

# CLOUD COMPUTING

**Name:** An In-Depth Empirical Investigation of State-of-the-Art Scheduling  
Approaches for Cloud Computing Task Scheduling

**Published in:** IEEE Access, vol. 8, pp. 128282–128294, 2020,

**First Publisher Name:** M. Ibrahim et al.

**Page(s):** IEEE Access ( Volume: 8)

**Date of Publication:** 128282 – 128294

**Electronic ISSN:** 06 July 2020

**INSPEC Accession:** 2169–3536

**Number:** 19800515


**DOI:** 10.1109/ACCESS.2020.3007201

**Publisher:** IEEE



# INTRODUCTION : TASK SCHEDULING

Task scheduling in cloud computing is the process of **efficiently assigning computing tasks and workloads to available resources** within a cloud infrastructure. It is crucial for optimizing resource utilization, reducing operational costs, and ensuring that tasks are completed within the desired timeframes. Effective task scheduling helps balance the workload, improve system performance, and maximize the overall efficiency of cloud-based applications and services, making it a fundamental component in delivering reliable and cost-effective cloud computing solutions. However, **task scheduling in Cloud computing is NP-hard in nature**. With this project, we will evaluate many Task Scheduling Algorithms based on various parameters.







# PROBLEM STATEMENT:

## AN EMPIRICAL INVESTIGATION OF STATE-OF-THE-ART TASK SCHEDULING ALGORITHMS FOR CLOUD COMPUTING

**User Demands :** The execution time with minimum SLA violation is of primary concern.

**Service Providers Demands :** The focus is to obtain maximum profitability from their services.

In this presentation, We will delve into the research paper's focus on task scheduling and load balancing within the realm of cloud computing. The Authors emphasize the complexity of task scheduling, categorizing it as an NP-hard problem, and stress the importance of exploring and contrasting various scheduling strategies to enhance performance and optimize resource utilization in cloud environments. Our goal is to offer a comprehensive empirical exploration of scheduling methods and evaluate their efficiency in areas such as makespan, resource utilization, energy consumption, and throughput.





# EXPERIMENTAL SETUP

# TYPES OF EXPERIMENTAL SETUPS

## Experimental

The Experiment-based cloud setup uses real-world cloud infrastructure and resources to conduct empirical experiments. This approach involves deploying actual applications and workloads on a live cloud environment and collecting data on their performance.

The experimental techniques are expensive and difficult to set up and may require an expert to design the testbeds. Using this may result in high monetary costs.

## Analytical

The Analytical Methods include defining the cloud infrastructure's specifications and then using vast amount of data from tasks and workloads, gathering data on performance to draw conclusions and make improvements in cloud computing systems.

The analytical techniques are often limited in evaluating the proposed scheduling heuristics

## Simulation

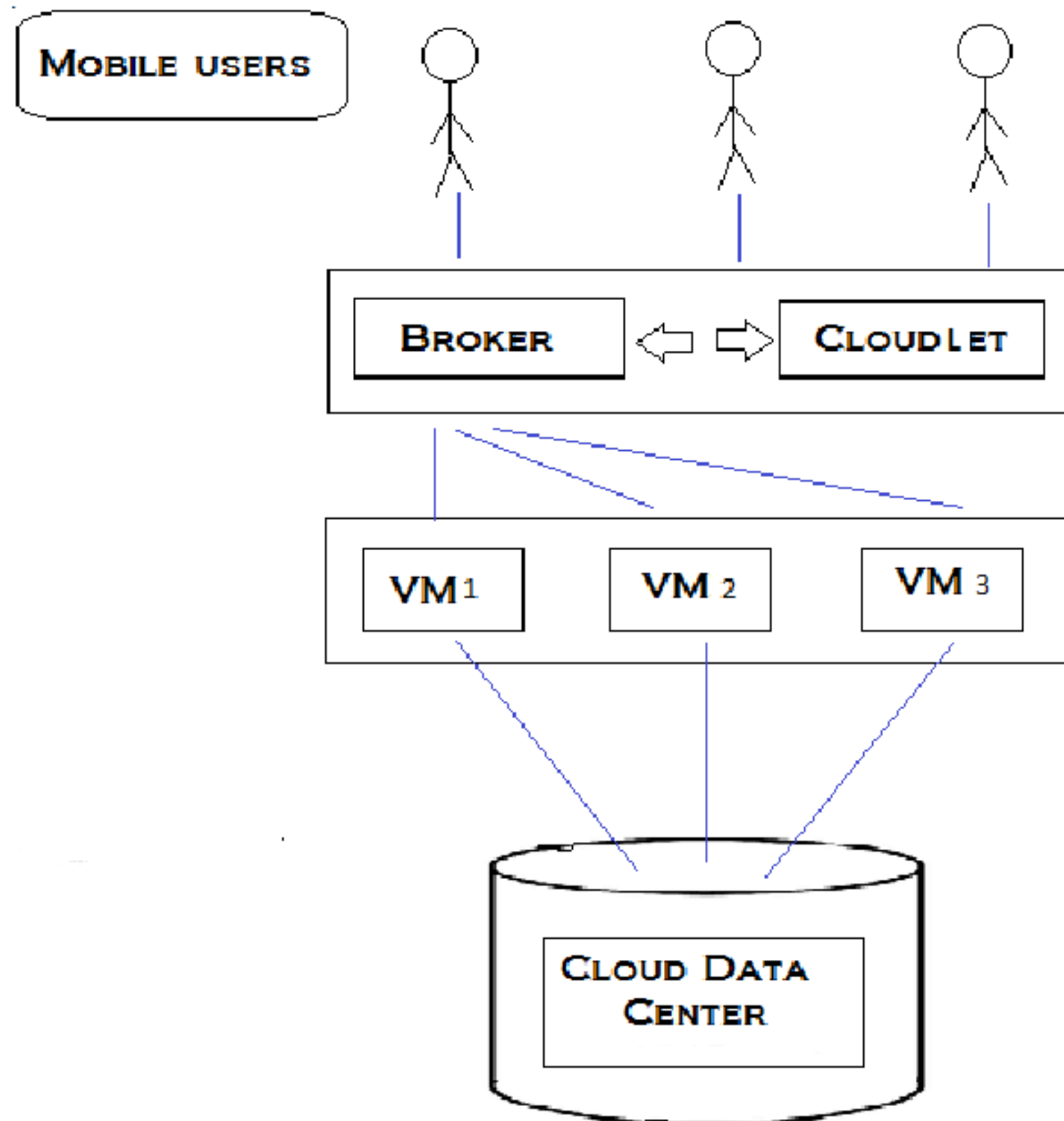
The Simulation-based cloud setup employs computer software to create a virtual, simulated cloud computing environment. This replicates the behavior and characteristics of real cloud systems to conduct controlled experiments without the cost and complexity of using physical resources

Simulation approaches are extensively used to evaluate the performance of the underlying scheduling approaches using a wide variety of configurations.

