CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 2014; **26**:2122–2133
Published online 31 March 2014 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/cpe.3267

SPECIAL ISSUE PAPER

Calibrating building energy models using supercomputer trained machine learning agents

Jibonananda Sanyal[1,*,†], Joshua New[1], Richard E. Edwards[2] and Lynne Parker[3]

[1]*Oak Ridge National Laboratory, One Bethel Valley Road, PO Box 2008, MS-6324, Oak Ridge, TN 37831-6324, USA*
[2]*Amazon.com, Inc., Wainrright (SEA 23), 535 Terry Ave N., Seattle, WA 98109, USA*
[3]*The University of Tennessee in Knoxville, Min H. Kao Building, Suite 401, 1520 Middle Drive, Knoxville, TN 37996, USA*

SUMMARY

Building energy modeling (BEM) is an approach to model the energy usage in buildings for design and retrofit purposes. EnergyPlus is the flagship Department of Energy software that performs BEM for different types of buildings. The input to EnergyPlus can often extend in the order of a few thousand parameters that have to be calibrated manually by an expert for realistic energy modeling. This makes it challenging and expensive thereby making BEM unfeasible for smaller projects. In this paper, we describe the 'Autotune' research that employs machine learning algorithms to generate agents for the different kinds of standard reference buildings in the US building stock. The parametric space and the variety of building locations and types make this a challenging computational problem necessitating the use of supercomputers. Millions of EnergyPlus simulations are run on supercomputers that are subsequently used to train machine learning algorithms to generate agents. These agents, once created, can then run in a fraction of the time thereby allowing cost-effective calibration of building models. Published 2014. This article is a US Government work and is in the public domain in the USA.

Received 5 November 2013; Revised 21 January 2014; Accepted 25 February 2014

KEY WORDS: building energy modeling; supercomputer; parametric ensemble; machine learning; calibration; big data

1. INTRODUCTION

In the year 2010, buildings consumed 41% of the primary energy (up from 39% in 2006 [1]) constituting a larger share than either transportation (28%) or industry (31%) [2]. This is approximately $400 billion and up from $220 billion in 2006. Buildings consumed 74% of all electricity and 34% of all natural gas produced in the United States thereby contributing 40% of the carbon dioxide emissions in the United States [2]. The Department of Energy's (DOE's) Building Technologies Office has a significant stake in improving the energy footprint and efficiency of the buildings sector for economic, social, and environmental benefits.

The accurate modeling of buildings can serve as an important tool in understanding the energy footprint for designing new buildings as well as retrofitting existing buildings for energy savings. The large number of existing buildings that do not employ energy efficient technologies present a low-hanging fruit that could significantly and cost-effectively contribute to energy savings across the entire country by the application of retrofit packages. Primary to this objective is the realistic modeling of the buildings that accurately reflect potential energy savings for different scenarios [3–6].

*Correspondence to: Jibonananda Sanyal, Oak Ridge National Laboratory, One Bethel Valley Road, PO Box 2008, MS-6324, Oak Ridge, TN 37831-6324, USA.
†E-mail: sanyalj@ornl.gov

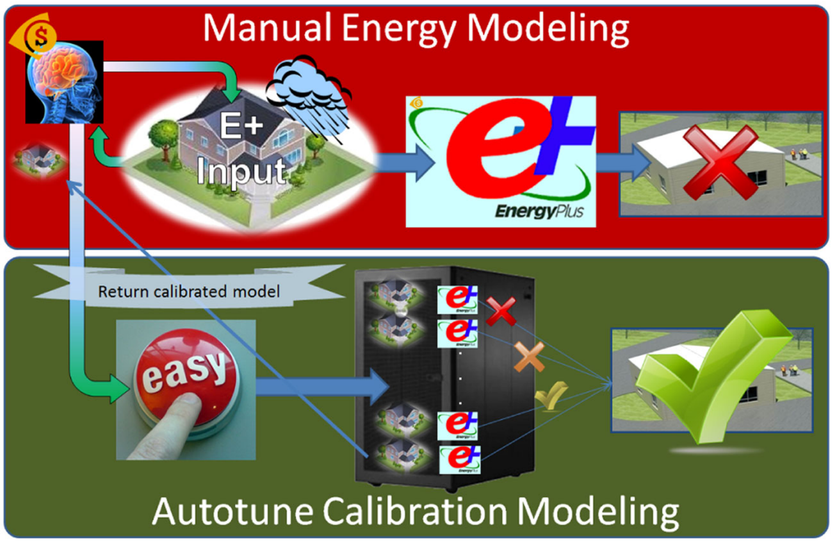

Figure 1. The business-as-usual approach for model calibration requires significant time from skilled experts, rendering simulation-based decisions infeasible for most building efficiency projects. Autotune uses intelligent algorithms to remove cost by automatically calibrating an energy model to utility bill data.

