A Data-Driven Approach to Nation-Scale Building Energy Modeling

Andy S. Berres Member, IEEE
Computational Sciences and Engineering
Oak Ridge National Laboratory
Oak Ridge, Tennessee
berresas@ornl.gov

Brett C. Bass
Grid-Interactive Controls
Oak Ridge National Laboratory
Oak Ridge, Tennessee
bassbc@ornl.gov

Mark B. Adams Member, IEEE
Verification Technologies
Oak Ridge National Laboratory
Oak Ridge, Tennessee
adamsmb@ornl.gov

Eric Garrison
Department of Electrical Engineering and Computer Science
University of Tennessee, Knoxville
Knoxville, Tennessee
egarris4@vols.utk.edu

Joshua R. New Senior Member, IEEE
Grid-Interactive Controls
Oak Ridge National Laboratory
Oak Ridge, Tennessee
newjr@ornl.gov

Abstract— **In 2019, 125 million U.S. residential and commercial buildings consumed \$412 billion in energy bills. These buildings currently consume 40% of the nation's primary energy, 73% of electricity, 80% of energy during peak electric grid use, and responsible for 39% of greenhouse gas emissions [14].**

Urban-scale building energy modeling has grown significantly in the past decade, allowing individual campuses or communities of buildings to be modeled, simulated, and cost-effective solutions for intelligent management to be identified and implemented. While traditionally limited to individual counties and usually less than 2,000 buildings, the Automatic Building Energy Modeling (AutoBEM) software suite has been developed to process unconventional, nation-scale data sources to generate unique OpenStudio and EnergyPlus models of each building. Through the use of High Performance Computing (HPC) resources, every U.S. building has been simulated. This paper showcases the data layout, node partitioning, algorithmic approaches, and analytic results that were used to create, share, and analyze 124.4 million U.S. building models.

I. Introduction

Buildings consume vast amounts of energy and produce a large portion of the world's emissions. In 2019, buildings consumed 35% of all energy use and contributed 38% of emissions world-wide [21]. The United States alone consumed about 15% of all energy used world-wide, at only 4.25% of the world population [17]. Heating and cooling are major contributors to world-wide energy consumption [1]. Residential heating alone contributes to half of all residential energy use, and 11% of all energy use. It is crucial to gain a better understanding of how much energy is currently consumed in order to determine how much energy could be saved through improvements in Heating, Ventilation, and Air Conditioning (HVAC) systems, insulation, and other building properties.

Building energy simulation plays an important role in assessing the status quo, and understanding which building improvements can make the biggest impact in energy savings. They can also be a helpful tool in understanding how energy use will change with urban population growth [4], or in response to climate change [5].

Traditionally, building energy simulation is performed for a single building or a small number of building on a desktop computer or a small cluster. In recent years, there have been efforts to move towards urban scale modeling, which is already enough to push a small cluster to its limits [35]. National-scale modeling has not been attempted so far.

In this work, we present the parallelization of a new building energy simulation workflow to simulate 125 million U.S. buildings on the Theta supercomputer at Argonne Leadership Computing Facility (ALCF). We present a semi-automated workflow to

- set up and start large batches for supercomputer runs for millions of buildings with minimal manual intervention, and optimizing the input data to minimize I/O and synchronization issues

- automatically analyze and check each run's outputs for missing buildings, and create the data required to resubmit them in an iterative process,
- postprocess and package up the results to publish data while minimizing storage needs.

II. RELATED WORK

Building energy simulations have been an active topic of research for the past 50 years. Today's building energy simulation codes support a vast array of different parameters that can be configured, including building use type, building standards (which depend on location and year built), geometry, number of floors, type of HVAC system, window-to-wall ratio (relevant in heating loss), insulation, and many more [12]. By running different simulations, one can compare different options during new constructions and to retrofit buildings with more energy-efficient infrastructure. This can answer questions such as "Should I invest in a smart thermostat, a new water heater, or double-pane windows?" for an individual building (depending on the climate in a specific area, the answer to this question may vary), or inform decisions about energy efficiency technology investments for whole utility service areas [8] and cities [11]. It can also support policy-makers in decisions about introductions of new building codes [22] to pave the way towards a more energy-efficient future.

The severe impact of future weather and climate change on building energy and the electric grid has been studied based on the Intergovernmental Panel on Climate Change Assessment Report 6 [7], using an ensemble of climate simulations and empirical relationships between weather and energy consumption [34], using a combination of different downscaled climate models [6], and using LandCast [3], a population growth projection for the U.S. for 2030 and 2050 [27]. Similar works have been performed for the European Union [10] and other countries world-wide [33].

While fields like climatology and population dynamics have operated at national and even global scales for many years, building simulations have historically been challenging. Even recent works [20], [22], [32] focus primarily on on neighborhood to city scale modeling. To get to nation-scale modeling, the typical approach is to categorize the buildings into different building archetypes, simulate models of these building archetypes for all relevant climate zones, and simply multiply the results with the number of buildings for each archetype [35]. However, there are substantial changes between building geometries, materials, and codes between different areas, which are missed in this approach.

