NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Cloud Computing Report

A Greedy Load Balancing Algorithm for FaaS Platforms

Team Members:

Srujan S Bharadwaj - 191CS254

V Madhan Kumar - 191CS260

Vinesh S Talpankar -191CS265

Submitted to:

Prof. Sourav Kranti Addya

Abstract:

The main goal of this paper is to improve the performance of Function as a Service (FaaS) by introducing a new and efficient load-balancing algorithm called GReedy Algorithm for Faas platforms (GRAF) which tries to maximize the locality by increasing the cache-hit ratio as operations like virtualization and initialization consumes heavy resources which can be reduced using GRAF load balancing algorithm.

Introduction:

In short, we try to move the computation where the data resides rather than vice versa, like we check if the node has the cache which is required and then push the computation, rather than moving the data first and later computing over it which requires more time.

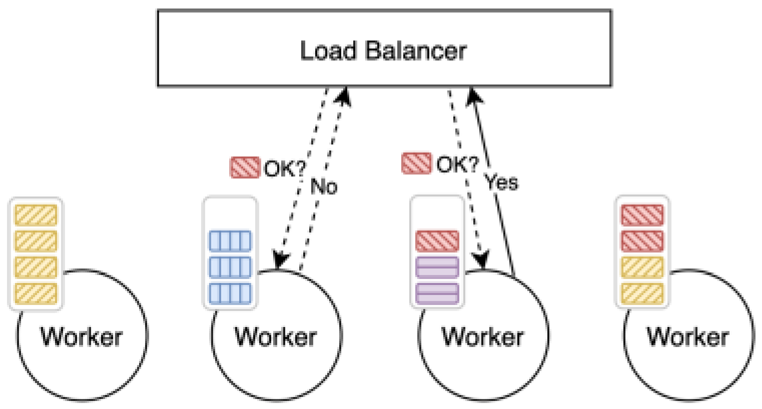


Figure 1: Simplified concept of GRAF, where worker nodes can reject the tasks in order to maximize the locality.

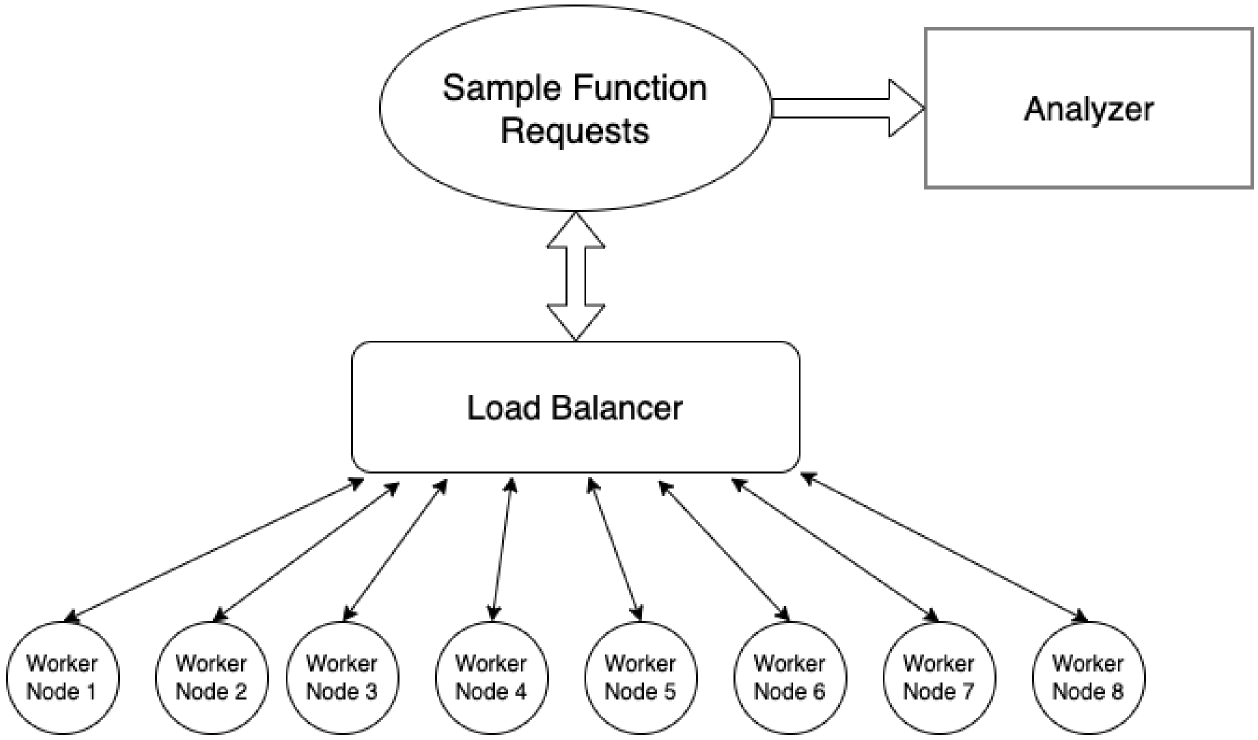
If you notice Worker 2, it doesn't have the red packet hence it rejects the task entirely, whereas Worker 3 has the red packet in its cache therefore computation will be faster in this case. Even Worker 4 has multiple red packets but its buffer is full therefore we stick to Worker 3 to finish the necessary task.

