**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

Asst. Prof. Dr. Worawan Diaz Carballo

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

Abstract

Improving energy efficiency through co-simulation frameworks can improve building and operations management, a key approach to reducing energy consumption per square meter by 35% by 2030. Buildings consume more than one-third of global energy consumption, with HVAC systems accounting for more than 70%. Unpredictable occupant behavior can lead to energy variability of 100-300%, even for similar buildings and systems. To improve the prediction accuracy of building energy management, a co-simulation that combines physical models with surrogate human behavior models is required. To handle this complexity and maximize simulation accuracy, HPC, coupled with AI accelerators, is required. However, energy-aware data movement analysis found that frequent data exchange and synchronization create bottlenecks that increase processing time by 20-50% compared to stand-alone simulations. This problem is exacerbated in distributed systems, where network data transfer consumes 0.06-0.2 kWh/GB. An experimental study conducted in a classroom at the Boonchoo building, Thammasat University, Lampang , Thailand, showed that the frequency of data exchange between simulators and the temporal resolution of synchronization have a significant influence on the simulation results, i.e., higher connection frequency increases the processing time and power consumption, lower connection frequency reduces the processing time and power consumption but reduces the accuracy. The Twin-B model serves as a testbed for HPC profiling, identification processing bottlenecks and energy-intensive processes, which can be used to inform data exchange optimization and load balancing strategies. This research lays the foundation for the development of energy-efficient simulation and data management systems that support the broader objective of sustainable computing.

**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 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, demanding, and complicated, which can cause inaccuracies in translating behavior into energy results [4].

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 important for simulation accuracy and energy efficiency. Smaller time steps enhance precision but elevate energy demand due to more frequent exchanges, while larger steps decrease computational overhead but may fail to capture transient events. Distributed co-simulation introduces further inefficiency through 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. The co-simulators via real-time data exchange and synchronization 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 to create 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.

1. A Co-Simulation Framework for Building Energy Management
   1. Technologies and Tools for Building Energy Simulation

Building Energy Simulation (BES) is a simulation tool used to analyze and predict energy use to achieve sustainability goals. It considers factors such as lighting, air conditioning, building structure, occupant behavior, and weather conditions. Commonly used simulation tools include:

EnergyPlus (EP) is a dynamic energy simulation tool. It uses mathematical and computer models to calculate heat transfer and analyze whole-building energy. It calculates PMV (Predicted Mean Vote), lighting conditions, and energy consumption to simulate the physical properties of the building and its control systems. It uses physics-based modeling techniques to analyze energy flow within a building. It uses input data files such as building materials and HVAC systems, and uses EnergyPlus Weather files to evaluate weather conditions in the simulation. EnergyPlus has three main components: The Simulation Manager, which controls the entire simulation process, controls the execution of submodules, and handles code issues. The next module, the Heat and Mass Balance Simulation Module, calculates and simulates the heat balance of both interior and exterior building surfaces and simulates daylighting. The Building Systems Simulation Manager is a module that controls the simulation of heating, ventilation, air conditioning, electrical systems, and climate control systems in zones where the user can manually set the air conditioning system. However, simulating occupant behavior has significant limitations, which affect the accuracy of energy forecasts.

Agent-based simulation (ABS) is a complex modeling approach by simulates autonomous units called agents, which have their own behavior and decision-making capabilities. Agent properties and behaviors are often determined by rules that govern the decisions and actions of individual agents. 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.

Due to its complexity and significant impact, co-simulation has been developed to integrate building physics models with human behavior models, such as EnergyPlus and Mesa agent-based models (ABMs). Co-simulation offers the advantages of high flexibility due to its modular nature, which facilitates parallel model development, reusability, and integration of hardware/software/human-in-the-loop simulations.

Digital twins enable researchers to analyze trends, predict behavior, and plan accurate energy management strategies. Previous research has used this data for real-time monitoring or forecasting of energy management. Research used a PPO algorithm to simultaneously control temperature and humidity, achieving 29.15% electricity savings and 46.11% operating costs [6]. Research presented a multi-agent deep RL. With a step-by-step decision for demand response, this method reduces energy costs by 12% and peak energy demand by 15% while maintaining occupant comfort up to 94% [7]. The research of Li et al. proposed a joint simulation framework, EnergyPlus-Fluent (CFD) to solve the thermal stratification problem, resulting in up to 43.5% cooling energy savings [6]. The research of Wang et al. presented EnergyPlus-CONTAM to simulate heat and indoor air quality (IAQ) and assess the risk of multiple pollutant [8]. However, the joint simulation still faces the limitations of high energy consumption in the data exchange process, delays in data transfer, which lead to inaccuracies in translating building occupant behavior into energy consumption results [4].

* 1. High-Performance Computing in Building 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 algorithm[[1]](#footnote-1) . A study by Tampuu et al. showed that 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 applications[[2]](#footnote-2). 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.

