Title
Uncertainty Analysis of Middleware Services for Streaming Smart Grid Applications

Journal
IEEE Transactions on Services Computing, 9(2)

ISSN
1939-1374

Authors
Akkaya, Ilge
Liu, Yan
Lee, Edward A

Publication Date
2016

DOI
10.1109/tsc.2015.2456888

Peer reviewed

Uncertainty Analysis of Middleware Services for Streaming Smart Grid Applications

Ilge Akkaya, *Student Member, IEEE,* Yan Liu, *Member, IEEE,* and Edward A. Lee, *Fellow, IEEE*

Abstract—Accuracy and responsiveness are two key properties of emerging cyber-physical energy systems (CPES) that need to incorporate high throughput sensor streams for distributed monitoring and control applications. The electric power grid, which is a prominent example of such systems, is being integrated with high throughput sensors in order to support stable system dynamics that are provisioned to be utilized in real-time supervisory control applications. The end-to-end performance and overall scalability of cyber-physical energy applications depend on robust middleware services that are able to operate with variable resources and multi-source sensor data. This leads to uncertain behavior under highly variable sensor and middleware topologies. We present a parametric approach to modeling the middleware service architecture for distributed power applications and account for temporal satisfiability of system properties under network resource and data volume uncertainty. We present a heterogeneous modeling framework that combines Monte Carlo simulations of uncertainty parameters within an executable discrete-event middleware service model. By employing Monte Carlo simulations followed by regression analysis, we quantify system parameters that significantly affect behavior of middleware services and the achievability of temporal requirements.

Index Terms—Systems engineering, data-driven design, uncertainty, simulation, middleware, power applications.

1 Introduction

SMART grid is the effort of transforming the existing electrical power grid infrastructure to be more intelligent, accurate and evolving by jointly employing the technologies in the areas of communications, controls and computing. Today, power grid systems are incorporating high throughput sensor devices into power distribution networks. As Figure 1 demonstrates, power grid operators rely on streaming sensor data to make real-time control decisions. With high-fidelity and trustworthy phasor data from hundreds of thousands of measurement points within the Wide Area Measurement System (WAMS), the future electric power grid will enable multiple advanced distributed control techniques. Phasor Measurement Units (PMUs) provide real-time and precisely time stamped measurements, which is a feature that enables the measurements to be time-aligned and aggregated for accurate real-time evaluation of the grid state. The challenge of combining phasor measurements for Wide Area Monitoring and Control (WAMC) applications has been extensively investigated by power engineers and researchers [1], [2], [3]. From a distributed cyber-physical system (CPS) perspective, the challenge is to autonomously coordinate the data flow and access within distributed power systems [4], [5].

Middleware plays an important role in linking applications and data that may be on different domains, enabling these to work together. Surveys presented in [6], [7] highlight that middleware services should be developed as a bridge between systems and operators. [6] proposes a middleware service architecture design that expands across three logic layers, namely, *transmission*, *control*, and the *user*. The transmission layer encompasses generation, distribution, and communication that provides data transfer to Advanced Metering Infrastructure (AMI), allowing users to configure energy consumption through smart devices. Even within the transmission layer, monitoring, and control applications based on intensive analysis of measurements requires certain middleware components to coordinate the data communication and analysis process.

In this middleware oriented service architecture, a number of factors that either depend on environmental conditions or cannot be predetermined at the time of deployment affect end-to-end response times of power applications, creating a challenge for realizing applications with strict timing requirements. Such uncertain factors include:

- run time of iterative and approximate distributed applications, which run until global convergence of state has been achieved;
- sensor data quality, which is highly affected by sensor failures and environmental conditions;
- sensor placement;
- smart grid partitioning decisions for distributed sensing and actuation.

These variations in smart grid design lead to considerable jitter in reliability, bandwidth, and latency, which are characterized as Quality of Services (QoS) attributes of integrated power applications. A systematic quantification of these factors gain importance in evaluating typical and worst-case performance of smart grid applications and be-

- *I. Akkaya and E.A. Lee are with the Department of Electrical Engineering and Computer Science, University of California, Berkeley, CA, USA E-mail: {ilgea,eal}@eecs.berkeley.edu*
- *Y. Liu is with Faculty of Engineering and Computer Science, Concordia University, Montreal, QC, Canada. E-mail: yan.liu@concordia.ca*

Manuscript received 12 Feb. 2015; revised 16 June 2015; accepted 1 July 2015. Date of publication 0 . 0000; date of current version 0 . 0000.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TSC.2015.2456888

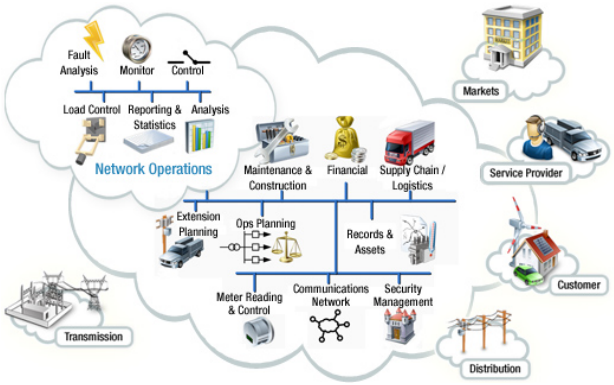

Fig. 1: National Institute of Standards and Technology (NIST) Smart Grid Conceptual Model: Operations [9]

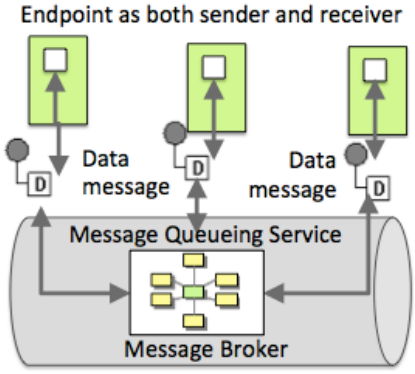

Fig. 2: Data Exchange Only Middleware Architecture

come metrics that are used in comparing alternative middleware architectures [8].

In this paper, we present a model-based design approach to specifying uncertainty parameters in middleware service architectures. We consider the smart-grid communication fabric that entails the entire data flow from sensor devices, through network and middleware components, to distributed application nodes. We pursue a model-based approach to (i) build executable Discrete-Event (DE) models of PMU networks, computation nodes, and the network fabric that interconnects the components, along with the middleware service layer to process and aggregate data streams, (ii) carry out a Monte Carlo (MC) simulation based approach to evaluate a parameter space that characterizes available sensor, middleware services, and network resources in the distributed model.

This paper is structured as follows: Section 2 presents an overview of middleware service architecture, Section 3 describes the modeling approach for distributed power applications, as well as model parameterization and sampling methods. Section 4 presents methodology and results of parametric uncertainty analysis and a comparative performance analysis of data volume and latency requirements on two middleware service architectures. Finally, related work is presented in Section 5, followed by Section 6, which concludes the article.

2 Middleware Service Architecture

For the purposes of utilizing sensor streams at power grid nodes from a large-scale transmission grid in a distributed application, the entire electric power grid topology is partitioned into non-overlapping subsystems that are connected via tie lines (buses). Each node of the distributed application, usually delegated to a balancing authority (BA) or a control center of a power utility, is able to run a local subapplication and consequently exchange data with neighboring substations for coordination. Middleware services that mediate data exchange between partitions of the grid are integrated into the communication infrastructure. In this paper, we consider two scenarios of a middleware service architecture that respectively perform (i) aggregation and dispatching; and (ii) mediation of data exchange only (without aggregation).

