NSDSYST - Final Project

Technical Documentation

GitHub Repository

https://github.com/jm55DLSU/NSDSYST/

System Specifications & Dependencies

System Specifications (Requirements)

1. Inputs:
    a. Input & Output Paths
    b. Brightness, Contrast & Sharpness Factors
2. Process: Image Processing
3. Outputs:
    a. Processed Images
    b. Report File: Input & Output Paths, # of Images Processed, # of Machines Used

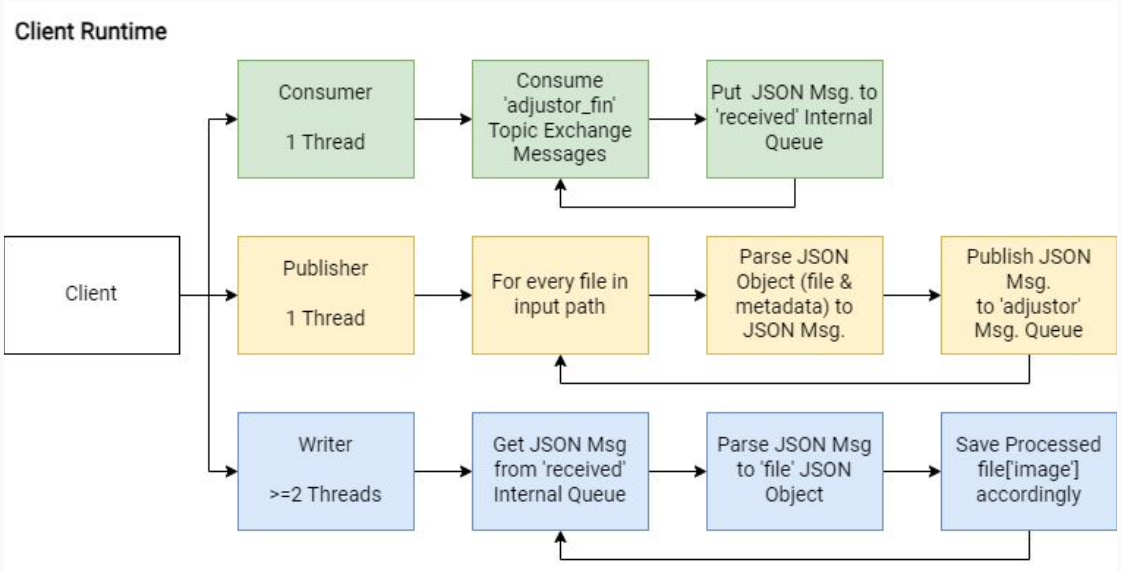
System Specifications (Implementation)

