Adaptive Energy Forecasting and Information Diffusion for Smart Power Grids

Yogesh Simmhan*, Vaibhav Agarwal†, Saima Aman†, Alok Kumbhare†, Sreedhar Natarajan†, Nikhil Rajguru†, Ian Robinson‡, Samuel Stevens‡, Wei Yin†, Qunzhi Zhou† and Viktor Prasanna*
*Department of Electrical Engineering †Computer Science Department
‡Green Technologies Program
University of Southern California, Los Angeles CA 90089
Email: {simmhan, agarwalv, saman, kumbhare, sreedhan, nrajguru, imrobins, spsteven, weiyin, qunzhizh, prasanna}@usc.edu

I. Introduction

Smart Power Grids exemplify an emerging class of Cyber Physical Applications that exhibit dynamic, distributed and data intensive (D^3) characteristics along with an always-on paradigm to support operational needs. Smart Grids are an outcome of instrumentation, such as Phasor Measurement Units and Smart Power Meters, that is being deployed across the transmission and distribution network of electric grids. These sensors provide utilities with improved situation awareness on near-realtime electricity usage by individual consumers, and the power quality and stability of the transmission network.

One of the characteristic applications of Smart Grids is demand response optimization (DR). The goal of DR is to use the power consumption time series data to reliable forecast the future consumption profile for individual consumers, and to use this information to detect potential demand-supply mismatch. Further, such a detection should trigger load curtailment strategies that will use incentives communicated to the consumer to shape, shift or shed load during the predicted period of duress to avoid brownouts.

As part of the Los Angeles Smart Grid Demonstration project, we are investigating scalable software infrastructure to support DR applications in the largest public utility in the United States. The University of Southern California (USC) serves as a microgrid testbed to deploy and test the DR software components and algorithms, with the intent to scale these applications to a city of 1.4 million customers. Specifically, we address (1) the adaptive scalability required by the information integration pipeline to continuously ingest sensor data, and (2) the ensemble scaling required to train machine learned forecasting models on accumulated sensor data. The former utilizes our *Floe* continuous dataflow engine to scale on a Eucalyptus Private Cloud to dynamically meet application quality of service needs; the latter utilizes our *OpenPlanet* Hadoop application for ensemble training in a cluster. The integrated information and training models are accessed through a web portal for decision support to the USC microgrid operations and for information diffusion in the USC community for energy awareness.

In the rest of this paper, we present details of the Smart Grid DR application and portal (Section II), describe and analyze the novel features of the adaptive software architecture (Section III), and illustrate the demonstration scenario for the challenge (Section IV).

II. Smart Grid Demand Response Management

The Smart Grid DR application is decomposed into three phases: information ingest, data analytics, and information diffusion. Here, we describe their characteristics.

A. Information Integration Pipeline

The DR application uses a variety of information sources that characterize the Smart Grid or microgrid for enhanced situation awareness and analytics. These sources pass through an information processing pipeline that retrieves data from realtime sources, parses the response into a canonical data structure,

annotates the data tuple with semantics, and inserts the RDF triples into a semantic repository. Semantics help manage the information complexity of diverse entities that affect energy use, such as the power grid, electrical equipment, building, academic and facility schedules, organizational details, and weather.

The information integration pipeline needs to adapt to several forms of dynamism. Sensors are a primary data source in the microgrid and include power meters in individual buildings or floors, equipment and lighting reporting operational status, and Heating Ventilation Air Conditioning (HVAC) units emitting setpoint and ambient temperature. These collectively number in the order of 50,000 but the ones that are monitored at a point in time depend on the current DR application needs (e.g. proximity to a peak load time period may cause additional sources to be monitored). Further, the sampling rates of these sensors may vary from once every minute to once an hour depending on the electric load (e.g. night times are less interesting since the energy use is limited). Hence, the macro- and micro-scopic spatial and temporal scales of data acquisition are determined dynamically, with a corresponding need to scale the ingest up and down. Slow changing data about buildings and equipment installation arrive in bulk mode, with low frequency but large sizes. Occasionally, historical sensor data may also be loaded from archives. The pipeline needs to respond to these variations while meeting the latency/data freshness needs of the DR application for operations and to optimize resource usage.

B. Machine Learned Demand Forecasting Models

The DR application requires forecasting of power demand that is robust over time and evolves as the microgrid behavior changes. Hence, we adopt machine learned forecasting models that use indirect indicators for power usage prediction. Specifically, we use regression tree learning that predicts building and campus level power consumption at 15-minute and daily granularities. The model operates on features like temperature and humidity, academic semester, weekday/holiday, and building type to make the forecast.

