Course: OS (CSC301)

Distributed Matrix Multiplication System

Members:
Moiz Farooq
Samad Farooq
Rabia Shaikh

Instructor:
Dr. Anjum Nazir

SALIM HABIB UNIVERSITY
(FORMERLY BARRETT HODGSON UNIVERSITY)

March 2, 2022

Project Description

- This project shows how to use the distributed system to perform some computations, in our case, we will use matrix multiplication.

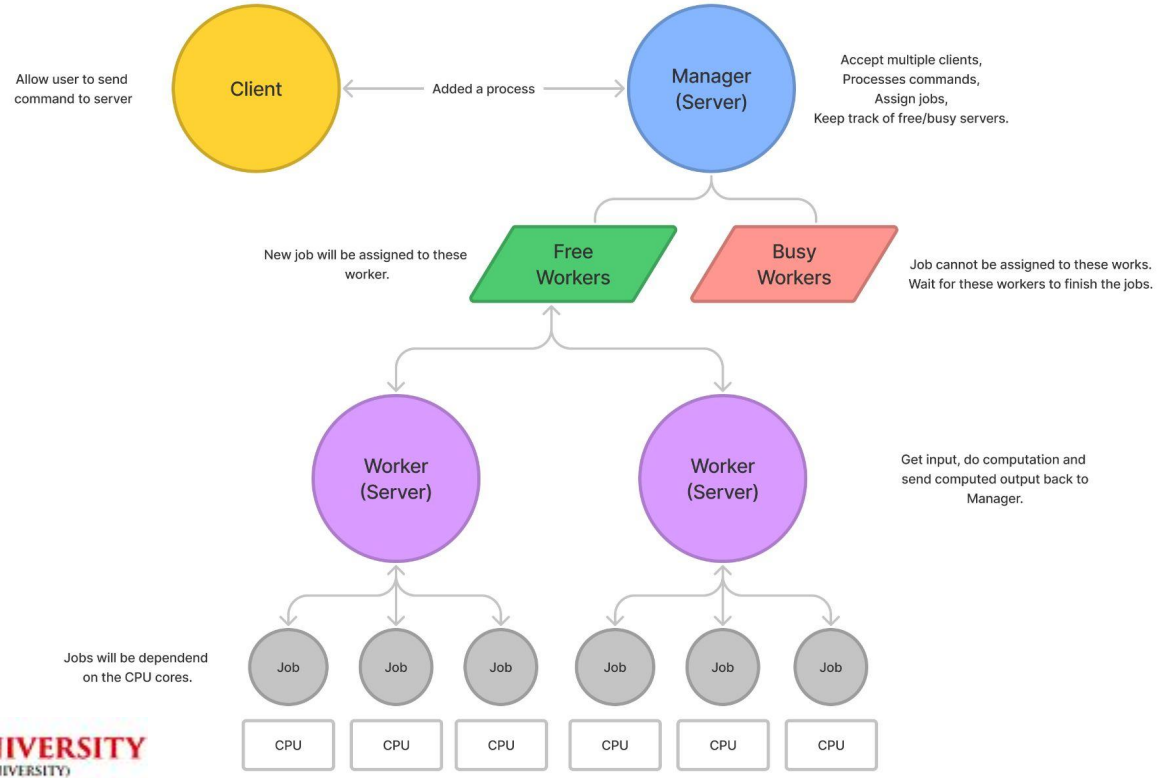
Propose Solution

This project includes multiple clients that can send two matrices to the Manager(Server) for computing the multiplication. The Manager is responsible for dividing the task and sending them to Workers (Servers), getting the computed result from Workers, and then merging all the task results and sending it to the client.

This project uses

- Distributed System Architecture
- Sockets for client/server communication
- Multithreading in Servers (Workers and Manager)
- Matrix Multiplication

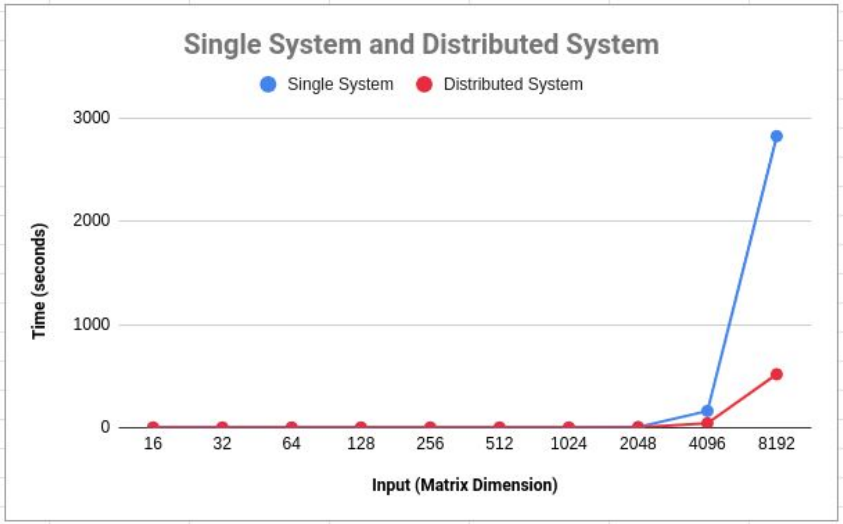
Architecture Diagram



Performance

	16	32	64	128	256	512	1024	2048	4096	8192
Single System	1.106	1.1	1.066	1.032	1.042	1.09	1.277	6.58	162.451	2825.387
Distributed System	1.106	1.061	1.021	1.025	1.042	1.092	1.287	4.025	43.183	517.781