# SIMULATION METHOD AND CLOUDSIM

- **Cost Efficiency** : Cloud-Sim allows users to simulate and evaluate different scheduling approaches without the need for expensive physical testbeds or real cloud platforms. This helps in reducing costs associated with experimentation and analysis.
- **Flexibility & Performance Evaluation** : Cloud-Sim provides a flexible and customizable simulation environment, allowing researchers and service providers to test and compare various scheduling algorithms and strategies. This flexibility enables them to adapt and optimize their resource allocation and task scheduling processes according to their specific requirements.
- **Decision Making** : By using Cloud-Sim, service providers can make informed decisions regarding resource allocation and task scheduling. They can analyze the trade-offs between different scheduling heuristics and choose the approach that best meets their performance and efficiency goals.

# SIMULATION METHOD AND CLOUDSIM



# HETEROGENEOUS COMPUTING SCHEDULING PROBLEM (HCSP) DATASET

The Heterogeneous Computing Scheduling Problem (HCSP) dataset is a dataset associated with research and experimentation in the field of heterogeneous computing and task scheduling within cloud or distributed computing environments.

It was proposed by Braunt et al. The used model is based on the Expected Time to Compute (ETC) matrix with  $m$  number of tasks and  $n$  number of VMs. The instance with size  $1024 \times 32$  is considered for the evaluation of the compared heuristic approaches

Each dataset instance denotes 1024 tasks also called Cloudlets (tasks) and 32 virtual machines (VMs). Four different instances (i.e., c-hilo, i-hilo, c-lohi, and i-lohi) are utilized.

hilo: Heavy set of tasks with the light capacity of resources

lohi: Light set of tasks and high capacity of resources



$$\text{ETC (Expected Time to Compute)} = \text{Task Length(MI)} / \text{VM Power (MIPS)}$$





# PARAMETERS/EXPERIMENTAL EVALUATION :

The evaluation of the proposed scheduling approaches in the document is based on several parameters. These parameters include the makespan, average resource utilization (ARUR), throughput, load imbalance, and energy consumption. The makespan measures the completion time of virtual machines (VMs), while the ARUR calculates the average makespan. Throughput is measured as the number of tasks divided by the makespan. Load imbalance refers to the distribution of tasks among VMs, and energy consumption is also considered as a performance metric. These parameters are used to compare and analyze the performance of different scheduling approaches in the document.






# MAKESPAN

## Formula

Makespan=Max{CT<sub>j</sub>}=Max{CT<sub>1</sub>,CT<sub>2</sub>,...,CT<sub>m</sub>}

## About

Makespan refers to the total time taken to complete all the tasks in a given schedule. Makespan is defined as the maximum completion time among all the virtual machines (VMs) involved in the scheduling process. The objective of task scheduling algorithms is to minimize the makespan, as a lower makespan indicates better efficiency and faster completion of tasks. The project also presents a comparison of different scheduling approaches based on their makespan values, showing the performance of each approach in terms of task completion time.






# ARUR

## Formula

$ARUR = \text{avgMakespan} / \text{Makespan}$

## About

ARUR (Average Resource Utilization Ratio) is a performance metric used to measure the load balancing and resource utilization achieved by scheduling approaches in cloud computing. It is calculated as the average makespan divided by the maximum makespan. The higher the ARUR value, the better the resource utilization. The ARUR value depends on the available tasks in the datasets.






# THROUGHPUT

## Formula

Throughput =  $\text{number of tasks} / \text{Makespan}$ .

## About

Throughput is a measure of the number of tasks that can be completed within a given time period. It is calculated by dividing the number of tasks by the makespan, which is the total time taken to complete all the tasks. In the context of cloud computing, throughput is an important metric as it indicates the efficiency and performance of the scheduling algorithms. The higher the throughput, the more tasks can be processed in a given time, leading to better resource utilization and improved overall system performance.





# LOAD IMBALANCE



## Formula


Load Imbalance =  $\Sigma(\text{Load\_VM}_i - \text{Average\_Load})$

Load \_VM =  $\Sigma(\text{Length of Tasks assigned to VM (MI)})$

Average\_Load =  $\Sigma(\text{LoadVM}) / \text{number of VM}$

## About

Load imbalance refers to the uneven distribution of workload among the virtual machines (VMs) in a cloud computing environment. It is important to measure and quantify load imbalance in order to assess the performance and efficiency of scheduling approaches. Load imbalance is measured as a percentage at the VM-level. We compared the workload distribution on each VM to determine the difference in utilization. The percentage load imbalance is calculated by comparing the actual workload on a VM to the ideal workload that would result in a perfectly balanced distribution.





