{Tp} & & N_{Tp} & & \\ & Y_{Tn} & & N_{Tn} & \\ & & Y_{Tz} & & N_{Tz} \\ N_{pT} & & & & \\ & N_{nT} & & Y_F & \\ & & N_{zT} & & \end{bmatrix} \begin{bmatrix} U_{Cp} \\ U_{Cn} \\ U_{Cz} \\ U_{Fp} \\ U_{Fn} \\ U_{Fz} \end{bmatrix} = \begin{bmatrix} I_{Tp} \\ \\ I_{Fp} \\ \\ \end{bmatrix} + \sum_{m=1}^{k} \Delta I_{kp}$$

Scattering the positive sequence solution
Check partial convergence in subsystem k

$$\| I_{km-CF,t1} - I_{km-CF,t+1} \| < \varepsilon_I, t - \text{iteration number}$$

Solve the node voltages in subsystem k

$$U_{kp} = Y_{kp}^{-1} \begin{pmatrix} I_{kp} - M_{kp-CF} \Delta Y_{kp}^{\times} \\ \left(M_{CF-kp} Y_{kp}^{-1} I_{kp} - U_{kp-CF} \right) \end{pmatrix}.$$

End Parallel Simulation
End For Iter (End Simulation Loop)
Compute computation results of subsystem k
Put the output information into the real-time database
Get new events from the outer system
End For TimeStep (End Simulation Loop)
End Simulation

The convergence checking scheme on parallel architecture is more complex than that on sequential systems because it has to take into account the coordination of the computing processes. We employed a new global convergence checking scheme to improve the computation efficiency, as follows.

1) Currents injected are regarded as variables to check convergence.
2) In every subsystem, convergence is checked locally. If local convergence occurs, the corresponding process informs the control process (the same as the process for computing the boundary system) with a local convergence flag instead of the new corrected vector. Then this subsystem waits for the global convergence flag or the new solutions of the boundary equations. If global convergence is reached, the simulation enters into the next time step. Otherwise, whether the local computation

is performed or not depends on the partial convergence checking.

3) The control process collects the local convergence flags from all subsystems. After the global convergence is checked, the results are sent to each subsystem. If global convergence is not reached, the boundary equations have to be solved once more.

In addition, some optimization schemes were developed to reduce the time spent on computation and communication and to improve the scalability of the program presented in this paper.

1) There are two iteration loops in the IAI algorithm. The inner one to solve nonlinear network equations is broken in the algorithm in order to cancel invalid substitutions. Numerical tests proved that the results obtained using our algorithm are as accurate as those of the traditional IAI algorithm.
2) During the simulation, state variables of dynamic devices, injected currents, and node voltages of the positive sequence network have to be updated in each iteration. Nevertheless, it is not necessary to compute the negative or zero sequence network equations in subsystems and boundary system. The electric variables of negative or zero sequences in subsystems and boundary equations will be solved once in every time step only when output is required.
3) Different schemes for computing faults are used for different fault forms, such as canceling the zero sequence network computation when inner-phase faults occur only.
4) Optimal node ordering, sparsity computation techniques for network solutions, and detailed code tuning are used to further improve the computation performance.

IV. Test Results

In this paper, three power systems shown in Table I were tested on an SMP Cluster. In the cluster, each node was a symmetric multiprocessor (SMP) computer and had four Intel Xeon PIII700 MHz CPUs with 36-GB hard disks and 1 GB of memory. The communication medium between SMP nodes was Myrinet with a bandwidth of 2.56 Gb/s. The software environments were Redhat Linux 7.2 (kernel version 2.4.7–10 smp), MPICH-1.2.1…7, and gm-1.5pre4. Gm was the driver of Myrinet.

For all test cases, five order generator model, typical exciter and governor model, and induction motor model were used [17]. An A-phase fault on a single 220-kV branch was assumed for each case. The fault occurred at 0 s, and the branch tripped at 0.16 s. In the simulations, the fixed time step 0.02 s was used, and the simulation time was 0.3 s, where the heaviest computational load was located. The convergence tolerance was 10^{-4} p.u.

The partition results using the multilevel scheme are listed in Table II.

