Toward a reliable, secure and fault tolerant smart grid state estimation in the cloud

Ketan Maheshwari*, Marcus Lim*, Lydia Wang*, Ken Birman*, Robbert van Renesse*
*Department of Computer Science, Cornell University, Ithaca, NY, USA

Abstract—**The collection and prompt analysis of synchrophasor measurements is a key step towards enabling the future smart power grid, in which grid management applications would be deployed to monitor and react intelligently to changing conditions. The potential exists to slash inefficiencies and to adaptively reconfigure the grid to take better advantage of renewables, coordinate and share reactive power, and to reduce the risk of catastrophic large-scale outages. However, to realize this potential, a number of technical challenges must be overcome. We describe a continuously active, timely monitoring framework that we have created, architected to support a wide range of grid-control applications in a standard manner designed to leverage *cloud computing*. Cloud computing systems bring significant advantages, including an elastic, highly available and cost-effective compute infrastructure well-suited for this application. We believe that by showing how challenges of reliability, timeliness, and security can be addressed while leveraging cloud standards, our work opens the door for wider exploitation of the cloud by the smart grid community. This paper characterizes a PMU-based state-estimation application, explains how the desired system maps to a cloud architecture, identifies limitations in the standard cloud infrastructure relative to the needs of this use-case, and then shows how we adapt the basic cloud platform options with sophisticated technologies of our own to achieve the required levels of usability, fault tolerance, and parallelism.**

I. Introduction

Accurate tracking of power grid state using Synchrophasor Measurement Units (PMU) [1], [2] devices is widely expected to play a significant role in the advancement of the smart grid [3], [4] in the coming years. Relative to past ways of monitoring grid state, synchrophasor technology makes possible significant improvements in the quality and timeliness of state estimation: it becomes reasonable to talk about obtaining estimates within delays (from data acquisition to availability of the new estimate) that could be as low as sub-seconds in a moderate-sized regional power grid [5].

To realize these objectives, several steps are required: a wide-scale deployment of PMU devices, the definition of standards for collection and archiving of data, and the creation of applications capable of analyzing this data and tranforming it into useful, actionable intelligence. Our work targets the first steps in this spectrum: those involved in collecting data, archiving it, and supporting computation upon it. Although we are interested in a variety of styles of analysis, for reasons of brevity this paper focuses on support for state estimation: the fitting of measured data against a model of the grid topology and its electric properties.

For many anticipated uses, state estimation is a time-critical operation. Timely visualization of the state of the smart grid aids controllers and observers in assessing the evolution of the grid state over time and helps them make decisions to ensure safety of devices and stability of power production and distribution rate. Protection mechanisms can be designed that compare observations with recent state estimates, tripping lines if sharp deviations occur. As we integrate renewables into the grid, real-time state estimation could be a key to managing the resulting structure.

Any substantial deployment of high-resolution synchrophasor devices can generate substantial quantities of data [6]. A monitoring capability suitable for wide-area deployments thus requires a large-scale infrastructure for collecting and communicating data, and must have access to flexible computational power, network bandwidth, and storage capacity. The distributed nature of data sources, the possibility that data may need to be collected from multiple (competitive) power producing and transport enterprises, and this need for timely state estimation all make matters more complicated. Fault tolerance, security, and reliability in data movement and computation are other key issues.

A variety of computing infrastructures [7] exist today which match this need, offering a range of qualities of service and scales of economy. These include dedicated clusters, on-premise networks of workstations, and cloud computing infrastructures available in commercial and academic settings, as well as in privately operated configurations. Clouds are best understood as virtualized pools of resources hosting a variety of applications in a manner that isolates each user's data and computation from unwanted interference with activities of other users. The virtualization structure presents the illusion that each application runs over a private distributed backbone network and computes within a private, dedicated infrastructure [8]. The cost efficiencies of sharing permit high utilization levels for the underlying hardware, which can also be administered, powered, and cooled in ways that leverage the dense packing and relatively steady load levels. The overall effect is to drive costs down significantly.

