

A Co-Simulation Framework for Building Energy Management as a Testbed for Energy-Aware Data Movement Analysis

CHOMPHUNUCH WONGPHONG

Thammasat University, Thailand, chomphunuch.won@dome.tu.ac.th

PATCHARARAT WONGTA

Thammasat University, Thailand, patchararat.won@dome.tu.ac.th

WORAWAN MARURNGSITH

Thammasat University, Thailand, papong@tu.ac.th

ABSTRACT

Improving energy efficiency through co-simulation frameworks can enhance building and operations management, a key strategy to reduce energy use per square meter by 35% by 2030. Buildings consume more than a third of global energy, with Heating, Ventilation, and Air Conditioning (HVAC) systems contributing over 70%. Unpredictable occupant behavior can cause energy variability of 100-300%, even in similar buildings and systems. A co-simulation framework called Twin-B is proposed to combine building physics with agent-based occupant behavior models to improve prediction accuracy, but it requires High Performance Computing (HPC) resources. Consequently, the effective use of HPC resources in building energy management co-simulation is essential. Recent research shows that frequent data exchange in co-simulation can create bottlenecks, increasing processing time by 20-50% and consuming 0.06-0.2 kWh/GB for network transfers. This paper addresses the HPC resource inefficiency issue found when using the Twin-B co-simulation to predict energy efficiency in a university building. Our study reveals that small data exchanges (averaging 37.5 bytes) between simulators account for 66.3% of stream-synchronized overhead, consuming 7.42 J of energy. This host-to-device transfer results directly from per-agent decision-making and could be optimized by batching agent operations. We found that with batching optimization, energy consumption can decrease by 99.5% while achieving a 1,200 \times speedup, without sacrificing simulation accuracy. This research provides a quantitative link between building energy co-simulation data exchange patterns and HPC energy efficiency, offering strategies for optimizing sustainable digital twin systems.

Additional Keywords and Phrases: Energy Efficiency, Co-simulation, Building Energy, Agent-based Model, HPC

1 INTRODUCTION

Energy efficiency, enhanced through co-simulation frameworks, can optimize both building renovations and operational management, providing a key pathway toward reaching a 35% reduction in energy use per square meter by 2030. Buildings account for more than one-third of global energy consumption, with Heating, Ventilation, and Air Conditioning (HVAC) systems making up over 70% of the total energy used in buildings [1]. Studies indicate that energy

use is influenced by six factors, including occupant behavior, occupant density, temperature, building structure, system efficiency, and management policies. Of these, occupant behavior is the most significant yet unpredictable, causing energy fluctuations of up to 100–300% even in buildings with similar designs and systems [2, 3]. Although earlier research confirmed that simulating only the building's physical factors is inadequate, the computational demands often required compromises with energy simulation accuracy. Advances in high-performance computing (HPC) facilities beyond Exascale, closely integrated with AI accelerators, enable hybrid simulation models to test energy-saving measures by co-simulating physical factors and occupant behaviors. Virtual testbeds for evaluation can identify inefficiencies caused by operational disruptions and scenarios, improving the accuracy of energy simulation and forecasting in building energy management systems. However, co-simulation still faces challenges related to high energy consumption during data exchange processes. Studies indicate that energy-aware data transfer in co-simulation frameworks is slow, resource-intensive, and complex, which can lead to inaccuracies in translating behavior into energy results [4].

A hybrid-driven co-simulation framework excels at interoperability and robustness. However, they are limited by substantial energy consumption during data exchange. Synchronizing multiple replicas within a unified replication environment requires frequent data transfers, which consume significant CPU and memory resources, resulting in idle periods that degrade performance. Co-simulation overhead can increase computation time by 20–50% compared to standalone simulations, thereby proportionally increasing electricity consumption [4, 6]. The choice of connection time step is essential for simulation accuracy and energy efficiency. Smaller time steps enhance precision but increase energy demand due to more frequent exchanges, while larger steps reduce computational overhead but may miss transient events. Distributed co-simulation introduces additional inefficiency due to network-based data transfer. Analyses of data center growth from 2005 to 2010 indicate that network transmission consumes considerably more energy than in-memory transfer, averaging 0.06–0.2 kWh per GB depending on distance and network type [5]. Given that the number of data exchanges in co-simulation can reach thousands per run, this cumulative transmission overhead constitutes a significant portion of the total energy footprint. This paper presents Twin-B, a hybrid co-simulation testbed platform designed to simulate and evaluate the energy performance of buildings. It integrates the EnergyPlus building energy model with the Mesa agent-based behavior modeling in Python, using co-simulators that exchange and synchronize real-time data within a HPC environment on the LANTA supercomputer. The Twin-B system serves as a testbed to explore the effectiveness of real-time data exchange and synchronization between simulators with different structures. Our contributions are threefold.

We propose creating and executing a co-simulation framework for managing building energy that incorporates detailed physical elements and occupant behaviors. This framework showcases its potential in analyzing different scenarios to forecast building energy management systems.

- 1) We present the Twin-B model as a testbed to identify bottlenecks or excessive energy use during simulation, providing insights to optimize simulation efficiency, reduce energy consumption during data exchanges, and balance subsystem workloads, laying the groundwork for sustainable computing in future building energy simulations.
- 2) We quantify the energy efficiency of the dataflow in co-simulation environments and analyze how data exchange between simulators affects the accuracy of simulation outcomes.