* 1. Challenge in Achieving Efficient Energy-Aware Data Movement within Co-Simulation Frameworks

Distributed co-simulation introduces further inefficiency through 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. A significant energy consumption issue arises during the exchange of data between models. The research of Gomes et al. stated that inter-simulator communication imposes a computational burden due to increased CPU and memory usage [4]. The findings of Karnouskos also found that processing time increases by 20–50% compared to single-simulator simulations [9]. The research of Schweiger et al. misselected time steps, resulting in larger time steps reducing workload and energy consumption at the expense of accuracy, while smaller time steps increase accuracy but significantly increase energy consumption. Hybrid-driven distributed co-simulation is characterized by network energy costs. Data transfer across networks consumes significantly more energy than intra-memory communication within a single machine. According to the report of Koomey estimated that transmitting 1 GB of data over a network consumes approximately 0.06–0.2 kWh of energy, depending on the distance and type of network [5]. This emphasizes the need for researchers to balance model accuracy with the carbon footprint of the processing. There is a lack of clear quantitative analysis of the evidence of energy savings. In modeling different occupant behaviors, high-frequency data exchange is essential to obtain accurate results. Such behavioral modeling requires very fine temporal resolution to capture decision-making processes realistically. Even in frameworks that use middleware for model integration, the overall modeling process remains complex and requires significant resources and processing time.

1. Twin-B Building co-simulation

This section describes the simulation model architecture of the Twin-B, a digital twin building simulation that combines Mesa agent-based modeling (ABM) with EnergyPlus building energy simulations. The system models occupant behavior and thermal comfort in a multi-zone building, using distributed computing with PyTorch DDP for scalability. Figure 1 shows the proposed simulation architecture as a testbed for evaluating different energy-saving policies at Thammasat University, Lampang Campus, Thailand, located in the Northern weather zone (code 483780), with coordinates 18.277, 99.504, and a climate zone of 7.0, 242. The building has been operational since 2018, so we can still obtain detailed materials and maintenance records. The digital twin of the building has been constructed in SketchUp, divided into 73 zones based on its physical structure and usability. Our demo model simulates three rooms that could accommodate a total of 1,875 occupants at full capacity.

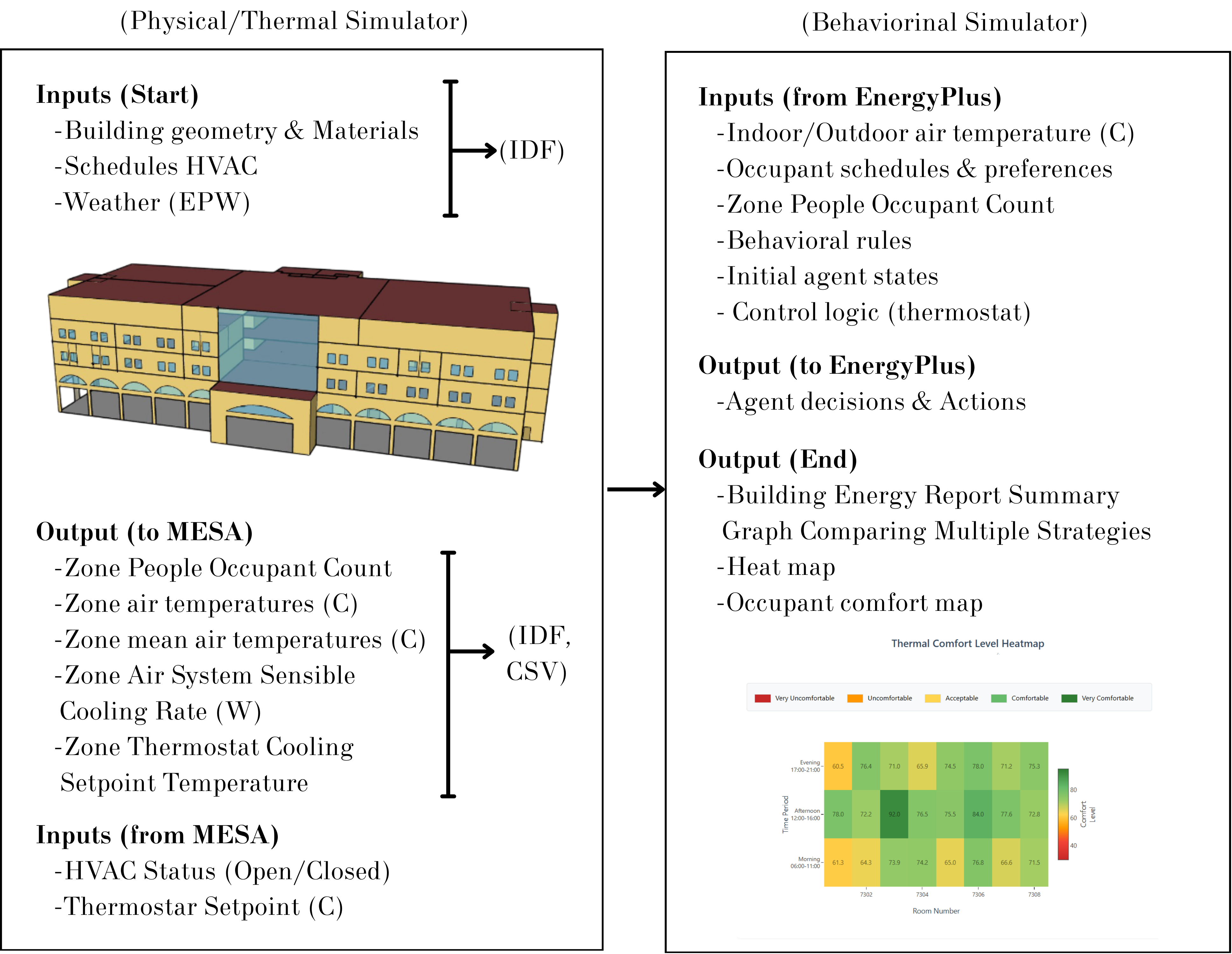
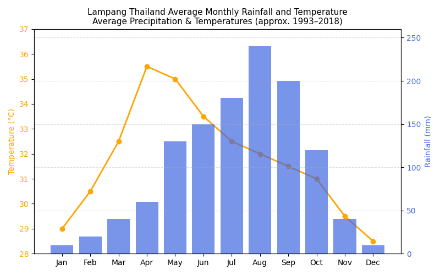
* 1. Data Preparation and Model Development

