PNNL-23830

Pacific Northwest NATIONAL LABORATORY

Proudly Operated by Battelle Since 1965

Sub-second Parallel State Estimation

October 2014

Y Chen
M Rice
K Glaesemann
S Wang
Z Huang

U.S. DEPARTMENT OF ENERGY

Prepared for the U.S. Department of Energy under Contract DE-AC05-76RL01830

PNNL-23830

Sub-second Parallel State Estimation

Y Chen
M Rice
K Glaesemann
S Wang
Z Huang

October 2014

Prepared for
the U.S. Department of Energy
under Contract DE-AC05-76RL01830

Pacific Northwest National Laboratory
Richland, Washington 99352

Summary

This report describes the performance of the Pacific Northwest National Laboratory sub-second parallel state estimation (PSE) tool and discusses the benefits of the fast computational speed for power system applications. The test data were provided by the Bonneville Power Administration (BPA) and consist of 2 days' worth of hourly snapshots that include power system model data and measurement sets in a commercial tool format. These data were extracted from the commercial toolbox and fed into the PSE tool. With the integrated advanced solvers, the PSE tool can solve each BPA hourly state estimation (SE) problem within 1 second, which is more than 10 times faster than today's commercial tool.

This improved computational performance can help increase the reliability value of SE in many aspects: (1) the less time required to execute SE, the more time there is for operators to take appropriate actions and/or to apply automatic or manual corrective control actions. This increases the chances of arresting or mitigating the impact of cascading failures; (2) the SE can be executed multiple times within the time allowance. Therefore, SE robustness can be enhanced by repeating the execution of the SE with adaptive adjustments, including removing bad data and/or adjusting different initial conditions to compute a better estimate within the same time as a traditional state estimator's single estimate.

There are other benefits with the sub-second SE; for example, the PSE results can potentially be used in local and/or wide-area automatic corrective actions that currently depend on raw measurements to minimize the impact of bad measurements, and provide opportunities to enhance power grid reliability and efficiency. PSE also can enable other advanced tools that rely on SE outputs and could be used to further improve operator actions and automated controls to mitigate effects of severe events on the grid.

The power grid continues to grow and the number of measurements is increasing at an accelerated rate due to the variety of smart grid devices being introduced. A parallel SE implementation will deliver better performance than traditional, sequential SE by using the power of high-performance computing. This increased performance positions parallel state estimators as valuable tools for operating the increasingly complex power grid.

Acknowledgments

This work is funded by the U.S. Department of Energy's Office of Electricity Delivery and Energy Reliability through its Advanced Grid Modeling program, managed by Gilbert Bindewald.

Acronyms and Abbreviations

AMD	approximate minimum degree
BLAS	Basic Linear Algebra Subprograms
BPA	Bonneville Power Administration
CG	conjugate gradient
EMS	energy management system
HPC	high-performance computing
ILU	incomplete lower upper
LMP	locational marginal price
LU	lower upper
PCG	preconditioned conjugate gradient
PIC	PNNL Institutional Computing (program)
PMU	phasor measurement unit
PSE	parallel state estimation
RAS	remedial action schemes
SCADA	Supervisory Control and Data Acquisition
SE	state estimation
SPD	symmetric positive-definite
WLS	weighted-least square
WECC	Western Electricity Coordinating Council

Contents

Figures

Tables

1.0 Motivation

Power system state estimation (SE) is used to estimate the system states (bus voltage magnitude and phase angle) based on field measurement data from remote terminal units through a Supervisory Control and Data Acquisition (SCADA) system at a given point in time. It is the centerpiece of power grid and market operations. With a proper redundancy level of measurements, a state estimator can accurately and reliably estimate bus voltage magnitude and phase angle based on a statistical criterion. The SE outputs (i.e., bus voltage magnitudes and phase angles) are critical for monitoring system states for reliability purposes, assisting the operator’s decision-making process, and obtaining a real-time model for subsequent energy management system (EMS) functions. EMS functions can include contingency analysis, unit commitment, and optimal power flow. The quality of a state estimator can significantly influence power system operations.

Traditionally, the SE problem is solved using a sequential algorithm on a Windows-based platform. This approach does not consider using the power of HPC techniques. As a result, SE typically runs every 30 seconds for large, interconnected power systems in control centers. This introduces a delay between the present power system status and the estimated system status. This delay might be acceptable when system moves slowly; however, when there is a severe disturbance, the system states could potentially be drastically different. If system states cannot be obtained in near real time, it is difficult for the operators to locate problems and maintain operating conditions in a secure state when there are bad measurements.

Meanwhile, with the increased complexity of power system due to the integration of smart grid technologies that contain new dynamic characteristics and uncertainties, as well as the evolution of information technology, the complexity of the future power system applications is increasing. The traditional serial mode of SE algorithm cannot meet the more time-constrained requirement for the future, more dynamic power grid, even with advanced computer hardware. This creates a strong need to decrease the computational time of SE to help the operators to identify system states quickly by using the power of HPC, and thus provide more opportunities to support grid security and efficiency.

2.0 Weighted-Least-Square SE

There are many SE algorithms available, including lower upper (LU) decomposition-based WLS SE, orthogonal decomposition-based WLS SE, the hybrid method, and other algorithms. This project uses the WLS SE algorithm. This section briefly introduces this algorithm, including the measurement model and the problem formulation. This is a common method (see [1]**Error! Reference source not found.** for more details).

2.1 Measurement Model

This measurement model for an *n*-bus, *m*-measurements power system SE problem can be represented as [1]

$$z = h(x) + w \tag{1}$$

where:

z is the measurement vector with a length of *m*;

x is the power system states (the bus voltage magnitude and phase angle) vector with a length of 2**n*-1, as the phase angle at the reference bus is known;

$h(x)$ is a vector of nonlinear functions of length *m* that relates power system measurements to system states, *x*;

w is a vector of measurement errors of length *m*.

The typical measurement type includes voltage magnitude, real and reactive power injections, and real and reactive branch power flows. Normally, the measurement errors are assumed to be Gaussian noise with zero mean, and the measurements are independent and identically distributed.

2.2 Problem Formulation

The power system SE problem can be mathematically considered as a WLS optimization method with an objective function of

$$\min\ J(x) = [z - h(x)]^T R^{-1} [z - h(x)] \tag{2}$$

subject to $z = h(x) + e$, where *R* is the measurement covariance matrix, a diagonal matrix.

To minimize the objective function, *J(x)*, (2) has to be satisfied:

$$g(x) = \partial J(x)/\partial x = -H(x)^T R^{-1} [z - h(x)] = 0 \tag{3}$$

where $H(x) = \frac{\partial h(x)}{\partial x}$

By neglecting the higher order terms of the Taylor series expansion of *g(x)*, and doing more manipulation, the WLS state estimate can be found by iteratively solving

$$G(x^k)(x^{k+1} - x^k) = H^T(x^k) R^{-1} [z - h(x^k)]\ , \tag{4}$$

where

k is the iteration index;

x^k is the solution vector at iteration k;

$G(x^k) = \frac{\partial g(x^k)}{\partial x} = H^T(x^k)R^{-1}H(x^k)$;

$g(x^k) = -H^T(x^k)R^{-1}[z - h(x^k)]$.

The solution of Eq. (4) can be found by iteratively solving a linear equation of

$$A\Delta x = b \tag{5}$$

where:

$A = G\left(x^k\right),$ a sparse symmetric positive-definite (SPD) gain matrix;

$\Delta x = (x^{k+1} - x^k)$, the state difference of two consecutive steps;

$b = H^T\left(x^k\right)R^{-1}[z - h\left(x^k\right)]$.