System specs:
Processor: Intel® Core™ i7-9700 CPU @ 3.00GHz × 8
RAM: 31.3 GiB
System type: 64-bit Operating System, x64-based processor



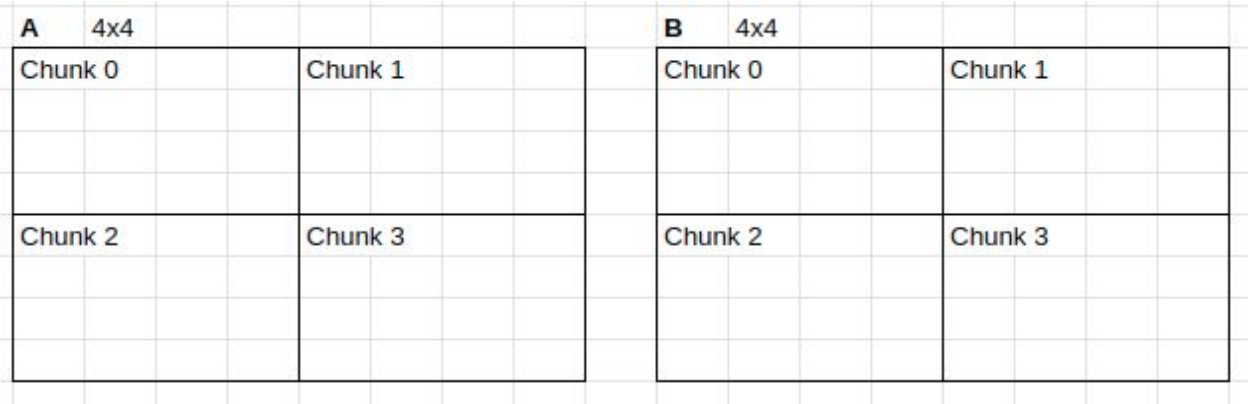
Methodology

Let’s take an 8×8 matrices

A	8x8									B	8x8						
1	2	3	4	5	6	7	8		1	2	3	4	5	6	7	8
1	2	3	4	5	6	7	8		1	2	3	4	5	6	7	8
1	2	3	4	5	6	7	8		1	2	3	4	5	6	7	8
1	2	3	4	5	6	7	8	x	1	2	3	4	5	6	7	8
1	2	3	4	5	6	7	8		1	2	3	4	5	6	7	8
1	2	3	4	5	6	7	8		1	2	3	4	5	6	7	8
1	2	3	4	5	6	7	8		1	2	3	4	5	6	7	8
1	2	3	4	5	6	7	8		1	2	3	4	5	6	7	8

Methodology (cont.)

We will send these two matrices to our Manager. If required, the manager will add zero paddings to these matrices. Now partition these matrices into a size of 4.



Methodology (cont.)

After dividing both matrices into chunks, we will now create tasks for our workers.

Task #0	Matrix A Chunk 0	x	Matrix B Chunk 0	+	Matrix A Chunk 1	x	Matrix B Chunk 2
Task #1	Matrix A Chunk 0	x	Matrix B Chunk 1	+	Matrix A Chunk 1	x	Matrix B Chunk 3
Task #2	Matrix A Chunk 2	x	Matrix B Chunk 0	+	Matrix A Chunk 3	x	Matrix B Chunk 2
Task #3	Matrix A Chunk 2	x	Matrix B Chunk 1	+	Matrix A Chunk 3	x	Matrix B Chunk 3

Methodology (cont.)

We will create a thread for each task depending on the free workers. Now we will have an output for each task from our workers and stored all the output in the result array.

Outputs				
Task #0	36	72	108	144
Chunk 0	36	72	108	144
	36	72	108	144
	36	72	108	144
Task #1	180	216	252	288
Chunk 1	180	216	252	288
	180	216	252	288
	180	216	252	288
Task #2	36	72	108	144
Chunk 2	36	72	108	144
	36	72	108	144
	36	72	108	144
Task #3	180	216	252	288
Chunk 3	180	216	252	288
	180	216	252	288
	180	216	252	288

Methodology (cont.)

We will merge all the output in a single matrix and will remove zero paddings if needed. (to make it 2^n)

AxB							
36	72	108	144	180	216	252	288
36	72	108	144	180	216	252	288
36	72	108	144	180	216	252	288
36	72	108	144	180	216	252	288
36	72	108	144	180	216	252	288
36	72	108	144	180	216	252	288
36	72	108	144	180	216	252	288
36	72	108	144	180	216	252	288

Future Scope

- This approach can be used to do heavy computations, and will be very helpful for machine learning application that require training on larger datasets.
- The future of the distributed system is bright, challenging, exciting, and filled with the promise of providing the tools for others to crack problems that we don’t even know exist yet.

Q/A