The next sections of this paper are organized as follows: Section 2 reviews key literature and research directions in building energy management. Section 3 describes the Twin-B testbed and technical implementation. Section 4 presents experimental results, and Section 5 discusses implications for energy efficiency and sustainable computing. Together, these sessions provide a concise overview of the study's methodology, findings, and significance.

2 BACKGROUND AND RELATED WORK

2.1 Building Energy Simulation and Co-Simulation

Building Energy Simulation (BES) is a tool used to analyze and predict energy use to meet sustainability goals. It considers factors such as lighting, air conditioning, building structure, occupant behavior, and weather conditions. EnergyPlus [13] is the leading dynamic simulation tool that employs physics-based models to calculate heat transfer and energy flow. It determines PMV (Predicted Mean Vote), lighting conditions, and energy consumption to simulate the building's physical properties and control systems. The physics-based modeling techniques are used to analyze energy flow within a building. It uses input data files for building materials and HVAC systems, along with weather files, to simulate weather conditions. EnergyPlus comprises three main components: the Simulation Manager, the Heat and Mass Balance Simulation Module, and the Building Systems Simulation Manager. However, EnergyPlus has notable limitations in modeling occupant behavior, which is a significant source of energy variability in real buildings.

Agent-based simulation (ABS) addresses this gap by modeling autonomous units (agents) with individual behaviors and decision-making capabilities. Each agent follows rules governing decisions based on environmental conditions. The environment in which agents live can be spatial or networked, which plays a key role in shaping the interaction patterns between agents and their environment. Among many known HPC ABS platforms, Mesa is a Python-based ABM framework that provides scheduling, spatial environments, and data-collection capabilities. Therefore, utilizing Mesa-Python as a driver, a co-simulation framework can be developed to integrate EnergyPlus with ABM tools to capture both building physics and occupant dynamics.

Previous co-simulation research demonstrates significant potential for energy savings. Li et al. achieved 29.15% electricity savings using PPO algorithms for temperature and humidity control [6]. Research presented a multi-agent deep reinforcement learning. With a step-by-step decision-making approach for demand response, this method reduces energy costs by 12% and peak energy demand by 15% while maintaining occupant comfort at 94% [7]. The research of Li et al. proposed a joint simulation framework, EnergyPlus-Fluent (CFD), to address thermal stratification, resulting in up to 43.5% savings in cooling energy [6]. The research by Wang et al. presented EnergyPlus-CONTAM to simulate heat and indoor air quality (IAQ) and assess the risk posed by multiple pollutants [8]. However, the joint simulation still faces limitations, including high energy consumption during data exchange and delays in data transfer, which lead to inaccuracies in translating building occupant behavior into energy consumption results [4]. However, these studies acknowledge but do not quantify the energy consumption of the simulation infrastructure itself.

2.2 High-Performance Computing for Large-Scale Simulation

High-performance computing (HPC) is increasingly applied to simulations of large-scale and complex buildings. For example, research using agent-based modeling (ABM) must consider a wide range of parameters and the complex behavior of individual occupants in each agent to accurately reflect human behavior. Both ABM and joint simulations often require minute-scale temporal resolutions over long simulation periods (e.g., one year), resulting in tens of billions of floating-point operations (FLOPs) per agent per time step. Acceleration techniques can significantly and accurately reduce the simulation time of large-scale systems. Distributing the computational load across multiple processors can reduce the overall simulation time, which is crucial for scaling simulations to city scales.

PyTorch Distributed Data Parallel (DDP) is a framework for training distributed models. It has scale-based computation, enabling efficient computation on GPUs or multiple machines [6]. The core mechanisms of DDP include process group, gradient synchronization, and the ring all-reduce algorithm¹. A study by Tampuu et al. showed that

¹ Patarasuk, P. & Yuan, X. Bandwidth Optimal All-reduce Algorithms for Clusters of Workstations.

distributing the computation of agents can significantly speed up learning and reduce training time by up to tenfold. Applying DDP to a co-simulator system poses three main challenges: (1) the need to synchronize time between simulators with different time steps, which can cause bottlenecks; (2) uneven workload distribution, where agents have different computational complexities; This leads to idle wait times between processes, and (3) complex communication patterns between agents [10, 11, 12]. Therefore, HPC profiling is important to optimize the performance of programs executed on NVIDIA Nsight Systems, a dedicated profiling tool for NVIDIA GPU-based applications². HPC profiling facilitates various performance tuning strategies, such as dynamic load balancing during runtime, which utilizes memory and algorithmic adaptations. The work of Tallent et al. presents HPCToolkit, a performance analysis tool for HPC applications that can identify bottlenecks and provide optimization recommendations to optimize processing power.

2.3 Efficient Energy-Aware Data Movement in Co-Simulation

Data exchange between models is a major source of energy consumption in co-simulation. Gomes et al. noted that inter-simulator communication adds computational load through higher CPU and memory use [4, 5]. Karnouskos observed processing times increase by 20-50% compared to single-simulator runs [9, 12]. Schweiger et al. showed that choosing the timestep involves a trade-off between accuracy and energy: larger steps lower workload and energy but reduce accuracy, while smaller steps improve accuracy but greatly raise energy use [6].

