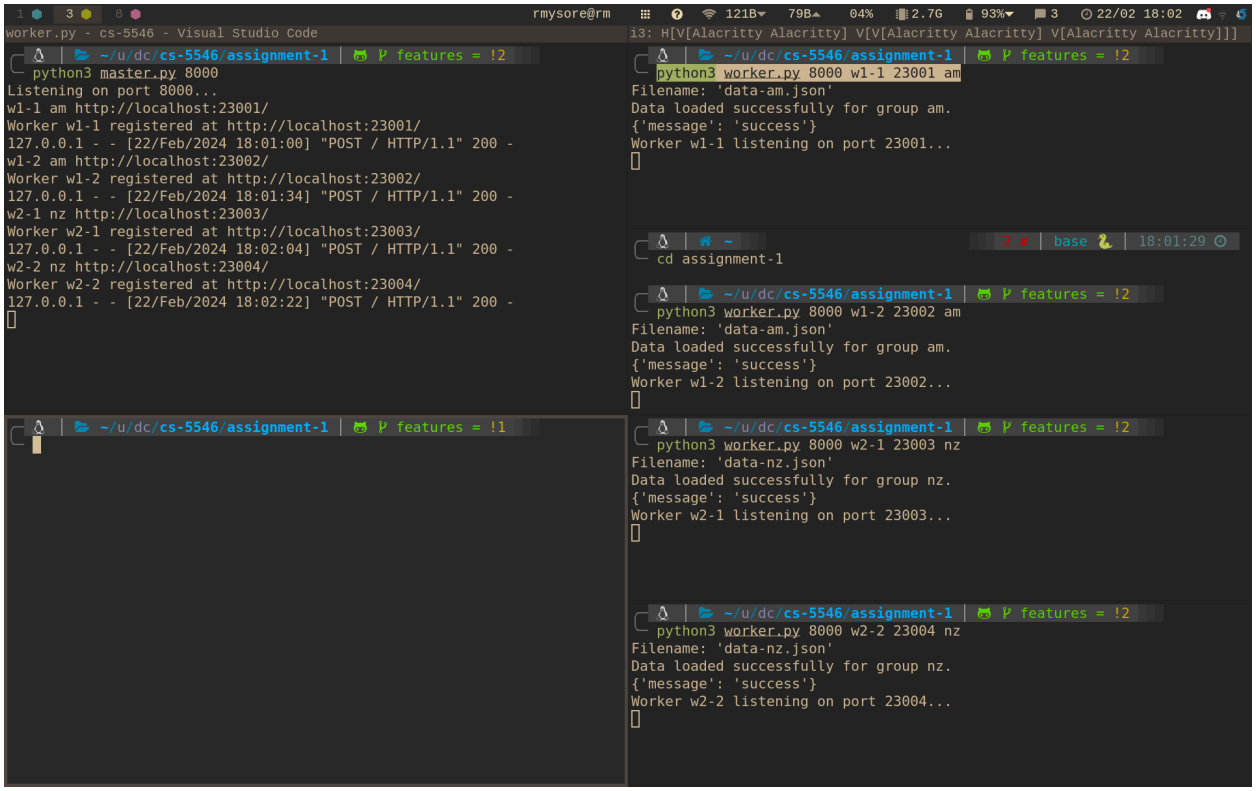
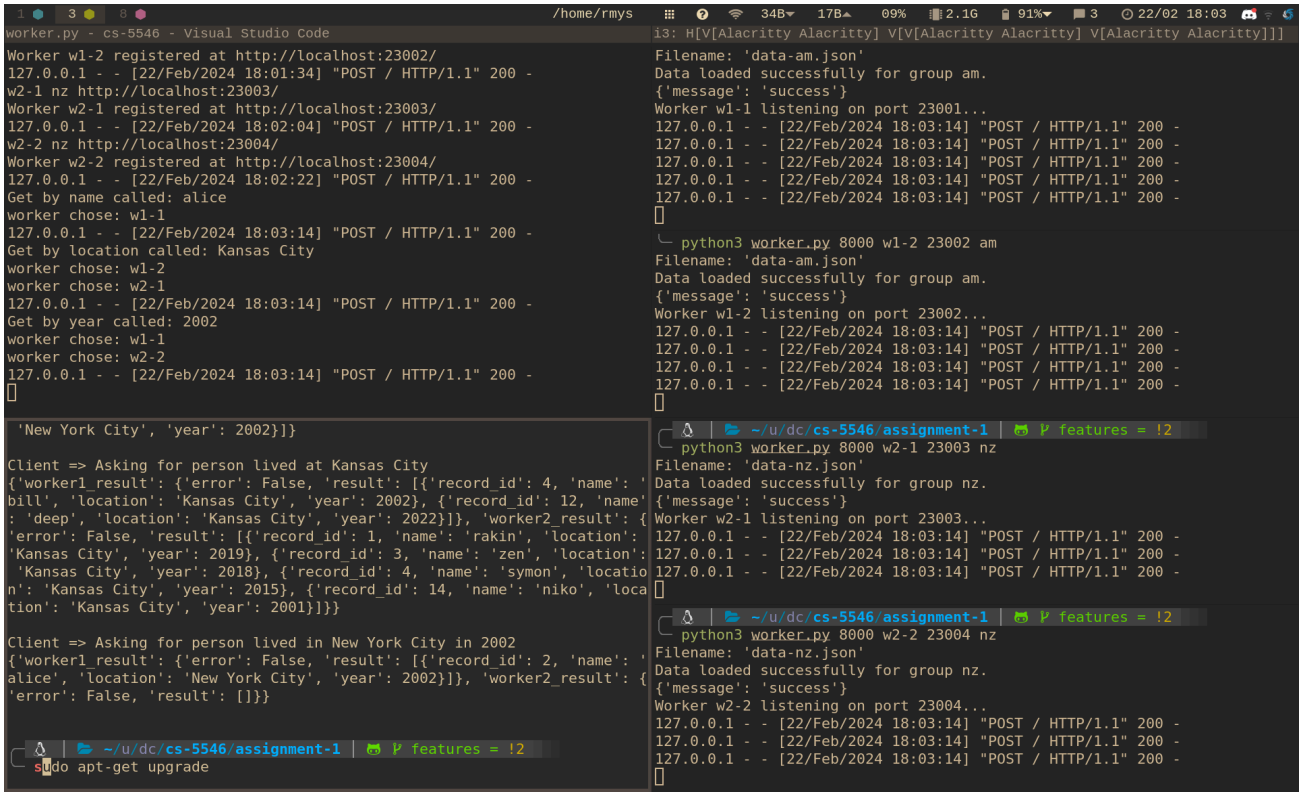
CS-5546 - Distributed master-worker system
GROUP - 2

