Computational Challenges for Power System Operation

Yousu Chen  
Pacific Northwest National Laboratory  
yousu.chen@pnnl.gov

Zhennu Huang  
Pacific Northwest National Laboratory  
zhennu.huang@pnnl.gov

Yan Liu  
Pacific Northwest National Laboratory  
yan.liu@pnnl.gov

Mark J Rice  
Pacific Northwest National Laboratory  
mark.rice@pnnl.gov

Shuangshuang Jin  
Pacific Northwest National Laboratory  
shuangshuang.jin@pnnl.gov

Abstract

As the power grid technology evolution and information technology revolution converge, power grids are witnessing a revolutionary transition, represented by emerging grid technologies and large scale deployment of new sensors and meters in networks. This transition brings opportunities, as well as computational challenges in the field of power grid analysis and operation. This paper presents some research outcomes in the areas of parallel state estimation using the preconditioned conjugated gradient method, parallel contingency analysis with a dynamic load balancing scheme and distributed system architecture. Based on this research, three types of computational challenges are identified: highly coupled applications, loosely coupled applications, and centralized and distributed applications. Recommendations for future work for power grid applications are also presented.

1. Introduction

Today’s power grids are undergoing a revolutionary transition with fast development in the areas of advanced power grid technology and information technology. The emerging power grid technologies cover the four main areas of the power system: generation, transmission, distribution and end-users, including renewable energy generation, plug-in hybrid vehicles, distributed generation and smart loads. The information technology revolution is represented by a wide-area deployment of smart sensors and meters. With these new technologies, the grid is transitioning from the traditional one-way flow of electrons to a two-way flow of both electrons and information, which results in a more complex power grid network, of increased model size, uncertainty brought by the penetration of distributed resources and smart loads, as well as a large amount of information data across the whole infrastructure. This unprecedented transition builds a technical barrier to future power grid analysis and operation: how to significantly increase the computational speed for power grid analysis in the future power grid environment to allow operators to quickly gain wide-area situational awareness, and manage the power grid more securely and more efficiently.

The analysis tools employed in today’s power grid operations are mostly on serial computers. This implementation creates a bottleneck for understanding power grid characteristics in real-time, and limits responsiveness in the adverse situations. High-performance computing (HPC) techniques, as well as new modeling methods and algorithms, are critical to overcome this technical barrier.

Section 2 of this paper highlights the mathematical base of general power grid computation. The research results of parallel state estimation, parallel contingency analysis, and distributed systems architecture are presented in Sections 3, 4, and 5, respectively. Section 6 describes three types of computational challenges identified based the authors’ research. Section 7 concludes this paper.

2. General mathematical view of power system computation

In general, power system computation can be categorized as steady-state and dynamic analysis. Steady-state analysis is based on a time-invariant model, such as a power flow model. Steady-state analysis determines the status of a power grid at a certain time stamp, without considering the transition between different time stamps. Dynamic analysis is based on dynamic models of certain dynamic devices.
in the power grid, such as generators and controllers, as well as the outputs of a static model. Unlike static-state analysis, which only captures the system information at a fixed snapshot, dynamic analysis is able to identify the transition pattern of a power system between each snapshot, giving a much finer details between each time interval.

Mathematically, when we consider both steady-state and dynamic analysis, the underlying mathematical theory includes matrix algebra, algebraic equations, differential equations and optimization methods. For example, the power flow analysis and state estimation require solving a set of algebra equations, dynamic simulation solves a set of differential-algebraic equations, small signal stability analysis involves eigen-analysis problems, and market applications often need optimization techniques.

Today’s power system operation is largely based on steady-state analysis. The functional structure of today’s power system operation can be found in Figure 1[4]. Sensor data are collected by telemetry and communication equipment with network topology information are sent into a state estimator — the core of power system monitoring and operations. The outputs of a state estimator (a full set of estimated states of a power system at current status) provide the inputs for other key functions in the control center, such as contingency analysis, optimal power flow, economic dispatch, and automatic generation control.