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 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.

A building with trees in the background

AI-generated content may be incorrect.

EPW Format: LOCATION,  
THA\_Lampang,TH,Northern,THA,  
483780,18.277,99.504,7.0,242

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

3.2 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.

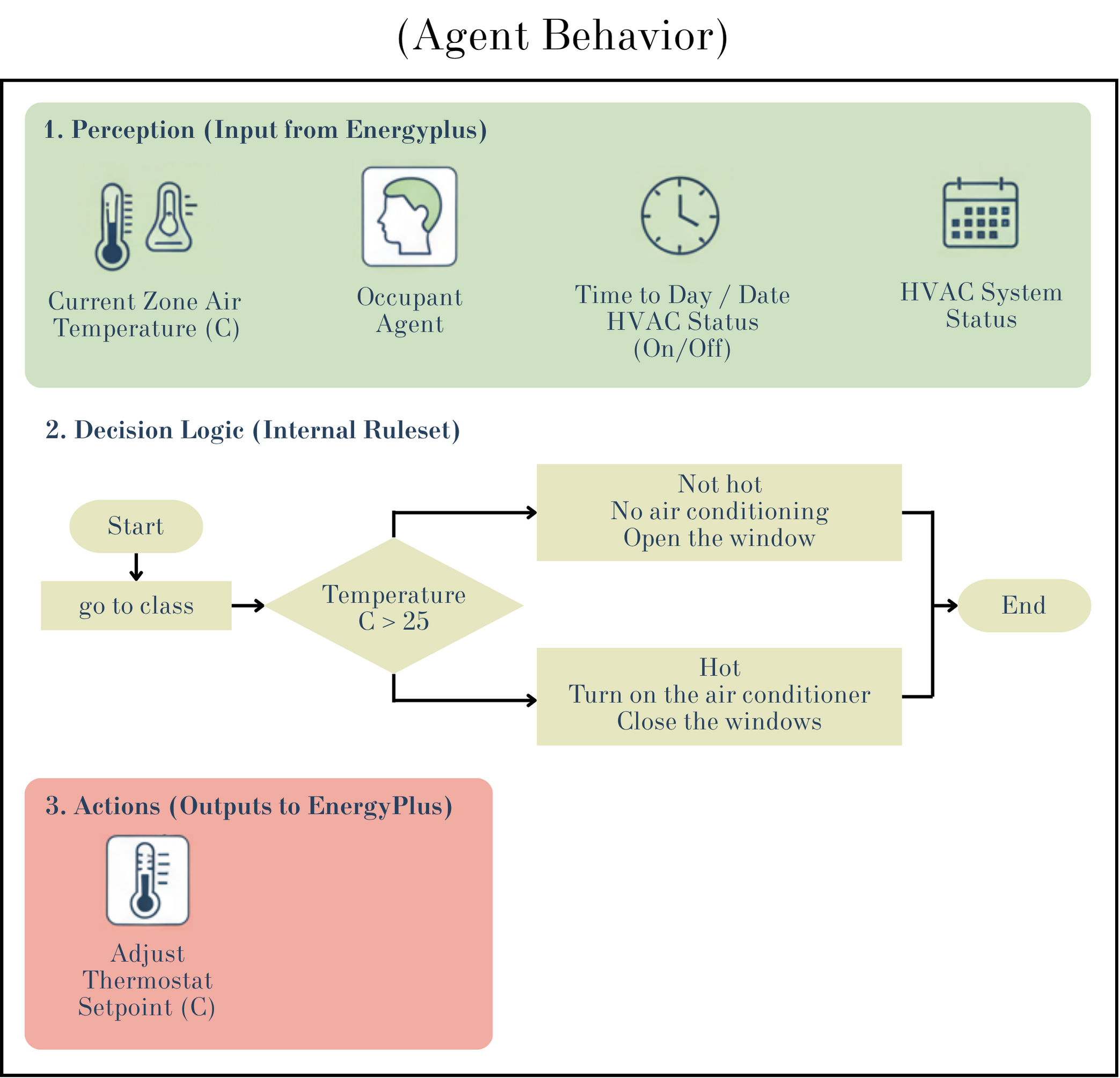


Figure 2: Agent Decision-Making Behavior

3.3 Validation and Experimentation

The Twin-B system uses a two-step verification method: Verification: Verifies the correctness of the model and code implementation, including the connection and synchronization between EnergyPlus and Mesa. Validation: Verifies the correctness of the results by comparing the modeled energy consumption with actual building data (e.g., electricity bills),and focuses the theoretical verification of the simulation system only on the cooling load from air conditioning, as it is the largest energy-consuming component in a building.

