# Deploying Reinforcement Learning based Economizer Optimization at Scale

**Jiaron Cui**
Amazon
San Francisco, CA
ivancui@amazon.com

**Wei Yih Yap**
Amazon
Singapore, SG
yihyap@amazon.com

**Charles Prosper**
Amazon
Brisbane, AU
cpro@amazon.com

**Bharathan Balaji**
Amazon
Seattle, WA
bhabalaj@amazon.com

**Jake Chen**
Amazon
San Francisco, CA
jakechen@amazon.com

## Abstract

Building operations account for a significant portion of global emissions, contributing approximately 28% of global greenhouse gas emissions. With anticipated increase in cooling demand due to rising global temperatures, the optimization of rooftop units (RTUs) in buildings becomes crucial for reducing emissions. We focus on the optimization of the economizer logic within RTUs, which balances the mix of indoor and outdoor air. By effectively utilizing free outside air, economizers can significantly decrease mechanical energy usage, leading to reduced energy costs and emissions. We introduce a reinforcement learning (RL) approach that adaptively controls the economizer based on the unique characteristics of individual facilities. We have trained and deployed our solution *in the real-world* across a distributed building stock. We address the scaling challenges with our cloud-based RL deployment on 10K+ RTUs across 200+ sites.

## 1 Introduction

As global temperatures continue to rise, the demand for cooling in buildings is expected to increase, especially in regions with hot climates [Khourchid et al.(2022)]. Without energy-efficiency measures, higher cooling demand will result in even higher emissions. We focus on economizer optimization in rooftop units (RTUs), typically deployed in industrial HVAC systems. We experiment on a site with 10K+ RTUs located in a hot region. The economizer balances the indoor and outdoor air during the conditioning process. By leveraging the utilization of free cold air, the RTU can significantly reduce mechanical energy usage, leading to decreased energy costs and associated emissions. The standard practice for economizer optimization follows the guidelines set by the American Society of Heating, Refrigerating and Air-Conditioning Engineers (ASHRAE). These guidelines rely on hard-coded setpoints based on regional climate data [Edition et al.(2010)]. This standard does not account for the dynamics of individual facilities, and therefore, overlooks valuable opportunities to capitalize on free cooling based on real-time conditions. To overcome these limitations, recent advancements in reinforcement learning (RL) have emerged as promising solutions [Yu et al.(2018), Cui et al.(2022)]. We present an RL approach to adaptively change economizer setpoints, that adjust to the characteristics of individual facilities in real-time.

**Contributions:** Prior works primarily focus on zonal setpoint changes (e.g. [Liu et al.(2022)]), to our knowledge we are the first to focus on economizer optimization. Further, prior works demonstrate RL solutions in simulation [Yu et al.(2018)] or a single building [Liu et al.(2022)], we deploy our

Tackling Climate Change with Machine Learning: workshop at NeurIPS 2023.

solution to 200+ sites with 10K+ RTUs. To our knowledge we are the first to discuss cloud-based deployment of RL based control at scale. We will open-source our code with a permissive license.

## 2 Reinforcement Learning with Soft Actor-Critic

RL has been extensively explored for building optimization tasks [Yu et al.(2018), Cui et al.(2022)]. We use the well-established Soft Actor-Critic (SAC) algorithm [Haarnoja et al.(2018)]. SAC utilizes the state-value function to estimate long-term accumulated reward expected from a given state, and an action-value function to estimate cumulative rewards for state-action pairs. Additionally, SAC employs an advantage function to measure the advantage of an action compared to the average expected value at a state. The Actor's policy neural network is updated using policy gradients and maximizes rewards, while the Critic neural network is refined to approximate the value function. Notably, SAC introduces an entropy regularization term to encourage exploration. This inclusion makes SAC an appropriate choice as the algorithm explores and learns efficiently in environments with stochasticity. Our experiments with SAC showed consistent improvement over the existing rule-based control, and we deem it sufficient for deployment without further experimentation due to development costs. However, other RL algorithms can be incorporated into our framework as a drop-in replacement.

## 3 Problem Formulation

The economizer reduces the load of cooling on the system, and we model how much energy can be saved with free cooling using available sensor measurements (Appendix A includes an overview). Our economizers are equipped to monitor outdoor temperature and humidity, and adjusts the damper position based on a setpoint. Damper position is maintained at a minimum of 10% to ensure sufficient air exchange. Economizers reduce the cooling load by minimizing the enthalpy required to reach the desired supply air temperature setpoint.