Why GRAF is better than other load-balancing algorithms:

We try to balance the load as well as take locality into account in our approach, the traditional techniques like Round Robin, and Least Loaded just consider load balancing and not locality and Consistent hashing just considers locality which leads to load imbalance. Later on, people introduced Consistent hashing with bounded load to provide better load balance and locality but it fails to do so.
We ensure GRAF takes both locality and load balancing into consideration to improve the performance of FaaS platforms.

Workflow:

Flowchart of functions being run through the GRaF algorithm.



Proposed Design for the GReedy Algorithm:

1. Tabular Data Structure:

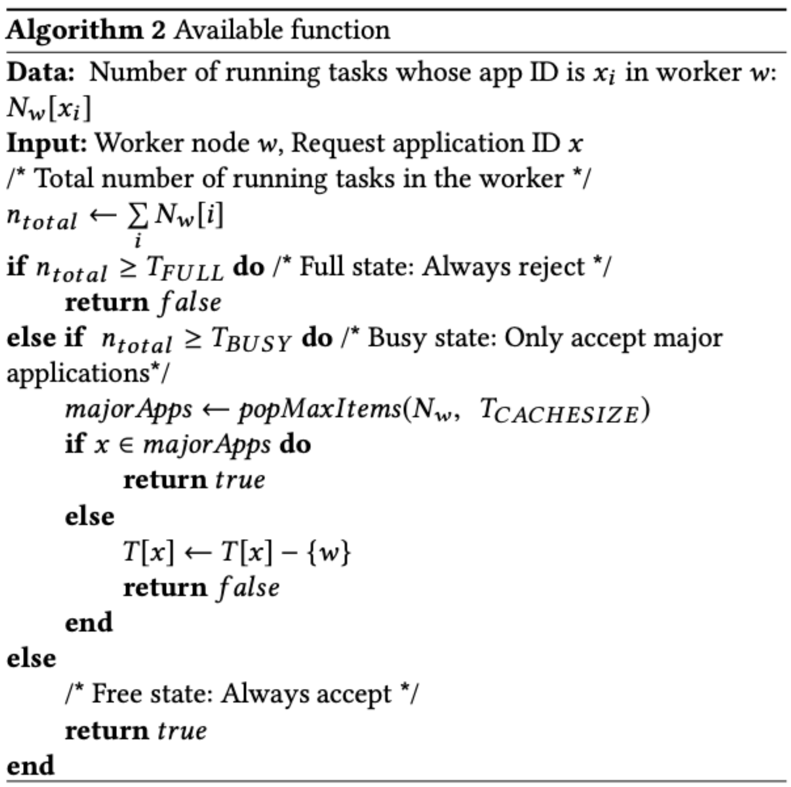
   AssignedTable contains information about which nodes are responsible for the application.