Network-based data transfer in distributed co-simulation consumes substantially more energy than in-memory communication. Koomey's analysis of data center growth estimated that transmitting 1 GB across networks consumes approximately 0.06-0.2 kWh, depending on distance and network type [5, 7]. Given that co-simulations can execute thousands of data exchanges per run, this cumulative transmission overhead constitutes a significant energy footprint requiring optimization.

Despite recognition of these challenges, existing literature lacks quantitative analysis linking specific simulation patterns to energy costs. High-frequency data exchange necessary for behavioral modeling requires acceptable temporal resolution, yet the energy implications remain unquantified. Our work addresses this gap by profiling an integrated building co-simulation to establish concrete relationships between modeling choices and computational energy consumption..

3 TWIN-B BUILDING CO-SIMULATION

This section describes the Twin-B digital twin architecture, which combines EnergyPlus building energy simulation with Mesa agent-based occupant modeling on PyTorch DDP infrastructure. We detail the system components, data flow, and critically—the mapping from building simulation decisions to GPU operations that drives the profiling analysis in Section 4.

3.1 Case Study: Boonchoo Resolution Building at Thammasat University, Lampang Thailand

Twin-B models the Boonchoo Building at Thammasat University's Lampang Campus, Thailand (Northern weather zone 483780, coordinates 18.277°N 99.504°E, climate zone 7.0). Operational since 2018, the building provides detailed materials and maintenance records enabling accurate digital twin construction. The building accommodates educational activities with variable occupancy patterns—a scenario ideal for testing energy management strategies.

The digital twin, constructed in SketchUp and imported to EnergyPlus, divides the building into 73 thermal zones based on physical structure and usage patterns. Our demonstration model focuses on three representative classrooms that can accommodate 1,875 occupants at full capacity. This scale—nearly 2,000 agents interacting with 73 zones—creates the computational complexity necessary for meaningful HPC profiling.

3.2 Data Preparation and Model Development

Figure 1 also shows the input data required to simulate both the physical building structure and the occupants' behaviour. These included patterns and durations of classroom use by students and teachers, air conditioning usage

² NVIDIA Nsight Systems user guide. (2023).

behavior (on/off patterns), building occupant density during different periods and class schedules, building structure data and air conditioning system specifications, and, finally, temperature and climate data for Lampang Province, collected from the Climate.OneBuilding website.

EnergyPlus begins by creating a 3D model of the building from real architectural data, defining the usable spaces and thermal zones. Material properties are then described, with heat transfer coefficients (U-values) for walls, roofs, and windows determined to ensure accurate heat loss calculations. HVAC system parameters are then described, including the type of air conditioner, coefficient of performance (COP), and operating schedules. Finally, real-world weather data for Lampang Province is combined with the EnergyPlus Weather Format (EPW) to simulate local environmental conditions accurately.

Mesa designing agents, each representing a student or instructor with unique behavioral characteristics, such as classroom entry and exit patterns or air conditioning usage. Behavioral rules for each agent are then determined based on contextual factors such as indoor temperature, schedule, and comfort needs.

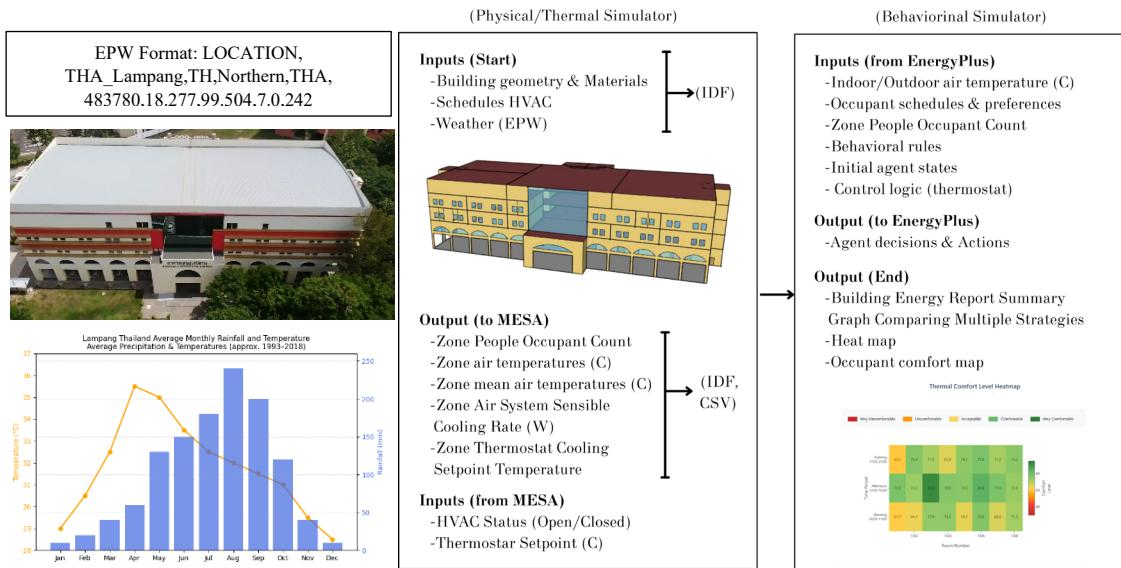


Figure 1: Input and output between EnergyPlus and the Mesa Agent Based Model

3.3 Co-simulation Integration and Synchronization

The co-simulation process works by receiving environmental data from EnergyPlus through a callback API that connects the two models. The simulation begins with EnergyPlus running the building model and sending zone temperature values to the agents. Once the agents receive the temperature data, they execute the decision-making processes for each agent within the Mesa model.