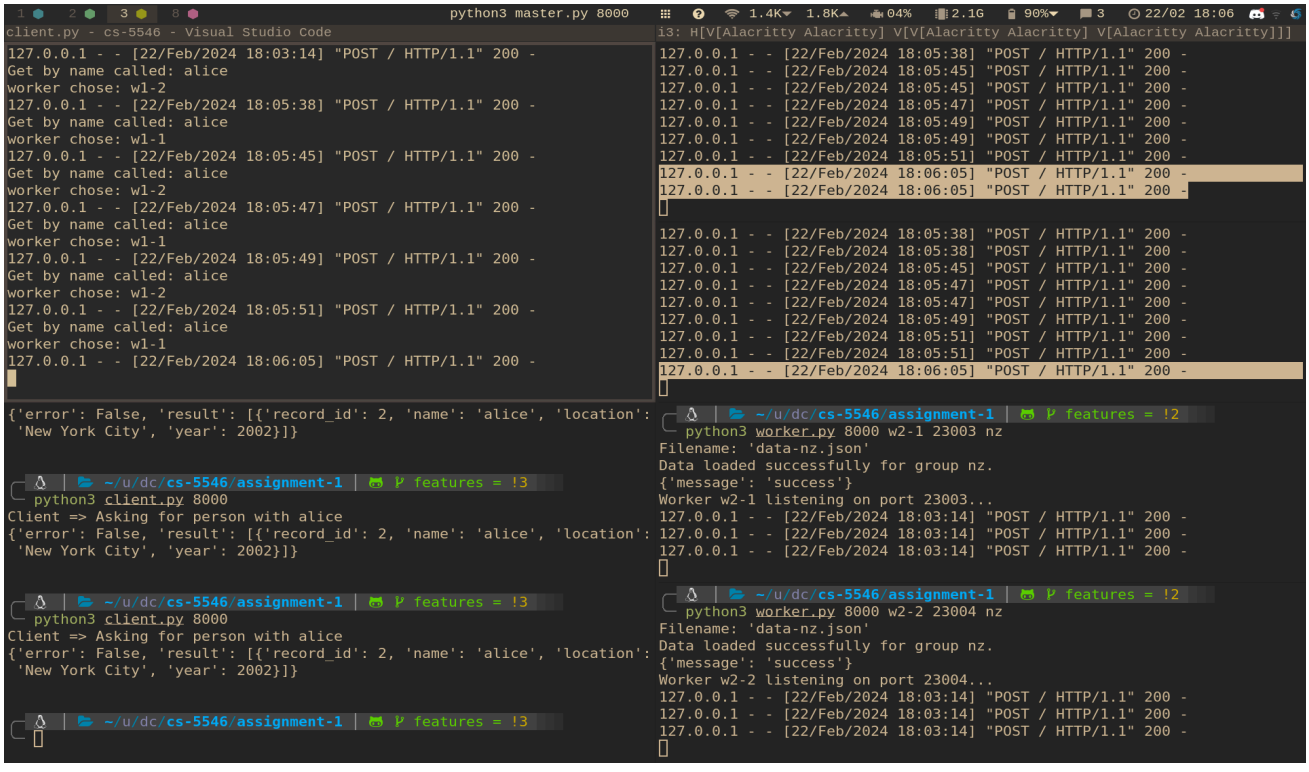
Note: Video demo is added to the zip file