The simulation experiment consisted of five building operation scenarios to capture diverse energy usage patterns:

(1) Regular semester, weekday. This scenario represents Monday-Friday operations throughout the semester with a total of 903 building occupants. This scenario served as a baseline case study for comparing energy efficiency and evaluating energy management strategies.

(2) Exam period. This scenario simulated midterm and final exam week with 665 occupants. During this time, activities are concentrated in exam rooms, overall movement is reduced, and comfort needs are increased to maintain focus and concentration.

(3) Conference/Conference Activity. This scenario tested maximum HVAC utilization with 972 occupants, including over 500 outside visitors. The building exceeded its design capacity in large spaces such as the main auditorium (120% capacity), and corridors and common areas were busy during breaks.

(4) Low-Use Weekend Scenario. This scenario represents Saturday-Sunday operations with only 189 occupants, primarily undergraduate students engaged in self-study group activities. This scenario is suitable for testing energy-saving measures under low-utility conditions.

(5) Minimum Operation During Summer Break. This scenario corresponds to the June-August period, when occupancy is at its lowest, with only 141 people. This scenario is suitable for testing the full potential of energy-saving strategies, especially during hot outdoor temperatures.

To evaluate the effectiveness of energy management strategies under these conditions, four policy interventions were designed and implemented:

(1) Minimum Room Activation Criteria. Establish a minimum utilization requirement before turning on the air conditioning system to prevent unnecessary HVAC operation in unoccupied rooms.

(2) Target Temperature Range Extension. Expand the target temperature range by ±1°C, ±2°C, and ±3°C above the baseline value to assess the impact of reducing HVAC load while maintaining acceptable thermal comfort.

(3) Mid-Term HVAC Break. Simulate a temporary shutdown of the air conditioning unit between classes, testing the shutdowns at 1.0 and 1.5 hours after the start of the class. With durations of 5, 10, and 15 minutes, this method takes advantage of the building's thermal inertia.

(4) Early HVAC shutdown. Simulate shutting down the air conditioning system 5, 10, and 15 minutes before class ends.

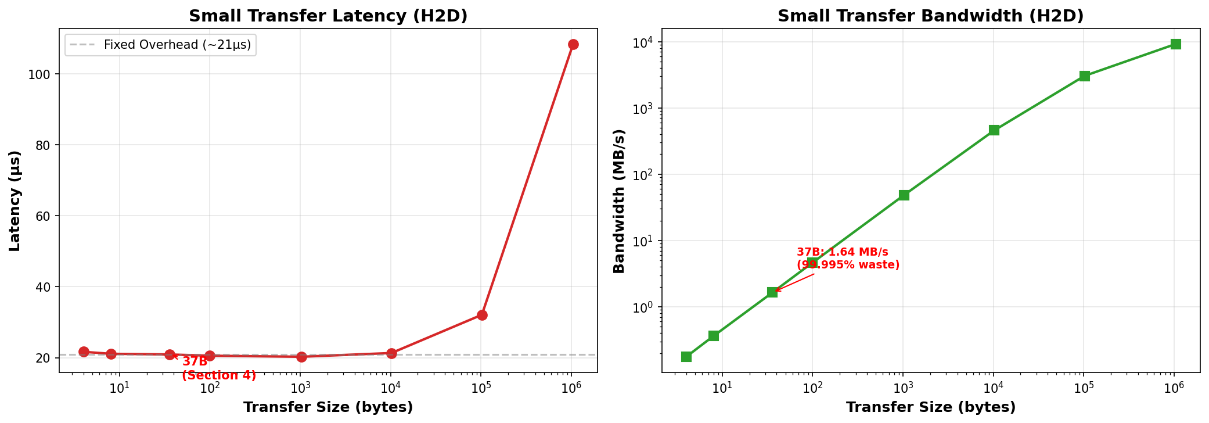
1. Profiling energy consumption during data exchange

Experiments were conducted on ThaiSC LANTA supercomputer’s GPU partition in three steps to quantify the energy efficiency of dataflow in co-simulation environments, reflecting how data exchange between simulators affects simulation accuracy. First, we identify the platform's characteristics, including computational capability, data transfer, 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. Third, we analyze the impact of data exchange between the simulators on the energy profiles and simulation accuracy.

