
Supplementary Material for “SustainLC: Benchmark for End-to-End Control of Liquid Cooling in Sustainable Data Centers”

Avishek Naug^{1†}, Antonio Guillen^{1†}, Vineet Kumar², Scott Greenwood², Wesley Brewer²,
Sahand Ghorbanpour¹, Ashwin Ramesh Babu¹, Vineet Gundecha¹,
Ricardo Luna Gutierrez¹, Soumyendu Sarkar^{1†*}

¹Hewlett Packard Enterprise, ²Oak Ridge National Laboratory

{avisek.naug, antonio.guillen, sahand.ghorbanpour, ashwin.ramesh-babu,
vineet.gundecha, rluna, soumyendu.sarkar}@hpe.com
{kumarv, greenwoodms, brewerwh}@ornl.gov

*Corresponding author. †These authors contributed equally.

1	Contents	
2	A GitHub and Documentation	3
3	B Installation	3
4	B.1 System Requirements	3
5	B.2 Installation Procedure	3
6	B.2.1 Repository Acquisiton Procedure	3
7	B.2.2 Virtual Environment Configuration	3
8	B.2.3 Dependency Installation	3
9	B.3 Dataset Preparation	3
10	C Advanced AutoCSM usage for model building	3
11	C.1 Software Installation	4
12	C.2 Repository Installation	4
13	C.3 Custom Sustain-LC models using AutoCSM	6
14	D Details of Data Center Liquid Cooling Models	6
15	D.1 Blade Group and CDU Modeling	6
16	D.2 Cooling Tower Model	6
17	D.2.1 YorkCalc Empirical Model for Outlet Water Temperature	7
18	D.2.2 Mass Balances	8
19	D.2.3 Energy Balances	8
20	D.2.4 Determining Evaporation Rate and Outlet Air State	9
21	D.2.5 Fan Power (from CoolingTower.mo)	9
22	E Sustain-LC environment implementation	9
23	F Sustain-LC Training Scripts Documentation	11
24	G Evaluation	12
25	H Policy Distillation using Decision Trees	12
26	I Additional Results on LLMs for Sustain-LC applications	12
27	I.1 Detailed Explanation (Llama-3.1-8B-Instruct)	12
28	I.2 Very Detailed Explanation (Llama 3.1-8B-Instruct)	13
29	I.3 Qwen 3-8B Short Explanation	15
30	I.4 Qwen 3-8B Detailed Explanation	15
31	I.4.1 Qwen 3-8B Very Detailed Explanation	16
32	I.5 Comparing performance of different LLMs as controllers	18

33 **A GitHub and Documentation**

- 34 • Repository:

35 `https://github.com/HewlettPackard/sustain-lc`

- 36 • Documentation Page:

37 `https://hewlettpackard.github.io/sustain-lc/`

38 **B Installation**

39 This section provides detailed instructions for setting up the SustainLC environment.

40 **B.1 System Requirements**

41 The implementation is compatible with various operating systems. All code and dependency installa-
42 tions were tested on macOS 15.4.1 and Ubuntu 22.04. Windows is also supported. The following
43 prerequisites are necessary:

- 44 • Python 3.10 or higher
- 45 • Git version control system
- 46 • Command-line interface: Unix-compatible shell (bash, zsh) or PowerShell on Windows

47 **B.2 Installation Procedure**

48 **B.2.1 Repository Acquisiton Procedure**

49 The codebase must be obtained via the following commands:

```
50 git clone https://github.com/HewlettPackard/sustain-lc.git  
51 cd sustain-lc  
52
```

53 **B.2.2 Virtual Environment Configuration**

54 For Unix-based or Windows systems:

```
55 conda env create -f environment.yml  
56 conda activate sustain-lc  
57
```

58 **B.2.3 Dependency Installation**

59 Any further dependency installation is not required, since the environment.yml will install everything.

60 **B.3 Dataset Preparation**

61 The implementation utilizes the Oakridge’s cold-day heat and weather data set². Researchers should
62 download the dataset from the repository and place the processed file in the following default location:

63 `data/input_04-07-24.csv`

64 **C Advanced AutoCSM usage for model building**

65 This section covers advanced topics like building custom models for sustain-lc. This requires the user
66 to have the following repositories and software installations

²Available at: `https://code.ornl.gov/exadigit/datacenterCoolingModel/-/raw/main/python/data/input_04-07-24.csv?ref_type=heads`

67 C.1 Software Installation

68 Dymola and OpenModelica both provide a GUI and a command-line interface (CLI) for creating, com-
69 piling, running Modelica model simulations as well as exporting them to binaries called Functional
70 Mockup Units (FMUs).

71 C.2 Repository Installation

72 Users need to clone the following repositories to their working folder that can be accessed by either
73 Dymola or the OpenModelica IDEs

74 1. **Modelica Buildings library:** `git clone https://github.com/lbl-srg/modelica-buildings.git`

75 The Modelica Buildings Library is a free, open-source library for modeling building energy and
76 control systems, developed by Lawrence Berkeley National Laboratory. It provides comprehensive
77 component models for HVAC systems, including heat exchangers, pumps, and valves essential
78 for liquid cooling applications. The library enables dynamic simulation of thermal systems with
79 fluid flow, heat transfer, and controls integration for performance analysis and optimization. Its
80 modular architecture allows users to construct complex cooling systems by connecting components
81 through standardized interfaces that preserve energy and mass balance. The library's extensive
82 validation against measured data makes it suitable for accurately simulating liquid cooling systems
83 in buildings and data centers.

84 2. **TRANSFORM:** `git clone https://github.com/ORNL-Modelica/TRANSFORM-Library.git`

85 The TRANSFORM (TRANSient Simulation Framework Of Reconfigurable Models) Library is an
86 open-source Modelica toolkit developed by Oak Ridge National Laboratory for modeling complex
87 thermal-hydraulic systems. It specializes in advanced energy systems with particular strength in
88 liquid-cooled applications, including advanced reactor designs and heat transfer loops. The library
89 provides detailed component models for heat exchangers, pumps, compressors, and specialized
90 fluid systems with comprehensive thermophysical property implementations. TRANSFORM
91 excels at simulating transient behaviors in cooling systems, making it valuable for studying system
92 responses during operational changes or upset conditions. The modular architecture enables
93 scaling from component-level to system-level simulations with various working fluids, including
94 specialized coolants used in high-performance liquid cooling applications.

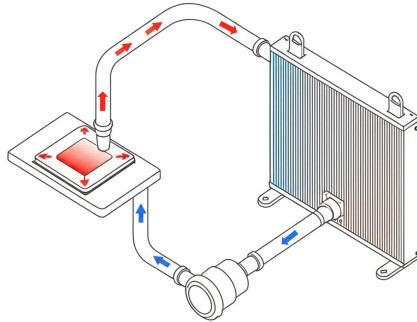


Figure 1: TRANSFORM for modeling thermal-hydraulic systems

95 3. **datacenterCoolingModel:** `git clone https://code.ornl.gov/exadigit/datacenterCoolingModel.git`

96 The Data Center Cooling Model is an ORNL-developed specialized simulation framework target-
97 ing liquid cooling systems specifically for high-performance computing facilities. The repository
98 provides detailed modeling capabilities for direct-to-chip, immersion, and rear-door heat exchanger
99 liquid cooling technologies increasingly adopted in modern data centers. Its component models
100 account for the complex interactions between IT equipment heat generation, coolant flow distribu-
101 tion, and thermal management systems at rack, row, and facility scales. The framework enables
102 performance assessment, optimization, and efficiency analysis of cooling systems under various
103 operating conditions and workloads. The models support integration with power consumption
104 data to enable comprehensive energy efficiency calculations and cooling infrastructure planning
105 for data centers.