Equation (5) is called the normal equation and needs to be solved for Δx in each Newton-Raphson iteration of the WLS SE process.

The solution procedure of WLS SE algorithm is as follows:

1. Set $k = 0$
2. Initialize the state vector x^k , typically a flat start
3. Calculate the measurement function $h\left(x^k\right)$
4. Build the measurement Jacobian $H\left(x^k\right)$
5. Calculate the gain matrix of G(x^k) =$H^T\left(x^k\right)R^{-1}H\left(x^k\right)$
6. Calculate the right-hand side of the normal equation $-H^T\left(x^k\right)R^{-1}[z - h\left(x^k\right)]$.
7. Solve Eq. (5) for Δx^k
8. Check for convergence using max $\left|\Delta x^k\right| \leq \epsilon$
9. If not converged, update $x^{k+1} = x^k + \Delta x^k$ and go to 3. Otherwise stop.

The majority of the SE process is done to solve the Eq. (5). Therefore, the focus of this work is to expedite the solving process of Eq. (5) with the aid of HPC techniques.

3.0 HPC Techniques in PSE

The normal equation [Eq. (5)] is a large, sparse, symmetric positive-definite linear equation. This section briefly describes typical methods and some popular software packages used to solve this type of equation.

3.1 Linear Equation Solving Methods

Generally, there are two categories of methods to solve linear equations: direct methods and iterative methods. Direct solution methods mostly perform an LU/LDU, QR, or Cholesky factorization of the gain matrix and try to reduce computational cost by minimizing fill-ins and/or ordering methods that reduce work and memory requirements, followed by the backward and forward substitutions. Normally, direct methods are more difficult to be parallelized than iterative methods.

Iterative methods begin with a given approximate solution, then modify the approximations at each iteration in a certain order, until the equation can converge within a pre-defined tolerance. The typical iterative method includes Jacobi, Gauss-Seidel, successive over-relaxation, and the conjugate gradient (CG) method.

The CG method is one of the best iterative methods for solving sparse symmetric positive definite (SPD) equations. It theoretically yields exact solutions within most N iterations, where N is the dimension of the linear equation. For a practical power system SE problem like the BPA system, the condition number of the gain matrix in Eq. (5) is largely due to the large range of measurement covariance. This large number leads to an expensive computational cost required for convergence. Therefore, the preconditioned conjugate gradient (PCG) method is more popular. It can reduce the condition number of the gain matrix by applying the inversion of a preconditioner matrix P at both sides of Eq. (5); Eq. (5) becomes

$$P^{-1}A\Delta x = P^{-1}b \tag{6}$$

A good preconditioner of P can result in a significant smaller condition number of matrix $P^{-1}A$, and thus a fast converge rate of the new linear Eq. (6). The PCG method is suitable for implementation on parallel computers. The detailed algorithm of a PCG method can be found in [2].

3.2 Linear Solver Packages Explored in PSE

There are many parallel sparse linear solvers for solving equations like Eq. (5). The following solvers have been studied in our work:

- Iterative solver package: Hypre [3][11][12], FASP [21], PARDISIO (MKL version) [17][18]
- Direct solver package: SuperLU [15], SuperLU_DIST [15], SuperLU_MT [15], KLU [16], UMFPACK [19], MUMPS [20], and NICSLU [22][23][24]

All these solvers are compile-time options within the software through a generic AXEQB.F software interface developed in this work. The AXEQB.F code can take advantage of the non-changing sparsity to improve computational performance during solver initialization for all solvers that support resource reuse.

Among these solvers, the Hypre solver and the NICSLU solver are the best in each category for the work conducted in this project. More information on these two solvers is provided below.

3.2.1 The Hypre Solver Package

The Hypre solver was previously selected based on its excellent scalability performance. The Hypre solver is based on Krylov methods, which are based on the principle that the inverse of a matrix can be found in terms of a linear combination of its powers. This yields an iterative method for solving large systems of linear equations while avoiding matrix-matrix operations. Orthogonalization is required because power iterations can lead to linear dependencies. Hypre uses conjugant gradient methods to solve the equations.

Hypre provides some built-in preconditioners. Three typical preconditioners were tested for solving Eq. (6): (1) Jacobian preconditioner (a diagonal preconditioner), (2) Euclid preconditioner (a family of incomplete LU [ILU] preconditioners) [5], and (3) ParaSails preconditioner (a parallel sparse approximate inverse preconditioner) [4][13][14]. Because the Jacobian preconditioner is simply the diagonal of the matrix A, it is the simplest to implement and computationally fastest to apply. For diagonally dominant A matrices, it is often very effective. The Euclid method divides the A matrix into pieces and performs an ILU factorization on each part. These approximate L and U matrices are then used to generate P^{-1}. The ParaSails preconditioner generates a sparse approximation to A^{-1}. The approximate inverse's sparsity pattern is forced to be the pattern of a power of a sparsified matrix M, formed by dropping nonzeros that are smaller than a threshold. Therefore, it is forced to be sparse, but also approximate. A least-squares minimization is used to compute the inverse M (i.e., minimizing the Frobenius norm of $I - AM$). A post-filtering technique further reduces the cost of applying the preconditioner [4][13][14]. Among these three, ParaSails performed best for the test problems with real data based on the performance test conducted last year [25].

3.2.2 The NICSLU Solver Package

More recently the NICSLU solver package has been found to provide better computational performance than the Hypre solver for the BPA test cases. The NICSLU solver is a parallel LU solver. It is specially targeted at circuit simulation, making it well-suited for the power grid matrices. NICSLU avoids using Basic Linear Algebra Subprograms (BLAS), and uses only custom sparse routines. NICSLU supports the iterative power grid case that requires many factorizations with the same sparsity pattern. Although generally fast, NICSLU is not as fast as some other methods if this reuse is not enabled. During this one-time setup phase, several steps occur: (1) the MC64 algorithm [30][31] is used to maximize the sum or product of the diagonal elements of the A matrix multiplied by the permutation matrix to improve numerical stability; (2) the approximate minimum degree (AMD) algorithm is used to reorder the matrix to reduce fill-in; and (3) a static symbolic factorization is used to suggest whether a matrix should use the parallel or sequential algorithm. The time-consuming step of the LU process is naturally the numerical factorization of the matrix. The NICSLU method is parallelized using threads. The code during initialization determines if the sparsity pattern is suitable for threaded parallelism—which it always was for matrices studied.

4.0 Test Case Results

This section discusses the BPA test data properties, the PSE performance results with the Hypre and NICSLU libraries, and some benchmarking results against commercial EMS SE functions.

4.1 BPA Data Properties

Real measurements and system models from BPA, rather than simulated data and research models, were used to test the performance of the PSE tool. System model extraction and measurement alignment of real measurements to the models that feed into the PSE solver was a challenging task. The difficulties are outlined in a previous paper [25]. The summary of the cases is being presented here for completeness of the report.

BPA provided SE data as 2 days' worth of hourly database snapshots in a commercial, proprietary format. The data in the snapshots are the measurement sets used and power system data (e.g., breaker settings, generator status, and load values) in node-breaker model format. The BPA system model includes approximately 7,500 buses, 9,300 branches, and 27,000 measurements. The measurement types include bus voltage magnitude, real and reactive power injection, and real and reactive power flow. Topological changes occurred between the 47 hourly snap shots. These changes can create a split bus or switch line status, which requires re-verification that the locations of the bus measurements are correct for each snapshot.