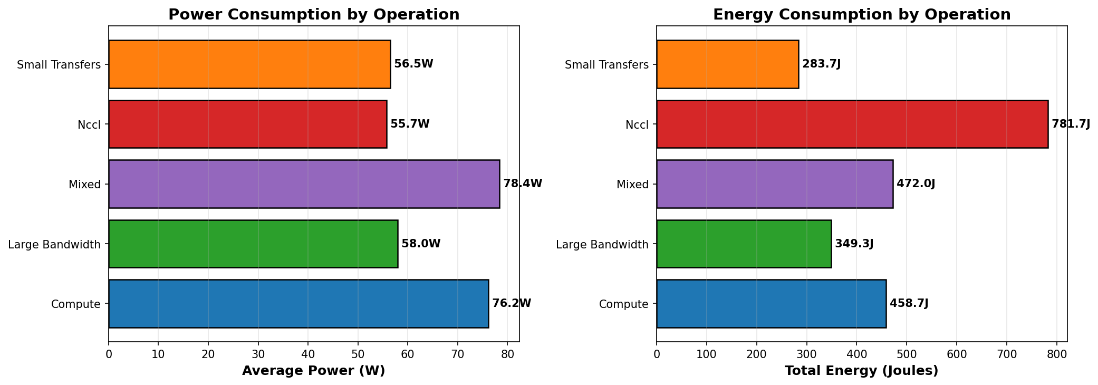
* 1. Platform characteristics

The ThaiSC LANTA supercomputer (HPE Cray EX) has 704 NVIDIA A100 SXM4 40 GB GPUs. Each GPU node uses an AMD EPYC 7713 64-Core Processor and provides Compute Capability 8.0, CUDA 11.8 with NCCL networking backend. We benchmark GPU compute capability using matrix multiplication operations across varying matrix sizes (1024×1024 to 8192×8192). Each test performs 10 iterations with CUDA synchronization to ensure accurate timing. Performance is measured in GFLOPS (billions of floating-point operations per second). Results achieve peak performance of 18,835 GFLOPS (96.6% of theoretical maximum) with excellent compute efficiency of 247 GFLOPS/Watt at 76.2W average power.

Memory bandwidth profiling reveals critical bottlenecks. Large transfers (1MB) achieve 9.2 GB/s H2D and 8.1 GB/s D2H (29% PCIe efficiency), while Device-to-Device reaches 64.3 GB/s (7× faster). However, small transfers matching the observed 37.5-byte average achieve only 1.64 MB/s (0.005% of peak) with fixed 20.89 μs PCIe overhead, representing 99.995% bandwidth underutilization. Energy profiling shows compute operations consume 76.2W, small transfers 56.5W, and NCCL communication 55.7W. For the 6,289 observed H2D operations at 37 bytes, total overhead is 131.4 ms and 7.42 J energy cost.



**Figure 3:** Small transfer overhead analysis (1B-1MB). Left: Latency remains constant ~21 μs for transfers <10KB, 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.



* 1. 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.

Table 1: Profiling Configuration

|  |  |
| --- | --- |
| Style Tag | Definition |
| Profiling Tool | NVIDIA Nsight Systems |
| SDK Version | HPC SDK 24.11 |
| Trace Options | cuda, nvtx, osrt, openmp |
| GPU Metrics | All devices (--gpu-metrics-devices=all) |
| CUDA Memory Tracking | Enabled (--cuda-memory-usage=true) |
| Events Collected | 8,011,365 |
| Total Threads Tracked | 56 |

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.

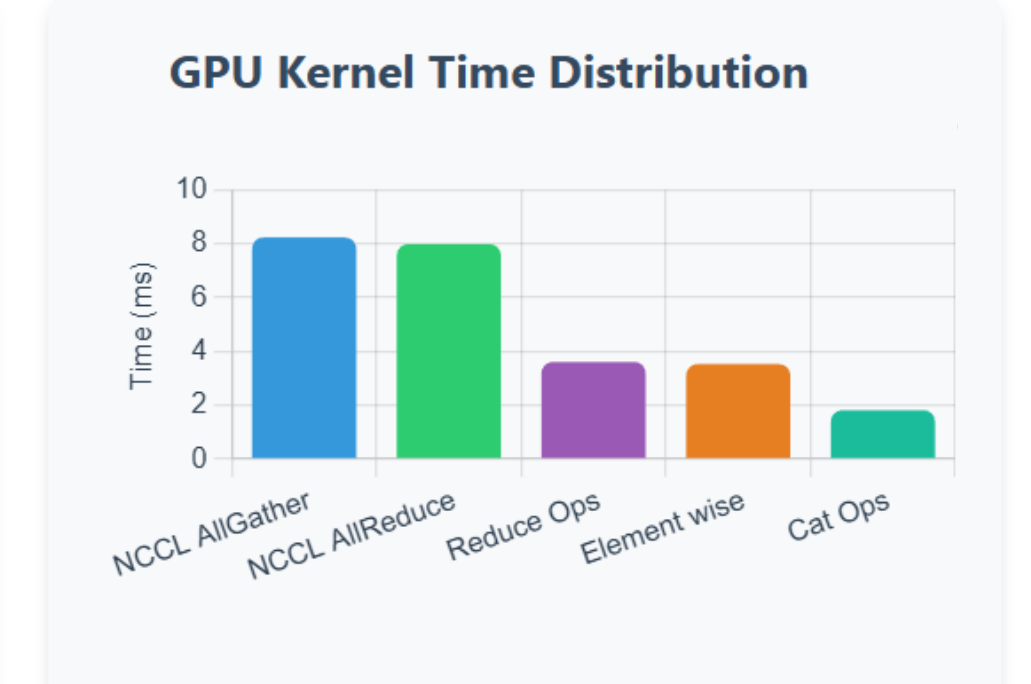
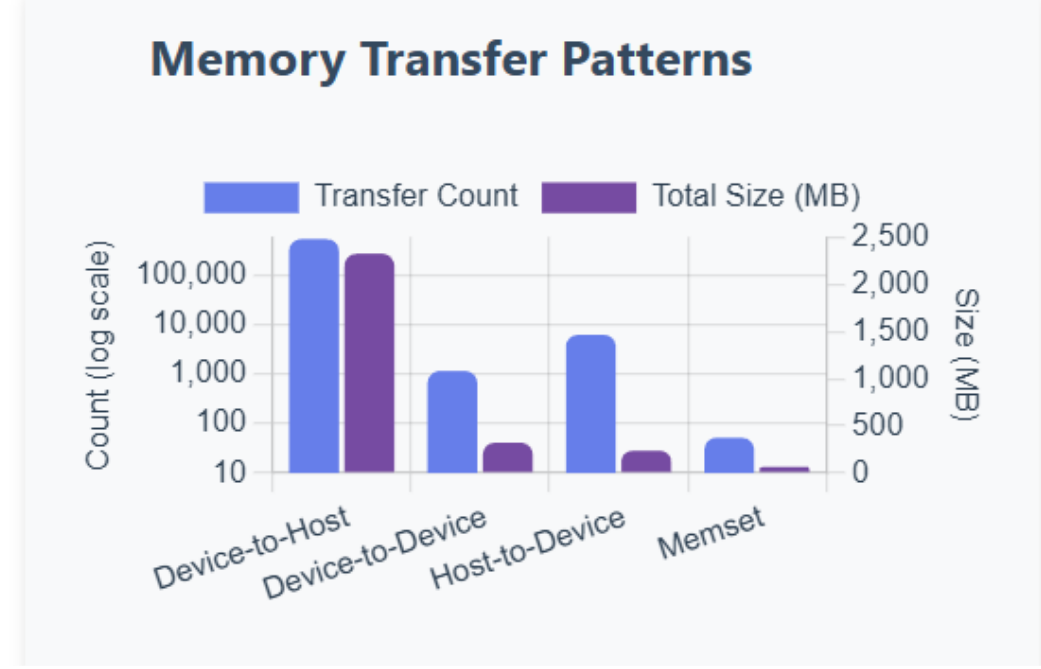
The analysis revealed four main bottlenecks. First, the overhead of Stream Synchronization is very high. cudaStreamSynchronize is called 547,393 times, taking 4.47 seconds, causing severe GPU and CPU concurrency blocking. Second, the NCCL communication shows that ncclAllGather and ncclAllReduce consume nearly 64.4% of the GPU kernel time, indicating high communication frequency. Third, the kernel is not fully computable. 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 c