We formulate the following MDP in OpenAI gym [Brockman et al.(2016)]:
**State:** Outdoor temperature ($T_{outdoor}$), outdoor humidity ($\mathcal{H}_{outdoor}$), return temperature ($T_{return}$) and return humidity $\mathcal{H}_{return}$. We assume supply air temperature and humidity to be constant. Appendix B lists the constants in our environment.
**Action:** The action space consists of 2 setpoints: economizer maximum temperature ($T_{max-econ}$) and maximum enthalpy ($H_{max-econ}$), both in continuous space. We constrained the action space between [20, 30] and [40, 75] respectively. We select actions every hour.
**Reward:** We estimate the power required to meet the required supply air temperature with the given economizer status using the Total Heat Gain equation, which indicates longer run time for mechanical cooling. Our reward function is:

$$reward = (H_{mixed} - H_{supply}) * airflow * 4.5/100000 \tag{1}$$

where, $H_{mixed}$ is the mixed air enthalpy, $H_{supply}$ is the supply air enthalpy, $airflow$ is measured in CFM, and the constants account for the standard air density, and the conversion factor from minutes to hours. The episode length is set to 200 steps.

We calculate the enthalpy and estimate the percentage of outside air ($\%OA$) in the mixed air, and make control decision with actions selected by the SAC model. The enthalpy of the mixed air can be estimated using temperature and humidity measurements. We use the following equation:

$$H = 0.240 * T + \mathcal{H} * ((0.444) * T + 1061) \tag{2}$$

where $H$ is enthalpy in Btu/lb, $\mathcal{H}$ is humidity and $T$ is temperature in °F. We use the same equation to compute $H_{supply}$, $H_{return}$ and $H_{mixed}$, with corresponding values of temperature and humidity.

The economizer system utilizes sensors to monitor the return temperature ($T_{return}$) and return air humidity ($\mathcal{H}_{return}$). Outdoor temperature ($T_{outdoor}$) and humidity ($\mathcal{H}_{outdoor}$) sensors provide information about the external conditions. When the outside air enters through the damper, it mixes with the return air, resulting in a mixed air stream that passes through the heating and cooling coil before being supplied to the facility. The temperature of this mixed air is denoted as $T_{mixed}$. Since most RTUs lack sensors for measuring mixed air temperature and humidity directly, an approximation can be made using the formula below:

$$\%OA = (T_{return} - T_{supply})/(T_{return} - T_{outdoor}) \tag{3}$$

Table 1: Comparison of power consumption of RL agent against the ASHRAE standard

| | **No Economizer** | **ASHRAE** | **RL Agent** |
|---|---|---|---|
| Average Power Consumption (BTU/hr) | 290.9 | 270.4 | 266.5 |

$$T_{mixed} = \%OA * T_{outdoor} + (1 - \%OA) * T_{return} \tag{4}$$

We also estimate $\mathcal{H}_{mixed}$ using the same method.

In our formulation, the hourly time step and the energy-based reward function does not necessarily require an RL solution because each time step can be optimized independently. However, the use of neural networks with deep RL does make even this greedy optimization much easier to solve compared to rule-based control as the agent learns the non-linear relationships and automatically optimizes control for a specific site. The MDP framework also makes it easy to switch to a dynamic pricing or carbon intensity based reward in a future deployment, which does have a dependency across time steps.

## 4 Validation With Historical Data

We collected historical outdoor temperature, outdoor humidity, return temperature, and return humidity data from an RTU, recorded every 15 minutes from May 2018 to October 2021. We filtered out records that were either missing or anomalous, converted both outdoor and return humidities to decimal value and clipped them between range of [1e-5, 1], and resampled the data into hourly granularity by taking the average. We hold out 1 month of data for validation. After data cleaning and pre-processing, the processed dataset contains around 12K records. We trained our SAC agent for a total of 200K time steps. Appendix C describes the full algorithm.

Table 1 shows the power consumption between different strategies in updating economizer setpoints for the validation dataset of one RTU. We compare against two baselines: no economizer, and the ASHRAE standard. The RL agent outperforms ASHRAE by taking advantage of facility specific conditions, and yields 5% reduction in average power use. We deployed the solution on 10K+ RTUs across 200+ sites for hourly inferencing. We train one agent per site, but use the same policy for all the RTUs in a site. We re-train the RL agent each quarter to account for drift in environment conditions. We redact online deployment results due to proprietary reasons.