All the measurements are assigned with different weights. The ratio of the maximum and minimum value in the measurement covariance matrix R is on the order of 10^6, a ratio that is larger than typical ratio in simulated measurements, which could lead to an ill-condition problem. Table 1 summarizes the measurements used by SE.

Table 1. Summary of state estimation model used in studies

Number of Line Measurements	9,696
Number of Injection Measurements	14,988
Number of Voltage Measurements	2,406

The 2-day hourly snapshots provided by BPA contain 47 cases. All the cases have been imported to the PSE tool and executed with following setup:

- Convergence tolerance for phase angle: 2.0E-2
- Convergence tolerance for voltage magnitude: 1.0E-2
- Maximum iteration number: 20
- Estimate voltage magnitude and phase angle together

4.2 Hypre PSE Performance

The PSE tool with the Hypre solver can run the SE within 5 seconds, comparable to today's SCADA rate [25]. The computational time for all 47 cases with the Hypre solver, ParaSail preconditioner, and running on eight cores is shown in Table 2. These numbers were obtained from a PNNL catamount eight-node cluster. Each node has two Intel Xeon X5570 central processing units (CPUs) (Nehalem) with four cores each for a total of eight CPU cores per node. The CPU frequency is 2.53 GHz. There is 24 GB of memory on each node. All the nodes are connected together with Mellanox DDR ConnectX InfiniBand and running OpenFabrics 1.4.2.

Table 2. Computational time for all 47 cases with the Hypre solver, Parasail preconditioner, and running on eight cores on the PNNL catamount machine (2.53 GHz CPU)

Case Number	PSE Time (s)	Case Number	PSE Time (s)	Case Number	PSE Time (s)
1	3.31	17	4.61	33	4.32
2	3.63	18	3.59	34	4.86
3	3.33	19	3.65	35	3.90
4	3.90	20	3.06	36	3.17
5	3.89	21	3.37	37	4.10
6	1.85	22	3.80	38	3.58
7	3.43	23	2.94	39	3.49
8	3.71	24	3.86	40	3.50
9	3.09	25	2.81	41	3.30
10	3.15	26	3.97	42	2.76
11	3.88	27	4.03	43	4.20
12	3.88	28	3.95	44	3.52
13	3.22	29	4.01	45	4.43
14	4.00	30	3.96	46	3.95
15	3.09	31	4.56	47	3.84
16	3.06	32	4.72		

4.3 NICSLU PSE performance

Recent work shows that the NICSLU solver performs the best for the test systems. This section focuses on the performance results of the NICSLU solver. A series of runs with one, two, four, and eight threads were performed. The PSE with NICSLU solver package was implemented on the Olympus HPC computer, supported by the PNNL Institutional Computing (PIC) program [26].

The Olympus machine has approximately 22,100 cores in 692 nodes. Each node has dual sockets with 16 cores per socket Interlagos processors running at 2.1 GHz (32 cores/node). Each node has 64 GB of 1600 MHz memory (2 GB/socket). The operating system running on Olympus is Red Hat Enterprise Linux Client release 5.7. A QDR InfiniBand high speed network (40 GB/s) is used for internal communication.

As a set of initial results, the computational time for all 47 cases with one thread for the NICSLU solver is shown in Table 3. These numbers are average values of three runs. Figure 1 shows the data in Table 3 and Figure 2 shows the results for one-, two-, four-, and eight-thread trials. For all of these simulations, all the cases can be solved in less than 1 second using one thread.

Table 3. The computational time for all 47 cases with NICSLU solver, one thread on the PIC machine (2.1 GHz CPU)

Case Number	PSE Time (s)	Case Number	PSE Time (s)
1	0.36	25	0.43
2	0.36	26	0.40
3	0.35	27	0.39
4	0.37	28	0.36
5	0.37	29	0.36
6	0.39	30	0.35
7	0.70	31	0.36
8	0.41	32	0.30
9	0.37	33	0.42
10	0.45	34	0.30
11	0.41	35	0.43
12	0.43	36	0.42
13	0.42	37	0.43
14	0.44	38	0.42
15	0.43	39	0.42
16	0.42	40	0.43
17	0.54	41	0.42
18	0.45	42	0.42
19	0.42	43	0.43
20	0.43	44	0.46
21	0.46	45	0.47
22	0.40	46	0.45
23	0.47	47	0.44
24	0.51		

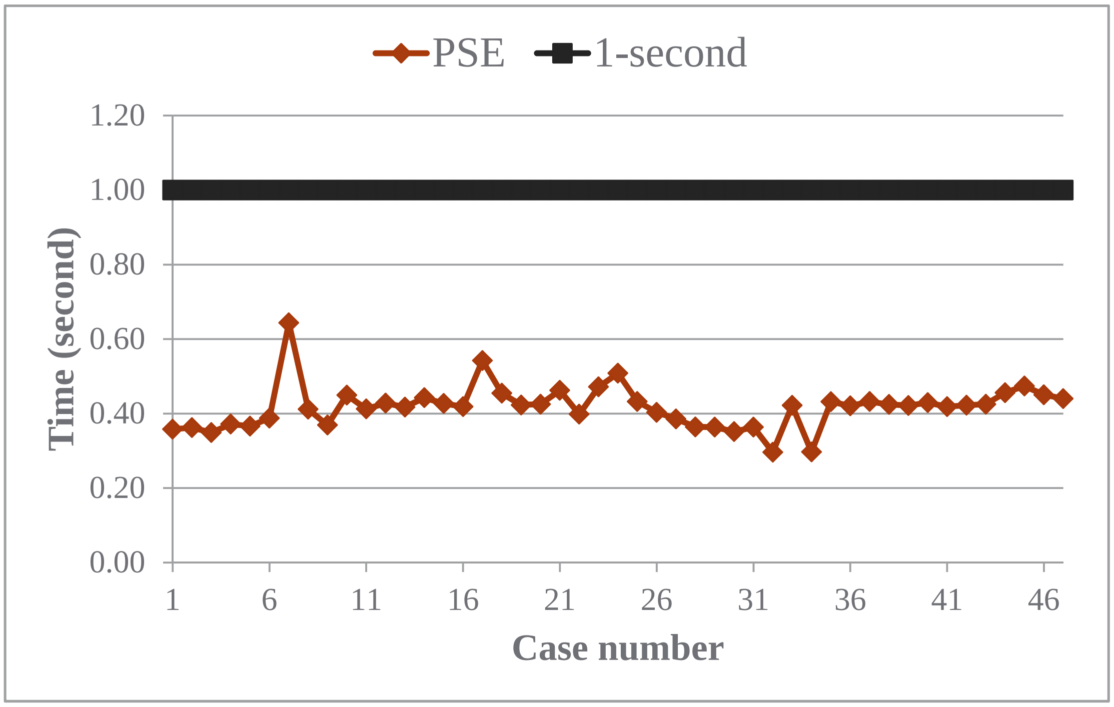

Figure 1. Computational time for all 47 cases with the NICSLU solver, one thread (flat start)