Figure 2: ExaDigiT supercomputing consortium

106 4. **AutoCSM**: `git clone https://code.ornl.gov/exadigit/AutoCSM.git`
 107 ExaDigiT AutoCSM is a template system-of-systems modeling approach for automating the devel-
 108 opment, deployment, and integration of **Cooling System Models (CSMs)** for supercomputing
 109 facilities within the **ExaDigiT framework**.
 110
 111 *ExaDigiT is a digital twin of supercomputers and their thermal infrastructures. It offers insights*
 112 *into operational strategies, “what-if” scenarios, as well as elucidates complex, cross-disciplinary*
 113 *transient behaviors. It also serves as a design tool for future system prototyping. It combines*
 114 *telemetry and simulations, providing a virtual representation of physical systems. It supports*
 115 *planning, construction, and operations, offering value in decision-making, predictive maintenance,*
 116 *and system efficiency. In design stages, it can evaluate energy efficiency, virtually prototype*
 117 *cooling systems, and model network performance. During operations, ExaDigiT aids in predictive*
 118 *maintenance and operational optimization.*
 119 *ExaDigiT is built on an open software stack (Modelica, SST Macro, Unreal Engine) with an*
 120 *aim to foster community-driven development, we have formed a **partnership with national su-***
 121 ***percomputer centers (Oak Ridge National Laboratories, Lawrence Livermore National Labs,***
 122 ***Los Alamos National Labs (USA), PAWSEY (Australia), LUMI (Finland), CINES (France),***
 123 ***CINECA (Italy), etc) around the world to develop an open framework for modeling supercomput-***
 124 ***ers.***
 125 AutoCSM is a Python-based framework to assist in CSM developers in accelerating the creation
 126 and deployment of system-level thermal-hydraulic CSMs. The intention is for this tool specifi-
 127 cally to help standardize digital twin workflows for ExaDigiT. However, this tool can be used
 128 independently of ExaDigiT (and even other systems besides CSMs).

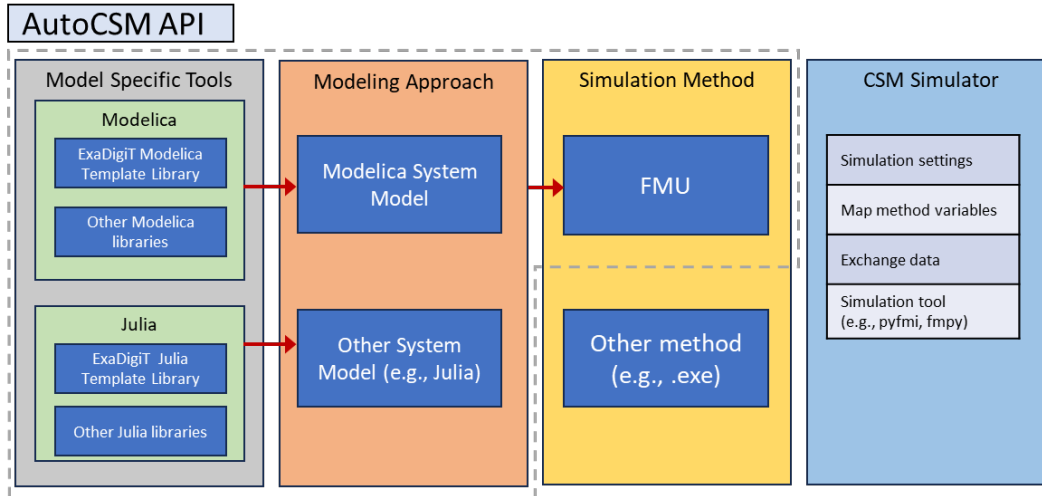


Figure 3: AutoCSM API in the broader ExaDigiT procedure.

129 Of these libraries, the user needs to access the **datacenterCoolingModel** to study the atomic structures
 130 of the thermodynamic components that can be used to build custom data center configurations. An
 131 example configuration is provided in Example JSON. This JSON describes an example hierarchical
 132 structure for the models. Further example hierarchical structures used for the results in the main
 133 paper are also included in the sustain-1c repository.

C.3 Custom Sustain-LC models using AutoCSM

The primary model building process based on the specified structure is executed by the **AutoCSM** API library. It reads the JSON file and then populates a Modelica file using elements from the `datacenterCoolingModel` library.

To execute this process, we simply run the

```
"python run_auto_csm.py"
```

from the CLI in which the JSON file and the Python files are located in the AutoCSM library. The user needs to specify the path to the desired JSON file inside the `run_auto_csm.py` file as well as compilation parameters like solver information, steps to solve etc.

The above process generates the FMU which is then wrapped inside a Gymnasium Environment for Sustain-LC. Most of the common application requirements are already covered by the default Sustain-LC environment file `frontier_env.py`. If the user wishes to specify highly custom variables for logging, they have to specify those variables in the info dictionary for the environment.

D Details of Data Center Liquid Cooling Models

D.1 Blade Group and CDU Modeling

The thermodynamics of the blade-groups (BGs) to which the heat is added and then removed via liquid cooled plates is governed by the two equations differential equations: the heat capacitor and the conduction model. The heat capacitor C represents the server's thermal storage capacity,

$$C \cdot \frac{dT}{dt} = Q_{port}(t)$$

where T is the BG temperature and Q_{port} is the net heat flow. Q_{port} is calculated based on the heat generated by the BG based on its load heat generation rate P_{branch} .

$$Q_{port} = P_{branch}$$

The conduction model used to transfer heat between the server plate and the cooling liquid is represented by:

$$Q_{flow} = G_c \cdot (T_{solid} - T_{fluid})$$

where G_c is the effective convection thermal conductance, dependent on coolant properties and flow rate (m_{flow}) and the temperature of the incoming fluid (T_{fluid}). The overall heat transfer problem is described by

$$\Phi(Q_{port}) + Q_{flow} = C \cdot \frac{dT_{server}}{dt}$$

Server load heat generation rate P_{branch} is an exogenous temporal variable determining heat generation at different temperatures (T). An example trace of this heat load generation is shown in `input_04-07-24.csv`. The primary optimization goal is to minimize energy consumption by controlling the cooling liquid flow rate m_{flow} and its temperature entering the servers T_{fluid} . Φ is a polynomial function that helps us experimentally tune the heat generation behavior under server workloads to demonstrate the ability of RL algorithms to work with non-linear heat load generation. The quadratic term is usually adjusted to 0.015 with the linear term set at 1. The goal is to make the problem hard for RL and heuristic controllers compared to linear differential equations for heat generation.

