Simulation and Big Data Challenges in Tuning Building Energy Models

Jibonananda Sanyal, *Member, IEEE,* and Joshua New, *Member, IEEE*

***Abstract*—EnergyPlus is the flagship building energy simulation software used to model whole building energy consumption for residential and commercial establishments. A typical input to the program often has hundreds, sometimes thousands of parameters which are typically tweaked by a buildings expert to "get it right". This process can sometimes take months. "Autotune" is an ongoing research effort employing machine learning techniques to automate the tuning of the input parameters for an EnergyPlus input description of a building. Even with automation, the computational challenge faced to run the tuning simulation ensemble is daunting and requires the use of supercomputers to make it tractable in time. In this paper, we describe the scope of the problem, particularly the technical challenges faced and overcome, and the software infrastructure developed/in development when taking the EnergyPlus engine, which was primarily designed to run on desktops, and scaling it to run on shared memory supercomputers (*Nautilus*) and distributed memory supercomputers (*Frost* and *Titan*). The parametric simulations produce data in the order of tens to a couple of hundred terabytes. We describe the approaches employed to streamline and reduce bottlenecks in the workflow for this data, which is subsequently being made available for the tuning effort as well as made available publicly for open-science.**

***Keywords*—Building energy modeling, parametric ensemble, simulation, big data.**

I. Introduction

In the United States, buildings consume 39% of all primary energy and about 73% of total electricity of which less than 8% is met by renewable energy resources [1]. To meet the various environmental, social, and financial challenges, the U.S. Department of Energy has set aggressive goals for improving energy efficiency not just in the US buildings sector, but also the industrial and transportation sectors.

Building energy modeling is an important tool for architects and engineers to estimate the energy usage for buildings. The major challenge in designing models for either new buildings or retrofits is to realistically model specific types of buildings and then replicate them across various scales and geographic locations [2, 3, 4, 5]. Each building is unique and unlike the manufacture of consumer goods, there is no single definition of a model that describes the properties for all buildings of a type. As such, it becomes important to be able to accurately model the various physical and material properties of individual buildings to be able to make a realistic assessment of its energy footprint. Having said that, DOE provides a set of standard reference building models [6] that are used nationwide and are representative of the U.S. building stock. These building models are used for normative analysis to determine how policy changes would affect energy consumption in the US, determine tax trade-offs, design building codes, trade-off incentives, and evaluation of the effect of climate change on buildings.

While the simulation engines are not perfect, the outcome of such analyses depends a lot on the correct representation of the physical building in the model. The typical order of business is to employ a building energy modeling expert who painstakingly tunes the various parameters manually. The outcome of the tuning exercise is always a function of the experience and expertise of the modeler. The tuning exercise is very time consuming and sometimes has been observed to take months to "get right".

"Autotune EnergyPlus" [7] is an ongoing research project that aims to employ computers to automatically calibrate the various input model parameters for building retrofit packages (figure 1). In this paper, we describe the Autotune approach and focus on elaborating the technical challenges faced and overcome. Specifically, we focus on elaborating the necessity, experience, and lessons learned using different architectures of supercomputing systems, and the management of large amounts of generated simulation data.

The rest of the paper is divided into the following parts: we begin with describing the Autotune methodology, followed by a description and elaboration of the computational challenge. We follow this by a description of the big-data challenges such as the amount of data generated, its management, storage, and analysis. This is followed by a description of the simulation workflow and the evolving overarching software architecture.

II. Autotune Methodology

There are about 20 major software tools that perform building energy modeling with each having certain strengths and weaknesses. The primary whole building simulation engine supported by the U.S. Department of Energy is EnergyPlus (E+), which is roughly 600,000 lines of FORTRAN code. A typical input to the simulation engine can contain over 3,000 input parameters for a regular residential building which must be tuned to reflect the building properties. Good sources of data or handy reference values are not easily available for many of these parameters. Experiments by [cite Dewitt] has established that there are often mismatches between product specifications as per the ASHRAE handbook and the manufacturer's supplied data. Again, a finished building often deviates from its design and as a result, the model of the building no longer matches that of the real building. For existing buildings, retrofit

Jibonananda Sanyal and Joshua New are with the Building Technologies Research and Integration Center, Oak Ridge National Laboratory, Oak Ridge, TN, 37830 USA e-mail: {sanyalj, newjr}@ornl.gov

packages are often implemented leading to substantial energy savings. In most cases, there is significant return on investment within a reasonable break-even period for retrofits in existing structures.