Each building is unique, and the model representation of the building tends to be a one-off activity. The accurate modeling of all building properties is laborious and requires expertise. In the 'business-as-usual' model, experts manually adjust different input parameters until satisfactory results are obtained. The time required for this can sometimes be in the order of a few months. Clearly, the cost of such an activity makes building energy modeling (BEM) unfeasible for smaller buildings.

While buildings may be one-off, they still must comply to building code. DOE has standard reference models of buildings that are used nationwide and are representative of the US building stock [7]. 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. To a large extent, this addresses the challenge of accurately modeling and scaling the building types across American Society of Heating, Refrigerating, and Air-Conditioning Engineers (ASHRAE) climate zones [3, 4, 6] and projecting how specific policies or retrofit packages would maximize return on investment with subsidies through federal, state, local, and utility tax incentives, rebates, and loan programs.

This paper describes the 'Autotune' methodology [8, 9] being developed at the Oak Ridge National Laboratory, which automatically calibrates a building energy model using supplied utility data for building retrofit purposes. This is achieved by using trained machine agents that are generated from large parametric simulations run on supercomputing systems using machine learning algorithms. Millions of parametric simulations are run for residential and commercial building types across various climate zones for training these machine agents.

The development of an autotuning capability [9] (Figure 1) to intelligently adapt building models or templates to building performance data would significantly facilitate market adoption of energy modeling software, aid in accurate use cases such as the effective retrofit strategies for existing buildings, and promote Department of Energy's Building Technologies Program's goals of increased market penetration for energy modeling capabilities.

2. AUTOTUNE METHODOLOGY

EnergyPlus (E+) is DOE's whole building energy simulation tool. It currently consists of approximately 600,000 lines of Fortran code and has multiple algorithms for simulating thermal heat flow of a building, occupancy schedules, equipment performance curves, water heating, lighting, weather interaction, and so on. Given a building description, including such things as the thermal conductance and specific heat of every material in every wall assembly, E+ can generate outputs such as whole-building energy electrical consumption or temperature at a given location. E+ can contain over 3000 input parameters for a regular residential building that must be tuned to reflect the building properties. Besides tuning, another challenge is obtaining quality data or representative reference values for the different material properties. Various experiments [10] have established that there are significant mismatches between the supplier's product specifications and those provided by the ASHRAE handbook. There are often deviations in material properties of the product against those on its label and as specified by the supplier. These and various construction factors cause the finished building to deviate from the original model of the building. Retrofit packages are useful for existing buildings because such efforts provide a significant return on investment for existing buildings.

To be able to reasonably model the aforementioned scenarios, a building energy model must be adjusted to match measured data. This matching is typically performed manually by a building modeling expert and is a non-repeatable, time-consuming, expensive, and laborious process. A very large number of building models start out with the DOE reference buildings (which are most representative of the US building stock) and go through the manual adjustment of geometry; heating, ventilation, and air conditioning properties; insulation; fenestration; infiltration properties; and so on. The high cost of building energy model generation and calibration makes it unfeasible to apply to smaller projects thereby creating a gap in the energy assessment for a large majority of smaller buildings. 'Autotune' [9] provides a methodology to cost-effectively perform building energy model calibration providing practitioners in the field with an option to apply BEM to smaller buildings.

The goal of the 'Autotune' project 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). An initial model of the building is provided to Autotune along with a set of measured sensor data. Autotune then spins off the trained machine learning agents and returns a tuned model of the building. Early experiments [11] showed that even with a subset of manually tuned parameters, Autotune was 60% as accurate in an overnight tuning process as two man-months from experts over two calendar years.

The individual trained machine agents are generated by performing machine learning on large parametric simulations for the different classes of standard DOE buildings [7] across different climate zones. At the core of the Autotune methodology is a set of multi-objective machine learning algorithms that characterize the effect of individual variable perturbations on E+ 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 is reduced to matter of a few hours using widely available desktop computational resources.