In this paper we describe the development of a cloud-based software framework aimed at supporting smart grid computations and data storage for applictions such as state estimation. Our decision to use cloud-based approach brings challenges, and our contributions center on the ways that we have overcome those issues, and on the performance and scalability of the resulting solutions. For example, the cloud security model works well for transactions with web sites, but is less well matched to the mission-critical security needs.

We overcome this issue by using a public key infrastructure; any grid-related data that transits the web into our cloud infrastructure is encrypted for confidentiality. In a similar way, we have identified and resolved issues of fault-tolerance, erratic timing, and other problems specific to the cloud.

For reasons of brevity, this paper limits itself to a discussion of a pipeline we have implemented that emulates synchrophasor data capture, state estimation, and visualization in a realistic but controlled network setting. We use experimental results to perform trade-offs between timeliness and completeness of results. We take a series of sophistication measures on the techniques employed and report how they affect the performance. We make no effort to improve or optimize the state estimation software used in this experiment, although we believe that a substantial opportunity to do so exists. Interested readers are referred to [9] for an early treatment of the subject and to [10] for a recent treatment based on synchrophasor measurements. A detailed architectural and algorithmic treatment is presented in [11] and [12] respectively. Our algorithm is a mock version of the one described in [12].

To summarize, key contributions of our **grid-cloud** infrastructure are:

1) A software framework that supports execution of state estimation workflow while leveraging internet and cloud standards;
2) A demonstration that highly assured and timely data availability can be provided even in the wake of sporadic network or cloud node failures and/or delays;
3) Concurrent data processing and visualization of the synchrophasor data captured from scattered sources;
4) Flexible and elastic scaling (up/down) of the infrastructure based on dynamically added or removed PMUs;
5) Economy of computation by maximally using the cores and bandwidth available in the cloud.

II. Related Work

There has been considerable prior work relevant to our effort, but none that matches our proposed architecture. Our work touches on a number of areas of prior study: (a) Distributed and cloud computing, (b) Wide-area and distributed computations and data handling in smart power grid, (c) Asynchronous and parallel execution of application workflows, and (d) Development and usage of a simulated smart grid environment.

Recent years have seen an increase in employing distributed computing principles and practices to applications from a wide array of scientific and engineering domains [7]. Distributed computing principles have been employed in the general area of power transmission, distribution and related communications in the early days [13], [4] as well as the recent past [14]. Clouds have been a relatively recent model of distributed computing. Owing to their flexibility, cost effectiveness and high availability, clouds are enjoying much success as computational and data management infrastructures for large-scale applications. However, not much attention has been paid to the potential benefits they could bring to smart grid operations. The work described in [15] is the closest treatment of steering smart grid computations into the clouds.

Smart grid is a catch-all term for a wide range of digital technologies aiding the power grids. Consequently, a number of applications have evolved over time taking advantage of growing computational power supporting various operations. Wide-area observation and monitoring has been a key application of synchrophasors in the US. Representative examples are the "FNET" (*fnetpublic.utk.edu*) [16] and UC-CIEEE (*uc-ciee.org/electric-grid*) [17] implementation for the east and west coast grid sections respectively.

There has been some studies relying on simulating or emulating the power grid conditions in order to study its behavior [18], [19]. In the present work, we favor this approach and use simulated PMU data based on the standard IEEE.c37.118 format [20], [2].

III. Application Characteristics and Requirements

In this section we characterize the application and present requirements emerging from this characterization. Of interest are two kinds of requirements: (1) functional requirements, which dictate the structure of the solution, and (2) performance and scalability requirements of the application, which can be characterized both qualitatively and quantitatively. The basic workflow is shown in figure 1. The boxes show major computational stages. The directed lines between boxes show the flow of data from a computational stage to the next. For instance, measurements (from PMUs) are generated from the actual state of the smart grid. The state estimator computes state based on PMU measurements, and visualization shows the estimated state to the users for monitoring purposes, and control applies control action based on the state estimates and this, in turn affects the state of the power grid. Combined, this yields a closed-feedback data and control flow loop.