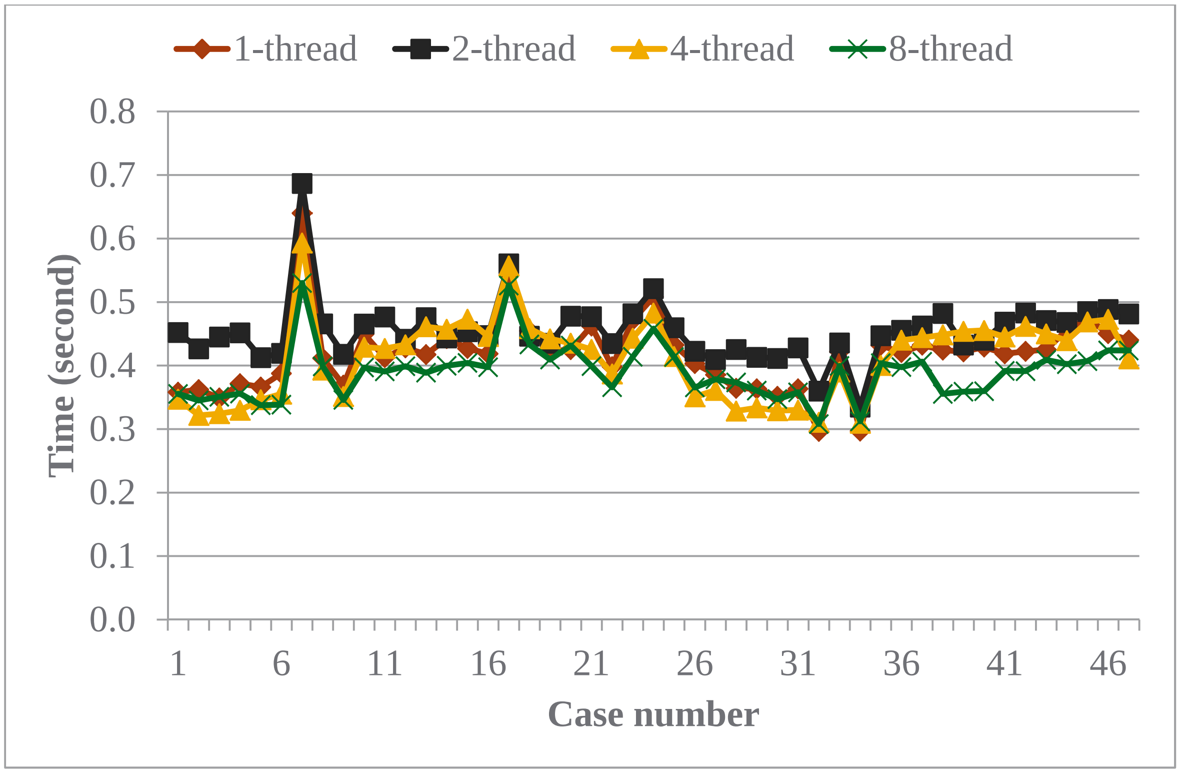

Figure 2. Performance of the NICSLU solver with one, two, four, and eight threads

To better see the performance with different threads, the average, minimum, and maximum time for all 47 cases are shown in Table 4. Again, these are average numbers from three runs. Table 4 suggests that using more threads improves the computational time a little, but not significantly. This is mainly because the size of the test system is relative small. The threaded version of the NICSLU solver requires extra setup and initialization. Therefore, for small systems this initialization will exceed the parallel speedup obtained with threading. In the example shown, the setup time is greater than the speedup for two threads, but not four or eight. For larger problems, the linear solver time (which is proportional to the system size to the third power) will dominate any setup time. For more difficult problems, even the two-threaded example problem will see a speed increase since the initialization is only done once and is independent of the number of iterations required. Better scalability is expected when the solver is applied to larger systems. While not explored as part of the work in this report, examining this scalability is a portion of the future work for this research.

Table 4. The average, maximum, and minimum computational times for all 47 cases with NICSLU solver using one, two, four, and eight threads on the PIC machine (2.1 GHz CPU)

	1-thread	2-thread	4-thread	8-thread
Average	0.42	0.45	0.41	0.39
Minimum	0.30	0.33	0.30	0.30
Maximum	0.70	0.70	0.61	0.54

4.4 PSE Benchmark

The main differences between the EMS tool environment and the PSE environment need to be pointed out before conducting the PSE benchmark against the commercial EMS tool. These differences are (1) the CPU frequency for the commercial EMS is 2.67 GHz, while the CPU frequency on the PIC machine is 2.1 GHz; (2) commercial EMS is running on Windows platform, while the PIC machine is running on Redhat Linux operation system; and (3) the SE algorithm on the commercial EMS tool is the QR orthogonal decomposition-based factorization approach (non-square matrix), while the PSE uses the WLS gain matrix-based factorization algorithm. The WLS gain matrix-based factorization algorithm is easier to use parallel linear solver techniques than the QR based approach. More details on the differences between the two algorithms can be found in [7].

Three aspects were considered for comparing the EMS tool against the PSE tool: accuracy, number of iterations, and computational time. The accuracy is measured by a cost function defined in the commercial tool manual of [7]. The cost is a scalar to represent the system-wide fit of measurements to the model that can be used to measure the accuracy of the SE algorithm.

$$cost = \sum \left(\frac{residual}{stdDev} * systembase \right)^2 \tag{7}$$

where the residual is the difference between the measurement and the estimated values using the calculated states.

The performance comparison between the PSE and EMS tools in terms of cost, number of iterations, and computational time is shown in Table 5. The EMS cost is directly obtained from the EMS screen outputs, while the PSE cost is calculated based on Eq. (7). The state estimates of both tools verified that the voltage profiles were very close. Table 5 includes the PSE computational time and a normalized PSE time that compensates for the CPU frequency difference (denoted as PSE_CPU). Table 5 suggests that the PSE tool takes fewer iterations and less computational time to obtain the SE outputs with the same level of accuracy as the commercial EMS tool.

Table 5. Benchmark comparison of cost, iteration numbers and computational time for all 47 cases between EMS tool and PSE tool (EMS with 2.67 GHz CPU, PSE with 2.1 GHz CPU)

	Cost		# of Iterations		Time (s)		
	EMS	PSE	EMS	PSE	EMS	PSE	PSE_CPU
Avg	114,053	135,570	13	6	5.70	0.42	0.33
Min	103,325	109,613	11	4	4.11	0.30	0.23
Max	126,840	149,958	18	10	6.46	0.65	0.51

A scatter plot of the number of iterations and execution time of the PSE (eight-thread) and EMS tools for all 47 BPA cases is shown in Figure 3. The plot provides an overall picture of the PSE tool performance for all test cases. The PSE tool took less time to execute than the EMS tool for all 47 cases.

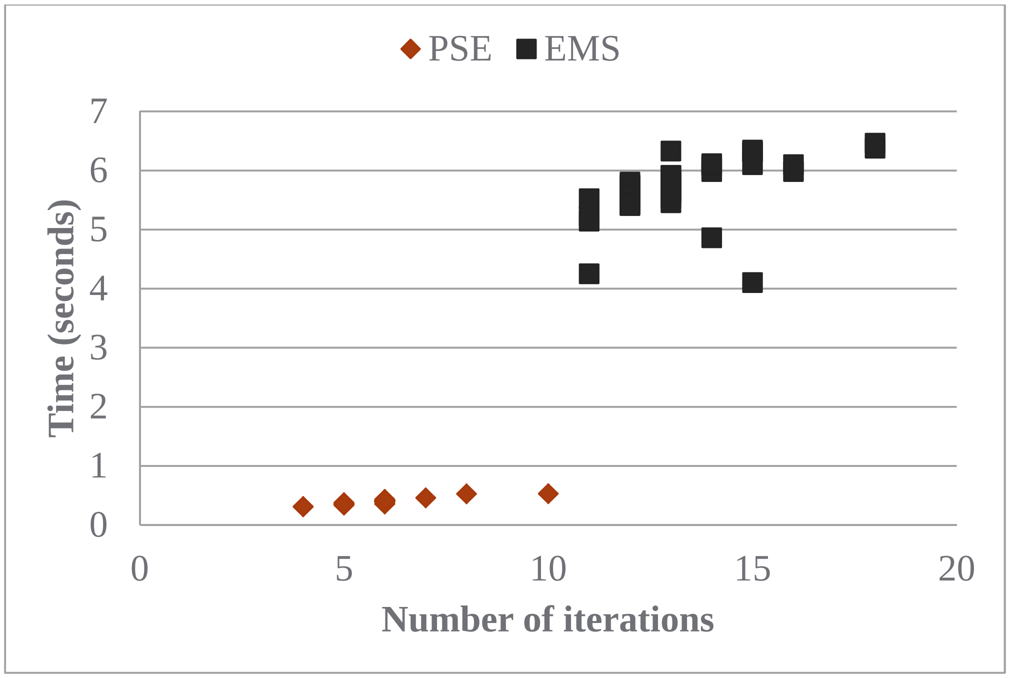

Figure 3. Scatter plot of the number of iterations and execution times of the PSE and EMS tools for all 47 BPA cases