The system is currently being demonstrated to match a subset of 250 sensors of 15-min resolution data in a heavily instrumented residential building in addition to DOE's standard reference build-

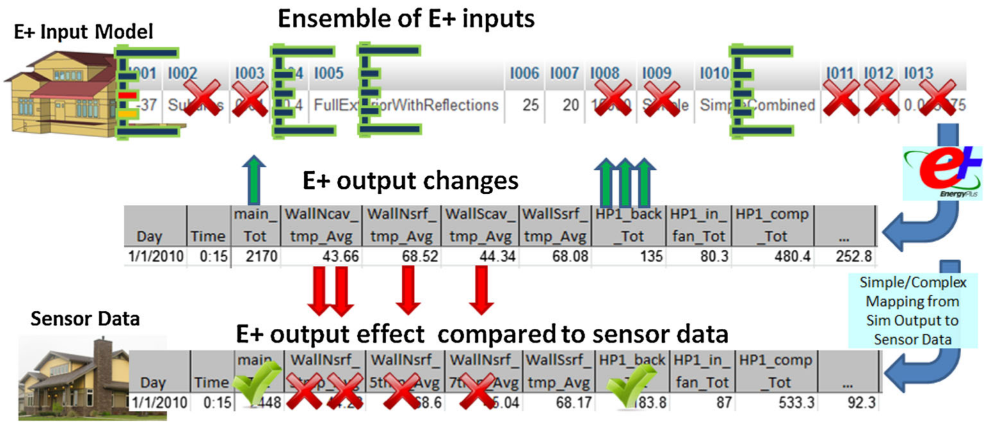
E+ Input Model
Ensemble of E+ inputs
I001 I002 I003 I004 I005 I006 I007 I008 I009 I010 I011 I012 I013
FullExteriorWithReflections 25 20 Simple Combined
E+ output changes
Day Time main_Tot WallNcav_tmp_Avg WallNsrf_tmp_Avg WallScav_tmp_Avg WallSsrf_tmp_Avg HP1_back_Tot HP1_in_fan_Tot HP1_comp_Tot ...
1/1/2010 0:15 2170 43.66 68.52 44.34 68.08 135 80.3 480.4 252.8
Simple/Complex Mapping from Sim Output to Sensor Data
Sensor Data
E+ output effect compared to sensor data
Day Time main_Tot WallNsrf_tmp_Avg WallNsrf_5th_Avg WallNsrf_7th_Avg WallSsrf_tmp_Avg HP1_back_Tot HP1_in_fan_Tot HP1_comp_Tot ...
1/1/2010 0:15 2448 68.6 5.04 68.17 183.8 87 533.3 92.3
EnergyPlus

Figure 2. Comparison of two models in an ensemble (top) and the corresponding sensitivity analysis of changes in simulation output (middle) is compared with sensor data from a real building (bottom) in order to allow optimized calibration of input parameters so simulation output matches measured data.

ing models [7] for a medium-sized office, a warehouse, and a stand-alone retail building. Further, the simulations comprise 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.

2.1. Parametric space

EnergyPlus requires about $\sim$ 3000 input parameters. The computational requirements for performing parametric analysis on $\sim$ 3000 E+ inputs would require 1×10^{138} lifetimes of the known universe on the world's fastest supercomputer. Several numerical algorithms and intelligent experimental design are used that allow convergence to a solution without brute forcing every combination of inputs. Even with intelligent optimization, simulation runtime forces consideration of advanced techniques with parametric sampling, high performance computing (HPC), and machine learning to make calibration of software models computationally feasible.

2.2. Parametric input sampling

Building technology experts who routinely perform calibration of BEMs analyzed the various inputs and picked 156 most important parameters in their experience for a heavily instrumented residential building. The modelers further ranked these into three importance categories and defined realistic bounds and incremental step size values for the parameters. Furthermore, various meta-parameters were determined that allow several individual parameters to be varied as a function of a single input parameter.

Even with $\sim$ 156 input parameters and three levels of incremental values for each of the simulations, we are looking at 10 million simulations for one building that translates to about two million compute hours for just the simulations alone, let alone time required for management of the data, machine learning, and subsequent analysis. Effective, scalable methods to sample the input space are crucial.

Besides grouping of the sample space by 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.

In summary, a total of eight million simulations are planned of which about five million are for multiple residential homes and about one million each for medium offices, warehouses, and stand-alone retail buildings.

2.3. High performance computing

Time on several supercomputing systems was 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 the 299,008-core *Titan* that runs at 20 petaflops. The *Nautilus* machine has 4 TB of global shared memory visible to every processor on the system, while the other machines have a distributed memory model. It is worthwhile to mention that bulk of the machine learning and tuning efforts has been performed on Nautilus, which is an XSEDE resource.

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, E+ 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 carried out as the simulation workflow executes. Third, an annual simulation for warehouses with 15-min output of 82 channels is 35 MB in size and currently needs to be stored on disk for later analysis. For other types of buildings, the output data are much larger in number of variables and, consequently, file size.

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.

The particularly I/O intensive nature of the software engine delivered very poor scaling performance, which was expectedly traced to the bandwidth and Lustre file-system saturation. In order to alleviate the filesystem bottleneck, we made use of the memory-based virtual file-system that gave us multiple orders of magnitude improvement. In addition, we block-partitioned and streamlined our input and output mechanisms as outlined in the succeeding text:

Step 1: EnergyPlus has 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 required for our simulations, heavily optimized the execution scripts to reduce I/O, and compressed the minimized file structure for quick loading into the virtual filesytem. In adition, we compiled a statically linked version of the main simulation engine [12].

