The Journal of CPRI,
Vol. 7, No. 2, September 2011 pp. 145–152

Parallel Contingency Analysis for Power Systems Operation and Planning

Ezhilarasi G A* and Swarup K S*

This paper proposes a parallel processing approach to Contingency Analysis (CA) for power system security evaluation. Full AC power flow analysis is done for each contingency and violations of bus voltages and line flows are evaluated. All possible line outages are considered to avoid overlooking of certain critical cases. Parallel Processing is employed to increase the speed of execution. Parallelism is achieved by sharing each contingency across processors. The proposed methodology is implemented in a Linux Cluster. Data communication is performed through the Message Passing Interface (MPI). The effectiveness of parallelism is demonstrated by performing contingency analysis on standard IEEE Systems. The performance of the algorithm is analyzed in terms of computation speed and efficiency in comparison with the sequential approach.

1.0 INTRODUCTION

Power system is a large interconnected system conceptually studied based on two main functions namely economics and security. Power system functions include economic dispatch, optimal power flow and automatic generation control and may others included in the energy management system. Power system security analysis encompasses three functions namely system monitoring, contingency analysis and corrective control. Power system monitoring is realized exclusively through the SCADA system at the control centers. Contingency Analysis involves two functions namely contingency selection and contingency evaluation. The system operator determines the preventive or corrective control measures based on the alarms generated by the contingency analysis.

Several research works (C. P. Arnold, M. I. Parr and M. B. Dewe 1983) [1] propose the application of both artificial intelligence and probabilistic methodologies to obtain less accurate but fast solutions of the security assessment problem by identifying, the set of the "most credible contingencies" in a preliminary phase called the contingency screening (Qiang, Jun Wu and Anjan Bose 1995) [2]. The application of these methodologies reduces the complexity of the security assessment problem, and consequently the required computational efforts, at the cost of accepting a fixed level of risk. On the other hand, considerable research efforts have also been done to develop dedicated computer architectures based on supercomputers or network of workstations for the fast solution of the power system state equations (C. P Arnold *et al*). These platforms are designed to meet the computational requirements of security analysis for a reference electrical network given the maximum number of contingencies that should be analyzed, the interval time width of the dynamic simulations, the maximum computation time, etc. (Wu and Bose) [3].

The constant growth of the electrical network's complexity and the need of system operators to continuously improve the plant infra-structures exploitation raising the reliability

*Department of Electrical Engineering, Indian Institute of Technology Madras, Chennai – 600036.
E-mail: angel.ezhil@gmail.com, swarup@ee.iitm.ac.in

levels by more detailed and timely control actions, and more advanced protective units, make these values subject to large variations. Consequently, the computational efforts required by online security analysis may rise dynamically and so more scalable processing systems are required. Hence this paper presents a parallel processing approach to contingency analysis that can be easily implemented with the advanced technology and communication protocols. This could be easily implemented in any parallel architecture with very less data communication.

2.0 PARALLEL PROCESSING APPROACH

The power system has a complex infrastructure. It is composed of highly interactive, non linear, dynamic entities that spread across vast areas. In any situation, including disturbances caused by natural disasters, high demands, centralized control require multiple, high-speed, two-way communication links, powerful central computing facilities and an elaborate operations control centers to give rapid response as shown in Figure 1.

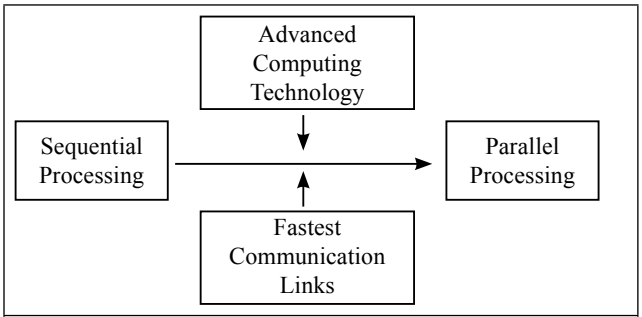

FIG. 1 PARALLEL PROCESSING TRANSFORMATION

A centralized control might not be practical if the remedial response is delayed. The lack of response can cripple the entire power system. Under these conditions an effective means to monitor and control a complex power system is found to robustly operate with hierarchical and distributed layers. An effective approach in such cases is to have some way of intervening in the faulted system locally at places where disturbances have originated in order to stop the problem from propagating through the network. This is the essence of distributed processing.