The speedup between the EMS and PSE tools (NICSLU solver and eight threads) for the average, minimum, and maximum computational time for all 47 cases is shown in Table 6. The speedup numbers include one with original performance and one normalized value, the latter again considering the CPU frequency difference (denoted as speedup_CPU). From Table 6, when considering the average computational time, the PSE tool is about 13 times faster. If CPU frequency differences are considered, the speedup is more than 17. This suggests that operators could have time to stop a cascading failure compared with today's method. Some benefits of this speedup are further discussed in section 5.0.

Table 6. Benchmark computational comparison for all 47 cases between EMS tool and PSE tool

	EMS (s)	PSE (s)	PSE_CPU (s)	speedup	speedup_CPU
Average time	5.7	0.4	0.33	13.57	17.27
Minimum time	4.11	0.3	0.23	13.70	17.87
Maximum time	6.46	0.7	0.51	9.94	12.67

5.0 Benefits of Sub-second State Estimation

Sub-second state estimation will allow operators to know system status faster compared with today's practice. This improvement can help increase the reliability value of SE. Generally speaking, the less time required for execution of state estimation, the more time remains for operators to take appropriate actions and/or to apply automatic or manual corrective control actions. This would increase the chances of arresting a cascading failure or mitigating its impact.

5.1 Example of the 2011 Pacific Southwest Blackout

The 2011 Pacific Southwest Blackout provides example of the need for a fast state estimator. During the event, the time from the 500 kV line tripping to San Diego Island collapsing was 11 minutes. There are 18 notable events in the blackout, mainly associated with transformer and/or transmission line overloading, especially for some longer time outages. These situations could be prevented with re-dispatching of generation or topology changes within a certain timeframe before the system gets worse. The time intervals between each event are shown in Figure 4.

At today's state estimator periodicity (30 seconds or longer), only four state changes would have been observed (referring to the black line in Figure 4). With the sub-second PSE, the operator could have observed 14 of 18 state changes (the red line indicates the 1-second SE computational time in Figure 4). This early detection can provide timely, correct information to the operators during disturbances, which can help stop the propagation of cascading failures.

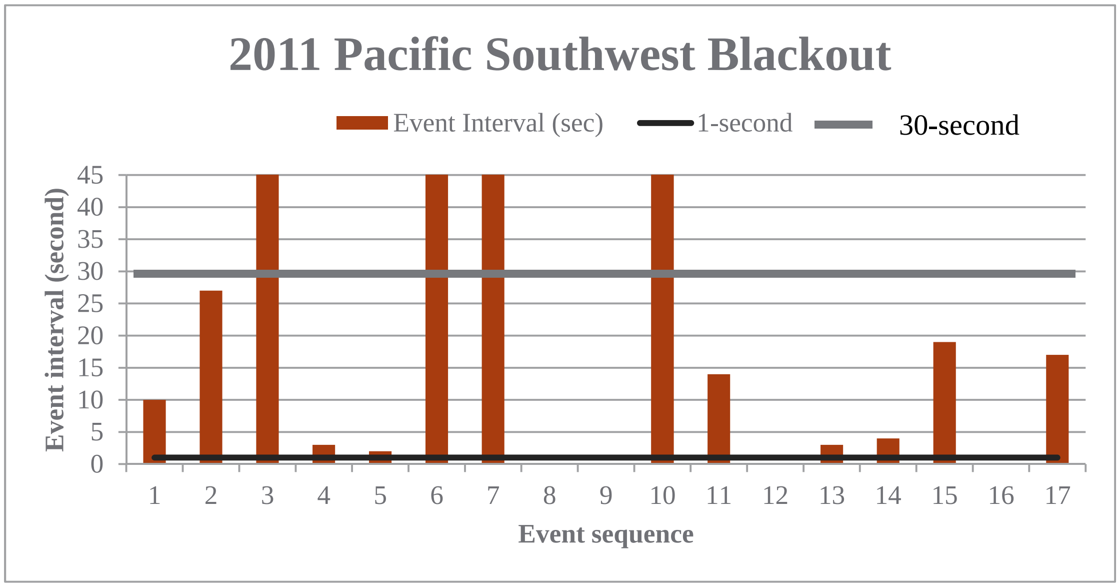

Figure 4. 2011 Pacific Southwest Blackout event time interval [9][10]

5.2 Case Study with a 1081-bus System

To further examine the benefits of improved reliability, a cascading failure case study using a 1081-bus, reduced Western U.S. power system model is performed. The system map of the 1081-bus system, as well as the locations of the simulated events, is shown in Figure 5.

5.2.1 Baseline Scenario

The baseline scenario described in this section is a cascading outage that takes 39.5 seconds to reach new equilibrium. The event sequence (F1, F2, and F3) is shown in the zoomed view of the square box. Quantities of relevant elements of the 1081-bus system demonstrate the event sequence shown in Figure 6. The details of the events, assuming no actions to be taken, are as follows:

- **Event 1**: A three-phase short circuit failure occurs in the line between Bus 340 and Bus 1011 at 5 seconds. This event results in the overload of the line between Bus 362 and Bus 1010, which triggers Event 2.
- **Event 2**: The line between Bus 362 and Bus 1010 trips because of overload. This results in the overload of the line between Bus 360 and Bus 341, which triggers Event 3 at 36 seconds.
- **Event 3**: The line between Bus 360 and Bus 341 trips because of overload at 39.5 seconds.

