Energy-Aware Load Balancing in Distributed Systems Using Machine Learning

Dr Ajay Pratap[1], Dr. Shashi Kant Gupta[2], Prof (Dr) Midhunchakkaravarthy[3], Dr Shantanu Shahi[4]

[1] Post Doctorate Fellow, Lincoln University College, Kota Bharu, Malaysia; [2]Adjunct Research Faculty, Lincoln University College, Malaysia; [3] Lincoln University College, Malaysia; [4] Lincoln University College, Malaysia;

Email ID: pdf.ajay@lincoln.edu.my, raj2008enator@gmail.com, midhun@lincoln.edu.my, pdf.shantanu@lincoln.edu.my

---

Abstract: As partitioned or distributed systems are an increasing part of today's computing environment, the issue of energy control in addition to performance is receiving special attention. Conventional load balancing methods may merely balance loads by the service demand of CPU or memory without considering energy saving. This means higher operational cost and more of an environmental impact, especially in larger data centers. The focus of this paper is the energy-efficient load balancing based on machine learning to scale-out distributed systems. A machine learning model is learned from a simulated server workload data (which include CPU load, power consumption, and response time) to estimate the best-fit server type for an incoming job. The system works to minimize energy consumption through intelligent workload routing, without degrading its performance. A web application is created to illustrate the idea. The system can simulate task inputs, monitor server loads, and see how the ML model allocates tasks in real-time. The interface also automatically visualizes energy savings and efficiency gains compared with other traditional approaches such as round-robin allocation. This research emphasizes the reliance of ML in creating sustainable and intelligent systems encouraging green computing. Future work can extend this to real time cloud environments, add renewable energy metrics or try out more advanced ML models for further optimization.

Keywords: Distributed Systems, Load Balancing, Energy Efficiency, Green Computing, Power Consumption, Performance Optimization

Introduction

In today's IT systems, distributed setups help run cloud services and apps that use a lot of data, but they also use more and more energy. Regular load balancing helps make things run faster by dividing tasks among servers, but it often doesn't care about how much energy is used, which can lead to higher costs and more harm to the environment. As being eco-friendly becomes more important, machine learning provides a better way. By looking at past data and checking things like how much the CPU is used, how much power is consumed, and how fast the system responds, machine learning can figure out which server uses the least energy for each task. This project uses an ML-based method for load balancing and shows it through a web app to prove that energy use can go down without hurting performance.

As systems with many computers become more complicated, it's harder to manage how much work each computer has to do. Old ways of balancing the work, like round-robin or least-connections, focus on making sure computers don't get too busy or use too much memory. But they don't take into account how much power the computers are using. This can cause higher energy bills, more costs to run the

system, and more harm to the environment, especially in big data centers. To fix this, we need smarter ways to balance the workload that also care about using less energy. This project suggests using machine learning to look at how much work each computer is doing and then send tasks to the ones that use the least energy. A web tool shows that this method uses less energy and works better than older methods. Load balancing is the way of spreading incoming traffic or work across several servers so that no one server gets too busy. This helps the system work better, makes responses faster, and keeps the system up and running reliably. There are different ways to do load balancing, like round-robin, least connections, or methods that look at how much CPU or memory each server is using. This helps keep things stable when there's a lot of activity and uses resources efficiently. In today's systems, load balancers might also take into account things like energy use or where users are located, which is important for building systems that can grow and stay strong.

The main focus of this project is on utilizing machine learning to optimize load balancing in distributed systems for energy efficiency. Given the increasing concern for energy consumption in large-scale computing environments, this project is committed to developing a system that balances server loads efficiently while using minimal power and maintaining optimal performance. This project is designed with the following objectives:

1. To assess the significance of server metrics like CPU usage, response time, and power consumption in determining optimal strategies.
2. To create an online simulation of task distribution and server activity in a distributed system.
3. To design a machine learning-based model that can anticipate the most energy-saving server for managing incoming tasks.
4. To evaluate the efficiency and performance of the ML-based technique in comparison to conventional load balancing methods such as round-robin or random selection.

