

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

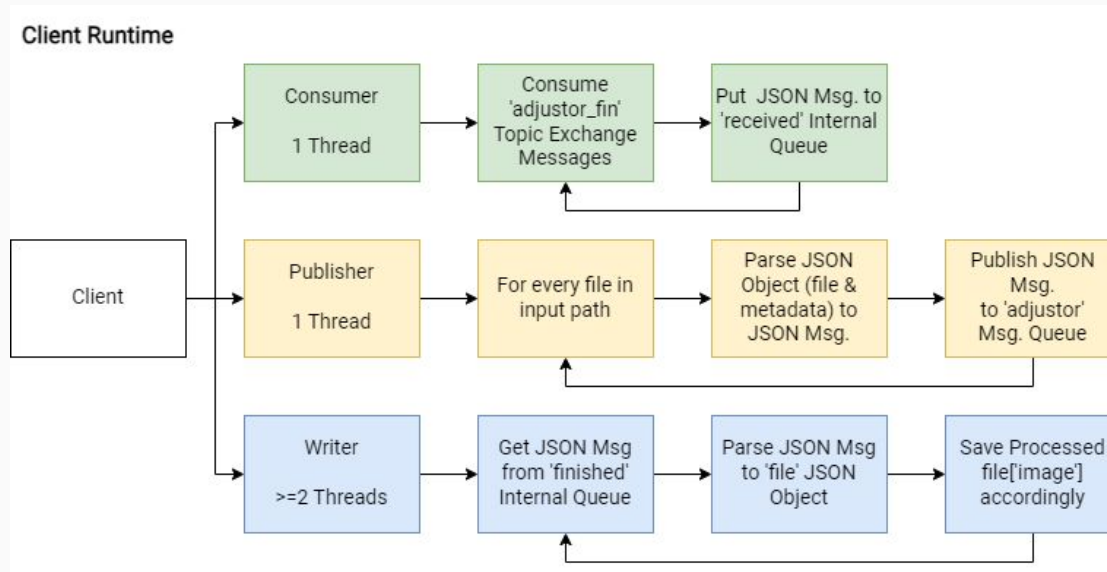
System Specifications (Implementation)