Building simulation codes such as [40] and [39], the codes used in this work, take much time and effort to set up. A typical program often has hundreds of input parameters that need tuning. Previous efforts have been made to "autotune" EnergyPlus models [36] to streamline their setup. Both building simulation codes we use in this work can be run in an "embarrassingly parallel" fashion (i.e. independent of each other), as models for different buildings are typically independent. However, the scale of the data processing for setup,

the output data, and the computational resources required to run the simulations at nation scale, have been prohibitive.

As our application has big data and big compute, these two aspects have to be balanced against each other [19], [38]. Fiore et al. [18] provide an overview of the challenges of the availability of compute time, and scaling under strict performance (e.g. processing time) and cost (e.g. storage). There has been work to enhance the functioning of HPC architectures [2] and scheduling [25] to improve I/O efficiency in big data HPC applications. Other approaches focus on the data transfer from HPC simulations to real-time analysis [26].

In this work, we address the challenge of big data and big compute by focusing our efforts on three main aspects: (1) We minimize I/O on the main lustre file system by running the main application entirely in node memory, and only synchronize data between the node and the lustre filesystem at the beginning and end of each node's compute. (2) We optimize data management with big compute in mind, by splitting the input data a priori and packaging it up by core and node to take advantage of the embarrassingly parallelizable nature of the task. (3) Finally, we reduce overall storage use by limiting data extraction from compressed outputs (tarballs) by examining the list of contents where possible, or extracting just specific files we need.

III. METHODOLOGY

In this section, we will introduce all relevant steps along the large-scale Automatic Building Energy Modeling (AutoBEM) workflow. Figure 1 provides a high-level overview of the workflow, from inputs, through several iterations of compute, to outputs, and the various data extracts in between. We begin by describing a typical building simulation workflow and the modifications we have made to it to enable large-scale computing (Section III-A), and we discuss data preparations required (Section III-B). Then we discuss the different steps along the workflow: preprocessing steps (Section III-C), different job configuration parameters (Section III-D), an overview of the parallel workflow (Section III-E), handling missing data (Section III-F), and postprocessing steps (Section III-G).

A. Typical Building Simulation Workflows

Physical building energy modeling simulation engines such as EnergyPlus [39] take a set of thousands of building properties and combine them with weather data to calculate thermal loads, system responses to those loads, and energy use, along with several other related metrics. Building simulations for a small number of buildings can be done on a desktop computer. We expand an existing workflow [32], [37] which uses pre-generated EnergyPlus [13], [39] building models in Input Data File (IDF) format. If such IDF building models are available, this workflow scales well. However, the generation of such models for individualized buildings takes about as long as simulating the building models, rendering this workflow insufficient for nation-scale modeling.

We alleviate this issue by integrating OpenStudio [40] in our HPC workflow. OpenStudio is a tool which can generate
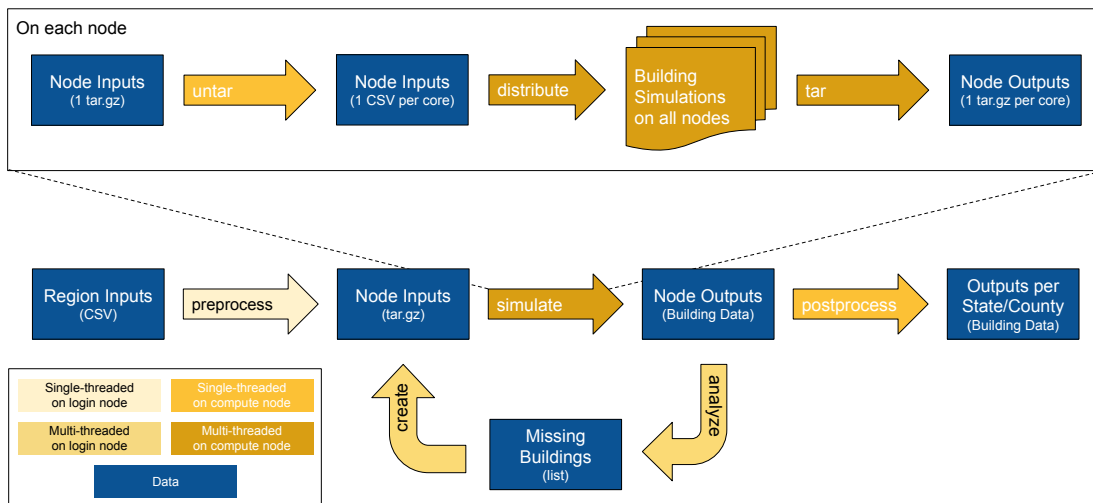
Fig. 1: High-level workflow for large-scale building simulation with AutoBEM: The bottom part of the workflow represents the overall workflow while the top portion provides more detail about the steps which run on compute nodes. The legend at the bottom contains the color-coding: all data is represented in blue, while compute steps are highlighted in different shades of yellow, depending on parallelism (darker colors indicating more parallel execution).

OpenStudio and EnergyPlus models from a set of building property inputs: Geometry, Centroid, Height, Number of Floors, Window-to-Wall, Building Type, Building Standard, and Climate Zone. Then, a building energy simulation is run for each of these models using EnergyPlus. With this integration, we are able to both generate and simulate the models on HPC resources.