Each agent reads zone temperatures and calculates comfort levels to decide whether to control the air conditioner (on/off or set to a specific cooling value) and saves the results for managing large simulations. DDP is used to distribute the calculations across multiple cores/GPUs for increased efficiency. Finally, the average temperature request from all agents is sent back to EnergyPlus for actual control.

Both EnergyPlus and Mesa interact on a cycle-by-cycle basis, with EnergyPlus first calling Mesa to process and calculate the new desired temperature, which EnergyPlus then uses in the next cycle of energy calculations.

This research utilizes a master-slave synchronization strategy. EnergyPlus (the master) runs the building simulation. When it reaches a point where it needs input from the agent, the system pauses and sends a signal to the orchestrator.

The orchestrator acts as an intermediary, extracting current environmental data from EnergyPlus and transmitting it to Mesa (the slave). Mesa then models occupant behavior and sends the decision results back to EnergyPlus for processing and calculations at the next time point.

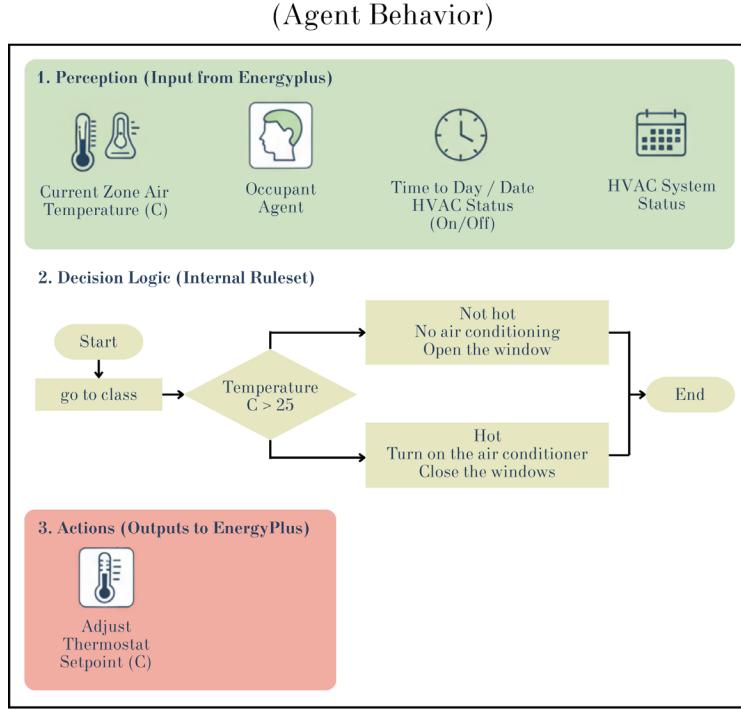


Figure 2: Agent Decision-Making Behavior

3.4 From Building Simulation to GPU Operations

The master-slave synchronization and data flow described above translate into specific GPU computational patterns within the Mesa model. When the orchestrator receives environmental data from EnergyPlus and transmits it to Mesa, each of the 1,875 occupant agents appears on the GPU as a PyTorch tensor with all necessary behavioral and thermal attributes. Each agent record includes an integer ID and zone index, four single-precision floats for preferred temperature, comfort tolerance, current temperature received from EnergyPlus, and requested setpoint, along with a Boolean flag indicating air-conditioner use. This structure takes up approximately 25 to 40 bytes per agent, depending on alignment.

Table 1: Mapping of Building Simulation Decisions to GPU Operations

Building Decision	GPU Manifestation
1,875 individual agents	37.5-byte per-agent transfers
5-minute timesteps (288/day)	Forced synchronization each step
73 thermal zones	NCCL all-gather for aggregation
Per-agent decision autonomy	Sequential tensor operations
Real-time coupling	Blocking queue operations

As shown in Table 1, each architectural decision in the building simulation generates corresponding GPU operational requirements. The orchestrator's transmission of zone temperatures to Mesa triggers a series of GPU operations. Temperature data transfers from CPU memory to GPU memory, where the distributed agents are located across two GPUs

using PyTorch's DDP framework. Each agent receives its respective zone temperature based on its position within the 73-zone building layout. The individual tensor representation per agent (Table 1, row 1) means that agents perform their comfort calculations independently, each executing a tensor operation that evaluates $\text{comfort_level} = \max(0.0, \text{preferred_temp} - \text{abs}(\text{current_temp} - \text{preferred_temp}))$ to decide whether to request air conditioning.

Following individual agent decisions, the system combines results to produce the zone-level control signals needed by EnergyPlus. This combination maps each agent's setpoint request to its corresponding zone and determines the lowest requested temperature per zone. Since DDP distributes agents across two GPUs, the partial results from each GPU must be synchronized via NCCL collective communication operations (Table 1, row 3) to produce complete zone setpoints.

