# A cost revenue model extension of OpenDC

Group 7 - Lab Report

Mark Wiering
m.a.wiering@student.vu.nl

Jordy Montanaro
j.montanaro@student.vu.nl

Daniel Mol
d.f.mol@student.vu.nl

Nico de Gier
n.de.gier@student.vu.nl

Aayush Joglekar
a.joglekar@student.vu.nl

Thomas Glansdorp
t.p.glansdorp@student.vu.nl

Jure Antunović
j.antunovic@vu.nl
*Lab supervisor*

Jesse Donkervliet
j.donkervliet@atlarge-research.com
*Course instructor*

Alexandru Iosup
a.iosup@atlarge-research.com
*Course instructor*

December 2025

## Abstract

Modern society has fully entered the age of distributed systems. This means much work has been done to improve the performance and reliability of data centers. However, tools for understanding the financial implications of data center design choices remain limited or inaccessible. While the OpenDC simulator enables the analysis of workloads and resource management, it currently lacks a built-in framework for economic analyses. To address this gap, we present a cost–revenue model extension for OpenDC that integrates Total Cost of Ownership (TCO) metrics directly into the simulation workflow. The model accounts for three primary cost categories: energy costs, utilization costs, and degradation costs. The model is validated through four distinct experimental scenarios ranging from low-cost to high-cost configurations. The results demonstrate that the system successfully meets defined requirements for calculating and visualizing energy, maintenance, and operational costs. Furthermore, the model adheres to performance constraints, maintaining a simulation runtime overhead of less than 3% while scaling effectively to larger topologies. This extension transforms OpenDC into a unified environment for analyzing the interplay between architectural decisions, sustainability, and financial viability.

## 1 Introduction

Modern society increasingly depends on the use of data centers, as they form the backbone of the digital world that is now an integral part of everyday life. Indeed, they power a multitude of essential services, from cloud computing and online communication to financial systems and scientific research, making their performance and reliability crucial. As demand increases, understanding and improving data center behavior has become more important than ever. To this end, the *@Large research group* developed OpenDC, an open-source data center simulator for designing, analyzing, and experimenting with data center architectures [11]. It allows researchers, engineers, and students to model workloads, explore resource-management strategies, and evaluate energy and performance trade-offs in a controlled environment. Consequently, OpenDC's simulations can help users make informed decisions that contribute to more efficient and resilient data centers. However, it currently lacks a built-in framework for assessing the financial implications of data center design choices.

The main metric used to determine the costs of a datacenter is the Total Cost of Ownership (TCO) [2]. There are many different TCO models [13] [8] [3], but in general a TCO model provides an overview of expected costs. Usually, costs fall into one of two categories: capital costs (e.g. hardware prices, installation costs) and operational costs (e.g. power consumption, employee salaries, hardware amortization)[2]. Previous studies have have focused on optimizing datacenter revenue through the well-established and often used Cobb-Douglas production function [17] [16]. Beneficial features of this function include positively decreasing marginal product and constant output elasticity [16]. While these models offer valuable insights into pricing structures and cost behaviors, they are generally not integrated into simulation tools.

In terms of data center simulators, there are some industry alternatives to OpenDC, though to our knowledge there are none that reach the scale of OpenDC whilst also being open source. Most simulators do not offer financial metrics in their simulation, such as NVIDIA Air [12] and RTDS Simulator [6]. These offer similar features as OpenDC in terms of dynamic topologies and workloads, and also implement other systems such as cooling infrastructure. Some have included TCO tracking in their simulation, such as EkkoSim [5]. However, as previously mentioned, these tools are not open source, making them highly unusable for educational scenarios.

In general, existing approaches do not provide a unified environment in which workload behavior, architectural decisions, sustainability metrics, and financial outcomes can be analyzed together in an educational and open source method.

To this end, we aim to extend OpenDC with a dedicated cost–revenue model that allows users to evaluate the economic impact of data-center design choices. Our system introduces a set of financial tools that translate resource usage into cost indicators. The model will be fully integrated into the OpenDC simulation workflow, allowing users to conduct experiments that combine performance, energy use, and financial outcomes. With this extension, we enable more comprehensive analysis within OpenDC and provide a foundation for economically informed decision making in data center research.

This paper will first discuss the inner workings of OpenDC, as well as formalizing the requirements for our model. Next, it discusses the technical design, before moving on to the experimental design. Finally, we analyze and visualize the model's results and followed by the conclusion and discussion.

## 2 Background

As mentioned in 1, our model extends that of the OpenDC tool. OpenDC is a platform for datacenter simulation, which allows its users to explore different workloads and topologies in a controlled environment [11]. It is mostly written in Java and Kotlin. OpenDC consists of three components: the front-end, the simulator and additional simulation tools. Of main importance to this paper is the simulator,.