B. Data Preparation

The building simulation workflow with OpenStudio and EnergyPlus requires the inputs listed above. The OpenStudio workflow takes these inputs and creates an building energy model. Then, it calls EnergyPlus to simulate building energy use, outputting a year of hourly building energy data. Each model is simulated using the Typical Meteorological Year (TMY) weather file for the representative city associated with the climate zone for the building location based on ANSI/ASHRAE/IES Standard 90.1-2016 [9]. The outputs of this workflow include an OpenStudio model, an EnergyPlus model, building energy use data, as well as log files with data about the simulation.

In this work, our goal was to simulate 125.7 million buildings in the United States. To better manage this workload, we divided the buildings into the 9 U.S. census regions as well as California (Table I), which was separated due to the large number of buildings and outside interest.

When simulating a single building, modelers typically know all relevant properties for the building in question. However, at national scale, not all information is known, and we had to apply heuristics to generate suitable input data for some building properties like building type. For this work, we use building geometries from Microsoft's building footprint dataset [28] while building heights were provided by a data

TABLE I: Region definitions: each region number corresponds to a census region which includes between one and nine states. The total number of buildings contained in a region varies between 5.1 million and 23.6 million.

Region	States	#Buildings
1	Connecticut, Maine, Massachusetts, New Hampshire, Rhode Island, Vermont	5,435,392
2	New Jersey, New York, Pennsylvania	12,637,184
3	Illinois, Indiana, Michigan, Ohio, Wisconsin	22,528,155
4	Iowa, Kansas, Minnesota, Missouri, Nebraska, North Dakota, South Dakota	12,463,109
5	Delaware, District of Columbia, Florida, Georgia, Maryland, North Carolina, South Carolina, Virginia, West Virginia	23,558,752
6	Alabama, Kentucky, Mississippi, Tennessee	9,977,403
7	Arkansas, Louisiana, Oklahoma, Texas	15,480,692
8	Arizona, Colorado, Idaho, Montana, Nevada, New Mexico, Utah, Wyoming	7,551,785
9	Alaska, Hawaii, Oregon, Washington	5,149,591
10	California	10,933,546

partner. The representative city EnergyPlus Weather (EPW) files come pre-loaded with OpenStudio.

The result of this data preparation step is one Comma-Separated Values (CSV) file per region, which contains building IDs, building type, building standard, height, number of floors, the building footprint geometry (embedded json), window-to-wall ratio, climate zone, and other parameters.

C. Preprocessing

The preprocessing step takes the CSV for each region, produces suitable inputs for HPC jobs, and submits these jobs

to the queue. This step happens on a login node, which is a shared resource between many users.

First, we need to choose what maximum size we want the jobs to be. This depends on the number of buildings in each region, as well as the HPC configuration. Theta has 4,392 compute nodes. Each node has a 64-core, 1.3-GHz Intel Xeon Phi 7230 processor, so we can simulate up to 64 buildings in parallel on each node. Next, we have to decide how many buildings each core will simulate during a job. We let each core process a few buildings during a given job, and we call the number of buildings a core simulates during a job the number of *churns*.

We process each region as a separate batch of jobs in order to better keep track of which buildings have been processed, and which regions have more difficulties than others. For each batch, we choose a batch name. The last job is typically a bit smaller than the others, as we want to minimize the resources used, and keep the load between nodes evenly balanced. The output of this script is a batch script which calls the run generation code for each job in a region.

The run generation code takes the number of nodes and churns, a desired name for the list of jobs, and the CSV input (subset from the preparation script) for the region. During preprocessing, we first split up the region CSV file: each core gets a CSV file with enough buildings for the number of churns requested. These CSV files are compressed into one tarball for each node, which is saved to disk. In addition to chunking up the input into appropriate sizes for nodes, the preprocessing script also produces a job submission script for the job, which is automatically submitted to the queue.

D. Job Configuration

Finding a good job configuration can be challenging. The three main considerations for our job configuration are:

- **How much data needs to be processed?** We have 125 million buildings to process. At 80% Theta usage, we can process $3,514 \cdot 64 = 224,896$ buildings in parallel. If every building took 15 minutes to process, we would need about 139 hours of time on 80% of all nodes.
- **How long will our jobs wait in the queue?** This depends on the job size (number of nodes) and its requested walltime (duration of the reservation). For a job with more nodes, more jobs will have to finish running before there is enough space to run a large job. The requested walltime also has an impact (longer walltime generally results in a longer wait).
- **How can we best use the available resources?** We want the majority of cores to be busy during the entire run, as idle cores are wasted resources. However, the processing time can vary dramatically between buildings. We have to balance the number of completed buildings against keeping all cores busy.

Prior to our production runs, we experimented with different job configurations to heuristically determine suitable job parameters.