Related work

The paper presents a framework for energy-efficient load balancing in smart grids. It uses cloud and fog computing. The framework tackles issues with latency and resource allocation by distributing tasks dynamically. Results indicate better performance and energy savings [1]. The paper proposes an energy-aware load balancer for virtual machine placement that uses a fog-based classifier. It focuses on energy efficiency and resource management in cloud-fog environments [2].The paper presents energy-saving load balancing policies for cloud systems. It focuses on reducing energy consumption while keeping performance levels high [3]. The paper presents an imitation-based optimization method for managing resources efficiently in data centers. It aims to cut down energy consumption while keeping service quality intact through smart task scheduling [4]. The paper presents a look-ahead energy-efficient virtual machine (VM) allocation method for data centers. It predicts future resource needs to optimize VM placement and cut down on energy use [5]. The paper evaluates energy-efficient algorithms for virtual machine (VM) consolidation in cloud computing. It compares different techniques that reduce energy use while keeping service quality intact [6]. The paper presents a hybrid resource allocation method for cloud infrastructure that combines Genetic Algorithm and Random Forest. The goal is to improve decision-making for efficient and balanced resource use [7]. The paper explores software techniques to improve energy efficiency in cloud data centers. It discusses workload scheduling, virtualization, and resource management strategies. The review identifies important approaches, challenges, and research gaps in sustainable cloud computing

[8]. The paper looks at job scheduling behavior with the Google Cluster dataset. It focuses on key factors that influence scheduling. It studies how things like resource demand, job duration, and priority impact scheduling efficiency [9]. The paper reviews energy management strategies in cloud data centers. It emphasizes machine learning techniques. It looks at ML-based workload prediction and dynamic resource allocation to improve energy efficiency [10]. The paper uses the Random Forest method to predict energy consumption in buildings with high accuracy. It tackles the challenge of modeling complex energy usage patterns based on historical data [11]. The paper presents a load balancing strategy for cloud environments that is both sustainable and mindful of costs and energy use. It aims to reduce energy consumption and operational costs while maintaining performance efficiency [12]. The paper presents an AI-driven load balancing method to improve energy efficiency in data centers. It uses smart decision-making to optimize resource allocation and cut power consumption [13]. The paper presents a strategy for balancing load in Content Delivery Networks (CDNs) that considers energy use. It adjusts server usage based on demand to lower energy consumption while maintaining performance [14]. The paper looks at recent improvements in energy-efficient machine learning models in different computing environments. It focuses on methods for lowering energy usage while keeping model accuracy and efficiency high [15].

Research Methodology

Distributed systems are used to run big applications, and they need tasks spread out efficiently to work well and stay reliable. Usually, load balancing doesn't take energy use into account, which makes things less efficient and more expensive. This project introduces a new way to balance workloads that considers energy use, using machine learning. A Decision Tree model predicts how much energy each server will use based on factors like CPU usage, memory, task load, and energy cost. Tasks are then given to the server that is expected to use the least amount of energy. The system is built as a web app with a Python FastAPI backend and an HTML/CSS/JavaScript frontend, helping make computing more sustainable by managing workloads in a smarter and more energy-efficient way.

The proposed solution introduces a machine learning-based load balancing system that aims to reduce energy consumption in distributed environments. Instead of distributing tasks based only on CPU or memory load, this system prioritizes energy efficiency during task allocation. The system gathers real-time input from users or monitoring tools, including CPU usage, memory usage, task load, and energy rate for each available server. This data goes to a pre-trained Decision Tree regression model, which predicts the expected energy consumption of each server for the incoming task. The backend, built with FastAPI and Python, processes the input and uses the model to find the most energy-efficient server. The frontend, created using HTML, CSS, and JavaScript, enables users to submit server information and see the recommended allocation. This smart approach lowers energy usage while keeping system performance stable, making it suitable for green and scalable distributed systems.

The project's design brings together user interaction, machine learning, and database work in a smooth way. Users input task needs, like how much load they want, and server details, such as CPU usage, memory, and energy use, through a web interface. This information is sent to a backend system using HTTP POST requests. The backend uses a trained machine learning model, which is part of a pipeline that includes scaling data, to predict how much energy each server will use. It then selects the server that is expected to use the least amount of energy. This choice, along with all the related data, is stored in a MySQL database using SQLAlchemy, making sure the information is kept safely. There's also an endpoint

called /predictions that lets users see all the stored energy use predictions. The whole system runs on a local server using Uvicorn, and it has middleware to help the frontend and backend talk to each other. This setup is built in a way that makes it easy.