The simulator is formed by a topology, which consists of clusters and hosts within them. Each host has a number of GPU and CPU cores, core speed, and memory. The simulator workloads are bags of tasks, where each task has a set of fragments linked to it that determine the computational complexity of the task. It is also possible to define the behavior of a number of processes, such as checkpointing, allocation policies, and failure models. Combining all this, the simulator moves through timesteps, updating the tasks in accordance with all the aforementioned parameters. Finally, it produces a multitude of output Parquet files, which contain information ranging from timestamps and CPU failures to carbon emissions and power draw. Our project focuses on adding more outputs here, namely the financial metrics we describe in the next section 2.1.

### 2.1 Requirements Definition

- **R1-EnergyCost:** The system must calculate the total energy costs of the data center at any time, both for fixed and variable energy costs.

- **R2-MaintenanceCost:** The system must be able to calculate the maintenance cost of the data center, by considering hardware amortization, salaries, and property costs.

- **R3-RevenueComparison:** The system must be able to visualize costs versus revenue when provided with a revenue model to be able to more easily calculate prices, for example for FaaS.

- **R4-CostBreakdown:** The system must give a clear visualization of the distribution of costs of the data center, to provide users of OpenDC with succinct information on which costs are most significant in their operations.

- **R5-Performance:** The model should have a low impact on the performance of OpenDC, i.e., the runtime of a particular workload should not increase more than 3% when using the cost model compared to the base.

- **R6-Scalability:** The model must be able to maintain accurate cost prices with increasingly large topologies and/or workloads.

## 3 System Design

To develop the cost revenue model, we analyzed the TCO model provided by [2], which can be seen in figure 3. The most relevant costs are selected and divided into three categories: energy cost, utilization cost and degradation cost. The energy cost depends on the total amount of energy consumed by the data center. The utilization cost varies between data centers and includes expenses such as building rent or maintenance, salary costs, and other operational overhead. Finally, the degradation cost reflects the expenses associated with component replacement over time.

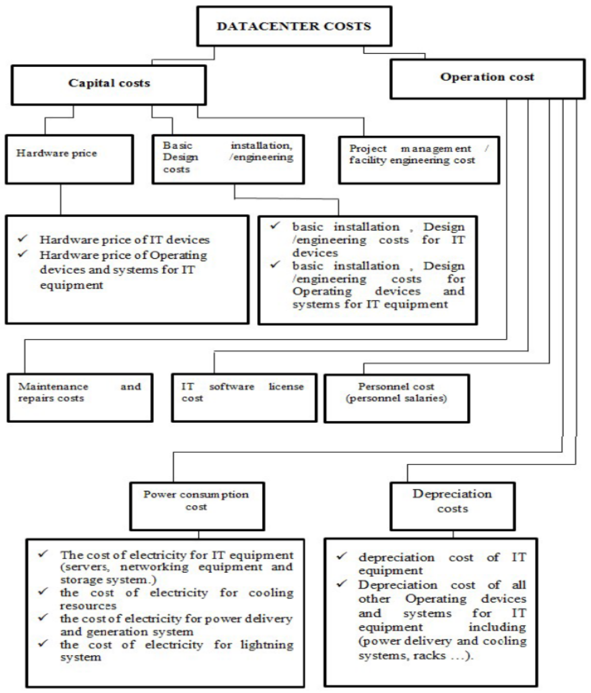

DATACENTER COSTS
Capital costs
Operation cost
Hardware price
Basic installation, Design /engineering costs
Project management / facility engineering cost
✓ Hardware price of IT devices
✓ Hardware price of Operating devices and systems for IT equipment
✓ basic installation , Design /engineering costs for IT devices
✓ basic installation , Design /engineering costs for Operating devices and systems for IT equipment
Maintenance and repairs costs
IT software license cost
Personnel cost (personnel salaries)
Power consumption cost
Depreciation costs
✓ The cost of electricity for IT equipment (servers, networking equipment and storage system.)
✓ the cost of electricity for cooling resources
✓ the cost of electricity for power delivery and generation system
✓ the cost of electricity for lightning system
✓ depreciation cost of IT equipment
✓ Depreciation cost of all other Operating devices and systems for IT equipment including (power delivery and cooling systems, racks …).

Figure 1: A TCO model of a datacenter

The impact of reduced CPU and GPU performance due to wear is accounted for in both power consumption and revenue. As performance declines, processors must either run longer or extra processors need to be supplemented to complete the same tasks. This reduces the total number of jobs the data center can complete at a given time, lowering potential revenue, while the extended run times also increase power usage.

In this model, the total revenue is provided by the user, since only the user can accurately estimate the types of jobs and contracts their data center will handle and the income these activities will generate in their specific case.