All these reasons compel us to adjust a building energy model to match measured data, the tweaking of which is typically performed manually by a building modeling expert. This is a painful, time-consuming, and an error-prone process. The manual tuning approach is also highly subjective and non-repeatable. A very large number of building models start out with the DOE reference buildings (which are most representative of the U.S. building stock) and go through the manual adjustment of geometry, HVAC properties, insulation, fenestration, infiltration properties, etc. Because of the notoriety of the process, energy modeling is expensive and is done primarily on large projects. Unless there are savings in the assessment, professionals in the field will not adopt it as a viable operational practice.

The goal of the "Autotune" project [7] is to save the time and effort spent by energy modelers in adjusting simulation input parameters to match the measured data by providing an easy button (figure 1). To achieve this objective, an initial model of the building and sensor data for a time period is provided to the Autotune software stack, which then spins off the trained machine learning agents [8] and returns a tuned model of the building. At the heart of the Autotune capability is a set of multi-objective machine learning algorithms [8] that characterize the effect of individual variable perturbations on EnergyPlus simulations and adapts the given model to match its output to the supplied sensor data (figure 2). Once machine learning agents are tuned and available, the computational cost of tuning a typical user's building model will be reduced to matter of a few hours using widely available desktop computational resources. This paper focuses on the supercomputing and big-data challenges. Readers interested in the machine language algorithms are referred to [8].

The system is currently in a late stage of development and is being demonstrated to match a subset of 250 sensors of 15-minute resolution data in a heavily instrumented residential building in addition to DOE's standard reference building models [6] for a medium sized office, a warehouse, and a stand-alone retail building. Further, the simulations comprise of three vintages (old, new, and recent) of the DOE commercial reference buildings across 16 different cities representing the different ASHRAE climate zones and sub-zones.

III. Computational Challenge

The computational space for performing parametric analysis is combinatorially large. Brute-force exploration for 156 of 3000+ input parameters using only minimum, average, and maximum would require 5×10^{52} EnergyPlus simulations and 2×10^{28} lifetimes of the known universe when running on *Titan*... for a single building! For this reason, several numerical algorithms (discussed below) are used which allow convergence to a solution without brute-forcing every combination of inputs and intelligent experimental design can guide parametric analysis to a much smaller sample size.

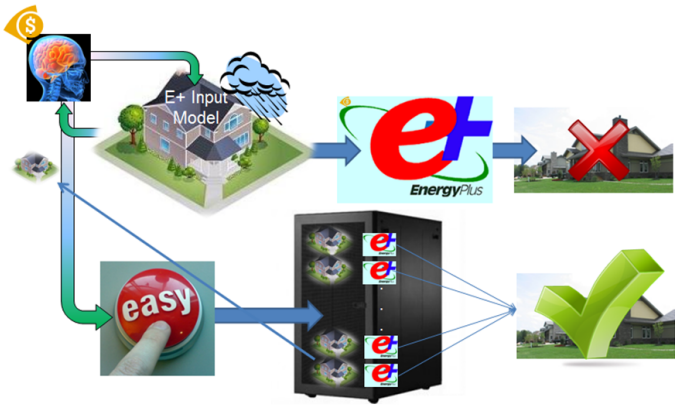

Fig. 1. Autotune workflow for E+ building energy models as a cost-effective solution for generating accurate input models. The conceptual figure illustrates how a conventional calibration process is expensive and inaccurate. The 'easy' button illustrates the 'Autotune' pathway for calibration using computational methods.

A. Parametric Sampling and Input Generation

We worked with building technology experts to pick only the most important parameters, thereby reducing the count to 156 for a residential building. These are parameters that are most commonly used by energy modelers. The parameters were further ranked into three importance categories. Domain experts further defined realistic bounds and incremental step size values for the parameters. In addition, various meta-parameters were determined which allow several individual parameters to be varied as a function of a single input parameter.