2. Algorithm 1:

```
Algorithm 1 Worker Node Selection
Data: List of worker nodes $W$,
Table consists of assigned workers $T$
Input: Application ID $x$
$W' \leftarrow T[x]$ /*Assigned worker node set */
$W^* \leftarrow available(W', x)$ /* Available nodes in W' */
if $W^* \neq \emptyset$ do
    return $leastLoaded(W^*)$
else
    $w \leftarrow leastLoaded(available(W, x))$
    if $w$ is $null$ do
        $w \leftarrow leastLoaded(W)$
    end
    $T[x] \leftarrow T[x] \cup \{w\}$
    return $w$
end
```

- Worker Node Selection is our basic node selection process by using the AssignedTable.
- If nodes are assigned to the application x and any of the assigned nodes are available, the node with the least loads is selected.
- If no node is assigned, the load balancer picks the least loaded node and registers it to the table.
- It does not include a process for deleting entries in the AssignedTable since we redefine the available function from the worker's viewpoint.

3. Algorithm 2:



Algorithm 2 Available function

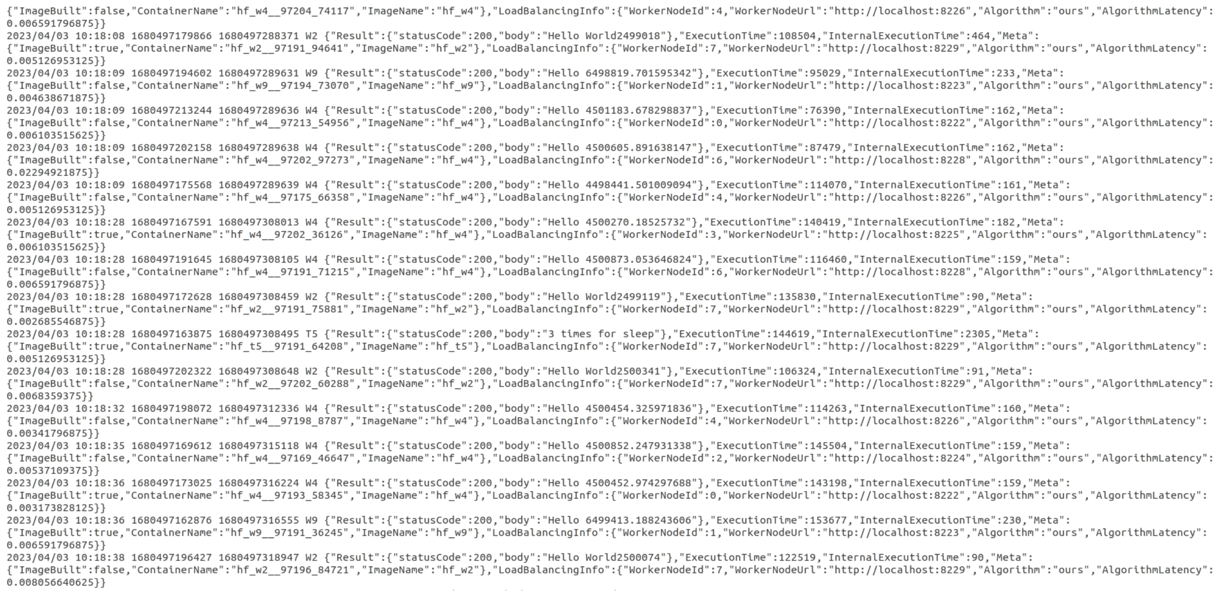
```
Data: Number of running tasks whose app ID is $x_i$ in worker $w$: $N_w[x_i]$
Input: Worker node $w$, Request application ID $x$
/* Total number of running tasks in the worker */
$n_{total} \leftarrow \sum_i N_w[i]$
if $n_{total} \geq T_{FULL}$ do /* Full state: Always reject */
    return $false$
else if $n_{total} \geq T_{BUSY}$ do /* Busy state: Only accept major applications*/
    $majorApps \leftarrow popMaxItems(N_w,\ T_{CACHESIZE})$
    if $x \in majorApps$ do
        return $true$
    else
        $T[x] \leftarrow T[x] - \{w\}$
        return $false$
    end
else
    /* Free state: Always accept */
    return $true$
end
```