1. Uses OpenCV as image processing library.
2. Input images limited to 5MB .

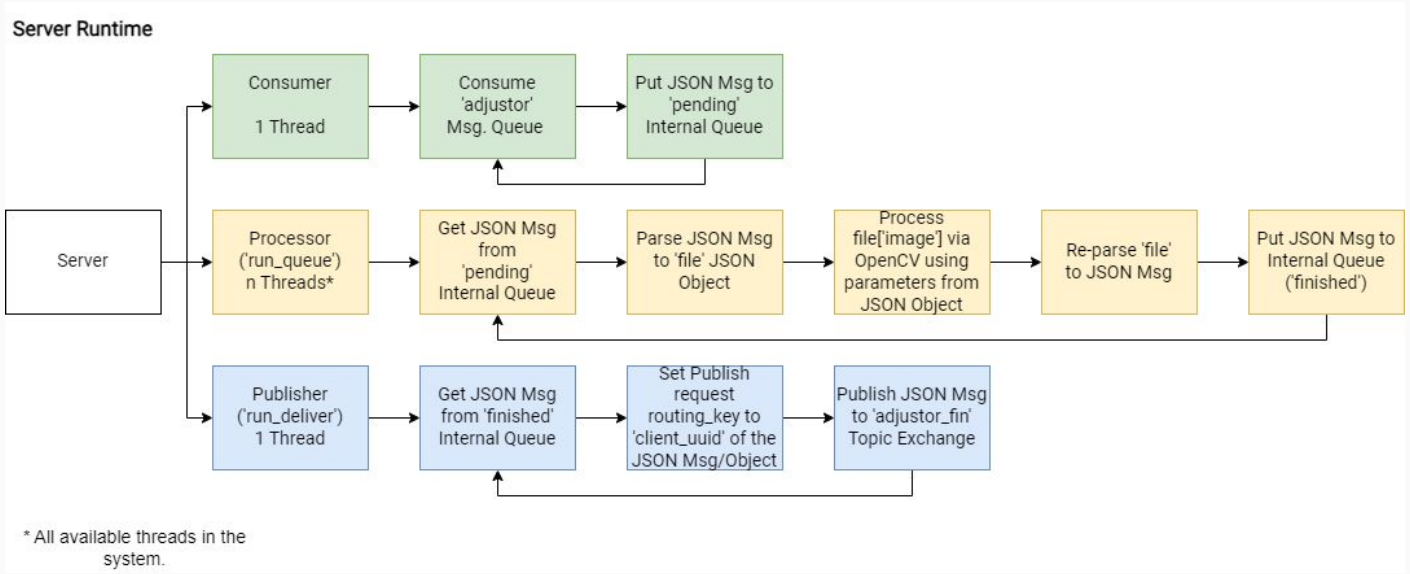
System Dependencies

1. Python 3 (at least 3.10.x)
2. Python Packages
    a. pika
    b. pillow (PIL)
    c. Opencv-python

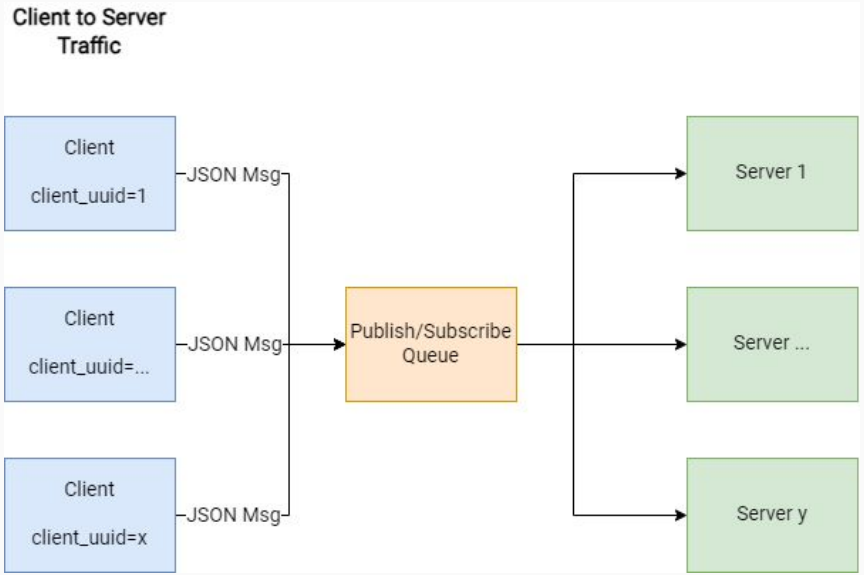
System Runtime Design (Client - MIMD)



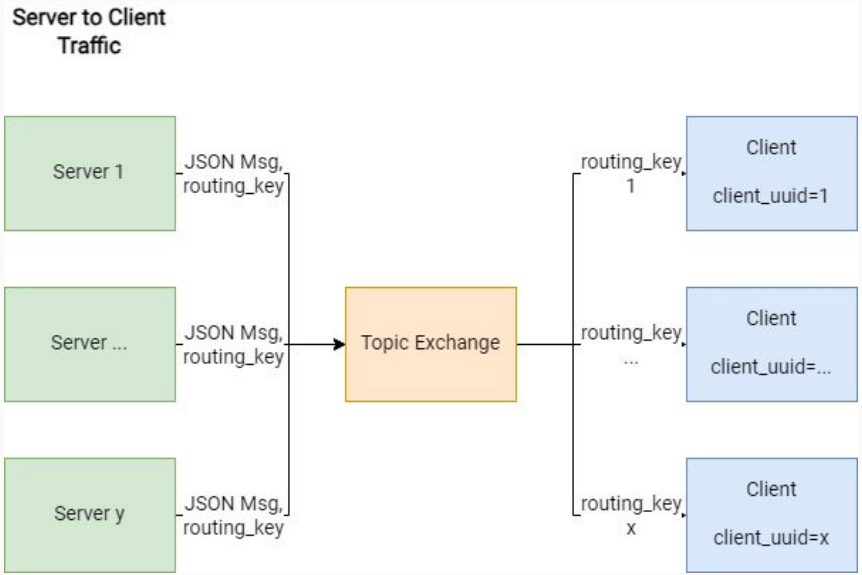
System Runtime Design (Server - MIMD)



System Traffic Design (Client to Server)



System Traffic Design (Server to Client)



Delivery Mechanisms (Client to Server)

- It uses a Message Queue mechanism to pass messages from Client to Server.
- The identity of the message origin is not monitored here, as the server should accept/consumer messages regardless of who delivered them.
- The distribution of workload or message on each available server/consumer is handled by the message broker (i.e., RabbitMQ).