Even with ~156 input parameters and three levels of incremental values for each of the simulations, we are looking at 10 million simulations. Each individual simulation takes roughly 8 minutes which translates to 2 million compute hours accounting for overheads. Using Oak Ridge National Laboratory's *Titan* supercomputer (currently ranked as the fastest supercomputer in the world), this would take 2 months of calendar time to just run the simulations, let alone manage the data, perform machine learning, and subsequent analysis. Effective, scalable methods to sample the input space is crucial.

We use the expert's grouping of important parameters to divide the sampling space into groups of relative importance. We have also used low-order Markov ordered simulations to determine variables with a monotonic effect on sensor data that can reliably be interpolated to estimate impact of a given input. The source of variance of individual variables is being used to guide sampling rates of the more sensitive inputs. Finally, experts in multi-parameter optimization will be investigating computational steering algorithms to determine the optimal sampling strategy for the remaining space beyond the brute-force sampling of higher order Markov chains of Monte Carlo simulations.

Each simulation requires its own input file. The individual input files are generated using two strategies:

1) A program in Perl: We wrote a program in Perl that reads a template input file with 156 special markers for variable

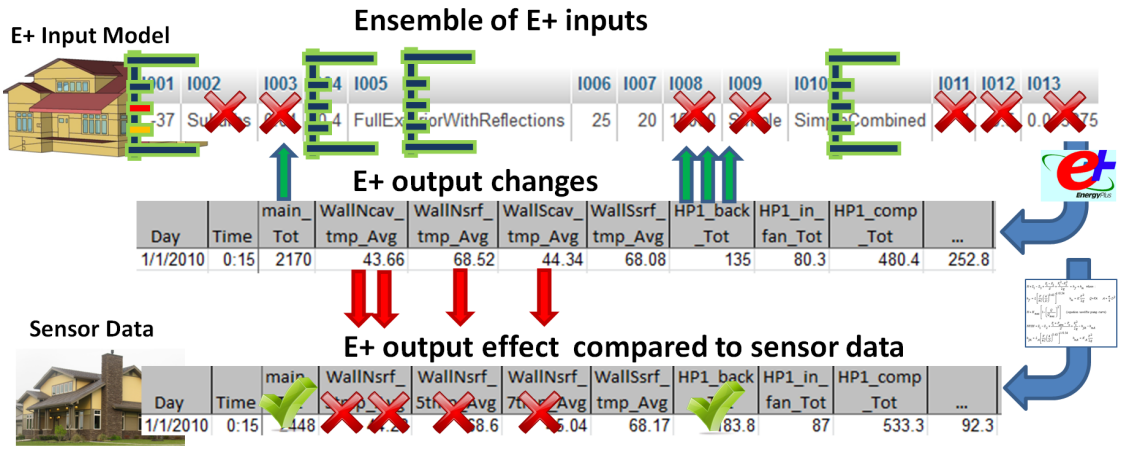

Fig. 2. A virtual building model (top) in software space and a real building (bottom) having sensors, when viewed as vectors of numbers, allows a mathematical mapping between vector spaces for direct comparison between sensed world state and sensor space. The layer in the middle illustrates the output of EnergyPlus as a vector of numbers and how output is affected by the ensemble of parametrc input distributions (green bars). The red and green arrows illustrate parameters being tuned down or up during the tuning process. The crosses and checks illustrate progressive acceptability of tuning as the machine learning progresses.

values and another file that specifies value ranges and step sizes. Different sampling algorithms such as maximum and minimum bounds check, Markov Order 1, or Markov Ordering 2 were programmed as modules. When invoked, they would sample the input space and fill in values for the special markers in the template and write it out as individual input files.

2) E+ supplied parametric preprocessor: The EnergyPlus simulation tool comes with a parametric preprocessor program which takes an input file embedded with parameters and generates the individual files. The parametric values are supplied within the file where one must list all the possible values, with repetition, for nested block designs. The algorithm scan one line at a time from each set of parameters to generate the individual input files. For example, if we had 4 variables with each having 3 levels, we would have $4 \times 3 = 12$ lines of values for each variable adding a total of $12 \times 4 = 48$ additional lines of input. While convenient for small sampling spaces, it becomes cumbersome very fast. We used this approach initially for the residential building but quickly switched to using our own Perl program. For the DOE reference buildings, we used the parametric preprocessor mainly because we were using meta-parameters to define other parameters and each building type was replicated for three vintages (old, new, and recent) across 16 locations in the United States, which made the management of bulky parametric descriptions somewhat easier.

