Group 5: System Prototype and Initial Performance Results: Load balancing after profiling straggler tasks

Khushi Mahesh
RIT
km1139@rit.edu

Ramprasad Kokkula
RIT
rk1668@rit.edu

Mona Anil Udasi
RIT
mu9326@rit.edu

Snehith Reddy Baddam
RIT
sb3994@rit.edu

ABSTRACT

In contemporary data centers, the optimization of job performance is hindered by the presence of heterogeneous hardware, leading to the occurrence of straggler tasks on slower resources. This project proposal explores load balancing after profiling straggler tasks, focusing on the Negative Unlabeled Random Forest (NURD) algorithm. NURD, designed for non-straggler bias, will be simulated on diverse high-performance computing (HPC) traces. Our objective is to harness the capabilities of NURD in identifying stragglers within distributed environments that execute iterative workloads. Additionally, the project seeks to devise techniques for remapping tasks to non-straggler resources. The proposed load-balancing algorithm adapts dynamically to changing conditions in iterative patterns, aiming to enhance system performance. The challenges encountered in this endeavor include evaluating NURD on real-world workloads and recent HPC traces, addressing complexities associated with distributed setups, and managing induced heterogeneity.

KEYWORDS

Loadbalancing, Straggler, MachineLearning

ACM Reference Format:
Khushi Mahesh, Ramprasad Kokkula, Mona Anil Udasi, and Snehith Reddy Baddam. Year. Group 5: System Prototype and Initial Performance Results: Load balancing after profiling straggler tasks. In *Proceedings of Conference Name*. ACM, New York, NY, USA, 6 pages. https://doi.org/DOI

1 INTRODUCTION

Modern data centers are increasingly heterogeneous with hardware of different generations and diverse accelerators attached to the same servers. This heterogeneity affects the execution of jobs on data center resources because the most powerful resources complete their allocated tasks faster than the resources with less power. Also known as stragglers, tasks executed on slow computation resources affect overall job performance.

Conference Name, Month, Year, Location

ACM ISBN ISBN
https://doi.org/DOI

Datacenters face a significant challenge in dealing with straggler tasks, stragglers are extremely slow tasks that affect overall job performance.

Existing research e.g NURD [7] proposes a negative unlabelled approach, which does not require any labeling or foreknowledge of the latency distribution. NURD uses finished tasks and is biased towards non-stragglers as they are high in number.

In this work, we plan to study the following: (1) Simulate the NURD algorithm on other available traces from HPC systems; (2) Leverage the NURD algorithm for identifying the straggler tasks in our distributed setup environment running reference workloads of iterative nature (e.g. PyTorch, TensorFlow, Spark, etc.); (3) Designing and developing techniques to remap the workload tasks to non-straggler resources; and (4) Performing extensive experimental evaluations for different workload configurations.

The focus of this work is on applications characterized by iterative patterns. In these scenarios, workloads are divided into epochs, and the proposed load-balancing algorithm is designed to reevaluate and adjust resource allocations at the end of each iteration. This iterative approach enables the system to adapt dynamically to changing conditions and evolving straggler profiles.

We have added our initial findings code in https://github.com/Sin317/652-ds-project

2 CHALLENGES

While NURD has proposed an extensive set of optimizations based on the 2011 Google traces and 2018 Alibaba traces [2], they have not evaluated the merits of NURD on real-world workloads, or large HPC clusters [3].

This study focuses specifically on understanding the real-world implications of NURD, which entrails challenges in terms of distributed computing setup, inducing heterogeneity in resources if required, and mimicking the straggler patterns targetted by NURD for Google and Alibaba's traces.

3 RELATED WORK

The following papers have helped in understanding the straggler problem and the different load-balancing strategies:

- Factors Causing Stragglers [8]
- Efficient fair queuing using deficit round-robin [13]
- Strategies for dynamic load balancing on highly parallel computers [17]
- Straggler mitigation at scale [1]

- Addressing the straggler problem for iterative convergent parallel ML [9]
- Execution Analysis of Load Balancing Algorithms in cloud computing Environments [15]
- Semi-Dynamic Load Balancing: Efficient Distributed Learning in Non-Dedicated Environments [5]
- START: Straggler Prediction and Mitigation for Cloud Computing Environments using Encoder LSTM Networks [14]
- Load balancing with memory [11]
- Load balancing algorithms in cloud computing: A survey of modern techniques [4]
- Load balancing algorithms in cloud computing: A survey [10]
- Observations on using genetic algorithms for dynamic load-balancing [18]
- DBSCAN Clustering Algorithm Based on Density [6]
- A bagging SVM to learn from positive and unlabeled examples [12]