Distributed approach can greatly enhance the reliability and improve the flexibility and efficiency of power system monitoring and control by using parallel processing for suitable applications in the local control centers. The Distributed Processing is a flexible means for the power system operators to manage the system efficiently within a limited period. The first step in implementing this is to divide the entire power system into a number of control areas, and then set up a lower level control center for each area. It has many advantages over the centralized control like higher efficiency, economy, enhanced reliability and flexibility, even though it increases the capital investment for communication system.

Parallel processing is applied in the local control centers to improve the speed of dynamic simulation so that one can look into the future to predict the system status. In power grid operations, contingency analysis assesses the effect of various combinations of power system component failures based on state estimates and the database is updated every five minutes only. This is not enough to predict the system status because a power grid could become unstable and collapse within seconds. Hence to speedup the complex algorithms and bring them to the reach of real time grid operations, high performance computing is essential.

The high performance computing environment consists of a 256 node Linux cluster with 8 processors in each node. This paper presents a simulation of contingency of large scale power system in a cluster with the parallel architecture as shown in Figure 2. It consists of n nodes with a head node for end user communication. Job scheduling is done in such a way that every process runs in an individual node. It is an inifiniband cluster; hence the communication delay is minimum or predictable.

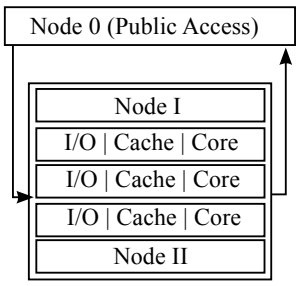

FIG. 2 ARCHITECTURE OF SIMULATION ENVIRONMENT

3.0 PARALLEL CONTINGENCY ANALYSIS

Contingency Analysis is an embarrassingly parallel application in power system. It assesses the effect of various combinations of power system component failures based on the state estimates and periodically updates the system database. Due to the effectual computations involved, contingency screening is done to select the credible contingencies and full AC load flow (Allen J Wood & B F Wollenberg) [4] is done on the selected cases to study the post contingency effects on the system. This risks overlooking certain vital contingencies that can cause a catastrophe to the power system. Therefore dominant analytical approach to sequential contingency analysis is required to detect potential contingencies with high efficiency and reliability.

The advantage of parallel processing approach is exploited for contingency analysis of the power system. Each contingency is completely independent of the other. The only data dependency is on the maximum power flow of all the lines, to calculate the performance index of the line. Only single line outages are considered here for simulation. Multiple line outages and generators outages would increases the number of processes involved and would be further suitable for parallel processing. The algorithm is shown by means of a flowchart in Figure 3.

Task allocation is simple as calculation involved in each contingency is independent of the other. A root process is created along with as many child processes as the number of contingencies in the root node. The root process takes care of the I/O operations. It reads the input data of the system

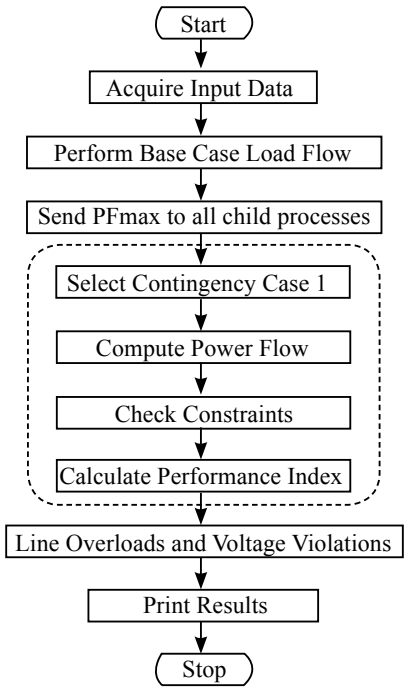

FIG. 3 FLOWCHART FOR PARALLEL CONTINGENCY ANALYSIS

under study from the database. As depicted, the root process performs the base case load flow and communicates to all the child processes through MPI directives. Single line outage is simulated in each child process and the post contingency analysis is done. Full AC load flow is done on each case and analysis is done to check the line overload, voltage limit violations and feasibility of the solution [5].