Step 2: To reduce I/O, 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 the size by $\sim 60\%$.

Step 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 that runs a job on a core). After completion, the data are moved to the physical file system, and the next batch of four compressed files is loaded. This is repeated 90 times. Table I illustrates the scaling performance on *Nautilus*. The increased shared memory access penalty in the case of jobs larger than 128 cores can be attributed to the memory abstraction layer and the heavily modified job submission script that itself added a significant overhead for the individual dispatching of jobs to their target cores.

Step 4: For *Frost* and *Titan*, the virtual file system (*tmpfs*) is shared memory visible within a node. Additionally, these are Cray systems and do not use 'dplace' for direct placement of individual jobs on a processor. We wrote a Message Passing Interface (MPI) program that makes each node load the engine and a block of 64 runs into its shared memory. Because 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.

Step 5: Once a block of simulations is complete, all the output files are added to a tarball and moved to disk reducing I/O significantly. This ranges from 1.5 to 5.3 GB in size depending on the type of simulation. The virtual file system is cleaned and prepared for the next block to run.

Table I. EneryPlus run times for a residential building on the *Nautilus* supercomputer, inclusive of disk write time.

Procs	Wall-clock time (h:mm:ss)	E+ sims	Avg. E+ time (mm:ss)
32	0:02:08	32	2:08
64	0:03:04	64	3:04
128	0:04:11	128	4:11
128	0:34:24	1024	4:18
256	1:25:17	2048	10:40
512	0:18:05	1024	9:02

E+, EnergyPlus.

Table II. EnergyPlus run times scaling chart for a commercial building on the *Titan* supercomputer, inclusive of disk write time.

Procs	Wall-clock time (mm:ss)	E+ sims	Avg. E+ time (mm:ss)
16	18:14	64	04:34
32	18:19	128	04:35
64	18:34	256	04:39
128	18:22	512	04:36
256	20:30	1024	05:08
512	20:43	2048	05:11
1024	21:03	4096	05:16
2048	21:11	8192	05:18
4096	20:00	16384	05:00
8192	26:14	32768	06:34
16384	26:11	65536	06:33
65536	44:52	262144	11:13
131072	68:08	524288	17:02

E+, EnergyPlus.

With *Titan*, we have observed weak scalability up to 131,072 cores and expect the trend to continue if we run even larger batches. Tables I and II illustrate the scalability observed on the *Nautilus* and *Titan* machines.

About 270 TB of raw data, compressible to approximately 70 TB (which is still very large), are expected to be generated from these simulations. This is the size of simulation data prior to any operations 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 that helps us in breaking down the data management space. While many database-related technologies have been tested and explored, including MySQL, noSQL/key-value pair, columnar, and time-series database formats, effectively storing this data for quick retrieval and analysis remains a challenge. We 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. These data are currently provided with no guarantees because the entire data queryable with an assured turnaround time (a solution similar to a hadoop stack) for queries are currently infeasible.

3. CALIBRATION APPROACHES

There are many different simulation calibration approaches in the literature. All approaches can be grouped into manual procedural calibration methodologies and semi-automated statistical calibration procedures. The manual calibration methodology generally involves constructing a manual audit process for incorporating environmental information into the engineer's simulation parameter calibration process. The statistical method involves engineers or domain experts selecting the most important simulation parameters, model outputs, environmental measurements for calibration, and mappings between simulation outputs and environmental measurements, which results in a semi-automated calibration process. The calibration methodology employed in Autotune uses feature selection to determine the most important sensors with respect to the overall most important calibration target. In addition, surrogate modeling is employed for the construction of large-scale surrogate models that can facilitate calibration and reduce the calibration runtime to a range manageable on commodity hardware for a casual user base.

3.1. Surrogate and inverse modeling

Surrogate generation or meta-modeling typically uses a few classical statistical techniques—ordinary least squares linear regression, multivariate adaptive regression splines, Kriging, and radial basis functions. In a general sense, any simulation engine can be conceived of as a function that maps simulation inputs to simulation outputs. Surrogate models are a way to take a set of input and output data from a simulation engine and learn to predict output from the input by effectively approximating the simulation engine function. Inverse models, on the other hand, can be used to predict simulation engine inputs from the simulation engine outputs (or corresponding sensor data) by effectively approximating the inverse of the simulation engine function. We discuss both use cases in the context of this project.