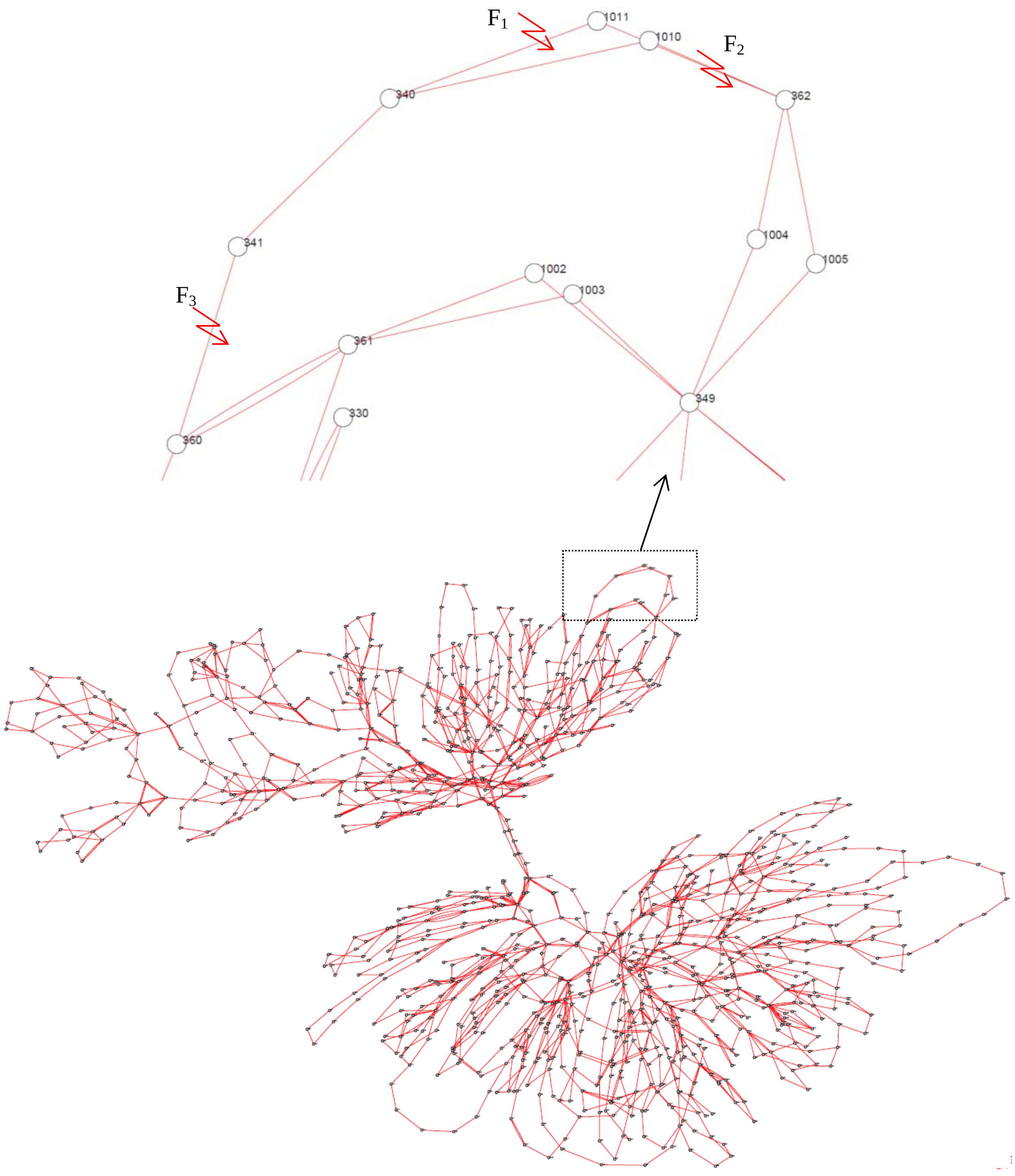

Figure 5. A 1081-bus test system, with a close-up view of the sequence of the simulated events

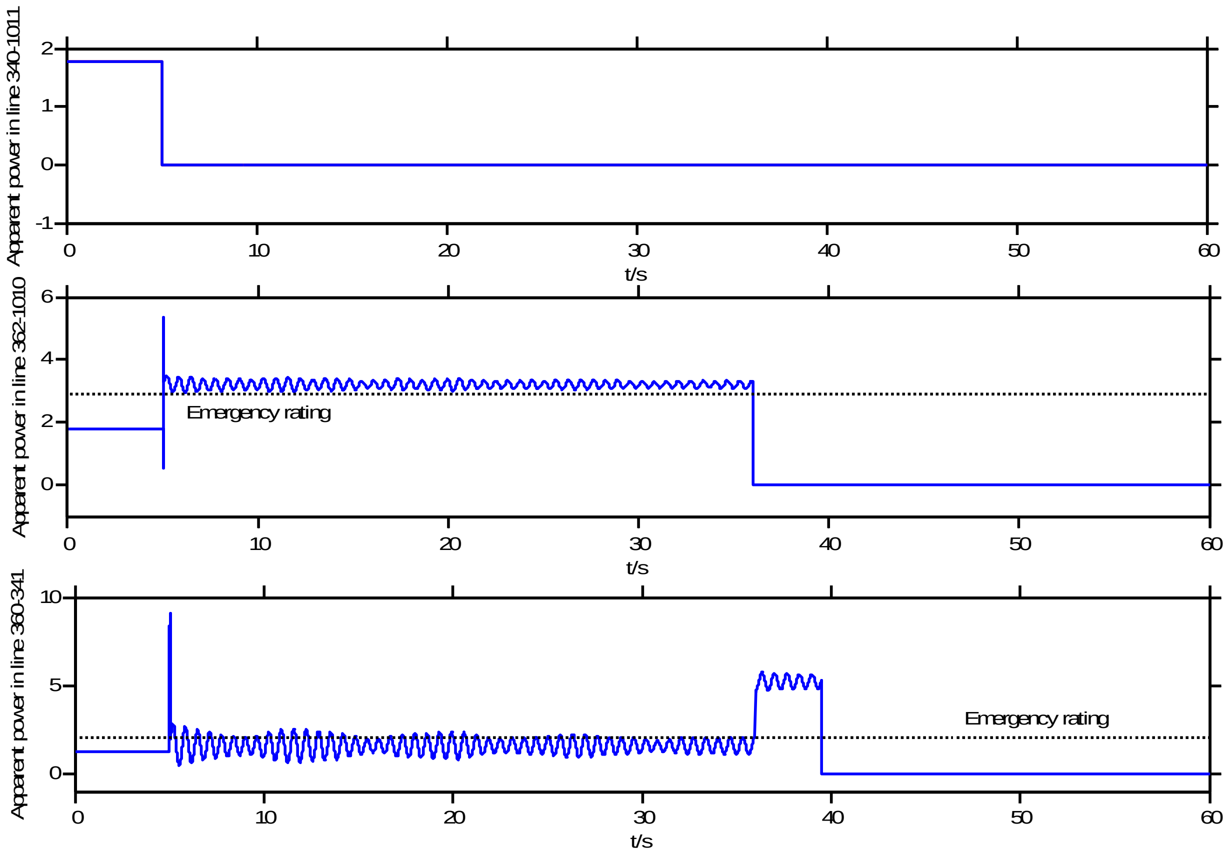

Figure 6. Cascading failure sequences of the simulated events

5.2.2 Improved Grid Operation

To show how a fast SE can help increase system reliability, more simulations were run to determine the consequences of taking action at different times. The operator action adopted here is to modify power flow distribution in the network to mitigate overloaded transmission lines. Specifically, the output of the generator connected to Bus 341 was reduced. This generator has a ramp-down restriction of about 20% per minute. The impact of the same action with different delays is shown in Figure 7. In the figure, the cyan lines show the effect of the action with 5 seconds of delay and the blue lines show the effect of the action with 10 seconds of delay. All these line trip times are determined based on [27][28], an extension of the authors' previous work in [8].

As Figure 7 shows, the cascading failures can be prevented if actions are taken in time. If actions are not taken fast enough, a cascading failure occurs that results in the loss of line, generation, and load, summarized in Table 7. Reduction in SE times allows improvements in reducing the reaction delays shown in the scenario, improving overall system reliability.