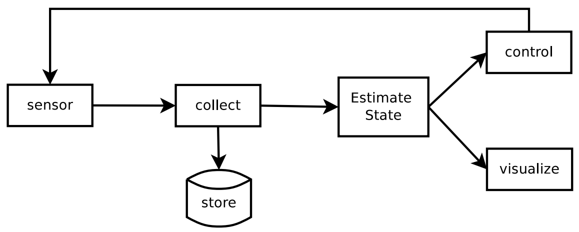


Fig. 1. A logical representation of smart grid state estimation workflow

A. PMU

Compared to older SCADA (Supervisory Control and Data Acquisition) techniques used in the power grid, PMU data can reveal much more information at a high resolution which could be used for timely state estimation of an aggregated portion of a smart grid power distribution [21]. The high-quality data produced by synchrophasor can be used to efficiently estimate the state of grid and to mitigate risks such as device damage and blackouts in a timely fashion.

PMUs are GPS-synchronized through onboard atomic clocks on the GPS satellites. The increased frequency and accuracy of measurements achieved by modern synchrophasors enables a wide variety of new applications, but also implies that the amount of data generated has increased manifold. PMUs operate at 30 Hz, and measurements include the 1st (and optionally 2nd) derivatives. A wide-scale deployment of synchrophasor technology, including thousands or tens of thousands of PMUs, will enable many applications in the smart grid: monitor, control, estimate, analyze, and trade. The scale of such a deployment, however, suggests strongly that a cloud infrastructure may be inevitable [6].

B. State Estimation

State estimation [22] is an essential element of the modern smart grid. It takes as input the grid topology and a collection of measurements, and attempts to remove measurement errors and provide estimates of missing measurements. State estimation is computed by the least squares solution to the following linear equation:

$$y = A.x + \epsilon \tag{1}$$

where y is an observation matrix representing the measured power quantities, x is the true state matrix, A is a matrix with more rows than columns and ϵ is a matrix representing errors in measurement. The term $A.x$ represents an assumption that the real state is a function of observed state.

Power instruments and lines operate optimally and durably under a predefined range of state parameters, and state estimation can be used to monitor and mediate risks by permitting the discovery of conditions that might violate these optimality and safety characteristics. A variety of possible remedies can then be explored: human intervention, tripping relays, or even some form of computer-mediated response. It should be noted that today's grid includes few mechanisms that might be viewed as closed-loop monitoring and control structures, in which computing systems act directly without a human operator in the loop. Nonetheless, rapid state estimation has immediate value for grid operators and will enable control automation.

C. Synchronization and Latency

State estimation is a time-critical application. In order to determine the state reliably, the data from scattered PMUs must reach the state estimator in a time synchronized manner. Latency of the connection between the PMU and state estimator directly affects this. In addition, the clock synchronization accuracy across these PMUs affects the data reaching state estimators. For this reason, PMUs must be synchronized at sub-millisecond accuracy. Overcoming network and computational delay is thus a key goal in our work.

D. Fault Tolerance, High Assurance, and Security

Our cloud-focused approach exposes us to a variety of failure and security challenges. These start at the lowest levels: as the power community and the NASPI consortium has noted, standard TCP connections are not very well suited to synchrophasor data transmission because high-latency can occur on lossy links [5]. Consequently, a mechanism is required which ensures uninterrupted and fast data transmissions in the wake of poor latencies or fault conditions in TCP connections.

Since data is moving over insecure connections across the web, a security mechanism is required to mitigate unauthorized data injection and other types of attacks. Suitable techniques such as encryption and firewalls are required in order to address the information security issues.

Once data reaches the cloud, we confront challenges emerging from the very mechanisms that make the cloud so cost effective: nodes are *packed* to optimize use, and as a result can become overloaded and slow. Failure is surprisingly common, in part because cloud management systems often reboot computers that seem balky. Thus our solution needs to replicate data and computation as a remedy for an expected high rate of failures.

E. Data Storage and caching

Some applications require that measurements be stored for "historical" computations. Each PMU produces close to 500 Kb of data per minute. Thus, a modest scale of 10 PMUs running for one day will result in a historical data of $500 \times 10 \times 24 \times 60$ or 7.2 Gb. Our architecture uses a threefold replication to overcome failures and delays, thus further increasing the volumes of data to be managed. A smart storage strategy is required. One approach is to cache the data in memory as it arrives and haul it to disk in optimal sized chunks in a concurrent thread.