Fitting a surrogate model that uses the key inputs allows researchers to calibrate simulations around these key inputs much more quickly than by solving the physics equations encoded in the entire simulation engine. This speedup comes with a trade-off in modeling accuracy, so surrogate models should only be used for quick initial exploration of model options in the search space before transitioning to the full simulation engine. The importance of surrogate modeling for a given calibration task is proportional to the simulation engine runtime, surrogate model runtime, and the accuracy of the surrogate model for the current calibration task. Given a surrogate model's dependence on an adequate training set, care should be taken in selection of the proper inputs, (building) models, and simulation outputs. When the examples used for training the surrogate model are not representative of the current calibration task, the training database and surrogate model are considered 'brittle' with a proportional inaccuracy to be expected from surrogate estimations. As discussed in Section 2.3, $\sim$ 300 TB of simulation data were created from ensemble runs to serve as training data. The models used were a standard residential model as well as DOE reference buildings for a medium-sized office, a warehouse, and a stand-alone retail building because they are the most popular building types in the USA by either number of buildings or total square footage. The simulation engine used in this study is E+, so we will refer to the surrogate model thus derived as approximately E+ ($\sim$ E+). Different surrogate modeling algorithms were scaled to HPC systems [13] and were tested on 156 building inputs with 80–90 outputs for a subset of the simulation data. Lasso regression and feed forward neural networks were able to attain an average per-variable cross-validation error of 1% with 4 se of runtime rather than 3–7 min of whole simulation runtime [14]. In practice, this is used to quickly prune the search space.

The goal of model calibration could be realized with a sufficiently accurate inverse model. This is difficult to achieve in practice for many reasons: there is often not a 1-to-1 correspondence between available sensor data and simulation output, a mathematical mapping from sensor data to output variables can be difficult or intractable, sensor data are imperfect (missing values, saturation, sensor drift), brittleness due to available training data, and inaccuracy of current inverse models to capture temporal variability. Few simulation engines permit one to 'run the simulation backwards', but inverse modeling allows creation of this inverse function. This is carried out by solving an inverse optimization problem using actual measured data, which is more efficient than directly solving the inverse optimization problem with the exact software simulation. Lasso regression with alternating direction method of multipliers versus Bayesian regression with a Markov Network called a Gaussian graphical model was able to reliably identify each input parameter's average value but with spread that could not be captured with the temporal dynamics of building performance [15]. Because we are using E+, we refer to the inverse model so derived as inverse E+ (!E+). In practice, this inverse function allows creation of potentially close building models to function as seed points for the search algorithm to perform calibration.

3.2. *Sensor-based energy modeling*

Sensor-based energy modeling (sBEM) was a proposed data-driven methodology to begin addressing BEM's inaccuracy of both model inputs and complex algorithmic encoding of simulation functionality. With the traditional BEM approach involving an average of approximately 3000 simulation inputs and 600,000 lines of Fortran code, building scientists and practitioners often lack sufficient information to properly encode information required for the traditional BEM process, such as: physical properties of all materials in a building, detailed weather data at that location, gap between the as-designed and as-built building, actual use of the building (e.g., occupant behavior), and scalable algorithmic interactions for new products or assembly methods. sBEM can be viewed as a hybrid between 'forward' modeling and 'inverse' modeling approaches. This data-driven approach assumes that the sensor data provide a viable model for the entire building: the 'forward' component. This means that a building equipped with sensors creates sensor data that define the state of the building, including weather, building envelope, equipment, and operation schedules over time. Through the application of machine learning algorithms, an approximation of the engineering model is derived from sensor data statistically: the 'inverse' component. A machine learning case study was employed using one of our heavily instrumented field demonstrate buildings to determine the best algorithms for predicting future whole building energy consumption [16] from 144 sensors collecting data every 15 min as well as the best subset of sensors.

This sBEM approach is significantly different from standard domain knowledge and engineering practice because of the use of sensor data and machine learning techniques rather than inputs (often based on averages) and manually derived algorithmic approximations of physics. Additional simulation hybridization work is needed to bridge the gap between sBEM's complex sensor-level terms such as 'reduce heat flows through the surface at the framing in terms of units of British thermal units per hour per square-foot degrees Fahrenheit' to the simpler and actionable conceptual approach of 'add insulation for a higher R value'. In today's business environment, the traditional approach is necessary for cost-effective design or retrofit to achieve energy efficient buildings. Potential sBEM advantages include increased specificity of calculations to the needs of a given set of inputs and question, increased accuracy, significantly reduced runtime, and the potential that manual development of simulation algorithms may no longer prove necessary.

4. MACHINE LEARNING AGENTS

Machine Learning Suite (MLSuite) is a collection of several different machine learning algorithms and software implementations into a single high performance computing HPC-enabled system [17] for mining on large data and has general extensibility to search and optimization problems across multiple domains.

