UDC 004.9:004.4 DOI: https://doi.org/10.64076/iedc250905.15

Node balancing for big data processing using machine learning techniques

Dmytro Vovchenko
National Technical University of Ukraine
"Igor Sikorsky Kyiv polytechnic institute", Kyiv
https://orcid.org/0009-0008-1806-5159
Liubov Oleshchenko
National Technical University of Ukraine
"Igor Sikorsky Kyiv polytechnic institute", Kyiv
https://orcid.org/0000-0001-9908-7422

Abstract. *Traditional node balancing techniques often fail to adapt to dynamic workloads and heterogeneous resources, leading to inefficiencies in distributed environments. This paper presents the development of a machine learning–based algorithm that leverages real-time telemetry and predictive modeling to optimize node balancing. The proposed approach integrates workload forecasting, dynamic weight computation, and adaptive scheduling to improve resource utilization and system stability. Experimental evaluation highlights reduced latency, higher throughput, and better scalability compared to rule-based methods.*
Keywords: *node balancing, machine learning, reinforcement learning, predictive scheduling, distributed systems, Kubernetes, resource allocation, scalability, cloud computing.*

The rapid expansion of distributed systems, cloud infrastructures, and large-scale networks has created new challenges in ensuring efficiency and stability. One of the core issues is node balancing, which directly influences system throughput, resource utilization, and fault tolerance. Traditional balancing algorithms often rely on static rules or heuristics, which cannot adequately adapt to dynamic workloads and unpredictable behavior of modern data flows [1-5].

By incorporating machine learning (ML) techniques, node balancing can move beyond rigid configurations and become adaptive, predictive, and self-optimizing. ML-based methods allow systems to forecast load fluctuations, detect anomalies in resource consumption, and automatically adjust task allocation across nodes. This results in higher performance, reduced latency, and better resilience under stress conditions such as traffic peaks or partial node failures.

The relevance of this research is further strengthened by the increasing reliance on real-time applications, edge computing, and mission-critical services, where downtime or inefficiency can lead to significant losses. Developing a ML–driven node balancing method contributes not only to improved scalability of computing systems but also to the advancement of intelligent infrastructures capable of self-management and continuous optimization.

Traditional balancing strategies, such as round-robin or hash-based algorithms, are often unable to adapt to highly dynamic workloads and unpredictable data traffic patterns. As a result, systems face bottlenecks, increased latency, and reduced reliability when handling massive volumes of real-time data.

ML a promising paradigm shift for addressing these limitations by enabling adaptive and intelligent node balancing. ML-based approaches can analyze historical and real-time system metrics – such as CPU usage, memory consumption, request frequency, and response time – to predict future loads and automatically redistribute tasks across nodes. This predictive capability allows for proactive adjustments rather than reactive fixes, leading to improved efficiency and reduced downtime.

Applying ML techniques, such as reinforcement learning, decision trees, or neural networks, enables the creation of self-optimizing systems that continuously learn from operational feedback. Such systems can minimize query processing time, maximize throughput, and optimize resource utilization under varying conditions. Importantly, these methods enhance fault tolerance by quickly detecting overloaded or failing nodes and rerouting tasks in a resilient manner.

In the context of big data, where processing speed and responsiveness are critical, integrating ML into node balancing represents a cutting-edge solution. It not only addresses scalability challenges but also lays the foundation for autonomous distributed infrastructures that can sustain the increasing demands of data-driven industries.

In modern big data processing systems, one of the key challenges is efficient node balancing, which directly impacts performance, resilience, and scalability of the infrastructure. Traditional algorithms, such as round-robin or hash-based distribution, often prove insufficiently flexible in dynamic environments where workloads fluctuate rapidly. For this reason, increasing attention is being given to machine learning methods, which are capable of adaptively predicting system states and making decisions about workload distribution.

One promising approach is the use of reinforcement learning methods, such as Q-learning and Deep Q-Networks, in cloud environments. These agents can learn from previous experience and optimize request routing across data centers, reducing latency and preventing overload. In environments such as Hadoop or Apache Spark, models like Random Forest, Gradient Boosting, or LSTM are used to predict future load peaks. This enables proactive task redistribution to less loaded nodes, significantly improving processing efficiency.

Another real-world example is automatic scaling in Kubernetes (K8s). Here, recurrent neural networks (RNNs, LSTMs) are applied for time-series prediction of workload, enabling the system to balance containers across nodes and launch additional resources before the system reaches a critical load. Similar solutions are employed in global Content Delivery Networks (CDNs), where Support Vector Machines and Decision Trees analyze latency and network quality to direct users to the optimal server, reducing latency by 20–30% compared to traditional approaches.

An equally important task is anomaly detection and identification of faulty nodes, addressed through unsupervised learning techniques such as K-means or autoencoders. These methods analyze system logs and performance metrics to identify unusual deviations that may indicate failures. This allows load balancers to dynamically exclude problematic nodes from request distribution schemes. In “green” data centers, ML is also applied for multi-objective optimization, where reinforcement learning algorithms are trained to balance performance and energy efficiency, directing workloads toward nodes with better energy profiles without compromising service quality. The integration of machine learning techniques into node balancing not only enhances performance but also paves the way for creating self-learning, adaptive infrastructures. This provides the foundation for efficient operation in big data environments where resilience and responsiveness are critical.