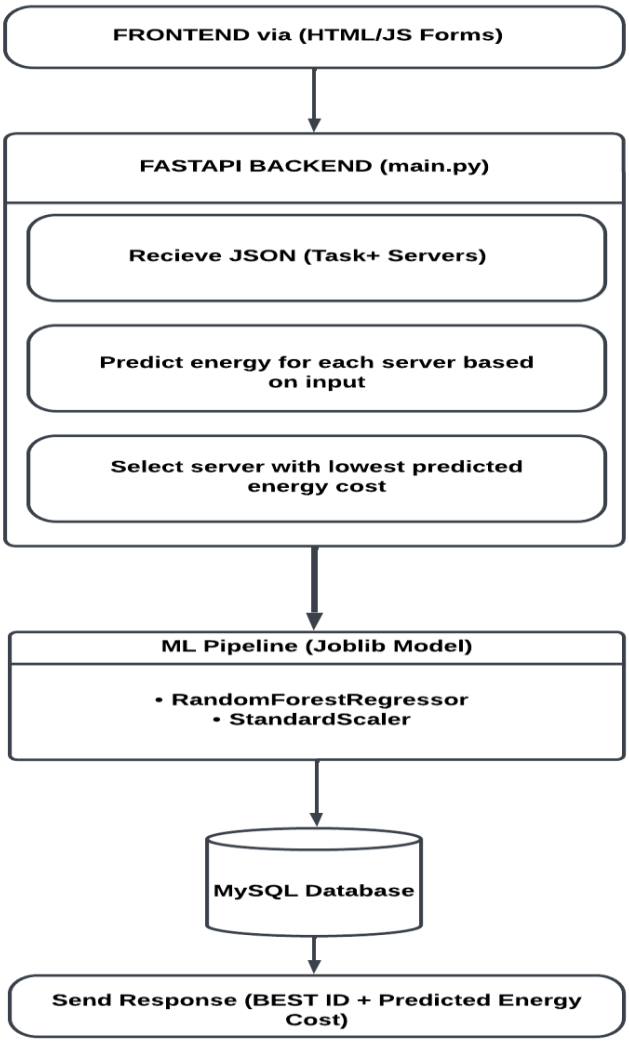

Figure 1. Architectural diagram for Proposed Model

As shown in Figure 1, the system has four main components. Frontend is a simple web interface where users enter task load and server metrics. Backend (FastAPI) receives this data, processes it, and interacts with the trained machine learning model. The ML Model predicts energy consumption for each server based on data from Google cluster usage. Using these predictions, the backend chooses the most energy-efficient server. The Database (MySQL) records each prediction, keeping server metrics, task load, and predicted energy. These components work together to create an intelligent energy-aware task allocator that supports real-time decisions and tracks historical data.

The functional requirements outline the specific behaviors and functions that the system must perform is shown in Table 1. The system is built to collect server metrics, predict energy costs using a machine learning model, and assign tasks to the most energy-efficient server.

Table 1. Functional Requirement

SN	REQUIREMENT	DESCRIPTION
1	User Input Form	Let users enter server metrics such as CPU usage, memory, energy rate, and task load through a web interface.
2	Prediction API	Sends the input data to a FastAPI backend, where the ML model predicts the energy cost for each server.
3	Best Server Selection	Compares predicted energy costs and picks the server with the lowest cost for task allocation.
4	Data Logging	Stores input parameters and prediction results in a MySQL database for record-keeping and future analysis.
5	Prediction History Retrieval	Offers an API endpoint (/predictions) to access all previously stored predictions for review or reporting.
6	Error Handling and Validations	Validates input data properly and returns appropriate error messages for invalid or missing values.

Non-functional requirements define the system's performance characteristics, constraints, and quality attributes as shown in Table 2. These do not directly affect what the system does; they determine how well it performs its functions.