For the data exchange only middleware service architecture, each area is assumed to share its local data stream with all other participating areas, as shown in Figure 2. No coordination or mediation is performed at the middleware level; middleware only relays messages from the source to the destination. Main components of middleware consists of the service endpoints defined by a message queuing service framework and a message broker that relays data streams.

For applications in which data streams collected from each local area are not fully accessible to other areas or when only intermediate data need be shared between designated nodes, a middleware service with *aggregation and dispatch* functionality comes into play. Figure 3 demonstrates the conceptual architecture for an aggregate-and-dispatch middleware service. In this scenario, local streams from each area are transferred into the middleware to be aggregated and time-aligned. Additionally, the middleware fabric detects time-aligned faulty readings from the sensor network, then merges and broadcasts this information to all remote participants for situational awareness [4]. An additional decision component, which is also part of the middleware service, receives these intermediate results and notifies each area when a consensus on the power system state has been made. Centralized aggregation also becomes desirable for applications, in which broadcasting high intensity data streams individually would incur significant network load and would impact end-to-end system latency.

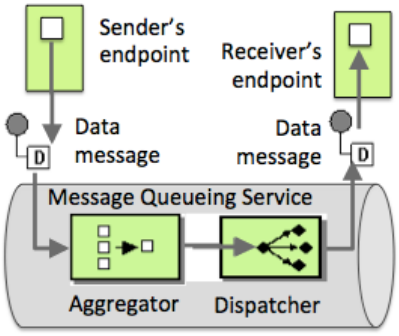

Fig. 3: Aggregate-and-dispatch Middleware Architecture

2.1 Middleware for Centralized Aggregation

As an example application for which middleware that performs centralized aggregation proves necessary, we consider a distributed state estimation scenario that operates on PMU data streams. The middleware coordinates multiple data sources as inputs to the power application that consumes (i) PMU sample files that contain measurements from a power node (typically collected with a sampling rate of 30 Hz), received at minutely intervals from each sensor; (ii) a SCADA file received at 1-4 minute intervals; and (iii) an error log file that contains records of time stamp errors from all PMUs across areas. The middleware components that process these data sources are highlighted in green in Figure 4. Next, we will characterize the middleware components that are required for this functional workflow.

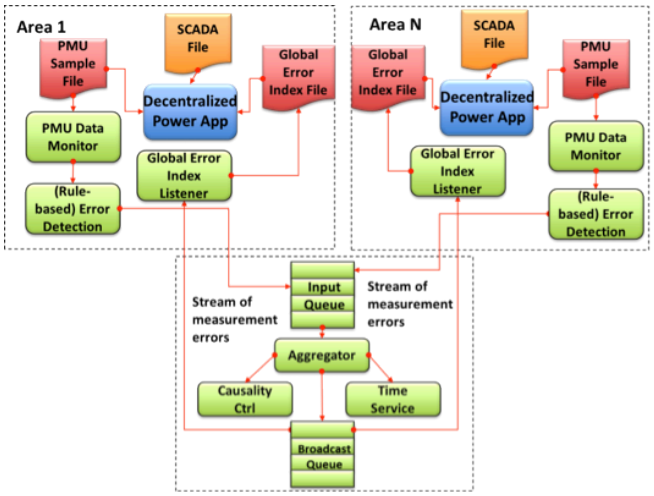

Fig. 4: Middleware for coordinating multiple sources of data streams

Aggregation. At each estimation step, PMU sensor readings from the previous time interval are obtained from the local areas. The `PMU Data Monitor` detects the arrival of the PMU data file, and pipelines the data stream to the `Error Detection` component for identification of records with time stamp errors. An error file containing the records for local PMUs is then sent to a message broker associated with a message queue denoted as the `Input Queue`. Arrival messages in the input queue are processed by the `Aggregator` that combines and aligns the records in the error index files sent from distinct areas. The aggregator performs time alignment according to the GPS clock reference obtained via the `Time Service` that listens to one channel of PMU data over TCP.

The aggregate time stamp error index file is then broadcast to individual areas through the `Broadcast Queue`. This enables all the areas to become aware of the time stamp for which at leas one PMU reported erroneous data in the previous time interval. All PMU records for any such time stamp are discarded even when the local PMU measurements are of good time quality, for consistency considerations.

Convergence Control. One of the essential functionalities of the middleware is to moderate the consistency of state across participating areas. In our scenario, neighboring areas exchange state information through peer-to-peer communications. The `Causality Control` component monitors intermediate states to determine and declare global convergence.

Listing 1: Monitoring data files and filtering errors in middleware

```
1  //create a pipeline
2  MifPipeline pipeline = new MifPipeline();
3  //set the file connector
4  MifConnector conn = pipeline.addMifConnector(
       EndpointProtocol.FILE);
5  conn.setProperty("streaming", true);
6  conn.setProperty("autoDelete", true);
7  conn.setProperty("pollingFrequency", 60000);
8  conn.setProperty("fileAge", 60100);
9  conn.setProperty("moveToDirectory",
       DseMiddlewareProps.PMU_DSE_WORKING_DIR));
10 ...
11 // add a component to process the file
12 MifModule pmuAreaAModule = pipeline.
       addMifModule(
13 PmuRemoveErrorsFileProcessor.class,
       areaADataAppUri, "jms://topic:pmuIndexTopic
       ");
14 pmuAreaAModule.setName("AreaA-PMU-RECEIVER");
15 ...
16 //start a pipeline; a file arriving will
       trigger PmuRemoveErrorsFileProcessor
17 pipeline.start();
```

2.2 Middleware Component Implementation

The aggregation and queuing behavior is implemented using the Apache ActiveMQ service, which is an open-source message broker [10]. The middleware components that connect to message queues are designed using MeDICi [11], an open source middleware framework for building services. MeDICi has been utilized in other data intensive analysis and its scalability has been demonstrated in coordinating large scale concurrent data analysis tasks [12].

An example MeDICi implementation for a PMU `Data Monitor` routine is given in Listing 1 (lines 1-9). A file connector is added to a `MifPipeline` (the execution component of MeDICi) that handles any arriving files. The file connector implements services such as getting the file as a byte array or an input stream, polling frequency, and auto-deleting processed files. The file stream is then processed by `PmuRemoveErrorsFileProcessor` (line 11-14). This processor implements the `Error Detection` component. It extracts the records with clock errors and sends the error file to an aggregator. Its inbound endpoint (`reaADataAppUri`, line 13) refers to the directory that the file connector monitors the arriving files. The outbound endpoint specifies the URL (jms://topic:pmuIndexTopic, line 13) of the `Input Queue`.

These MeDICi components and queuing structure form the aggregation process of the middleware by merging multiple data sources as inputs to distributed power applications.

2.3 Uncertainty Parameters of Middleware Services

We evaluate a number of significant parameters that account for the uncertainty within the middleware level end-to-

end latency, based on the aggregate-dispatch architecture presented in Figure 4.

1) *Number of PMU streams per area.* For each PMU, the measurements to be extracted are determined already given by the state estimation algorithm. Each PMU packet follows the IEEE C37.118 Synchrophasor Standard. It contains multiple measurements of frequency, voltage magnitude and angle, current magnitude and angle, quality and so on. In our testbed environment, a Java application listens to the broadcast from a PMU TCP socket, extracts measurement data of interests and generates a PMU data file in a minute cycle. So at the top of every minute, the middleware is triggered to process PMU data files as the number of available PMU devices.
2) *Middleware Concurrency*. Upon arrival of a PMU file, the MeDICi component automatically creates a new instance of a thread for concurrent processing. The aggregation and queuing behavior is implemented using the ActiveMQ message broker. Application specific requirements of middleware concurrency may require determining an optimal level of concurrency given the expected input data volume. For critical applications, availability requirements can be imposed to ensure peak loads are process within the temporal application deadline.
3) *Intermediate data exchange sessions until convergence.* Intuitively, the number of intermediate data exchange sessions required until global convergence depends on the data quality, namely, the accuracy of PMU measurements. Such a relation is hard to formulate as it highly relies on state estimation algorithms as well as the distribution of the noise level. In [13], the data noise is assumed to have a linear relation to the number of iterations until convergence and are investigated under a Gaussian distribution assumption. For specific applications, these assumptions may need to be re-calibrated.