# TASK SCHEDULING ALGORITHMS

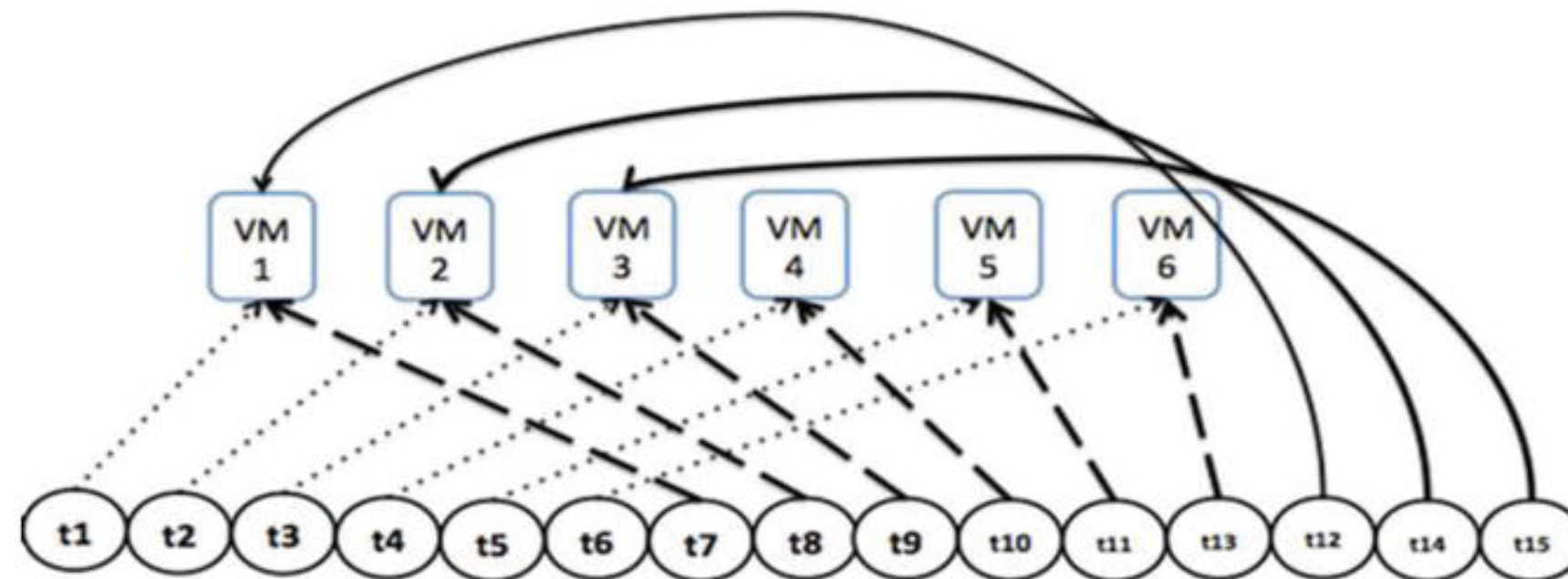
# FIRST COME FIRST SERVE

First Come First Serve (FCFS) is a simple and intuitive scheduling algorithm widely used in various computing environments. FCFS determines the order in which tasks are executed based on their arrival time.

Here is a step-by-step explanation of the First Come First Serve algorithm:

- Tasks arrive at the cloud system with varying execution times.
- Maintain a queue data structure to store the incoming tasks in the order they arrive. Map the tasks onto the selected VM.
- When a resource becomes available, the task at the front of the queue is selected for execution.
- Once a task completes its execution, release the allocated resources and remove the task from the queue.
- Select the next task in the queue for execution.

The First Come First Serve (FCFS) task scheduling algorithm allocates tasks to available resources in the order they arrive, without considering their execution time or priority.





# SHORTEST JOB FIRST




Shortest Job First (SJF) is a scheduling algorithm used in various computing environments to optimize task execution by prioritizing tasks with the shortest Task Length.

Here is a step-by-step explanation of the Max-Min algorithm:

- Tasks arrive at the cloud system with varying execution times and resource requirements.
- Maintain a queue data structure to store the incoming tasks in order of their Task Length (MI).
- When a resource becomes available, the task at the front of the queue (with the shortest estimated execution time) is selected for execution.
- Once a task completes its execution, release the allocated resources and remove the task from the queue.
- After a task is completed, select the next task in the queue for execution.


The Max-Min algorithm prioritizes larger tasks, which can lead to a higher makespan for larger datasets. However, it may result in reduced throughput for smaller tasks.







# ROUND ROBIN




The Round Robin algorithm distributes incoming network traffic or service requests evenly among a set of servers.

Here's how the round-robin algorithm works in a cloud computing environment:

- Tasks arrive at the cloud system with varying execution times and resource requirements.
- When a resource becomes available, the task at the front of the queue is selected for execution.
- Time Quantum is Set by the Algorithm for each Task
- Once the Time of Execution exceeds the Time Quantum then the progress is stored and the Task is deallocated the resources.
- The next task in the Queue is given the resource and this process is repeated until all tasks are completed.

The length of the time quantum, also known as the time slice, is usually configurable. This parameter determines how long each VM gets to execute before the scheduler moves on to the next VM.





# MIN-MIN ALGORITHM




The Min-Min algorithm is a static task scheduling heuristic used in cloud computing. It follows the basic principles of the Minimum Completion Time (MCT) approach for task-to-VM mapping.

Here's how the Min-Min algorithm works:

- Calculate the Expected Completion Time (ECT) for all tasks on each VM.
- Select the task with the minimum ECT.
- Assign the selected task to the VM that can execute it in the minimum time.
- Update the load of the assigned VM and remove the task from the task list.
- Repeat steps 2-4 until all tasks are assigned to VMs.

The Min-Min algorithm favors smaller tasks and penalizes tasks with larger sizes. While it can reduce the makespan (total time taken to complete all tasks), it may result in reduced resource utilization and throughput.



# MAXIMUM COMPELITION TIME ALGORITHM

The Maximum Completion Time (MCT) algorithm is a task scheduling heuristic that aims to minimize the makespan, which is the total time taken to complete all tasks.

Here are the steps involved in the MCT algorithm:

- Start by initializing the VM load/ready time for each VM to zero.
- Scan all VMs for each task and calculate their completion time. Select the task with the maximum completion time.
- Select the VM with the minimum completion time among the already selected tasks.
- Assign the selected task to the chosen VM.
- Update the VM load/ready time based on the completion time of the assigned task.
- Repeat steps 2 to 5 until all tasks are assigned to VMs.
- Start executing the assigned tasks on their respective VMs.

MCT algorithm aims to minimize the makespan by selecting tasks in the given order and assigning them to VMs that can execute the tasks in minimum time. MCT algorithm may lead to poor resource utilization as faster VMs may become overloaded while slower VMs remain idle.