Figure 3: 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.

* 1. Analysis and Optimization Opportunities

Comparing the platform baseline with Twin-B profiling reveals quantifiable optimization opportunities. The platform demonstrates 20.89 μs fixed latency per small transfer. With 6,289 H2D operations averaging 37.5 bytes, theoretical overhead is 131.4 ms for transfers alone. Combined with 547,393 cudaStreamSynchronize calls averaging 8.16 μs (4.47 seconds total), synchronization overhead dominates execution time. The platform baseline shows 37-byte transfers achieve only 0.005% of peak H2D bandwidth (1.64 MB/s vs 9,221 MB/s at 1MB), wasting 99.995% of available PCIe bandwidth.

Aggregating the 6,289 small H2D transfers (totaling 233 KB) into a single 1MB transfer would reduce latency from 131.4 ms to 0.108 ms (1,215× improvement) while increasing bandwidth utilization from 0.005% to 29% of PCIe capability. Energy cost would decrease from 7.42 J (131.4 ms × 56.5W) to 0.006 J (0.108 ms × 56.5W), a 99.5% energy reduction for the same data movement.

Similarly, reducing cudaStreamSynchronize frequency by 10× through batched execution would save 4.02 seconds of the observed 4.47 second overhead. The platform's demonstrated 6.2% speedup potential in mixed workloads suggests asynchronous stream execution could enable compute-transfer overlap. However, the Twin-B profiling shows forced synchronization prevents this optimization. Implementing CUDA streams with async memory copies and delayed synchronization could recover this performance while maintaining correctness, as theplatform baseline confirms operations can overlap effectively when synchronization constraints are relaxed.

NCCL communication shows minimal optimization potential at the operation level—individual AllGather (0.09-1.87 ms) and AllReduce (0.03-0.05 ms) operations are already fast and energy-efficient (55.7W). The 64.4% GPU kernel time consumed by NCCL stems from operation frequency (1,158 calls) rather than per-operation cost. Batching multiple gradient synchronizations or employing gradient accumulation could reduce NCCL invocation frequency while maintaining distributed training correctness.

Device-to-Device transfers achieve 64.3 GB/s bandwidth (7× faster than H2D's 9.2 GB/s), suggesting opportunities to restructure data flow. Moving preprocessing operations to GPU kernels to exploit D2D bandwidth would eliminate many of the 548,037 CPU-GPU transfers entirely. The platform's excellent compute efficiency (247 GFLOPS/Watt) makes GPU-side processing energy-favorable compared to CPU processing followed by H2D transfer overhead.

Energy efficiency improvements from these optimizations compound. Eliminating 99.5% of small transfer energy overhead (7.4 J → 0.04 J), reducing synchronization overhead by 90% (estimated 40J → 4 J based on GPU power during blocking), and maintaining NCCL efficiency would reduce data movement energy consumption by approximately 90% while improving simulation throughput proportionally. The platform baseline provides quantitative evidence that co-simulation performance is limited not by GPU compute capability or NCCL communication speed, but by inefficient small-transfer patterns and excessive forced synchronization, both of which are addressable through batching and asynchronous execution strategies.