The code for this model can be found on a public fork of the OpenDC GitHub here: `https://github.com/zerefwayne/ds-opendc`, and the code for the necessary demos is found on the public fork of the demo GitHub: `https://github.com/Nico213/opendc-demos`.

## 3.1 Energy costs and pricing

As this analysis focuses on data centers in the Netherlands, the Dutch legal and market framework for large electricity consumers is applied. According to the Dutch Energy Act, effective per January 1st 2026, an electricity consumer is classified as a large consumer (Dutch: grootverbruiker) once their connection capacity exceeds 3x80 amperes, typically corresponding with an annual electricity capacity of 70,000 kWh or higher [18].

Large consumers are not supplied with energy through standard retail contracts. Instead, they must arrange their own contracts directly with the grid operator and have to choose their own metering company for the measurement of electricity consumption [18]. As a result, large consumers procure electricity directly from the wholesale power market.

Electricity prices are set on wholesale markets based on marginal pricing. Gas-fired power plants often act as the price-setting technology. However, with the share of intermittent renewable energy sources increasing and grid constraints persisting, the volatility of electricity prices is expected to continue to increase [1]. Due to their high and continuous electricity demand, data centers are particularly exposed to volatility in wholesale electricity.

For this model we look at two types of energy contracts: fixed and variable.

### 3.1.1 Fixed price energy contracts

Power Purchase Agreements (PPAs) are commonly used as a contractual structure for electricity procurement, to avoid this forementioned volatility [1]. Under a PPA, a data center commits to buying electricity from an energy developer at a set price for a set period of time. In doing so, the data center decreases their exposure to the volatility of electricity prices in the evolving electricity markets, thereby improving their cost predictability [1].

As data centers qualify as large-scale electricity consumers, they typically enter into corporate PPAs. Merchant PPAs, in which electricity is purchased by an intermediary for resale on the market, are therefore excluded from this analysis. Most commonly, PPAs are long term contracts that range from 10-15 years, according to [9] and [1]. This aligns with the PPAs signed by Google and Microsoft, both of which have a duration of 15 years [14].

Though fixed prices have many advantages, mainly stability and security guarantees, there are also downsides to having a fixed price contract. Firstly, energy suppliers often charge extra for these types of contract. Secondly, if market prices end up going below the fixed price, then the data center ends up paying above market value [7].

As PPA prices can vary significantly between countries and prices vary depending on the start date of the PPA, the electricity prices used in the simulation are derived from several sources [1] [15] [10]. Based on this data, the electricity price for the fixed price energy contracts is set at €90 per MWh due. This number is somewhat high due to the energy crisis during the data in the simulations.

### 3.1.2 Variable price energy contracts

In contrast to the stability of fixed-rate agreements, variable rate contracts tie electricity costs directly to market values, allowing for dynamic pricing based on real-time supply and demand conditions [7].

The primary advantage of variable pricing is the potential for cost reduction. Because suppliers do not need to build a risk premium into the rate to hedge against future market volatility, variable contracts often have lower initial costs compared to a PPA [7]. Furthermore, this structure allows data centers to capitalize immediately on favorable market conditions. For example, if wholesale prices drop due to excess supply or reduced demand, the data center's costs decrease right away [4].

However, this approach introduces significant exposure to market volatility. Prices fluctuate unpredictably due to weather events, grid constraints, or demand surges [7]. Consequently, success with variable pricing requires data centers to become active market participants [4].

We try to simulate this market volatility for variable energy prices. To this end, we take into account seasonal load demands, time-of-day load variations and occasional random spikes. In the Netherlands, higher energy demand for heating in winter leads to generally higher prices during that season. Moreover, energy usage during the day steadily climbs before peaking in the evening and declining at night. This pattern is reflected in the energy pricing. Random spikes are also simulated with low probability to capture short-term fluctuations. For a more detailed implementation, the code can be found in the demo GitHub under the `test-generator.py` file.

The comparison of both energy price contracts highlights the importance of analyzing fixed versus variable contracts, as shown later.

## 3.2 Degradation cost

In this model, each component is associated with a health value H(t) that evolves over time. The health is updated at every time step based on the degradation the component experienced during that timestep, denoted as $\Delta H(t)$. Thus, the health is updated over time based on equation 1.

$$H(t+1) = H(t) - \Delta H(t) \tag{1}$$

Component degradation is split into three contributing factors, each weighted by a corresponding coefficient and normalized where appropriate, as shown in Equation 2.