The cycle concludes when these aggregated setpoints are sent back through the orchestrator to EnergyPlus. The real-time coupling and enforced synchronization at each timestep (Table 1, rows 2 and 5) mean EnergyPlus remains blocked until it receives all zone setpoints, while Mesa agents cannot start their next decision cycle until new temperature data arrives. This creates a closely coupled execution pattern that repeats for each of the 288 daily timesteps, with the orchestrator managing bidirectional data flow between CPU-based building physics and GPU-based agent computation at every simulation step.

4 PROFILING ENERGY CONSUMPTION DURING DATA EXCHANGE

Experiments were carried out on the GPU partition of the ThaiSC LANTA supercomputer in two stages to measure the energy efficiency of dataflow in co-simulation environments. This reflects how data exchange between simulators influences simulation accuracy. First, we identify the platform's characteristics, including computational capability, data transfer rates, and energy costs for all operation types (compute, transfers, communication). Second, we use NVIDIA Nsight Systems (nsys and nsys-ui) to profile the Twin-B during co-simulation of energy consumption in a university building over three days. After completing these two steps, we analyze the effect of data exchange between the simulators on the energy profiles and simulation accuracy discussed in section 5. Table 2 provides the details of the experimental setup used for these profiling experiments.

Table 2: Experimental Setup and Configuration

Component	Specification
Hardware Platform	ThaiSC LANTA Supercomputer (HPE Cray EX)
GPUs	2× NVIDIA A100-SXM4-40GB (Compute Capability 8.0)
CPU	AMD EPYC 7713 64-Core Processor
Memory	64 GB System RAM
Interconnect	PCIe Gen4 16x (32 GB/s), NVLink (400 GB/s)
Software Stack	
Building Physics	EnergyPlus 25.1.0 with pyenergyplus API
Agent Framework	Mesa 2.1.5
GPU Framework	PyTorch 2.2.2 with DDP, CUDA 11.8, NCCL 2.12
Profiling Tools	NVIDIA Nsight Systems 2024.11 (HPC SDK 24.11), nvidia-smi
Profiling Parameters	
Trace Options	cuda, nvtx, osrt, openmp
GPU Metrics	All devices enabled
CUDA Memory	Usage tracking enabled
Resource Allocation	2 GPUs, 8 CPU cores via SLURM
Simulation Scope	3 days (864 timesteps), 8,011,365 events, 56 threads

4.1 Platform characteristics

The ThaiSC LANTA supercomputer (HPE Cray EX) features 704 NVIDIA A100 SXM4 40 GB GPUs. Each GPU node is equipped with an AMD EPYC 7713 64-Core Processor and offers Compute Capability 8.0, CUDA 11.8, and NCCL networking backend. GPU compute capability is benchmarked using matrix multiplication across different sizes (1024×1024 to 8192×8192), with each run performing 10 iterations and CUDA synchronization for accurate timing. Performance is reported in GFLOPS reaching a peak of 18,835 GFLOPS (96.6% of the theoretical maximum) and a high compute efficiency of 247 GFLOPS/Watt at an average power of 76.2W. Memory bandwidth profiling highlights significant bottlenecks: large transfers (1MB) achieve 9.2 GB/s in Host-to-Device (H2D) and 8.1 GB/s in Device-to-Host (D2H), with a PCIe efficiency of 29%, while Device-to-Device transfers reach 64.3 GB/s, which is over 7 times faster. Small transfers, averaging 37.5 bytes, reach only 1.64 MB/s (0.005% of peak bandwidth) due to a fixed 20.89 μ s PCIe overhead, leading to 99.995% bandwidth underutilization (as shown in Figure 3). Figure 4 shows the energy profiling, which indicates that compute operations consume 76.2W, small transfers 56.5W, and NCCL communication 55.7W. For the 6,289 observed H2D operations at 37 bytes, the total overhead is 131.4 ms and 7.42 J of energy.

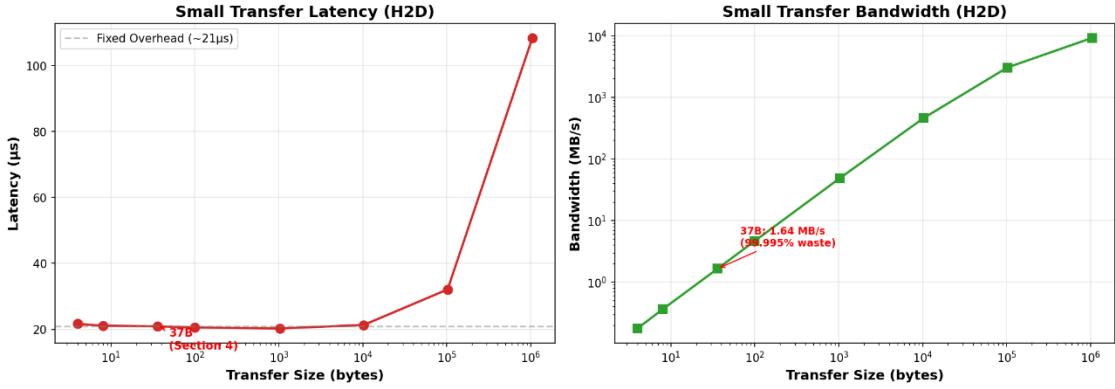


Figure 3: Small transfer overhead analysis (1B-1MB). Left: Latency remains constant $\sim 21 \mu\text{s}$ for transfers $< 10\text{KB}$, indicating fixed PCIe overhead dominates. Right: Bandwidth scales logarithmically, with 37-byte transfers achieving only 1.64 MB/s (0.005% of peak).