D.2 Cooling Tower Model

A cooling tower cools a stream of hot water by bringing it into contact with a stream of air. The primary cooling mechanisms are:

1. **Evaporative Cooling (Latent Heat Transfer):** A small portion of the hot water evaporates into the air stream. This phase change from liquid to vapor requires a significant amount of energy (latent heat of vaporization), which is drawn from the remaining bulk water, thus cooling it. This is usually the dominant cooling effect.

174 **2. Sensible Heat Transfer (Convection):** If the air is cooler than the water, there will be direct heat
 175 transfer from the water to the air due to the temperature difference. This effect is typically smaller
 176 than evaporative cooling.

177 The coolest temperature the water can theoretically reach is the **wet-bulb temperature** of the
 178 incoming air. The difference between the actual outlet cold water temperature and the air's wet-bulb
 179 temperature is called the **approach**.

180 In the Modelica models used in this work, we have three main files YorkCalc.mo for Base Em-
 181 pirical Correlation, coolingTower_Towb.mo for Detailed Physics and Mass/Energy Balances and
 182 CoolingTower.mo for System Integration and Fan Model.

183 The physics-based equations are primarily from coolingTower_Towb.mo and concepts in
 184 YorkCalc.mo). The variables used for the modeling are listed below for convenience first:

185 • T_{w_in} : Inlet water temperature (°C or K)	199 • X_{a_out} : Outlet air humidity ratio ($kg_{water_vapor} /$ 100 kg_{dry_air})
186 • T_{w_out} : Outlet water temperature (°C or K)	201 • \dot{m}_a : Dry air mass flow rate (kg_{dry_air} / s)
187 • \dot{m}_{w_in} : Inlet water mass flow rate (kg/s)	202 • h_{a_in} : Enthalpy of moist inlet air (J / kg_{dry_air})
188 • \dot{m}_{w_out} : Outlet water mass flow rate (kg/s)	203 • h_{a_out} : Enthalpy of moist outlet air (J / kg_{dry_air})
189 • \dot{m}_{w_evap} : Water evaporation rate (kg/s)	204 • U_w : Internal energy of water in the tower (J)
190 • c_{p_w} : Specific heat capacity of water (J/kg.K)	205 • U_a : Internal energy of air in the tower (J)
191 • h_{fg} : Latent heat of vaporization of water (J/kg)	206 • m_{w_sump} : Mass of water in the tower sump (kg)
192 • T_{a_in} : Inlet air dry-bulb temperature (°C or K)	207 • m_{a_vol} : Mass of dry air in the tower volume (kg)
193 • T_{a_out} : Outlet air dry-bulb temperature (°C or K)	208 • m_{v_vol} : Mass of water vapor in the tower volume 194 (kg)
194 • T_{wb_in} : Inlet air wet-bulb temperature (°C or K)	209 • Q_{tot} : Total heat rejected by water (W)
195 • ϕ_{a_in} : Inlet air relative humidity (-)	210 • Q_{sen} : Sensible heat transfer from water to air (W)
196 • ϕ_{a_out} : Outlet air relative humidity (-)	211 • Q_{lat} : Latent heat transfer due to evaporation (W)
197 • X_{a_in} : Inlet air humidity ratio ($kg_{water_vapor} /$ 100 kg_{dry_air})	212 • P_{fan} : Fan power consumption (W)
198	213

214 D.2.1 YorkCalc Empirical Model for Outlet Water Temperature

215 The core of predicting the cooling tower's performance in this model suite comes from an empirical
 216 correlation. The outlet water temperature (T_{w_out}) is determined by adding an "approach temperature"
 217 (ΔT_{app}) to the inlet air's wet-bulb temperature (T_{wb_in}).

$$T_{w,out} = T_{wb,in} + \Delta T_{app} \quad (1)$$

218 The approach temperature ΔT_{app} itself is calculated using a polynomial function that depends on:

- 219 • T_{wb_in} : Inlet air wet-bulb temperature.
- 220 • $R_F = \frac{\dot{m}_w}{\dot{m}_a}$: Flow ratio (water flow rate to air flow rate).
- 221 • $R_{F,nom}$: Nominal (design) flow ratio.
- 222 • A_R : Approach ratio at nominal conditions (a design parameter).
- 223 • $Q_{ratio} = \frac{Q_{actual}}{Q_{nominal}}$: Ratio of actual heat rejection to nominal heat rejection.

224 The specific polynomial form in YorkCalc.mo for ΔT_{app} is:

$$\begin{aligned} \Delta T_{app} = & A_R \cdot (c_1 + c_2 T_{wb,in} + c_3 T_{wb,in}^2 + c_4 R_F + c_5 R_F^2 \\ & + c_6 R_F T_{wb,in} + c_7 Q_{ratio} + c_8 Q_{ratio}^2 + c_9 Q_{ratio} T_{wb,in} + c_{10} Q_{ratio} R_F) \end{aligned} \quad (2)$$

225 where c_1, \dots, c_{10} are empirical coefficients. (Note: The Modelica code simplifies this based on
 226 *useQRatio* flag. The most general form includes Q_{ratio} . For simplicity, we have presented a common

227 structure; the exact terms might vary slightly based on the if conditions in the Modelica code, but the
 228 principle is a polynomial fit).

229 D.2.2 Mass Balances

230 These equations describe how the amount of water and air (and vapor in air) changes over time within
 231 the control volume of the cooling tower. $\text{der}(X)$ means dX/dt .

232 The change in water mass in the tower sump is the inlet water flow minus the outlet water flow and
 233 minus the evaporated water.

$$\frac{d(m_{a,vol})}{dt} = \dot{m}_{a,in} - \dot{m}_{a,out} \quad (3)$$

234 In steady-state, $\dot{m}_{w,out} = \dot{m}_{w,in} - \dot{m}_{w,evap}$. The outlet water flow is less than the inlet due to
 235 evaporation.

236 Assuming dry air mass flow rate is controlled and constant through the tower.

$$\frac{d(m_{a,vol})}{dt} = \dot{m}_{a,in} - \dot{m}_{a,out} \quad (4)$$

237 In steady-state, $\dot{m}_{a,in} = \dot{m}_{a,out} = \dot{m}_a$

238 The change in water vapor mass in the air inside the tower depends on vapor entering with inlet air,
 239 vapor leaving with outlet air, and water evaporating into the air.

$$\frac{d(m_{v,vol})}{dt} = \dot{m}_a X_{a,in} - \dot{m}_a X_{a,out} + \dot{m}_{w,evap} \quad (5)$$

240 In steady-state, $\dot{m}_{w,evap} = \dot{m}_a (X_{a,out} - X_{a,in})$. This directly links the evaporation rate to the
 241 change in air humidity.