Extra Feature 1: As shown in the below screenshot, workers are registering with the Master server. We can register any required instances for each worker with the Master server.

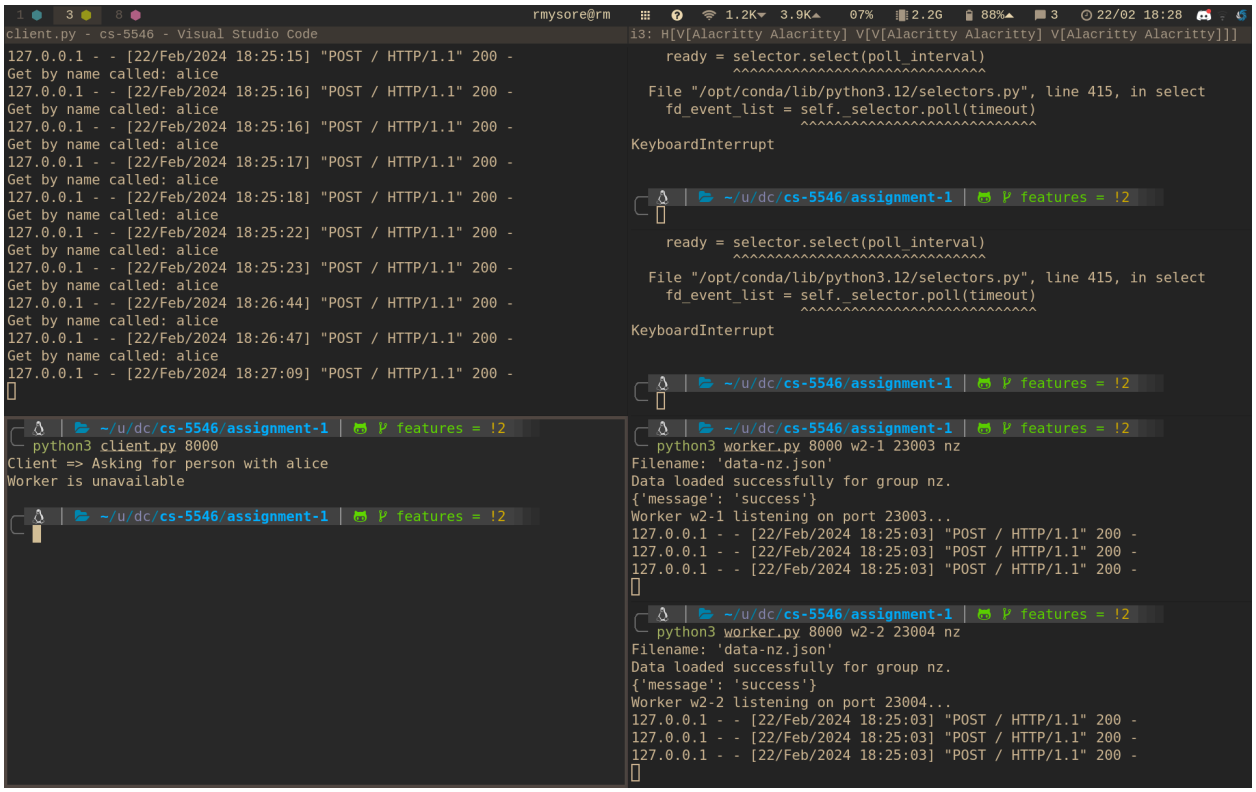


Extra feature 2: In this setup, the master server dynamically balances the load of requests among available worker instances. When a client makes an RPC call, such as getbyname(name), the master server first determines which worker (worker 1 or worker 2) should handle the request based on the first letter of the requested name. Then, among the available instances of that worker, the master server assesses the workload of each instance. Finally, it selects the worker instance with the least number of served requests to process the client's request. This approach optimizes resource utilisation and ensures efficient handling of incoming requests across the worker instances. When a client makes an RPC call, such as getbylocation(location) or getbyyear(location, year), the master server sends the request to both workers but chooses only one instance among the workers which serve a minimum number of requests.