Based on our experience discussed in [4], it was expensive and time consuming to install a networked testbed that connects physical PMU devices to distributed state estimators running on remote High Performance Computing (HPC) clusters. In addition to the cost of the PMU devices, at least 200 engineering hours were spent solely on the integration of PMU related software. Moreover, measuring the middleware and end-to-end response times in a testing environment has an even higher deployment cost, since it involves emulation scenarios over hundreds of PMUs for a single state estimation area alone. Considering the cost of actual deployments and the status of the developing state of smart grid technology, for which no standardized middleware requirements have been set to date, a simulation based approach that aids in modeling uncertain parameters as random variables gains further value. We aim to determine the influence of the three fundamental sources of uncertainty listed above as a result of our simulation studies. The variables of uncertainty are sampled from a parameter space and used to configure a simulation model to characterize the simulated end-to-end and local latency behavior of distributed state estimation runs.

3 Modeling Approach

The conceptual modeling workflow is depicted in Figure 5. The end goal of our study is to derive the most significant uncertain parameters as random variables from the middleware design.

Samples for each of these random variables are generated by the Monte Carlo (MC) method. MC methods are a family of algorithms that perform repeated random sampling from a random space to simulate complex random variables.

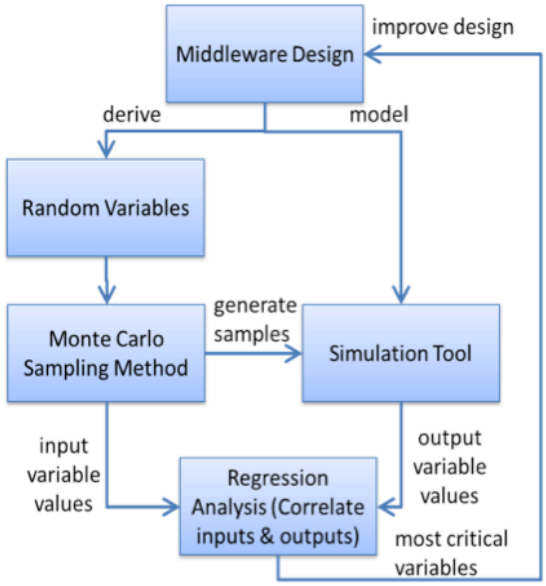

Fig. 5: Middleware design workflow guided by uncertainty analysis

We use the Ptolemy II [14] framework, which is a heterogeneous modeling and simulation environment extensively used in CPS design, for modeling and simulation purposes. We provide a hierarchical actor-oriented model of the middleware components and the data flow through the network and middleware fabrics to account for the temporal behavior of the system. In this work, some general elements that need to be modeled in particular are sensors, network and middleware components. Communication delays, software scheduling and timing properties are also considerations of the modeling process.

Note that Ptolemy II is used first to generate MC samples of tuples of parameters, which are then used to parameterize a Discrete-Event (DE) sub-model of the end-to-end communication flow of the distributed power grid application. The DE model is simulated to yield Monte Carlo samples of end-to-end completion times of the distributed state estimation application. Following the Monte Carlo simulation, input parameters and output samples are utilized for regression analysis, which is used to identify input variables that are statistically significant in explaining the temporal variation at the output.

3.1 Ptolemy II

Ptolemy II [14] is an actor-oriented design tool that provides an integrated modeling and simulation environment for the conceptual procedure in Figure 5. Ptolemy II has extensive actor libraries for modeling CPS, and has been demonstrated to be an effective design platform [15], [16], [17]. A CPS model consists of physical processes that interact with models of network and computation platforms, usually including feedback relations.

Actors in Ptolemy II are concurrently executed components that communicate via *events* sent and received through input and output actor *ports*. An *actor* in Ptolemy II consists of a set of input and output ports along with an internal state, which itself can be a graphical submodel. A *director* implements a model of computation (MoC) and mediates actor execution at each level of the hierarchy, which is the mechanism that leverages integration of multiple MoCs within a single model. Passage of time at each compositional level is governed by a *local clock* that allows to define different clock rates, clock drifts and the relation of each *local clock* to the wall clock time. For the purposes of uncertainty analysis, MC sampling is carried out using the Synchronous Data Flow (SDF) MoC, and the temporal execution of the middleware communication model uses a Discrete-Event (DE) submodel within SDF.

Being an actor-oriented framework, Ptolemy II also presents a clear semantics for defining cross-cutting concerns of the behavioral models by the use of *aspects*. In the context of smart grid middleware modeling, network fabric and middleware models are examples of *communication aspects* that characterize aggregation, queuing and delaying of events within the behavioral model that consists of an interconnection of sensors, intermediate storage components and computation nodes. Our modeling effort follows this design pattern.

In aspect-oriented Ptolemy models, when a connection between two ports need to be delegated to an intermediate communication aspect, the input port of the destination actor is annotated with a parameter reference to the relevant `CommunicationAspect` component. Figure 6 demonstrates the use of communication aspects. The details of the aspect-oriented communication models are discussed in [18].

3.2 Modeling System Architecture

We first model the top level systems architecture and proceed to developing a hierarchical model that concentrates on quantifying the uncertainty in middleware coordination.

3.2.1 Top Level Model

In Ptolemy II, the high-throughput devices communicating over the network with packets can be abstracted to DE components communicating via time stamped events, where each network packet is represented by a discrete event. DE execution is based on events composed of a tag-value pair, representing the time stamp and the payload of a token. The DE scheduler guarantees that events are processed in time stamp order. Within one hierarchical level of a DE model, all actors share a global notion of time, namely the *model time*. This imposes a global ordering of events within the model and therefore ensures determinacy. The modeling details of the domain-specific system entities are presented below.

PMUs. Phasor Measurement Units are the main sources of phasor data, abstracted as DE events as they are represented in the Ptolemy execution. We collectively model the PMUs residing within a local area in the grid as a cluster and denote this component as a `PMUCluster<i>`, where `<i>` refers to the area index. This actor is parameterized by the `PMUCount` parameter and generates the corresponding number of events every iteration interval (parameterized by the `PMUPeriod`). `PMUCount` is one of the inputs to the executable model one level up in the hierarchy, and is sampled from a user-defined prior distribution.

Phasor Data Concentrators (PDCs). A PDC is responsible for receiving data from multiple PMUs and producing an aggregated data packet for each PMU at an application specific rate. We only model the relaying function of the PDC, where it produces data packets for each PMU and processes the packets in a FIFO pattern.

Distributed Power Application Workflow. Each distributed power application is locally run on a computation cluster and is expected to consist of multiple iterations, interleaved with peer-to-peer communications with the coordinating areas. We use a generic computation node model that accepts SCADA and PDC packets and produces intermediate packets after each algorithm iteration. The execution time for each iteration is characterized by a random variable that allows the user to associate a platform-dependent stochastic profile with the algorithm execution time at each iteration.

The top-level model is illustrated in Figure 6. In this architecture, the PMUs of each area produce data at 30 samples per second that are transmitted to the local PDC via a communication fabric, named `PMULink`. Data streams are relayed at the PDC and sent to the local computation node of each area to be utilized in the periodic power applications.