- Available function provides a greedy approach to optimize the AssignedTable.
- We introduce three states of the worker: full, busy, and free:
- The full state indicates too many running tasks, so a new task cannot be run on the node.
- The busy state is an intermediate state where the worker is not full, although many tasks are running.
- In the free state, the worker can afford to accept a new task.
- When the worker node receives a new task, it makes different decisions based on these states. In the full state, the request is always rejected. In the free state, workers always accept the task. In the busy state, workers selectively accept the task through their own decisions.
- From the worker's viewpoint, the best way to maximize the locality is to receive the same types of applications as much as possible.
- So busy workers accept the task only if it is one of the major applications, which is a set of most running applications in the worker.

- If the worker node rejects the application in the busy state, it is also removed from the AssignedTable.
- Consequently, the entries in the table will no longer grow indefinitely and the locality could be further improved.

Simulator Log:

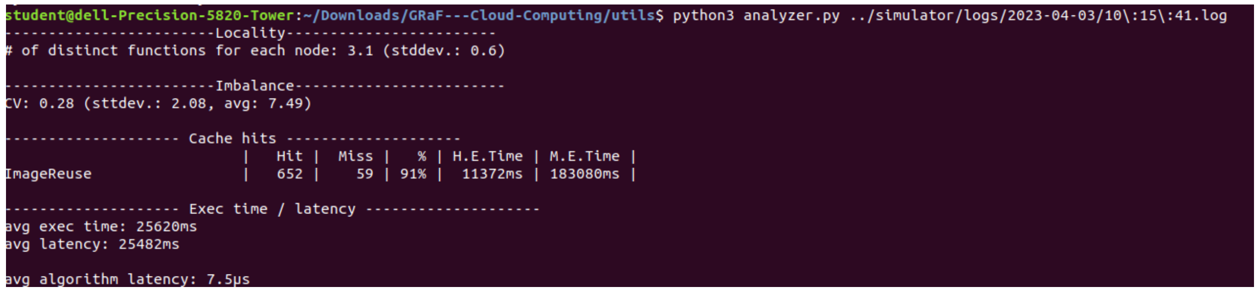
This log file is generated by running the simulator, which takes the events file as an input (list of sample functions) and displays which sample function is being assigned to which worker node respectively using the GRaF algorithm.



```
{"ImageBuilt":false,"ContainerName":"hf_w4__97204_74117","ImageName":"hf_w4"},"LoadBalancingInfo":{"WorkerNodeId":4,"WorkerNodeUrl":"http://localhost:8226","Algorithm":"ours","AlgorithmLatency":
0.006591796875}}
2023/04/03 10:18:08 1680497179866 1680497288371 W2 {"Result":{"statusCode":200,"body":"Hello World2499018"},"ExecutionTime":108504,"InternalExecutionTime":464,"Meta":
{"ImageBuilt":true,"ContainerName":"hf_w2__97191_94641","ImageName":"hf_w2"},"LoadBalancingInfo":{"WorkerNodeId":7,"WorkerNodeUrl":"http://localhost:8229","Algorithm":"ours","AlgorithmLatency":
0.005126953125}}
2023/04/03 10:18:09 1680497194602 1680497289631 W9 {"Result":{"statusCode":200,"body":"Hello 6498819.701595342"},"ExecutionTime":95029,"InternalExecutionTime":233,"Meta":
{"ImageBuilt":false,"ContainerName":"hf_w9__97194_73070","ImageName":"hf_w9"},"LoadBalancingInfo":{"WorkerNodeId":1,"WorkerNodeUrl":"http://localhost:8223","Algorithm":"ours","AlgorithmLatency":
0.004638671875}}