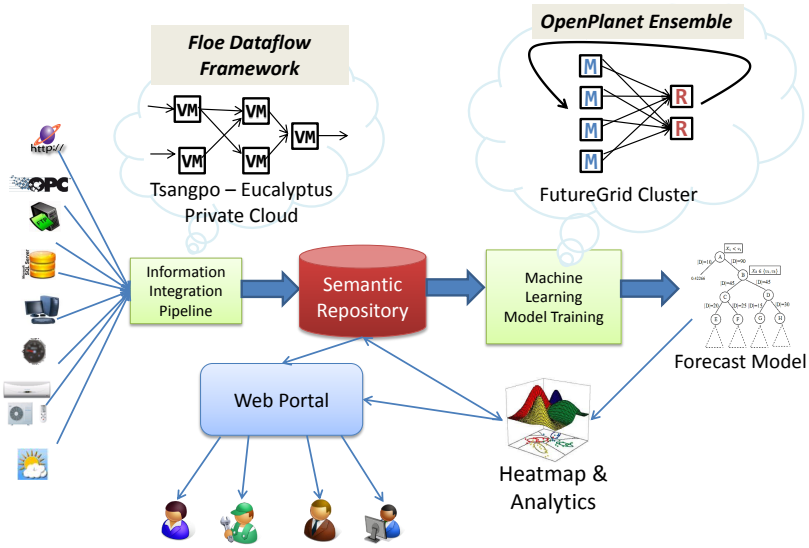
Constructing the regression tree model requires training it on historical time series data on these features and the power consumption. This data itself is extracted from the semantic repository being updated by the pipeline. We use 3 years worth of training data for the training; for the 170 buildings on campus and using 15-min time granularities, the training data is on the order of 20 Million tuples. In addition, distinct models can be constructed for different combinations of spatial collections (each building, collection of buildings, the whole campus), temporal granularities (15-min, 1-hour, 24-hour) and combinations of features to determine the ones that offer the best prediction accuracy. This requires the ability to perform ensemble runs of training that further need to be refreshed when newer data is accumulated. The training can happen in batch mode or on-demand when a new spatio-temporal combination is required – the larger models take on the order of several days to train on a single machine.

C. Information Diffusion Portal

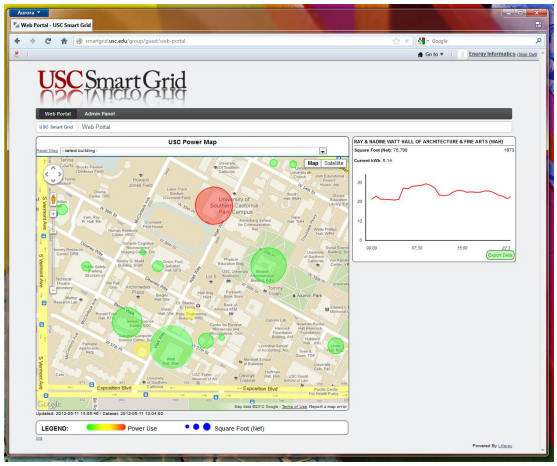
There are three primary types of information consumers within the USC microgrid – campus facility operations, data analysts and the USC faculty/staff/students. Operators visualize realtime information in the repository and energy forecasts to decide on operational changes such a initiating direct energy curtailment in specific buildings by changing its thermostat temperatures. Analysts evaluate the effectiveness of different forecasting models to select and configure appropriate ones for use. The USC public use the current energy profile of buildings, their carbon footprint and the forecast power loads to understand energy use within the campus and voluntarily take actions to limit their energy impact for sustainability. A web portal serves as a central service for this information diffusion. It offers a campus map interface, heatmap overlays, and graphical plots to explore the energy use and forecasts for different buildings and time periods (Figure (b)). It relies on data present in the repository and realtime predictions made using models that have already been trained.

III. Adaptive Architecture

Our scalable software architecture to support the D^3 needs of the DR application is centered on two frameworks, *Floe* and *OpenPlanet*, that we discuss here and analyze their innovative scalability features.



(a) Information Integration and Model Training



(b) USC Smartgrid Portal

A. *Information Ingest Pipeline using* Floe

"Floe" [1] is a continuous data flow framework that incorporates adaptability and adaptivity to dynamic application characteristics and the underlying infrastructure. Floe is used to model the information integration pipeline for DR. Floe goes beyond traditional workflow engines in using hybrid edge types between tasks that can refer to both file and streaming data for uniform composition of continuous as well as batch processing applications. Floe also allows a limited form of dynamic recomposition of the data flow at runtime to allow new data sources to be incorporated. The Floe runtime consists of a data flow coordinator for orchestration and a resource manager for provisioning. These are particularly suited to leverage the elasticity offered by public and private Cloud infrastructure. The resource manager acquires virtual machine (VM) instances on-demand from a Cloud provider for a data flow execution and the coordinator instantiates and activates the data flow on these. Elastic support for multi-cores ensures CPU resource for tasks can be incrementally scaled up and down to meet QoS needs. Adaptability knobs present in the coordinator allow the parallelism of the tasks to be raised or lowered to improve core utilization. QoS may be specified in terms of the latency for processing each unit of work and as a cost metric for bounding core-mins used. These features allow us to dynamically scale out the data ingest from a few hundred to over 50,000 sensors at data rates of every 5–15 minutes, ensuring that the required data freshness is maintained while optimizing VM resource utilization.