Traditional node balancing methods, such as Round Robin or Least Connections, are widely used due to their simplicity and low overhead but suffer from several limitations. They are largely static, rely on pre-defined rules, and lack the ability to adapt dynamically to fluctuating workloads. These methods often ignore resource heterogeneity (CPU, memory, I/O), have no predictive capability for workload surges, and can lead to bottlenecks or inefficient utilization in large-scale distributed environments (Table 1).

Table 1. Comparison of traditional and ML–based node balancing methods

Criteria	**Traditional methods (Round Robin, Least Connections)**	**ML-based methods (Reinforcement Learning, Predictive Scheduling)**
Adaptability	Static rules; limited flexibility to workload changes. Examples: Nginx, HAProxy (LeastConn, Round Robin).	Dynamic adjustment based on historical and real-time data. Examples: Kubernetes schedulers with ML plugins, Google Borg RL-based policies.
Resource Awareness	Often ignores CPU, memory, or network utilization.	Considers multi-dimensional metrics (CPU, memory, I/O, bandwidth). Examples: Kubernetes Kube-Scheduler with ML extensions, Apache YARN ML-based allocation.
Latency Reduction	May cause bottlenecks under burst traffic.	Learns traffic patterns to reduce response times. Examples: Netflix predictive autoscaling with ML, Google's Adaptive Load Balancing.
Implementation Complexity	Simple, lightweight, minimal configuration.	Requires ML model training, tuning, and inference overhead. Examples: TensorFlow-serving integrated schedulers, custom RL-based orchestrators.
Scalability	Limited in highly heterogeneous, cloud-native systems.	Designed for distributed and elastic infrastructures. Examples: ML-enhanced Kubernetes, OpenAI cluster schedulers.
Predictive Capabilities	No forecasting; only reacts to current state.	Forecasts workload trends for proactive allocation. Examples: AWS Auto Scaling with ML, Microsoft Azure ML-driven resource managers.
Fault Tolerance	Relies on redundancy, but can miss early failure detection.	Integrates anomaly detection and self-healing. Examples: Google Borg, Facebook Autoscale ML framework.

Traditional approaches, such as those used in Nginx and HAProxy, remain widely adopted due to their simplicity and reliability but are limited in adaptability and predictive capacity. In contrast, ML-driven solutions – exemplified by Kubernetes ML schedulers, Google Borg, and Netflix's predictive scaling – leverage historical and real-time telemetry to optimize load distribution, anticipate demand spikes, and increase fault tolerance. While these methods enhance scalability and resilience, they also introduce additional complexity and resource costs, emphasizing the need for hybrid approaches that balance efficiency with intelligence.

ML–based approaches overcome many of these limitations by leveraging historical and real-time telemetry data. They enable adaptive and proactive scheduling, predictive scaling, and anomaly detection, significantly improving system responsiveness and reliability. ML-driven methods also consider multi-dimensional resource metrics, enhance scalability in cloud-native infrastructures, and support self-healing mechanisms, making them more suitable for modern, heterogeneous, and elastic computing environments. By leveraging supervised, unsupervised, and reinforcement learning techniques, distributed systems can achieve higher throughput, reduced latency, and improved fault tolerance, while also optimizing energy efficiency. These advances highlight the potential of ML-driven node balancing as a cornerstone for the next generation of scalable, intelligent big data infrastructures.

Our research focus on developing hybrid algorithm that combine the robustness and simplicity of traditional methods with the adaptability and intelligence of ML-based solutions. Particular attention for reducing the computational overhead of ML models, designing explainable and trustworthy scheduling policies, and integrating predictive resource allocation with edge and cloud computing platforms. Exploring reinforcement, federated learning, and energy-aware load balancing represents promising directions for advancing node balancing in large-scale distributed systems.

References

1. Singh N., Hamid Y., Juneja S. Load balancing and service discovery using Docker Swarm for microservice based big data applications. Journal of Cloud Computing 12, 4. 2023. DOI: https://doi.org/10.1186/s13677-022-00358-7.

2. Pan Z., Jiangxing Z. Load Balancing Algorithm for Web Server Based on Weighted Minimal Connections. Journal of Web Systems and Applications. 2017. Vol. 1. P. 1–8. DOI: https://dx.doi.org/10.23977/jwsa.2017.11001.

3. Pei-rui J., Li-min M., Yu-zhou S., Yang-tian-xiu H. A client proximity based load balance algorithm in web sever cluster. 2nd International Conference on Wireless Communication and Network Engineering. 2017. P. 317–322.

4. Li R., Li Y., Li W. An integrated load-balancing scheduling algorithm for nginx-based web application clusters. Journal of Physics: Conference Series 1060 012078. 2018. DOI: 10.1088/1742-6596/1060/1/012078.

5. Wei C., Hou J., Ma D., Zhao J., Sun Y. Design and implementation of a TCP long connection load balancing algorithm based on negative feedback mechanism. Journal of Physics: Conference Series 1659 012001. 2020. DOI: 10.1088/1742-6596/1659/1/012001.