3.1 Factors Causing Stragglers

An in-depth analysis categorizes dominant factors leading to straggler occurrences, including high CPU utilization, disk overloading, unbalanced workload aggregation, poor user code, server faults, network conditions, and task request surges. Insights from the study provide guidance for handling various straggler scenarios in cloud data centers through historical data analysis and real-time monitoring using offline and online analytics components. [8]

3.2 Efficient fair queuing using deficit round-robin

Fair queuing is a technique that allows each flow passing through a network device to have a fair share of network resources. This paper talks about a new approximation of fair queuing, called as deficit round-robin. This achieves nearly perfect fairness in terms of throughput, requires only O(1) work to process a packet, and is simple enough to implement in hardware. [13]

3.3 Strategies for dynamic load balancing on highly parallel computers

The analysis compares five dynamic load balancing strategies tailored for highly parallel systems to minimize total job completion time. The Dimension Exchange Method (DEM) stands out, with variable efficiency based on system topology. Strategies include Sender (Receiver) Initiated Diffusion (SID-IRID), Hierarchical Balancing Method (HBM), Gradient Model (GM), DEM, and Load Balancing Profitability Determination. They address aspects like global load balancing, hierarchical organization, proximity-based migration, dimension-based synchronization, and profitability assessment, collectively enhancing load balancing efficiency in highly parallel systems. [17]

3.4 Straggler Mitigation at scale

This paper presents a cost (pain) vs. latency (gain) analysis of executing jobs of many tasks by employing replicated or erasure coded redundancy. The tail heaviness of service time variability is decisive on the pain and gain of redundancy and we quantify its effect by deriving expressions for cost and latency. [1]

3.5 Mitigation of the straggler problem for iterative convergent parallel ML

FlexRR [9] addresses the straggler problem in distributed machine learning (ML) environments with an adaptable synchronization model that dynamically reallocates work among worker threads, ensuring scalability and efficiency. Unlike traditional Bulk Synchronous Parallel (BSP) methods, FlexRR mitigates the impact of stragglers on overall performance through peer-to-peer work reassignment. Experimentation on Amazon EC2 and Microsoft Azure platforms, along with stress tests simulating straggler behavior, validate FlexRR's efficacy. It consistently achieves near-optimal runtimes across diverse straggler scenarios, demonstrating significant reductions ranging from 15% to 56% compared to existing approaches. FlexRR's integration of flexible consistency bounds and temporary work reassignment strategies outperforms BSP and SSP methods, making it a promising solution for optimizing distributed ML workflows and enhancing performance and productivity. [9]

3.6 Execution Analysis of Load Balancing Algorithms

The analysis compares various load-balancing algorithms in cloud computing, considering factors like stability, resource utilization, scalability, and cost-effectiveness. Token Routing minimizes costs but struggles with scalability, while Round Robin offers simplicity but may lead to uneven loads. Randomized algorithms allocate processes based on static probabilities, posing challenges in handling workload complexities. Central Queuing buffers requests dynamically, and the Connection Mechanism uses least connection algorithms for load balancing. Despite diversity, challenges persist in managing non-uniform load arrivals, emphasizing the need for comprehensive evaluations. Additionally, algorithms are classified as static or dynamic based on workload assignment timing, each with its benefits depending on system requirements and environmental conditions. [15]

3.7 Semi-Dynamic Load Balancing: Efficient Distributed Learning in Non-Dedicated Environments

The paper introduces LB-BSP, a novel load balancing strategy for distributed machine learning (ML) workloads. LB-BSP aims to eliminate stragglers—slow workers—in clusters with heterogeneous resources. It balances ML workers at iteration boundaries without disrupting intra-iteration execution, adapting load based on workers' processing capabilities. LB-BSP is implemented as a Python module for TensorFlow and PyTorch and accelerates distributed training by up to 54% on Amazon EC2. Overall, LB-BSP offers a practical and effective solution for improving model convergence in non-dedicated clusters.[5]