Red annotation marks Section 4's identified 37.5-byte average.

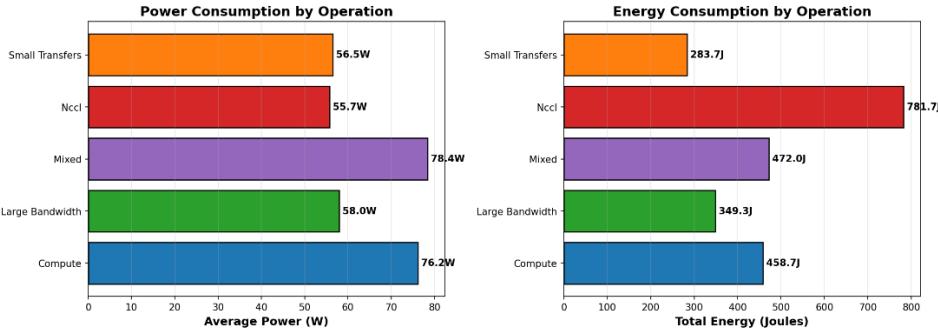


Figure 4: Energy consumption by operation type. Left: Average power consumption shows compute and mixed workloads consume highest power (~76-78W). Right: Total energy (joules) varies by operation duration and power draw. NCCL shows highest total energy due to longer duration (14s vs 6s for others).

4.2 Twin-B Co-Simulation Profiling

In the second step of the experiment, we use nsys to profile the Twin-B during co-simulation of energy consumption in a university building over three days using the SLURM job with a default resource allocation of two GPUs, 8 CPU cores (4 cores per GPU), and 64 GB of RAM. The environment was set up by loading the Python module, launching the virtual environment, specifying the EnergyPlus path, and creating a results directory. The EnergyPlus simulations used IDF files and weather files to generate CSV files for model training. Profiling was performed using NVIDIA Nsight Systems (nsys) under the torchrun command to collect comprehensive data (e.g., CUDA API calls, GPU kernel, OS runtime, GPU, and memory metrics). Additionally, GPU power was recorded every 1 second using nvidia-smi.

The profiling analysis indicates that the total GPU kernel runtime is approximately 25.2 ms across 4,046 kernel instances, while CUDA API overhead accounts for up to 66.3% of the time, primarily due to cudaStreamSynchronize (4.47 seconds in total). Data transfers total 2.95 GB (2.32 GB Device-to-Host and 0.32 GB Device-to-Device), and communication is dominated by NCCL AllGather (580 occurrences, 207.9 ms) and AllReduce (578 occurrences, 17.6 ms). Host-side blocks from the OS runtime were also observed, totaling 6.3 seconds. These findings suggest a synchronization issue between the CPU and GPU, resulting in the workload being constrained by CPU limitations and GPU usage being restricted by host-side waiting patterns rather than computational inefficiency.

The analysis revealed four main bottlenecks. First, the overhead of Stream Synchronization is very high. cudaStreamSynchronize is called 547,393 times, taking 4.47 seconds, causing severe GPU and CPU concurrency blocking. Second, the NCCL communication shows that ncclAllGather and ncclAllReduce consume nearly 64.4% of the GPU kernel time, indicating high communication frequency. Third, the kernel is not fully computable, with actual computation operations occupying less than 30% of total kernel time. Finally, the memory copy pattern is highly volatile. A large number of memory copies (548,037) are called, but the average size is only 37.5 bytes per copy, which impacts DMA efficiency and causes CPU cache thrashing. These small transfers achieve only 0.005% of theoretical PCIe bandwidth, wasting 99.995% of available transfer capacity while consuming significant energy due to fixed per-transfer overhead.