In summary, a total of 8 million simulations are planned for following building types:

- Residential: ~5 million
- Medium office: ~1 million
- Stand-alone retail: ~1 million
- Warehouse: ~1 million

A set of ~1 million input files (1M each for retail, warehouse, and medium-office buildings) were generated on a 16-core shared-memory compute server using four parallel processes which took about 18 hours to complete.

B. Supercomputing

In the initial stage of the effort, a Qt based tool named "EplusGenius" was developed that leveraged the power of idle desktops to run EnergyPlus simulations and upload the results to a database. It was engineered to provide the choice to run only after office hours and on weekends. While effective in harnessing unused computational power from a network of computers, the computational requirements to achieve a reasonable time-to-solution necessitate the use of HPC resources. Time on several supercomputing systems have been competitively awarded and used to demonstrate the scalability of our algorithms and code for the massively parallel leadership-class computing systems. Systems include the 1024-core shared memory *Nautilus*, 2048-core *Frost*, and 299,008-core *Titan* which is currently the world's fastest supercomputer at 20 petaflops. The *Nautilus* machine has 4 TB of global shared memory visible to every processor on the system. *Titan* (and *Frost* too) has a distributed memory model with each node having 16 processors and 32 GB of RAM.

While in theory this is an embarrassingly parallel problem (parametric ensemble) and should be easy to parallelize, various complicating factors make this difficult to scale in practice. First, EnergyPlus was developed as a desktop application and was not supercomputer ready. In a typical execution trace for a single simulation, a sub-directory and a large number of files (12+ files amounting to 100+MB) are created. Second, constant moving and soft-linking of the files are done as the simulation workflow executes. Third, an annual simulation with 15-minute output of 82 channels is 35MB in size and currently needs to be stored on disk for later analysis. In other words, the entire process is particularly I/O intensive, which complicates the scalability of parallel execution on supercomputers. We attempt to mitigate these issues in many ways.

A branch of the source code for the engine has been made open-source; however, writing a wrapper for over 600,000 lines of code to streamline I/O for use on supercomputers is outside

the scope of this work. We treat E+ as a black-box and use it to simulate our 8 million runs. In addition, the workflow depends on a number of additional executables, the source-code of which is not available.

The current job submission scripts were designed to pick up a set of input files and execute them in parallel on the number of cores requested and moving the output to a directory when complete. Initial experiments on the supercomputing systems delivered very poor scaling performance which were expectedly traced to the bandwidth and Lustre file-system saturation with the frequent number of large I/O requests. Most supercomputing systems use a parallel file-system named Lustre. Software communicates with the file-system through the metadata server which instructs the hardware to access the right set of disks. The file-system is typically located in its own server rack, separate from the computation nodes and connected via a high-speed, high-bandwidth communication network. The biggest bottleneck was determined to be the communication with the Lustre metadata server and the storage targets.

In order to alleviate the filesystem bottleneck, we made use of the memory-based virtual file-system which gave us more than two orders of magnitude improvement over using the Lustre filesystem. In addition, we block-partitioned and streamlined our input and output mechanisms. To outline the steps performed:

1) EnergyPlus comes with a large number of supporting executable programs and associated files. A typical E+ simulation is essentially a workflow where multiple executables are invoked with each producing temporary files ingested by subsequent programs. We minimized the engine's folder structure to include only the binaries and libraries required for our simulations, modified the execution scripts to use relative paths, and compressed the minimized file structure to make it ready to be loaded into the virtual filesytem.
2) In an effort to reduce the number of input files fetched, we performed a pre-processing step in which we grouped the inputs into blocks of 64 simulations each and packed them into compressed tarballs. This reduces the number of files fetched by a factor of 64 and reduces size by $\sim$60%.
3) For *Nautilus*, individual jobs can be placed on individual processors using the 'dplace' command. A heavily modified job submission script allows us to request a specific number of cores and provide a count of the number of batches to run. For example, a request for 256 cores with 90 batches would start out by picking out $256/64 = 4$ blocks of compressed input files and the simulation engine, and then parallelly extract them to the virtual file-system. Each core then executes a simulation (using an explicit 'dplace' command which runs a job on a core). After completion, the data is moved to the physical file-system and the next batch of 4 compressed files is loaded. This is repeated 90 times.
4) For *Frost* and *Titan*, the virtual file system (*tmpfs*) is shared-memory visible within a node. Additionally, the systems do not use 'dplace' for direct placement of individual jobs on a processor. We wrote a message passing interface (MPI) program that would make each node load the engine and a block of 64 runs into its shared-memory. Since each node has 16 processors and there are 64 files in a compressed block of inputs, the node makes four iterations to run all 64 simulations.
5) Once a block of simulations is complete, the output files are added to a tarball and moved to disk. This is typically about 1.5 – 2.4 GB in size depending on the type of simulation. This also reduces the number of I/O interactions with the Lustre filesystem by a factor of 64. The virtual file system is cleaned and prepared for the next block to run. We experimented with compression of the output in memory and its transfer to disk but found that the idling of other processors was unacceptable while a subset of the processors performed the compression.

We have been able to complete a batch of 1 million E+ simulations for the warehouse building using *Nautilus* and *Frost* in under 2 weeks of continual execution (along with other users running other jobs on the systems). The theoretical runtime using average job execution metrics was estimated at about 7.6 days for the batch of 1 million simulations. We were also able to complete the 1 million Standalone Retail simulations in $\sim$2 weeks.

Using *Titan*, we have been able to achieve excellent scalability completing $\sim$250,000 EnergyPlus simulations using 65,536 processors in under 45 minutes, and expect our code to scale to be able to use all 299,008 cores.

IV. Big data challenges

The output from these simulations produce a file between 35 MB to 70 MB each constituting anywhere from 80 to a 180 output variables at 15 minute intervals for a whole year. With 8 million simulations underway, a number of challenges emerge.

A. Data storage

We are looking at approximately 270 TB of raw data when all simulations are complete. We have estimated that this can be compressed down to about 70 TB, which is still a large amount of data. This is the size of simulation data prior to any operations or storage performed as part of the analysis processes. There are certain logical partitions in the data such as type of building simulation, its vintage, location, and also the parameter sampling and grouping strategy which helps us in breaking down the data management space. While many database-related technologies have been tested and explored, effectively storing this data for quick retrieval and analysis remains a challenge.

We have explored a number of databases including MySQL, noSQL/key-value pair, columnar, and time-series database formats for simulation/sensor data and currently implement a hybrid solution with a part of the summarized data entered in a database and readily accessible for querying and analysis while and the raw data being fetched on demand. This data

is currently provided with no guarantees since the entire data queryable with an assured turnaround time (a solution similar to a hadoop stack) for queries is currently infeasible.

We currently have an 80 TB Synology disk station which primarily serves as a data repository with fast access to individual files and are in the process of acquiring additional hardware for some of the analysis needs. Although additional hardware is being procured, it is geared towards adding compute capabilities for analysis but is not geared towards assured turn-around times for queries.

B. Data transfer

There is a learning process in planning for such large data efforts. One of the things we learned is the cost of transferring data from the supercomputing infrastructure to our data repository. Although we have a 1 Gbps Ethernet backbone, initial data transfer speeds using ftp between computers that housed the output data and the data repository ranged from 7 MB/s to 12 MB/s indicating that it would take over three months to transfer all data for just the warehouse simulations. This was a multi-hop transfer because of various firewall rules within the laboratory. We overcame the firewall rules and were able to sustain 40 MB/s peer-to-peer transfer rates.

Most large supercomputing facilities have dedicated data transfer mechanisms such as *GridFTP* or *BBCP*, which are tools that optimize and parallelize data transfer. We are currently using GridFTP and are moving data at consistent speeds of around 120 MB/s. This works out to about 4 days for transferring the output of 1 million warehouse simulations which is a huge improvement over the initial estimate of over three months.