3.8 START: Straggler Prediction and Mitigation for Cloud Computing Environments using Encoder LSTM Networks

The paper introduces START, a Straggler Prediction and Mitigation Technique designed to address performance issues caused by straggler tasks in large-scale computing systems. START utilizes an Encoder Long-Short-Term-Memory (LSTM) network to predict and mitigate potential stragglers, dynamically adapting scheduling to optimize response times. Through simulation in a cloud environment, START is compared with state-of-the-art techniques, demonstrating reductions in execution time, resource contention, energy consumption, and SLA violations, ultimately improving overall system performance and quality of service.[14]

3.9 Load balancing with memory

The paper explores a load balancing model where balls are sequentially placed into bins, with each ball having a choice of d possible locations. Introducing memory into the model, where the least loaded choice of the previous ball is remembered and considered for the next ball's placement, significantly improves performance. For instance, with just one random choice and one choice from memory, the maximum number of balls in a bin is log log n/2 log /spl phi/ + O(1) with high probability, outperforming models with only random choices. This suggests that incorporating memory, akin to providing a small amount of choice, greatly enhances load balancing performance. The study also extends its findings to continuous-time variations resembling queueing systems, yielding similar improvements. [11]

3.10 Load balancing algorithms in cloud computing: A survey of modern techniques

The abstract highlights the importance of load balancing in cloud computing and outlines the research conducted on load balancing algorithms from 2004 to 2015. Various algorithms such as Round Robin, Min-Min, Max-Min, Ant colony, Carton, and Honey bee are compared based on characteristics like fairness, throughput, fault tolerance, overhead, performance, response time, and resource utilization. The paper aims to provide a structured overview of these algorithms, categorizing them based on their underlying models. The conclusion emphasizes the need for addressing issues like fairness and high throughput in load balancing algorithms and suggests exploring hybrid approaches for better performance and system security in the future. [4]

3.11 Load-balancing algorithms in cloud computing: A survey

This paper highlights the literature on the task scheduling and load-balancing algorithms and presents a new classification of such algorithms, for example, Hadoop MapReduce load balancing category, Natural Phenomena-based load balancing category, Agent-based load balancing category, General load balancing category, application-oriented category, network-aware category, and workflow specific category. [10]

3.12 Observations on using genetic algorithms for dynamic load-balancing

Genetic algorithms have gained immense popularity over the last few years as a robust and easily adaptable search technique. The work proposed here investigates how a genetic algorithm can be employed to solve the dynamic load-balancing problem. A dynamic load-balancing algorithm is developed whereby optimal or near-optimal task allocations can "evolve" during the operation of the parallel computing system. [18]

3.13 DBSCAN Clustering Algorithm Based on Density

The density-based clustering algorithm can cluster arbitrarily shaped data sets in the case of unknown data distribution. DBSCAN is a classical density-based clustering algorithm, which is widely used for data clustering analysis due to its simple and efficient characteristics. [6]

3.14 A bagging SVM to learn from positive and unlabeled examples

The paper proposes an algorithm that iteratively trains many binary classifiers to discriminate the known positive examples from random subsamples of the unlabeled set, and averages their predictions. The proposed method can also run considerably faster than state-of-the-art methods, particularly when the set of unlabeled examples is large. [12]

3.15 Summary

The literature review, we explore dynamic load balancing and straggler mitigation in parallel computing environments, focusing on cloud computing and distributed machine learning (ML). It identifies factors contributing to performance issues and emphasizes the importance of real-time monitoring. Various load balancing strategies are analyzed, with the Dimension Exchange Method (DEM) highlighted for its efficiency. Innovative solutions like FlexRR and LB-BSP show promise in mitigating stragglers in distributed ML environments. Load balancing algorithms in cloud computing are compared based on stability, resource utilization, and scalability. Challenges in managing non-uniform load arrivals are noted, emphasizing the importance of selecting appropriate algorithms. Different Machine Learning Algorithms to detect straggler tasks are also iddentified.

4 IMPACT

Data centers are adversely affected by stragglers, which are slow tasks that delay job completion and degrade overall performance. Precise anticipation of stragglers could empower data center operators to intervene proactively, preempting any delays they might cause to a job.

With ML models especially in the domain of federated learning where models are trained locally on individual mobile devices, there is a high chance of having straggler tasks as well, to identify and load balance such tasks this would improve the performance of the models.

5 METHODOLOGY

The load-balancing algorithm introduced in this extension operates in conjunction with the NURD algorithm. It utilizes the output generated by NURD, which includes information about the resources exhibiting straggler behavior. This information forms the basis for determining the weight for the load balancing.