It should be noted that the same compiler flags and code tuning schemes were used for all the test programs in this paper. In the figures below, *Sp* stands for speedup, which is the ratio of the time required for parallel simulation with partitions to the time required for sequential simulation without partitions. *SV* stands for the simulation velocity, which is the ratio of the actual running time of the power grid transient process to the simulation time on the cluster system for the same power grid case. The efficiency is expressed as $E = Sp/P$, where P is the number of CPUs.

TABLE I
Network Information for Three Power Systems

System		Case 1	Case 2	Case 3
Scale	number of nodes	706	1103	2115
	number of branches	1069	1622	2614
	number of generators	88	183	248
	number of loads	459	695	544

TABLE II
Results of the Partition Scheme for Three Power Systems

Network	Partition number	Max/Min[a]	CutBrn[b]
Case 1	2	422/284	4
	4	301/121	7
	5	169/121	12
	6	149/75	17
	8	119/68	34
Case 2	2	656/447	1
	4	387/215	19
	5	268/182	18
	6	232/118	29
	8	217/65	43
Case 3	2	1088/1027	2
	4	611/477	22
	5	641/310	20
	6	401/323	31
	8	328/199	40
	10	258/172	47
	12	230/123	57

[a] Max/Min: the ratio of the number of nodes in the maximal partition to the number in the minimal partition.

[b] CutBrn: the number of branches between sub-areas.

A. Validity of the Algorithm Proposed in This Paper

For Case 1, Fig. 1 shows the maximum deviation of node voltage with the parallel program presented in this paper and the software named PSASP. PSASP is the standard sequential software package developed by EPRI China, which is widely used for power system simulation in China [18].

According to Fig. 1, the maximum deviation of node voltage between our algorithm and PSASP was less than 10^{-5} p.u. This proves that our algorithm is accurate and feasible.

B. Computation Performance of Our Algorithm

Figs. 2–4 show the speedups and simulation velocity of the parallel computing for Cases 1–3. Figs. 5 and 6 show the efficiency (E) and the simulation time (T) of parallel transient simulations of the three power networks (Cases 1–3).

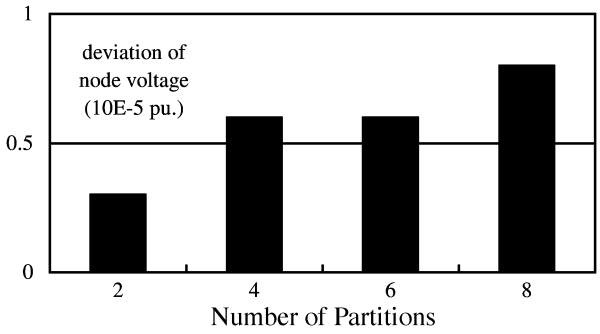
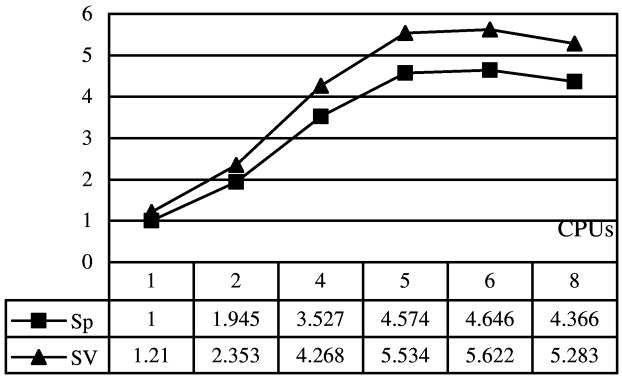
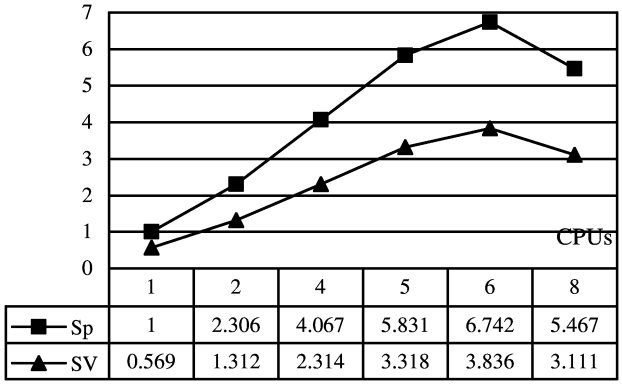


Fig. 1. Maximal deviation of node voltage using our algorithm and PSASP (PSASP is regarded as the standard software).