Network Fabric. We consider the supporting network fabric (both in `PMULink` and `LocalNet`) to follow the link specifications by the North American SynchroPhasor Initiative (NASPI) [https://www.naspi.org]. In this case, each PMU is connected to the local PDC by a T1 line. The packet size of each PMU message is assumed to be 128 bytes in compliance with the data format of IEEE C37.118 standard [19]. Each of these links is chosen to be 50 miles long, representing an average physical distance from a bus to the nearest PDC. We assume each PDC is placed 300 miles away from the local area and the middleware component is symmetrically located, at a maximum distance of 350 miles from the PDCs. The in-depth characterization of network links are studied in [13].

3.2.2 Middleware Model for Centralized Aggregation

We model middleware components and their interactions with other systems components in the entire data flow of centralized aggregation. Figure 6 implements a middleware aspect `MW`, that receives intermediate results from three areas, as well as from PDCs, and performs the following tasks:

Aggregation. Middleware is responsible for receiving packets from the PDCs of all distributed areas and aggregating the packets into a universal index file. The functional

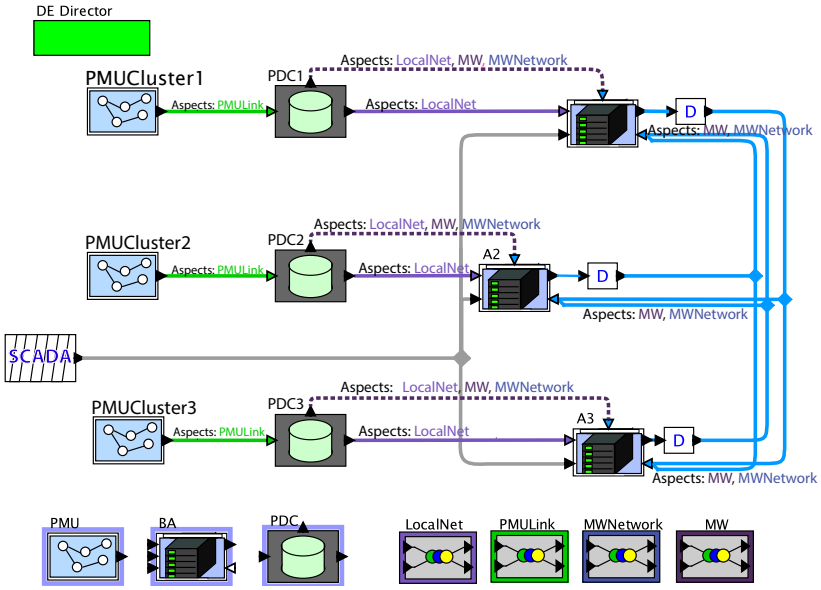

Fig. 6: Top Level Distributed Smart Grid Communication Model with Communication and Middleware Aspects

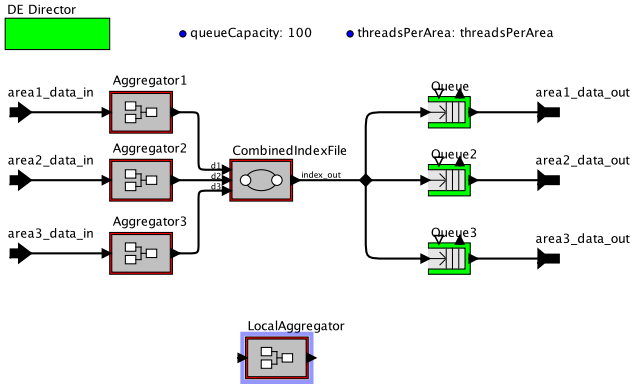

Fig. 7: Top Level Model of Middleware Aggregation (implemented by the `MW` aspect)

model of the top-level middleware aggregator is given in Figure 7. PMU streams from each area are first taken into a local aggregator (named `Aggregator` in the Ptolemy model), where they are processed on a per-file basis. A synchronization unit, called `CombinedIndexFile` then aggregates the streams from all areas into one global output packet.

The local aggregator model is given in Figure 8, where each received PMU stream is randomly assigned to one of the process threads, which internally impose a delay on the packet. The `MiddlewareQueue` components is designed as a thread pool, where the per-packet delay is modeled following benchmarks that will be studied in Section 3.4. The outputs of all instances are then merged to yield a stream containing all the processed PMU streams for this particular area.

Convergence Control. Another important role of the middleware simulation is to determine whether a distributed power application has reached a convergence state. For distributed state estimation and similar iterative power applications, the number of iterations until convergence relies on several parameters such as PMU data redundancy and quality. To be comprehensive, we assume a variable number of iterations, ranging in the discrete set $\{1, 2, ..., 20\}$ until convergence has been declared.

3.3 Data-Exchange Only Middleware Model

We now elaborate on an alternative middleware scenario for the middleware architecture, in which the centralized aggregation may require extensive resources and network fabric, and therefore become infeasible to couple streams from hundreds of sensors at a centralized node due to geographical and logical constraints . The data-exchange only middleware enables each distributed node to share streams, without the aggregate-dispatch behavior. Figure 9 demonstrates the Ptolemy model that considers such a middleware functionality.

For the data-exchange only middleware, the task of the middleware architecture becomes trivially to route each data stream to its respective destination. For the distributed state estimation case study, it is assumed that each BA has access to PMU data from all PMUs (i) within its local substation and (ii) from the neighboring area(s). According to this topology, BA1 and BA3 both receive data from PDC2 in addition to PDCs within their own area. BA2, on the other hand, remains a neighbor of both BA1 and BA3, therefore, the middleware must route PMU streams from *all* areas BA2.

Given the simplified role of routing of the middleware, whose delays are captured as a part of the `LocalNet` component, the local aggregation is performed at a local middleware, delegated separately to each distributed computation node.

A prototype model utilizing a local middleware is provided in Figure 9, as a refinement of `MW1`. For Area 1, the middleware is responsible for delivering data streams from PDCs 1 and 2, and does not receive any data streams from PDC3. The workload on the centralized middleware is

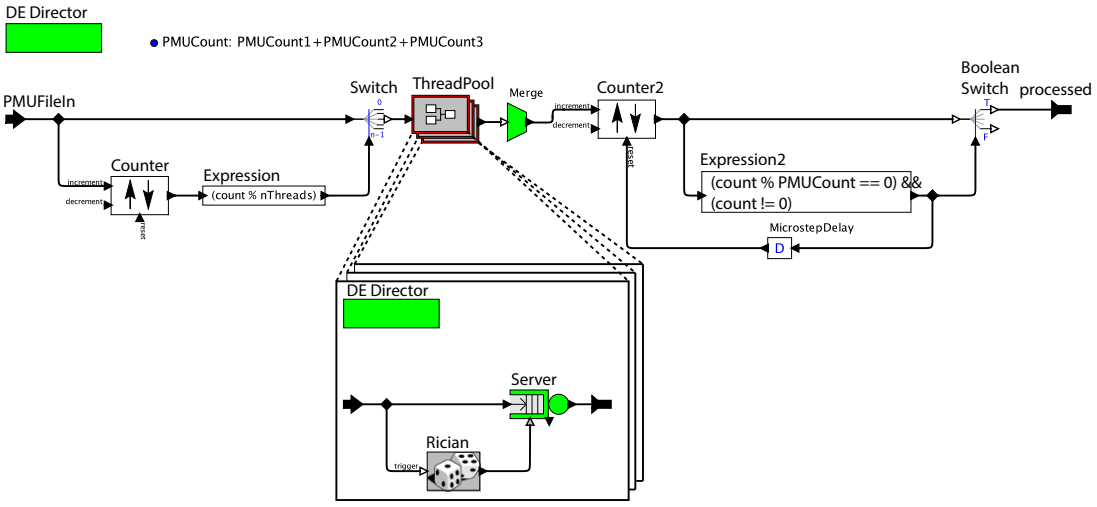

Fig. 8: The Local Aggregator Model

therefore reduced, at the expense of increased data redundancy over the network.