The weighted least outstanding requests (WLOR) algorithm dynamically adjusts the distribution of tasks among computing resources based on the identified stragglers, optimizing resource utilization and minimizing the impact of performance variations.

A crucial aspect of the experiment involves exploring different configurations to identify the ideal update frequency for reallocating workloads to each worker process to ensure overall optimal utilization of resources.

The Alibaba trace data "ClusterData201708" contains cluster information of a production cluster in 12 hours period: about 1.3k machines that run both online service and batch jobs. We use this dataset after modelling it as a time-series to predict straggler tasks based on the technique defined by [7].

Each job was split into tasks. Every task could be further split into parallel computed batch instances that ran on different machines. The tasks are characterized as "stragglers" if any of the instance would use significantly more resources than the other instances.

6 INITIAL RESULTS

In our analysis, we have used the Alibaba cluster trace dataset, specifically focusing on the 'batch_task.csv' and 'batch_instance.csv' files. We chose the Alibaba dataset due to its manageable size compared to the vast Google dataset, as preprocessing the latter was proving to be time-intensive. Additionally, considering our limited computational resources such as GPU, CPU, and RAM, we opted to work with a subset of the Alibaba dataset to ensure efficient processing and analysis.

The preprocessing stage involved transforming the raw time series data into a format conducive to analysis. We extracted key attributes such as task creation and modification timestamps, job identifiers, instance numbers, status, and resource specifications like CPU and memory requirements. Tasks were segmented into instances, each characterized by attributes such as start and end timestamps, machine IDs, status, and resource utilization metrics. Subsequently, the dataset was partitioned into training and testing sets, with a specific focus on tasks and instances labeled as 'terminated' to identify potential straggler instances. Furthermore, we characterized selected tasks from jobs to identify their CPU usage, memory usage, and latency.

Figure 1 depicts this analysis. After predicting the straggler task from outlier detection (DBSCAN) [6], we used the Weighted Least Outstanding Requests (WLOR) [16] algorithm to determine which machine could take on the straggler task.

In addition to the algorithmic analysis, we developed a simulation tool using Python and the Tkinter library for the graphical user interface. This tool allows users to input a time series dataset containing task information and resource usage metrics. It visualizes workload distribution among resources and simulates load balancing in real-time, providing insights into workload management strategies.

Overall, our analysis and simulation results contribute to a comprehensive understanding of the NURD algorithm's capabilities in identifying and mitigating straggler instances within the selected trace.