First, each component experiences a constant base degradation over time ($\beta_0$). Second, an additional degradation term $\beta_1$ accounts for the utilization of the component during the time step. Finally, a degradation term $\beta_2$ captures the effect of the power drawn by the component, normalized by the maximum power capacity $P_{max}$.

$$\Delta H(t) = \beta_0 + \beta_1 * \bar{u}(t) + \beta_2 \frac{P(t)}{P_{max}} \tag{2}$$

During the simulation, component state updates, referred to as fragments, are generated within each time step. To determine the utilization of a component over a single time step, the average utilization is computed across all fragments in that time step. For each component $m$, the average utilization $\bar{u}(t)$ is calculated by summing the utilization values of all fragments and dividing by the total number of fragments f, see equation 3.

$$\bar{u}(t) = \frac{1}{f} \sum_{m=1}^{60} componentUtilization(m) \tag{3}$$

A component is considered to have reached the end of its operational lifetime once its health falls below or equals a predefined critical threshold $H_{crit}$. When this condition is reached, the component is replaced and the replacement cost is added to the total costs.

$$H(t) \leq H_{crit} \tag{4}$$

### 3.3 Utilization costs

The utilization costs have been divided into monthly salary costs and general utilization costs. These costs should be given by the user. This enables the running of several scenarios in which these costs can be adjusted.

## 4 Experimental Results

This section discusses the experimental setup for model validation and the model's results.

### 4.1 Experimental Setup

To allow for comparisons, the evaluation consists of four experiment setups, each corresponding to a distinct cost scenario and workload/topology configuration:

1. **first_experiment**: Uses a standard topology with the cost revenue model enabled and default parameter values for energy prices and utilization costs.

2. **first_experiment_100**: This setup has the same topology and workload as the first experiment, but has a decreased amount of host of 100 instead of 200, representing a smaller-scale data center scenario and focusing on the impact of workload scale.

3. **cheap_experiment**: This setup has the same topology and workload as the first experiment, but has a decreased amount of host of 100 instead of 200, representing a smaller-scale data center scenario and focusing on the impact of workload scale. The difference with first_experiment_100 is that it was intended to use a low cost configuration, including lower prices, general utilization, and higher degradation costs, to simulate a cost-optimized data center scenario.

4. **expensive_experiment**: This holds the same topology and workload as the cheap experiment, but with a high-cost configuration using more expensive, higher performance components. These have a generally less quick wear down. This simulates a higher investment setup.

Each experiment produces five distinct files, running under hourly time-steps. All experiments export the following Parquet traces:

1. **costmodel.parquet**: financial outputs (energy, employee, general, degradation costs)

2. **host.parquet**: CPU/GPU resource usage, energy draw, uptime

3. **powerSource.parquet**: energy usage and carbon emissions

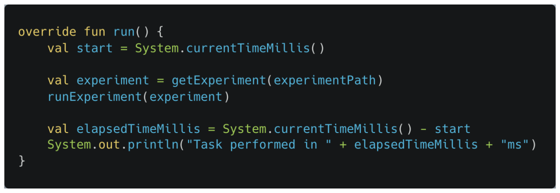
4. **service.parquet**: job orchestration state

5. **task.parquet**: execution performance, failures, scheduling delays

To understand the properties of the cost model added to OpenDC, both the costmodel and powerSource parquet files are important. These two parquet files give the costs of the data center over time.

To study both small and large-scale conditions, it is possible for the experiments to use two workload regimes where one has fewer hosts and a smaller bag-of-tasks workload to represent a modest-scale data center scenario and the other, which stresses scalability of both the simulator and the cost model. These workload regimes are represented in the host and service files.

To evaluate the performance impact of the cost model, we measured the end-to-end wall-clock execution time of each simulation run with and without enabling the model. Timing was collected inside the experiment driver by instrumenting the run() method in ExperimentCli, recording the elapsed time around the full experiment execution. Each configuration was executed multiple times and runtimes were averaged to reduce the influence of JVM warm-up and system noise.



```
override fun run() {
    val start = System.currentTimeMillis()

    val experiment = getExperiment(experimentPath)
    runExperiment(experiment)

    val elapsedTimeMillis = System.currentTimeMillis() - start
    System.out.println("Task performed in " + elapsedTimeMillis + "ms")
}
```

Figure 2: Code snippet for computing runtime

## 4.2 Experiments

### 4.2.1 Fixed versus variable pricing comparison

For the fixed pricing experiment, we evaluate the cost model using a constant energy price of €80 per MWh, representative of long-term corporate PPAs. Both the fixed and variable price simulations are performed with the topology and parameters as used in the first_experiment setup. This is done to isolate the effect of price stability against price variation. Figure 3 shows the aggregated total cost over time for both fixed and variable pricing, computed by summing energy, employee, general, and degradation costs at each simulation timestep.

