DISTRIBUTED SYSTEMS ASSIGNMENT REPORT

Assignment ID: 1

Student Name: 罗嘉俊

Student ID: 12012023

Design

> 💡 Scatter the left matrix `A` row-wise and broadcast the right matrix `B`.

The entire process is divided into the following phases:

1. Master generates random matrix A and B, we want to obtain A*B.
2. Scatter matrix A row-wise
3. Broadcast matrix B
4. Worker nodes obtain their own sub-results via local matrix multiplication.
5. Master gathers sub-results.

Implementation

Store the matrix in a row-major fashion, as it simplifies the scatter operation.

```
using Matrix = std::vector<double>;
```

To obtain the result `A*B`, start by initializing two matrices from uniform distribution: `A` and `B`.

```
Matrix initializeRandomMatrix(int rows, int cols) {
    std::random_device rd;
    std::mt19937 gen(rd());
    std::uniform_real_distribution<> dis(0.0, 1.0);
    Matrix m(rows * cols);
    for (int i = 0; i < rows * cols; i++) {
        m[i] = dis(gen);
    }
    return m;
}
```

First scatter matrix `A` row-wise, since it is an uneven split, it should use `MPI_Scatterv`.

```
MPI_Scatterv(&A[0], &sendCounts[0], &displacements[0], MPI_DOUBLE, &localA[0], sendCounts[rank], MPI_DOUBLE, 0, MPI_COMM_WORLD);
```

Simply broadcast matrix `B` .

```
MPI_Bcast(&B[0], MATRIX_SIZE * MATRIX_SIZE, MPI_DOUBLE, 0, MPI_COMM_WORLD);
```

Each node perform its local multiplication

```
Matrix subResult = multiply(localA, B, sendRowCounts[rank]);
```

The master node then gather all sub-results, since its an uneven split, use `MPI_Gatherv`

```
MPI_Gatherv(&subResult[0], sendCounts[rank], MPI_DOUBLE, &result[0], &sendCounts[0], &displacements[0], MPI_DOUBLE, 0, MPI_COMM_WORLD);
```

After obtaining the result, compare it with the result obtained through brute-force method.

```
bool compareMatrices(const Matrix &A, const Matrix &B) {
    for (int i = 0; i < MATRIX_SIZE * MATRIX_SIZE; i++) {
        if (std::abs(A[i] - B[i]) > 1e-9) {
            return false;
        }
    }
    return true;
}
```

Evaluation

Experiment Setup

The experiment is conducted on docker containers.

```
clover@DESKTOP-1MPCHVJ:~$ docker ps -a
CONTAINER ID   IMAGE        COMMAND       CREATED       STATUS                   PORTS     NAMES
cb0a8cf1b1d1   cs328_node   "/bin/bash"   7 hours ago   Exited (0) 6 hours ago             nodeA
bc26b78b1eea   cs328_node   "/bin/bash"   7 hours ago   Exited (0) 6 hours ago             nodeB
```

OpenMPI and SSH server are installed on these containers.