Performance index is sent back to the root process from all the child processes to ensure synchronization. The performance index of the line *l* is defined as

$$\mathrm{PI}_l = \sum_{\mathrm{i}=1}^{\mathrm{n}l} \frac{\mathrm{pflow}_\mathrm{i}}{\mathrm{pmax}_l} \qquad (1)$$

where,

nl : number of line

pflow *i* : power flow of that line *i*

pmax *i* : power flow limit of line *l*

4.0 IMPLEMENTATION METHODOLOGY

Contingency Analysis is a coarse grained parallel application in power system, due to its

natural independency and adaptability for these environments. The availability of multiprocessor architectures (Michele Di-Santo 2004) and cluster computing techniques offers a practical solution for such real time problems. This is further enhanced by advanced computer network architectures and used effectively in power system control centers.

```
main (int argc, char *argv [])
{
    MPI_Status status;
    MPI_Init (&argc, &argv);
    MPI_Comm Rank (MPI CW, & my rank);
    MPI_Comm Size (MPI CW, &p);
    if (rank = 0)
    Parent_Process();
    else
    Child_Process();
    MPI_Finalize ();
}

Parent _Process()
{
    BaseCase_PowerFlow();
    MPI_Send (PFmax, count, MPI_DOUBLE,
    dest, tag, MPI_CW);
    MPI_Recv (PI, count, MPI_DOUBLE, src,
    tag, MPI_CW, &status);
}

Child _ Process()
{
    MPI_Recv(PFmax, count, MPI DOUBLE,
    src, tag, MPI_CW, &status);
    Contingency_Analysis ();
    MPI_Send (PI, count, datatype, dest, tag,
    MPI_CW);
}
```

In this work, the parallel processes are created by means of MPI (Pacheco 1997) [6] as shown in the pseudo code above. If 'n' contingencies are to be analyzed then the number of processes required is 'n+1'. Using MPI n+1 instances of the program are executed simultaneously. Among these processes one is the rook process and other instances handles a contingency each. They are scheduled to run in separate processors in the cluster to account for the communication delay. The master process or the root process created handles the execution of the main program, the input - output operations and the task distribution. The detailed output required in each contingency case is also written to file. The master process drives all other processes. For example, the IEEE 30 Bus System has 30 Buses and 41 Lines. To simulate the 41 line outages we need to allocate 42 processors including the root process. In this simulation environment this process is taken care by job scheduling *apriori* and allocate 6 nodes each consisting of 8 processors.

5.0 DATA COMMUNICATION

Message Passing Interface (MPI) is a widely used tool for parallel and distributed processing. MPI programs are portable. MPI is used to share data between the processors (M. Shahidehpour) [7]. All processes communicate by sending and receiving messages. Data communication is done through a list of communication routines such as point to point communication or collective communication. MPI is exclusively used for problems that can be broken down into several processes running in parallel with information exchanged between them. Here a root or parent process and several child processes are created. The root process takes care of the input output operations and synchronization with the child processes. The maximum power flow of all the lines are required to find the performance index of the every contingency case. Hence the root process computes and sends the PF_{max} of all the lines to the child processes. The line outages are simulated in the child processes and line overloads and voltage deviation at the buses are tabulated. Then the performance index is communicated back to the root process to ensure synchronization of all the child processes. The data communication using MPI is shown in Figure 4.

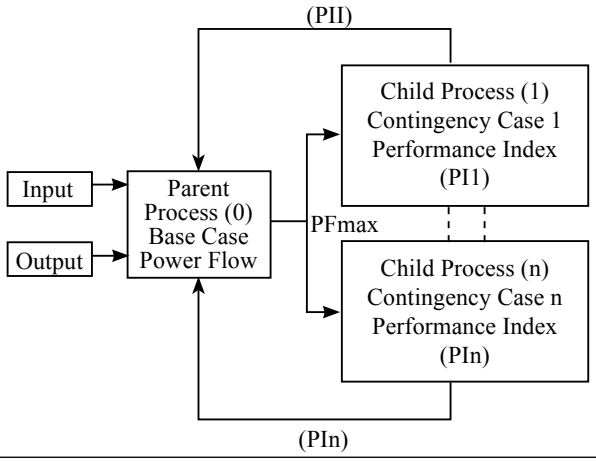

FIG. 4 DATA COMMUNICATION USING MPI

6.0 PERFORMANCE MEASURES

6.1 Granularity

In parallel computing granularity is a qualitative measure of computation to communication. It can be categorized into two categorized based on granularity as coarse grained parallelism and fine grained parallelism. In coarse grained parallelism relatively large amount of computation work is done between communication events, whereas in fine grained parallelism small amounts of computational work is done between communication events. The contingency analysis (Baleeiro A C and Montecelli A 1995) is an embarrassingly parallel application and hence falls under the coarse grained category [9,10].