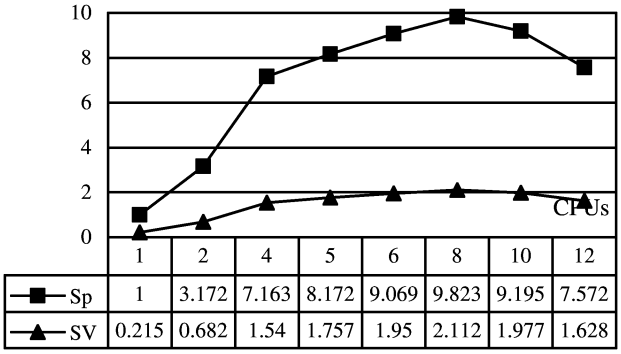
	1	2	4	5	6	8
Sp	1	1.945	3.527	4.574	4.646	4.366
SV	1.21	2.353	4.268	5.534	5.622	5.283

Fig. 2. Speedups and simulation velocity of Case 1.



	1	2	4	5	6	8
Sp	1	2.306	4.067	5.831	6.742	5.467
SV	0.569	1.312	2.314	3.318	3.836	3.111

Fig. 3. Speedups and simulation velocity of Case 2.



	1	2	4	5	6	8	10	12
Sp	1	3.172	7.163	8.172	9.069	9.823	9.195	7.572
SV	0.215	0.682	1.54	1.757	1.95	2.112	1.977	1.628

Fig. 4. Speedups and simulation velocity of Case 3.

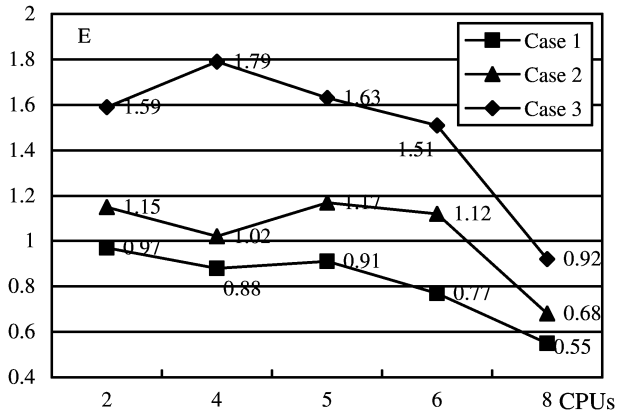


Fig. 5. Efficiency of power grid of Cases 1–3.

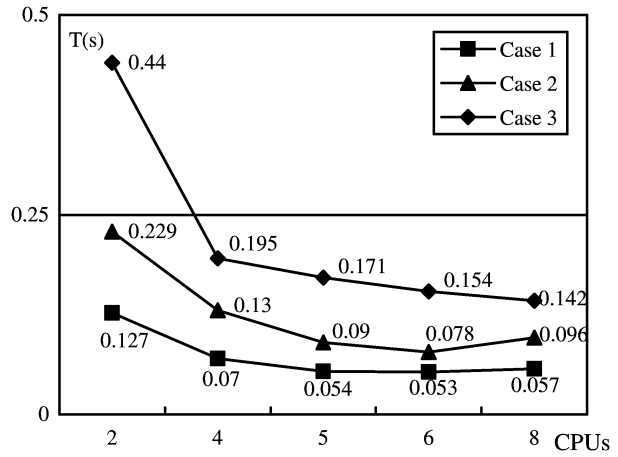


Fig. 6. Parallel simulation time for Cases 1–3.

Comparing the results of the three cases shown in these figures, the following conclusions can be drawn.