F. Visualization and Control

The state estimation results are streamed back to visualization hosts. Manual and automated controllers use the visual evolution of the grid for regulatiion and historical analysis. The visualizer must show the time of measurement and should be tunable to view individual elements.

IV. Cyberinfrastructures

Our decision to work with cloud technologies is a natural response to the scale and economics of the task at hand, but not the only option. The question is how to make best use of available resources lying in different domains, with a variety of access methods and ever changing quantities and availabilities. The cloud model of computation is especially favorable for smart grid applications since the computational tasks are not batched and hence there is no waiting time. This enables a time-critical model of computation. Furthermore, a pay-as-you-go scheme of resource utilization leads to economy of scale. We can leverage the ad hoc and elastic nature of clouds to benefit the smart grid at the economy of scale expected of cloud computing while efficiently utilizing power as we scale up [23]. Researchers in cloud computing at Berkeley envision that large-scale applications will increasingly adopt this model of computing [8]. Our work of running smart grid computations in the cloud is one such initiative in this direction.

A. Small and Medium-scale Clouds

Cloud computing evokes an image in which Amazon.com's EC2 plays central roles, but in fact this need not be the case. The same virtualization and sharing technologies that make Amazon's cloud (and others) so cost-effective can also be deployed in on-premise or per-enterprise data centers. An example of a small scale cloud is Cornell's Redcloud (*www.cac.cornell.edu/redcloud*) (96 cpus) while that of a medium scale cloud are Open Cloud Consortium sponsored Bionimbus (*www.bionimbus.org*) ($\sim$2000 cpus) and Indiana University led Futuregrid cloud (*portal.futuregrid.org*) ($\sim$5000 cpus). All the aforementioned clouds run Eucalyptus-based management and configuration tools [24]. Our work leverages these platform standards and hence could run as easily on a private cloud of this sort as on a larger shared cloud.

B. Large-scale Clouds

Large-scale clouds of today are capable of providing virtually unlimited resources on demand. These are mostly offered commercially in the form of services to the paid subscribers. The resources are generally billed on a pay-as-you-go basis. The promise of this computing model is flexibility, availability, and cost effectiveness compared to the cost of owning resources. For our work, we've experimented with Amazon's EC2 cloud and associated services (*aws.amazon.com/ec2*). In addition to compute resources, a variety of services are available such as storage and application hosting.

V. Implementation

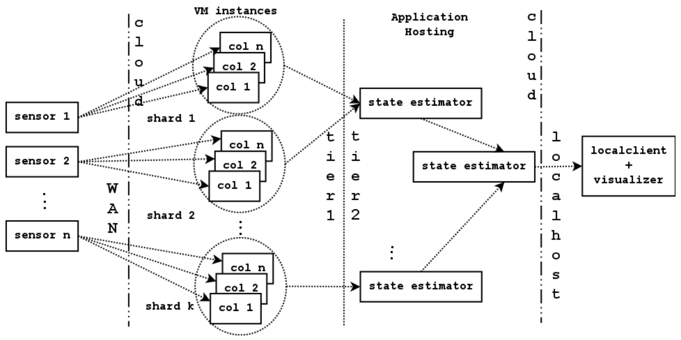


Fig. 2. Implementation of the grid-cloud framework

In this section we describe an implementation of the framework that addresses the application requirements and adapts to its characteristics.

Figure 2 shows the various components of the grid-cloud framework as they are implemented. The virtual PMU sensors on the leftmost side are setup to run on nodes external to cloud. We use Planetlab [25] nodes to run PMU sensors. The data collectors (*col 1, ..., col n*) run on the outermost tier of the cloud where virtual machine instances are elastically obtained. The collectors perform the following three tasks: (a) receive and store the data on tier 1 of cloud, (b) Forward the data to the deeper tiers of the cloud, and (c) record the time of arrival of data. The inner tier compute nodes (often called "app hosts") runs the state estimators in a hierarchical fashion. The state estimators forward the state information to the localhost for live visualization and analysis. The data is encrypted throughout the chain. SSL-based access and strict firewall policies ensure security of the cloud virtual machines.