From a software perspective, MLSuite is able to easily coopt the implementation of dozens of algorithms across different software packages through the use of an eXtensible Markup Language (XML) interface. MLSuite currently supports algorithms from Matlab and related toolboxes, the statistical programming language R, libSVM (an open source library for support vector machines), and others. It also supports single-line parametric definitions of algorithm-specific parameters, input orders, supervisory signals, and so on. This allows large-scale testing to identify the best algorithm instance for a given problem, saves error-prone reimplementation of existing algorithms, allows integration of industry-leading software packages, and supports frequent updates for those packages. The collection of assimilated machine learning algorithms relevant to this scalability study are shown in Table III. MLSuite also allows arbitrary definitions of sequential jobs and parallel tasks so that 'ensemble learning' can be instantiated; by allowing the output of a specific algorithm to become the input of another, entire networks of machine learning agents can be constructed. This is used in the hierarchical mixture of experts and Fuzzy C-means examples where clusters of similar simulations are identified and then individual machine learning agents are trained only a specific cluster in order to generate a local expert. This type of data space partitioning is beneficial for scalable problem solving on big data.

Table III. Different types of learning systems in use and their acronyms for reference in the scalability tables.

Learner type	Acronym
Linear regression	REG
Feed forward neural network	FFNN
Hierarchical mixture of linear regression experts	HME-REG
Nonlinear support vector regression	SVR
Hierarchical mixture of feed forward neural network experts	HME-FFNN
Hierarchical mixture of least-square support vector machine experts	HME-LSSVM
Fuzzy C-means clustering with local feed forward neural networks	FCM-FFNN
All learning systems combined (suite run)	All

From a hardware perspective, MLSuite has the capability to operate on supercomputers or a network of individual computers. A modifiable Python script is used to launch an MLSuite file via the HPC job submission system. If all required resources are in place, the MLSuite job will continue to run parametric ensembles of machine learning agent tasks until completion and then provide a multi-metric summary at the end that quantifies how well each instance performed. Although MLSuite was extended to allow submission to multiple supercomputers, it currently performs no load balancing for the defined set of tasks. MLSuite also supports a network of individual computers where IP addresses can be listed and authentication information provided to run agents on individual computers. This has been invoked reliably on $\sim$ 40 Linux machines in a university computer lab using maximal niceness settings.

MLSuite is used in several specific ways as part of the Autotune project [9]:

First is the exploration of a more data-driven sBEM [18] through machine learning prediction of building performance as a function of observed data using statistical methods rather than complex, physics-based, engineering algorithms [16].

Second, usage is speeding up annual E+ simulations by creating surrogate simulation engines that trade off accuracy for order of magnitude speedup [14].

Third, creating inverse simulation engines that use sensor data (corresponding to simulation output) to predict an anticipated set of simulation input parameters, which can act as seed points for further exploration of the true inputs.

Fourth, it is being used as a sensor quality assurance mechanism. It both fills in missing values and detects outliers in time-series data by intelligently predicting the anticipated value as a function of temporal patterns for a specific data channel in combination with spatial patterns of other data channels (e.g., inferential sensing).

The cross-validated prediction accuracy and scalability of machine learning can vary dramatically by algorithm, algorithmic parameters, input features, data input order, supervisory signal for the question under study, and other dimensions. It is important to take these into account when leveraging HPC resources in order to increase effective use of the hardware resources. For compute-bound machine learning tasks, traditional HPC or networked computers were used, whereas memory-bound machine learning tasks were run on a shared-memory HPC architecture. Tables IV, V, and VI illustrate the runtime and scalability of many algorithms investigated, primarily by use of the shared-memory *Nautilus* supercomputer.

The major limitations for big data mining is the computational and memory requirements of current algorithms. Some algorithms traditionally require the entire data set to reside in memory so partitioning the problem or using out-of-core techniques becomes important. One could also modify the algorithm itself for enhanced scalability. As an example, we consider least squares support vector machines and kernel ridge regression. Contributions have been made to the theory behind these algorithms to allow these $O\left(n^3\right)$ algorithms to be approximated with a onefold cross-validation

Table IV. MLSuite scalability of actual run times by algorithm.

Learner	Jobs	1 core [hours]	4 cores [hours]	90 cores (1-core per job) [secs]	120 cores (4 cores per job) [secs]
REG	10	0.008	0.001	0.273	0.226
FFNN	60	1.370	0.688	82.295	82.295
SVR	10	0.047	0.040	17.097	14.344
HME-REG	90	1.329	1.172	53.17	140.590
HME-FFNN	540	482.903	245.633	21200.617	30376.072
HME-LSSVM	90	1.257	0.536	50.288	64.348
FCM-FFNN	420	395.850	197.908	1979.273	2770.711
All	1220	533.092	271.160	22924.789	33241.826

Table V. MLSuite average run time by algorithm using 1 and 4 cores per job.