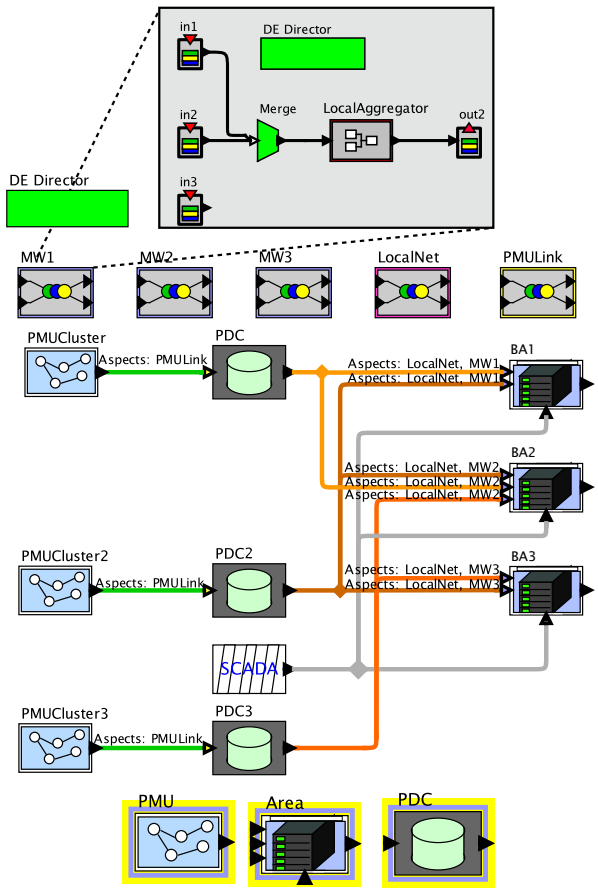

Fig. 9: Distributed Data-Exchange only Middleware Model

3.4 Model Calibration

For calibrating the model to accurately represent the characteristics of packet processing times in the middleware layer, we use the data obtained from benchmarks carried out on the ActiveMQ message broker that implements the aggregation and queuing behavior of the middleware. The trend for the cumulative completion times obtained on an ActiveMQ queue instance is presented in Figure 10. The pattern suggests that increasing the level of parallelism in the queue processing level is essential to improve the end-to-end latency, due to the long-tailed behavior of the process latency. The benchmarks are not only important for obtaining an average completion time, but are also essential to characterize middleware latency distribution.

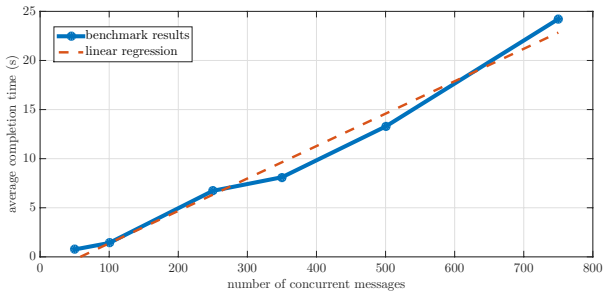

Fig. 10: Apache ActiveMQ Benchmark Results on Average Completion Times for Concurrent Arrivals

Following the distributions for the benchmarked number of sensor streams per area, we carry out a regression analysis on the time series obtained by the benchmarks to yield a distribution fit for the middleware processing overhead. The maximum-likelihood distribution fit for the benchmarked data is given by a Rician distribution parameterized as follows:

$$\begin{aligned} T_i &\sim Rice(\nu, \sigma) \\ \nu &= 0.0302 \cdot \log(N_{PMU_i}) + 0.055 \\ \sigma &= 0.0007 \cdot N_{PMU_i} + 0.0414 \end{aligned} \tag{1}$$

where T_i is a random variable that denotes the processing time in seconds of a PMU stream associated with Area i, $Rice(\cdot)$ denotes a Rician distribution with parameters ν and

σ, and N_{PMU_i} is the number of concurrent PMU streams delegated to Area i.

Figure 11 demonstrates a sample histogram for the distributions of per-file middleware processing times, obtained for 250 concurrent PMU files. The Rician distribution fitting is then used to parameterize the random delay imposed by the local aggregator as modeled in Figure 8, on a per-event basis.

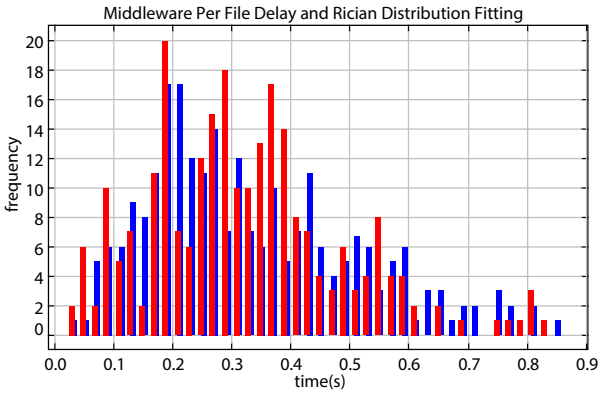

Fig. 11: Histograms for benchmarked [Red] and simulated [Blue] middleware delay

To further simulate the concurrent queue architecture, we make use of the Ptolemy actor called `MultiInstanceComposite`, which is capable of generating a user-defined number of copies of a component and to simulate the instances concurrently. The local aggregator model allows each received PMU stream to be randomly assigned to one of the processing unit instances within a thread pool, each simulating a server with stochastic latency that follows (1). The outputs of all instances are then merged to yield a stream containing all the processed PMU streams for this particular area.

3.5 Monte Carlo Sampling

Smart grid topologies are expected to be highly variable in the number of PMUs per area and the inter-area network characteristics. Middleware needs to be adaptive and flexible to support numerous scenarios of data volume and network characteristics. Monte Carlo simulations integrated with the distribution system topology is a useful tool to evaluate the effect of certain model parameters on middleware performance. Figure 12 displays the top level Ptolemy model that is used to generate Monte Carlo samples of selected model parameters, characterized in detail in Table 1. The `PowerGrid` component in Figure 12 is the top-level power grid to be simulated, and is given in Figure 6. Here, the parameters are assumed to be uniformly distributed over a finite set of integer values to avoid any bias towards a particular fashion for the PMU distribution. However, it should be noted that the number of iterations may follow a Gaussian-like distribution in practical applications. As we explained in Section II, calibrating the distribution for iteration quantification will require significant engineering effort. For simplicity, we also assume uniform distribution of the number of iterations.

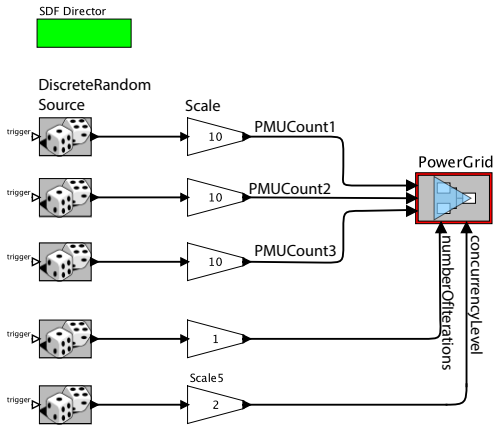

Fig. 12: Monte Carlo sampling in Ptolemy II

TABLE 1: Monte Carlo Variables and Respective Probability Mass Functions (range format:initial:increment:final)

Parameter Name	PMF	Range
PMUCount1	Uniform	10:10:500
PMUCount2	Uniform	10:10:500
PMUCount3	Uniform	10:10:500
concurrencyLevel	Uniform	2:2:20
numberOfIterations	Uniform	1:1:20

For a power system partitioned into three subareas, we are interested the parameter samples given by the 5-tuple: `<PMUCount1, PMUCount2, PMUCount3, concurrencyLevel, numberOfIterations>`. The model is executed for 6000 seconds in model time for each parameter sample, roughly corresponding to 100 complete cycles of the application. In the specific case of the distributed state estimation application, the cycle deadline is set to 60 seconds, which is equal to the period. For each 5-tuple that parameterizes the model, maximum and average end-to-end run times are recorded over model execution.