- The high speedups and parallel efficiency are achieved in three actual power networks, and some super-linear speedups are even achieved. These results suggest that the parallel algorithm proposed in this paper is efficient and practical and can be used for real-time power system simulations.
- The transient process simulations run faster than the real-time process for these three large-scale power systems. For example, when a single CPU was used in Case 1, the transient simulation took a little less time than the actual transient process, which meets the initial requirement of real-time simulation. But for online use, simulations have to take much less time and leave enough time for prediction, control, and event handling. When six CPUs were used in Case 1, the simulation time was only 18% of the actual transient process time and a super real-time simulation was achieved. This was almost four times faster than that on a single CPU. For Cases 2 and 3, simulations carried out with a single CPU were not adequate for the requirement of real-time simulation. So parallel processing was necessary to improve the simulation velocity. When six CPUs were used for parallel computation in Case 2, the simulation velocity reached 3.836, only 26% of the actual transient process time, and about six times faster

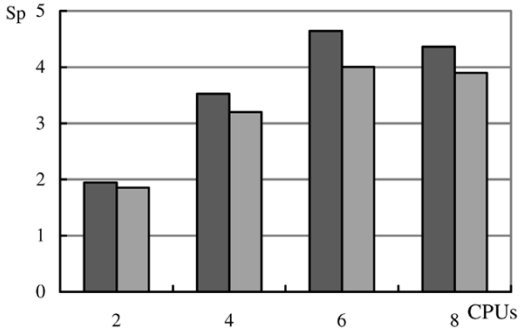


Fig. 7. Performance comparison using the hierarchical BBDF algorithm and the traditional algorithm (left for our algorithm).

than that on a single CPU. A super real-time simulation was also achieved. As shown in Fig. 4, the parallel simulation for the largest power network (Case 3) on eight CPUs was almost nine times faster than the sequential simulation and twice as fast as the real-time process. These results prove that the partition scheme and the hierarchical algorithm for parallel transient stability analysis proposed in this paper are very efficient.

- The speedups of parallel simulations become saturated with the increase of subareas and computing processors, reaching a maximum between 5 and 10. For Case 1, six processors were being used when the speedup and the simulation velocity reached the maximum. There were also six processors in use when the speedup and simulation velocity for Case 2 reached the maximum. For Case 3, eight processors were in use when the speedup and simulation velocity reached the maximum. From Case 1 to Case 3, the optimal partition number and speedup increased with the increase of power system size. It should be noted that the algorithm presented in this paper is scalable, and simulations using this algorithm are able to achieve better efficiency and gains on larger power systems.

C. Performance Comparison Between Our Algorithm and the Traditional BBDF Algorithm on a Cluster

For Case 1, the performance comparison between our hierarchical BBDF algorithm and the algorithm proposed in [6] and [9] is shown in Fig. 7. When computing the speedups, the same sequential time was used in both algorithms. Meanwhile the same partitions and cluster system were used in these two programs.

As shown in Fig. 7, the hierarchical BBDF algorithm proposed in this paper can gain higher performance than the coarse granularity IAI algorithm [6], [9]. The more partitions used, the higher speedups are achieved. When six CPUs are used, the two algorithms both get optimal performance. It was also found that the performance advantage of our algorithm over the algorithm in [6] and [9] reaches the maximum with six CPUs. The maximum speedup of our algorithm was 4.65, which was about 16% higher than that of the algorithm proposed in [6] and [9]. This is due to reduced communication time and the optimization of boundary system computation in our algorithm. Therefore, it can be concluded that our algorithm is superior to the well-known coarse granularity algorithm.

TABLE III
COMPARISON OF THE MULTILEVEL SCHEME AND ALGORITHM IN METIS

Partition number	Multilevel scheme		METIS algorithm	
	Max/Min	CutBrn	Max/Min	CutBrn
2	422/284	4	379/327	12
4	301/121	7	207/156	28
8	119/68	34	145/74	54

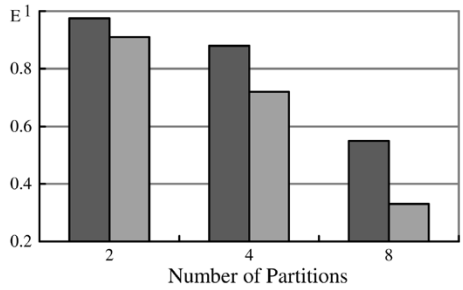


Fig. 8. Performance comparison using the multilevel partition algorithm and the graph partition algorithm in METIS (left for our algorithm).

D. Performance Comparison Between Our Multilevel Partition Scheme and METIS

Table III lists the partition results using the multilevel scheme and the recursive bisection method in METIS [19]. It should be noted that the different partition parameters have been tested for the METIS algorithm in these cases, and the best results are shown here. Furthermore, a comparison of the simulation performance for the proposed multilevel partition scheme and the algorithm in METIS is shown in Fig. 8. Our hierarchical spatial algorithm was used to simulate the transient process for two partition methods.