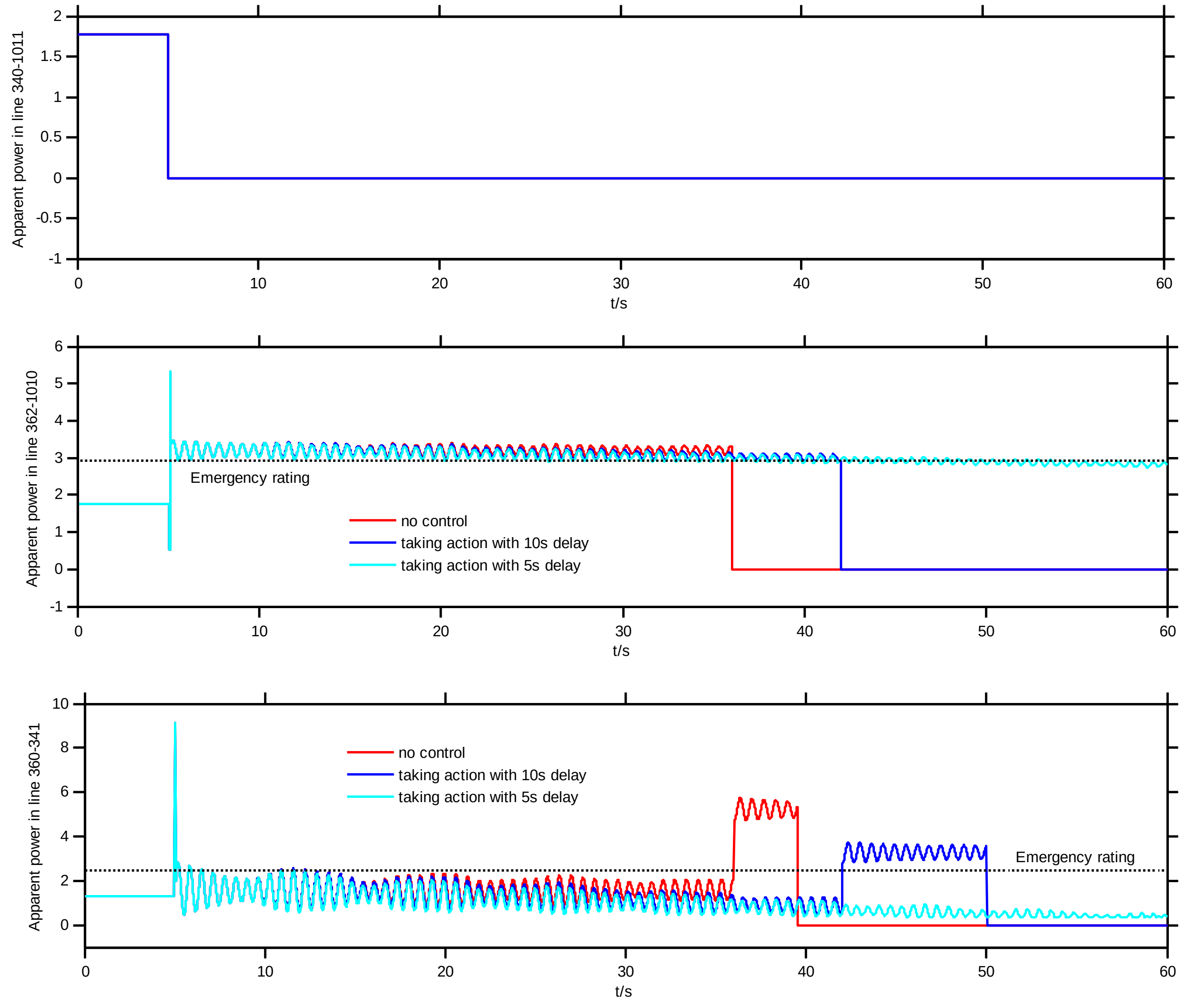

Figure 7. Scenarios with operator action taken at different times

Table 7. Results of action with different delays

Loss / Actions	Loss of load	Loss of generation	Loss of lines
Action with 10s delay	40 MW	440MW	5
Action with 5s delay	0	0	1

While not directly explored in this scenario, if SE becomes fast enough, SE results could also be used in automatic corrective control actions. However, most current automatic corrective actions, called remedial action schemes or special protection systems, do not rely on SE results. Instead, the automatic corrective actions use dedicated monitoring systems from a subset of the system, i.e., local control. A fast, system-wide state estimator can not only help provide a faster, more correct control signal to automatic corrective control actions than taken in local mode, it can broaden the potential of designing and deploying wide-area automatic control schemes by taking advantage of information from larger areas of the system. There is an expectation that the industry is moving toward a more complex (e.g., state-based) remedial action schemes (RAS). A leader in this technology is Southern California Edison, with smart RAS that is a centralized system tightly coupled with the EMS [29]. The centralized RAS can perform actions in 50 milliseconds using SCADA measurements and simple gate logic. While PSE cannot operate this fast at this stage, it can inform the analytics with more current information than traditional SE to improve operation of the RAS schemes.

5.3 Improve the Robustness of State Estimation

State estimation can experience divergence issues due to many factors, such as topology errors, parameter errors, unknown noise sources, and measurement outliers. Consequences of such divergence can often significantly affect grid operations. As an example, there have been situations where locational marginal prices (LMPs) cannot be calculated because there are no valid SE results for more than an hour. Without this pricing information, the overall economic dispatch of generators and other grid resources is more difficult and directly reduces the reliability of the power grid.

Fast computation of SE can provide opportunities to re-execute the SE process without delaying the availability of SE results. To help prevent divergent state estimates and to improve the robustness of the estimates, the SE solution process can be repeated multiple times with guided adjustments to work around such issues. For example, if today's SE were performed every 30 seconds, the sub-second state estimate presented in this report could run 30 times within the same 30-second allowance. The basic idea is shown in Figure 8, where multiple parallel state estimates can be completed in the time of a single, traditional state estimate. Each time the SE is re-executed, adaptive adjustments can be applied to improve the chance of convergence. These adaptive adjustments can be determined by the methods for detecting bad data or different initial conditions, or even using existing numerical methods in commercial tools.

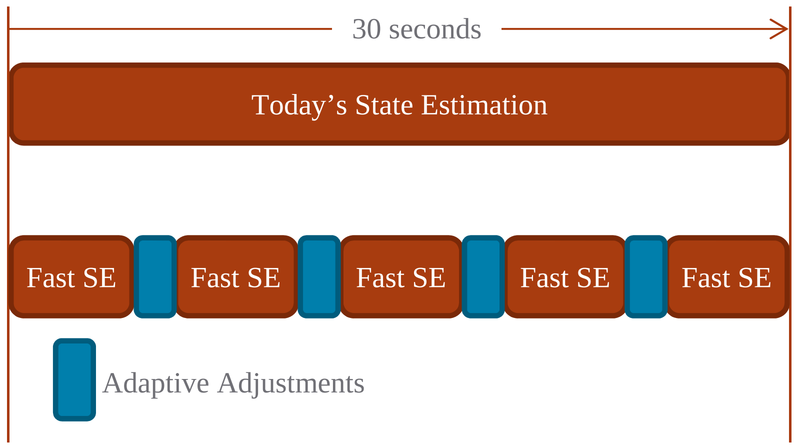

Figure 8. Multiple executions of fast SE for the same interval as today's SE

Not only can the improved robustness of fast SE prevent the LMP failure described earlier, it can also improve the efficiency of the LMP market. Even if bad data did not cause a divergent SE case, it could influence the LMP market with incorrect values. With fast SE, bad data in the estimation process can be detected and removed more quickly to enhance decision support and reduce the risk of an unstable grid status.