1. Results and discussion

The analysis results show that there is a clear imbalance of CPU workload distribution in the distributed co-simulation, with the primary Python process consuming 96.99% of the CPU resources, while the secondary process consumes only 2.02% of the CPU resources. Furthermore, the thread-level analysis reveals that the primary execution thread consumes 85.68% of the CPU resources, suggesting that parallel execution may not be able to efficiently utilize the computing resources available on both GPUs.

The CUDA API profile reveals a significant synchronization bottleneck. The cudaStreamSynchronize function takes up 66.3% of the total CUDA API time, taking 4.47 seconds to execute 547,393 times, with an average of 8.16 microseconds each, indicating systematic blocking behavior rather than latency. This excessive synchronization severely limits asynchronous execution between the GPU and CPU, forcing the CPU to wait for the GPU to finish before executing the next instruction.

Memory transfer patterns reveal inefficiencies in data exchange. The cudaMemcpyAsync operation consumes 29.8% of the CUDA API bandwidth across 548,037 invocations, with an average transfer size of only 3,376 bytes. Transfer size distribution is particularly problematic, with host-to-device transfers averaging only 37.5 bytes per operation across 6,289 invocations. Such small, fragmented data transfers underutilize PCIe bandwidth and incur disproportionate overhead compared to the actual data transferred. A total of 2.95 GB of data is transferred primarily via device-to-host (2.32 GB across 540,594 invocations), indicating frequent fetching from GPU memory for host-side processing or validation.

Analysis of the GPU kernels shows that communication has the highest throughput. The combined NCCL communication (via ncclDevKernel\_AllGather and ncclDevKernel\_AllReduce) accounts for 64.4% of the GPU kernel's total processing time, with AllGather taking 32.7% and AllReduce taking 31.7%. In contrast, the actual processing time for the kernels is only about 28.3%. This time distribution indicates that the system spends approximately twice as much time synchronizing data between GPUs as it does for actual neural network computations, which is a significant limitation on the efficiency of distributed training.

The memory operation timeline confirms this inefficiency, with device-to-host transfers accounting for 98.8% of the total memory operation time. Despite this small size, the massive volume of these transfers (540,594) creates a significant cumulative overhead.

The NVTX mark provides some insight into high-level execution times, showing that NCCL initialization takes a considerable amount of time, totaling 290.6 milliseconds, exhibiting highly variable timings. The ncclGroupEnd operation takes 281 nanoseconds to 134.9 milliseconds, and the ncclAllGather operation takes 32.3 microseconds to 89.8 milliseconds. The widespread causes some AllGather operations to be delayed from synchronization, while most are executed efficiently.

Analysis of the operating system runtime reveals severe host-side blocking, a fundamental limitation of GPU utilization. The polling system call takes 2,363 seconds, but with high variability (standard deviation 135.6 ms). Primitive thread synchronization is extremely time-consuming, with pthread\_cond\_timedwait taking a cumulative 1,233 seconds and pthread\_cond\_wait taking 985 seconds. The epoll\_wait operation takes a total of 999 seconds, while select and sem\_clockwait take approximately 253 seconds. These long blocking times indicate that CPU threads are spending their time waiting, rather than doing productive work, creating bottlenecks.

Performance bottlenecks jointly limit the throughput of the co-simulator: First, excessive stream synchronization creates pseudo-serialization points that hinder concurrency. Second, the NCCL co-communication operation takes up the most GPU processing time at 64.4%, indicating that distributed training applications prioritize data synchronization over computational efficiency. Third, the compute kernel is underutilized; the actual computation kernel takes up only 28.3% of the total GPU kernel. Fourth, the distributed memory transfer scheme has 548,037 asynchronous copies, but their average size is only 3,376 bytes (and H2D averages only 37.5 bytes), which introduces significant DMA overhead and CPU cache issues. Fifth, host-side blocking from waiting a total of 6.3 seconds in various synchronization primitives, fundamentally limiting the rate at which workloads are sent to the GPU.

Analysis indicates that the system operates as a CPU-bound workload despite abundant GPU resources, with the GPUs being left idle due to host-side waiting patterns and synchronization requirements, frequent small data transfers, and large imbalances in CPU utilization. These shortcomings suggest that distributed training frameworks cannot efficiently utilize available CPU resources, and there are opportunities for improvement by employing batch execution, overlaying communication with the computation, and reducing the frequency of the CPU-GPU coordination points.