1) Number of Nodes: The theoretical maximum job size would use 4,392 nodes. However, sometimes a few nodes are off-line, or a certain percentage of nodes is reserved for a specific project and not available. The theoretical minimum job size is 128 nodes as this is the minimum requirement for production runs, as the reasoning is that any job smaller than this does not require a supercomputer as it could run on a small cluster. Through experiments, we determined that jobs that used 20-25% of Theta's nodes had queue times of 4-5 days, whereas jobs that used 80% of Theta's nodes had queue times of 6-7 days. In addition, there is a limit on the number of jobs a single user can have in the queue, so it was in our best interest to minimize the number of jobs to submit. At a goal of 125 million buildings to process, 80% was an easy choice. For our project, the maximum number of nodes used is 3,514 nodes, or 80% of all available nodes on Theta. This size is used for 59% of our jobs for this project. 73% of our jobs used at least half of the available nodes. If the last job becomes too small (under the minimum number of nodes), we reduce the size of the other jobs of the batch to get a valid size for all jobs. For example, if a region has enough buildings to run on 3,600 nodes, we could split them into two jobs of 1,800 nodes to avoid having one job that is too small to run as a production job.

2) Number of Churns: We ran some initial experiments on a small subset of the data to determine the expected processing time per building. We found that that on average, it takes about 15 minutes to generate a model and simulate building energy for it, using our OpenStudio/EnergyPlus setup. This means we can process about four buildings per hour on each core. At the determined job size of 3,514 nodes, we expect to process $3,514 \cdot 64 \cdot 4 = 899,584$ buildings per hour.

3) Walltime: Based on initial experiments with walltime, we determined that increases in walltime were theoretically worth the additional wait, with 120 hours for a 2-hour job and 196 minutes for a 10-hour job. However, depending on the distribution of the buildings, this may result in many cores running idle while waiting for a few cores to finish the complex buildings they are processing. This would waste a lot of valuable compute time. To minimize this risk, we decided to use a walltime of 2 hours. This means that the number of buildings we expect to process almost 1.8 million buildings per full-size job with $2 \cdot 4 = 8$ churns.

E. Parallel Execution Workflow

The parallel execution workflow begins as soon as a job from the queue starts executing on the compute node. A visual representation of this part of the workflow is shown at the top of Figure 1. During initialization, the building simulation code and input data are copied from lustre and untarred in the on-node memory. This initialization step is carried out in single-thread mode on the compute node as we only have two inputs.

Then, we spread out the work between all cores on the node. Each core receives a CSV file containing building data for a *#churns* buildings. Then, we start the OpenStudio/EnergyPlus

simulation workflow, which simulates each of the buildings and writes the building models, energy use, and any error logs to the local memory.

When the job's walltime is nearing its limit, the outputs of each core are compressed into one tarball per core, and written back out onto the lustre file system.

F. Handling Incomplete Jobs

Ideally, each job can finish processing all buildings it was assigned. However, this is not always the case as building processing times can vary. There are many parameters that influence processing times, such as building size, number of floors, or HVAC can play a substantial role in the time it takes to set up a model or simulate it. Certain building types inherently take longer to simulate based on complexity while other individual models that are much larger or have complex geometries may take a long time to converge. There have been extensive works on fault management [24] and checkpointing [23], [41], and redundancy [16], as well as their tradeoffs [15]. We take a redundancy-based approach: we provide jobs with enough resources to complete the majority of their buildings, but not so much that too many cores or nodes run idle while a few are still busy.

In order to strike this balance of processed buildings and computational resources (compute time on Theta), we devised the following submission strategy, which is illustrated in Figure 2. At a high level, we begin with 8 churns, and then cut the number of churns in half every time we have to resubmit a building which did not complete processing in the previous step, which is shown in the top row. If there are not enough buildings left for a full job (80% of all nodes) at the current number of churns, the missing buildings are added to a "remainder" pool of buildings from miscellaneous regions, to enable more efficient processing, and minimize the number of jobs in the queue. The white diamonds indicate a check to determine whether the buildings are resubmitted as individual jobs or as part of the pool, based on the number of missing buildings from each set of runs. If there are not sufficient buildings to fill a production run for the next iteration of job settings, the missing buildings are added to the remainder pool. The process of reducing churns with each iteration remains the same for the remainder pool, but unlike the pools for each region, the remainder pool does not always shrink in size with each iteration because it receives additional building inputs whenever one of the region is sufficiently complete.

Based on initial testing, we determined that most buildings can be processed within 15 minutes, including model generation and building energy simulation. Therefore we began by submitting an inital set of runs with 2 hours of walltime and 8 churns per core, i.e. 8 buildings assigned to each core. We submitted 92 jobs of varying node count, using 8 churns. 88.4% of all submitted buildings were processed successfully during 8-churn runs.

The buildings which did not complete were collected and resubmitted in a *1st Rerun* using the same walltime, but only 4 churns per core, which gives each building half an hour

to complete. We submitted 21 jobs using 4 churns, and we were surprised to find that only 34.3% of the buildings were processed successfully during these runs.

In the *2nd Rerun* iteration, we again cut the number of churns in half, and submitted 11 jobs with only 2 churns. These jobs fared much better than the previous iteration, with 86.2% successfully processed buildings.