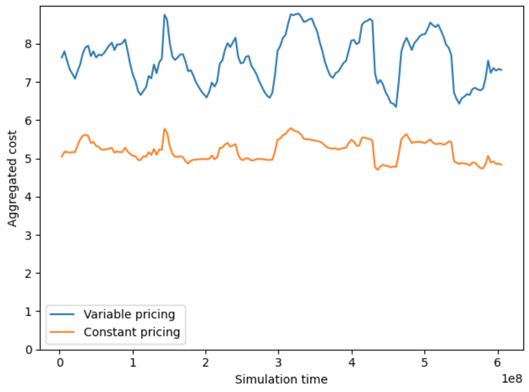


Figure 3: Cost comparison between constant and variable pricing

### 4.2.2 Experiments under variable energy prices

Figure 4 shows the cost breakdown over time, illustrating the variability introduced by market conditions. Figure 5 highlights the cost per task across experiments, and Figure 6 compares the total costs across the four experiment groups under variable pricing.

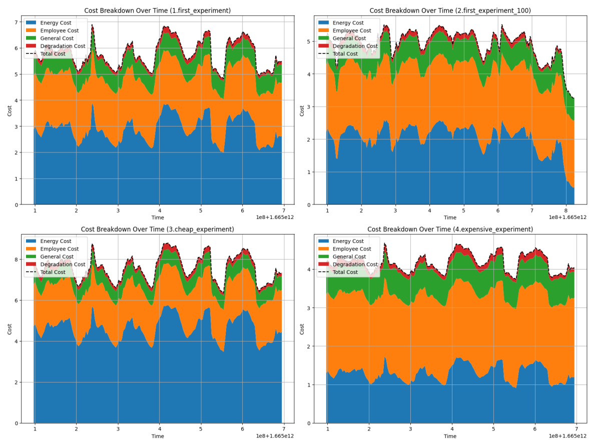


Figure 4: Cost breakdown over time with each experiment group.

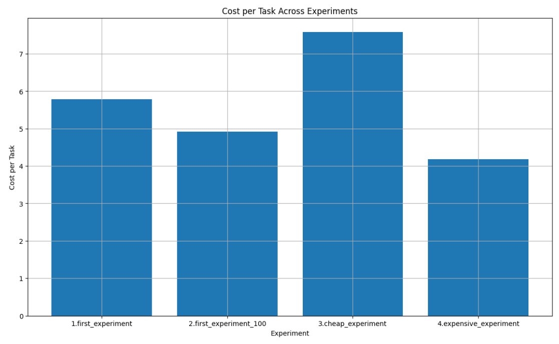


Figure 5: Cost per task across experiments.

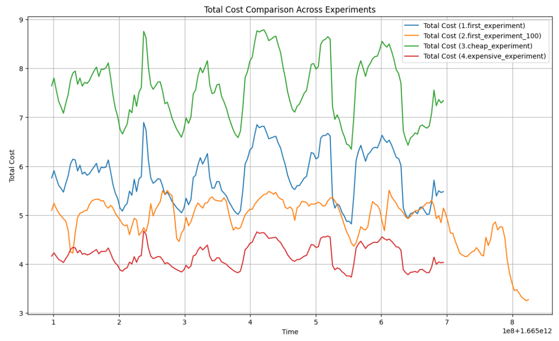


Figure 6: Total cost comparison across the four experiment groups.

### 4.2.3 Power Draw and Energy Efficiency

This subsection focuses on power draw and energy efficiency. Figure 7 shows the power draw over time for each experiment group, while Figure 8 compares energy efficiency across the four experiment groups, by measuring the average energy needed per task performed. These figures help illustrate how different workloads and pricing models impact power consumption and efficiency.

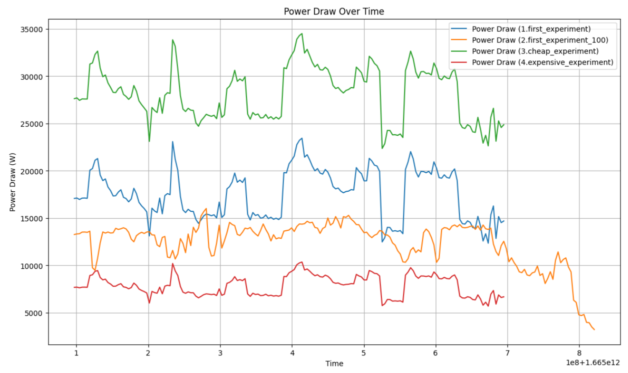


Figure 7: The power draw over time.