Upon arrival in the cloud, received data is sharded for scalability and replicated at the data collectors interface in order to achieve highly assured availability and fault tolerance in the wake of degraded or broken TCP-connections. The Isis^2 library runs in the background to monitor this configuration and maintain continuous availability and correct system operation, restarting failed components on new nodes and reconfiguring the system as required.

We briefly describe the tools and techniques used and discuss their utility to the area of cloud computing in general and to the state estimation application in particular.

A. Isis^2

Isis^2 (http://isis2.codeplex.com) is a cloud-based high assurance computing library that we created at Cornell to support a variety of data and computation replication options, optimized for high performance. The system implements a theoretically rigorous model termed "virtual synchrony," for which a substantial body of prior research exists [26]. Prior versions of Isis were oriented towards smaller cluster-style computing settings, but found use in such settings as the New York Stock Exchange, the French Air Traffic Control System, the US Navy AEGIS control system, telecommunications switch control, process control, etc. Isis^2 is a new implementation focused on the cloud, but drawing heavily on insights gained both from our earlier systems, and from research conducted over the past decade on scaling these techniques up.

In GridCloud, Isis^2 plays roles centered on the management and control of the GridCloud platform: it is used to monitor the status of the components of the system, to trigger reconfiguration when components crash or are started, instruct the components on how to "wire" themselves and what configuration parameters to use, and carry out a variety of load-balancing tasks. The system is particularly helpful in managing the sharded data replication structure discussed earlier: process groups are a natural match to our shards.

B. TCPR

TCPR is a NAT-like person-in-the-middle solution that permits us to easily migrate the shard representatives associated with each of our GridCloud sensors (the PMU data capture components) in ways that are non-disruptive to existing TCP connections and, indeed, completely hidden from the PMU devices themselves. Using Isis^2, we build a small replication group consisting of the current shard component handling a given PMU device and the new component that will take over this role. We can then migrate responsibility and, using TCPR, simultaneously move the TCP endpoint without breaking the existing TCP connection, an event that would introduce delay. The old representative can then be shut down.

C. *Swift*

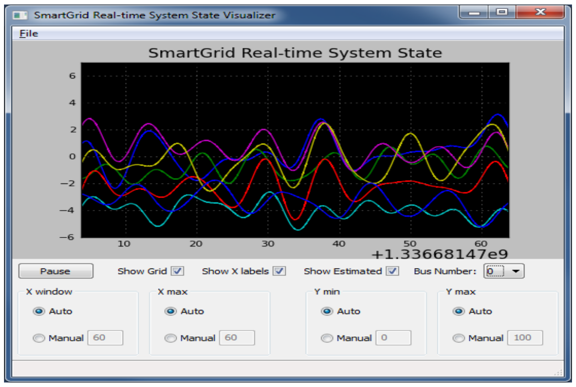
Swift [27] is a parallel scripting framework used as a "driver" to orchestrate a multi-staged computation in a concurrent manner automating the interdependencies among the involved stages. Swift is interfaced to clouds via specific 'providers'. Swift framework has been successfully applied to clouds for HPC applications [28] via its "coaster providers".

We use Swift to express and execute the pipeline of grid-cloud components as shown in figure 1. Swift allows to express the chain of computation with implicit, best-effort parallelism while keeping dependencies intact.

D. *Experiment Setup*

We implemented a basic linear state estimator, which acts as a baseline used for performance measurements. The estimator assumes that the state of the system expressed as a multi-dimensional vector varies linearly within a small time window. PMU measurements are taken at 30 Hz, and each measurement is an 8-tuple (7 power quantities + 1 timestamp) data packet of size 42 bytes. State estimation is run at 5 Hz, so 6 time-steps of measurements go into the state estimator per iteration. This data rate may actually fill up the bandwidth of a T1 line with just 2 PMUs. Such high data rates would be useful for fast local hardware-based controllers that would provide reactionary control (*e.g.*, trip circuit breakers if the controller notices a power surge or phasor angle inversion.) However, higher level state estimators would not be able to handle such high data rates, and the PMUs will compute aggregates (perhaps averages of actual measurements that make up one "measurement") and send that data up to the state estimators.