# ENHANCED MAX-MIN ALGORITHM

The Enhanced Max-Min task scheduling algorithm is an improvement over the Max-Min approach for task scheduling in cloud computing. It aims to achieve load balancing and efficient resource utilization.

Here is a step-by-step explanation of the Max-Min algorithm:

- The algorithm first calculates the execution time of all tasks on each virtual machine (VM). This is done to determine the row with the highest execution time.
- In the already selected row, the algorithm selects the VM with the minimum execution time. This VM is chosen to map the tasks.
- After each scheduling decision, the load of the VM is updated. This ensures that the VM load is balanced and optimized.
- Once all the tasks are mapped to their respective VMs, the execution of the tasks begins. Update the VM load on each scheduling decision.

The Enhanced Max-Min algorithm improves upon the Max-Min approach by considering the size of the tasks and their execution time. It aims to favor larger tasks while penalizing smaller tasks, resulting in better resource utilization and load balancing.

# SUFFARAGE ALGORITHM

The Suffrage algorithm is a task scheduling heuristic used in cloud computing. It works by selecting a suitable virtual machine (VM) for a given task based on the difference between the minimum completion time (MCT) of the task and the second MCT. Suffrage aims to minimize makespan and improve throughput, particularly for larger datasets with a substantial number of larger tasks.

Here is a step-by-step explanation of the Suffrage Algorithm :

- Suffrage scheduling calculates the suffrage value for each job.
- To calculate the suffrage value (i.e., a penalty in terms of longer execution time), the minimum completion time and the second best minimum completion time producing VMs are determined for each job (in each scheduling iteration).
- Afterward, the job experiencing the highest suffrage value is assigned to the machine (producing minimum completion time for that job).

Suffrage heuristic produces good results often with reduced makespan; however, this scheduling mechanism causes higher scheduling overhead (due to the calculation of suffrage value for each job in each scheduling iteration) as compared to OLB, MCT, Max-Min, and Min-Min




# RASA ALGORITHM

The RASA (Resource Aware Scheduling Algorithm) combines the strengths of the Min-Min and Max-Min task scheduling heuristics. RASA calculates the execution time of each task on every available virtual machine (VM) and stores this information in a matrix.

- In the first phase, RASA uses the Min-Min scheduling heuristic, which favors smaller jobs. This phase helps in achieving load balancing by evenly distributing tasks among the available resources.
- In the second phase, RASA employs the Max-Min task scheduling heuristic to select a suitable virtual machine (VM) for each task. The Max-Min algorithm selects the row with the highest execution time and then selects the VM with the minimum execution time within that row. The tasks are then mapped to the selected VMs.

Mostly, RASA results in a lower makespan when it considers smaller and larger jobs in alternate scheduling steps. However, RASA penalizes smaller size jobs (causing delayed execution) when the number of larger jobs is higher in the workload. RASA addresses this issue by providing a fair task allocation criteria for both small and large tasks. However, it should be noted that RASA may lead to load imbalance if there are a higher number of large size tasks.








# TASA ALGORITHM




The Task Aware Scheduling Algorithm (TASA) aims to optimize task scheduling based on the characteristics of the tasks and available resources. TASA works in two phases:

1. In the first phase, TASA uses the Min-Min scheduling heuristic, which favors smaller jobs. This phase helps in achieving load balancing by evenly distributing tasks among the available resources.
2. In the second phase, TASA employs the sufferage task scheduling heuristic to select a suitable virtual machine (VM) for each task. This heuristic considers the difference between the minimum completion time (MCT) of a task and the MCT of the second task. By selecting the VM with the highest sufferage value, TASA aims to further optimize task scheduling.

TASA has been found to be particularly effective for both small and large tasks, as well as for consistent and inconsistent heterogeneous datasets. TASA's ability to achieve load-balanced scheduling and efficient resource utilization makes it a suitable choice for cloud service providers.

