

DISTRIBUTED SYSTEMS ASSIGNMENT REPORT



ASSIGNMENT REPORT

Assignment ID: Assignment1 - Parallel Matrix Multiplication

Student Name: 王谦益

Student ID: 12111003

DESIGN

- *Module design: structure block - flow chart and description*
 - **Main Modules**
 - **Matrix Initialization and Output:**
 - Functions to print 1D and 2D arrays, like `outputArr_1d`, `outputArr_2d`, `ouputInfo`.
 - **Matrix Multiplication:**
 - `brute_force_matmul` handles matrix multiplication.
 - **MPI Communication:**
 - Uses `MPI_Scatterv`, `MPI_Bcast`, `MPI_Gatherv` to split work across processes and gather results.
 - **Result Comparison:**
 - The result is compared with brute force multiplication for validation.
 - **Steps**
 1. **Initialization:** Process 0 initializes matrices.
 2. **Scatter Operation:** Matrices are divided among processes.
 3. **Broadcast:** Matrix B is broadcasted to all processes.
 4. **Computation:** Each process computes its portion.
 5. **Gather Operation:** Results are gathered at the root process.
 6. **Validation:** Compare with brute force result.
 7. **Finalization:** MPI finalizes the execution.

- *Class design: UML class diagram and description*

This is a procedural program, so there are no classes. But, conceptually:

- **Matrix Class**

- Attributes: `size`, `data[]`.
- Methods: `outputArr_1d()`, `outputArr_2d()`, `multiply()`.

- **MPI_Manager Class**

- Attributes: `rank`, `world_size`.
- Methods: `MPI_Scatterv()`, `MPI_Gatherv()`, `MPI_Finalize()`.

RUNNING RESULT

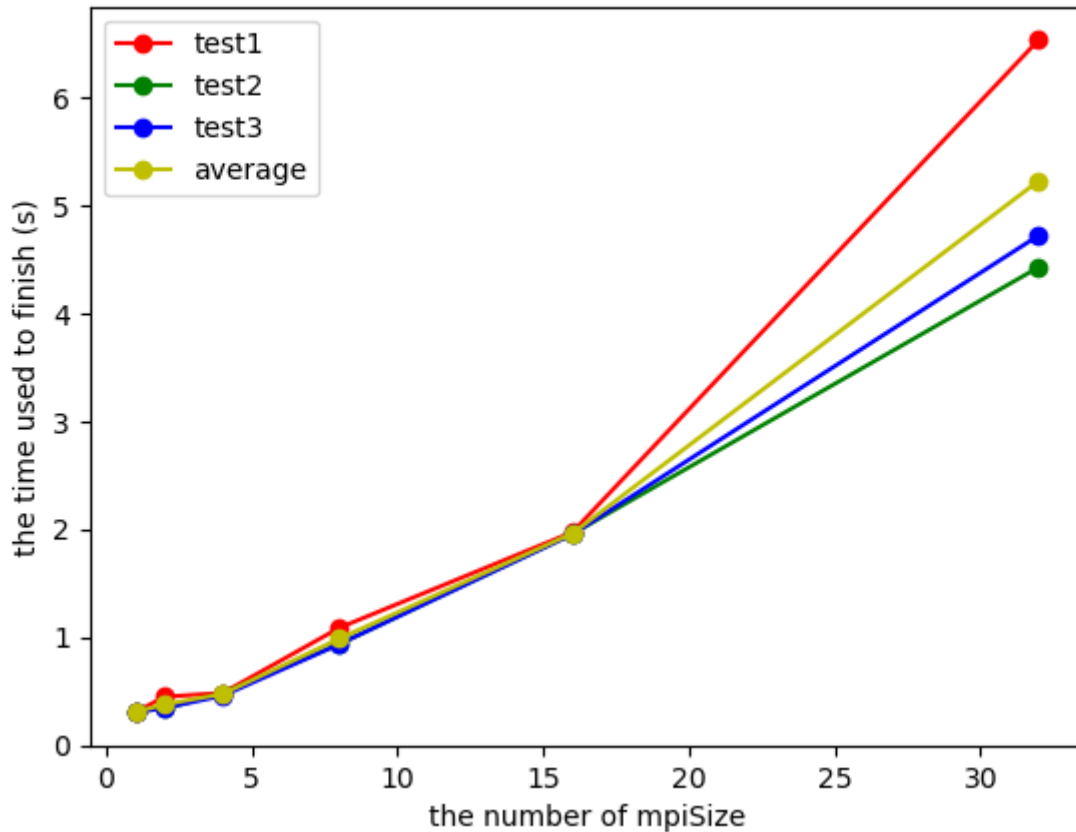
```

root@DESKTOP-Q3V6AG6:/home/win/CS328_Distributed_and_Cloud_Computing/assignment/assignment1# mpirun --allow-run-as-root --oversubscribe -np 1 ./mpi_files/mpi_matr
ix
Done in 0.326772 seconds.
Result is correct.
root@DESKTOP-Q3V6AG6:/home/win/CS328_Distributed_and_Cloud_Computing/assignment/assignment1# mpirun --allow-run-as-root --oversubscribe -np 2 ./mpi_files/mpi_matr
ix
Done in 0.370660 seconds.
Result is correct.
root@DESKTOP-Q3V6AG6:/home/win/CS328_Distributed_and_Cloud_Computing/assignment/assignment1# mpirun --allow-run-as-root --oversubscribe -np 4 ./mpi_files/mpi_matr
ix
Done in 0.515506 seconds.
Result is correct.
root@DESKTOP-Q3V6AG6:/home/win/CS328_Distributed_and_Cloud_Computing/assignment/assignment1# mpirun --allow-run-as-root --oversubscribe -np 8 ./mpi_files/mpi_matr
ix
Done in 1.288955 seconds.
Result is correct.
root@DESKTOP-Q3V6AG6:/home/win/CS328_Distributed_and_Cloud_Computing/assignment/assignment1# mpirun --allow-run-as-root --oversubscribe -np 16 ./mpi_files/mpi_matr
ix
Done in 2.433692 seconds.
Result is correct.
root@DESKTOP-Q3V6AG6:/home/win/CS328_Distributed_and_Cloud_Computing/assignment/assignment1# mpirun --allow-run-as-root --oversubscribe -np 32 ./mpi_files/mpi_matr
ix
Done in 5.698805 seconds.
Result is correct.