Figure 1: Functional structure of today's power system operations

With conventional techniques in control centers, the state estimation and contingency analysis are normally computed and updated in an interval of minutes, which is not fast enough to recognize and anticipate system status if there is an emergency. In the meantime, the contingency cases conducted today are normally a small set of pre-selected cases, which might miss some critical contingencies, resulting in an incomplete representation of the system.

In addition, today’s state estimation and contingency are conducted within each local area, such as a balancing authority. While considering outside areas as some equivalent nodes, little information exchanged between each balancing authority, and between balancing authorities and the reliability coordinator. We expect that more information will be exchanged in the future power grid, which will lead to a much larger problem size and longer computational time. A distributed state estimation structure is considered to be suitable for future state estimation.

In order to examine the computational challenges for power grid operation, we selected two key functions of state estimation and contingency analysis to start our HPC implementation efforts. We implemented parallel state estimation using preconditioned conjugate gradient methods, parallel contingency analysis with a counter-based dynamic load balancing scheme, and also studied distributed system architecture to support de-centralized power grid applications. The next three sections will discuss these three areas in detail, respectively.

3. Parallel state estimation

This section describes the mathematical basis of weighted-least-square (WLS) state estimation, the solver method of sparse linear equations, as well as our research results of parallel state estimation utilizing both direct and iterative methods. The direct method was implemented on a shared-memory machine with the integration of the SuperLU solver package [13]; the iterative method, i.e., Preconditioned Conjugate Gradient (PCG), was implemented on both a shared-memory computer and a cluster machine. Some experiments were conducted to compare the performance between each method.

3.1. Weighted-least-square state estimation

As described in Section 2, power system state estimation is a central component in power system Energy Management Systems. It collects system field measurements from the supervisory control and data acquisition (SCADA) system, estimates system states, and generates critical inputs for other power system operational tools.

The WLS state estimation is the most widely used state estimation algorithm. Mathematically, the WLS state estimation algorithm of an n-bus power system with m measurements can be described below:

\[ \min J(x) = \sum_{i=1}^{m} R_i^{-1}(z_i - h_i(x))^2 \]
\[
(z - h(\hat{x}))^T R^{-1} [z - h(\hat{x})] = 0
\]  
(1)

Subject to: \( z = h(x) + e \)

\( z \) is a measurement vector of dimension \( m \), \( x \) is a system state vector of dimension \( 2n-1 \), \( R \) represents the covariance matrix of the measurement noise, \( h(x) \) is a vector of non-linear functions relating states vector \( x \) to measurement vector \( z \), and \( e \) is a vector of measurement errors of dimension \( m \).

To minimize the object function of \( f(x) \), equation (2) has to be satisfied:

\[
g(x) = \frac{\partial f(x)}{\partial x} + R^{-1} [z - h(x)] = 0
\]  
(2)

where

\[
H(x) = \frac{\partial h(x)}{\partial x}
\]

By neglecting the higher order terms of the Taylor series of \( g(x) \), the following equation (3) is obtained:

\[
x^{k+1} = x^k - [G(x^k)]^{-1} g(x^k)
\]  
(3)

where \( k \) = the iteration index

\( x^k \) = the solution vector at iteration \( k \)

\[
G(x^k) = \frac{\partial g(x^k)}{\partial x} = H^T(x^k) * R^{-1} * H(x^k)
\]

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

Therefore, the WLS state estimate can be found by iteratively solving the following equation

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

Equation (4) can be simplified to a linear equation of (5)

\[
A \Delta x = b
\]  
(5)

where \( A = G(x^k) \), \( \Delta x = (x^{k+1} - x^k) \), and \( b = H^T(x^k)R^{-1}[z - h(x^k)] \)

The \( A \) matrix is a sparse symmetric positive-definite (SPD) gain matrix. This linear equation (5) is being solved in each iteration of WLS state estimation.