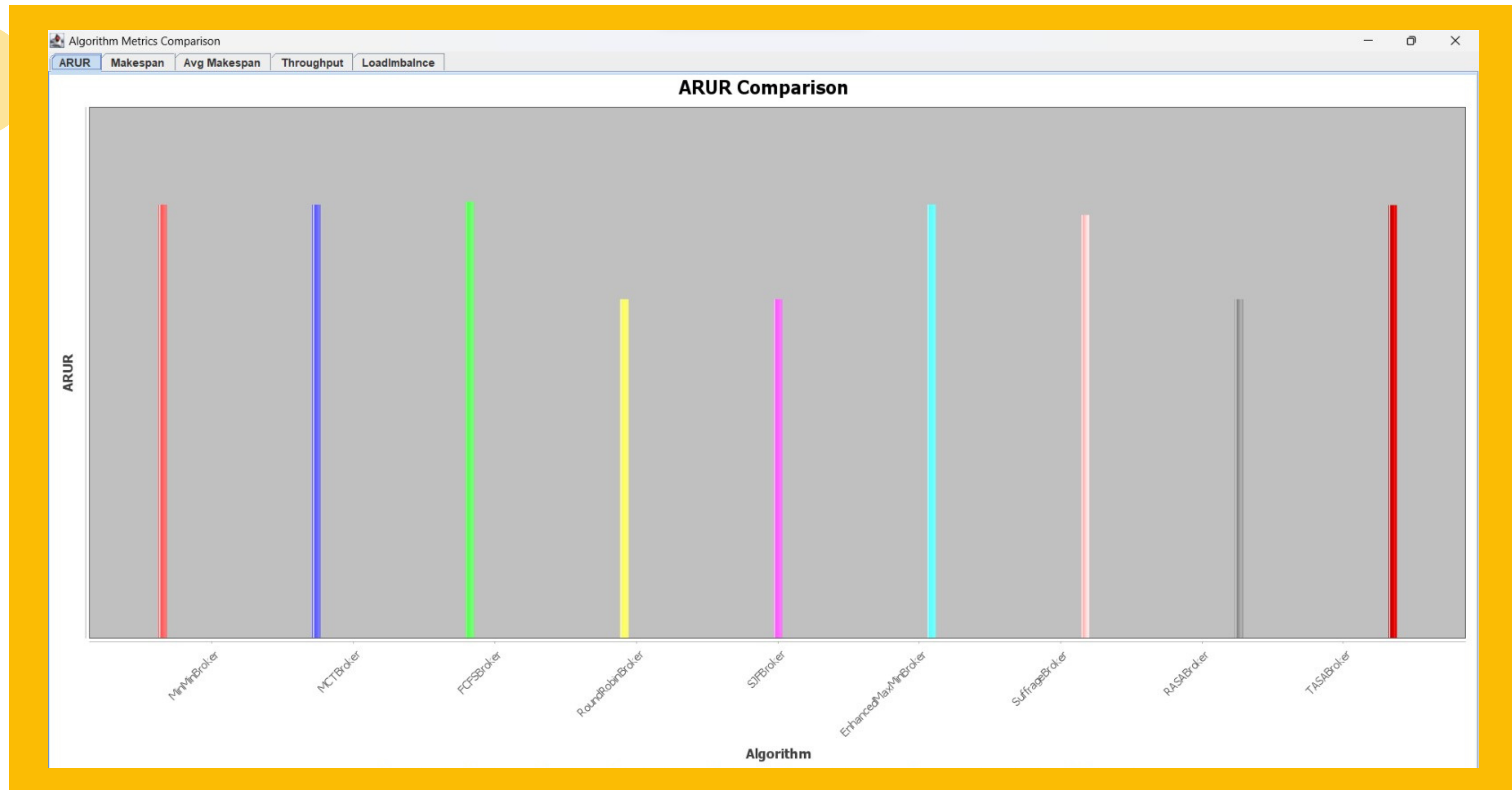




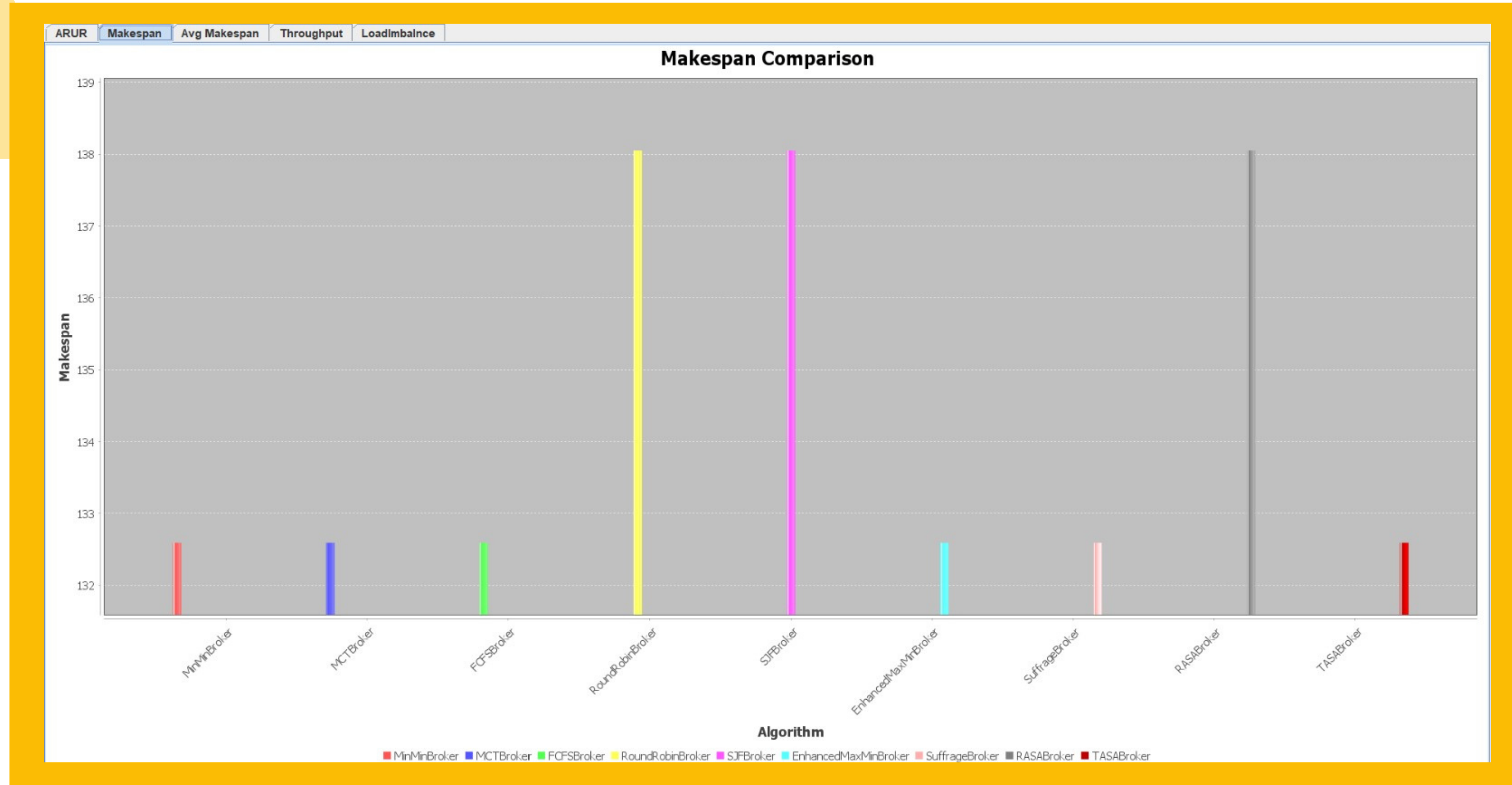
# RESULTS



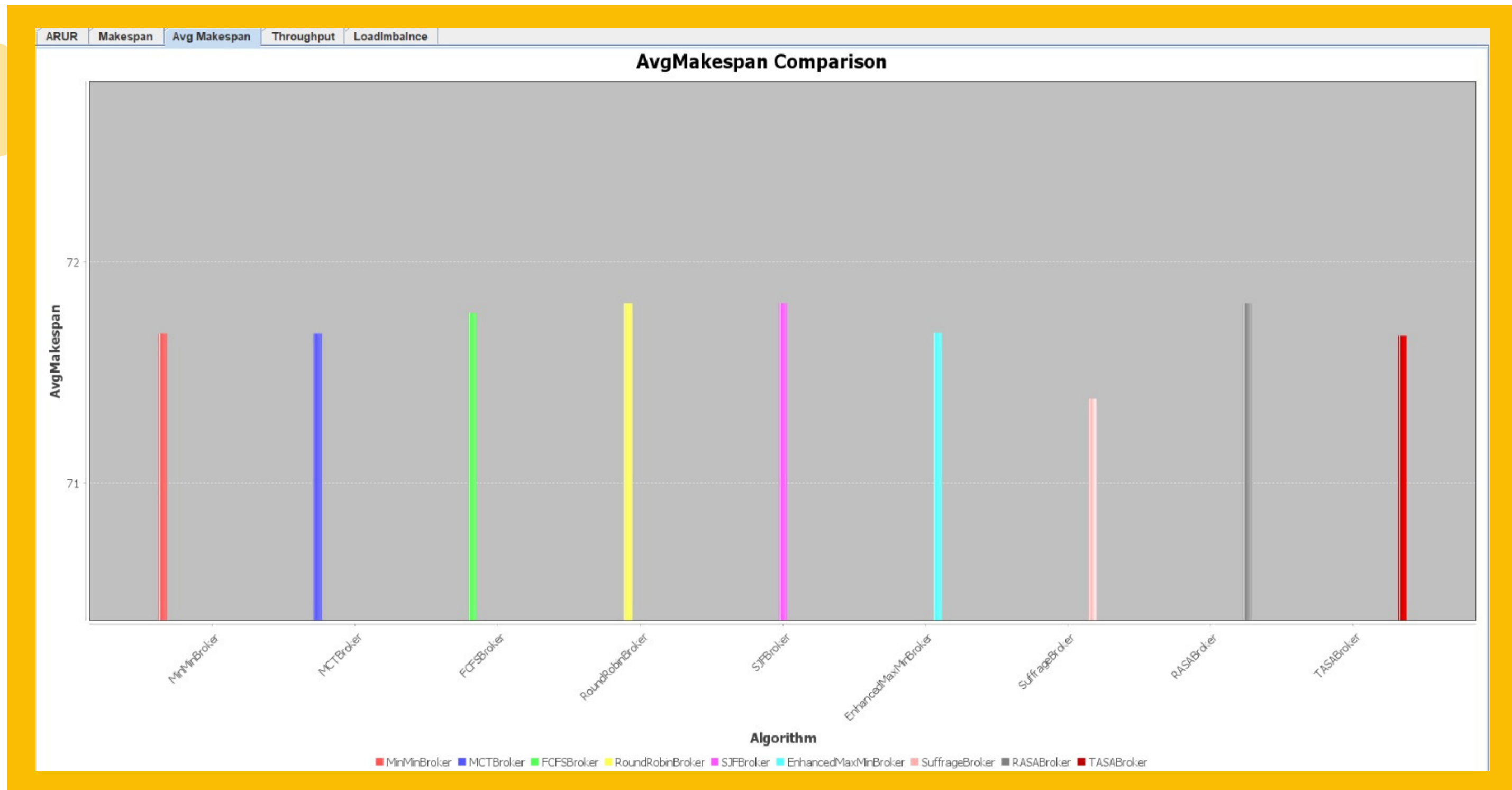
# ARUR



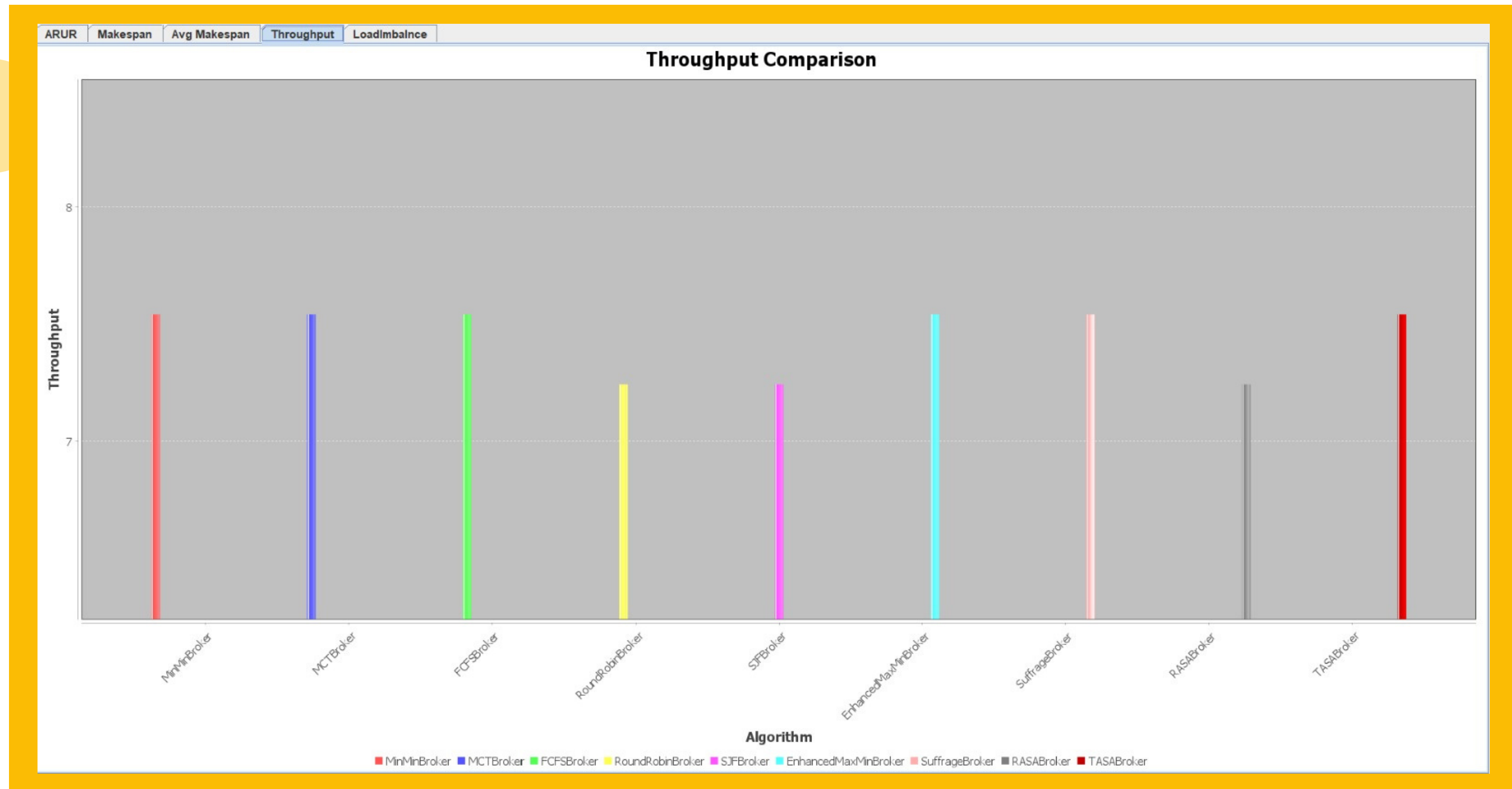
# MAKESPAN



# AVERAGE MAKESPAN



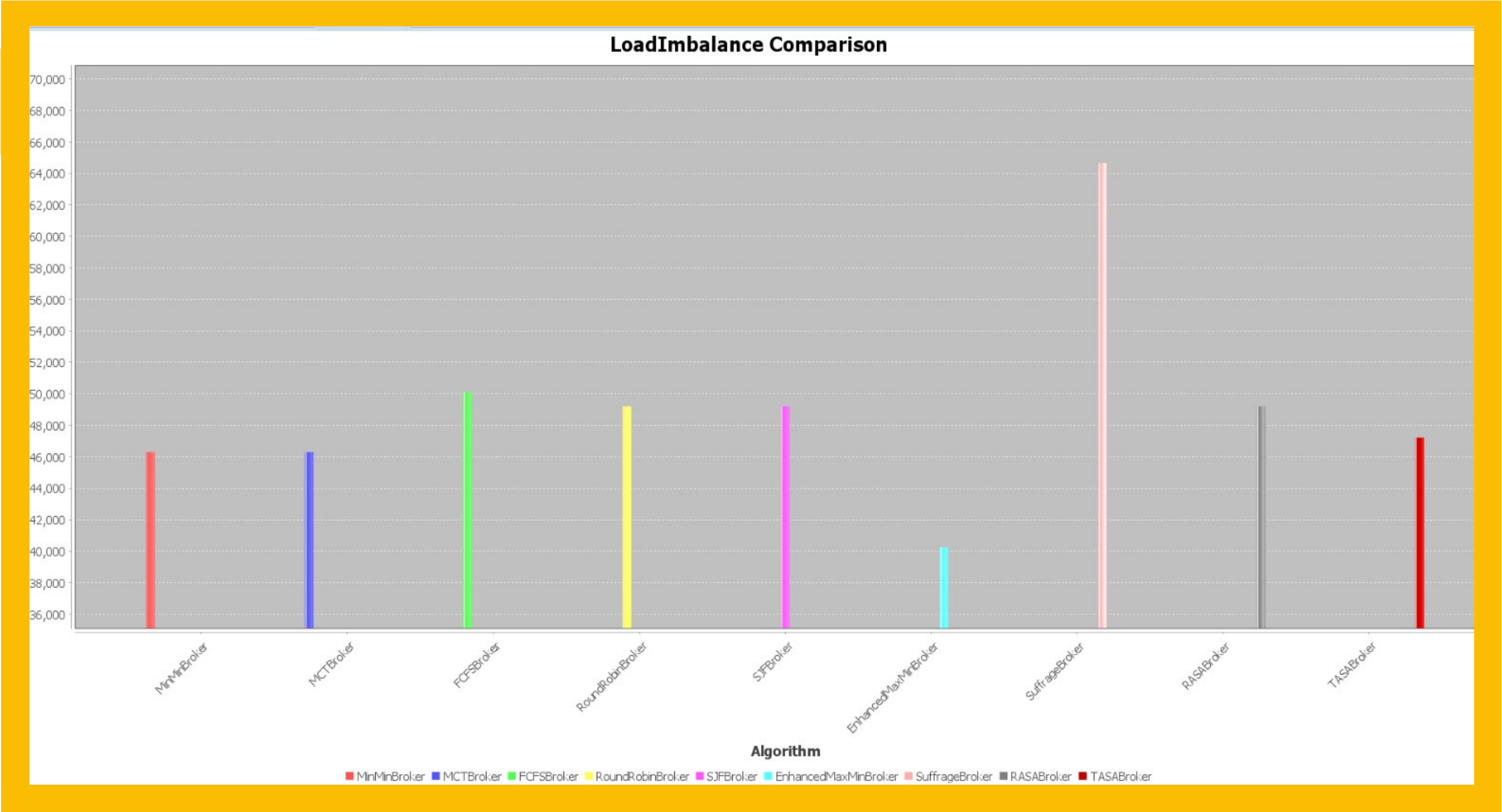
# THROUGHPUT



# INDIVIDUAL LOAD IMBALANCE

VmId	Minmin	MCT	FCFS	RoundRobin	SJFS	MaxMin	Sufferage	RASA	TASA
0	0.601	0.806	0.802	0.941	0.941	0.806	0.822	0.941	0.807
1	0.897	0.897	0.892	0.89	0.89	0.897	0.8	0.89	0.897
2	0.64	0.64	0.636	0.634	0.634	0.64	0.647	0.634	0.633
3	0.706	0.706	0.702	0.7	0.7	0.706	0.619	0.7	0.706
4	0.503	0.503	0.499	0.497	0.497	0.503	0.516	0.497	0.503
5	0.471	0.471	0.467	0.466	0.466	0.471	0.484	0.466	0.471
6	0.643	0.643	0.639	0.637	0.637	0.642	0.657	0.637	0.643
7	0.335	0.335	0.436	0.33	0.33	0.335	0.347	0.33	0.335
8	0.325	0.325	0.322	0.321	0.321	0.325	0.265	0.321	0.326
9	0.152	0.152	0.149	0.148	0.148	0.152	0.131	0.148	0.152
10	0.284	0.284	0.281	0.28	0.28	0.284	0.255	0.28	0.285
11	0.244	0.244	0.241	0.24	0.24	0.244	0.256	0.24	0.245
12	0.05	0.05	0.047	0.046	0.046	0.049	0.059	0.046	0.05
13	0.058	0.058	0.056	0.055	0.055	0.058	0.047	0.055	0.059
14	0.038	0.038	0.041	0.042	0.042	0.038	0.03	0.042	0.038
15	0.061	0.061	0.063	0.064	0.064	0.061	0.052	0.064	0.06
16	0.04	0.04	0.043	0.044	0.044	0.04	0.032	0.044	0.04
17	0.119	0.119	0.121	0.122	0.122	0.119	0.116	0.122	0.123
18	0.199	0.199	0.201	0.202	0.202	0.2	0.192	0.202	0.199
19	0.16	0.16	0.162	0.163	0.163	0.16	0.152	0.163	0.16
20	0.265	0.265	0.267	0.268	0.268	0.265	0.258	0.268	0.265
21	0.293	0.293	0.295	0.296	0.296	0.293	0.287	0.296	0.293
22	0.296	0.296	0.298	0.298	0.298	0.296	0.305	0.298	0.296