242 D.2.3 Energy Balances

243 The following equations describe how the energy content of the water and air changes. The change in
 244 internal energy of the water in the tower is due to enthalpy flow in, enthalpy flow out, convective heat
 245 transfer to the air, and energy lost due to evaporation (where evaporated water carries away enthalpy
 246 $h_{w,evap}$, often approximated as enthalpy of saturated liquid at $T_{w,out}$).

$$\frac{d(U_w)}{dt} = \dot{m}_{w,in} h_{w,in} - \dot{m}_{w,out} h_{w,out} - Q_{sen} - \dot{m}_{w,evap} h_{w,evap} \quad (6)$$

247 Where $h_w = c_{p,w} T_w$ (approximately, if using a reference temperature of 0°C for enthalpy). The term
 248 Q_{sen} represents sensible heat transfer. The term $\dot{m}_{w,evap} h_{w,evap}$ is closely related to Q_{lat} .

249 The change in internal energy of the air in the tower is due to enthalpy flow in, enthalpy flow out,
 250 convective heat received from water, and enthalpy gained from evaporated water vapor.

$$\frac{d(U_a)}{dt} = \dot{m}_a h_{a,in} - \dot{m}_a h_{a,out} + Q_{sen} + \dot{m}_{w,evap} h_{v,evap} \quad (7)$$

251 Where h_a is the enthalpy of moist air (J/kg dry air), and $h_{v,evap}$ is the enthalpy of water vapor at
 252 the evaporation temperature (often taken as $T_{w,out}$). Note that $h_a = c_{p,da} T_a + X_a (h_{fg,0} + c_{p,v} T_a)$
 253 where $h_{fg,0}$ is latent heat at 0°C, and $c_{p,da}$ and $c_{p,v}$ are specific heats of dry air and vapor.

254 Finally, for the heat transfer calculation, we start with the total heat rejected by water (steady state).
 255 This is the primary quantity of interest from the water side.

$$Q_{tot} = \dot{m}_{w,in} c_{p,w} (T_{w,in} - T_{w,out}) \quad (8)$$

256 The $T_{w,out}$ here is the one determined by the YorkCalc model.

257 Relating Total Heat to Air Side (Steady State): This total heat rejected by the water is transferred to
 258 the air as a combination of sensible and latent heat.

$$Q_{tot} = \dot{m}_a (h_{a,out} - h_{a,in}) \quad (9)$$

259 This equation must hold, and it's used to find the $h_{a,out}$ (and thus $T_{a,out}$ and $X_{a,out}$)

260 Latent Heat Transfer:

$$Q_{lat} = \dot{m}_{w,evap} h_{fg} \quad (10)$$

261 where h_{fg} is the latent heat of vaporization, typically evaluated at the average water temperature or
262 $T_{w,out}$.

263 Sensible Heat Transfer: The model doesn't explicitly calculate Q_{sen} using a heat transfer coefficient
264 and LMTD (Log Mean Temperature Difference) typical in detailed heat exchanger models. Instead,
265 after $T_{w,out}$ is found from YorkCalc, and Q_{tot} is known, the model solves for $\dot{m}_{w,evap}$ and the outlet
266 air state ($T_{a,out}$, $X_{a,out}$) such that the air-side energy and mass (vapor) balances are satisfied. The
267 Q_{sen} is implicitly:

$$Q_{sen} = Q_{tot} - Q_{lat} \quad (11)$$

268 Or, from the air side:

$$Q_{sen} \approx \dot{m}_a c_{p,moist_air} (T_{a,out} - T_{a,in}) \quad (12)$$

269 The Modelica code calculates Q_{conv} , which is the sensible heat transfer. It is derived from the overall
270 energy balance once $m_{flow_w_evap}$ is determined.

271 D.2.4 Determining Evaporation Rate and Outlet Air State

272 Once $T_{w,out}$ is known from YorkCalc, and thus Q_{tot} is known:

- 273 • The model needs to find $\dot{m}_{w,evap}$, $T_{a,out}$, and $X_{a,out}$ (or $\phi_{a,out}$).
- 274 • This is an iterative process or a simultaneous solution. The key constraints are:
 - 275 – $\dot{m}_{w,evap} = \dot{m}_a (X_{a,out} - X_{a,in})$ (Vapor mass balance)
 - 276 – $Q_{tot} = \dot{m}_a (h_{a,out}(T_{a,out}, X_{a,out}) - h_{a,in}(T_{a,in}, X_{a,in}))$ (Air energy balance)
 - 277 – Psychrometric relations: $h_{a,out}$ depends on $T_{a,out}$ and $X_{a,out}$. Also, $X_{a,out}$ is related
278 to $\phi_{a,out}$ and $T_{a,out}$ via saturation pressure.
 - 279 – A common assumption is that the outlet air is saturated or near-saturated at $T_{a,out}$ if
280 the tower is efficient, but the model calculates ϕ_{out_a} based on X_{out_a} and T_{out_a} .

281 The Modelica code uses `Medium.temperature_phX` and similar functions to solve for
282 these outlet air properties based on its calculated enthalpy and humidity ratio. The
283 `evaporationAndCondensation` record within `coolingTower_Towb.mo` attempts to model this
284 more fundamentally, calculating m_{flow_Sender} , which is $m_{flow_w_evap}$.

285 D.2.5 Fan Power (from CoolingTower.mo)

286 The `CoolingTower.mo` model incorporates a fan model. Fan power P_{fan} is typically calculated
287 based on the air volume flow rate \dot{V}_a and the pressure rise Δp_{fan} provided by the fan, and fan
288 efficiency η_{fan} :

$$\dot{V}_a = \frac{\dot{m}_a}{\rho_a} \quad (\text{where } \rho_a \text{ is air density}) \quad (13)$$

$$P_{fan} = \frac{\dot{V}_a \Delta p_{fan}}{\eta_{fan}} \quad (14)$$

289 The fan model `SpeedControlled_y` in `Buildings` library uses performance curves (polynomials) to
290 relate flow rate, pressure rise, speed, and power, often normalized by nominal values. For example:

$$P_{fan} = P_{fan,nom} \cdot f(N/N_{nom}, \dot{V}_a/\dot{V}_{a,nom}) \quad (15)$$

291 where N is the fan speed.

292 E Sustain-LC environment implementation

293 The environment interfaces with a high-fidelity Modelica model compiled as a Functional Mock-up
294 Unit (FMU) version 2.0 for Co-Simulation, leveraging the PyFMI library for interaction.

295 FMU Integration and Simulation Core

296 The core of the simulation is a Modelica model representing the data center's liquid cooling thermody-
297 namics. This model is compiled into an FMU, for example `LC_Frontier_5Cabinet_4_17_25.fmu`.
298 The environment utilizes PyFMI to:

- 299 1. Load the FMU and parse its model description.
- 300 2. Instantiate the FMU for simulation.
- 301 3. Set up the experiment parameters, including start time (0.0) and a tolerance (if specified, default is
302 FMU's choice).
- 303 4. Initialize the FMU into its starting state.
- 304 5. During an episode step:
 - 305 • Set input values (actions from the RL agent) to specified FMU variables.
 - 306 • Advance the simulation time by `sim_time_step` using the `fmu.do_step()` method. This is
307 repeated until the agent's `step_size` is covered.
 - 308 • Get output values (observations for the RL agent and values for reward calculation) from
309 specified FMU variables.
- 310 6. Terminate and free the FMU instance upon closing the environment or resetting for a new episode.