worker.py - cs-5546 - Visual Studio Code
Worker w1-2 registered at http://localhost:23002/
127.0.0.1 - - [22/Feb/2024 18:01:34] "POST / HTTP/1.1" 200 -
w2-1 nz http://localhost:23003/
Worker w2-1 registered at http://localhost:23003/
127.0.0.1 - - [22/Feb/2024 18:02:04] "POST / HTTP/1.1" 200 -
w2-2 nz http://localhost:23004/
Worker w2-2 registered at http://localhost:23004/
127.0.0.1 - - [22/Feb/2024 18:02:22] "POST / HTTP/1.1" 200 -
Get by name called: alice
worker chose: w1-1
127.0.0.1 - - [22/Feb/2024 18:03:14] "POST / HTTP/1.1" 200 -
Get by location called: Kansas City
worker chose: w1-2
worker chose: w2-1
127.0.0.1 - - [22/Feb/2024 18:03:14] "POST / HTTP/1.1" 200 -
Get by year called: 2002
worker chose: w1-1
worker chose: w2-2
127.0.0.1 - - [22/Feb/2024 18:03:14] "POST / HTTP/1.1" 200 -
'New York City', 'year': 2002}]}
Client => Asking for person lived at Kansas City
{'worker1_result': {'error': False, 'result': [{'record_id': 4, 'name': '
bill', 'location': 'Kansas City', 'year': 2002}, {'record_id': 12, 'name'
: 'deep', 'location': 'Kansas City', 'year': 2022}]}, 'worker2_result': {
'error': False, 'result': [{'record_id': 1, 'name': 'rakin', 'location':
'Kansas City', 'year': 2019}, {'record_id': 3, 'name': 'zen', 'location':
'Kansas City', 'year': 2018}, {'record_id': 4, 'name': 'symon', 'locatio
n': 'Kansas City', 'year': 2015}, {'record_id': 14, 'name': 'niko', 'loca
tion': 'Kansas City', 'year': 2001}]}}
Client => Asking for person lived in New York City in 2002
{'worker1_result': {'error': False, 'result': [{'record_id': 2, 'name':
alice', 'location': 'New York City', 'year': 2002}]}, 'worker2_result': {
'error': False, 'result': []}}
~/u/dc/cs-5546/assignment-1 | features = !2
sudo apt-get upgrade
i3: H[V[Alacritty Alacritty] V[V[Alacritty Alacritty] V[Alacritty Alacritty]]]
Filename: 'data-am.json'
Data loaded successfully for group am.
{'message': 'success'}
Worker w1-1 listening on port 23001...
127.0.0.1 - - [22/Feb/2024 18:03:14] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [22/Feb/2024 18:03:14] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [22/Feb/2024 18:03:14] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [22/Feb/2024 18:03:14] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [22/Feb/2024 18:03:14] "POST / HTTP/1.1" 200 -
python3 worker.py 8000 w1-2 23002 am
Filename: 'data-am.json'
Data loaded successfully for group am.
{'message': 'success'}
Worker w1-2 listening on port 23002...
127.0.0.1 - - [22/Feb/2024 18:03:14] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [22/Feb/2024 18:03:14] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [22/Feb/2024 18:03:14] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [22/Feb/2024 18:03:14] "POST / HTTP/1.1" 200 -
~/u/dc/cs-5546/assignment-1 | features = !2
python3 worker.py 8000 w2-1 23003 nz
Filename: 'data-nz.json'
Data loaded successfully for group nz.
{'message': 'success'}
Worker w2-1 listening on port 23003...
127.0.0.1 - - [22/Feb/2024 18:03:14] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [22/Feb/2024 18:03:14] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [22/Feb/2024 18:03:14] "POST / HTTP/1.1" 200 -
~/u/dc/cs-5546/assignment-1 | features = !2
python3 worker.py 8000 w2-2 23004 nz
Filename: 'data-nz.json'
Data loaded successfully for group nz.
{'message': 'success'}
Worker w2-2 listening on port 23004...
127.0.0.1 - - [22/Feb/2024 18:03:14] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [22/Feb/2024 18:03:14] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [22/Feb/2024 18:03:14] "POST / HTTP/1.1" 200 -