There are mainly two types of methods for solving this kind of linear equation: directed methods and iterative methods. Direct solution methods mostly perform an LU/LDU [15] or Cholesky factorization [3] of the matrix and try to reduce computational cost by minimizing fill-ins, followed by the backward and forward substitutions. Typically, the direct method includes four stages: pre-ordering, symbolic factorization, numerical factorization, and forward/backward sweeps. A state-of-the-art direct solver package is SuperLU [13]. 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 (0.0001 in our experiments). The typical iterative method includes Jacobi [12], Gauss-Seidel [7], Successive Over Relaxation [23], and the Conjugate Gradient (CG) method. The CG method is one of the best iterative methods for solving sparse SPD equations [9]. It theoretically yields exact solutions within at most \( N \) iterations, where \( N \) is the dimension of the linear equation. Normally, the condition number of the \( A \) matrix in equation (5) is a large number for a practical power system problem, which results in an expensive computational cost required for convergence. By applying the inversion of a preconditioner matrix \( P \) at the both side of equation (5), the equation (5) becomes

\[
P^{-1} A \Delta x = P^{-1} b
\]  
(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 equation (6). This is called the PCG method. It is suitable for implementation on parallel computers. The detailed algorithm of a PCG method can be found at [20].

The parallel state estimation work has been conducted by many researchers [1][2][6]. Our parallel state estimation work is focused on solving equation (5) faster.

### 3.2. SuperLU vs. shared memory based CG

A multithreaded version of SuperLU 3.0 SuperLU and parallel PCG with a Jacobian preconditioner (diagonal scaling matrix) were implemented on a SGI Altix machine, shared memory architecture with software multi-threading. Shared memory architectures are known to be able to perform well on applications with irregular communication patterns [24]. The SGI Altix 3000 [22] is located at the Pacific Northwest National Laboratory. It has 128 1.5GHz Itanium-2 processors, 256GB memory and runs a Linux 2.4.21 kernel. The SuperLU solvers were compiled with the Intel C++ and Fortran Itanium compiler version 7.1. In the SuperLU package, the version of “\( MMD A'A' \)” algorithm was implemented. The “\( MD A'A' \)” means Multiple Minimum Degree permutation re-ordering scheme applied to \( A'A' \). A power system with 1177 buses, 1770 lines, and 6144 measurements was used to evaluate these two algorithms. The focus is on the computational time and speedup of solving the equation (5) in one iteration of the Newton-Raphson procedure.

The execution time of the “\( MMD A'A' \)” algorithm in SuperLU and parallel PCG method with a Jacobian preconditioner is compared in Figure 2. The parallel
PCG method outperforms “MMD $A'A$” algorithm in both scalability and absolute execution time. In particular, the execution time increases with SuperLU running on more than two processors [11].

In order to evaluate the PCG algorithm for a large system, a large model derived from the Western Electricity Coordinating Council (WECC) planning model was used. This model contains 14084 buses, 16230 lines, and 46544 measurements. The performance of the PCG algorithm on the SGI Altix 3000 can be found in Figure 3. With our implementation, the solution time for one iteration is about five seconds with 16 processors.

In order to further reduce the overall computational time, a cluster-based PCG algorithm is implemented on a cluster-based NWICEB machine. The NWICEB machine is located at the Pacific Northwest National Laboratory. It contains 192 nodes and two 2.33 GHz quad-core Xeon E5345 processor per node. The size of memory per node is 16GB and the interconnection is Gigabit Ethernet and Infiniband. The implementation of the cluster-based PCG algorithm utilizes the Hypre library package [10], which contains several preconditioners for solving very large sparse linear systems with good scalability. The main preconditioners we studied with the Hypre library package are the Jacobian preconditioner, ParaSails preconditioner, and the Euclid preconditioner. ParaSails is a parallel sparse approximate inverse preconditioner for the iterative solution of large, sparse systems of linear equations [20]. The Euclid is a family of Incomplete LU methods for sparse linear systems [5]. The Euclid preconditioner is based on a parallel incomplete LU decomposition algorithm. The comparison of execution time and speedup for the first iteration of the state estimation Newton-Raphson procedure, with all these three preconditioners, are shown in Figure 4 and Figure 5, respectively. The speedup is calculated by dividing the execution time with single core and a Jacobian preconditioner by other execution time with a different preconditioner and/or a different number of cores.