Delivery Mechanisms (Server to Client)

- Uses a Topic Exchange mechanism to selectively pass messages from the Server to the appropriate client via the Topic/routing_key value (i.e., Client UUID).
- The Server delivers the message to the Topic Exchange with a routing_key attached (i.e., Client UUID). The message broker then forwards the message to the appropriate Client that matches the routing_key specified.
- During its setup, the Client binds itself to the message broker's Topic Exchange, where the routing_key is set as its UUID. The Client thereafter listens and consumes any relevant messages the message broker delivers.

JSON Object Structure

Key	Raw Data Type	Description
input	String	Input folder path
filename	String	The filename of the image
output	String	Output folder path
brightness	Integer	Brightness factor (0-100)
contrast	Integer	Contrast factor (0-100)
sharpness	Integer	Sharpness (0-100)
image	OpenCV Mat Class *Stringified via pickle.dumps() → Base64 Encode → ASCII Decode	Actual image data
client_uuid	String	Client UUID

Benchmark Datasets

1. “Tokyo”
    a. File Size Range: 367 KB to 12.9 MB
    b. Total File Size: 134.2 MB
    c. Total Count: 55 Images
    d. Acceptable File Count (File Sizes <= 5 MB): 49 Images (89.09%)
    e. Note: Represents processing heavy image files
2. “Crops”
    a. File Size Range: 2.9 KB to 1.8 MB
    b. Total File Size: 76.8 MB
    c. Total Count: 773 Images
    d. Acceptable File Count (File Sizes <= 5 MB): 773 Images (100%)
    e. Note: Represent processing small image files

Benchmark System (Laptop)

1. Server (Guest VM):
    a. CPU (Threads): 4
    b. RAM: 2048 MB
2. Client (Host):
    a. CPU (Threads): 8
    b. RAM: 8192 MB
3. RabbitMQ: Same as Client’s

Benchmark Results (Laptop)

	Benchmark ID (No. of Machines:Run)	Time (s) Tokyo Dataset	Time (s) Crops Dataset
4T	1:1	67.8779	38.1426
4T	1:2	45.0550	34.7252
4T	1:3	45.8793	33.6548
4T	1:4	48.2554	35.0385
8T	2:1	39.5424	29.7509
8T	2:2	39.7682	29.6472
8T	2:3	46.0564	30.4623
8T	2:4	40.4689	29.4173

Benchmark Results (Laptop)

No. of Machines	Average Time (s) Tokyo Dataset	Average Time (s) Crops Dataset
1 (4 Threads Total)	51.7669	35.3093
2 (8 Threads Total)	41.4590	29.8194