4 Uncertainty Analysis

4.1 Influence of Concurrency Level and Number of PMU Streams on End-to-End Runtime

Following data collection using Monte Carlo methods, we proceed to polynomial regression analyses to account for the effect and significance of the model parameters on the maximum end-to-end delay. We define the variables to be used in the regression analysis on Table 2.

The initial question addressed is the influence of the middleware concurrency level and the maximum number

TABLE 2: Variables for Regression Analysis

Variable	Explanation
x_1	concurrency level
x_2	$max\{$`PMUCount1, PMUCount2, PMUCount3`$\}$
x_3	number of iterations
y	maximum end-to-end runtime

TABLE 3: Polynomial Regression Coefficient Estimates

Coefficient	Estimate	Confidence Interval
$Intercept$	18.32	[14.73, 21.93]
x_1	-6.9699	[-7.47, -6.46]
x_2	0.14869	[0.13, 0.17]
x_1^2	0.73883	[0.70, 0.78]
x_1x_2	-0.01544	[-0.02, -0.01]
x_1^3	-0.02236	[-0.02, -0.02]

TABLE 4: Goodness of fit statistics for polynomial regression analysis

SSE	R^2	RMSE	AIC	BIC
2519	0.968	1.784	3209	3251

of PMUs per area on the end-to-end delay. The polynomial curve fitting expression given by (2) that uses x_1 and x_2 as independent variables and y as the dependent variable. This method reveals that a bivariate polynomial regression equation that is cubic in x_1 and quadratic in x_2 is the best-fitting model in the studied set of polynomial fits. The best fit is evaluated by the metrics of minimum Bayesian Information Criterion (BIC), Akaike Information Criterion (AIC), and sum-of-squares error (SSE), which were consistent on the decision.

which are desired to be high, and the sum-of-squares error (SSE) that should be minimized.

$$f(x_1,x_2) = p_{00} + p_{10}x_1 + p_{01}x_2 + p_{20}x_1^2 + p_{22}x_1x_2 + p_{02}x_2^2 + p_{30}x_1^3 + p_{21}x_1^2x_2 + p_{12}x_1x_2^2 \quad (2)$$

Table 3 presents the maximum-likelihood estimates of the polynomial coefficients for the regression fit, together with the confidence intervals for the coefficients. The p-value, which indicates the false alarm probability for the variable being estimated, is less than 10^{-6} for all the variables taken into account. This provides high confidence that the regression model is accurate and avoids overfitting.

The goodness of fit metrics presented in Table 4 further confirm that the variables chosen for Monte Carlo simulation sufficiently relate to the trend in the run time distribution. The R^2 value indicates that the independent variables x_1 and x_2 account for explaining 96.8% of the observed data. Moreover, the fit yields the optimal BIC among all bivariate fits constrained to have at most third order dependence to each parameter. Since AIC and BIC are goodness of fit metrics that establish a trade-off between model accuracy and complexity, considering both helps avoiding parameter overfitting.

The regression analysis that best fits obtained simulated outputs with the input data set is given in Figure 13 with the 95% confidence bounds. The concurrency level of at least 10 is necessary for the middleware to scale in handling increasing number of PMUs. The response surface has linear trend as the number of PMU increases, but remains planar as the concurrency level increases. This indicates further increasing the concurrency level would have marginal benefit.

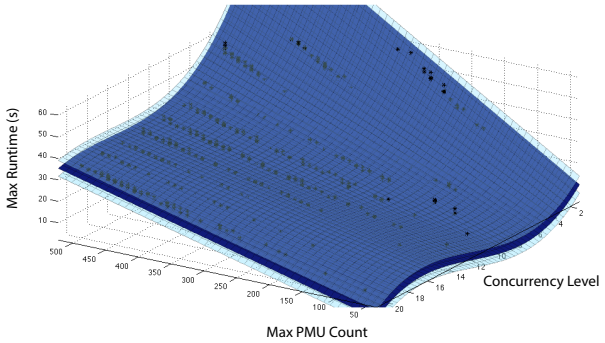

Fig. 13: Polynomial Regression Analysis of Significant Monte Carlo Variables

4.2 Influence of Number of Iterations on End-to-End Runtime

We continue uncertainty analysis on model parameters by considering the impact of DSE iterations on end-to-end run time. A scatter plot that demonstrates the Monte Carlo results as a function of DSE iterations is given by Figure 14. The additional polynomial regression analysis includes the number of iterations (x_3) as the third explanatory variable candidate. The notation follows from the previous analysis.

The best fit is selected by evaluating the AIC and BIC on a set of polynomial fits up to order 3 in each explanatory variable. AIC favors the model order of $\{3, 2, 2\}$ and BIC favors the model of order $\{3, 2, 3\}$, respectively, in $\{x_1, x_2, x_3\}$. The AIC-optimal polynomial fit is presented in Table 5. The significant coefficient estimates are given with confidence intervals centered around zero, and the higher-order parameters are omitted from the result.

Comparison of Table 4 and Table 6 reveals that, incorporating x_3 in the regression analysis has little improvement on the overall fit. Moreover, the coefficient estimates given in Table 5 that depend on x_3 are either numerically insignificant or have confidence bounds that intersect 0, suggesting that the coefficient values most likely do not express a significant explanatory relation to the observed variable y. This result provides more confidence that the number of iterations is a less significant variable that influences the maximum run time of the distributed state estimation algorithm, compared to the concurrency level and the maximum number of PMU streams per area.

To explain the insignificance of number of iterations in the analysis, we refer to the system model presented in section 3.2. As only intermediate estimate data on tie-line buses are exchanged, the data communication load is only approximately 65K bytes per iteration. Even for a larger scale power system such as the Western Electricity Coordinating Council (WECC) with more than 1300 buses and 6700 loads, the data exchange between neighboring areas would remain to be a relatively small overhead and would have little contribution to the end-to-end delay.