## 5 Deploying at Scale

Appendix D describes our deployment architecture.

### 5.1 Data standardization

Building data is disorganized, with inconsistent naming conventions for sites, assets, and sensors due to various factors such as different teams, vendors, or engineers involved during setup. As a result, deploying downstream analytics, including RL policies, becomes a time-consuming process that requires customization per each site, and thus limits scalability. These challenges are well-known, and have been documented in prior works [Balaji et al.(2015), Balaji et al.(2016)].

To address the data standardization challenge, we leveraged the Brick schema, which provides a well-defined ontology for building-related data, ensuring consistency and uniformity in data representation and structure [Balaji et al.(2016)]. We load standardized sites, RTUs, sensors, and their relevant relationships into a Graph Database. We use SPARQL queries to retrieve relevant sites, assets, and points associated with our MDP. Appendix D includes examples of data points and sample queries.

### 5.2 Data scalability

We needed a scalable solution to store and process 1 year worth of historical training data, totaling 18.6 billion data point for training 200 RL models quarterly. The model inference occurs hourly, where we need to retrieve 157 million data points, perform inference, and update the building management system for 10K+ RTUs.

We utilize a combination of long-term (cold) storage and short-term (hot) storage to ensure a robust and scalable data management approach for model training, model inference, and other ad-hoc analytical use cases. As long-term storage of training data is only accessed for quarterly retraining, we leverage Block Storage to store years of data at the lowest granularity possible. This method is cost efficient for infrequent reads yet allows for the transformation of the raw data for additional analytical use cases. For model inference and real-time analytics, we store incoming streaming sensor data into purpose-built timeseries SQL databases. These tools are specifically designed to handle the simultaneous reading and writing of high-volume and high-frequency sensor data.

After retrieving the relevant data points required from a SPARQL query (results include a timeseries id), we parallelize the SQL queries to retrieve the time series data across sites and RTUs efficiently. Each query focuses on a specific site and RTU combination. By combining the power of a Graph Database and parallel SQL queries to timeseries databases, we effectively leverage standardized data to scale our deployment.

### 5.3 Compute scalability

We aim to concurrently perform inference across hundreds of sites every hour, and run model training every quarter. We rely on the serverless capabilities provided by Cloud compute to meet these scaling requirements. In our use case, the SPARQL queries would return a list of sites with a list of RTUs per site. During model training, we use serverless orchestration tools to map training jobs to every RTU associated with a site, resulting in tens-of-thousands of concurrent jobs. We process the concurrent training jobs using a batch processing service, which is suited for GPU intensive process of environment simulation and subsequent training of the RL agent. We store the resulting policies in long-term storage for later model inference. These concurrent training jobs take up to an hour each.

During model inference, we also use workflow orchestration to map training jobs to every RTU associated with a site. However, we process the concurrent jobs using function-as-a-service (FaaS) instead of batch processing. We use CPUs in FaaS to reduce costs and availability issues, and complete concurrent inferences within 5 minutes on average.

Using this architecture, we have already scaled to 10K+ RTUs, and do not foresee difficulties in continuing to scale. The cost of our deployment is $50K/year, and we estimate saving >$250K in annual energy bills.

### 5.4 Human-in-the-loop Control

Although we have been able to automatically recommend economizer setpoints across hundreds of sites, we have not yet automated the command and control of the RTUs themselves. Instead, we output the recommended setpoints to a web-based front-end that displays the setpoints as well as the potential energy savings. Building operators review and input these recommended setpoints manually. The reasons for this are twofold. First is the ongoing security discussions associated with automating command-and-controls. While integration with the command and control suite will allow setpoint automation, it would also grant access to more critical setpoints that may disable the RTU completely. The second reason is the novelty and subsequent lack of trust with the algorithm, especially given the complexity of an RL solution. We are currently evaluating the performance with end users and stakeholders to establish trust in the recommended setpoints. Once the security issues are resolved, we expect building operators will slowly taper off the manual validations and allow more setpoints to be automated.