client.py - cs-5546 - Visual Studio Code
127.0.0.1 - - [22/Feb/2024 18:03:14] "POST / HTTP/1.1" 200 -
Get by name called: alice
worker chose: w1-2
127.0.0.1 - - [22/Feb/2024 18:05:38] "POST / HTTP/1.1" 200 -
Get by name called: alice
worker chose: w1-1
127.0.0.1 - - [22/Feb/2024 18:05:45] "POST / HTTP/1.1" 200 -
Get by name called: alice
worker chose: w1-2
127.0.0.1 - - [22/Feb/2024 18:05:47] "POST / HTTP/1.1" 200 -
Get by name called: alice
worker chose: w1-1
127.0.0.1 - - [22/Feb/2024 18:05:49] "POST / HTTP/1.1" 200 -
Get by name called: alice
worker chose: w1-2
127.0.0.1 - - [22/Feb/2024 18:05:51] "POST / HTTP/1.1" 200 -
Get by name called: alice
worker chose: w1-1
127.0.0.1 - - [22/Feb/2024 18:06:05] "POST / HTTP/1.1" 200 -
{'error': False, 'result': [{'record_id': 2, 'name': 'alice', 'location':
'New York City', 'year': 2002}]}
~/u/dc/cs-5546/assignment-1 | features = !3
python3 client.py 8000
Client => Asking for person with alice
{'error': False, 'result': [{'record_id': 2, 'name': 'alice', 'location':
'New York City', 'year': 2002}]}
~/u/dc/cs-5546/assignment-1 | features = !3
python3 client.py 8000
Client => Asking for person with alice
{'error': False, 'result': [{'record_id': 2, 'name': 'alice', 'location':
'New York City', 'year': 2002}]}
~/u/dc/cs-5546/assignment-1 | features = !3
i3: H[V[Alacritty Alacritty] V[V[Alacritty Alacritty] V[Alacritty Alacritty]]]
127.0.0.1 - - [22/Feb/2024 18:05:38] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [22/Feb/2024 18:05:45] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [22/Feb/2024 18:05:45] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [22/Feb/2024 18:05:47] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [22/Feb/2024 18:05:49] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [22/Feb/2024 18:05:49] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [22/Feb/2024 18:05:51] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [22/Feb/2024 18:06:05] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [22/Feb/2024 18:06:05] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [22/Feb/2024 18:05:38] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [22/Feb/2024 18:05:38] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [22/Feb/2024 18:05:45] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [22/Feb/2024 18:05:47] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [22/Feb/2024 18:05:47] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [22/Feb/2024 18:05:49] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [22/Feb/2024 18:05:51] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [22/Feb/2024 18:05:51] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [22/Feb/2024 18:06:05] "POST / HTTP/1.1" 200 -
~/u/dc/cs-5546/assignment-1 | features = !2
python3 worker.py 8000 w2-1 23003 nz
Filename: 'data-nz.json'
Data loaded successfully for group nz.
{'message': 'success'}
Worker w2-1 listening on port 23003...
127.0.0.1 - - [22/Feb/2024 18:03:14] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [22/Feb/2024 18:03:14] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [22/Feb/2024 18:03:14] "POST / HTTP/1.1" 200 -
~/u/dc/cs-5546/assignment-1 | features = !2
python3 worker.py 8000 w2-2 23004 nz
Filename: 'data-nz.json'
Data loaded successfully for group nz.
{'message': 'success'}
Worker w2-2 listening on port 23004...
127.0.0.1 - - [22/Feb/2024 18:03:14] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [22/Feb/2024 18:03:14] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [22/Feb/2024 18:03:14] "POST / HTTP/1.1" 200 -

Any worker or server errors are also handled here. As shown in the below screenshot, if one or all the worker 1 instances are down, the master service is not impacted and the error is handled on the client side.



Observations (Based on our implementation) :

1. The master server will not be impacted if one or more workers are down.
2. The master server should be running before any worker instances start. This is because the workers try to register with the master server as soon as they are started.
3. When the query is handled by the master server, it makes a call to each of the worker instances and gets the load then makes another call to the worker with less load.
4. Improvement - The calls made to the workers are made synchronously, a better way to handle this is to make asynchronous calls.