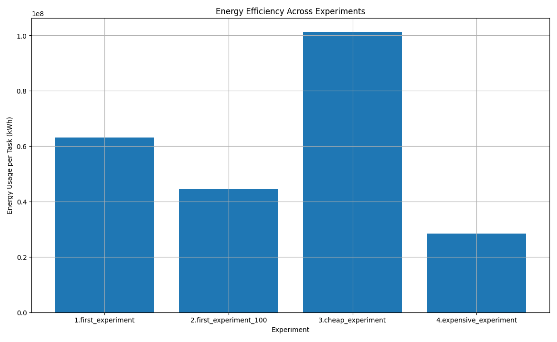


Figure 8: Energy efficiency across the four experiment groups.

### 4.2.4 Cost Distributions

In this section, we examine the distribution of specific cost components. Figure 9 shows the distribution of energy costs across experiments, and Figure 10 illustrates the distribution of degradation costs. These distributions provide insight into which costs are most significant and how they vary across experiments.

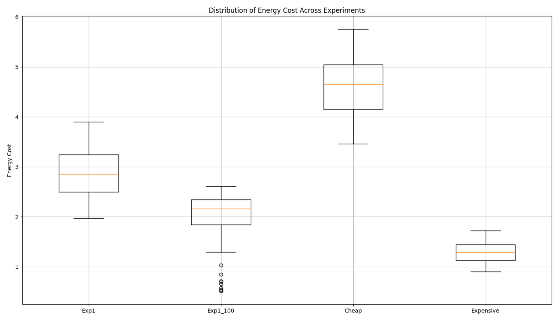


Figure 9: The distribution of energy cost across experiments.

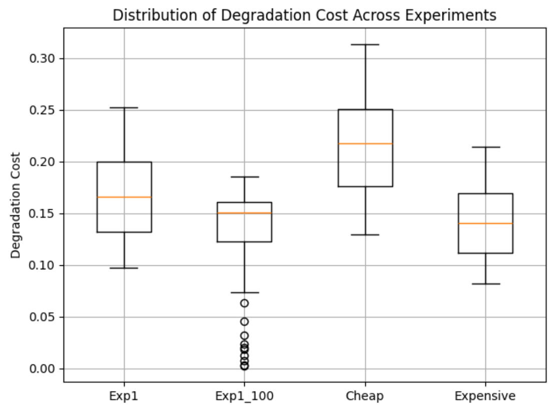


Figure 10: The distribution of degradation cost across experiments.

## 4.3 Performance results

| Run ID | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| NC(ms) | 2376 | 2275 | 2105 | 2220 | 2287 | 2261 | 2102 | 2278 | 2115 | 2216 |
| C(ms) | 2493 | 2377 | 2213 | 2257 | 2255 | 2218 | 2273 | 2285 | 2223 | 2243 |

Table 1: Runtimes of the 100-host configuration. NC are the runs without the cost model, C are the runs with the cost model

For the 100-host configuration, we can see the results in table 1. We calculate that the baseline execution time without the cost model averaged 2224 ms, while enabling the cost model resulted in an average runtime of 2288 ms, an increase of 2.87%.

| Run ID | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| NC(ms) | 2556 | 2480 | 2414 | 2553 | 2479 | 2492 | 2435 | 2427 | 2466 | 2533 |
| C(ms) | 2757 | 2659 | 2562 | 2480 | 2506 | 2471 | 2482 | 2488 | 2792 | 2489 |

Table 2: Runtimes of the 500-host configuration. NC are the runs without the cost model, C are the runs with the cost model

For the 500-host configuration, we can see the results of the runtimes in table 2. We calculate that the baseline averaged 2484 ms, compared to 2551 ms with the cost model enabled, an increase of 2.69%.

## 5 Discussion

This section discusses the results and their contextual meaning.

Firstly, it should be noted that some of the variables used are assumed values that reflect what we considered an acceptable value, i.e. a value that made theoretical sense and is inspired by, but not equal to, real world values. For example, the utilization costs vary greatly in reality and are thus somewhat simplified. Although it is quite difficult to have a simulation match reality, these assumed costs are included for functionality purposes and illustrate the model's proof of concept.

In this study, the aim is to analyze the costs of different size data centers, with experimental setup 1 using 200 hosts and setups 2, 3 and 4 using 100 hosts. Several cost parameter set ups are compared between setups 2, 3 and 4. Due to limitations of results that can be discussed in this study the different cost parameter setups are not also compared in a larger data center setup. This configuration of cost parameter setup and data center size setup could be analyzed in future studies.

The fixed pricing setup results in a smoother and more predictable total cost over time compared to variable pricing. While both configurations follow the same workload patterns, fixed pricing removes short-term fluctuations caused by energy market dynamics, making cost changes primarily driven by workload intensity and power consumption. In contrast, variable pricing introduces higher variability even when resource usage remains similar, showing that the choice of energy contract mainly affects cost stability rather than overall consumption.