## References


[Balaji et al.(2016)] Bharathan Balaji, Arka Bhattacharya, Gabriel Fierro, Jingkun Gao, Joshua Gluck, Dezhi Hong, Aslak Johansen, Jason Koh, Joern Ploennigs, Yuvraj Agarwal, et al. 2016. Brick: Towards a unified metadata schema for buildings. In *Proceedings of the 3rd ACM International Conference on Systems for Energy-Efficient Built Environments*. 41–50.

[Balaji et al.(2015)] Bharathan Balaji, Chetan Verma, Balakrishnan Narayanaswamy, and Yuvraj Agarwal. 2015. Zodiac: Organizing large deployment of sensors to create reusable applications for buildings. In *Proceedings of the 2nd ACM international conference on embedded systems for energy-efficient built environments*. 13–22.

[Brockman et al.(2016)] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. Openai gym. *arXiv preprint arXiv:1606.01540* (2016).

[Cui et al.(2022)] Can Cui, Chunxiao Li, and Ming Li. 2022. An Online Reinforcement Learning Method for Multi-Zone Ventilation Control With Pre-Training. *IEEE Transactions on Industrial Electronics* 70, 7 (2022), 7163–7172.

[Edition et al.(2010)] SI Edition, DH Erbe, MD Lane, SI Anderson, PA Baselici, S Hanson, R Heinisch, J Humble, S Taylor, and R Kurtz. 2010. Energy standard for buildings except low-rise residential buildings. *ASHRAE, Atlanta, GA, USA, Tech. Rep. IP Edition* (2010).

[Haarnoja et al.(2018)] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*. PMLR, 1861–1870.

[Khourchid et al.(2022)] Ammar M Khourchid, Salah Basem Ajjur, and Sami G Al-Ghamdi. 2022. Building cooling requirements under climate change scenarios: Impact, mitigation strategies, and future directions. *Buildings* 12, 10 (2022), 1519.

[Liu et al.(2022)] Hsin-Yu Liu, Bharathan Balaji, Sicun Gao, Rajesh Gupta, and Dezhi Hong. 2022. Safe hvac control via batch reinforcement learning. In *2022 ACM/IEEE 13th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, 181–192.

[Yu et al.(2018)] Liang Yu, Di Xie, Chongxin Huang, Tao Jiang, and Yulong Zou. 2018. Energy optimization of HVAC systems in commercial buildings considering indoor air quality management. *IEEE Transactions on Smart Grid* 10, 5 (2018), 5103–5113.