The *3rd Rerun* uses a single churn. We submitted four jobs: one job for Region 1 with about 225k buildings which had a completion rate of 92.1%, and three jobs for Region 6 which totaled about 486k buildings and which had a completion rate of only 20.1%. The missing buildings from these were transferred to a dataset for further exploration why they were so time-consuming to simulate.

G. Postprocessing and Analysis

The output data from this process is quite large. The project presently fills over 80TB of data on Theta's lustre filesystem. We employed a multi-step process for postprocessing and analysis which avoids extracting data unless it is absolutely necessary.

The first important step after a job had completed was to determine the number of completed buildings, and identify missing buildings. To quickly and efficiently fulfil this task, we used the `tarfile` python library to examine the contents of each node's tarball outputs without having to untar all of its contents. This was performed in parallel using python's `multiprocessing` library.

- If a building is complete, the outputs include an IDF model named `[buildingID].idf`.
- If a building was not processed completely or was never simulated, this model is missing from the node outputs, or there may be no outputs at all.

We kept track of completed and missing buildings using lists, and once the processing was complete, we filtered the input data submitted for the run to just the rows for missing buildings and wrote out the file to be used for the next iteration of runs.

Once we collected the missing buildings for all jobs in a given region, we merged the files into one CSV, which was then handed back to the preprocessing step (Section III-C) for the next iteration of runs (Section III-F).

Next, we wanted to know how different parts of the code performed, such as the chunking and tarballing during preprocessing. The processing times for these two steps, as well as various steps of processing on compute nodes (such as individual building processing times), can be found in log files. The postprocessing step includes a sweep through all relevant logs to extract these data for analysis and future use.

Finally, as this level of processing is not widely available, we want to share the building models with the wider building simulation community. Most users are interested in a specific area, e.g. a single county. To facilitate easier access, we processed the building outputs to separate them by data type (e.g. IDF EnergyPlus building model, or OpenStudio Model (OSM)) and county. Each successfully processed building
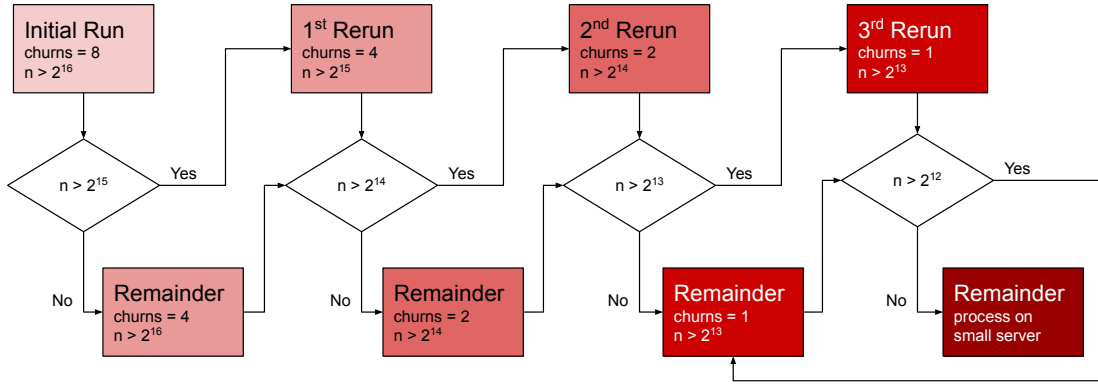
Fig. 2: Overview of our strategy to handle incomplete jobs: with each iteration, we cut the number of churns in half. While there are enough buildings for individual runs per region, we follow the top row, otherwise buildings are added to a remainder pool and follow the bottom row from that iteration onwards. In both rows, the color indicates the time provided per building, growing darker with each iteration of jobs.

results in five files (with much higher numbers of files for incomplete buildings, as there are temporary files during simulation). Simultaneously untarring all results would result in at least 628.6 million files on disk, which could result in adverse effects on the file system.

Instead, only OSM and IDF files were untarred. State level aggregation allowed for models to be selected for a given region within a state easily as data requests were processed. This also allowed for the models to be zipped by county, allowing for easier access to the data for the general public.

The 122.9 million buildings (97.8%) of buildings have already been published as part of the Model America dataset [29]. Smaller extracts are available for Los Angeles (with detailed climate inputs from the WRF climate simulation) [31] and Clark County (Las Vegas) [30].

IV. RESULTS

Using the methodology described in the previous section, we processed 124.5 million of the 125.7 million buildings, or 98.9% of all U.S. buildings.

For testing purposes, a total of 2.3 million buildings from region 2 were submitted with 2 churns rather than 8 churns on their initial run, and had a success rate of 86.9%, which is a little higher than the success rate of 8-churn runs for the same region (75.2%), but not as good as the very successful regions. These 2.3 million buildings were included in the Initial category for comparisons by iteration, and the 2 churns category for comparisons by churn count.

Table II lists the number of buildings available (Total), and how many buildings were successfully processed during each iteration of runs (Initial, Rerun1, Rerun2, Rerun3). Regions for which the Rerun columns are empty had such high success rates, that there were not enough buildings left to justify a rerun for the region. These missing buildings were added to the remainder pool (R). The first iteration of remainder reruns included about 742k buildings from regions 3, 5, and 7-10.