Table 2. Non-Functional Requirement

SN	REQUIREMENT	DESCRIPTION
1	Performance	The prediction API needs to respond in 1 to 2 seconds under normal usage to stay fast and responsive.
2	Scalability	The system should handle more servers and more requests without much effect on its speed.
3	Reliability	The backend must keep working well and give correct predictions all the time, even when it's running continuously.
4	Maintainability	The code is organized into parts and has explanations, making it easier to update or connect with new machine learning models.
5	Security	There is basic cleaning of inputs and protection against cross-origin requests.
6	Portability	The app can work on systems that support Python, FastAPI, and MySQL with only small changes to setup.
7	Usability	The user interface should be easy to use and simple, so people can send tasks without needing technical skills.

Design of Machine Learning Model

This project uses Google Cluster Data, which is a big set of information containing more than 100 million task events. This data includes details about how much CPU is used, how much memory is used, what kind of machines are involved, and when tasks are scheduled. To make it easier to work with, a smaller part of this data is chosen. This smaller part looks at a specific time period and a limited number of machines. This selection includes important scheduling events and how resources are used, which is great for creating models that focus on energy efficiency and load balancing in distributed systems. Tasks that are short, failed, or not completed are removed to make the data cleaner and simpler for training models. By using real data from actual clusters, the project builds a realistic base for predicting how efficient servers are in terms of energy use and how tasks are assigned in large systems.

To get the dataset ready for training, we adjust the features to match real-world server units. CPU and memory numbers are multiplied by 1000 to show actual usage in cores or megabytes. The energy rate is scaled by 10 to give an estimate of wattage. Task load is also adjusted to reflect real-world work demands. This scaling helps the model work better and act like it would in a real environment. After scaling, the features go through a machine learning process that uses StandardScaler to make sure all data is in a similar range. This makes the model more accurate and stops it from favoring larger numbers as shown in Figure 2..

This project includes a group of features that show what's happening on each server right now and how well it's working:

(1) CPU Utilization (%), shows how much the server is being used. If it's low, the server isn't busy. If it's high, the server might be too busy.

(2) Memory Usage (MB) shows how much memory the server is using. It helps find if there's not enough free memory for new tasks.

(3) Energy Rate is an estimate of how much energy the server is using. It's based on CPU and memory use. A lower number means the server uses less energy.

(4) Task Load measures how much work the server is handling. It helps spread tasks evenly across servers so no one server gets too much, and performance stays good.

(5) Energy Cost is the final cost of running the server, based on energy rate and task load. This is a key target for optimization. The ML model is trained to suggest servers with the lowest energy cost based on the current load.