4.3 Evaluation of the Influence of Middleware Architecture on Temporal and Resource Constraints

The previous uncertainty analysis considers a centralized middleware architecture as opposed to a data-exchange

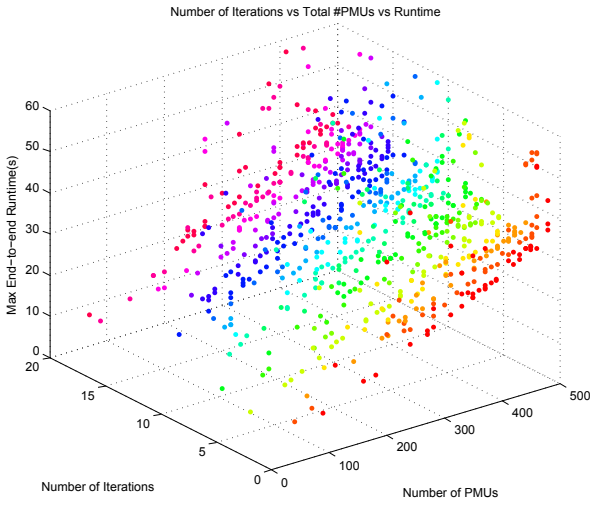

Fig. 14: Impact of Iterations and Number of PMU Streams on end-to-end Runtime

TABLE 5: Extended Polynomial Regression Coefficient Estimates

Coefficient	Estimate	Confidence Interval
x_1	12.2991	[7.82, 16.7761]
x_2	-6.9866	[-7.49, -6.4822]
x_3	0.1743	[0.15, 0.1997]
x_1^2	0.6242	[0.21, 1.0379]
x_1x_2	0.7524	[0.72, 0.7884]
x_2^2	-0.0162	[-0.02, -0.0145]
x_1x_3	0.0001	[0.00, 0.0001]
x_2x_3	-0.0065	[-0.03, 0.0185]
x_3^2	-0.0019	[-0.00, -0.0003]
x_1^3	-0.0091	[-0.02, 0.0064]
$x_1^2x_2$	-0.0225	[-0.02, -0.0214]

only middleware that enables reduced data transfer over the network layer, at the expense of requiring additional resources at the distributed computation end.

We initially address the temporal effects of adapting a centralized middleware architecture that performs aggregate-and-dispatch versus a data-exchange only middleware, which requires distributed middleware components to aggregate data locally at each node. Figure 15 presents a comparison of end-to-end delay under the two alternative middleware architectures as a function of worst-case middleware capacity, obtained for a PMU distribution of {200,100,100} over the three areas. Note that the distributed middleware provides marginally lower mean and worst-case latency, due to the avoided expense of central aggregation. As middleware capacity is increased beyond the point where any queuing occurs at middleware level (for Thread Pool Size greater than 40), the two architectures are observed to exhibit comparable latency behavior, as the

TABLE 6: Goodness of fit statistics for extended polynomial regression analysis

SSE	R^2	RMSE	AIC	BIC
2093	0.974	1.63	3079	3163

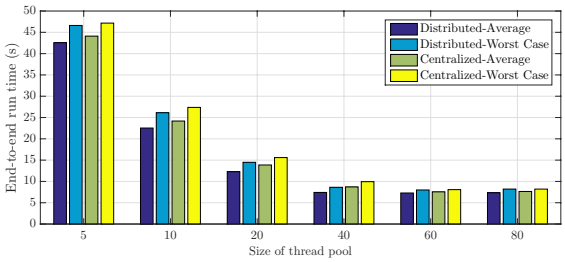

Fig. 15: End-to-end run time comparison of middleware architectures with variable concurrency levels

overhead of data aggregation becomes comparable to the noise level in the per-packet latency.

We would also like to investigate the data volume imposed by these two middleware architectures to the network fabric. Since smart grid applications will likely be publishing data at high rates and increased resolution, network will soon become a scarce resource and data level optimizations will gain importance. Note that the distributed middleware only requires transfer of the PMU files from PDCs to local middleware, after which the per-area state estimation can be performed in a standalone fashion. On the other hand, in the case of centralized middleware, PDC-to-MW communication needs to be followed by MW-to-BA, as well as peer-to-peer BA-to-BA intermediate communication until convergence.

Figure 16 demonstrates the data volume exchanged under the two aforementioned middleware topologies, as a function of partitions within the middleware. It is important to note that, for a coarse partitioning of the power grid, where a large number of PMU streams are delegated to a single area for reduced inter-area communication, a centralized aggregate-and-dispatch architecture requires lower total data volume to be transferred over network fabrics, despite the intermediate data needed until algorithm convergence. The exchange-only distributed middleware architecture requires high volumes of network traffic, since each area contains a large number of PMU files that need to be shared with neighboring areas redundantly. As the grid becomes finely partitioned, i.e., the number of grid areas are increased, a distributed topology becomes more advantageous, due to the intermediate data exchange overhead that dominates data volume in the centralized case.

Figures 15 and 16 provide a well-defined basis for our decision to utilize the centralized middleware architecture for the uncertainty analysis provided in Section 4.

The demonstrated model-based design workflow, of course, applies to a more general set of stream computing frameworks, such as Apache Spark [20], and InfoSphere Streams by IBM [21]. For analyzing alternative stream processing frameworks within the context of smart grid applications, the presented simulation framework can be extended to issue service calls to the message queuing service, and receive processed data in real-time. In fact, Ptolemy II enables WebSocket based communication to external services [22], which alleviates the process of assessing a large set of stream processing platforms.

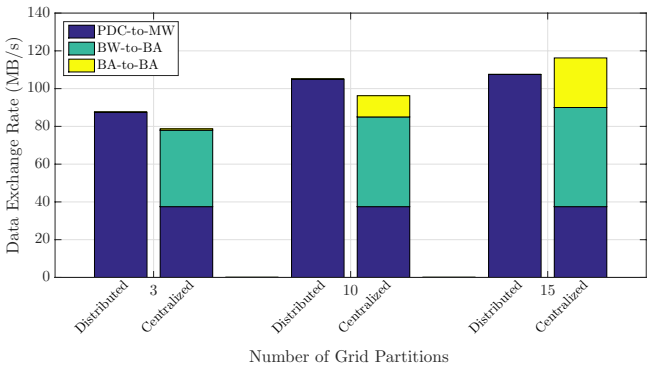

Fig. 16: Data exchange overhead imposed by middleware architectures for variable grid partitions

5 Related Work

The power industry has been evolving to incorporate electric distribution systems with redundant communication networks, sensor nodes and controllers to form a data-intensive framework, referred to as the "smart grid". The role of middleware in this service-oriented context, which assembles heterogeneous services, becomes prominent due to the data-oriented services that are provisioned to grow exponentially in number [6]. Surveys on middleware design and architectures [6], [7] present the architecture patterns, principles and existing middleware tools tailored for transmission, distribution and control services. Quantitative evaluation of middleware solutions in the smart grid, however, still demands additional consideration to facilitate the architecture design of distributed power applications.

Current model-based evaluation methods use various sets of probabilistic models. Markov chains, petri nets, queuing networks, finite state automata, stochastic processes, dependency graphs, and fault trees are widely-applied models for evaluating quality attributes of reliability, performance, safety, energy consumption. A comprehensive survey on the topic is presented in [23].

These models are mathematically complex, and in general, it is an involved task to derive quality metrics as a function of input distributions. In our approach, we use MC simulations to account for parametric uncertainty in the middleware architecture, where the MC simulation is natively composed with the executable DE application model. Our contribution in this work is to present a highly reconfigurable simulation framework for evaluation of middleware-centric distributed cyber-physical applications and to account for the feasibility of different middleware architectures for a family of distributed state estimation applications to be deployed in the future power grid.

One of the key challenges in smart grid application design is the complexity and heterogeneity of hardware and software components interacting in the grid. Model-based design has proven to be an effective way of modeling complex systems and has been applied in the context of evaluation of attributes of the smart grid, including a model-driven analysis framework for smart distribution networks based on the Common Information Model (CIM) [24], and for model-based CPES co-simulation [25].

Actor-oriented design tools has gained popularity for modeling parts of CPES applications. Many examples of actor-oriented tools exist, including Simulink®, Ptolemy II and NI LabView®, some of which include native interfaces to numerical computing environments such as MATLAB®. To our knowledge, Ptolemy II is one of the most flexible actor-oriented environment, which has been used for CPES simulation by many previous studies, as outlined in Section 1. Additionally, due to its capabilities including aspect-oriented modeling [18], heterogeneous composition of a large number of MoCs, as well as support for web service implementations, Ptolemy II provides a standalone framework that can be used as a comprehensive evaluation platform.