We deployed and ran our application on the "Red Cloud" supported by Cornell University Advanced Center for Computing. Red Cloud is an on-demand research computing service available by subscription.


SmartGrid Real-time System State Visualizer
File
SmartGrid Real-time System State
6
4
2
0
−2
−4
−6
10
20
30
40
50
60
+1.33668147e9
Pause

Show Grid ✓ Show X labels ✓ Show Estimated ✓ Bus Number: 0

X window: (●) Auto ( ) Manual 60
X max: (●) Auto ( ) Manual 60
Y min: (●) Auto ( ) Manual 0
Y max: (●) Auto ( ) Manual 100

Fig. 3. Visualization interface of grid-cloud for the smart grid state in quasi-realtime. Bus numbers can be configured to view individual elements. Shown here are the seven power quantities measured by PMU devices.

E. *Results and Evaluation*

Figure 4 shows latencies of end-to-end data movement across the web from Planetlab nodes to the cloud nodes

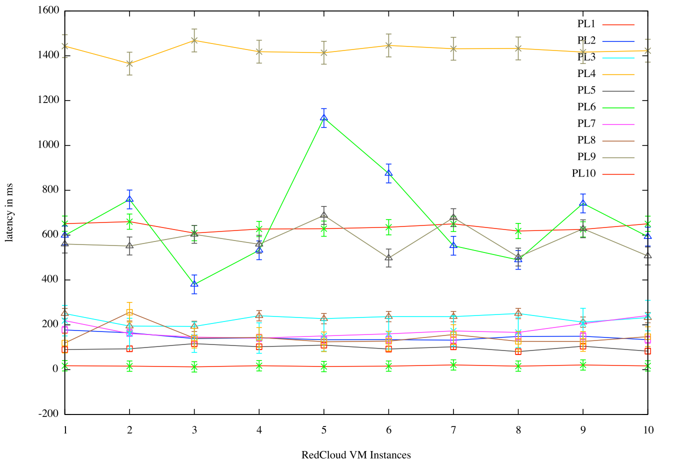


Fig. 4. Latencies in ms between ten Planetlab nodes and cloud VMs each. The error bars show measured clock inaccuracies between the corresponding endpoints.

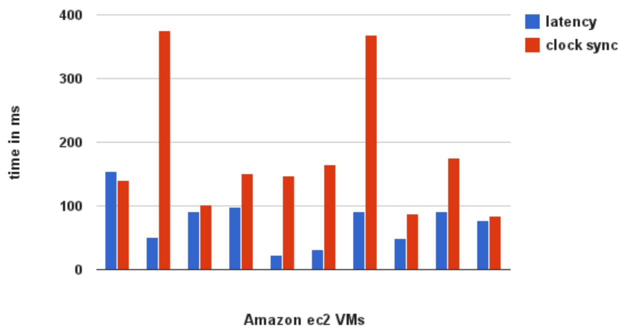


Fig. 5. Latencies and clock synchronization inaccuracies in ms between ten Amazon ec2 cloud VMs each.

with the NTP-based clock synchronization inaccuracies as error margins (between 10 and 60 msec). We chose Planetlab nodes from multiple global locations with the hope of having more pronounced latencies. As the plot shows, the worst case latencies are about 1.4 seconds for Planetlab node across the Pacific ocean while the best case of about 10 msec is for a node in Michigan in the US. Given a possibility of wide range of latencies in arrival of data at the state estimation node, a best-effort state-estimation computation had to be performed with available data and dropping the data arriving beyond a threshold time. A configurable interval of time of 16 msec was established corresponding to the 5 Hz frequency of state estimation and data arriving beyond this window was discarded. The missed PMU data is available for historical analysis from the storage logs. In another measurement, we measured the latencies between Amazon ec2 cloud VMs. We see the following two phenomena in figure 5: (a) clock inaccuracies are more pronounced; and (b) the latency variability is less pronounced. Phenomena (a) is seen owing to the fact that cloud VMs tends to be located on same datacenters and (b) to the fact that a on a cloud VM, rather limited actual cpu slices are available to NTP for synchronization purposes. A live visualization setup provides a continuous view of the state of the smart grid on a local submit host. The snapshot shown in figure 3 shows one such state of the eight power quantities in the form of plots.