Figure 3 provides a visual view of these numbers and how they relate to other factors, such as number of jobs

TABLE II: Each row gives the number of buildings available, and how many of them were processed during each iteration. The bottom two rows provide the sum across all regions for each iteration, as well as the percentage of buildings completed during each iteration.
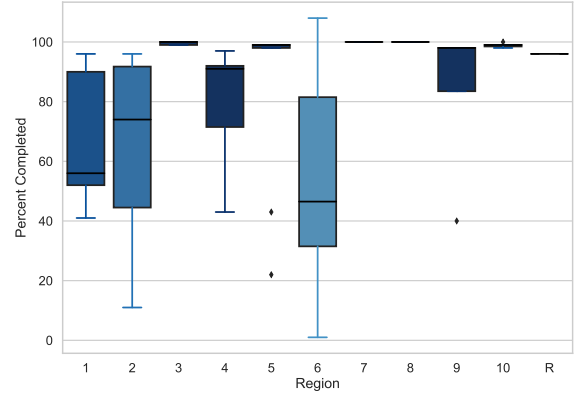
#	Buildings	Initial	Rerun1	Rerun2	Rerun3
1	5,435,392	4,362,202	518,576	291,327	207,103
2	12,637,184	9,019,214	1,358,741	1,821,282	
3	22,528,155	22,467,370			
4	12,463,109	10,748,054	791,385	820,444	
5	23,558,752	23,207,301			
6	9,977,403	4,517,672	1,442,752	3,046,398	97,638
7	15,480,692	15,452,216			
8	7,551,785	7,531,113			
9	5,149,591	5,051,863			
10	10,933,546	10,781,507			
R			742,174		
$\sum$	125,716,992	113,138,512	4,853,628	5,979,451	305,741
%		89.99%	3.86%	4.76%	0.16%

and number of churns. Figure 3a shows that 90% of all buildings completed during the initial iteration of runs. There are more Rerun1 jobs than Rerun2 jobs, but Rerun2 yielded better results. The results for Rerun3 has the worst overall performance. Rerun1 performs very badly with a median of about 40, whereas Rerun2 performs well with almost all jobs having 80-100% success rates. Finally, Figure 3b gives more insight into the regional differences. The 90% of successful building completion are not evenly distributed: where Regions, 3, 5, 7, 8 and 10 have extremely consistent and high success rates of over 98%, other regions are much more spread out in performance. Regions 4 and 9 have a much higher variation, but high completion rates as seen from their dark blue coloring (based on overall completion, which includes several reruns for both regions). Region 6 has the worst overall performance, with a current completion rate of 91.25%. This region also had the single worst run of all, with only 1% success rate.

The big differences in success between different regions can likely be explained by differences in the number of computationally complex buildings. Further investigation into
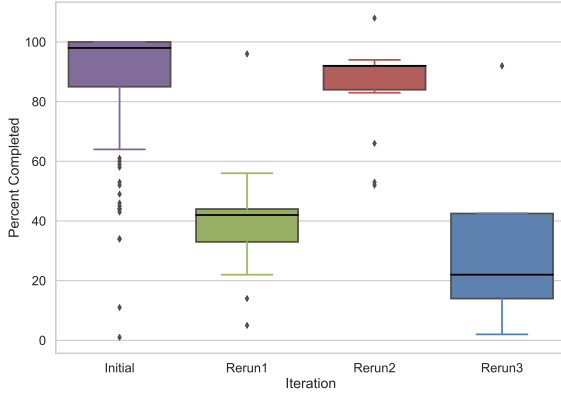
(a) Percentage of each iteration's completed buildings.

(b) Percentage of each region's completed buildings.

Fig. 3: Box plots comparing the percentage of jobs completed by iteration (3a) and by region (3b). Colors are kept consistent with Table II: 8 churns/Initial, 4 churns/Rerun1, 2 churns/Rerun2 and 1 churn/Rerun3. Figure 3b uses are a linear blue colormap, with darker hues indicating higher overall completion for a region throughout all iterations (values ranging from 90.2%-99.8%).

this will be an interesting topic for future studies. Based on the bad performance of Rerun1 (4 churns), it is plausible that the majority of buildings are either very simple (compute successfully in the Initial, 8-churn iteration), or they are very complex and require more than half an hour to compute. This could explain the relative success of Rerun 2 (2 churns) which provides an hour of time per building.

Finally, we would like to acknowledge that none of this work would have been possible without access to the Theta supercomputer. In order to generate 124.5 million building models and simulate their energy use, we have used over 10 days of walltime on Theta. 72% of the jobs used more than 50% of its nodes, and 59% of them used 80% of its nodes. We have spent 752k node hours or 48 million core hours. It would take a 64-core server almost 86 years to perform the same task. An average 8-core desktop would have to spend 687 years on the same task.

V. CONCLUSION AND FUTURE WORK

We have presented the AutoBEM workflow which processes large amounts of data to generate and simulate building models for almost all buildings in the continental United States. Our workflow has proven to be reliable and highly scalable. We have successfully demonstrated the use of this workflow at nation-scale for 124.4 million buildings. The building models have been published and shared with the building energy modeling community, and through this work, we have also collected detailed information on building processing times, which can be used in future work.