6 Conclusion

In this paper, a model-based parametric approach for uncertainty analysis of middleware service architectures for smart grid applications has been presented. The DE models encompassed the entire data flow from sensors to application nodes, being processed through network and middleware fabrics. Using the Ptolemy II framework, we created an integrated design environment that supports Monte Carlo sampling of model parameters and subsequent execution of the simulation model using the generated parameter set. We have then carried out regression analysis to discover significant model parameters and quantified their degree of effect on the end-to-end run time.

The results show that for both middleware service architectures that have been evaluated, the maximum number of PMU streams for each area and the middleware concurrency level directly influence the maximum runtime of the end-to-end process. Data exchange triggered by iterations until convergence is a less effective parameter in determining the empirical worst-case run time given that only intermediate data are exchanged. This indicates that power applications can benefit from scalable middleware services to support monitoring and control applications with temporal requirements. It is key to communicate these results to power engineers to help calibrate the scale and configuration of distributed power applications for the future power grid and to assess the need for extended scalability analyses using a model-based design approach.

Acknowledgments

This work was supported in part by the TerraSwarm Research Center, one of six centers supported by the STARnet phase of the Focus Center Research Program (FCRP) a Semiconductor Research Corporation program sponsored by MARCO and DARPA.

References

[1] W. Jiang, V. Vittal, and G. Heydt, "A distributed state estimator utilizing synchronized phasor measurements," *Power Systems, IEEE Transactions on*, vol. 22, no. 2, pp. 563 –571, may 2007.

[2] G. Korres, "A distributed multiarea state estimation," *Power Systems, IEEE Transactions on*, vol. 26, no. 1, pp. 73–84, 2011.

[3] A. Gomez-Exposito, A. Abur, A. de la Villa Jaen, and C. Gómez-Quiles, "A multilevel state estimation paradigm for smart grids," *Proceedings of the IEEE*, vol. 99, no. 6, pp. 952–976, 2011.

[4] Y. Liu, J. M. Chase, and I. Gorton, "A service oriented architecture for exploring high performance distributed power models," in *Service-Oriented Computing*, ser. Lecture Notes in Computer Science, C. Liu, H. Ludwig, F. Toumani, and Q. Yu, Eds. Springer Berlin Heidelberg, 2012, vol. 7636, pp. 748–762. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-34321-6_59

[5] Y. Chen, Z. Huang, Y. Liu, M. Rice, and S. Jin, "Computational challenges for power system operation," in *Proceedings of the 2012 45th Hawaii International Conference on System Sciences*, ser. HICSS '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 2141–2150. [Online]. Available: http://dx.doi.org/10.1109/HICSS.2012.171

[6] L. Zhou and J. Rodrigues, "Service-oriented middleware for smart grid: Principle, infrastructure, and application," *Communications Magazine, IEEE*, vol. 51, no. 1, pp. 84–89, January 2013.

[7] J.-F. Martínez, J. Rodríguez-Molina, P. Castillejo, and R. De Diego, "Middleware architectures for the smart grid: survey and challenges in the foreseeable future," *Energies*, vol. 6, no. 7, pp. 3593–3621, 2013.

[8] I. Akkaya, Y. Liu, E. Lee, and I. Gorton, "Modeling uncertainty for middleware-based streaming power grid applications," in *Proceedings of the 8th Workshop on Middleware for Next Generation Internet Computing*, ser. MW4NextGen '13. New York, NY, USA: ACM, 2013, pp. 4:1–4:6. [Online]. Available: http://doi.acm.org/10.1145/2541608.2541612

[9] NIST, "Smart grid conceptual model, http://smartgrid.ieee.org/ieee-smart-grid/smart-grid-conceptual-model," 2014. [Online]. Available: http://smartgrid.ieee.org/ieee-smart-grid/smart-grid-conceptual-model

[10] "Apache ActiveMQ," Accessed: June 2015. [Online]. Available: http://activemq.apache.org/

[11] I. Gorton, A. Wynne, Y. Liu, and J. Yin, "Components in the pipeline," *Software, IEEE*, vol. 28, no. 3, pp. 34 –40, may-june 2011.

[12] I. Gorton, Y. Liu, C. Lansing, T. Elsethagen, and K. Kleese van Dam, "Build less code deliver more science: An experience report on composing scientific environments using component-based and commodity software platforms," in *Proceedings of the 16th International ACM Sigsoft Symposium on Component-based Software Engineering*, ser. CBSE '13. New York, NY, USA: ACM, 2013, pp. 159–168. [Online]. Available: http://doi.acm.org/10.1145/2465449.2465460

[13] I. Akkaya, Y. Liu, and I. Gorton, "Modeling and analysis of middleware design for streaming power grid applications," in *Proceedings of the Industrial Track of the 13th ACM/IFIP/USENIX International Middleware Conference*, ser. MIDDLEWARE '12. New York, NY, USA: ACM, 2012, pp. 1:1–1:6. [Online]. Available: http://doi.acm.org/10.1145/2405146.2405147

[14] C. Ptolemaeus, Ed., *System Design, Modeling and Simulation using Ptolemy II*, 2014. [Online]. Available: http://ptolemy.org/books/Systems

[15] P. Derler, E. Lee, and A. Vincentelli, "Modeling cyber-physical systems," *Proceedings of the IEEE*, vol. 100, no. 1, pp. 13 –28, Jan. 2012.

[16] J. C. Jensen, D. Chang, and E. A. Lee, "A model-based design methodology for cyber-physical systems," in *Wireless Communications and Mobile Computing Conference (IWCMC), 2011 7th International*, July 2011, pp. 1666 – 1671. [Online]. Available: http://chess.eecs.berkeley.edu/pubs/837.html

[17] E. A. Lee, "CPS Foundations," in *Proc. of the 47th Design Automation Conference (DAC)*. ACM, June 2010, pp. 737–742. [Online]. Available: http://chess.eecs.berkeley.edu/pubs/804.html

[18] I. Akkaya, Y. Liu, and E. A. Lee, "Modeling and simulation of network aspects for distributed cyber-physical energy systems," in *Cyber Physical Systems Approach to Smart Electric Power Grid*. Springer Berlin Heidelberg, 2015, pp. 1–23.

[19] "IEEE standard for synchrophasor measurements for power systems," *IEEE Std C37.118.1-2011 (Revision of IEEE Std C37.118-2005)*, pp. 1 –61, 28 2011.

[20] "Apache Spark," Accessed: June 2015. [Online]. Available: https://spark.apache.org/

[21] P. Zikopoulos, C. Eaton *et al.*, *Understanding big data: Analytics for enterprise class hadoop and streaming data*. McGraw-Hill Osborne Media, 2011.

[22] E. Latronico, E. Lee, M. Lohstroh, C. Shaver, A. Wasicek, and M. Weber, "A vision of swarmlets," *Internet Computing, IEEE*, vol. 19, no. 2, pp. 20–28, Mar 2015.

[23] A. Aleti, B. Buhnova, L. Grunske, A. Koziolek, and I. Meedeniya, "Software architecture optimization methods: A systematic literature review," *Software Engineering, IEEE Transactions on*, vol. PP, no. 99, p. 1, 2012.

[24] A. Koziolek, L. Happe, A. Avritzer, and S. Suresh, "A common analysis framework for smart distribution networks applied to survivability analysis of distribution automation," in *Software Engineering for the Smart Grid (SE4SG), 2012 International Workshop on*, June 2012, pp. 23–29.

[25] A. Faruque, M. Abdullah, and F. Ahourai, "A model-based design of cyber-physical energy systems," in *Design Automation Conference (ASP-DAC), 2014 19th Asia and South Pacific*. IEEE, 2014, pp. 97–104.