VI. Conclusions and Future Work

GridCloud is a new technology for hosting smart grid computations. The contributions of the present paper centered on modeling, measuring, and characterizing smart grid computations in the cloud. For brevity, we focused on state estimation: an important and demanding use case with time-critical roles, and in which fault-tolerance and data security pose significant challenges. Our success in addressing these issues supports our contention that the cloud is a suitable infrastructure for supporting at least some kinds of smart grid computational tasks.

The work presented in this paper is just a start. Our long term vision is of a large-scale distributed operating system for real-time applications running in commodity clouds.

Acknowledgments

We thank our funding agency, ARPA-e for their research grant. We are thankful to our collaborators at Washington State University, our students and colleagues for their valuable insights during this work.

References

[1] K. Martin, G. Benmouyal, M. Adamiak, M. Begovic, J. Burnett, R.O., K. Carr, A. Cobb, J. Kusters, S. Horowitz, G. Jensen, G. Michel, R. Murphy, A. Phadke, M. Sachdev, and J. Thorp, "IEEE standard for synchrophasors for power systems," *Power Delivery, IEEE Transactions on*, vol. 13, no. 1, pp. 73–77, Jan. 1998.

[2] K. Martin, D. Hamai, M. Adamiak, S. Anderson, M. Begovic, G. Benmouyal, G. Brunello, J. Burger, J. Cai, B. Dickerson, V. Gharpure, B. Kennedy, D. Karlsson, A. Phadke, J. Salj, V. Skendzic, J. Sperr, Y. Song, C. Huntley, B. Kasztenny, and E. Price, "Exploring the IEEE Standard C37.118 2005 synchrophasors for power systems," *Power Delivery, IEEE Transactions on*, vol. 23, no. 4, pp. 1805–1811, Oct. 2008.

[3] A. Ipakchi and F. Albuyeh, "Grid of the future," *Power and Energy Magazine, IEEE*, vol. 7, no. 2, pp. 52–62, March-April 2009.

[4] E. Lightner and S. Widergren, "An orderly transition to a transformed electricity system," *Smart Grid, IEEE Transactions on*, vol. 1, no. 1, pp. 3–10, Jun. 2010.

[5] A. Armenia and J. Chow, "A flexible phasor data concentrator design leveraging existing software technologies," *Smart Grid, IEEE Transactions on*, vol. 1, no. 1, pp. 73–81, Jun. 2010.

[6] K. P. Birman, L. Ganesh, and R. van Renesse, "Running smart grid control software on cloud computing architectures," in *Workshop on Computational Needs for the Next Generation Electric Grid, Ithaca, NY*, Apr. 2010.

[7] D. Atkins, "Revolutionizing science and engineering through cyberinfrastructure: Report of the National Science Foundation Blue-Ribbon Advisory Panel on Cyberinfrastructure." [Online]. Available: http://hdl.handle.net/10150/106224

[8] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the clouds: A Berkeley view of cloud computing," Feb 2009.

[9] D. M. Falcao, F. F. Wu, and L. Murphy, "Parallel and distributed state estimation," *Power Systems, IEEE Transactions on*, vol. 10, no. 2, pp. 724–730, May 1995. [Online]. Available: http://dx.doi.org/10.1109/59.387909

[10] A. Phadke and J. Thorp, *Synchronized Phasor Measurements and Their Applications*, ser. Power Electronics and Power Systems. Springer, 2008.

[11] T. Yang, H. Sun, and A. Bose, "Transition to a Two-Level Linear State Estimator – Part I: Architecture," *Power Systems, IEEE Transactions on*, vol. 26, no. 1, pp. 46 –53, feb. 2011.

[12] ——, "Transition to a Two-Level Linear State Estimator – Part II: Algorithm," *Power Systems, IEEE Transactions on*, vol. 26, no. 1, pp. 54 –62, feb. 2011.