6.2 Scalability

Scalability refers to a parallel systems ability to demonstrate a proportionate increase in parallel speed with the addition of more processes. The factors that contribute to the scalability include hardware, application algorithm and the parallel overhead.

6.3 Speedup

The speedup is the ratio of the runtime of a serial solution time to a problem to the parallel execution time (J C Mendes, O.R. Saavedra, S.A. Feitosa 2000) [8]. It can also be defined as the ratio of the fastest known serial program on one processor to that of the parallel program running on 'p' processors of the parallel system. Speedup is given by

$$S(n,p) = \frac{T(n)}{T(n,p)} \quad (2)$$

where,

p is the number of processes

T(n) is the serial execution time

T(n, p) is the mparallel execution time

For a fixed value of p, $0 < S(n, p) \leq p$

If S(n, p) = p, a program is said to have a linear speedup. This is a very rare condition since most parallel solutions add some communication overhead among other processors. A common occurrence is slowdown, in which case the parallel program running on more than one processor is actually slower than the serial program. This unfortunate occurrence results from an excessive amount of communication overhead.

The run time of a serial program is the actual time a program takes to solve a problem from the beginning to the completion of execution including the I/O operations.

$$T(n) = T_{calc} + T_{I/O} \quad (3)$$

The run time of a parallel program is the time that has elapsed from the moment the first process actually begins execution to the moment the last process completes execution of its last statement. In order to make good estimate of the performance of the parallel program the cost of communications must be included along with the I/O operations.

$$T(n) = T_{calc} + T_{I/O} + T_{comm} \quad (4)$$

6.4 Efficiency

Efficiency is the measure of the processor utilization in the parallel program, relative to the serial program. It is defined as

$$E(n,p) = \frac{S(n,p)}{p} \quad (5)$$

As $0 < S(n,p) \le p, \; 0 < E(n,p) \le 1$

If, E(n, p) = 1

then the program has a linear speedup.

If, E(n, p) <1/P

then the program has slowdown.

7.0 RESULTS AND DISCUSSION

The parallel processing approach to contingency analysis is implemented in the high performance computing server, a Linux Super cluster. It has about 2048 nodes, with dual and quad core processors and local cache memory. Each node having Intel Xeon E5472, 3.0 GHz 16 GB RAM and 250 GB hard disk with non blocking infiniband switch configuration. This facilitates to run a large number of processes simultaneously, thereby reducing the computation time. The effectiveness of the parallel processing approach is justified by the speedup and efficiency in comparison with the sequential algorithm. The performance is tested on IEEE Standard test cases such as 14 bus, 30 bus, 118 bus, 162 bus and 300 bus Systems. The simulation results are given in Table 1.

As the size of the system increases, the execution time for the complete contingency analysis increases. The parallel processing approach shows a considerable reduction in the total execution time for large systems. However there is no favorable improvement for small systems like IEEE 14 Bus and 30 Bus. Scalability is good, as the size of the system increases the speedup also increases. This is verified and shown in Figure 5, as the size of the system increases the number of iteration involved load flow increases, still a good parallel speedup is achieved using the proposed method. As defined in the earlier section, Speedup as well as Slowdown in exhibited in the test cases. Due to the communication overhead, the efficiency of the system is less than the inverse of the number of processes involved. Hence Slowdown is observed in small test cases, whereas speedup is achieved for large scale systems as shown in Figure 6.