3.3. Cluster-based PCG

The test results show that, with the help of good preconditioners, the computational time can be 10 times less than with the simple Jacobi preconditioner. On 16 processors, both the Euclid preconditioner and ParaSails precondition take around one second to solve the first iteration of the WLS state estimation problem. The Euclid preconditioner also demonstrates better scalability in solving the problem as shown in Figure 5.
Currently, a full state estimation package has been implemented using the PCG method. It can achieve five-second solution time for the full state estimation problem of the WECC 14,084-bus system, which is comparable with today’s SCADA cycles.

![Figure 5: PCG speedup comparison with different preconditioners on the NWICEB for a WECC 14084-bus system (first iteration)](image)

4. Parallel contingency analysis

Contingency analysis is another critical component to help determine the impact of potential equipment failures by considering “what-if” situations: i.e., what would the power grid status be if one or more pieces of equipment, such as power plants or transmission lines, are lost? It is used to assess the ability of the power grid to sustain various combinations of component failures based on state estimates. The outputs of contingency analysis, as well as other EMS function outputs, provide the basis for situational awareness and the design of preventive and corrective actions. It is also widely used in today’s power market operation for feasibility test of market solutions.

Due to the large size of contingency cases and heavy computation involved, contingency analysis is currently limited to be selected “N-1” and/or “N-x” contingency cases (where x ≥ 2). A common industrial practice is to solve 500 contingency cases in a time interval of five minutes. The current North American Electric Reliability Corporation transmission operation standard TOP-004-02 requires that the loss of most severe single contingency, or multiple outages in the power grid, specified by its Reliability Coordinator [19], should not cause system instabilities. A limited set of “N-1” cases may not be adequate to assess the vulnerability of today’s power grids due to new development in power grid and market operations, resulting in an incomplete picture of full system status.

In the meantime, preventing and mitigating blackouts requires “N-x” contingency analysis for better understanding of system behaviors. Therefore, from a system reliability point of view, there is a great need for a massive number of contingency cases to be analyzed. From the power market prospective, the same statement holds true. For example, Financial Transmission Rights (FTR) is a function to provide market participants a means to hedge risks due to power transmission congestions. The feasibility of the FTR solution has to be evaluated by contingency analysis (as constraints). When multiple FTR types (annual, seasonally, monthly) are considered coupled with contingency analysis, the number of contingency cases increases quickly. Parallel contingency analysis is the solution to accelerating power grid contingency analysis for improving system reliability and power market applications.

Due to its nature, contingency analysis is inherently a parallelizable process because contingency cases are relatively independent (loosely coupled) of one another; the communication between each processor is minimal. Therefore, cluster-based machines are suitable for implementing a parallel contingency analysis algorithm. In our experiment, computers – Colony2A and HP MPP2 – were used to test different load balancing schemes. Colony2A has 24 Itanium-2 computer nodes from Hewlett Packard. Each node has two 1.0 GHz processors, 6 GB memory, and 36 GB disk space. The network protocols include Myrinet-2000, Infiniband, Ethernet and GigE on all nodes. The MPP2 machine consists of 980 Hewlett-Packard Longs Peak nodes with dual Intel 1.5 GHz Itanium-2 processors. FatNodes with 10 GB of memory (5 GB/processor) and 430 GB of local disk space and ThinNodes with 10 GB of memory (5 GB per processor) and 10 GB of local disk space. A single rail QSNetII/Elan-4 interconnect from Quadrics is used to obtain fast inter-processor communication.