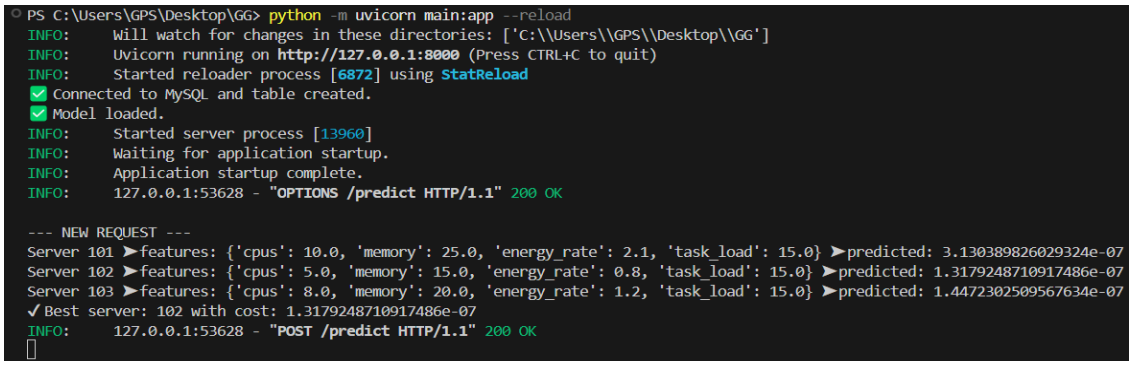

Figure 2. Terminal showing Execution

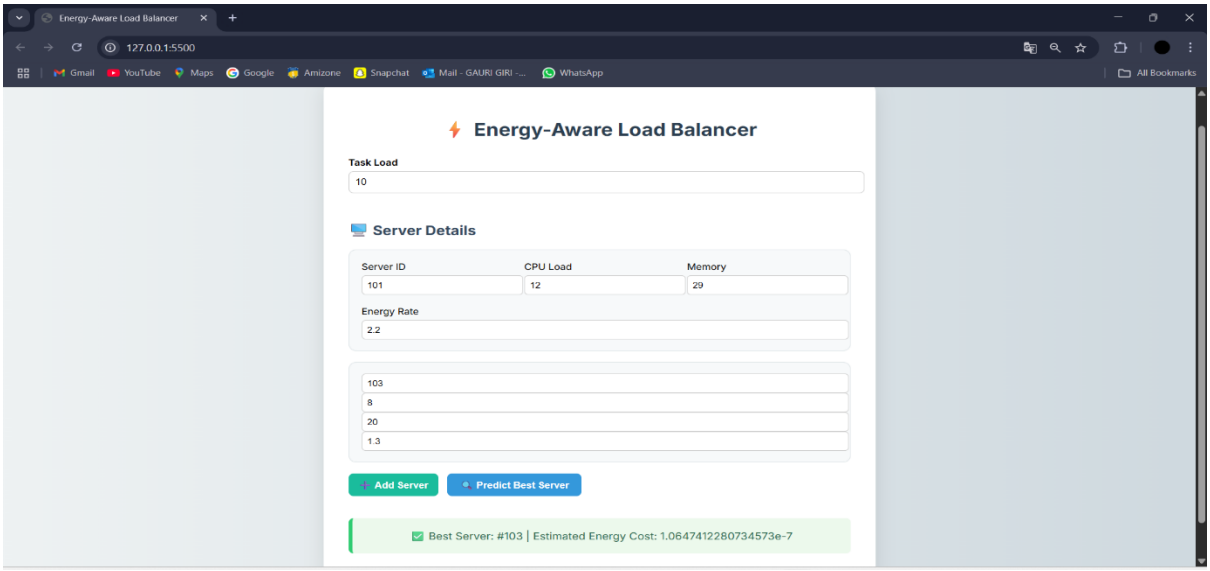

Figure 3. Server Input Form

A scikit-learn pipeline is used for the model. It includes a StandardScaler and a RandomForestRegressor. The StandardScaler makes sure all input features are on the same scale and have a consistent distribution. This helps with convergence and improves model performance. The RandomForestRegressor has 100 trees and is chosen for its strength and capacity to manage non-linear relationships among features. This ensemble method helps reduce overfitting, gives insights into feature importance, and works well without needing much tuning of hyperparameters. The model predicts energy costs based on server metrics and supports energy-efficient decision-making in load balancing situations as shown in Figure 3.

The model is trained on 80% of the dataset and tested on the remaining 20%. After training, we evaluate predictions from the test set using three metrics: R^2 Score, Mean Absolute Error (MAE), and Mean Squared Error (MSE). These metrics help us measure the model's accuracy and ability to generalize. For instance, a high R^2 shows that the model explains most of the variance in energy costs. A scatter plot showing actual values compared to predicted values visually demonstrates how accurate the model is. Once we finish the evaluation, we save the entire pipeline, including scaling and the model, using pickle as energy_model.pkl. This makes it ready for use in the backend system.

The backend uses FastAPI. It offers a fast, asynchronous REST API for server-side logic. It loads a trained machine learning model called energy_model.pkl. The system has a /predict POST endpoint that accepts server data and task load. The model predicts the energy cost for each server, chooses the most efficient one, and returns that information. It also stores predictions in a MySQL database with SQLAlchemy. The /predictions GET endpoint lets users retrieve all stored results. CORS is enabled to allow frontend requests. The backend is strong, modular, and connects prediction with persistent storage and real-time API access.

The front end interacts with the backend through RESTful API calls. Users provide task load and server details, including CPU, memory, and energy rate. This information is sent to the /predict endpoint in JSON format. The backend replies with the most energy-efficient server ID and its estimated energy cost. The frontend can then show this recommendation in a clear interface. It can also call /predictions to get past

prediction history for analysis or visualization. This easy interaction connects the technical backend model with practical usability, enabling informed task allocation in a distributed system.