Table III shows that the multilevel partition scheme results in much fewer cutting edges than the recursive bisection algorithm in METIS but leads to a little more imbalance between subareas. The results in Fig. 8 suggest that the multilevel scheme achieves higher efficiency in transient stability simulations on cluster systems, and its performance does not noticeably suffer from the imbalance of subareas. Furthermore, higher efficiency is achieved with the increase of partition number. With eight CPUs, the efficiency of our algorithm was about 70% higher than that of METIS. Therefore, it can be concluded that the multilevel partition scheme fits our cluster-based spatial algorithm well, especially when more CPUs are used. The multilevel partition scheme and the hierarchical algorithm give an integrated solution to parallel transient stability analysis and can provide very satisfying results.

E. Analysis of Super-Linear Speedup Phenomenon in Test Cases

The efficiency of this parallel algorithm exceeded 100% in some test cases. According to the analysis of our algorithm, the communication and computation time required for solving boundary equations increase with the increase of processors,

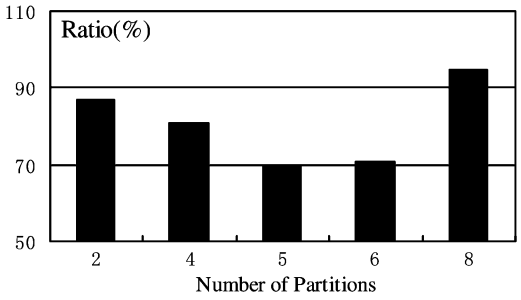


Fig. 9. Ratios of the sequential computation time to the computational time of the new algorithm with partitions running on a single processor.

while the computation time required for dealing with new events, which includes reforming and factoring the admittance matrix, decreases with the increase of partition number. Meanwhile, the partial convergence checking mechanism cancels some redundant iteration cycles of subareas and greatly reduces the total computation of parallel simulations. So the total computation required by this algorithm may decrease in some cases. This can be proven with the ratios of the computational time of the new algorithm with partitions running on a single processor to the sequential computation in Fig. 9 for Case 2. This is one important cause of the abnormally high speedup and dramatically improved efficiency.

Another cause is the dynamic multithread scheme used in this algorithm. The threads share some of the computation and reduce the amount of communication between computing processes. At the same time, the so-called "cache effect" is also considered as a potential reason for these abnormal performance results.

V. Conclusion

This paper proposes a multilevel partition scheme based on power network regional characteristics and a hierarchical BBDF power network algorithm. The algorithm uses message-passing and shared-memory models simultaneously. Optimization schemes such as convergence checking are also presented to improve the simulation efficiency. The multilevel partition scheme and the hierarchical spatial algorithm give an integrated solution to parallel transient stability analysis and are implemented on a cluster system. Simulations were performed for three large-scale power systems in China. The numerical results suggest that the algorithms and optimization schemes are efficient and scalable. Compared to some other spatial parallel algorithms for transient stability analysis [6], [20], this algorithm shows remarkably improved performance. This algorithm, with adequate efficiency and scalability, is a feasible selection for the real-time transient simulation of China's future nationwide power grid.

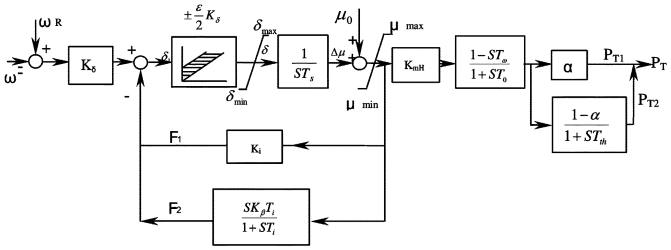
Appendix

For all test cases in this paper, a five order generator model, typical exciter and governor model, and induction motor model were used and are listed below.