Mathematically, there is a relatively straightforward parallelization path, however, the computational challenge is not the low level parallelization algorithm, but a computational load balancing scheme to achieve the evenness of execution time for all processors. A well-designed computational load balancing scheme of massive contingency analysis is key to improve the performance of parallel contingency analysis.

4.1. Static load balancing scheme vs. dynamic load balancing scheme

The most straightforward load balancing scheme is to pre-assign an equal number of contingency cases to each processor. However, due to different execution
time for each contingency case, the computational resource is not fully utilized because many processors might complete their tasks earlier while waiting for other processors to finish their work.

A better load balancing scheme is to allocate tasks dynamically to each processor based on its availability – a dynamic load balancing scheme. In other words, a processor requests the next contingency case when it completes the assigned task so that the cases are more optimally allocated in terms of execution time, even the number of cases assigned to each processor might be different.

Our implementation of the dynamic load balancing scheme is based on a shared variable (task counter) updated by an atomic fetch-and-add operation. Even the dynamic load balancing scheme can bring some overhead of managing the task counter, the overall performance is much better than the one with simple static load balancing scheme. A study of 512 “N-1” contingency cases and full “N-1” cases (20094) of a 14084 WECC system on the Colony2A machine is shown in Figure 6 [11]. It is clear that the dynamic scheme shows better linear scalability.

Figure 6: Performance comparison of static and dynamic computation load balancing schemes with a WECC system: (a) 512 contingency cases; (b) full (20094) contingency cases

4.2. Massive contingency analysis with a dynamic load balancing scheme

In order to test the dynamic load balancing scheme with a large number of processors, three scenarios are tested on the MPP2 machine with 512 processors. There three scenarios are full “N-1” contingency cases (20094 cases), 150K N-2 contingency cases, and 300K “N-2” contingency cases. The results, including computational time, disk I/O time, and counter updating time, are summarized in Table 1. The test results indicate excellent scalability of the dynamic load balancing scheme, especially with a large number of cases. With this implementation, full “N-1” WECC contingency analysis can be completed within half a minute.

In addition, the counter time is very insignificant compared to the computation time. It indicates that the implementation of the dynamic balancing scheme using a counter adds very little overhead to the overall process, and good scalability can be ensured. The speedup increases as the number of cases increases.

Table 1: Summary of the massive contingency analysis with dynamic load balancing scheme on the MPP2 machine using 512 processors

<table>
<thead>
<tr>
<th>Number of cases</th>
<th>Wall clock time (s)</th>
<th>Total computation time (s)</th>
<th>Total I/O time (s)</th>
<th>Total counter time*</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>20,094</td>
<td>31.0</td>
<td>14235.2</td>
<td>82.7</td>
<td>0.899</td>
<td>462</td>
</tr>
<tr>
<td>150,000</td>
<td>187.5</td>
<td>93115.5</td>
<td>489.1</td>
<td>5.550</td>
<td>503</td>
</tr>
<tr>
<td>300,000</td>
<td>447.9</td>
<td>226089.8</td>
<td>1087.1</td>
<td>9.984</td>
<td>507</td>
</tr>
</tbody>
</table>

* Includes waiting time.

5. Distributed Systems Architecture

The goal of the distributed systems architecture is connecting distributed power application models, each running on HPC platforms, to efficiently exchange necessary raw data and intermediate processed results. It provides a high level interconnection between each remote control center. A reference architecture is depicted in Figure 7 that demonstrates the elements integral to the architecture.

The bottom layer represents the peer level control centers that can exchange data such as measurement data between external models for state estimation. The
data source of each control centers can be uniquely identified through an endpoint. The endpoint can be a unique URL or an entry to a data repository.

The upper layer is the middleware and underlying data communication infrastructure that deliver the data from the data sources to the power models running on HPC nodes.

The top layer is the parallel code running for power models. Using message passing interfaces (MPI), the parallel power model has a master processor to fetch the data for computing, partition the data, and dispatch the partitioned data to the worker processors running the computing tasks in parallel. Hence, only the master node needs to explicitly invoke the interface as a client to the remote site. We refer to the master node as the communicator node and the rest as application nodes.