```

- use different matrix size to test the performance, and the time cost increases as the matrix size increases.

relationship between number of processes and computation latency

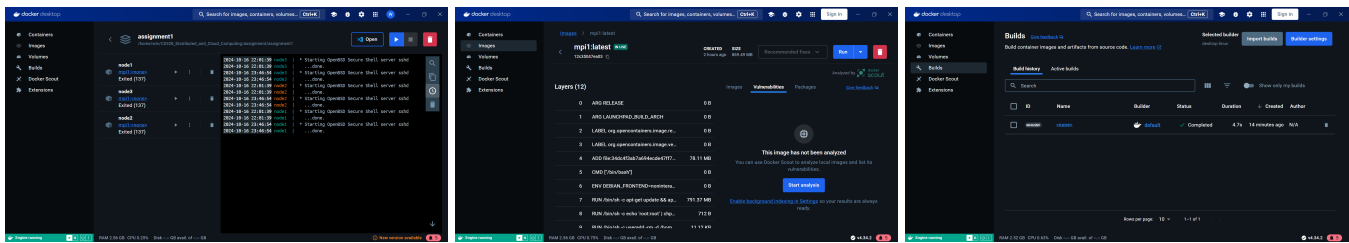


- run the program for three times to get the average time cost, and the result shows that the time cost decreases as the number of processes increases.

```

=> [internal] load metadata for docker.io/library/ubuntu:latest 4.7s
=> [internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> [1/6] FROM docker.io/library/ubuntu:latest@sha256:d4f6f70979d0758d7a6f81e34a61195677f4f4fa576eaf808b79f17499fd93d1 0.0s
=> CACHED [2/6] RUN apt-get update && apt-get install -y build-essential openssh-server openmpi-bin openmpi-common libopenmpi-dev 0.0s
=> CACHED [3/6] RUN echo 'root:root' | chpasswd 0.0s
=> CACHED [4/6] RUN useradd -m -d /home/mpi -s /bin/bash -g root -G sudo -u 1001 mpi && echo 'mpi:mpi' | chpasswd 0.0s
=> CACHED [5/6] RUN mkdir -p /home/mpi/.ssh && ssh-keygen -q -t rsa -N '' -f /home/mpi/.ssh/id_rsa && cat /home/mpi/.ssh/id_rsa.pub >> /home/mpi/.ssh/authorized_keys 0.0s
=> CACHED [6/6] WORKDIR /home/mpi 0.0s
=> exporting to image 0.0s
=> => exporting layers 0.0s
=> => writing image sha256:12c35047ea0345ded4d624ae547192d93da9375a313ea4dd6f5a44c143e1365d 0.0s
=> => naming to docker.io/library/mpi1 0.0s
root@DESKTOP-Q3V6AG6:/home/win/CS328_Distributed_and_Cloud_Computing/assignment/assignment1# docker compose up -d
[+] Running 3/3
  ✓ Container node1 Started 0.9s
  ✓ Container node2 Started 0.9s
  ✓ Container node3 Started 0.9s
root@DESKTOP-Q3V6AG6:/home/win/CS328_Distributed_and_Cloud_Computing/assignment/assignment1# docker exec -it node1 /bin/bash
root@node1:/home/mpi# su mpi
mpi@node1:~$ echo node1 slots=2 > host
mpi@node1:~$ echo node2 slots=2 >> host
mpi@node1:~$ echo node3 slots=2 >> host
mpi@node1:~$ cat host
node1 slots=2
node2 slots=2
node3 slots=2
mpi@node1:~$ sudo chmod 777 ./mpi_files/mpi_matrix
[sudo] password for mpi:
mpi@node1:~$ mpirun -n 6 --hostfile host ./mpi_files/mpi_matrix
Done in 1.107703 seconds.
Result is correct.
mpi@node1:~$
  
```

- setup 3 docker container with 2 slots in each and run the program in docker container.



- Some screenshots of the docker from docker desktop.

PROBLEMS

- *Problems and Solutions*
 - Segmentation fault
 - when I run the program with `MAT_SIZE > 300`, I got a segmentation fault.

```
-----
Primary job  terminated normally, but 1 process
returned a non-zero exit code. Per user-direction,
the job has been aborted.
-----
```

```
-----
mpirun noticed that process rank 0 with PID 0 on
node DESKTOP-Q3V6AG6 exited on signal 11 (Segmentation
fault).
-----
```

- The problem is caused by the memory overflow. There are too many arrays used in the program, which leads to the memory overflow, especially when the matrix size is large.
- To solve this problem, I deleted some useless arrays and use `malloc()` and `free()` to decrease the memory usage.