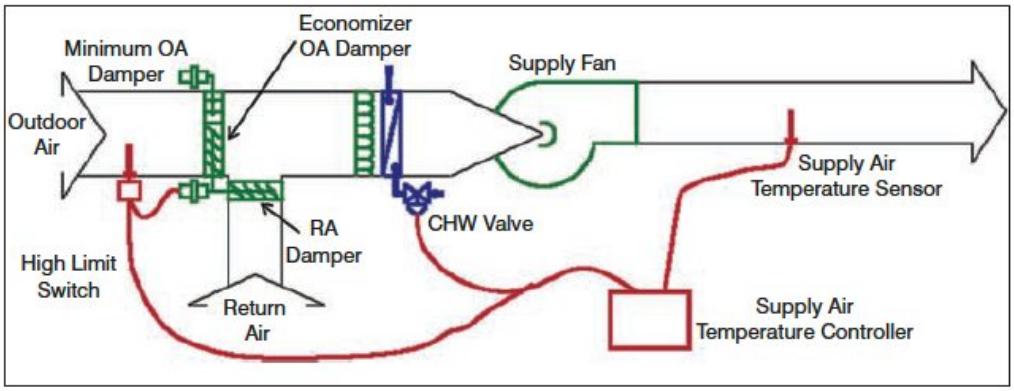


Figure 1: Illustration of an RTU equipped with an economizer

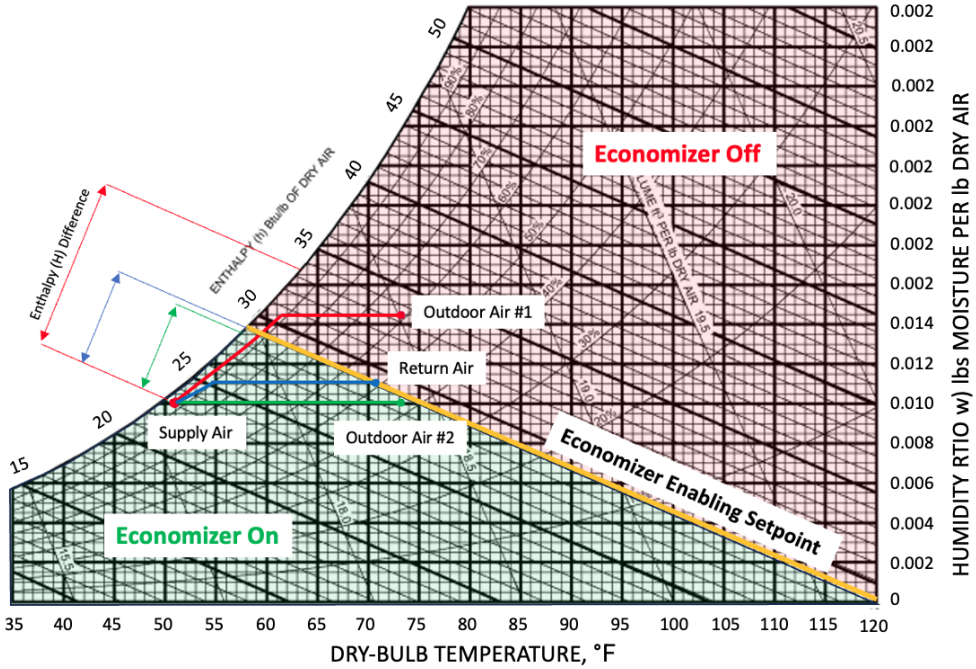


Figure 2: Psychrometric Chart for Economizer Operation

## A Basics of Economizer Operation

Figure 1 shows the basic operation of an economizer. A damper controls the amount of outside air mixed in with the return air for conditioning the building. A minimum amount of outside air is necessary for maintaining air quality, but the amount of air ingressed has a significant impact on energy used for cooling or heating the supply air.

Figure 2 shows the psychrometric chart for economizer operation, which captures the relationships between temperature, humidity, and enthalpy. The neural network in our RL agent learns these relationships, and optimizes the economizer for the local weather patterns by maximizing the long-term reward.

In the case of differential enthalpy control with fixed dry-bulb temperature, the economizer enabling setpoints depend on the maximum enthalpy and maximum dry-bulb temperature allowed for the economizer. This results in the division of the psychrometric chart into two distinct operating regions as shown in Figure 2.

Table 2: Constants in the RL environment.

| Name of Variable | Value (Unit) |
|---|---|
| Supply Temperature ($T_{supply}$) | 55 ($^\circ$F) |
| Supply Humidity ($\mathcal{H}_{supply}$) | 0.5 (PPM) |
| Minimal Economizer Enthalpy Setpoint ($H_{min-econ}$) | 20 (BTU/lb) |
| Maximal Economizer Enthalpy Setpoint ($H_{max-econ}$) | 30 (BTU/lb) |
| Min. Economizer Temperature Setpoint ($T_{min-econ}$) | 40 ($^\circ$F) |
| Max. Economizer Temperature Setpoint ($T_{max-econ}$) | 75 ($^\circ$F) |
| Minimal Outside Air Ratio ($OA_{min}$) | 0.1 |
| Supply Airflow ($U$) | 8000 (CFM) |
| Cooling Enable Setpoint ($Cool_{sp}$) | 55 ($^\circ$F) |

# B Constants used in RL formulation

Table 2 lists the constants used in the problem formulation.

# C Algorithm to train RL policy for economizer optimization

Algorithm 1 describes each step in the process, with $f$ denoting Equation 2.

**Algorithm 1:**

**Input :** $T_{outdoor}, \mathcal{H}_{outdoor}, T_{return}, \mathcal{H}_{return}, max\_steps$
**Output :** R, action, SAC
Initialize: SAC, steps, RL agent SAC
**for** *each step* **do**
  $state = T_{outdoor}, T_{return}, \mathcal{H}_{outdoor}, \mathcal{H}_{return}$
  $action = SAC(state)$
  $H_{supply} = f(T_{supply}, \mathcal{H}_{supply})$
  $H_{outdoor} = f(T_{outdoor}, \mathcal{H}_{outdoor})$
  $H_{return} = f(T_{return}, \mathcal{H}_{return})$
  **if** *($H_{outdoor} > H_{max-econ}$) or ($T_{outdoor} > T_{max-econ}$)* **then**
    $\%OA = OA_{min}$
  **else**
    $\%OA = (T_{return} - T_{supply})/(T_{return} - T_{outdoor})$
  $H_{mixed} = \%OA * H_{outdoor} + (1 - \%OA) * H_{return}$
  $T_{mixed} = \%OA * T_{outdoor} + (1 - \%OA) * T_{return}$
  $R = ((H_{mixed} - H_{supply}) * U * 4.5))/100000$
  $SAC = update(SAC, state, action, R)$
  $steps = steps + 1$
  **if** *(steps > max_steps)* **then**
    break

# D Cloud deployment architecture

We leverage cloud services to scale our deployment, Figure 3 shows our architecture. We discuss the challenges in scaling and the solutions employed below.