Comparison of constant and variable pricing highlights the trade-off introduced by energy contracts. Variable pricing allows costs to decrease during favorable market conditions, but introduces higher short-term variability, while constant pricing offers stability at the expense of flexibility. These results illustrate why evaluating both pricing schemes is important, as they lead to different cost profiles even under identical workloads and infrastructure.

The cost break down for the different data center sizes, between experimental setup 1 and 2, show similar cost breakdown, however, as seen in figure 6 the total cost is lower for the smaller data center. With a smaller data center the degradation costs and energy cost scale down as well. The utilization costs are kept the same in both situations, thus utilization becomes a larger percentage of the total cost breakdown. As these costs are not scaled down, the total cost does not decrease by 50% like the data center size does. It should be noted that utilization costs do not decrease in the simulation as this is a set input value. Future research could look into implementing a variable set up for utilization costs to allow it to vary with the data center size.

Between the different cost parameter setups, figure 4 shows that the cheap experiment setup has a higher energy cost percentage than the expensive setup. This can be explained due to the fact that the expensive hardware setup is more efficient than the cheaper hardware and thus requiring less power to accomplish the same tasks, see figure 9. This efficiency is also visible in figure 5, in which the cheaper setup has a high cost per task and the expensive setup has a lower cost per task.

Figures 7 and 8 validate what we find in previous experiments. The power draw closely matches the increases and decreases in costs, which makes sense as energy consumption forms a large part of total costs as previously discussed. Moreover, we can see that when comparing setup 1 and 2, the smaller topology of setup 2 causes lower energy draw whilst having to run longer, which follows expectations. Additionally, when comparing the setup 3 and 4, we can see that the more energy efficient setup 4 (which can also be seen in 8) yields much lower power draw, and thus lower costs.

As for the degradation cost distributions in figure 10, we can read from the graph that the largest cost would be incurred when using cheap equipment. This is somewhat contradictory, as we expected the tradeoff for the poor performance and energy efficiency of cheap equipment to be balanced by its lower costs. This however, does not seem to be the case. One possible explanation could be that the cheaper hardware has a lower health than the expensive one, as such it would require more frequent replacements.

In terms of performance, the experimental results show that adding the cost model does not significantly affect the runtime of OpenDC. The measured end-to-end execution times with the cost model enabled remain comparable to the baseline runs without the model for both the 100-host and 500-host configurations. These findings confirm that Requirement R5 is satisfied, as the runtime impact of the cost model remains well within the defined performance budget.

Overall, these figures show that Requirements R1, R2, R3 and R4 and R6 are satisfied. We provide the correct visualization and calculate values that are feasible. Furthermore, we show that the performance and calculations do not suffer under an increase in workload and topology.

### 5.1 Future works

Although, the previous section already discusses some extensions that can be done in future studies to the experiments performed, these are specific improvements for variable implementation or extending cross experimental setup comparisons. This next section will describe overall improvements to the cost revenue model which could be researched in future studies.

Due to time and scope constrains, this research was limited and can be improved through future studies. One aspect of this study that can be improved is the component replacement conditions. In our study this has been implemented with a cut off value, however, this can also be implemented with a stochastic method. With such a method the component has a chance to fail once the components health crosses a threshold, thus simulating the randomness of hardware failures in data centers. Furthermore, OpenDC and our cost revenue model can be extended with high resolution analytics that enable analysis at the level of individual nodes or clusters within the data center. This would allow costs and revenues to be attributed to specific infrastructure components, providing clearer insight into where economic and operational effects originate. These targeted analysis could also improve other existing models and shed light on other metrics beside costs, such as performance. Overall, this would allow the cost revenue model to be a broader support tool beyond aggregate cost modeling.

## 6 Conclusion

In this work, we successfully extended the OpenDC simulator with a comprehensive cost revenue model, addressing the gap in open-source tools for analyzing the financial implications of data center design. By integrating a Total Cost of Ownership (TCO) framework, we transformed OpenDC into a unified environment where researchers and students can evaluate the interplay between workload performance, energy sustainability, and economic viability.

Our system decomposes costs into three functional categories: energy, utilization, and degradation. A key contribution of this model is the use of the Dutch energy market in simulations, which allows users to compare the stability of fixed-price Power Purchase Agreements (PPAs) against the volatility of variable-rate contracts. Furthermore, we implemented a dynamic degradation mechanism where component health decays based on base rates, utilization, and power draw, triggering replacement costs when a critical threshold is reached.

The experimental results confirm that the model satisfies the requirements defined at the beginning. The system successfully calculates and visualizes total energy, maintenance, and operational costs, scaling effectively across different network topologies. Notably, the experiments revealed that low-cost hardware configurations can result in higher long-term costs due to accelerated degradation and frequent replacements.