311 The FMU variable names used for interfacing are explicitly defined within the environment:

- 312 • **Action Variables:** `self.fmu_action_vars` (e.g., `pump1.speed_in`, `valve1.position_in`)
- 313 • **Observation Variables:** `self.fmu_observation_vars` (e.g., `serverRack1.T_out`, `ambient.T`)
- 314 • **Power Consumption Variables (for reward):** `self.fmu_power_vars` (e.g., `[pump1.P, fan1.P]`)
- 315 • **Target Temperature Variable (for reward):** `self.fmu_target_temp_var` (e.g.,
316 `controller.T_setpoint`)

317 State (Observation) Space

318 The observation space is defined as a continuous `gymnasium.spaces.Box` with specific lower and
319 upper bounds. It comprises the following variables retrieved from the FMU:

- 320 • `FrontierNode.AvgBladeGroupTemp`: Average temperature of a Blade Group in a cabinet (K).
- 321 • `FrontierNode.AvgBladeGroupPower`: Average power input to each Blade Group in a cabinet
322 (W).

323 The bounds for these observations are set to e.g., 273.15 K and e.g., 373.15 K for tempera-
324 ture measurements, and between e.g., 0.0 kW and e.g., 400 kW, for power input measurements.

325 For the Cooling Tower Markov Decision Process, we have a similar observation space:

- 326 • `FrontierNode.CoolingTower.CellPower`: Average power consumption of each cell of the
327 cooling tower (W).
- 328 • `FrontierNode.CoolingTower.WaterLeavingTemp`: Average temperature of the water leaving
329 each cooling tower (K).
- 330 • `T_owb`: Outside air wetbulb temperature.

331 Action Space

332 The action space is a hybrid of continuous `gymnasium.spaces.Box` and discrete
333 `gymnasium.spaces.Discrete`, allowing the agent to control:

- 334 • `FrontierNode.CDU.Pump.normalized_speed`: Scaled speed of the CDU pump (-1 to 1).
- 335 • `FrontierNode.CDU.TempSetpoint`: Scaled Coolant supply temperature setpoint (-1 to 1).

- `FrontierNode.CDU.AvgBladeGroupValve`: Scaled Valve opening to allow coolant to collect heat from the corresponding blade group (-1 to 1).
- `FrontierNode.CoolingTower.WaterLvTSPT`: Discrete setting of cooling tower water leaving temperature setpoint delta.

These scaled values allow the neural network models used for the RL agents to learn properly and not saturate at the activation layers.

Reward Function

The reward function guides the RL agent towards desired operational states. It is calculated at each step as:

$$R_{blade} = - \sum_{i,j} T_{i,j} \quad (16)$$

which is the negative of the aggregate temperature of the blade groups

$$R_{coolingtower} = - \sum_{i,j} P_{i,j} \quad (17)$$

which is the negative of the total cooling tower power consumption at each time step Where:

- $T_{i,j}$ is the temperature of the j^{th} blade group of the i^{th} cabinet $\forall j$ in $1 \dots B$ and $\forall i$ in $1 \dots C$
- $P_{i,j}$ is the power consumption of the j^{th} cell of the i^{th} cooling tower $\forall j$ in $1 \dots m$ and $\forall i$ in $1 \dots N$

The goal is to minimize server temperatures below the target and minimize energy consumption.

Episode Dynamics and Simulation Control

An episode runs for a maximum of `max_episode_duration`. The agent interacts with the environment at discrete time intervals defined by `step_size`. For each agent step, the FMU's `do_step()` method is called `step_size / sim_time_step` times. The `reset()` method terminates the current FMU instance, re-instantiates and re-initializes it, ensuring a consistent starting state for each new episode. Initial observations are drawn from the FMU after initialization.

F Sustain-LC Training Scripts Documentation

- **Script:** `train_mh_ma_ca_ppo.py`
This script is designed to train a Proximal Policy Optimization (PPO) agent in an environment that involves multiple agents and components. It can be configured to use Multi-Head (MH), Centralized Action (CA), and Multi-Agent (MA) features, allowing for flexible and expressive policy representations. CA implies there is a shared policy for multiple homogeneous agents within the specified environment.

Basic Run Command: `python train_mh_ma_ca_ppo.py`

Key Configurable Parameters

The script uses has the following relevant parameters you can modify:

- `exp-name` (*str*, *default:* `ppo_ma_ca`) Name for the experiment, used for logging.
- `seed` (*int*, *default:* `123`) Random seed for reproducibility.
- `cuda` (*flag*, *default:* `True`) Enables CUDA for GPU acceleration if available. Set `-cuda False` to force CPU.
- `env_name` (*str*, *default:* `MH_SmallFrontierModel`) Name of the environment
- `agent_type` (*str*, *default:* `MultiHead_CA_PPO`) Type of the RL Agent
- `max_training_timesteps` (*int*, *default:* `5e6`) Total budget for training.

374 `-max_ep_len (int, default: 200)` Maximum episode length.
 375 `-lr_actor (float, default: 3e-4)` Learning rate for the actor optimizer.
 376 `-lr_critic (float, default: 1e-3)` Learning rate for the critic optimizer.
 377 `-K_epochs (float, default: 50)` Epochs of training to run for each update.
 378 `-eps_clip (float, default: 0.2)` clip parameter for PPO.
 379 `-num_centralized_actions (int, default: 4)` Number of centralized actions for each
 380 environment.
 381 `-gamma (float, default: 0.80)` Discount factor for future rewards.
 382 `-gae_lambda (float, default: 0.95)` Lambda for General Advantage Estimation (GAE).
 383 `-minibatch_size (int, default: 32)` Mini-batch size for each epoch
 384 `-ent-coef (float, default: 0.01)` Entropy coefficient for exploration.
 385 `-vf-coef (float, default: 0.5)` Value function loss coefficient.
 386 `-num_agents (int, default: 2)` Specifies the number of agents in the custom environment.

- 387 • **Script:** `train_multiagent_ca_ppo.py`
 388 This script trains multiple PPO agents for a multi-agent reinforcement learning (MARL) task. Each
 389 agent has its own policy and value function, for the blade group control and the cooling tower
 390 control. It specifically employs a Centralized Action (CA) mechanism. The script is designed to
 391 work with MARL environments.

392 **Basic Run Command:** `python train_multiagent_ca_ppo.py`
 393 The key configurable parameters for this script is identical to `train_mh_ma_ca_ppo.py`

394 G Evaluation

395 We provide a more user-friendly example to evaluate the agents via Jupyter notebooks. Interested
 396 users may also simply export the notebook to a Python script and run the resulting file

397 For evaluating the centralized action policies, users may run the `evaluate_ma_ca_ppo.ipynb` and
 398 for multihead policies, they may run `evaluate_mh_ma_ca_ppo copy.ipynb`.

399 H Policy Distillation using Decision Trees

400 To distill the policies for the pretrained agents, the users may run the `policy_distillation.ipynb`
 401 notebook

402 I Additional Results on LLMs for Sustain-LC applications

403 We provide further examples of RL agent actions provided by different LLM agents. The LLMs are
 404 passed the input observations and the corresponding actions taken by the RL agent in evaluation, and
 405 then asked to explain with different levels of detail.