While the LMP scenario represents a specific example, many power grid functions currently depend on SE outputs. A more robust SE can help minimize the impact of unknown system states and provide more accurate inputs for these processes. The overall result is a more reliable and more efficient power grid.

6.0 Conclusion and Future Work

This report presents an implementation of a PSE tool developed by PNNL and its performance evaluation results using real power system SE data from BPA. The test performance shows that the PSE tool can solve practical large-system SE problems within 1 second, which is much faster than today's SCADA sampling cycle. The benchmarking test against a commercial tool demonstrated that the developed PSE tool is more than 10 times faster than today's tool. This increased performance positions PSE as a valuable tool for operating the increasingly complex power grid.

This report also discusses the benefits of sub-second SE. For example, the fast SE computational speed allows operators to gain the knowledge of the power grid faster and take action earlier. A case study of a reduced Western Electricity Coordinating Council (WECC) model cascading event demonstrates that fast SE can help mitigate the impact of a cascading failure. This sub-second computational time can also improve the robustness of the SE by re-executing SE multiple times within time allowance. This ability helps minimize the impact of bad data for subsequent analysis. The sub-second SE outputs can also potentially enable advanced control by replacing raw measurements with more accurate, system-wide state estimates as inputs to a control system. Such applications provide more potential and opportunities to enhance power grid reliability and efficiency.

With the fast deployment of smart grid technologies and devices, variable components will cause rapid state changes to be part of everyday operations. The size of the power system problems will also increase quickly. Complexity will also be increased with the greater availability of phasor measurement unit (PMU) values and their incorporation into state estimators. PMU-introduced factors, such as multi-time scale data (e.g., PMU and SCADA data sampled at different rates) and the sheer volume of data will increase the overall size of the SE problem. This will require a faster and more frequently updated state estimator. To meet this challenge, the authors will extend the study to larger systems in the near future.

7.0 References

[1] A. Abur, and A. Gomez Exposito, *Power System State Estimation Theory and Implementation*, CRC Press, 2004.

[2] M.R. Hestenes, and E.L. Stiefel, “Methods of Conjugate Gradient for Solving Linear Systems,” *J. of Research of the Nat. Bureau of Standards*, 49:409-436, 1952.

[3] High Performance Preconditioner Library (Hypre) Library. [Online]. Available: http://acts.nersc.gov/hypre/, Lawrence Livermore National Laboratory.

[4] ParaSails preconditioner. [Online]. Available: https://computation.llnl.gov/casc/parasails/parasails.html.

[5] Euclid preconditioner. [Online]. Available: http://www.cs.odu.edu/~pothen/Software/Euclid/.

[6] Y. Chen, Z. Huang, Y. Liu, M. Rice, and S. Jin, “Computational challenges for power system operation,” in *Proc. 2012 the 45th Hawaii International Conference on System Sciences*, Maui, HI, Jan. 2012, 2141-2150.

[7] Y. Chen, M. Rice, S. Jin, and Z. Huang, “Parallel State Estimation Assessment with Practical Data,” In *Power and Energy Society General Meeting (PESGM)*, Vancouver, BC, Canada, pp. 1-5, July 21-25, 2013.

[8] M. Elizondo, Y. Chen, and Z. Huang, “Reliability value of fast state estimation on power systems,” in *Proc. 2012 IEEE PES Transmission and Distribution Conference and Exposition*, Orlando, FL, 2012.

[9] Western Electricity Coordinating Council Staff, *White Paper High Level Summary of September 8, 2011 Pacific Southwest Outage Survey Results*, Western Electric Coordinating Council, Salt Lake City, UT, July 21, 2012.

[10] FERC/NERC report, *Arizona-Southern California Outages on September 8, 2011 Causes and Recommendations*, FERC/NERC, Washington, D.C., April 2012

[11] R.D. Falgout, J.E. Jones, and U.M. Yang, “Pursuing scalability for Hypre’s conceptual interfaces,” *ACM Trans. Math. Software* 31(3):326–350, 2005.

[12] R.D. Falgout, J.E. Jones, and U.M. Yang, “The Design and Implementation of Hypre, a Library of Parallel High Performance Preconditioners,” In *Numerical Solution of Partial Differential Equations on Parallel Computers, A.M. Bruaset and A. Tveito, eds., Springer-Verlag* 51:267–294, 2006.

[13] E. Chow, “Parallel implementation and practical use of sparse approximate inverses with a priori sparsity patterns,” *Int. J. High Perf. Comput. Appl.* 15:56-74, 2001.

[14] E. Chow, “A priori sparsity patterns for parallel sparse approximate inverse preconditioners,'” *SIAM J. Sci. Comput.* 21:1804-1822, 2000.

[15] SuperLU (Supernodal LU). [Online]. Available: http://crd-legacy.lbl.gov/~xiaoye/SuperLU/.

[16] KLU: Sparse LU Factorization. [Online]. Available: http://www.cise.ufl.edu/research/sparse/klu/.

[17] PARDISO 5.0.0 Solver Project. [Online]. Available: http://www.pardiso-project.org/.

[18] Intel® Math Kernel Library. [Online]. Available: https://software.intel.com/en-us/intel-mkl.

[19] UMFPACK: unsymmetric multifrontal sparse LU factorization package. [Online]. Available: http://www.cise.ufl.edu/research/sparse/umfpack/.

[20] MUMPS : a parallel sparse direct solver. [Online]. Available: http://graal.ens-lyon.fr/MUMPS/.

[21] FASP: Fast Auxiliary Space Preconditioning. [Online]. Available: http://fasp.sourceforge.net/.

[22] C. Xiaoming, Y. Wang, and H. Yang, "NICSLU: An Adaptive Sparse Matrix Solver for Parallel Circuit Simulation," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on* 32(2):261-274, 2013.

[23] C. Xiaoming, Y. Wang, and H. Yang, "An adaptive LU factorization algorithm for parallel circuit simulation," *Design Automation Conference (ASP-DAC), 2012 17th Asia and South Pacific*, pp.359-364, Jan. 30, 2012 through Feb. 2, 2012.

[24] C. Xiaoming, W. Wu, Y. Wang, H. Yu, and H. Yang, "An EScheduler-based Data Dependence Analysis and Task Scheduling for Parallel Circuit Simulation," *Circuits and Systems II: Express Briefs, IEEE Transactions on* 58(10):702-706, 2011.

[25] Y.Chen, S. Jin, M. Rice, and Z. Huang, "SCADA-rate parallel state estimation assessment with utility data," In *Proc. Power and Energy Society General Meeting (PESGM),National Harbor, MD*, July 2014.

[26] PNNL Institutional Computing (PIC) Program. [Online]. Available: http://pic.pnnl.gov/.

[27] IEEE Standards Board, "IEEE standard inverse-time characteristic equations," 1996.

[28] G. Benmouyal and S.E. Zocholl, "Testing dynamic characteristics of overcurrent relays," in *Proc. Communications, Computers and Power in Modern Environment, IEEE conference*, Saskatoon, Canada, May 17-18, 1993.

[29] S. Wang, G. Rodriguez, "Smart RAS (Remedial Action Scheme)," *Innovative Smart Grid Technologies (ISGT), 2010* pp. 1, 6, 19-21, 2010

[30] I.S. Duff and J. Koster, "The design and use of algorithms for permuting large entries to the diagonal of sparse matrices," *SIAM J. Matrix Anal. Appl.* 20(4):889–901, 1999.

[31] I.S. Duff and J. Koster, "On algorithms for permuting large entries to the diagonal of a sparse matrix," *SIAM J. Matrix Anal. Appl.* 22(4):973-996, 2000.