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

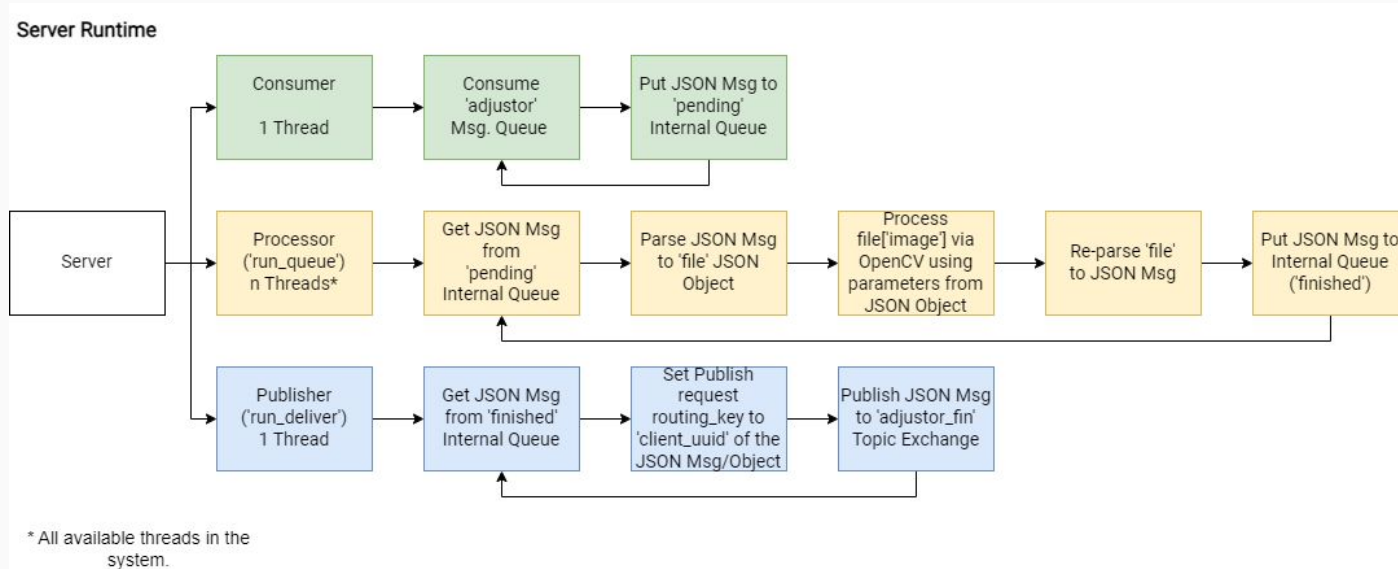
System Dependencies

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

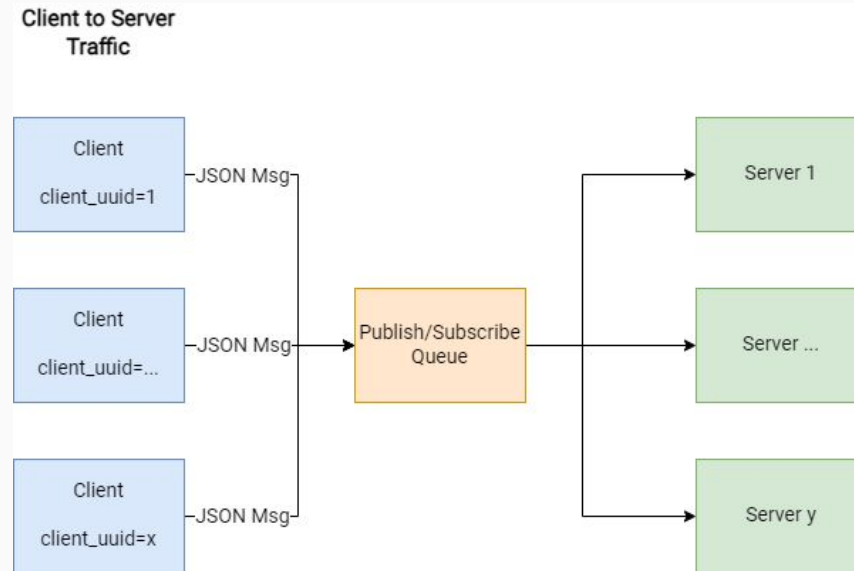
System Runtime Design (Client - MIMD)



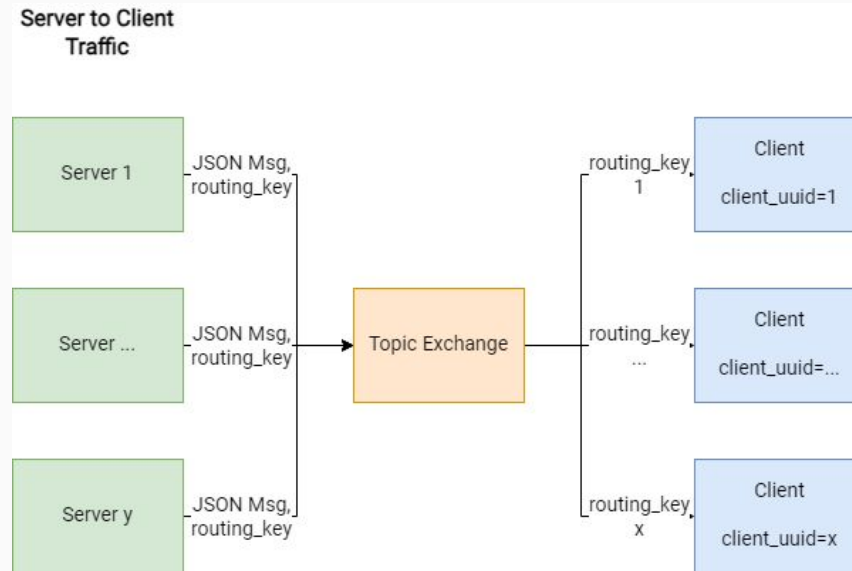
System Runtime Design (Server - MIMD)



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/consume 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 \leq 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 \leq 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
1:1	67.8779	38.1426
1:2	45.0550	34.7252
1:3	45.8793	33.6548
1:4	48.2554	35.0385
2:1	39.5424	29.7509
2:2	39.7682	29.6472
2:3	46.0564	30.4623
2:4	40.4689	29.4173

Benchmark Results (Laptop)

No. of Machines	Average Time (s) Tokyo Dataset	Average Time (s) Crops Dataset
1	51.7669	35.3093
2	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)

The image displays a multi-client single-server transaction system running in a virtual machine environment. The interface is split into three main sections: a code editor, a terminal, and a log viewer.