The same structure extends to distributed hierarchical power models: for example, distributed state estimation, within this distributed systems architecture.

![Figure 7: A reference distributed systems architecture for the power grid](image)

To achieve the architecture goal, the following architecture issues need to be addressed.

The communication needs of geographically distributed parallel models entail interfaces to the data management infrastructure. These models are usually implemented using a parallel computing programming paradigm such as MPIs. Application programming interfaces (API) such as `MPI_send` and `MPI_receive` are used to communicate across nodes clustered locally. However, for an instance of the distributed state estimation, communication across the local MPI boundary is necessary to exchange monitoring data or analysis results between remote control centers.

The interface needs to wrap the connections to middleware in the distributed environment. Middleware plays a key role in the delivery of operational status information of the power grids. Two ongoing examples are NASPInet [16] and GridStat [8]. To support real-time control and operations of power grids, the data communication across sites require dedicated middleware that can handle high throughput and large amount of sensor reading data in the power grid. An interface layer between the MPI code and the middleware should include middleware clients to identify the data source location, establish the connection, and send/receive data.

The interface layer devises key components of the data processor and data buffer as the middleware clients. The data processor resolves the location of data demanded (either from the local region or from remote sites) from a centralized data registration service. The data processor then uses the location information to connect to the corresponding server and fetch the data through the middleware. Depending on the protocol that the middleware supports, the data processor needs to extract the required fields of data (such as bus voltage, phase angle) and assemble them as inputs to the parallel power models.

Hence, the interface layer loosely couples the parallel code running on HPC nodes and the data communication. This benefits distributed power models in a heterogeneous distribution environment as middleware of different products or purpose can be deployed across the data communication network. In that case, only the middleware clients need to be changed and repacked according to the middleware APIs without modifying the parallel code.

From quality of service (QoS) aspect, the architecture needs to support efficient data communication to achieve QoS including optimizing the latency of data delivery, coordinating the disparity between the computation delay and the data inputs interval, and handling the quantity of data exchanged. This indicates it is essential to reduce the overhead incurred by the extra layer introduced between the HPC code and the middleware.

6. Computational challenges

The HPC-enhanced state estimation, contingency analysis, and distributed systems architecture could make a significant impact on power grid operation and planning. Based on our research on the work of parallel
state estimation, parallel contingency analysis and distributed computer architecture, we identified three types of computational challenges for power grid operation: highly coupled equations, loosely coupled tasks, and centralized computation vs. distributed computation.

6.1. Algorithm selection

Before we start to solve a problem with the help of HPC, the selection of a suitable algorithm is critical to success. Traditionally, software has been written for serial computation, which means a problem is broken into a discrete set of computer instructions that are executed one after another, and only one instruction may execute at any moment in time on a single processor. With parallel computing, the problem is broken into discrete parts that can be solved concurrently; each part is broken into instructions, which can be executed simultaneously on multiple processors. The algorithms that are optimized for and work well with sequential computing may not work well with parallel computing.

A key consideration for the selection of an algorithm for a linear system is its structure. Generally, direct algorithms, such as SuperLU package, are good for dense matrices, where iterative methods, such as PCG and generalized minimal residual method (GMRES) [21], are good for large and sparse matrices. For most power system applications, the matrices are very sparse and iterative methods normally work well. The performance and robustness of preconditioned iterative methods is sensitive to the choice of preconditioner and solver parameters. Once a good algorithm is selected, we can concentrate our efforts on other computational challenges.

6.2. Highly coupled equations

The parallel state estimation study in Section 3 provides a good example of an application utilizing HPC techniques for highly coupled equations in power system analysis. In general, mathematical algorithms and computational issues are essential for studying coupled equations. These include linear and nonlinear solvers, as well as preconditioners for coupled equations. There are other applications in the power system, such as eigenvalue analysis and dynamic simulation, which complement but have significant differences.