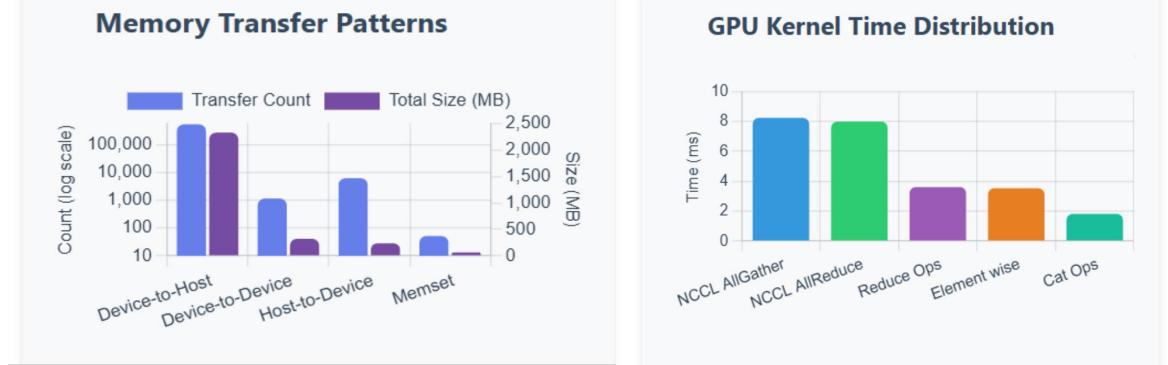


Figure 5: Memory transfer patterns and GPU kernel time distribution during co-simulation are shown. Frequent small CPU–GPU copies (~2.95 GB) reduce DMA efficiency and cause CPU-side blocking. Most delays stem from CUDA synchronization and NCCL communication, indicating performance is limited by CPU–GPU coordination rather than GPU computation.

5 RESULTS AND DISCUSSION

This section presents the profiling results from the Twin-B co-simulation framework and analyzes their implications for the design of building energy management systems. Figure 4 depicts part of the results generated from the Twin-B simulation, showing the occupants' comfort levels when building energy-saving measures are enforced for all occupants. This confirms that the co-simulation approach can be used to help building administrators evaluate the effectiveness of diversion policies across different scenarios. To ensure the efficient use of HPC resources, we examine the bottlenecks

identified during the three-day simulation of the Boonchoo Building and evaluate optimization opportunities for achieving computational sustainability in digital twin deployments.

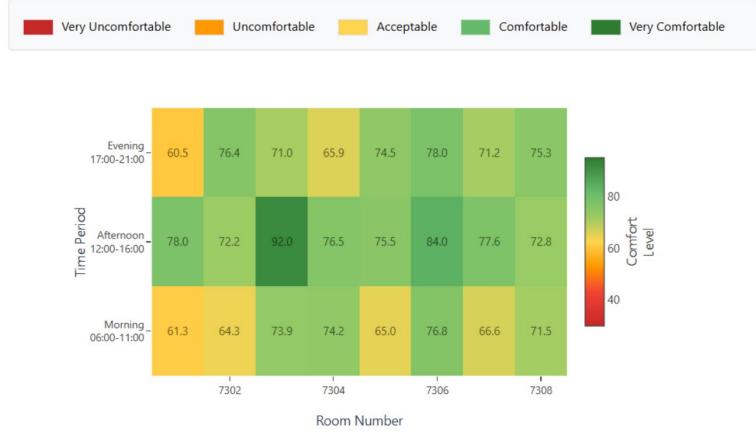


Figure 4: heatmap of thermal comfort levels obtained from the co-simulation run

5.1 Co-simulation Performance Analysis

The profiling of Twin-B revealed significant inefficiencies in resource utilization during the co-simulation of 1,875 agents across 73 thermal zones. Table 3 summarizes the key performance metrics observed during the three-day simulation period (864 timesteps).

Table 3: Twin-B Profiling Results Summary

Metric	Observed Value	Performance Impact
Total GPU kernel runtime	25.2 ms	0.4% of wall time
cudaStreamSynchronize calls	547,393	66.3% of CUDA API time (4.47s)
Average H2D transfer size	37.5 bytes	0.005% PCIe efficiency
Total H2D transfers	6,289	131.4 ms overhead
Total data transferred	2.95 GB	2.32 GB D2H, 0.32 GB D2D
NCCL operations	1,158	64.4% of GPU kernel time
Primary process CPU usage	96.99%	Severe load imbalance
Host-side blocking time	6.3 seconds	System-level synchronization
Energy Impact		
Energy per H2D transfer	1.18 mJ	Total: 7.42 J
Compute operations	76.2W	optimal efficiency: 247 GFLOPS/W
Small transfers	56.5W during H2D operations	74% of compute power for 0.005% efficiency
NCCL communication	55.7W during collective operations	reasonable for distributed execution

The results confirm that co-simulation overhead significantly exceeds the 20-50% increase cited in literature. The system spends only 25.2 ms on actual GPU computation while accumulating 4.47 seconds of synchronization overhead—a 177× ratio between waiting and computing. This extreme imbalance validates our hypothesis that data exchange patterns, not computational complexity, limit co-simulation performance.

Data exchange patterns showed four sources of bottlenecks: small transfer inefficiency, synchronization overhead, communication dominance and cpu-gpu imbalance. The 6,289 H2D transfers averaging 37.5 bytes directly correspond to the per-agent data structure described in Section 3.4. Each transfer achieves only 1.64 MB/s (0.005% of PCIe peak bandwidth), with a fixed 20.89 μs overhead dominating transfer time. This pattern consumes 7.42 J total energy—

equivalent to 29,680 FP32 operations on the A100 GPU. For perspective, this energy could perform the comfort calculations for all 1,875 agents 15 times over if properly batched. The 547,393 cudaMemcpySync calls reveal that each agent decision triggers immediate CPU-GPU synchronization rather than allowing asynchronous execution. With 1,875 agents making decisions at 288 timesteps per day, this results in approximately 2 synchronization points per agent-timestep pair, forcing sequential execution of inherently parallel operations. NCCL collective operations (580 AllGather, 578 AllReduce) consume 64.4% of GPU kernel time despite individual operations completing in 14-17 μ s. This high percentage reflects operation frequency rather than inefficiency—the 73-zone aggregation requires global synchronization at every timestep. The primary Python process consumes 96.99% of CPU resources, while the secondary process uses only 2.02% revealing a severe load imbalance.