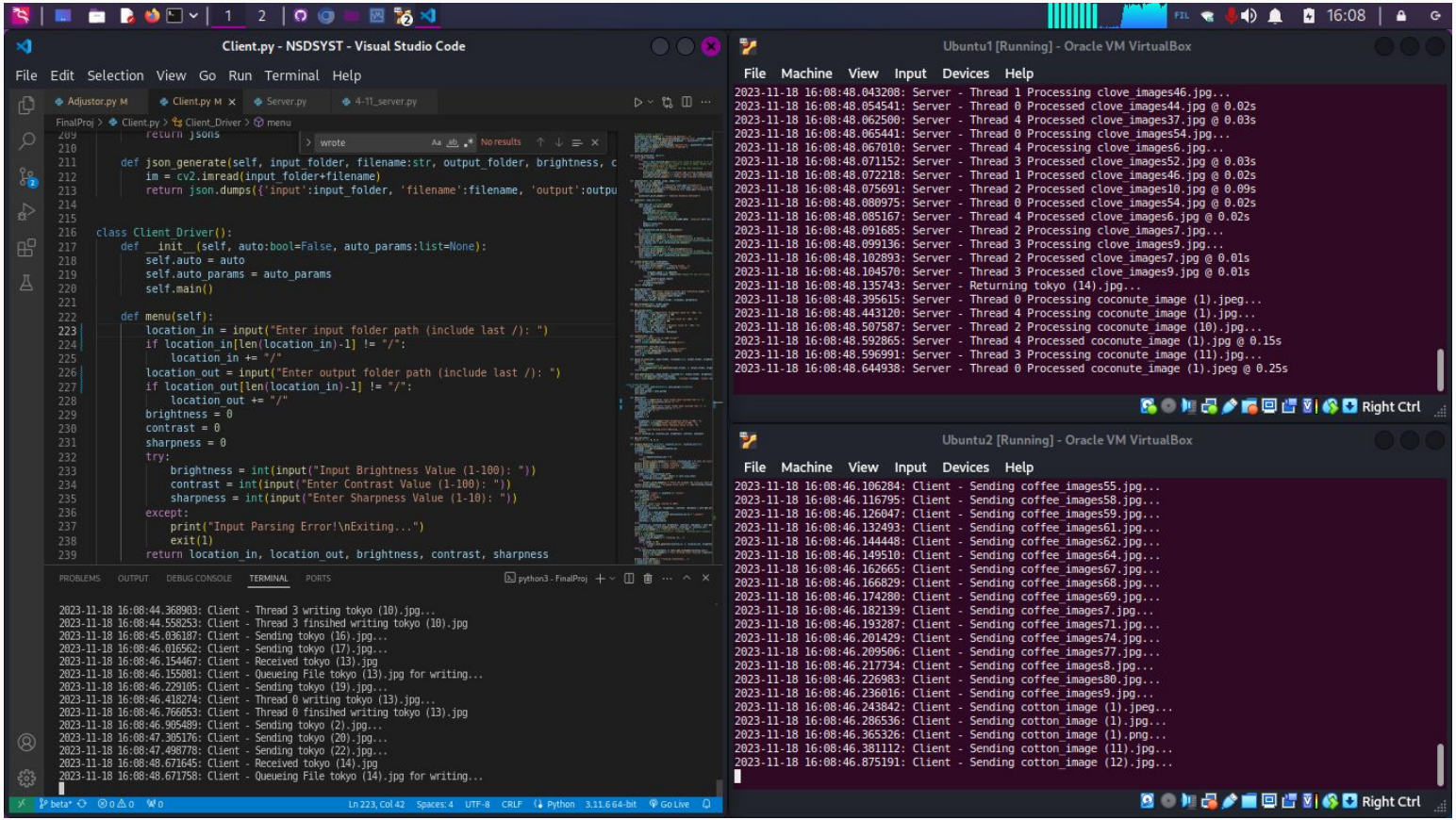
No. of Machines	Time Reduction Tokyo Dataset	Time Reduction Crops Dataset
1 to 2 Machines	19.91%	15.55%

Known Issues (as of Nov. 19, 2023)