406 I.1 Detailed Explanation (Llama-3.1-8B-Instruct)

Detailed Explanation

EXECUTIVE SUMMARY:

The current system is focused on balancing temperature stability and energy efficiency. The cabinet valves are distributed to match the power distribution across blade groups, while the cooling tower is set to very low cooling to avoid energy wastage. However, there is room for improvement in aligning temperature setpoints with goals and considering trade-offs between stability and efficiency.

TEMPERATURE STABILITY:

- **Overall Assessment:** The overall temperature stability is rated as moderate (score: 0.6341). While the system maintains reasonable temperature levels, there are deviations from desired goals, especially in cabinets 1 and 5.
- **Cabinet Temperature Control:** The valve positions are distributed to match the power distribution across blade groups, which helps maintain consistent temperature levels across all cabinets. However,

407

this comes at the cost of increased energy usage compared to other configurations.

- **Cooling Tower Impact:** The cooling tower action is set to very low cooling to avoid excessive energy consumption. While this reduces the overall impact on temperature stability, it also limits the system's ability to compensate for large deviations from desired goals.

ACTION RATIONALE:

- **Temperature Setpoints:** The temperature setpoints do not fully align with the desired goal temperatures for each cabinet. This results in deviations from target values, especially in cabinets 1 and 5.

- **Valve Positions:** The valve positions are distributed to match the power distribution across blade groups, which helps maintain consistent temperature levels across all cabinets. However, this results in higher energy usage compared to other configurations.

- **Cooling Tower:** The cooling tower action is set to very low cooling to avoid excessive energy consumption. While this reduces the overall impact on temperature stability, it also limits the system's ability to compensate for large deviations from desired goals.

ENERGY EFFICIENCY:

- **Assessment:** The current actions have a moderate energy efficiency score of 0.6166. While the valve positions are distributed to match the power distribution across blade groups, this results in higher energy usage compared to other configurations. Additionally, the cooling tower action is set to very low cooling which reduces energy wastage but also limits the system's ability to compensate for large deviations from desired goals.

- **Trade Offs:** There is a trade-off between temperature stability and energy efficiency in the current actions. The valve positions are distributed to match the power distribution across blade groups, which helps maintain consistent temperature levels across all cabinets but comes at the cost of increased energy usage. The cooling tower action is set to very low cooling to avoid excessive energy consumption, which reduces the overall impact on temperature stability but also limits the system's ability to compensate for large deviations from desired goals.

RECOMMENDATIONS:

One potential improvement could be to adjust the temperature setpoints to better align with desired goals, especially in cabinets 1 and 5. This would help reduce deviations from target values and improve overall temperature stability. Another option could be to consider alternative valve distribution configurations that balance temperature stability and energy efficiency. By optimizing both factors simultaneously, the system can achieve better results overall.

Inference Time: 4.77 seconds

408

409 **I.2 Very Detailed Explanation (Llama 3.1-8B-Instruct)**

410 This explanation is significantly longer (approximately 1,500 words) and includes detailed scientific
411 analysis across multiple categories. For complete response, refer to Appendix. A representative
412 excerpt:

Very Detailed Explanation

EXECUTIVE SUMMARY:

The cooling system is currently maintaining a stable operating temperature across all five CDUs. This is achieved through a combination of active cooling provided by the cooling towers and passive thermal regulation via the cabinet valves. The cooling towers are operated at a very low cooling level, resulting in a slight increase in temperature compared to ideal operating conditions. The cabinet valves are adjusted to distribute cooling fluid equitably among the CDUs, balancing temperature differences while minimizing energy consumption. Overall, the system is operating within acceptable parameters and maintaining a stable thermal environment for the computational hardware.

SYSTEM STATE ANALYSIS:

- **Temperature Overview:**

The temperature distribution across the five CDUs is relatively uniform, with each cabinet maintaining a temperature within a narrow range. The highest temperatures are observed in CDU-5, which has experienced a higher workload and is currently operating at a temperature of 26.38 Celsius. The lowest temperatures...

- **Workload Distribution:**

The workload distribution across the five CDUs is relatively even, with each cabinet processing an approximately equal share of the total computational workload. The CDUs are configured to handle the workload in a balanced manner, ensuring that each...

- **Historical Context:**

The current system state is characterized by stable temperatures and power consumption levels, indicating that the system has been operating in a consistent and reliable manner over the past several months.

413

The cooling system has maintained...

- **System Goals:**

The current system goals for the cooling system are to maintain a stable operating temperature across all five CDUs, while minimizing energy consumption and maximizing computational performance. The system is currently operating within...

CABINET CONTROL ANALYSIS:

- **Temperature Setpoints:**

- **Description:** The temperature setpoints for the five CDUs are configured to maintain a uniform operating temperature across all cabinets. The setpoints are optimized to balance performance and energy efficiency, taking into account...
- **Justification:** The temperature setpoints for the five CDUs are justified by the specific thermal requirements of the computational hardware. The setpoints are configured to maintain a temperature range that is optimal for...
- **Impact Assessment:** The impact of the current temperature setpoints on system stability and performance is positive. By maintaining a uniform temperature distribution across all five CDUs, the system is able to operate...

- **Valve Distributions:**

- **Description:** The valve distributions for the five CDUs are adjusted to balance cooling fluid flow and minimize energy consumption. The valve positions are optimized to distribute cooling fluid equitably among the cabinets...
- **Power Alignment:** The valve distributions for the five CDUs are aligned with the power consumption distribution across the cabinets. The valve positions are adjusted to ensure that each cabinet receives an equitable...
- **Hydraulic Considerations:** The hydraulic considerations for the valve distributions include ensuring that the cooling fluid is distributed equitably among the cabinets, while also minimizing energy consumption...

COOLING TOWER ANALYSIS:

- **Action Details:** The current cooling tower action is 'very low cooling', indicating that the cooling towers are currently operating at a low cooling level to ensure that the system remains stable and efficient. This action is justified...
- **Thermal Dynamics:** The thermal dynamics of the cooling towers are characterized by a low cooling level, resulting in a slight increase in temperature compared to ideal operating conditions. This is justified by the specific thermal...
- **Weather Interactions:** The current weather conditions are not directly influencing the cooling tower action. The cooling towers are adjusted to maintain a uniform temperature distribution across all five CDUs, ensuring that...

ENERGY EFFICIENCY ANALYSIS:

- **Component Efficiency:**

- **Cooling Tower:** The cooling tower efficiency is relatively low, resulting in a slight increase in temperature compared to ideal operating conditions. This is justified by the specific thermal requirements...
- **Pumps And Valves:** The pump and valve efficiency is relatively high, resulting in minimal energy consumption. The valve positions are adjusted to ensure that each cabinet receives an equitable share...
- **Secondary Cooling:** The secondary cooling loop efficiency is relatively high, resulting in minimal energy consumption. The valve positions are adjusted to ensure that each cabinet receives an equitable...

- **Optimization Strategy:** The current optimization strategy involves balancing energy efficiency and system stability. The temperature setpoints are configured to maintain a uniform operating temperature across all cabinets...

- **Performance Metrics:** The performance metrics for the cooling system include temperature stability, energy efficiency, and workload distribution. The temperature stability is relatively high, with minimal fluctuations...