23	0.266	0.266	0.268	0.269	0.269	0.266	0.26	0.269	0.266
24	0.367	0.367	0.368	0.369	0.369	0.367	0.361	0.369	0.366
25	0.44	0.44	0.442	0.442	0.442	0.441	0.436	0.442	0.44
26	0.529	0.529	0.53	0.531	0.531	0.529	0.525	0.531	0.529
27	0.576	0.576	0.577	0.577	0.577	0.576	0.586	0.577	0.576
28	0.633	0.633	0.634	0.634	0.634	0.633	0.63	0.634	0.633
29	0.578	0.578	0.579	0.579	0.579	0.578	0.574	0.579	0.577
30	0.634	0.634	0.635	0.635	0.635	0.604	0.642	0.635	0.634
31	0.62	0.62	0.649	0.649	0.649	0.648	0.467	0.649	0.616

# LOAD IMBALANCE





# LIMITATIONS

## **Limited Dataset:**

The limited dataset may not fully represent the diverse nature of real-world cloud computing scenarios, potentially limiting the generalizability of the findings.

## **Single Data Center:**

The simulation in the study considers only a single data center comprised of 32 servers. This may not accurately reflect the complexities and scale of large-scale cloud computing environments with multiple data centers, potentially limiting the applicability of the results.

## **Lack of Energy Consumption Analysis:**

Although the study mentions the need to investigate energy consumption, the specific analysis of energy consumption for the compared approaches is not provided. This omission limits the understanding of the energy efficiency implications of the scheduling algorithms.

## **Limited Performance Metrics:**

The study primarily focuses on execution time (Makespan) and average resource utilization (ARUR) as performance metrics. Other important metrics, such as throughput, SLA violation, and profitability, are mentioned but not extensively analyzed. This limited focus may overlook important aspects of scheduling effectiveness and efficiency.