Result and Analysis

To test the model, submitted different server configurations with task loads. Each request included values for CPU, memory, and energy rate. For example:

- Input Example:

  Task load: 15,
  Server 1: CPUs = 12, Memory = 25 GB, Energy Rate = 2.1
  Server 2: CPUs = 6, Memory = 15 GB, Energy Rate = 1.2
- Predicted Costs:

  Server 1 = 3.59e-07
  Server 2 = 1.32e-07
- Selected Server: Server 2 (lowest energy cost)

These cases confirm that the system accurately predicts energy costs and picks the best server.

Using matplotlib and seaborn, following visual insights has been generated:

- Feature Importance: The Random Forest model ranked features as follows: energy_rate, task_load, cpus, memory. This shows the model prioritizes cost-related metrics as shown in Figure 4.

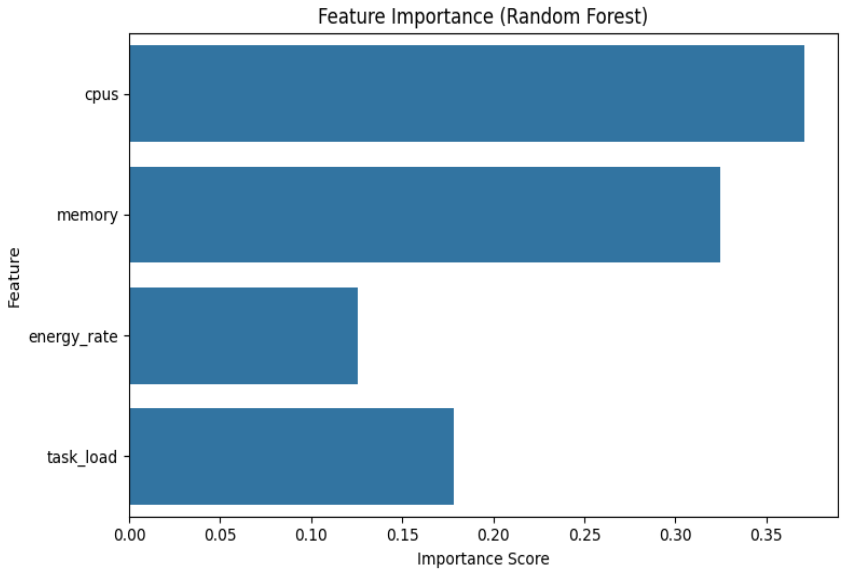

Figure 4: Feature importance graph

- Predicted vs. Actual Energy Cost: A scatter plot revealed close clustering along the y=x line. This indicates accurate predictions as shown in Figure 5.

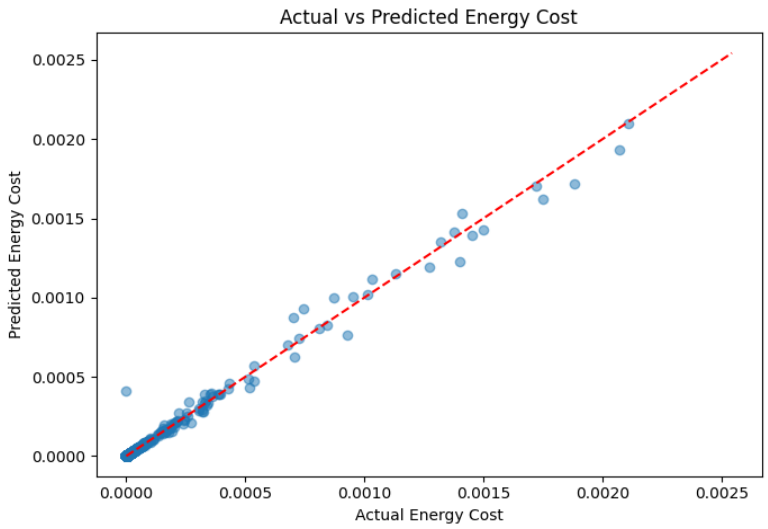

Figure 5: Predicted Energy Cost Vs Actual Energy Cost Graph

- Distribution of Energy Costs: Most predictions were near 0, as expected, with a long tail for higher-cost configurations as shown in Figure 6.