Code Editor (Client.py): The code defines a `Client_Driver` class with methods for generating JSON data, processing images, and handling client requests. The `menu` method is the primary interface for the user, allowing them to input folder paths, brightness, contrast, and sharpness values. The `process_image` method is responsible for processing the input images and returning the results.

Terminal: The terminal shows the execution of the `Client.py` script. It displays the output of the `menu` method, including the input folder path, output folder path, and the processed image results. The terminal also shows the output of the `process_image` method, which returns the processed image data.

Log Viewer: The log viewer displays the system logs, showing the sequence of events and the processing of client requests. The logs include the following information:

- Timestamp: 2023-11-18 16:08:48.043208
- Server: Thread 1
- Processing clove_images46.jpg...
- 2023-11-18 16:08:48.054541: Server - Thread 0 Processed clove_images44.jpg @ 0.02s
- 2023-11-18 16:08:48.062500: Server - Thread 4 Processed clove_images37.jpg @ 0.03s
- 2023-11-18 16:08:48.065441: Server - Thread 0 Processed clove_images54.jpg...
- 2023-11-18 16:08:48.067010: Server - Thread 4 Processing clove_images6.jpg...
- 2023-11-18 16:08:48.071152: Server - Thread 3 Processed clove_images52.jpg @ 0.03s
- 2023-11-18 16:08:48.072218: Server - Thread 1 Processed clove_images46.jpg @ 0.02s
- 2023-11-18 16:08:48.075691: Server - Thread 2 Processed clove_images10.jpg @ 0.09s
- 2023-11-18 16:08:48.080975: Server - Thread 0 Processed clove_images54.jpg @ 0.02s
- 2023-11-18 16:08:48.085167: Server - Thread 4 Processed clove_images6.jpg @ 0.02s
- 2023-11-18 16:08:48.091685: Server - Thread 2 Processing clove_images7.jpg...
- 2023-11-18 16:08:48.099136: Server - Thread 3 Processing clove_images9.jpg...
- 2023-11-18 16:08:48.102893: Server - Thread 2 Processed clove_images7.jpg @ 0.01s
- 2023-11-18 16:08:48.104570: Server - Thread 3 Processed clove_images9.jpg @ 0.01s
- 2023-11-18 16:08:48.135743: Server - Returning tokyo (14).jpg...
- 2023-11-18 16:08:48.395615: Server - Thread 0 Processing coconute_image (1).jpeg...
- 2023-11-18 16:08:48.443120: Server - Thread 4 Processing coconute_image (1).jpg...
- 2023-11-18 16:08:48.507587: Server - Thread 2 Processing coconute_image (10).jpg...
- 2023-11-18 16:08:48.592865: Server - Thread 4 Processed coconute_image (1).jpg @ 0.15s
- 2023-11-18 16:08:48.596991: Server - Thread 3 Processing coconute_image (11).jpg...
- 2023-11-18 16:08:48.644938: Server - Thread 0 Processed coconute_image (1).jpeg @ 0.25s