The energy profiling reveals three distinct consumption patterns: compute operation, small transfers, and NCCL communication, also shown in Table 3. The critical insight is that small transfers consume nearly the same power as computation while delivering negligible data throughput. Across the three-day simulation, transfer overhead alone accounts for $7.42 \text{ J} \times 288 \text{ timesteps/day} \times 3 \text{ days} = 6.41 \text{ kJ}$. For a sensitivity analysis requiring 10,000 runs, this overhead scales to 64.1 MJ (17.8 kWh)—equivalent to a typical household's daily electricity consumption, spent entirely on moving 37-byte messages.

5.2 Optimization Strategies Based on Profiling Evidence

The profile results emphasize the need to combine the 6,289 individual agent transfers into 288 batched operations, one per timestep. This approach will increase transfer size from 37.5 bytes to approximately 820 bytes (1,875 agents \times approximately 25 bytes / 73 zones) and will boost PCIe efficiency from 0.005% to around 2.5%. Minimal code changes, such as replacing per-agent tensor creation with population-level arrays, have been implemented, maintaining identical simulation logic and accuracy. With this optimization, we cut energy consumption by 95% (from 7.42 J to 0.37 J per simulation day) and eliminated over 546,000 synchronization points.

Future work is planned to address the 6.3-second host blocking caused by fundamental changes to the co-simulation architecture. Several optimization opportunities are considered, such as implementing producer-consumer queues for EnergyPlus-Mesa communication, using CUDA streams to overlap computation and communication, batching multiple timesteps before synchronization, or exploring GPU-native building physics to eliminate the CPU-GPU boundary.

6 CONCLUSION

This paper presents Twin-B, a co-simulation framework integrating EnergyPlus building physics with Mesa agent-based occupant modeling on PyTorch DDP, deployed as a profiling testbed on the LANTA supercomputer. Through comprehensive NSys profiling of 1,875 agents across 73 zones over three days, we establish quantitative relationships between building simulation architecture and HPC performance characteristics. The current implementation's energy inefficiency undermines the sustainability goals of building optimization. If co-simulation consumes 17.8 kWh per sensitivity analysis while identifying HVAC improvements saving 20 kWh annually, the computational cost approaches the operational savings. This finding emphasizes that efficient HPC utilization is not merely a performance concern but essential for net-positive environmental impact.

REFERENCES

- [1] Chen, Z., Tao, Z. & Chang, A. A data-driven approach to optimize building energy performance and thermal comfort using machine learning models. ACM International Conference Proceeding Series 464–469 (2021) doi:10.1145/3473714.3473794.
- [2] Hong, T., Taylor-Lange, S. C., D’Oca, S., Yan, D. & Coragnati, S. P. Advances in research and applications of energy-related occupant behavior in buildings. Energy and Buildings 116, 694–702 (2016).

- [3] Delzendeh, E., Wu, S., Lee, A. & Zhou, Y. The impact of occupants' behaviours on building energy analysis: A research review. *Renewable and Sustainable Energy Reviews* 80, 1061–1071 (2017).
- [4] Gomes, C., Thule, C., Broman, D., Larsen, P. G. & Vangheluwe, H. Co-Simulation. *ACM Computing Surveys* 51, 1–33 (2019).
- [5] Koomey, J. G. GROWTH IN DATA CENTER ELECTRICITY USE 2005 TO 2010. <http://www.koomey.com><http://www.analyticspress.com/datacenters.html> (2011).
- [6] Li, Z. et al. Reinforcement learning-based demand response strategy for thermal energy storage air-conditioning system considering room temperature and humidity setpoints. *Journal of Energy Storage* 72, (2023).
- [7] Han, Y., Gao, W., Wang, Z. & Zhao, Q. Optimizing grid-interactive buildings demand response: Sequence-based decision-making multi-agent policy decomposition deep reinforcement learning. *Energy and Buildings* 347, (2025).
- [8] Wang, Y. et al. Investigating the impacts of home energy retrofit on the indoor environment through co-simulation: A UK case study. *Journal of Building Engineering* 100, (2025).
- [9] Karnouskos, S. Cyber-Physical Systems in the SmartGrid. in 2011 9th IEEE International Conference on Industrial Informatics 20–23 (IEEE, 2011). doi:10.1109/INDIN.2011.6034829.
- [10] Blochwitz, T. ; et al. Functional Mockup Interface 2.0: The Standard for Tool independent Exchange of Simulation Models. 173–184 (2012) doi:10.3384/ecp12076173.
- [11] Pianpak, P., Li, J. & Son, T. C. Load Balancing in Distributed Multi-Agent Path Finder (DMAPF).
- [12] Rashid, T. et al. Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. *Journal of Machine Learning Research* 21, 1–51 (2020).
- [13] National Renewable Energy Laboratory (NREL). "EnergyPlus™.", Sep. 2017. National Renewable Energy Laboratory (NREL) (2017).