[13] A. Bose, "Smart transmission grid applications and their supporting infrastructure," *Smart Grid, IEEE Transactions on*, vol. 1, no. 1, pp. 11–19, Jun. 2010.

[14] J. Hazra, K. Das, D. P. Seetharam, and A. Singhee, "Stream computing based synchrophasor application for power grids," in *Proceedings of the first international workshop on High performance computing, networking and analytics for the power grid*, ser. HiPCNA-PG '11. New York, NY, USA: ACM, 2011, pp. 43–50. [Online]. Available: http://doi.acm.org/10.1145/2096123.2096134

[15] S. Rusitschka, K. Eger, and C. Gerdes, "Smart grid data cloud: A model for utilizing cloud computing in the smart grid domain," in *2010 First IEEE International Conference on Smart Grid Communications (SmartGridComm)*, Oct. 2010, pp. 483–488.

[16] Y. Zhang, P. Markham, T. Xia, L. Chen, Y. Ye, Z. Wu, Z. Yuan, L. Wang, J. Bank, J. Burgett, R. Conners, and Y. Liu, "Wide-area frequency monitoring network (FNET) architecture and applications," *Smart Grid, IEEE Transactions on*, vol. 1, no. 2, pp. 159–167, Sep. 2010.

[17] D. Novosel, V. Madani, B. Bhargava, K. Vu, and J. Cole, "Dawn of the grid synchronization," *Power and Energy Magazine, IEEE*, vol. 6, no. 1, pp. 49–60, January-February 2008.

[18] R. Podmore and M. Robinson, "The role of simulators for smart grid development," *Smart Grid, IEEE Transactions on*, vol. 1, no. 2, pp. 205–212, Sep. 2010.

[19] C. Queiroz, A. Mahmood, and Z. Tari, "SCADASim: A framework for building SCADA simulations," *Smart Grid, IEEE Transactions on*, vol. 2, no. 4, pp. 589–597, Dec. 2011.

[20] K. E. Martin, "Synchrophasor standards development– IEEE C37.118 & IEC 61850," in *Proceedings of the 2011 44th Hawaii International Conference on System Sciences*, ser. HICSS '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 1–8. [Online]. Available: http://dx.doi.org/10.1109/HICSS.2011.393

[21] M. Hurtgen and J.-C. Maun, "Advantages of power system state estimation using phasor measurement units," in *Power Systems Computation Conference (PSCC), Glasgow, Scotland*, Jul. 2008.

[22] C. Gomez-Quiles, A. Gomez-Exposito, and A. de la Villa Jaen, "State estimation for smart distribution substations," *Smart Grid, IEEE Transactions on*, vol. 3, no. 2, pp. 986–995, Jun. 2012.

[23] J. Baliga, R. Ayre, K. Hinton, and R. Tucker, "Green cloud computing: Balancing energy in processing, storage, and transport," *Proceedings of the IEEE*, vol. 99, no. 1, pp. 149–167, Jan. 2011.

[24] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, "The Eucalyptus open-source cloud-computing system," in *9th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID '09)*, May 2009, pp. 124–131.

[25] L. Peterson and T. Roscoe, "The design principles of PlanetLab," *SIGOPS Oper. Syst. Rev.*, vol. 40, no. 1, pp. 11–16, Jan. 2006. [Online]. Available: http://doi.acm.org/10.1145/1113361.1113367

[26] K. Birman, *Guide to Reliable Distributed Systems: Building High-Assurance Applications and Cloud-Hosted Services*, ser. Texts in Computer Science. Springer, 2012. [Online]. Available: http://books.google.com/books?id=yG9y-VMluwYC

[27] M. Wilde, M. Hategan, J. M. Wozniak, B. Clifford, D. S. Katz, and I. Foster, "Swift: A language for distributed parallel scripting," *Parallel Computing*, vol. 39, no. 9, pp. 633–652, September 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167819111000524

[28] K. Maheshwari, J. M. Wozniak, A. Espinosa, D. Katz, and M. Wilde, "Flexible cloud computing through Swift Coasters," in *Proc. Cloud Computing and its Applications*, 2011.