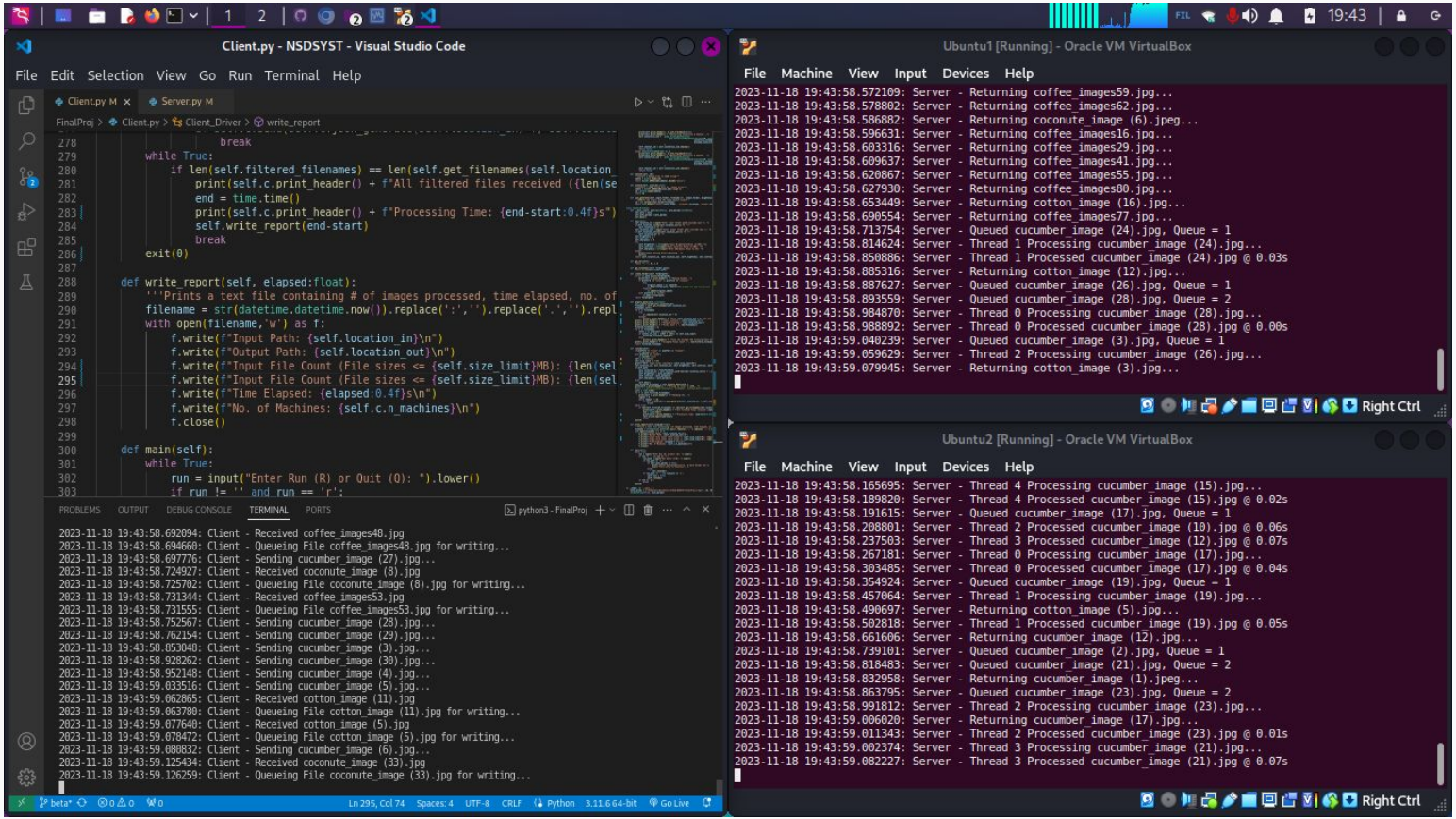
- The Client does not gracefully terminate (However, it completes its task properly).
- Bottleneck was determined to be the communications between the Server and the Message Broker (i.e., reliant on its performance), which is especially noticeable for large image files. Hence, the times may not always be as what’s seen in the benchmarks.

Screenshots

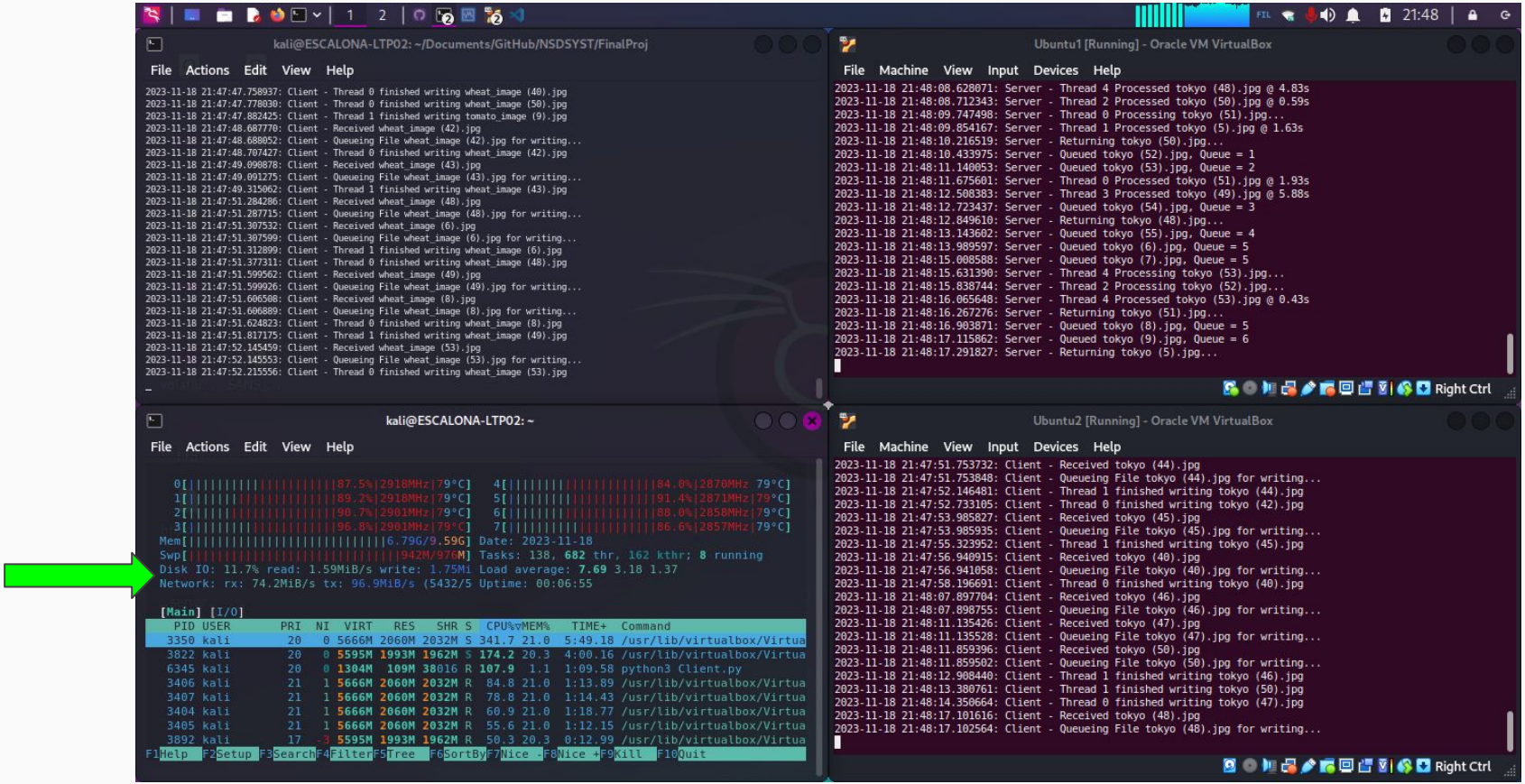
(During Development)