2023/04/03 10:18:09 1680497213244 1680497289636 W4 {"Result":{"statusCode":200,"body":"Hello 4501183.678298837"},"ExecutionTime":76390,"InternalExecutionTime":162,"Meta":
{"ImageBuilt":false,"ContainerName":"hf_w4__97213_54956","ImageName":"hf_w4"},"LoadBalancingInfo":{"WorkerNodeId":0,"WorkerNodeUrl":"http://localhost:8222","Algorithm":"ours","AlgorithmLatency":
0.006103515625}}
2023/04/03 10:18:09 1680497202158 1680497289638 W4 {"Result":{"statusCode":200,"body":"Hello 4500605.891638147"},"ExecutionTime":87479,"InternalExecutionTime":162,"Meta":
{"ImageBuilt":false,"ContainerName":"hf_w4__97202_97273","ImageName":"hf_w4"},"LoadBalancingInfo":{"WorkerNodeId":6,"WorkerNodeUrl":"http://localhost:8228","Algorithm":"ours","AlgorithmLatency":
0.02294921875}}
2023/04/03 10:18:09 1680497175568 1680497289639 W4 {"Result":{"statusCode":200,"body":"Hello 4498441.501009094"},"ExecutionTime":114070,"InternalExecutionTime":161,"Meta":
{"ImageBuilt":false,"ContainerName":"hf_w4__97175_66358","ImageName":"hf_w4"},"LoadBalancingInfo":{"WorkerNodeId":4,"WorkerNodeUrl":"http://localhost:8226","Algorithm":"ours","AlgorithmLatency":
0.005126953125}}
2023/04/03 10:18:28 1680497167591 1680497308013 W4 {"Result":{"statusCode":200,"body":"Hello 4500270.18525732"},"ExecutionTime":140419,"InternalExecutionTime":182,"Meta":
{"ImageBuilt":true,"ContainerName":"hf_w4__97202_36126","ImageName":"hf_w4"},"LoadBalancingInfo":{"WorkerNodeId":3,"WorkerNodeUrl":"http://localhost:8225","Algorithm":"ours","AlgorithmLatency":
0.006103515625}}
2023/04/03 10:18:28 1680497191645 1680497308105 W4 {"Result":{"statusCode":200,"body":"Hello 4500873.053646824"},"ExecutionTime":116460,"InternalExecutionTime":159,"Meta":
{"ImageBuilt":false,"ContainerName":"hf_w4__97191_71215","ImageName":"hf_w4"},"LoadBalancingInfo":{"WorkerNodeId":6,"WorkerNodeUrl":"http://localhost:8228","Algorithm":"ours","AlgorithmLatency":
0.006591796875}}
2023/04/03 10:18:28 1680497172628 1680497308459 W2 {"Result":{"statusCode":200,"body":"Hello World2499119"},"ExecutionTime":135830,"InternalExecutionTime":90,"Meta":
{"ImageBuilt":true,"ContainerName":"hf_w2__97191_75881","ImageName":"hf_w2"},"LoadBalancingInfo":{"WorkerNodeId":7,"WorkerNodeUrl":"http://localhost:8229","Algorithm":"ours","AlgorithmLatency":
0.002685546875}}
2023/04/03 10:18:28 1680497163875 1680497308495 T5 {"Result":{"statusCode":200,"body":"3 times for sleep"},"ExecutionTime":144619,"InternalExecutionTime":2305,"Meta":