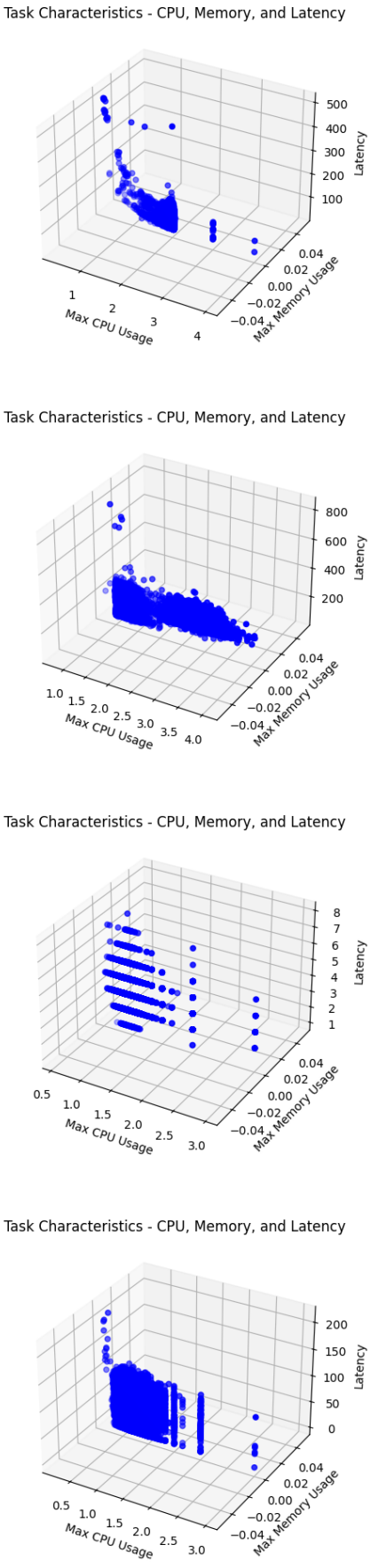

Figure 1: Task characteristics

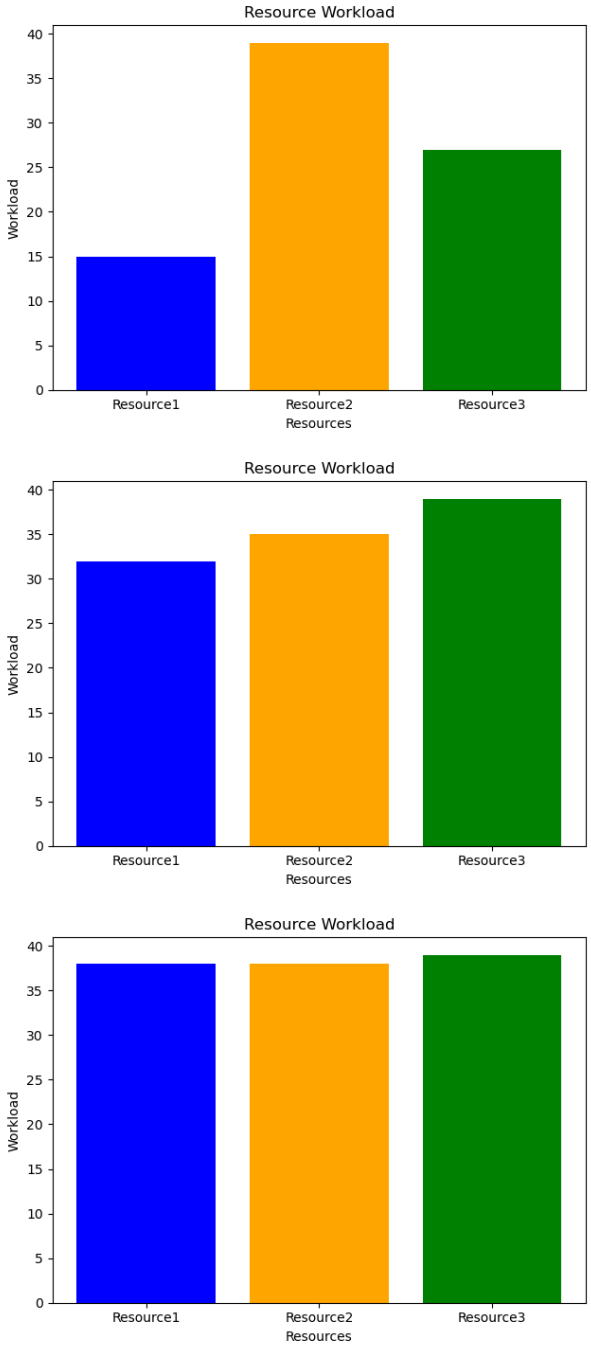

Figure 2: Resource utilization
shows sample result of load-balancing the straggler tasks identified in the Alibaba trace dataset.

7 CONCLUSION

Overall, we have been able to emulate NURD's approach to predict stragglers. This foundational understanding enables us to progress to the next phases outlined in our proposal.

REFERENCES

[1] Mehmet Fatih Aktaş and Emina Soljanin. 2019. Straggler Mitigation at Scale. *IEEE/ACM Transactions on Networking* 27, 6 (2019), 2266–2279. https://doi.org/10.1109/TNET.2019.2946464

[2] Alibaba. [n. d.]. Alibaba production cluster trace data. *https://github.com/alibaba/clusterdata* ([n. d.]).

[3] Moiz Arif, M. Mustafa Rafique, Seung-Hwan Lim, and Zaki Malik. 2020. Infrastructure-Aware TensorFlow for Heterogeneous Datacenters. In *2020 28th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*. 1–8. https://doi.org/10.1109/MASCOTS50786.2020.9285969

[4] Sidra Aslam and Munam Ali Shah. 2015. Load balancing algorithms in cloud computing: A survey of modern techniques. In *2015 National Software Engineering Conference (NSEC)*. 30–35. https://doi.org/10.1109/NSEC.2015.7396341