## D.1 Data standardization

Building data is often disorganized, with inconsistent naming conventions for sites, assets, and sensors due to various factors such as different teams, vendors, or engineers involved during setup. As a result, deploying downstream analytics, including RL policies, becomes a time-consuming process that requires customization per each site, and thus limits scalability. These challenges are well-known, and have been documented in prior works [Balaji et al.(2015), Balaji et al.(2016)].

To address the data standardization challenge, we leveraged the Brick schema, which provides a well-defined ontology for building-related data, ensuring consistency and uniformity in data representation

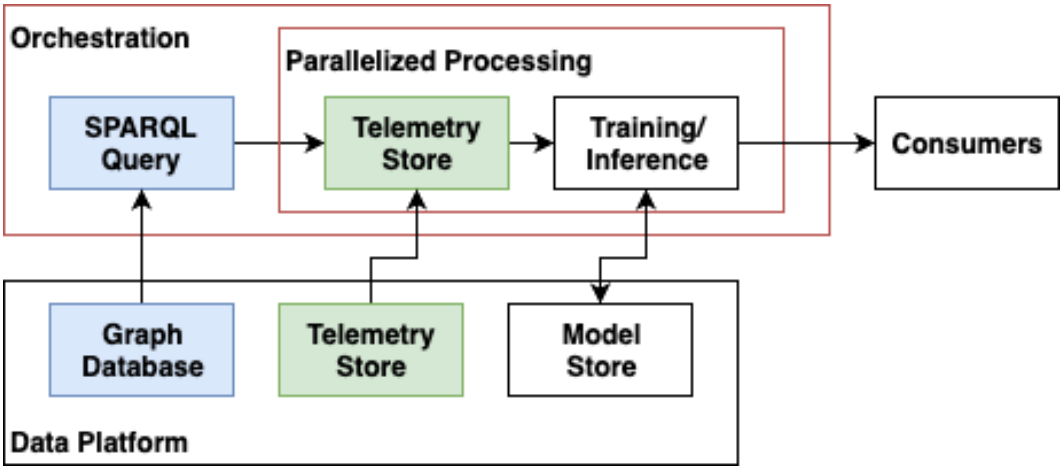


Figure 3: Cloud deployment architecture

and structure [Balaji et al.(2016)]. For example, our use case involved handling multiple vendors with different naming conventions for Zone Temperature Sensor, as shown below.

```
Vendor1: ZoneTemperature
Vendor2: Zone_Temperature
Vendor3: zoneTemp
```

We created a mapping file that maps all 17 sensors with a list of all possible name variants to the corresponding brick name during data ingestion. We also ingest their relevant relationships into a Graph Database to establish a standardized foundation for subsequent building systems analytics. This standardizing is crucial for achieving scalability and enabling efficient analytics processes such as mapping the question below into a Graph query.

What is the Zone_Air_Temperature_Setpoint for RTU1 from site1?

```
SELECT ?point ?timerseries_id
WHERE {
  ?site BRICK:hasName "site1" .
  ?rtu BRICK:hasName "RTU1" .
  ?site BRICK:hasLocation ?rtu .
  ?rtu a BRICK:RTU .
  ?rtu BRICK:hasPoint ?point .
  ?point a BRICK:Zone_Air_Temperature_Setpoint .
  ?point BRICK:timeseries ?timerseries_id .}
```

This standarization also allowed a parallelized approach for model training and inference pipeline using parameterized query templates. We first use SPARQL queries to retrieve relevant sites, assets, and points associated with our MDP. A sample query is shown below. This query selects the desired entities based on the specific requirements of the energy optimization model. The output of this SPARQL query provides the necessary inputs for the subsequent parallelization process.

```
SELECT ?site ?asset ?point ?timerseries_id
WHERE { ?asset BRICK:hasLocation ?site;
      a BRICK:RTU;
      BRICK:hasPoint ?point.
?point BRICK:timeseries ?timeseries_id.}
```