We have set up and tested a preliminary workflow to load balance jobs for buildings with known processing times. We expect that this load balancing module will be a great asset for planned building simulations which will explore which types of building retrofitting (e.g. new HVAC unit, upgraded windows, etc.) have the highest potential to reduce energy consumption and emissions. This can inform policies and building codes, and hopefully reduce the overall U.S. energy use and carbon footprint. Furthermore, while a couple of factors contributing to long building processing times are known, a reliable prediction of processing times based on input parameters has not been possible. Finally, it would be interesting to compare the simulated data to measured energy use in areas in which such data is available.

REFERENCES

[1] International Energy Agency. Energy efficiency indicators: Overview, dec 2018.

[2] Awais Ahmad, Anand Paul, Sadia Din, M Mazhar Rathore, Gyu Sang Choi, and Gwanggil Jeon. Multilevel data processing using parallel algorithms for analyzing big data in high-performance computing. *International Journal of Parallel Programming*, 46(3):508–527, 2018.

[3] Melissa R Allen, Steven J Fernandez, Joshua S Fu, and Mohammed M Olama. Impacts of climate change on sub-regional electricity demand and distribution in the southern united states. *Nature Energy*, 1(8):1–9, 2016.

[4] Melissa R. Allen, Amy Rose, Joshua R. New, Olufemi Omitaomu, Jiangye Yuan, Marcia L. Branstetter, Linda M. Sylvester, Matthew Seals, Thomaz M. Carvalhaes, Mark B. Adams, Mahabir S. Bhandari, Som S. Shrestha, A. Berres, Carl P. Kolosna, Katherine Fu, and Alexandra C. Kahl. Impacts of urban morphology on microclimate and building energy use. 2020.

[5] Maximilian Auffhammer and Anin Aroonruengsawat. Simulating the impacts of climate change, prices and population on californias residential electricity consumption. *Climatic change*, 109(1):191–210, 2011.

[6] Maximilian Auffhammer, Patrick Baylis, and Catherine H Hausman. Climate change is projected to have severe impacts on the frequency and intensity of peak electricity demand across the united states. *Proceedings of the National Academy of Sciences*, 114(8):1886–1891, 2017.

[7] Brett Bass and Joshua New. Future meteorological year weather data from IPCC scenarios. In *ASHRAE Topical Conference Proceedings*, pages 471–477. American Society of Heating, Refrigeration and Air Conditioning Engineers, Inc., 2020.

[8] Brett Bass, Joshua New, and William Copeland. Potential energy, demand, emissions, and cost savings distributions for buildings in a utilitys service area. *Energies*, 14(1):132, 2021.

[9] Liu Bing, Michael Rosenberg, and Rahul Athalye. National impact of ANSI/ASHRAE/IES standard 90.1-2016.

[10] Philip Cafaro and Patrícia Dérer. Policy-based population projections for the european union: a complementary approach. *Comparative Population Studies*, 44, 2019.

[11] Yixing Chen, Tianzhen Hong, and Mary Ann Piette. Automatic generation and simulation of urban building energy models based on city datasets for city-scale building retrofit analysis. *Applied Energy*, 205:323–335, 2017.

[12] Drury B Crawley, Jon W Hand, Michaël Kummert, and Brent T Griffith. Contrasting the capabilities of building energy performance simulation programs. *Building and environment*, 43(4):661–673, 2008.

[13] Drury B Crawley, Linda K Lawrie, Frederick C Winkelmann, Walter F Buhl, Y Joe Huang, Curtis O Pedersen, Richard K Strand, Richard J Liesen, Daniel E Fisher, Michael J Witte, et al. Energyplus: creating a new-generation building energy simulation program. *Energy and buildings*, 33(4):319–331, 2001.

[14] DOE Office of Energy Efficiency & Renewable Energy. Energy efficiency trends in residential and commercial buildings.

[15] Nosayba El-Sayed and Bianca Schroeder. To checkpoint or not to checkpoint: Understanding energy-performance-i/o tradeoffs in hpc checkpointing. In *2014 IEEE International Conference on Cluster Computing (CLUSTER)*, pages 93–102, 2014.

[16] James Elliott, Kishor Kharbas, David Fiala, Frank Mueller, Kurt Ferreira, and Christian Engelmann. Combining partial redundancy and checkpointing for hpc. In *2012 IEEE 32nd International Conference on Distributed Computing Systems*, pages 615–626, 2012.

[17] Enerdata. Global energy statistical yearbook, 2021.

[18] Sandro Fiore, Mohamed Bakhouya, and Waleed W. Smari. On the road to exascale: Advances in high performance computing and simulationsan overview and editorial. *Future Generation Computer Systems*, 82:450–458, 2018.

[19] Geoffrey Fox, Judy Qiu, Shantenu Jha, Saliya Ekanayake, and Supun Kamburugamuve. Big data, simulations and HPC convergence. In *Big Data Benchmarking*, pages 3–17. Springer, 2015.