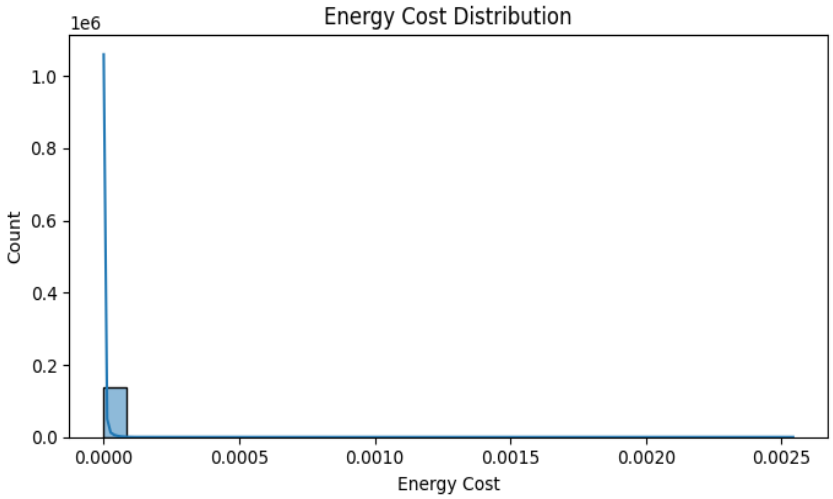

Figure 6: Energy Cost Distribution Graph

These visualizations help explain the model's behavior and confirm that no major bias exists. The system was tested using the same task loads on different servers. Servers with balanced resources and lower energy rates consistently had lower costs. The model does not just choose the most powerful server; it balances efficiency and cost by mimicking smart load balancing behavior.

Limitations

While the proposed system successfully predicts energy-efficient task allocation, there are still some practical and technical limitations due to the project's scope and level of implementation. It's important to acknowledge these constraints for future improvements and real-world use:

- Manual Inputs: Users must manually enter CPU, memory, and energy rates, which limits real-time automation.
- Simplified Model Scope: The current model does not consider network bandwidth, I/O usage, or interference from concurrent tasks.
- Static Dataset: The system is trained on a fixed dataset. Without regular retraining, predictions may decline over time.
- No User Authentication: It lacks security features such as login, which are necessary in production environments.

Future Scope

The current project shows that using machine learning to optimize server selection based on predicted energy consumption is possible. However, there are several ways to improve its functions and move it closer to real-world use:

- Real-Time Metrics Integration: Instead of manually entering CPU and memory values, we can use system APIs, like psutil, Prometheus, or cloud SDKs, to automatically fetch live server metrics.
- Cloud Infrastructure Support: Expand the model to work with AWS, Azure, or GCP environments where dynamic resource allocation and billing information are available in real time.
- Reinforcement Learning for Adaptivity: Train models that learn from system feedback, such as performance after task placement, to continuously improve predictions.
- Task Scheduling Over Time: Add scheduling logic that selects not just which server to use but also when to run a task for better energy efficiency.
- Model Comparisons: Evaluate other algorithms, like XGBoost, SVR, or Neural Networks, to compare performance improvements.
- Security and Access Control: Incorporate user authentication and role-based access to prevent unauthorized access to API endpoints or server details.
- Admin Dashboard for Monitoring: Create a visualization panel to track predictions, server usage trends, and energy savings over time.

These improvements would make the system more independent, scalable, and usable in real-world data centers.

Conclusion

This research paper offers a machine learning solution for energy-efficient task allocation in cloud environments. It uses server metrics like CPU usage, memory availability, energy consumption rates, and task load to estimate the energy cost of running a task on different servers. We used a Random Forest Regressor, trained on Google Cluster data, for predictions because of its reliability and precision. The system was built with FastAPI for backend services and connected to a MySQL database to log predictions for future analysis. The solution shows how smart resource scheduling can lower energy use and cut operational costs in data centers. While it currently requires manual input, the project sets a solid

groundwork for real-time monitoring and automation. This implementation demonstrates that data-driven load balancing is possible and paves the way for incorporating real-world sensors, auto-scaling, and adaptive learning in future versions.