TABLE 1

PERFORMANCE MEASURES OF TEST CASES

Test case	No. of process	Serial time (s)	Parallel time (s)
14 Bus 20 Lines	21	0.01	0.24
30 Bus 41 Lines	42	0.17	0.48
118 Bus 179 Lines	180	15.65	6.94
162 Bus 283 Lines	284	19.79	21.73
300 Bus 411 Lines	412	1236.8	47.89

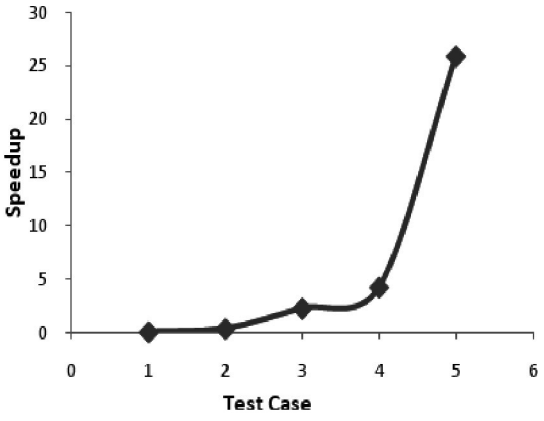

FIG. 5 SPEEDUP OF PARALLEL PROCESSING

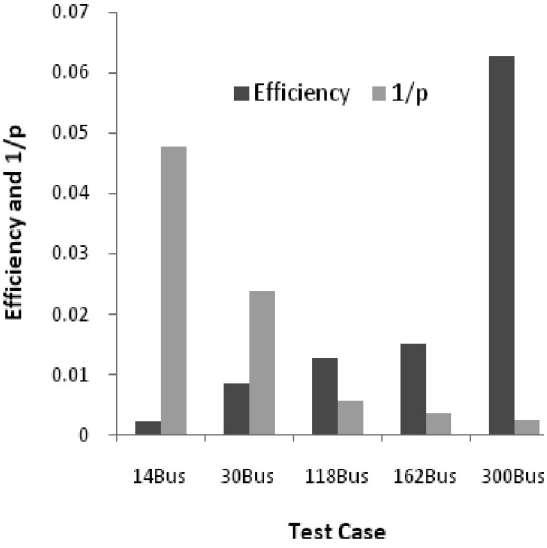

FIG. 6 EFFICIENCY OF PARALLEL PROCESSING

8.0 CONCLUSION

This work presents a method that employs a parallel processing approach to contingency analysis in power system security studies. This will enhance the real time simulation and make it simple, fast and accurate. Simulation is carried out in a high performance Linux cluster. Tests have been performed on IEEE standard test systems like 14 Bus, 30 Bus, 118 Bus and 162 Bus and 300 Bus. Only single line outages are considered for simulation. It shows appreciable performance in terms of speedup and efficiency for large systems. However for very small systems like 14 Bus System it reports a slowdown in performance. This can be further implemented for real time systems and will be a suitable tool for smart grid applications and planning.

REFERENCES

[1] Arnold, C P, Parr M I and Dewe M B. "An efficient parallel algorithm for the s olution of large sparse linear matrix equations", *IEEE, Trans. Comput*., Vol. C-32, No. 3, pp. 265–273, 1983.

[2] Qiang Jun Wu and Anjan Bose. "Parallel solution of large sparse matrix equations and parallel power flow", *IEEE, Trans. Power Syst*., Vol. 10, No. 3, pp. 1343–1349, 1995.

[3] Allen J W and Wollenberg B F. "Power generation operation and control", *A John Wiley & Sons, Inc*., 2003.

[4] Shahidehpour M. "Communication and control in electric power system", *IEEE, Press, A John Wiley & Sons*, Inc., 2003.

[5] Michele Di Santo, Alfredo Vaccaro, Domenico Villacci and Eugenio Zimeo. "A distributed architecture for online power systems security analysis", *IEEE, Trans. Indust. Electron*., Vol. 51, No. 6, 2004.

[6] Peter S Pacheco. "Parallel programming with MPI", *Morgan Kaufmann Publishers, Inc*., 1997.

[7] Baleeiro A C & Montecelli A. "Parallel and distributed solutions for contingency analysis in energy management system", *Proceedings of Circuits and Systems*, Vol. 1, pp. 449–452, 1995.

[8] Mendes J C, Saavedra O R and Feitosa S A. "A parallel complete method for real time security analyisis in power systems", *Electric Power Sys. Res*., Vol. 56, pp. 27–34, 2000.

[9] Alves A and Monticelli A. "Parallel and distributed solutions for contingency analysis in energy management systems", Proceedings of the 38th Midwest Symposium on *IEEE*, Vol. 1, pp. 449–452, 2002.

[10] Wu J and Bose A. "Parallel solution of large sparse matrix equations and parallel power flow", *IEEE, Trans. Power Syst.* Vol. 10, No. 3, pp. 1343–1349, 1995.