{"ImageBuilt":true,"ContainerName":"hf_t5__97191_64208","ImageName":"hf_t5"},"LoadBalancingInfo":{"WorkerNodeId":7,"WorkerNodeUrl":"http://localhost:8229","Algorithm":"ours","AlgorithmLatency":
0.005126953125}}
2023/04/03 10:18:28 1680497202322 1680497308648 W2 {"Result":{"statusCode":200,"body":"Hello World2500341"},"ExecutionTime":106324,"InternalExecutionTime":91,"Meta":
{"ImageBuilt":false,"ContainerName":"hf_w2__97202_60288","ImageName":"hf_w2"},"LoadBalancingInfo":{"WorkerNodeId":7,"WorkerNodeUrl":"http://localhost:8229","Algorithm":"ours","AlgorithmLatency":
0.0068359375}}
2023/04/03 10:18:32 1680497198072 1680497312336 W4 {"Result":{"statusCode":200,"body":"Hello 4500454.325971836"},"ExecutionTime":114263,"InternalExecutionTime":160,"Meta":
{"ImageBuilt":false,"ContainerName":"hf_w4__97198_8787","ImageName":"hf_w4"},"LoadBalancingInfo":{"WorkerNodeId":4,"WorkerNodeUrl":"http://localhost:8226","Algorithm":"ours","AlgorithmLatency":
0.00341796875}}
2023/04/03 10:18:35 1680497169612 1680497315118 W4 {"Result":{"statusCode":200,"body":"Hello 4500852.247931338"},"ExecutionTime":145504,"InternalExecutionTime":159,"Meta":
{"ImageBuilt":false,"ContainerName":"hf_w4__97169_46647","ImageName":"hf_w4"},"LoadBalancingInfo":{"WorkerNodeId":2,"WorkerNodeUrl":"http://localhost:8224","Algorithm":"ours","AlgorithmLatency":
0.00537109375}}
2023/04/03 10:18:36 1680497173025 1680497316224 W4 {"Result":{"statusCode":200,"body":"Hello 4500452.974297688"},"ExecutionTime":143198,"InternalExecutionTime":159,"Meta":
{"ImageBuilt":true,"ContainerName":"hf_w4__97193_58345","ImageName":"hf_w4"},"LoadBalancingInfo":{"WorkerNodeId":0,"WorkerNodeUrl":"http://localhost:8222","Algorithm":"ours","AlgorithmLatency":
0.003173828125}}
2023/04/03 10:18:36 1680497162876 1680497316555 W9 {"Result":{"statusCode":200,"body":"Hello 6499413.188243606"},"ExecutionTime":153677,"InternalExecutionTime":230,"Meta":
{"ImageBuilt":true,"ContainerName":"hf_w9__97191_36245","ImageName":"hf_w9"},"LoadBalancingInfo":{"WorkerNodeId":1,"WorkerNodeUrl":"http://localhost:8223","Algorithm":"ours","AlgorithmLatency":
0.006591796875}}
2023/04/03 10:18:38 1680497196427 1680497318947 W2 {"Result":{"statusCode":200,"body":"Hello World2500074"},"ExecutionTime":122519,"InternalExecutionTime":90,"Meta":
{"ImageBuilt":false,"ContainerName":"hf_w2__97196_84721","ImageName":"hf_w2"},"LoadBalancingInfo":{"WorkerNodeId":7,"WorkerNodeUrl":"http://localhost:8229","Algorithm":"ours","AlgorithmLatency":
0.008056640625}}
```