C. Data analysis

Many domain experts and students are interested in various ways to analyze this valuable dataset. Moving the data for post-processing calculations is very expensive. The trend these days when working with big-data is to move the computation to the data. Map-reduce and distributed computing methods apply the same philosophy. When dealing with simulation data, especially as large and individually discrete as our simulations are, an added step of post-processing to calculate summarizing statistics is very beneficial. We have architecture our code to not only output data at 15 minute timestep intervals but also calculate hourly, daily, and monthly averages.

We are also working on engineering in-line processing of the data across simulations to calculate different kinds of metrics including those that can be used for sensitivity analysis. Typically, a batch of simulations is run and calculation of different metrics across simulations while the data is still in memory will help us save on expensive IO operations. In addition, techniques to coalesce calculated statistics across multiple batches to reflect the state of the entire class of simulations is also underway.

V. Evolving system architecture

Figure 3 illustrates the software architecture of the system. Various parts of the system architecture are still evolving since many of the supercomputing and big-data challenges require us to revisit standard approaches to data management. We are working towards a scalable software framework that allows one to design massive building energy simulation experiments and allows different software agents to access the data and perform analysis. In addition to calculating summary statistics in-stream, we are also adding the capability of running machine learning agents on the data as they are generated.

The final product of the effort is envisioned to be a variant on software as a service. For Autotune's non-HPC side, users of the system will be able to visit a web-site and submit an autotuning job. Since the DOE reference buildings are used as a starting point for many users, we expect to provide these as templates. Techniques have been identified that will help provide customization features that will help users tweak these to more closely represent their buildings. The system will perform all the necessary computation and machine learning and come back with an answer that best fits the objective of the tuning. For Autotune's HPC side, advanced optimization techniques for mining large data will be ready to analyze new problems across domains. Execution traces and provenance of models and parameters previously tuned will be provided to analyze and identify the critical components in typical tuning processes. The Autotune process by itself is domain agnostic. The technology can be easily customized to tune smaller models and configured to run on single desktops, network of workstations, shared-memory supercomputers, distributed-memory supercomputers, as well as deployed on the cloud.

In addition to facilitating research activities and Autotune user access, the AutotuneDB approach of providing the entire simulation data for full public access is anticipated to be increasingly utilized by web services, researchers, companies/entrepreneurs, and home-owners to mine for their specific question of interest. The silent availability of this database has already triggered several interested parties (primarily involving researchers and future doctoral candidates) and dove-tails with the growing trend of increased transparency in government programs, as evidenced by popular White House initiatives such as GreenButton (for utility data) and many other data-sharing initiatives related to building performance (energy reporting now required in some cities) and high performance building databases.

VI. Conclusion

The successful completion of the presented Autotune effort underway at the Oak Ridge National Laboratory will go a long way in alleviating the tedious task of tuning a building energy model to sensor data. The employment of machine learning agents performs a multi-objective optimization (see [8]) of the input parameters to provide a solution that best matches the input sensor data. The refinement and dimension reduction of the input parameter space to 156 important ones identified by the experts helps to reduce the computational space. Further, various methods to scientifically sample the input parameter space helps to reduce the computational space.

One of the questions asked is how the Autotune workflow allows faulty operation detection in its workflow. There is a large

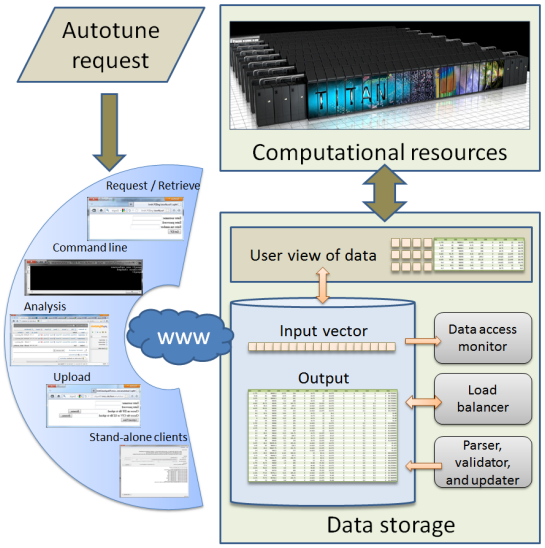

Fig. 3. Architecture of the Autotune software-as-a-service product.