To solve highly coupled equations with the HPC techniques, strong programming techniques also play an important role. This includes the partitioning of matrices and the communication and synchronization among processors. Strong scaling is very important while being power efficient.

6.3. Loosely coupled tasks

For loosely coupled tasks, such as parallel contingency analysis, the key is to optimally distribute the computational load to all processors based on the computational time, allowing the processors to be utilized efficiently. Different calculations can be performed on either the same or different sets of data. Task parallelism does not usually scale with the size of a problem. The counter-based dynamic load balancing scheme provided in Section 4 could serve as a general framework for the tasks included in this category.

When there are many tasks over many processors, a single counter scheme might not be good enough because counter congestion might occur when there are many requests at the same time. This congestion will increase the overall computational time. In this case, a multiple-counter dynamic scheme can be used to minimize counter congestion. This scheme divides processors into different groups. Each group has its own counter. An equal number of tasks are pre-allocated to each group. Inside each group, the dynamic load balancing scheme is applied. Once the tasks are finished in one group, the counter in this group can steal remaining tasks from other groups until all tasks are completed. This implementation can help to reduce the potential counter congestion.

A load balancing scheme should be the lowest possible level to archive maximum speed-up for loosely coupled tasks.

6.4. Centralized vs. distributed computation

Our research on distributed systems architecture begins an effort to support distributed power system applications from a system architecture point of view. The architecture allows for computing to continue even if one computer has a failure. In general, distributed computation uses multiple individual computers in different resources to perform the task while centralized computation is controlled through a central terminal computer/server, which provides control, computation and storage. Using state estimation as an example, centralized state estimation requires a longer execution time due to the large amount of measurement data, the large size of the system, and much higher complexity. Distributed state estimation distributes the computational efforts to multiple local state estimators who exchange information between them. One advantage of distributed state estimation is better accuracy. In addition, the introduction of
synchronized phase measurement data collected from different locations facilitates distributed state estimation. However, the setup of distributed computation is quite complex compared to the single server it replaces in a centralized location due to communication and coordination of the data flow between the individual computers. How to partition subsystems and manage the data flow is the main computational challenge for distributed computation.

7. Conclusion and future work

In this paper, some high-performance computing research outcomes in the areas of parallel state estimation using the PCG method, parallel contingency analysis with a dynamic load balancing scheme and distributed system architecture are presented. A full state estimation package has been implemented using the parallel PCG method which achieves a five-second solution time for the full state estimation problem of a 14,000-bus WECC-size system. A parallel contingency analysis package with a counter-based dynamic load balancing scheme can obtain a nearly linear speed-up. A distributed system architecture, which is suitable for future distributed large scale power grid applications, is proposed.

Based on our research, three types of computational challenges are identified: highly coupled equations, loosely coupled tasks, and comparison between centralized computation and distributed computation.

The power grid is evolving into a more dynamic, probabilistic, and complex system. In addition, the information revolution provides opportunities in operating and managing the future grid. This transition involves a large amount of data, more complex power system modeling, and more uncertainty issues brought by renewable energy and smart load. During these transitions, other challenges also require our attention. Some examples of these challenges are a multi-scale parallelism algorithm considering fault-tolerant communication, data movement (caching and replication schemes), system I/O balancing, new uncertainty quantification methods with broad impact on both generation and load, and post-processing/visualization of a large amount of data.

8. Acknowledgement

This work is supported in part the Advanced Grid Analytics Program funded by the U.S. Department of Energy and the Future Power Grid Initiative supported by the Pacific Northwest National Laboratory. The Pacific Northwest National Laboratory is operated by Battelle for the U.S. Department of Energy under Contract DE-AC06-76RL01830.

9. References

[18] Nieplocha, J., A. Marquez, V. Tipparaju, D. Chavarria-Miranda, R.T. Gutromson, and Z. Huang, “Towards Efficient Parallel State Estimation Solvers on Shared...