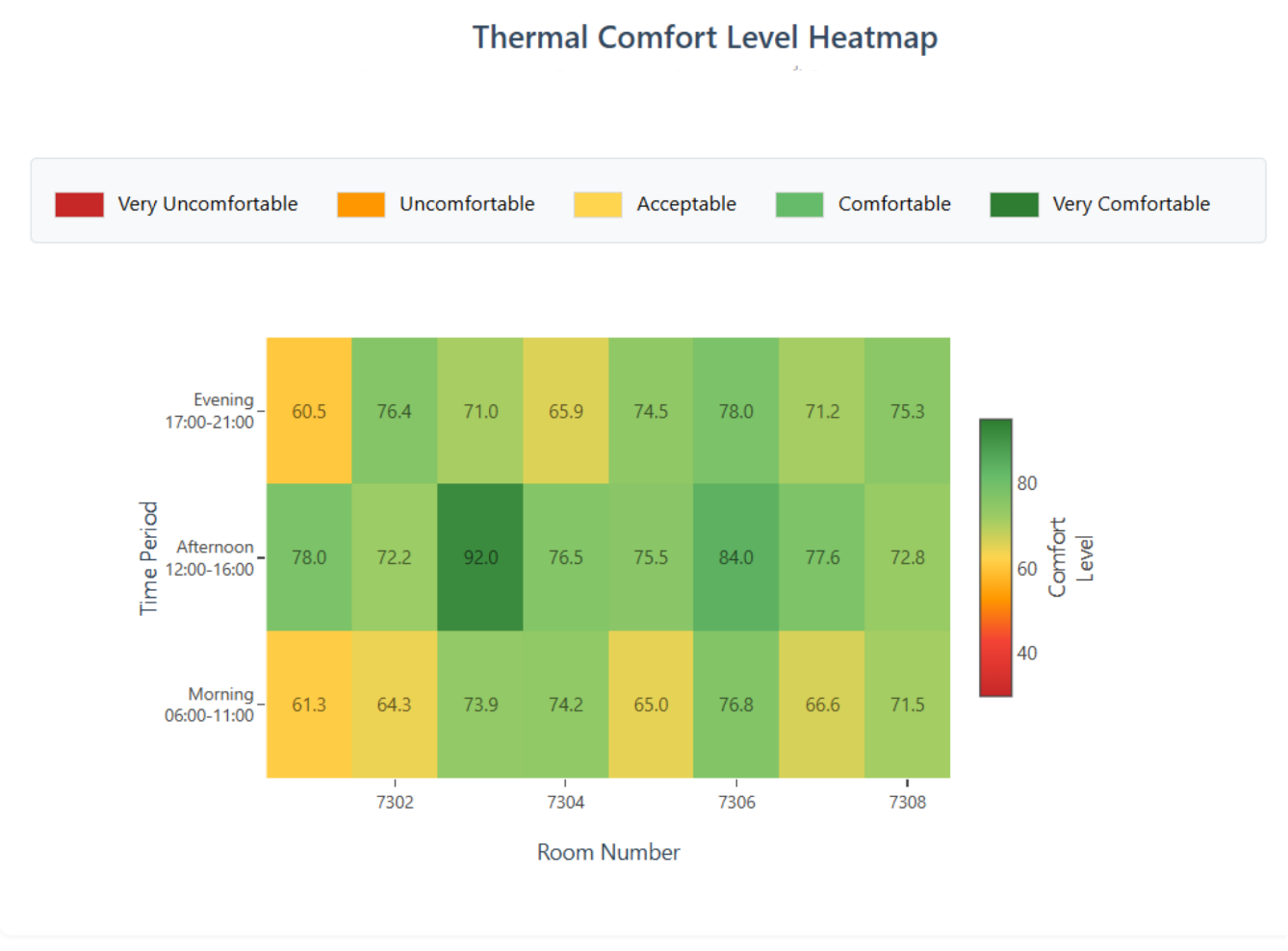


Figure 4: heatmap of thermal comfort levels obtained from the co-simulation run

The Twin-B model thus paves the way for utilizing HPC profiling to analyze simulation stages that may create bottlenecks or lead to excessive energy consumption. This platform is designed as a prototype to study the efficiency of data exchange and temporal synchronization between differently structured simulators, providing a foundation for future performance optimization strategies, such as reducing energy consumption during data exchange and balancing workloads among subsystems. The high energy usage of co-simulation has environmental impacts, especially when thousands of runs are needed for optimal policy or sensitivity analyses, highlighting the trade-off between model accuracy and computational carbon footprint. This research provides a crucial direction for designing energy-efficient simulation and data management systems, aligning with the modern concept of sustainable computing and laying the groundwork for sustainable computing in future building energy simulations.

Performance analysis of distributed co-simulation systems is limited by CPU, even with sufficient GPU resources. This is due to three issues. First, excessive synchronization is evident: 547,393 cudaStreamSynchronize calls, representing 66.3% of the total CUDA API time, are interrupted, forcing the CPU to wait for the GPU. Second, data communication operations such as NCCL, AllGather, and AllReduce consume 64.4% of the GPU kernel time, while the execution kernel consumes only 28.3%. This results in more than double the system's data synchronization time. Third, operating system runtime analysis shows that CPU threads wait 6.3 seconds between poll, pthread\_cond\_wait, and epoll\_wait operations, further limiting GPU utilization.

To address this challenge, system architectures reduce synchronization frequency and integrate batch operations. Overlaying communication with processing, increasing the transfer buffer size to minimize data transfer events, balancing the load between CPU processes, and implementing more asynchronous operations. To enable continuous GPU utilization and improve the overall performance of the distributed co-simulation system.

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. & Corgnati, 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.comhttp://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).

1. Patarasuk, P. & Yuan, X. Bandwidth Optimal All-reduce Algorithms for Clusters of Workstations. [↑](#footnote-ref-1)
2. NVIDIA Nsight Systems user guide. (2023). [↑](#footnote-ref-2)