Learner	Avg. Job Runtime (1 core) [secs]	Avg Job Runtime (4 cores) [secs]
REG	0.273	0.226
FFNN	82.295	41.302
SVR	17.097	14.344
HME-REG	53.170	46.863
HME-FFNN	3533.436	1687.560
HME-LSSVM	50.288	21.449
FCM-FFNN	395.855	197.908
All	1637.485	810.776

Table VI. Scalability improvements by replacing exact leave-one-out (eLOO) with approximate (aLOO) for least squares support vector machines (LS-SVM) and kernel ridge regression (KRR) involving minimal trade-off in accuracy with example sizes going from 8192 to 1 million [solution time in seconds].

# Examples	2^{13}	2^{14}	2^{15}	2^{16}	2^{17}	2^{18}	2^{19}	2^{20}
eLOO	4.43	35.25	281.11					
aLOO-LSSVM	1.3	2.6	5.32	10.88	22.45	47.41	101.36	235.83
aLOO-KRR	0.54	1.06	2.14	4.3	8.55	17.28	35.39	68.22

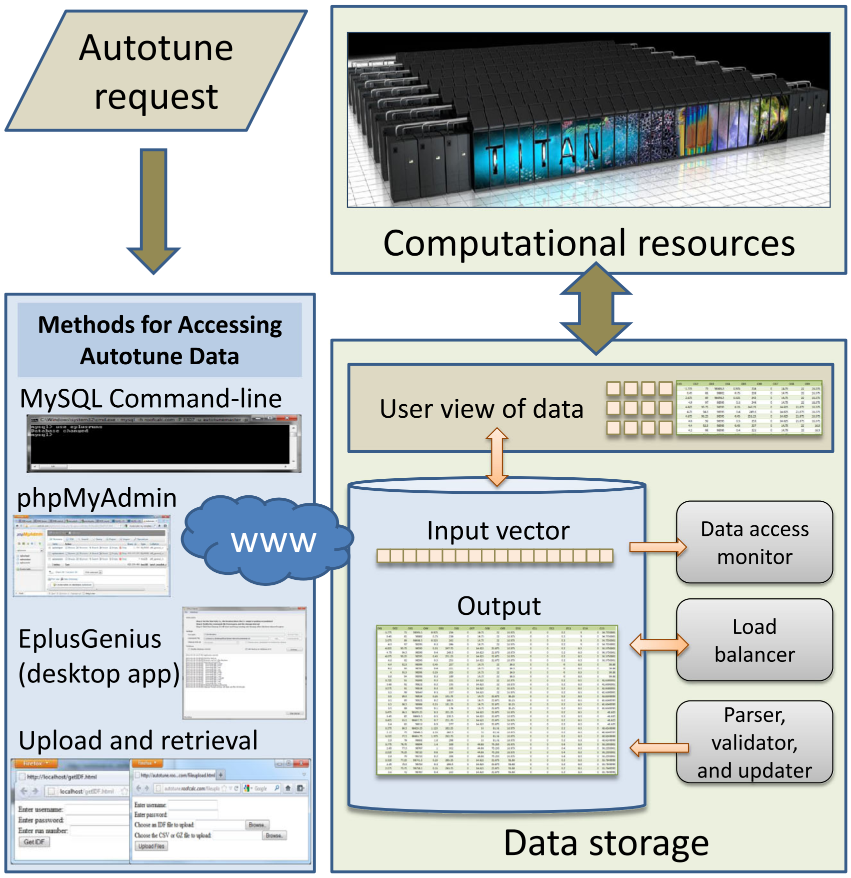

Figure 3. A portion of the high performance computing-generated parametric ensemble of building simulations is made publicly available through several Autotune software-as-a-service mechanisms at http://rsc.ornl.gov/autotune.

method with runtime $O(n \log n)$ that requires only a very small trade-off in accuracy for typical data sets [13]. This allows application of these algorithms to much larger data sizes in a computationally tractable amount of time as seen in Table VI.

5. PUBLIC FACING PORTAL

The parametric simulations run by supercomputers have been uploaded to a centralized database to allow public access to these data (Figure 3). Several tools have been developed for easily defining and running parametric E+ simulations, compressing data, and sending to a centralized server. The data storage and access software has been architected as a distributed, heterogeneous, client-server framework. Figure 3 shows the various software components and methods for accessing the data: a web-based input file reconstructor, command-line access for MySQL queries, phpMyAdmin for GUI-based interaction with a data subset, a webpage for uploading simulation data, and a software tool named 'EplusGenius' that leverages the power of idle desktops to run E+ simulations and upload of the results to a database. In addition to web-access to different tools, a means of providing a user with an easy way to develop a building model using available templates for different kinds of buildings is in progress. This, along with user provided sensor data, will help in making Autotune available as a web-service.