[20] Eric Garrison, Joshua New, and Mark Adams. Accuracy of a crude approach to urban multi-scale building energy models compared to 15-min electricity use (at-2019-c002). In *2019 Winter Conference*. ASHRAE, 2019.

[21] Ian Hamilton and Oliver Rapf. Executive summary of the 2020 global status report for buildings and construction, 2020.

[22] Tianzhen Hong, Yixing Chen, Xuan Luo, Na Luo, and Sang Hoon Lee. Ten questions on urban building energy modeling. *Building and Environment*, 168:106508, 2020.

[23] Kuldeep Kurte, Jibonananda Sanyal, A. Berres, Dalton Lunga, Mark Coletti, Hsuihan L. Lexie Yang, Daniel Graves, Benjamin Liebersohn, and Amy Rose. Performance analysis and optimization for scalable deployment of deep learning models for country-scale settlement mapping on titan supercomputer. volume 31, page e5305. Wiley Online Library, 2019. e5305 cpe.5305.

[24] Zhiling Lan and Yawei Li. Adaptive fault management of parallel applications for high-performance computing. *IEEE Transactions on Computers*, 57(12):1647–1660, 2008.

[25] Jialin Liu, Yu Zhuang, and Yong Chen. Hierarchical collective i/o scheduling for high-performance computing. *Big Data Research*, 2(3):117–126, 2015.

[26] Thomas Marrinan, Silvio Rizzi, Joseph Insley, Brian Toonen, William Allcock, and Michael E. Papka. Transferring data from high-performance simulations to extreme scale analysis applications in real-time. In *2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 1214–1220, 2018.

[27] Jacob J McKee, Amy N Rose, Edward A Bright, Timmy Huynh, and Budhendra L Bhaduri. Locally adaptive, spatially explicit projection of us population for 2030 and 2050. *Proceedings of the National Academy of Sciences*, 112(5):1344–1349, 2015.

[28] Microsoft. Microsoft building footprints, December 2020. https://www.microsoft.com/en-us/maps/building-footprints.

[29] Joshua New, Mark Adams, A. Berres, Brett Bass, and Nicholas Clinton. Model America data and models of every U.S. building, April 2021. Automatic Building Energy Modeling (AutoBEM).

[30] Joshua New, Brett Bass, Mark Adams, and A. Berres. Model America - Clark County (Vegas) extract from ORNL's AutoBEM, February 2021. Integrated Multisector Multiscale Modeling (IM3) project - https://im3.pnnl.gov/.

[31] Joshua New, Brett Bass, Mark Adams, A. Berres, and Xuan Luo. Los Angeles County Archetypes in Weather Research and Forecasting (WRF) Region from ORNL's AutoBEM, April 2021. Integrated Multisector Multiscale Modeling (IM3) project - https://im3.pnnl.gov/.

[32] Joshua R New, Mark B Adams, Piljae Im, Hsiuhan Lexie Yang, Joshua C Hambrick, William E Copeland, Lilian B Bruce, and James A Ingraham. Automatic building energy model creation (AutoBEM) for urban-scale energy modeling and assessment of value propositions for electric utilities. Technical report, Oak Ridge National Laboratory, Oak Ridge, TN (United States), 2018.

[33] Adrian E Raftery, Nan Li, Hana Ševčíková, Patrick Gerland, and Gerhard K Heilig. Bayesian probabilistic population projections for all countries. *Proceedings of the National Academy of Sciences*, 109(35):13915–13921, 2012.

[34] Deeksha Rastogi, James Scott Holladay, Katherine J Evans, Ben L Preston, and Moetasim Ashfaq. Shift in seasonal climate patterns likely to impact residential energy consumption in the united states. *Environmental Research Letters*, 14(7):074006, 2019.

[35] Christoph F. Reinhart and Carlos Cerezo Davila. Urban building energy modeling a review of a nascent field. *Building and Environment*, 97:196–202, 2016.

[36] Jibonananda Sanyal and Joshua New. Simulation and big data challenges in tuning building energy models. In *2013 Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES)*, pages 1–6, 2013.

[37] Jibonananda Sanyal, Joshua New, Richard E Edwards, and Lynne Parker. Calibrating building energy models using supercomputer trained machine learning agents. *Concurrency and Computation: Practice and Experience*, 26(13):2122–2133, 2014.

[38] Nemanja Trifunovic, Veljko Milutinovic, Jakob Salom, and Anton Kos. Paradigm shift in big data supercomputing: dataflow vs. controlflow. *Journal of Big Data*, 2(1):1–9, 2015.

[39] U.S. Department of Energy. EnergyPlus, 2021. https://energyplus.net/ (retr. 2021-09-03).

[40] U.S. Department of Energy. OpenStudio, 2021. https://www.openstudio.net/ (retr. 2021-09-03).

[41] Chao Wang, Frank Mueller, Christian Engelmann, and Stephen L. Scott. Hybrid Checkpointing for MPI Jobs in HPC Environments. In *2010 IEEE 16th International Conference on Parallel and Distributed Systems*, pages 524–533, 2010.