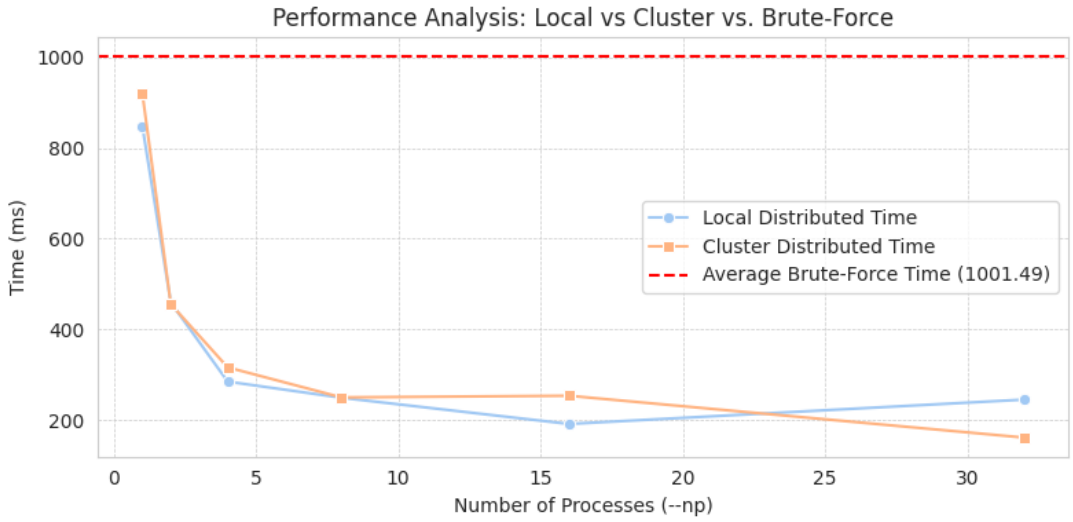
They reside in the same network.

```
"Containers": {
            "bc26b78b1eeacffecee5fc4138b326941878823ada7798f8cd3975482c7b8615": {
                "Name": "nodeB",
                "EndpointID": "23f40ecf7d7542228311a7de4d63151c23eaa9e81f0920d7e941d4d3ae14ae0c",
                "MacAddress": "02:42:ac:12:00:03",
                "IPv4Address": "172.18.0.3/16",
                "IPv6Address": ""
            },
            "cb0a8cf1b1d135d51f7f06935a5c9efff9a52bc2f2586762dd5b0e338185dbf9": {
                "Name": "nodeA",
                "EndpointID": "1c61f335bc056a835002a80db593469fd565911093293c6245b8e7225dc91caf",
                "MacAddress": "02:42:ac:12:00:02",
                "IPv4Address": "172.18.0.2/16",
                "IPv6Address": ""
            }
        },
```

Results

1. Distributed approach is generally faster than brute force.
2. As the number of processes increases, the performance improves.
3. However, as the number of processes exceeded the number of slots(cores), the performance won't improve any more.

4. Cluster is slower than local since when number of process is small, since clustering brings communication overhead.
5. Cluster is faster than local when the process count is large because it has more processes or the scheduling of Docker gives two containers more resources than one.



Screenshots

Run on single container (`nodeA`).

```
clover@DESKTOP-1MPCHVJ:~$ docker start nodeA
nodeA
clover@DESKTOP-1MPCHVJ:~$ docker attach nodeA
root@cb0a8cf1b1d1:/# ls
bin   dev  home  lib32  libx32  mnt  proc     root  sbin  sys  usr
boot  etc  lib   lib64  media   opt  project  run   srv   tmp  var
root@cb0a8cf1b1d1:/# cd project
root@cb0a8cf1b1d1:/project# ls
labs  matmul
root@cb0a8cf1b1d1:/project# cd matmul
root@cb0a8cf1b1d1:/project/matmul# sh run.sh
Running in LOCAL mode
Number of experiments: 1
Experiment run: 1
-----------------------------
| np  | Distributed |   BF   |
-----------------------------
| 1   | 911.46      | 860.967 |
-----------------------------
| 2   | 508.58      | 889.619 |
-----------------------------
| 4   | 300.068     | 950.085 |
-----------------------------
| 8   | 328.412     | 1593.18 |
-----------------------------
```

Start container `nodeB` .

```
clover@DESKTOP-1MPCHVJ:~$ docker start nodeB
nodeB
clover@DESKTOP-1MPCHVJ:~$ docker attach nodeB
root@bc26b78b1eea:/# service ssh start
 * Starting OpenBSD Secure Shell server sshd
root@bc26b78b1eea:/#
```

Run experiment using 2 containers.

```
root@cb0a8cf1b1d1:/project/matmul# sh run.sh -c
Running in CLUSTER mode
Number of experiments: 1
Experiment run: 1
-----------------------------
| np  | Distributed |   BF   |
-----------------------------
| 1   | 868.032     | 871.225 |
-----------------------------
| 2   | 621.47      | 1255.88 |
-----------------------------
| 4   | 432.88      | 1450.76 |
-----------------------------
```

Challenges

Challenge 1: split the matrix unevenly.

Solution: use `scatterv` and `gatherv` .

Challenge 2: OpenMPI error when `—-np` is larger than slots(cores).

Solution: use `--oversubscribe`

Challenge 3: My computer don’t have enough space for 2 VM, I don’t have time to configure 2 VM either.

Solution: use docker containers

Challege 4: Enable docker containers to communicate with each other.

Solution: add `—-network` when create containers, putting them into the same subnetwork.