6. CONCLUSIONS

The successful completion of the Autotune effort 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 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.

The paper highlights the scientific contribution of automating BEM calibration and 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. We also hope that some of the insight gained in analyzing, moving, and managing large data across hardware platforms will provide beneficial insight and help researchers in planning such endeavours. Lastly, we also expect that the open availability of the parametric simulation data sets for the standard DOE reference buildings will directly benefit the building sciences community.

ACKNOWLEDGEMENTS

This work was funded by field work proposal CEBT105 under the DOE Building Technology Activity Number BT0201000. We thank Amir Roth for his support, and our collaborators, Dr. Aaron Garrett from Jacksonville State University and Mr. Buzz Karpay from Karpay Associates. We appreciate the support from Pragneshkumar Patel at the Remote Data Analysis and Visualization Center of the University of Tennessee, Knoxville (NSF grant no.ARRA-NSF-OCI-0906324 and NSF-OCI-1136246) and Willam Renaud at the Oak Ridge Leadership Computational Facility (OLCF). Oak Ridge National Laboratory is managed by UTBattelle, LLC, for the US DOE under contract DE-AC05-00OR22725. This manuscript has been authored by UT-Battelle, LLC, under Contract Number DEAC05-00OR22725 with the US DOE. 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.: Silver Spring, MD, 2010.
2. U.S. Dept. of Energy. *Building Energy Data Book*. D&R International, Ltd.: Silver Spring, MD, 2011.
3. Briggs RS, Lucas RG, Taylor ZT. Climate classification for building energy codes and standards: part 1–development process. *ASHRAE Transactions* 2003; **109**(1):109–121.
4. Briggs RS, Lucas RG, Taylor ZT. Climate classification for building energy codes and standards: part 2–zone definitions, maps, and comparisons. *ASHRAE Transactions* 2003; **109**(1):122–130.
5. Deru M, Field K, Studer D, Benne K, Griffith B, Torcellini P, Liu B, Halverson M, Winiarski D, Rosenberg M, Yazdanian M, Huang J, Crawley D. US Department of Energy Commercial Reference Building Models of the National Building Stock, *NREL Report No. TP-5500-46861*, National Renewable Energy Laboratory: Golden, CO, 2011.
6. IECC 2009 and ASHRAE 90.1-2007. *Energy code climate zones*, 2009.
7. Field K, Deru M, Studer D. *Using DOE commercial reference buildings for simulation studies*, 2010.
8. New JR, Sanyal J, Bhandari MS, Shrestha SS. Autotune E+ building energy models. *5th National SimBuild of IBPSA-USA*, Madison, Wisconsin, 2012; 270–278.
9. Sanyal J, New J, Edwards R. Supercomputer assisted generation of machine learning agents for the calibration of building energy models. *Proceedings of the Conference on Extreme Science and Engineering Discovery Environment: Gateway to Discovery*, ACM, San Diego, CA, 2013; 22.
10. de Wit MS. Uncertainty in predictions of thermal comfort in buildings. *Ph.D. Thesis*, The Delft University, Delft, Netherlands, 2001.
11. Garrett A, New JR, Chandler T. Evolutionary tuning of building models to monthly electrical consumption, Technical paper DE-13-008. *In Proceedings of the ASHRAE Annual Conference and ASHRAE Transactions 2013*, Vol. 119, Denver, CO, 2013; 89–100. part 2.
12. Big Ladder Software. EPx, a modified version of energyplus v. 7, 2013.
13. Edwards RE, New JR, Parker LE. Approximate l-fold cross-validation with least squares SVM and kernel ridge regression. *IEEE 12th International Conference on Machine Learning and Applications (ICMLA13)*, Miami, Florida, USA, 2013.
14. Edwards RE, New JR, Parker LE. Constructing large scale energyplus surrogates from big data. *ORNL Internal Report ORNL/TM-2013/46861*, 2013.
15. Edwards RE, New JR, Parker LE. Estimating building simulation parameters via Bayesian structure learning. In *Proceedings of the 2nd International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications (BigMine '13)*. ACM: New York, NY, USA, 2013; 31–38. DOI: 10.1145/2501221.2501226, http://doi.acm.org/10.1145/2501221.2501226.
16. Edwards RE, New JR, Parker LE. Predicting future hourly residential electrical consumption: a machine learning case study. *Energy and Buildings* 2012; **49**:591–603.
17. Edwards RE, Parker L. MLSuite final report fy2012, Oak Ridge National Laboratory, TN, 2013.
18. Edwards RE, Parker L. Sensor-based energy modeling final report fy2012, Oak Ridge National Laboratory, TN, 2012.