THERMODYNAMIC IMPLICATIONS:

- **Heat Transfer:** The heat transfer dynamics within the cooling system are characterized by the distribution of cooling fluid through the cabinet valves, ensuring that each cabinet receives an equitable share of the cooling fluid...

- **Thermal Gradients:** The thermal gradients across the five CDUs are minimal, with each cabinet

maintaining a uniform temperature distribution. The valve positions are adjusted to minimize thermal gradients, ensuring that...

- **Fluid Dynamics:** The fluid dynamics within the cooling system are characterized by the distribution of cooling fluid through the cabinet valves, ensuring that each cabinet receives an equitable share of the cooling fluid...

RECOMMENDATIONS:

- **Short Term:** The short-term recommendations for the cooling system include monitoring temperature and workload distributions, adjusting valve positions as needed, and optimizing the Cooling tower action...

- **Long Term:** The long-term recommendations for the cooling system include ongoing monitoring of temperature and workload distributions, regular adjustments to valve positions as needed, and continued optimization...

- **Trade Off Analysis:** The trade-off analysis for the cooling system involves balancing system stability and energy efficiency. The current configuration and operating conditions of the cooling system are optimized to maintain...

SCIENTIFIC INSIGHTS:

The scientific insights from the current system behavior include the optimal temperature setpoints for the five CDUs, the appropriate valve positions to ensure equitable cooling fluid distribution, and the ideal cooling tower action to maintain stability and efficiency. These insights are derived from a thorough analysis of the system's performance metrics, including temperature stability, energy efficiency, and workload distribution.

Inference Time: 15.25 seconds

415

416 **I.3 Qwen 3-8B Short Explanation**

Short Explanation

EXPLANATION:

The cooling system adjusted valve positions across cabinets to balance cooling and energy efficiency. Cooling tower operation was set to very low cooling due to stable temperature trends.

TEMPERATURE IMPACT:

- **Overall:** Temperatures remain within the normal operating range with minor deviations.

- **Cooling Tower:** Very low cooling reduces system cooling capacity, which may limit temperature control effectiveness.

ENERGY EFFICIENCY IMPACT:

Very low cooling tower operation contributes to moderate energy efficiency.

Inference Time: 1.11 seconds

417

418 **I.4 Qwen 3-8B Detailed Explanation**

Detailed Explanation

EXECUTIVE SUMMARY:

The cooling system is maintaining stable temperatures with a focus on energy efficiency, but some cabinets are not meeting their temperature goals. Cooling tower action is at a very low level, and valve positions are evenly distributed across blade groups to balance power consumption.

TEMPERATURE STABILITY:

- **Overall Assessment:** The system is maintaining stable temperatures with a stability score of 0.634, indicating moderate control effectiveness. While most cabinets are within the normal operating range, some deviations suggest room for improvement.

- **Cabinet Temperature Control:** Valve positions are evenly distributed across blade groups (each group receiving ~33.3% of cooling), which helps maintain consistent temperature distribution. However, the current cooling tower action is at a very low level, limiting overall cooling capacity and causing some cabinets to exceed their temperature goals.

- **Cooling Tower Impact:** The very low cooling tower action (category 0) is significantly restricting cooling capacity, leading to higher than desired temperatures in some cabinets. This suggests that increasing cooling tower activity could improve temperature stability, though it would likely increase energy consumption.

ACTION RATIONALE:

- **Temperature Setpoints:** Temperature setpoints are aligned with the normal operating range (293.15K-

419

313.15K). Cabinet goals vary, with some cabinets having lower goals (e.g., cdu-cabinet-1 at 20.7°C) and others having higher goals (e.g., cdu-cabinet-5 at 39.9°C). The current temperatures are mostly within this range, but some cabinets, like cdu-cabinet-5, are significantly below their goals, indicating potential undercooling or incorrect setpoint alignment.

- **Valve Positions:** Valve positions are evenly distributed across blade groups (each group receiving ~33.3% of cooling), which ensures balanced cooling across all components. This distribution matches the power distribution across blade groups, maintaining thermal equilibrium and preventing localized overheating.

- **Cooling Tower:** The cooling tower is set to a very low cooling action (category 0), which minimizes energy consumption but also limits cooling capacity. This decision prioritizes energy efficiency but results in some cabinets not reaching their temperature goals, particularly those with higher cooling requirements.

ENERGY EFFICIENCY:

- **Assessment:** Energy efficiency is moderate, with a score of 0.616. The low cooling tower action directly contributes to energy savings, but this comes at the cost of reduced cooling capacity and potential temperature instability.

- **Trade Offs:** The current actions prioritize energy efficiency by limiting cooling tower activity, which results in some cabinets being cooler than their goals while others are warmer than desired. This trade-off between energy savings and temperature stability needs to be balanced for optimal system performance.

RECOMMENDATIONS:

Consider increasing cooling tower activity to improve temperature stability, especially for cabinets with higher cooling requirements. Additionally, review and adjust temperature setpoints to ensure they align with actual operational needs and system capabilities.

Inference Time: 4.91 seconds

420

421 I.4.1 Qwen 3-8B Very Detailed Explanation

Very Detailed Explanation

EXECUTIVE SUMMARY:

The cooling system shows stable temperature deviations with minimal fluctuations, indicating consistent cooling performance. The cooling tower is operating at very low cooling capacity (action 0), which is likely causing reduced cooling efficiency. Cabinet temperatures are below their respective goals, particularly cabinet 5, which has a significant deviation of -13.62K, suggesting potential overcooling.

SYSTEM STATE ANALYSIS:

- **Temperature Overview:**

Cabinet temperatures are predominantly below their setpoints, with cabinet 5 showing the most significant deviation (-13.62K) at 26.38°C versus a goal of 39.99°C. Cabinet 2 and 4 also exhibit notable deviations (-7.93K and -7.81K, respectively), while cabinets 1, 3, and 5 are significantly below targets. This suggests that the cooling system is overcooled...

- **Workload Distribution:**

Power distribution across cabinets is uniform, with each cabinet receiving approximately 33.33% of the total power. This equal distribution indicates a balanced computational workload, which is essential for maintaining thermal stability...

- **Historical Context:**

The system has maintained stable temperature deviations and efficiency scores over time, indicating consistent performance. However, the current low cooling tower operation (action 0) suggests a potential shift towards energy conservation...

- **System Goals:**

Current conditions deviate from the system's primary goal of maintaining optimal temperature while minimizing energy consumption. The overcooling of cabinets, particularly cabinet 5, indicates that the cooling tower's low operation is not effectively...

CABINET CONTROL ANALYSIS:

- **Temperature Setpoints:**

- **Description:** The temperature setpoints for each cabinet are set to specific target temperatures, with cabinet 5 having the highest goal of 39.99°C. The current cabinet temperatures are significantly below these setpoints...

- **Justification:** The setpoints are determined by the required thermal management for each cabinet's workload. Higher power cabinets (e.g., cabinet 5) have higher temperature goals to accommodate increased heat generation...

422

- **Impact Assessment:** Overcooling leads to increased energy use and operational costs, while also potentially causing unnecessary wear on cooling components. The system's stability is maintained, but the energy efficiency score...