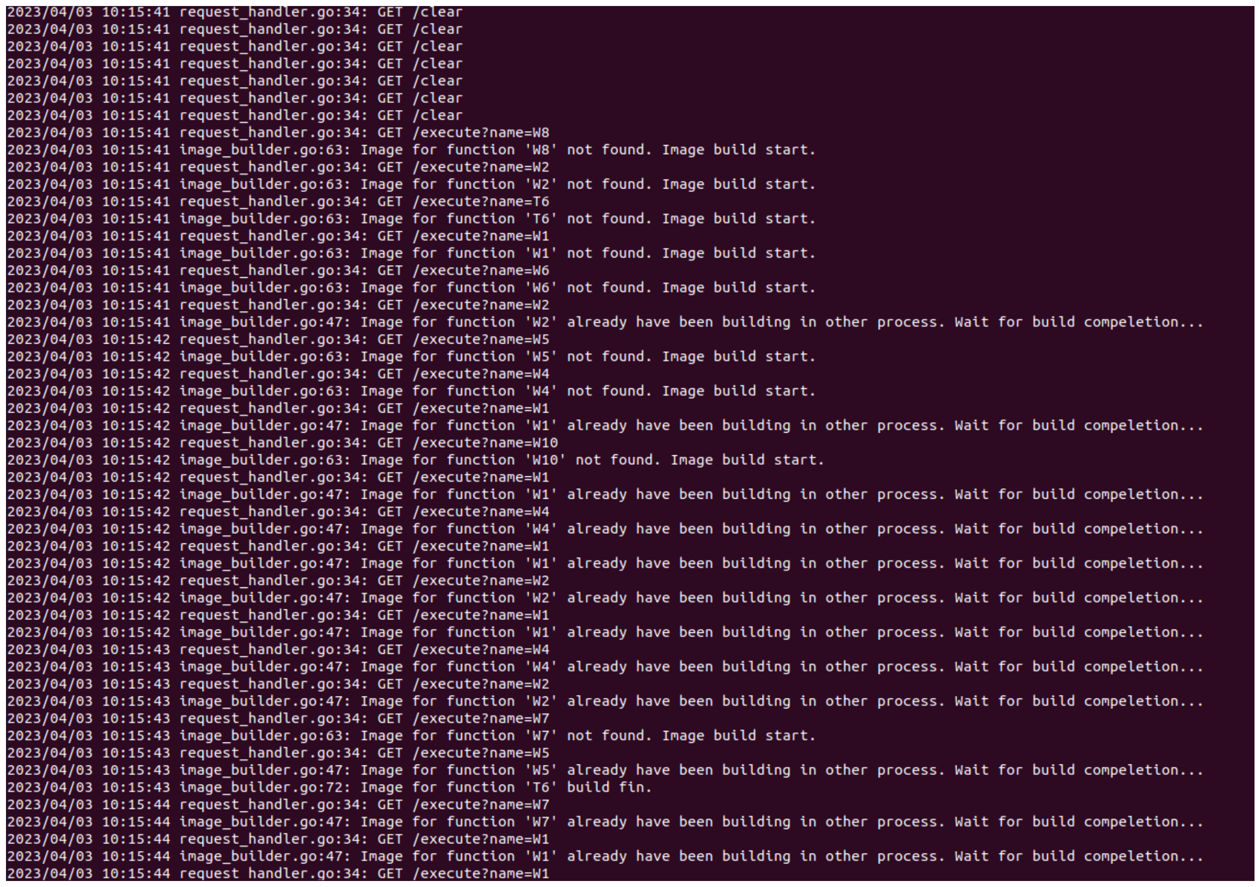
Results:

The simulator log is analyzed using the analyzer, which generates cache hit ratio, average latency, average algorithm latency, and average function execution time as shown in the screenshot below.



```
student@dell-Precision-5820-Tower:~/Downloads/GRaF---Cloud-Computing/utils$ python3 analyzer.py ../simulator/logs/2023-04-03/10\:15\:41.log
------------------------Locality------------------------
# of distinct functions for each node: 3.1 (stddev.: 0.6)

------------------------Imbalance------------------------
CV: 0.28 (sttdev.: 2.08, avg: 7.49)

-------------------- Cache hits --------------------
                           |   Hit |  Miss |   % | H.E.Time | M.E.Time |
ImageReuse                 |   652 |    59 | 91% |  11372ms | 183080ms |

-------------------- Exec time / latency --------------------
avg exec time: 25620ms
avg latency: 25482ms

avg algorithm latency: 7.5μs
```

Output of worker nodes:



```
2023/04/03 10:15:41 request_handler.go:34: GET /clear
2023/04/03 10:15:41 request_handler.go:34: GET /clear
2023/04/03 10:15:41 request_handler.go:34: GET /clear
2023/04/03 10:15:41 request_handler.go:34: GET /clear
2023/04/03 10:15:41 request_handler.go:34: GET /clear
2023/04/03 10:15:41 request_handler.go:34: GET /clear
2023/04/03 10:15:41 request_handler.go:34: GET /clear
2023/04/03 10:15:41 request_handler.go:34: GET /execute?name=W8
2023/04/03 10:15:41 image_builder.go:63: Image for function 'W8' not found. Image build start.
2023/04/03 10:15:41 request_handler.go:34: GET /execute?name=W2
2023/04/03 10:15:41 image_builder.go:63: Image for function 'W2' not found. Image build start.
2023/04/03 10:15:41 request_handler.go:34: GET /execute?name=T6
2023/04/03 10:15:41 image_builder.go:63: Image for function 'T6' not found. Image build start.
2023/04/03 10:15:41 request_handler.go:34: GET /execute?name=W1
2023/04/03 10:15:41 image_builder.go:63: Image for function 'W1' not found. Image build start.
2023/04/03 10:15:41 request_handler.go:34: GET /execute?name=W6
2023/04/03 10:15:41 image_builder.go:63: Image for function 'W6' not found. Image build start.
2023/04/03 10:15:41 request_handler.go:34: GET /execute?name=W2
2023/04/03 10:15:41 image_builder.go:47: Image for function 'W2' already have been building in other process. Wait for build compeletion...
2023/04/03 10:15:42 request_handler.go:34: GET /execute?name=W5
2023/04/03 10:15:42 image_builder.go:63: Image for function 'W5' not found. Image build start.
2023/04/03 10:15:42 request_handler.go:34: GET /execute?name=W4
2023/04/03 10:15:42 image_builder.go:63: Image for function 'W4' not found. Image build start.
2023/04/03 10:15:42 request_handler.go:34: GET /execute?name=W1
2023/04/03 10:15:42 image_builder.go:47: Image for function 'W1' already have been building in other process. Wait for build compeletion...
2023/04/03 10:15:42 request_handler.go:34: GET /execute?name=W10
2023/04/03 10:15:42 image_builder.go:63: Image for function 'W10' not found. Image build start.
2023/04/03 10:15:42 request_handler.go:34: GET /execute?name=W1
2023/04/03 10:15:42 image_builder.go:47: Image for function 'W1' already have been building in other process. Wait for build compeletion...
2023/04/03 10:15:42 request_handler.go:34: GET /execute?name=W4
2023/04/03 10:15:42 image_builder.go:47: Image for function 'W4' already have been building in other process. Wait for build compeletion...
2023/04/03 10:15:42 request_handler.go:34: GET /execute?name=W1
2023/04/03 10:15:42 image_builder.go:47: Image for function 'W1' already have been building in other process. Wait for build compeletion...
2023/04/03 10:15:42 request_handler.go:34: GET /execute?name=W2
2023/04/03 10:15:42 image_builder.go:47: Image for function 'W2' already have been building in other process. Wait for build compeletion...
2023/04/03 10:15:42 request_handler.go:34: GET /execute?name=W1
2023/04/03 10:15:42 image_builder.go:47: Image for function 'W1' already have been building in other process. Wait for build compeletion...
2023/04/03 10:15:43 request_handler.go:34: GET /execute?name=W4
2023/04/03 10:15:43 image_builder.go:47: Image for function 'W4' already have been building in other process. Wait for build compeletion...
2023/04/03 10:15:43 request_handler.go:34: GET /execute?name=W2
2023/04/03 10:15:43 image_builder.go:47: Image for function 'W2' already have been building in other process. Wait for build compeletion...
2023/04/03 10:15:43 request_handler.go:34: GET /execute?name=W7
2023/04/03 10:15:43 image_builder.go:63: Image for function 'W7' not found. Image build start.
2023/04/03 10:15:43 request_handler.go:34: GET /execute?name=W5
2023/04/03 10:15:43 image_builder.go:47: Image for function 'W5' already have been building in other process. Wait for build compeletion...
2023/04/03 10:15:43 image_builder.go:72: Image for function 'T6' build fin.
2023/04/03 10:15:44 request_handler.go:34: GET /execute?name=W7
2023/04/03 10:15:44 image_builder.go:47: Image for function 'W7' already have been building in other process. Wait for build compeletion...
2023/04/03 10:15:44 request_handler.go:34: GET /execute?name=W1
2023/04/03 10:15:44 image_builder.go:47: Image for function 'W1' already have been building in other process. Wait for build compeletion...
2023/04/03 10:15:44 request_handler.go:34: GET /execute?name=W1
```

References:

1. https://dl.acm.org/doi/10.1145/3481646.3481657