Multi-Client Single-Server Transaction

Client.py - NSDSYST - Visual Studio Code

```
File Edit Selection View Go Run Terminal Help

Client.py x Server.py M
FinalProj > Client.py > Client_Driver > write_report

278         break
279     while True:
280         if len(self.filtered_filenames) == len(self.get_filenames(self.location)):
281             print(self.c.print_header() + f"All filtered files received ({len(self.filtered_filenames)} files)")
282             end = time.time()
283             print(self.c.print_header() + f"Processing Time: {end-start:0.4f}s")
284             self.write_report(end-start)
285             break
286     exit(0)
287
288     def write_report(self, elapsed:float):
289         """Prints a text file containing # of images processed, time elapsed, no. of
290         filename = str(datetime.datetime.now()).replace(':', '').replace('.', '').replace(' ', '')
291         with open(filename+'.w') as f:
292             f.write(f"Input Path: {self.location_in}\n")
293             f.write(f"Output Path: {self.location_out}\n")
294             f.write(f"Input File Count (File sizes <= {self.size_limit}MB): {len(self.filtered_filenames)}\n")
295             f.write(f"Input File Count (File sizes <= {self.size_limit}MB): {len(self.filtered_filenames)}\n")
296             f.write(f"Time Elapsed: {elapsed:0.4f}s\n")
297             f.write(f"No. of Machines: {self.c.n_machines}\n")
298             f.close()
299
300     def main(self):
301         while True:
302             run = input("Enter Run (R) or Quit (Q): ").lower()
303             if run != 'r' and run != 'q':
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

python3 - FinalProj +

```
2023-11-18 19:43:58.692094: Client - Received coffee_images48.jpg
2023-11-18 19:43:58.694669: Client - Queuing File coffee_images48.jpg for writing...
2023-11-18 19:43:58.697776: Client - Sending cucumber_image (27).jpg...
2023-11-18 19:43:58.724927: Client - Received coconute_image (8).jpg
2023-11-18 19:43:58.725702: Client - Queuing File coconute_image (8).jpg for writing...
2023-11-18 19:43:58.731344: Client - Received coffee_images53.jpg
2023-11-18 19:43:58.731555: Client - Queuing File coffee_images53.jpg for writing...
2023-11-18 19:43:58.732567: Client - Sending cucumber_image (28).jpg...
2023-11-18 19:43:58.762154: Client - Sending cucumber_image (29).jpg...
2023-11-18 19:43:58.853048: Client - Sending cucumber_image (3).jpg...
2023-11-18 19:43:58.926262: Client - Sending cucumber_image (30).jpg...
2023-11-18 19:43:58.952148: Client - Sending cucumber_image (4).jpg...
2023-11-18 19:43:59.033516: Client - Sending cucumber_image (5).jpg...
2023-11-18 19:43:59.062665: Client - Received cotton_image (11).jpg
2023-11-18 19:43:59.063789: Client - Queuing File cotton_image (11).jpg for writing...
2023-11-18 19:43:59.077640: Client - Received cotton_image (5).jpg
2023-11-18 19:43:59.078472: Client - Queuing File cotton_image (5).jpg for writing...
2023-11-18 19:43:59.088832: Client - Sending cucumber_image (6).jpg...
2023-11-18 19:43:59.125434: Client - Received coconute_image (33).jpg
2023-11-18 19:43:59.126259: Client - Queuing File coconute_image (33).jpg for writing...
```

Ubuntu1 [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

```
2023-11-18 19:43:58.572109: Server - Returning coffee_images59.jpg...
2023-11-18 19:43:58.578802: Server - Returning coffee_images62.jpg...
2023-11-18 19:43:58.586882: Server - Returning coconute_image (6).jpeg...
2023-11-18 19:43:58.596631: Server - Returning coffee_images16.jpg...
2023-11-18 19:43:58.603316: Server - Returning coffee_images29.jpg...
2023-11-18 19:43:58.609637: Server - Returning coffee_images41.jpg...
2023-11-18 19:43:58.620867: Server - Returning coffee_images55.jpg...
2023-11-18 19:43:58.627930: Server - Returning coffee_images80.jpg...
2023-11-18 19:43:58.653449: Server - Returning cotton_image (16).jpg...
2023-11-18 19:43:58.690554: Server - Returning coffee_images77.jpg...
2023-11-18 19:43:58.713754: Server - Queued cucumber_image (24).jpg, Queue = 1
2023-11-18 19:43:58.814624: Server - Thread 1 Processing cucumber_image (24).jpg...
2023-11-18 19:43:58.850886: Server - Thread 1 Processed cucumber_image (24).jpg @ 0.03s
2023-11-18 19:43:58.885316: Server - Returning cotton_image (12).jpg...
2023-11-18 19:43:58.887627: Server - Queued cucumber_image (26).jpg, Queue = 1
2023-11-18 19:43:58.893559: Server - Queued cucumber_image (28).jpg, Queue = 2
2023-11-18 19:43:58.984870: Server - Thread 0 Processing cucumber_image (28).jpg...
2023-11-18 19:43:58.988892: Server - Thread 0 Processed cucumber_image (28).jpg @ 0.00s
2023-11-18 19:43:59.040239: Server - Queued cucumber_image (3).jpg, Queue = 1
2023-11-18 19:43:59.059629: Server - Thread 2 Processing cucumber_image (26).jpg...
2023-11-18 19:43:59.079945: Server - Returning cotton_image (3).jpg...
```

Ubuntu2 [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