[5] Chen Chen, Qizhen Weng, Wei Wang, Baochun Li, and Bo Li. 2020. Semi-dynamic load balancing: efficient distributed learning in non-dedicated environments. In *Proceedings of the 11th ACM Symposium on Cloud Computing* (Virtual Event, USA) *(SoCC '20)*. Association for Computing Machinery, New York, NY, USA, 431–446. https://doi.org/10.1145/3419111.3421299

[6] Dingsheng Deng. 2020. DBSCAN Clustering Algorithm Based on Density. In *2020 7th International Forum on Electrical Engineering and Automation (IFEEA)*. 949–953. https://doi.org/10.1109/IFEEA51475.2020.00199

[7] Yi Ding, Avinash Rao, Hyebin Song, Rebecca Willett, and Henry (Hank) Hoffmann. 2022. NURD: Negative-Unlabeled Learning for Online Datacenter Straggler Prediction. In *Proceedings of Machine Learning and Systems*, D. Marculescu, Y. Chi, and C. Wu (Eds.), Vol. 4. 190–203. https://proceedings.mlsys.org/paper_files/paper/2022/file/21d45762b38dbbd05c0ddbd6601d8f6c-Paper.pdf

[8] Peter Garraghan, Xue Ouyang, Renyu Yang, David McKee, and Jie Xu. 2019. Straggler Root-Cause and Impact Analysis for Massive-scale Virtualized Cloud Datacenters. *IEEE Transactions on Services Computing* 12, 1 (2019), 91–104. https://doi.org/10.1109/TSC.2016.2611578

[9] Aaron Harlap, Henggang Cui, Wei Dai, Jinliang Wei, Gregory R. Ganger, Phillip B. Gibbons, Garth A. Gibson, and Eric P. Xing. 2016. Addressing the straggler problem for iterative convergent parallel ML. In *Proceedings of the Seventh ACM Symposium on Cloud Computing* (Santa Clara, CA, USA) *(SoCC '16)*. Association for Computing Machinery, New York, NY, USA, 98–111. https://doi.org/10.1145/2987550.2987554

[10] Einollah Jafarnejad Ghomi, Amir Masoud Rahmani, and Nooruldeen Nasih Qader. 2017. Load-balancing algorithms in cloud computing: A survey. *Journal of Network and Computer Applications* 88 (2017), 50–71. https://doi.org/10.1016/j.jnca.2017.04.007

[11] M. Mitzenmacher, B. Prabhakar, and D. Shah. 2002. Load balancing with memory. In *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings.* 799–808. https://doi.org/10.1109/SFCS.2002.1182005

[12] F. Mordelet and J. P. Vert. 2014. A bagging SVM to learn from positive and unlabeled examples. *Pattern Recogn. Lett.* 37 (feb 2014), 201–209. https://doi.org/10.1016/j.patrec.2013.06.010

[13] M. Shreedhar and G. Varghese. 1996. Efficient fair queuing using deficit round-robin. *IEEE/ACM Transactions on Networking* 4, 3 (1996), 375–385. https://doi.org/10.1109/90.502236

[14] Peter Garraghan Rajkumar Buyya Giuliano Casale1 Nicholas R. Jennings Shreshth Tuli, Sukhpal S. Gill. 2021. START: Straggler Prediction and Mitigation for Cloud Computing Environments using Encoder LSTM Networks. IEEE Transactions on Services Computing. https://doi.org/10.1109/TSC.2021.3129897

[15] Ajanta De Sarkar Soumya Ray. 2012. EXECUTION ANALYSIS OF LOAD BALANCING ALGORITHMS IN CLOUD COMPUTING ENVIRONMENT. *International Journal on Cloud Computing: Services and Architecture* 2, 5 (2012). https://doi.org/10.5121/ijccsa.2012.2501

[16] Weikun Wang and Giuliano Casale. 2014. Evaluating Weighted Round Robin Load Balancing for Cloud Web Services. In *2014 16th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*. 393–400. https://doi.org/10.1109/SYNASC.2014.59

[17] M.H. Willebeek-LeMair and A.P. Reeves. 1993. Strategies for dynamic load balancing on highly parallel computers. *IEEE Transactions on Parallel and Distributed Systems* 4, 9 (1993), 979–993. https://doi.org/10.1109/71.243526

[18] A.Y. Zomaya and Yee-Hwei Teh. 2001. Observations on using genetic algorithms for dynamic load-balancing. *IEEE Transactions on Parallel and Distributed Systems* 12, 9 (2001), 899–911. https://doi.org/10.1109/71.954620