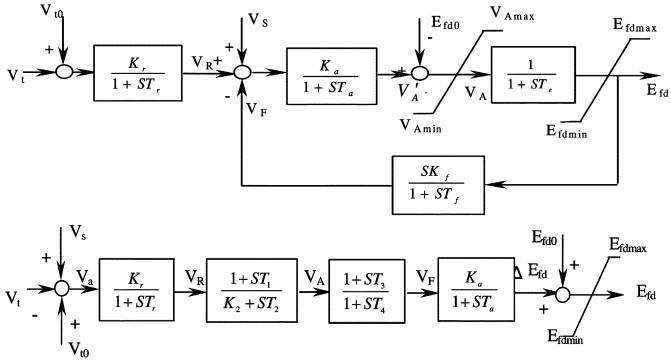
5 order generator model:

$$\begin{aligned}
T'_{d0}\frac{dE'_q}{dt} &= E_{fd} - (k_G - 1)E'^{(S_n)}_q - \frac{x_d}{x'_d}E'_q \\
&\quad + \frac{x_d - x'_d}{x'_d}\left(V_x\cos\delta + V_y\sin\delta\right) \\
T''_{q0}\frac{dE''_d}{dt} &= -E''_d + \left(x_q - x''_q\right)I_q \\
T''_{d0}\frac{dE''_q}{dt} &= -E''_q - \left(x'_d - x''_d\right)I_d + E'_q + T''_{d0}\frac{dE'_q}{dt} \\
T_J\frac{d\omega}{dt} &= \frac{P_T}{\omega} - \frac{P_e}{\omega} - D\left(\omega - \bar{\omega}_0\right)2\pi f_0 \\
\frac{d\delta}{dt} &= (\omega - 1)\,2\pi f_0.
\end{aligned}$$

The block diagram of governor model:



The block diagrams of exciter models:



The induction motor model:

$$\begin{aligned}
T'_{do}\frac{d\dot{e}'_M}{dt} &= -\dot{e}'_M - jk_Z\left(x - x'\right)\dot{I}_M - jT'_{d0}\dot{e}'_M S2\pi f_0 \\
T_J\frac{ds}{dt} &= M_M - M_E
\end{aligned}$$

in which $M_M = k_L\lceil\alpha + (1-\alpha)(1-S)^P\rceil$

$$M_E = -\,\mathrm{Re}\left(\dot{e}'_M\hat{\dot{I}}_M\right)$$

$$k_L = \frac{P^*_{t(0)}}{\alpha + (1-\alpha)(1-S_{(0)})^P}.$$

The definitions of parameters in the above formulas and diagrams are the same as those in EPRI China's book *Power System Analysis Software Package, Fundamental Database User Manual* published in 2001. PSASP is the standard se-

quential software package developed by EPRI China, which is widely used for power system simulation in China [17], [18].

ACKNOWLEDGMENT

The authors would like to thank the IEEE TRANSACTIONS ON POWER SYSTEMS reviewers for their valuable comments, which improved the quality of this paper.

REFERENCES

[1] D. J. Tylavsky and A. J. Bose, "Parallel processing in power systems computation," *IEEE Trans. Power Syst.*, vol. 7, no. 2, pp. 629–638, May 1992.

[2] J. S. Chai and A. J. Bose, "Bottlenecks in parallel algorithms for power system stability analysis," *IEEE Trans. Power Syst.*, vol. 8, no. 1, pp. 9–15, Feb. 1993.

[3] G. Kron, *Diakoptics—The Piecewise Solution of Large-Scale Systems*. London, U.K.: MacDonald, 1963.

[4] W. Xue, J. W. Shu, X. F. Wang, and W. M. Zheng, "Advance of parallel algorithm for power system transient stability simulation," *J. Syst. Simulat.*, vol. 14, no. 2, pp. 177–182, 2002.

[5] IEEE PES Power Systems Engineering Committee, "Parallel processing in power systems computation," *IEEE Trans. Power Syst.*, vol. 7, no. 2, pp. 629–638, May 1992.

[6] I. C. Decker, D. M. Falcao, and E. Kaszkurewicz, "Conjugate gradient methods for power system dynamic simulation on parallel computers," *IEEE Trans. Power Syst.*, vol. 11, no. 3, pp. 1218–1227, Aug. 1996.

[7] M. La Scala, G. Sblendorio, A. Bose, and J. Q. Wu, "Comparison of algorithms for transient stability simulations on shared and distributed memory multiprocessors," *IEEE Trans. Power Syst.*, vol. 11, no. 4, pp. 2045–2050, Nov. 1996.