- **Valve Distributions:**

- **Description:** Valve positions are uniformly distributed across cabinets, with each cabinet having a similar proportion of cooling resources allocated. This uniform distribution suggests a strategy of equal cooling...
- **Power Alignment:** The uniform valve distribution aligns with the balanced power distribution across cabinets, ensuring that each cabinet receives equivalent cooling capacity. This approach is suitable for systems...
- **Hydraulic Considerations:** The uniform valve distribution ensures consistent coolant flow across all cabinets, which is important for maintaining hydraulic balance in the cooling system. However, this approach may lead...

COOLING TOWER ANALYSIS:

- **Action Details:** The cooling tower is operating at action level 0, which corresponds to very low cooling capacity. This action decreases the cooling tower's heat rejection capacity, potentially leading to reduced cooling efficiency...
- **Thermal Dynamics:** The low cooling tower operation results in reduced heat rejection, which means the system is relying more on internal cooling mechanisms. This could lead to increased temperatures in cabinets...
- **Weather Interactions:** The cooling tower's performance is influenced by ambient temperature and humidity levels. If the ambient temperature is high, the cooling tower's effectiveness decreases, but the current low operation...

ENERGY EFFICIENCY ANALYSIS:

- **Component Efficiency:**

- **Cooling Tower:** The cooling tower's efficiency is reduced due to its operation at very low cooling capacity (action 0). This leads to higher energy consumption for the same cooling effect, decreasing overall system efficiency...
- **Pumps And Valves:** Pump energy consumption is likely stable due to the uniform valve distribution and consistent flow rates. Valves are operating within their efficient range, but the uniform distribution may not be optimal...
- **Secondary Cooling:** The secondary cooling loop is operating efficiently, as evidenced by the stable temperature deviations and consistent power distribution. The secondary cooling system is effectively managing...

- **Optimization Strategy:** The current strategy prioritizes energy conservation by operating the cooling tower at very low capacity. This reduces energy costs but may compromise temperature stability. An optimized strategy would involve...

- **Performance Metrics:** The system's energy efficiency score is 0.6166, indicating room for improvement. The cooling tower's low operation contributes to this score, as it is not effectively managing the thermal load...

RECOMMENDATIONS:

- **Short Term:** Increase cooling tower operation to a moderate level to improve heat rejection capacity and reduce unnecessary overcooling. Adjust valve positions to provide more targeted cooling to cabinets with higher thermal loads...
- **Long Term:** Implement a dynamic cooling tower control strategy based on real-time thermal loads and ambient conditions. Introduce variable valve distribution to match cooling resources with individual cabinet demands...
- **Trade Off Analysis:** Increasing cooling tower operation improves temperature stability but may increase energy costs. Adjusting valve positions for targeted cooling improves efficiency but requires more complex control strategies...

SCIENTIFIC INSIGHTS:

The system's performance highlights the importance of dynamic thermal management in data center cooling. Overcooling, while maintaining stability, leads to increased energy consumption and potential inefficiencies. The uniform valve distribution and low cooling tower operation suggest a conservative approach to energy management, which may not be optimal for long-term sustainability.

Inference Time: 13.49 seconds

I.5 Comparing performance of different LLMs as controllers

We use a multi-faceted evaluation framework to assess control quality.

- **Temperature Stability:** $1 - \frac{\text{avg deviation}}{10}$, normalized against a 10 K bound.
- **Cooling Power (W):** Average cooling tower energy usage.
- **Energy Efficiency:** $1 - \frac{\text{cooling power}}{\text{nominal power}}$, where nominal power reflects peak CT draw.
- **Weighted Reward** = $0.7 \times \text{Temperature Stability} + 0.3 \times \text{Energy Efficiency}$

The results in Figure 4 comprehensively compare base and fine-tuned versions of the LLaMA and Qwen LLM controllers across key thermal and energy metrics. The fine-tuned LLaMA model (LLaMA FT) demonstrates consistent and notable improvements over its base counterpart in nearly all categories. It achieves the highest temperature stability (0.7473), significantly outperforming the base version (0.7176), and exhibits a lower average temperature deviation (2.533 K vs. 2.830 K), indicating more precise thermal regulation. This translates into a higher weighted reward (0.7278), which combines thermal stability and energy efficiency, highlighting the effectiveness of imitation learning in aligning the model with optimal control behavior. Additionally, the LLaMA FT controller improves the time spent within the target temperature range from 1.4% to 17.4%, suggesting that it is better at maintaining desirable thermal conditions over sustained periods.

Qwen’s results offer an interesting contrast. Although the fine-tuned Qwen controller does not outperform its base version across all metrics, the Qwen base model itself achieves the best cooling power consumption (101091.6 W) and the highest energy efficiency (0.6919) among all four controllers, indicating an innate bias toward energy-conserving behavior even without task-specific fine-tuning. However, fine-tuning slightly degrades Qwen’s thermal regulation performance, reducing temperature stability and the time in the target range, possibly due to overfitting or reduced generalization. This performance divergence suggests that while Qwen excels in energy optimization, it may not benefit as uniformly from trajectory-based fine-tuning as LLaMA does.

Overall, these findings highlight that fine-tuning can significantly improve thermal control performance, particularly for LLaMA, while the energy efficiency gains of Qwen appear intrinsic to the model’s base behavior. The results affirm that LLMs can serve as reliable and interpretable controllers, with performance gains model-dependent and tied closely to the nature of the pretraining and fine-tuning alignment.

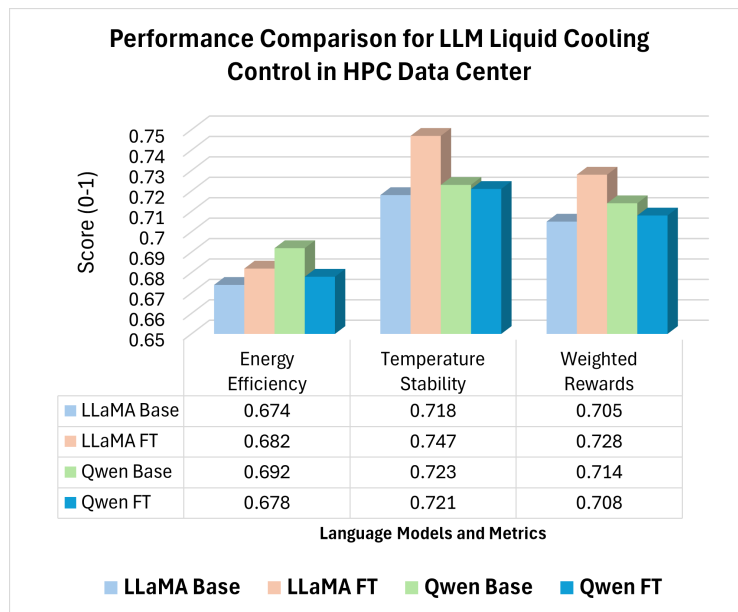


Figure 4: Performance comparison between different LLM controllers for end-to-end control for a HPC Liquid cooled Data Center. The controller optimizes both the Cooling Distribution Unit (CDU), which controls the HPC server cabinets, and the Cooling tower.