## **No Discussion on Scalability:**

The document does not explicitly discuss the scalability of the proposed scheduling approaches. The performance and effectiveness of the algorithms in larger-scale cloud environments with a higher number of tasks and resources are not addressed, which may limit their practical applicability in real-world scenarios.

## **No Real-World Deployment:**

The evaluation and analysis of the scheduling algorithms are conducted within a simulation environment (CloudSim). While simulations can provide valuable insights, the lack of real-world deployment and testing may limit the practical applicability and real-world performance of the proposed approaches.





# FUTURE SCOPE

**New Algorithm Testing and Innovation:** Investigating and developing more efficient algorithms and approaches to addressing load imbalance issues in the cloud computing environments.

**Energy consumption optimization:** Exploring methods to reduce energy consumption in cloud data centers, such as optimizing resource allocation and workload distribution.

**Performance evaluation:** Conduct more empirical evaluations and comparative analyses of scheduling approaches using different datasets and benchmarks to gain further insights into their performance and effectiveness.

**SLA violation and profitability analysis:** Investigating the impact of scheduling approaches on SLA violation rates and profitability for cloud service providers, and developing strategies to minimize SLA violations and maximize profitability.





# REFERENCES

## **Main Research Paper**

M. Ibrahim et al., "An In-Depth Empirical Investigation of State-of-the-Art Scheduling Approaches for Cloud Computing," in IEEE Access, vol. 8, pp. 128282–128294, 2020, doi: 10.1109/ACCESS.2020.3007201.

## **Max-Min Algorithm Reference**

Pradhan, Pandaba & Behera, Prafulla & Ray, B. (2020). Enhanced Max-Min Algorithm For Resource Allocation In Cloud Computing. 29. 1619–1628.

## **Task Scheduling Algorithms**

T. Aladwani, 'Types of Task Scheduling Algorithms in Cloud Computing Environment', Scheduling Problems – New Applications and Trends. IntechOpen, Jul. 08, 2020. doi: 10.5772/intechopen.86873.

[oai:ojs.journal.utem.edu.my:article/1243](https://ojs.journal.utem.edu.my/article/1243)

## **TASA Algorithm**

taherian dehkordi, Somayeh & Bardsiri, Vahid. (2015). TASA: A New Task Scheduling Algorithm in Cloud Computing. Advances in Computer Engineering and Technology.

[https://www.ijcseonline.org/pub\\_paper/6-IJCSE-04551.pdf](https://www.ijcseonline.org/pub_paper/6-IJCSE-04551.pdf)



# INDIVIDUAL CONTRIBUTION

## **Aarav Nigam – S20210010002**

Implemented Shortest job First, Round Robin Algorithm and TASA Algorithms  
Graph Plotting

## **Akansh Vaibhav – S20210010010**

Implemented First Come First Serve Algorithm and RASA.  
Presentation and Report

## **Akash Singh Narvariya – S20210010012**

Implemented Maximum Computing Time and Enhanced Max-Min Algorithm  
Worked with Parameters

## **Pranav Singh – S20210010180**

Implemented Min-Min Algorithm and Suffarage Algorithm  
Java Code Base



THANK YOU