References

1. G. Srivastava, S. Athithan, R. Kumar, *et al.*, "Energy Aware Load Balancing Framework for Smart Grid Using Cloud and Fog Computing," *Sensors*, vol. 23, no. 7, p. 3488, 2023. [Online]. Available: https://doi.org/10.3390/s23073488

2. A. Pratap, S. K. Gupta, and Midhunchakkaravarthy, "Analysis of Energy Consumption Pattern in Distributed Big Data Processing", SGSES, vol. 1, no. 1, May 2025.

3. A. Paya and D. C. Marinescu, "Energy-Aware Load Balancing Policies for the Cloud Ecosystem," *arXiv preprint*, arXiv:1401.2198, 2014. [Online]. Available: https://doi.org/10.48550/arXiv.1401.2198

4. G. Srivastava, *et al.*, "Energy Efficient Resource Management in Data Centers Using Imitation-Based Optimization," *Energy Informatics*, vol. 7, no. 1, p. 2024, 2023. [Online]. Available: https://doi.org/10.1186/s42162-024-00370-y

5. İ. Çağlar and D. T. Altılar, "Look ahead energy efficient VM allocation approach for data centers," *Journal of Cloud Computing*, vol. 11, no. 1, p. 11, 2022. [Online]. Available: https://doi.org/10.1186/s13677-022-00281-x

6. Q. Zhou, M. Xu, S. S. Gill, C. Gao, W. Tian, C. Xu, and R. Buyya, "Energy Efficient Algorithms based on VM Consolidation for Cloud Computing: Comparisons and Evaluations," *arXiv preprint*, arXiv:2002.04860, 2020. [Online]. Available: https://arxiv.org/abs/2002.04860

7. H. Lee, S. U. Khan, and J.-H. Park, "Hybrid approach for resource allocation in cloud infrastructure using Genetic Algorithm and Random Forest," *Scientific Programming*, vol. 2021, p. 4924708, 2021. [Online]. Available: https://doi.org/10.1155/2021/4924708

8. Y. Liang, W. Sun, and J. Li, "Energy efficiency in cloud computing data centers: a survey on software-level techniques," *Cluster Computing*, 2022. [Online]. Available: https://doi.org/10.1007/s10586-022-03713-0

9. D. Fernández-Cerero, *et al.*, "Analyzing the impact of various parameters on job scheduling in the Google Cluster dataset," *Cluster Computing*, 2024. [Online]. Available: https://doi.org/10.1007/s10586-024-04377-8

10. X. Wang and Y. Li, "A Systematic Review of Energy Management Strategies for Cloud Data Centers: ML based Workload Prediction and Resource Allocation," *Energies*, vol. 14, no. 17, p. 5322, 2021. [Online]. Available: https://doi.org/10.3390/en14175322

11. M. Martínez Ortiz, *et al.*, "Random Forest method for energy consumption prediction in buildings," *Energy and Buildings*, 2019. [Online]. Available: https://doi.org/10.1016/j.enbuild.2019.01.055

12. J. H. Abawajy, J. Magnus, and F. K. Hussain, “Sustainable Cost Energy Aware Load Balancing in Cloud,” *Journal of Sustainable Computing*, in press, 2025. [Online]. Available: https://doi.org/10.1016/j.suscom.2025.0012

13. H. Janardhanan, “AI Driven Load Balancing for Energy Efficient Data Centers,” *International Journal of Computer Trends and Technology*, vol. 72, no. 8, pp. 13–18, 2024. [Online]. Available: https://doi.org/10.14445/22312803/IJCTT-V72I8P103

14. V. Mathew, R. K. Sitaraman, and P. Shenoy, “Energy-Aware Load Balancing in Content Delivery Networks,” *IEEE/ACM Transactions on Networking*, 2018. [Online]. Available: https://doi.org/10.1145/nnnnnnn

15. A. Pratap, S. K. Gupta, Midhunchakkaravarthy, and S. Shahi, “Exploring Energy-Efficient Data Processing Technique for Sustainable Computing”, SGSES, vol. 1, no. 2, Jul. 2025.