[8] G. Aloisio, M. A. Bochicchio, M. La Scala, and R. Sbrizzai, "A distributed computing approach for real-time transient stability analysis," *IEEE Trans. Power Syst.*, vol. 12, no. 2, pp. 981–987, May 1997.

[9] K. W. Chan, R. C. Dai, and C. H. Cheung, "A coarse grain parallel solution method for solving large set of power systems network equations," in *Proc. Int. Conf. Power System Technology*, vol. 4, 2002, pp. 2640–2644.

[10] A. H. Jorge and R. M. Jose, "Real time network simulation with PC-cluster," *IEEE Trans. Power Syst.*, vol. 18, no. 2, pp. 563–569, May 2003.

[11] Y. L. Li, X. X. Zhou, and Z. X. Wu, "Parallel algorithms for transient stability simulation on PC cluster," in *Proc. PowerCon*, vol. 3, 2002, pp. 1592–1596.

[12] ——, "A parallel complex fault computation algorithm for large-scale power system digital simulation," *Proc. CSEE*, vol. 23, no. 12, pp. 1–5, 2003.

[13] W. Xue, J. W. Shu, J. F. Yan, and X. F. Wang, "A parallel implementation for real-time transient stability simulation of large-scale power system," in *Proc. 7th IASTED Int. Multiconf. Power Energy Systems*, K. M. Smedley, Ed., Palm Springs, CA, Feb. 24–26, 2003, pp. 133–137.

[14] K. Schloegel, G. karypis, and V. Kumar, "Graph partitioning for high performance scientific simulations," in *CRPC Parallel Computing Handbook*. San Mateo, CA: Morgan Kaufmann, 2000.

[15] J. Shu, W. Xue, and W. Zheng, "An optimal partition scheme of transient stable parallel computing in power system," *Automat. Elect. Power Syst.*, vol. 27, no. 19, pp. 6–10, 2003.

[16] A. Torralba, "Three methods for the parallel solution of a large, sparse system of linear equations by multiprocessors," *Int. J. Energy Syst.*, vol. 12, no. 1, pp. 1–5, 1992.

[17] "Power System Analysis Software Package," in *Fundamental Database User Manual*: EPRI China, 2001.

[18] C. Hua and X. Zheng, "Comparison of mathematical models for transient stability calculation in PSASP and PSS/E and corresponding calculation results," *Power Syst. Technol.*, vol. 28, no. 5, pp. 1–4, 2004.

[19] K. George and V. P. Kumar. (1998) METIS-a Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices, Version 4.0[EB/OL]. [Online]. Available: http:// www.cs.umn.edu/~karypis.

[20] M. Nagata and N. Uchida, "Parallel processing of network calculations in order to speed up transient stability analysis," *Elect. Eng. Jpn.*, vol. 135, no. 3, pp. 26–36, 2001.

Jiwu Shu was born in 1968. He received the master's degree from the Department of Computer Science, National Defense University of Science and Technology, Changsha, China, in 1991 and the Ph.D. degree from the Department of Computer Science, Nanjing University, Nanjing, China, in 1998.

In 2000, he finished his postdoctoral research at Tsinghua University, Beijing, China. Since 2000, he has been teaching at Tsinghua University, where he is now an Associate Professor at the Institute of High Performance Computing Technology, Department of Computer Science and Technology. His current research interests include storage area networks, parallel and distributed computing and networking, algorithm analysis and design and parallel processing techniques, and cluster systems and communication.

Wei Xue was born in Beijing in 1974. He received the Ph.D. degree in electrical engineering from Tsinghua University, Beijing, China, in 2003.

In 2003, he joined the faculty of the Department of Computer Science and Technology, Tsinghua University. His research interest includes power system analysis and simulation, parallel algorithm design, cluster computing, and network storage.

Weimin Zheng was born in 1946. He received the Master's degree from Tsinghua University, Beijing, China, in 1982.

He is the Director of the Institute of High Performance Computing Technology, Department of Computer Science and Technology, Tsinghua University. Since 1982, he has been working at Tsinghua University in the area of parallel and distributed processing. His research covers parallel computer architecture, parallel and distributed computing, AI-oriented computer architecture, compiler techniques and run-time system design for parallel processing systems and grid computing, and network storage.