Multi-Client Single-Server Transaction



Single-Client Multi-Server Transaction



Multi-Client Single-Server Transaction Network Throughput

References

[1] "AMQP 0-9-1 Model Explained — RabbitMQ," RabbitMQ. https://www.rabbitmq.com/tutorials/amqp-concepts.html (accessed Nov. 19, 2023).
[2] "Part 4: RabbitMQ Exchanges, routing keys and bindings," CloudAMQP, 2019. https://www.cloudamqp.com/blog/part4-rabbitmq-for-beginners-exchanges-routing-keys-bindings.html (accessed Nov. 19, 2023).
[3] "RabbitMQ Topic Exchange Explained," CloudAMQP, 2022. https://www.cloudamqp.com/blog/rabbitmq-topic-exchange-explained.html (accessed Nov. 19, 2023).
[4] "RabbitMQ tutorial," Topics — RabbitMQ. https://www.rabbitmq.com/tutorials/tutorial-five-python.html (accessed Nov. 19, 2023).
[5] Pankaj, "Python Multiprocessing Example," DigitalOcean, Aug. 03, 2022. Accessed: Nov. 19, 2023. [Online]. Available: https://www.digitalocean.com/community/tutorials/python-multiprocessing-example
[6] "RabbitMQ tutorial," Work Queues — RabbitMQ. https://www.rabbitmq.com/tutorials/tutorial-two-python.html (accessed Nov. 19, 2023).
[7] K41F4r, "Sending OpenCV image in JSON," Stack Overflow. https://stackoverflow.com/questions/55892362/sending-opencv-image-in-json/55900422#55900422 (accessed Nov. 19, 2023).
[8] pika, "pika/examples/basic_consumer_threaded.py at 1.0.1 · pika/pika," GitHub. https://github.com/pika/pika/blob/1.0.1/examples/basic_consumer_threaded.py (accessed Nov. 19, 2023).
[9] S. A. Khan, "How to change the contrast and brightness of an image using OpenCV in Python?," Tutorialspoint, 2023. https://www.tutorialspoint.com/how-to-change-the-contrast-and-brightness-of-an-image-using-opencv-in-python (accessed Nov. 19, 2023).
[10] "How to Sharpen an Image with OpenCV," How to use OpenCV, 2023. https://www.opencvhelp.org/tutorials/image-processing/how-to-sharpen-image/ (accessed Nov. 19, 2023).
[11] "Python OpenCV cv2.imwrite method," GeeksforGeeks, Aug. 05, 2019. Accessed: Nov. 19, 2023. [Online]. Available: https://www.geeksforgeeks.org/python-opencv-cv2-imwrite-method/
[12] M. W. Azam, "Agricultural crops image classification," Kaggle. https://www.kaggle.com/datasets/mdwaquarazam/agricultural-crops-image-classification (accessed Nov. 19, 2023)