```
2023-11-18 19:43:58.165695: Server - Thread 4 Processing cucumber_image (15).jpg...
2023-11-18 19:43:58.189820: Server - Thread 4 Processed cucumber_image (15).jpg @ 0.02s
2023-11-18 19:43:58.191615: Server - Queued cucumber_image (17).jpg, Queue = 1
2023-11-18 19:43:58.208801: Server - Thread 2 Processed cucumber_image (10).jpg @ 0.06s
2023-11-18 19:43:58.237503: Server - Thread 3 Processed cucumber_image (12).jpg @ 0.07s
2023-11-18 19:43:58.267181: Server - Thread 0 Processing cucumber_image (17).jpg...
2023-11-18 19:43:58.303485: Server - Thread 0 Processed cucumber_image (17).jpg @ 0.04s
2023-11-18 19:43:58.354924: Server - Queued cucumber_image (19).jpg, Queue = 1
2023-11-18 19:43:58.457064: Server - Thread 1 Processing cucumber_image (19).jpg...
2023-11-18 19:43:58.496697: Server - Returning cotton_image (5).jpg...
2023-11-18 19:43:58.502818: Server - Thread 1 Processed cucumber_image (19).jpg @ 0.05s
2023-11-18 19:43:58.661666: Server - Returning cucumber_image (12).jpg...
2023-11-18 19:43:58.739101: Server - Queued cucumber_image (2).jpg, Queue = 1
2023-11-18 19:43:58.818483: Server - Queued cucumber_image (21).jpg, Queue = 2
2023-11-18 19:43:58.832958: Server - Returning cucumber_image (1).jpeg...
2023-11-18 19:43:58.863795: Server - Queued cucumber_image (23).jpg, Queue = 2
2023-11-18 19:43:58.991812: Server - Thread 2 Processing cucumber_image (23).jpg...
2023-11-18 19:43:59.006020: Server - Returning cucumber_image (17).jpg...
2023-11-18 19:43:59.011343: Server - Thread 2 Processed cucumber_image (23).jpg @ 0.01s
2023-11-18 19:43:59.002374: Server - Thread 3 Processing cucumber_image (21).jpg...
2023-11-18 19:43:59.082227: Server - Thread 3 Processed cucumber_image (21).jpg @ 0.07s
```

Single-Client Multi-Server Transaction

The image displays a Kali Linux terminal window and two Oracle VM VirtualBox windows, all showing the results of a network throughput test. The terminal window, titled 'kali@ESCALONA-LTP02: ~/Documents/GitHub/NSDSYST/FinalProj', shows a list of transactions with timestamps, client IDs, and thread numbers. The two VirtualBox windows, titled 'Ubuntu1 [Running] - Oracle VM VirtualBox' and 'Ubuntu2 [Running] - Oracle VM VirtualBox', show the results of the test on the respective virtual machines. The terminal window also displays a summary of the test results, including the number of transactions, the total time taken, and the average throughput.

Kali Linux Terminal Output:

```
2023-11-18 21:47:47.758937: Client - Thread 0 finished writing wheat_image (48).jpg
2023-11-18 21:47:47.778039: Client - Thread 0 finished writing wheat_image (50).jpg
2023-11-18 21:47:47.882425: Client - Thread 1 finished writing tomato_image (9).jpg
2023-11-18 21:47:48.087770: Client - Received wheat_image (42).jpg
2023-11-18 21:47:48.088052: Client - Queueing File wheat_image (42).jpg for writing...
2023-11-18 21:47:48.207427: Client - Thread 0 finished writing wheat_image (42).jpg
2023-11-18 21:47:49.090878: Client - Received wheat_image (43).jpg
2023-11-18 21:47:49.091275: Client - Queueing File wheat_image (43).jpg for writing...
2023-11-18 21:47:49.315062: Client - Thread 1 finished writing wheat_image (43).jpg
2023-11-18 21:47:51.284286: Client - Received wheat_image (48).jpg
2023-11-18 21:47:51.287715: Client - Queueing File wheat_image (48).jpg for writing...
2023-11-18 21:47:51.307532: Client - Received wheat_image (6).jpg
2023-11-18 21:47:51.307599: Client - Queueing File wheat_image (6).jpg for writing...
2023-11-18 21:47:51.312809: Client - Thread 1 finished writing wheat_image (6).jpg
2023-11-18 21:47:51.377311: Client - Thread 0 finished writing wheat_image (48).jpg
2023-11-18 21:47:51.599562: Client - Received wheat_image (49).jpg
2023-11-18 21:47:51.599926: Client - Queueing File wheat_image (49).jpg for writing...
2023-11-18 21:47:51.600589: Client - Received wheat_image (8).jpg
2023-11-18 21:47:51.600889: Client - Queueing File wheat_image (8).jpg for writing...
2023-11-18 21:47:51.624823: Client - Thread 0 finished writing wheat_image (8).jpg
2023-11-18 21:47:51.817175: Client - Thread 1 finished writing wheat_image (49).jpg
2023-11-18 21:47:52.145459: Client - Received wheat_image (53).jpg
2023-11-18 21:47:52.145553: Client - Queueing File wheat_image (53).jpg for writing...
2023-11-18 21:47:52.215556: Client - Thread 0 finished writing wheat_image (53).jpg
```

Ubuntu1 [Running] - Oracle VM VirtualBox Output:

```
2023-11-18 21:48:08.628071: Server - Thread 4 Processed tokyo (48).jpg @ 4.83s
2023-11-18 21:48:08.712343: Server - Thread 2 Processed tokyo (50).jpg @ 0.59s
2023-11-18 21:48:09.747498: Server - Thread 0 Processing tokyo (51).jpg...
2023-11-18 21:48:09.854167: Server - Thread 1 Processed tokyo (5).jpg @ 1.63s
2023-11-18 21:48:10.216519: Server - Returning tokyo (50).jpg...
2023-11-18 21:48:10.433975: Server - Queued tokyo (52).jpg, Queue = 1
2023-11-18 21:48:11.140853: Server - Queued tokyo (53).jpg, Queue = 2
2023-11-18 21:48:11.675601: Server - Thread 0 Processed tokyo (51).jpg @ 1.93s
2023-11-18 21:48:12.508383: Server - Thread 3 Processed tokyo (49).jpg @ 5.88s
2023-11-18 21:48:12.723437: Server - Queued tokyo (54).jpg, Queue = 3
2023-11-18 21:48:12.849610: Server - Returning tokyo (48).jpg...
2023-11-18 21:48:13.143602: Server - Queued tokyo (55).jpg, Queue = 4
2023-11-18 21:48:13.989597: Server - Queued tokyo (6).jpg, Queue = 5
2023-11-18 21:48:15.008588: Server - Queued tokyo (7).jpg, Queue = 5
2023-11-18 21:48:15.631390: Server - Thread 4 Processing tokyo (53).jpg...
2023-11-18 21:48:15.838744: Server - Thread 2 Processing tokyo (52).jpg...
2023-11-18 21:48:16.065648: Server - Thread 4 Processed tokyo (53).jpg @ 0.43s
2023-11-18 21:48:16.267276: Server - Returning tokyo (51).jpg...
2023-11-18 21:48:16.903871: Server - Queued tokyo (8).jpg, Queue = 5
2023-11-18 21:48:17.115862: Server - Queued tokyo (9).jpg, Queue = 6
2023-11-18 21:48:17.291827: Server - Returning tokyo (5).jpg...
```

Ubuntu2 [Running] - Oracle VM VirtualBox Output:

```
2023-11-18 21:47:51.753732: Client - Received tokyo (44).jpg
2023-11-18 21:47:51.753848: Client - Queueing File tokyo (44).jpg for writing...
2023-11-18 21:47:52.146481: Client - Thread 1 finished writing tokyo (44).jpg
2023-11-18 21:47:52.733105: Client - Thread 0 finished writing tokyo (42).jpg
2023-11-18 21:47:53.985827: Client - Received tokyo (45).jpg
2023-11-18 21:47:53.985935: Client - Queueing File tokyo (45).jpg for writing...
2023-11-18 21:47:55.323952: Client - Thread 1 finished writing tokyo (45).jpg
2023-11-18 21:47:56.940915: Client - Received tokyo (40).jpg
2023-11-18 21:47:56.941058: Client - Queueing File tokyo (40).jpg for writing...
2023-11-18 21:47:58.106691: Client - Thread 0 finished writing tokyo (40).jpg
2023-11-18 21:48:07.897784: Client - Received tokyo (46).jpg
2023-11-18 21:48:07.898755: Client - Queueing File tokyo (46).jpg for writing...
2023-11-18 21:48:11.135426: Client - Received tokyo (47).jpg
2023-11-18 21:48:11.135528: Client - Queueing File tokyo (47).jpg for writing...
2023-11-18 21:48:11.859396: Client - Received tokyo (50).jpg
2023-11-18 21:48:11.859502: Client - Queueing File tokyo (50).jpg for writing...
2023-11-18 21:48:12.908440: Client - Thread 1 finished writing tokyo (46).jpg
2023-11-18 21:48:13.380761: Client - Thread 1 finished writing tokyo (50).jpg
2023-11-18 21:48:14.350664: Client - Thread 0 finished writing tokyo (47).jpg
2023-11-18 21:48:17.101616: Client - Received tokyo (48).jpg
2023-11-18 21:48:17.102564: Client - Queueing File tokyo (48).jpg for writing...
```

Kali Linux Terminal Summary:

```
0[|||||] [87.5%|2918MHz|79°C] 4[|||||] [84.0%|2870MHz|79°C]
1[|||||] [89.2%|2918MHz|79°C] 5[|||||] [91.4%|2871MHz|79°C]
2[|||||] [90.7%|2901MHz|79°C] 6[|||||] [88.0%|2858MHz|79°C]
3[|||||] [96.8%|2901MHz|79°C] 7[|||||] [86.6%|2857MHz|79°C]
Mem[|||||] [6.79G/9.59G] Date: 2023-11-18
Swp[|||||] [1942M/976M] Tasks: 138, 682 thr, 162 kthr; 8 running
Disk IO: 11.7% read: 1.59MiB/s write: 1.75MiB Load average: 7.69 3.18 1.37
Network: rx: 74.2MiB/s tx: 96.9MiB/s (5432/S Uptime: 00:06:55
```

Kali Linux Task List:

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
3350	kali	20	0	5666M	2060M	2032M	S	341.7	21.0	5:49.18	/usr/lib/virtualbox/Virtua
3822	kali	20	0	5595M	1993M	1962M	S	174.2	20.3	4:00.16	/usr/lib/virtualbox/Virtua
6345	kali	20	0	1304M	109M	38016	R	107.9	1.1	1:09.58	python3 Client.py
3406	kali	21	1	5666M	2060M	2032M	R	78.8	21.0	1:13.89	/usr/lib/virtualbox/Virtua
3407	kali	21	1	5666M	2060M	2032M	R	84.0	21.0	1:14.43	/usr/lib/virtualbox/Virtua
3404	kali	21	1	5666M	2060M	2032M	R	60.9	21.0	1:18.77	/usr/lib/virtualbox/Virtua
3405	kali	21	1	5666M	2060M	2032M	S	55.6	21.0	1:12.15	/usr/lib/virtualbox/Virtua
3892	kali	17	3	5595M	1993M	1962M	S	50.3	20.3	0:12.99	/usr/lib/virtualbox/Virtua

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)