body of work for Fault Detection and Diagnostics (FDD) with fault-injection and detection of equipment and whole-building systems currently being conducted that is outside the scope of this work. However, Autotune's latest unpublished work has been in automatic calibration of equipment schedules (in addition to properties) which should capture (and thus facilitate correcting) faulty/offline equipment, half-open dampers, and other faulty operations provided there is relevant sensor data on which to calibrate. Therefore, the normal Autotune workflow is expected (though hasn't been fully verified in ongoing work) to capture faulty and fixed operations throughout the span of time for which relevant sensor data is available. Since any time period can be handled, two models could be calibrated through two Autotune workflows, one for "faulty operation" and one for "proper operation" to allow energy-saving (or other) comparisons between them.

The main objective of the paper is to highlight the technical challenges faced in simulating a very large set of simulations using an engine that has been developed for desktop applications, and in the process, generating a large amount of data. We expect that lessons learned and software developed will be useful for researchers who intend to run large ensembles and perform data mining of very large data. use a similar engine and plan to run a large number of parametric simulations. We also hope that some of the challenges faced and lessons learned in analyzing, moving, and managing large data across hardware platforms will provide beneficial insight and help to researchers in planning such endeavours.

Lastly, we also expect that the open availability of the parametric simulation datasets for the standard DOE reference buildings will directly benefit the building sciences community.

ACKNOWLEDGEMENT

This work was funded by field work proposal CEBT105 under the Department of Energy Building Technology Activity Number BT0201000. We would like to thank Amir Roth for his support and review of this project. We would like to thank our collaborators which include Mr. Richard Edwards and Dr. Lynne Parker from The University of Tennessee, Dr. Aaron Garrett from Jacksonville State University, and Mr. Buzz Karpay from Karpay Associates. This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725. Our work has been enabled and supported by data analysis and visualization experts, with special thanks to Pragnesh Patel, at the NSF funded RDAV (Remote Data Analysis and Visualization) Center of the University of Tennessee, Knoxville (NSF grant no. ARRA-NSF-OCI-0906324 and NSF-OCI-1136246).

Oak Ridge National Laboratory is managed by UTBattelle, LLC, for the U.S. Dept. of Energy under contract DE-AC05-00OR22725. This manuscript has been authored by UT-Battelle, LLC, under Contract Number DEAC05-00OR22725 with the U.S. Department of Energy. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes.

REFERENCES

[1] U.S. Dept. of Energy, *Building Energy Data Book*. D&R International, Ltd., 2010. [Online]. Available: http://buildingsdatabook.eren.doe.gov/docs%5CDataBooks%5C2008_BEDB_Updated.pdf

[2] M. Deru, K. Field, D. Studer, K. Benne, B. Griffith, P. Torcellini, B. Liu, M. Halverson, D. Winiarski, M. Rosenberg *et al.*, "US Department of Energy Commercial Reference Building Models of the National Building Stock," 2011.

[3] R. Briggs, R. Lucas, and Z. Taylor, "Climate classification for building energy codes and standards: Part 1–development process," *ASHRAE Transactions*, vol. 109, no. 1, pp. 109–121, 2003.

[4] ——, "Climate classification for building energy codes and standards: Part 2–zone definitions, maps, and comparisons," *ASHRAE Transactions*, vol. 109, no. 1, pp. 122–130, 2003.

[5] IECC 2009 and ASHRAE 90.1-2007, *Energy Code Climate Zones*. [Online]. Available: http://resourcecenter.pnl.gov/cocoon/morf/ResourceCenter/article/1420

[6] K. Field, M. Deru, and D. Studer, "Using DOE Commercial Reference Buildings for Simulation Studies," 2010.

[7] J. R. New, J. Sanyal, M. S. Bhandari, and S. S. Shrestha, "Autotune e+ building energy models," *5th National SimBuild of IBPSA-USA*, 2012.

[8] R. E. Edwards, J. New, and L. E. Parker, "Predicting future hourly residential electrical consumption: A machine learning case study," *Energy and Buildings*, 2012.