Secondly, the SQL queries can be parallelized across sites and RTUs using the outputs from the previous SPARQL query. This means running multiple SQL queries in parallel, where each query focuses on a specific site and RTU combination. For example, if there are n sites and m RTUs, there will be a total of n*m parallel SQL queries running concurrently. This parallelization approach distributes the computational load across multiple resources, enabling faster processing and analysis of the data.

```
SELECT ?site ?asset ?point ?timerseries_id
TBD
```

### D.2 Data scalability

In our use case, the end goal is to scale the energy optimization model to thousands of sites, each containing tens to hundreds of Rooftop Units (RTUs). However, traditional relational databases are not designed to simultaneously handle both the long-term requirements for model training alongside real-time requirements for model inference and analytics, particularly in our case of multiple years of minute to sub-minute level data from millions of sensors. While certain data architecture optimizations like redundancy, joins, partitioning, and other techniques may be effective in specific use cases, the ever-changing and adhoc nature of analytics continually introduces new requirements.

We utilize a combination of long-term (cold) storage and short-term (hot) storage to ensure a robust and scalable data management approach for model training, model inference, and other ad-hoc analytical use cases. For long-term storage of training data that is only accessed for quarterly retraining, we leverage Block Storage to store years of data at the lowest granularity possible. This method is cost efficient for infrequent reads yet allows for the transformation of the raw data into other storage systems for analytical use cases outside of energy optimization. For model inference and real-time analytics, we store incoming streaming sensor data into purpose-built timeseries SQL databases. These tools are specifically designed to handle the simultaneous reading and writing of high-volume and high-frequency sensor data.

After retrieving the relevant data points required from a SPARQL query (results include a timeseries id), we parallelize the SQL queries using a Serverless Query Service to retrieve the time series data across sites and RTUs simultaneously. Each query focuses on a specific site and RTU combination. For example, if there are n sites each with m RTUs, there will be a total of n*m parallel SQL queries running concurrently. This means all all SQl queries return results around the time it takes a single query to complete.

By combining the power of a Graph Database and parallelized SQL queries to Serverless timeseries databases, we effectively leverage standardized data to scale our deployment.

### D.3 Compute scalability

Moreover, traditional server-based architectures struggle to handle the predictable but spiky demand associated with a parallelized approach required for scaling. In our case, we need a scalable solution to process 1 year worth of historical training data, totaling 18.6 billion data point for training 1000+ RL models quarterly. This pose compute challenges but not major concern due to training frequency. However, model inference takes place hourly, and within each hour, we need a scalable solution that can not only concurrently retrieve 157 million data points but also to perform inferences across all sites and all RTUs. Performing these inferences serially would not finish within the hour. If we parallelize the hourly inferences, they can be completed within 10 minutes every hour. In this case, traditional server-based approaches would leave the computational resources idle for the remaining 50 minutes, incurring additional infrastructure and operating cost that is unnecessary. While multiple methods exist to to smooth out this demand, these methods require additional architectural considerations such as queuing, artificial waits, or sub-parallelization across groups like sites or assets. These measures add complexity and development time when compared to a parallelized approach, leaving less time for the development of new models and analytical capabilities. Instead, we rely on the serverless compute capabilities provided by Cloud compute to meet these scaling requirements

In our use case, the SPARQL queries would return a list of sites with a list of RTUs per site. An example of the payload return is shown below:

```
{site: site1, asset: [RTU1, RTU2, ..., RTUn]}
```

During model training, we use serverless orchestration tools to map training jobs to every RTU associated with a site, resulting in tens-of-thousands of concurrent jobs. We process the concurrent training jobs using a batch processing service, which is suited for GPU intensive process of environment simulation and subsequent training of the RL agent. We store the resulting policies in long-term storage for later model inference. These concurrent training jobs take up to an hour each.

During model inference, we also use workflow orchestration to map training jobs to every RTU associated with a site. However, we process the concurrent jobs using function-as-a-service (FaaS) instead of batch processing. We use CPUs in FaaS to reduce costs and availability issues, and complete concurrent inferences within 5 minutes on average.

Using this architecture, we have already scaled to 10K+ RTUs, and do not foresee difficulties in continuing to scale.