The integration does not harm the runtime of OpenDC much, with experiments on 100-host and 500-host configurations demonstrating a runtime overhead of less than 3% compared to the baseline OpenDC simulation.

While the current model relies on estimated values for utilization costs to demonstrate functionality, it establishes a solid foundation for financial simulation. Future work should focus on replacing fixed component failure thresholds with stochastic methods to better simulate the randomness of hardware failures. Additionally, the model can be extended with high-resolution analytics to attribute costs to individual nodes or clusters, providing deeper insights into the economic impact of specific infrastructure components. Ultimately, this extension aids the OpenDC community to make economically informed decisions in data center management.

## References

[1] Moutaz Altaghlibi. Ppas…a key piece in the transition puzzle. Technical report, ABN AMRO, ESG Economist Group Economics, 2024. Accessed: 2025-12-15.

[2] Doaa Bliedy, Sherif Mazen, and Ehab Ezzat. Datacentre total cost of ownership (tco) models: A survey. *International Journal of Computer Science, Engineering and Applications (IJCSEA)*, 8(2/3/4):47–60, 2018.

[3] Yan Cui, Charles Ingalz, Ty G., and Ali Heydari. Total cost of ownership model for data center technology evaluation. pages 936–942, 05 2017.

[4] Data Center Asia. Understanding data center energy spot trading, 2025. Conference promotional material.

[5] EkkoSense. Ekkosim. *Data center simulation tool.*

[6] RTDS Technologies Inc. Rtds simulator. *Data center simulation tool.*

[7] Kb3 Advisors. Evaluating fixed vs. variable rate energy contracts, November 2024. Accessed via Kb3 Advisors Resources.

[8] Jonathan Koomey, Pitt Turner, John Stanley, and Bruce Taylor. A simple model for determining true total cost of ownership for data centers. 01 2007.

[9] Next Kraftwerke. Wat is een power purchase agreement (ppa)? [what is a power purchase agreement (ppa)?]. https://www.next-kraftwerke.nl/kennis/power-purchase-agreement-ppa, n.d. Accessed: 2025-12-02.

[10] Emilia Lardizabal. Ppa prices in europe dropped 4.3% in february, with portugal seeing the steepest decline, 2025. Accessed: 2025-12-16.

[11] Fabian Mastenbroek, Georgios Andreadis, Soufiane Jounaid, Wenchen Lai, Jacob Burley, Jaro Bosch, Erwin van Eyk, Laurens Versluis, Vincent van Beek, and Alexandru Iosup. Opendc 2.0: Convenient modeling and simulation of emerging technologies in cloud datacenters. 2021.

[12] NVIDIA. Nvidia air. *Data center simulation tool.*

[13] Chandrakant Patel and Amip Shah. Cost model for planning, development and operation of a data center. 01 2005.

[14] Rabobank Nederland. Datacenters kunnen het nederlandse energiesysteem belasten én versterken [data centers can tax and fortify the dutch energysystem], 2025. Accessed: 2025-12-02.

[15] Ryan Rudman. European ppa market in 2024, insights amid price drops, 2024. Accessed: 2025-12-16.

[16] Snehanshu Saha, Jyotirmoy Sarkar, Avantika Dwivedi, Nandita Dwivedi, Anand M. Narasimhamurthy, and Ranjan Roy. A novel revenue optimization model to address the operation and maintenance cost of a data center. *Journal of Cloud Computing*, 5(1):1, December 2016.

[17] Gambhire Swati Sampatrao, Sudeepa Roy Dey, Bidisha Goswami, Sai Prasanna M. S, and Snehanshu Saha. A Study of Revenue Cost Dynamics in Large Data Centers: A Factorial Design Approach. September 2016. arXiv:1610.00024.

[18] Vattenfall Nederland. Het verschil tussen energie kleinverbruik en grootverbruik uitgelegd [the difference between small and large energy consumers explained]. https://www.vattenfall.nl/grootzakelijk/zakelijke-energie/grootverbruik-kleinverbruik/, 2025. Accessed: 2025-12-21.

## Appendix A

| Name | think | dev | experiment | analysis | write | wasted | total |
|---|---|---|---|---|---|---|---|
| Aayush | 19 | 38 | 14 | 5 | 8 | 18 | 102 |
| Nico | 16 | 56 | 4 | 4 | 0 | 20 | 95 |
| Daniel | 20 | 1 | 1 | 2 | 44 | 16 | 80 |
| Thomas | 19 | 1 | 1 | 2 | 44 | 18 | 79 |
| Jordy | 16 | 1 | 4 | 6 | 4 | 30 | 61 |
| Mark | 8 | 1 | 8 | 6 | 3 | 24 | 50 |

Table 3: timesheet