Floe offers novel properties of hybrid data models, continuous and batch execution, dynamic recomposition and autoscaling on Cloud resources. There is limited work on such hybrid data flow models [2][3] and few if any workflows allow runtime recomposition needed for dynamic applications. A constrained form of autoscaling is offered by providers like Amazon EC2 for simple transaction processing applications, but not for complex data flows.

B. *Regression Tree Learning using* OpenPlanet *MapReduce*

The regression tree algorithm is a compute and memory intensive algorithm. This limits its possibility to scale *up* to meet large training datasets. As an example, the algorithm implemented using the popular Weka machine learning library uses 7GB RAM for a 6 Million tuple training dataset, beyond which the

[1]Floe: Designing a Continuous Data Flow Engine for Dynamic Applications on the Cloud, Yogesh Simmhan, Sreedhar Natarajan, Alok Kumbhare and Viktor Prasanna, *Under Review*

[2]Daniel Zinn, Quinn Hart, Timothy M. McPhillips, Bertram Ludäscher, Yogesh Simmhan, Michail Giakkoupis and Viktor K. Prasanna, Towards Reliable, Performant Workflows for Streaming-Applications on Cloud Platforms, *CCGrid* , 2011

[3]Chathura Herath and Beth Plale, Streamflow - A programming model for data streaming in scientific workflows, *CCGrid*, 2010

time complexity increases super linearly due to virtual memory swapping. *OpenPlanet*[4] is our distributed implementation of the PLANET MapReduce algorithm [5] for training regression trees using Hadoop. OpenPlanet uses a form of iterative MapReduce, where each iteration constructs one level of the decision tree by performing a full data scan on the training data file present in HDFS. There can be three MapReduce job types that are scheduled in every iteration, depending on the size of the training data partition corresponding to each leaf node introduced in the previous iteration. For leaf nodes whose data partition fits in a host's memory, an InMemoryWeka Hadoop job extracts that data partition from the training file and builds the rest of that subtree using the Weka Machine Learning Java Library. For leaf nodes whose data partitions do not fit in memory, a Histogram Hadoop job samples the data for candidate split points followed by an ExpandNodes Hadoop job that evaluates the best split point for each leaf node to generate the next level of its children. We make two unique contributions that exhibit the scalability characteristic of OpenPlanet. One, we optimize HDFS block sizing and memory threshold to switch from ExpandNodes to InMemoryWeka that significantly improves the speedup of OpenPlanet. Two, we support ensemble runs of OpenPlanet to allow construction of models with different configurations while using the same training data.

Existing statistical and machine learning packages such as Weka and Matlab are restricted to a single machine implementation of regression tree which does not scale out. Apache Mahout which offers a scalable machine learning toolkit does not support the regression tree model.

IV. Demonstration

Our online demonstration of the DR Application will show case the scalability of the information integration pipeline using Floe and forecast model training using OpenPlanet through the portal interface. Figure (a) shows the architecture of the demonstration setup. Sensor, weather and building data from the USC campus will be ingested by the integration pipeline that is instantiated by Floe on our private Cloud, `Tsangpo`, running the Eucalyptus v2 fabric. This Cloud infrastructure located at USC provides seventeen nodes, each with 8-core AMD Opterons rated at 2 GHz each and 16 GB RAM, for a total of 136 available cores. Floe will scale out on this Cloud through VM resource acquisition, CPU core-scaling and task parallelism as the stream rates increase at runtime, and, conversely, scale in when the load is lighter. This dynamism in ingested sources will be visible through the portal that can query and present information from the repository, itself hosted on a server co-located with Tsangpo. The scalability metrics of latency, core allocation and utilization, and pending messages will be plotted on an administrative portal. The OpenPlanet ensemble runs will be launched on the `Sierra` cluster at UCSD that is part of FutureGrid project. These nodes have an 8-core Intel Xeon 2.5GHz CPU and 32GB memory. A Hadoop environment is configured on these nodes on-demand using MyHadoop. The portal interface will allow a data analyst to select the parameter space for the ensemble training and launch the runs. Training data will be staged from USC to HDFS on the Sierra cluster nodes. The trained models will be moved back to the repository server and be used for forecasting power consumption for USC campus buildings. Multiple runs can be launched as an ensemble.

Acknowledgment

This material is based upon work supported by the United States Department of Energy under Award Number DEOE0000192 and the National Science Foundation under Award CCF-1048311. The views and opinions of authors expressed herein do not necessarily state or reflect those of the US Government or any agency thereof. We thank FutureGrid for resources provided to run these OpenPlanet experiment, under NSF Award 0910812.

[4]Scalable Regression Tree Learning on Hadoop using OpenPlanet, W. Yin, Y. Simmhan and V. Prasanna, *MapReduce Workshop*, 2012

[5]PLANET: Massively Parallel Learning of Tree Ensembles with MapReduce, Biswanath Panda, Joshua S. Herbach, Sugato Basu, Roberto J. Bayardo, *VLDB* 2009