COMP 6231 Assignment 3

Task 2: MPI Commands Explanation

Qiang Xue (40300671) Yicheng Cai (26396283) Yikai Chen (40302669)
Yifan Wu (40153584) Alexander Sutherland (40321783)

November 5, 2025

1 Overview

The MPI cluster architecture utilizes Docker Swarm for container orchestration, enabling parallel data processing across multiple containers. The master node distributes dataset rows to worker nodes, which process data in parallel using MPI and pandas, then aggregate final results back to the master.

2 Node Configuration

2.1 Docker Image Build

Dockerfile Content:

```
FROM python:3.10-slim

ENV DEBIAN_FRONTEND=noninteractive

RUN apt-get update && apt-get install -y --no-install-recommends \
    openssh-server \
    openmpi-bin \
    libopenmpi-dev \
    && rm -rf /var/lib/apt/lists/*

RUN pip install --no-cache-dir mpi4py
RUN pip install pandas

RUN mkdir -p /app
WORKDIR /app

RUN mkdir -p /root/.ssh && chmod 700 /root/.ssh

COPY id_rsa /root/.ssh/
COPY id_rsa.pub /root/.ssh/
RUN chmod 600 /root/.ssh/id_rsa && \
    cat /root/.ssh/id_rsa.pub >> /root/.ssh/authorized_keys && \
    chmod 600 /root/.ssh/authorized_keys

RUN echo "    StrictHostKeyChecking no" >> /etc/ssh/ssh_config && \
    echo "    UserKnownHostsFile /dev/null" >> /etc/ssh/ssh_config
```

```
ENV PATH_DATASET="/app/Books_rating.csv"

CMD ["/usr/sbin/sshd", "-D"]
```

Explanation of Dockerfile Layers:

- `FROM python:3.10-slim`: Uses a lightweight Python 3.10 base image
- `ENV DEBIAN_FRONTEND=noninteractive`: Prevents apt-get from prompting for user input
- `RUN apt-get install openssh-server`: Installs SSH server for inter-container communication
- `RUN apt-get install openmpi-bin libopenmpi-dev`: Installs OpenMPI libraries and binaries for parallel processing
- `RUN pip install mpi4py`: Installs Python bindings for MPI
- `RUN pip install pandas`: Installs pandas library for data processing
- `COPY id_rsa`: Copies locally generated SSH keys into container for passwordless authentication between containers
- `chmod 600`: Sets secure permissions on SSH keys
- `StrictHostKeyChecking no`: Disables SSH host key verification for automatic connection
- `ENV PATH_DATASET`: Sets environment variable for dataset location
- `CMD ["/usr/sbin/sshd", "-D"]`: Starts SSH daemon in foreground

Build MPI Image:

```
docker build -t mpi-node .
```

Explanation: Creates a Docker image named `mpi-node`. This single image is used for both master and worker nodes, providing a consistent environment across the cluster. The `-t` flag tags the image, and the `.` indicates the build context.

2.2 Generate SSH Keys

Generate SSH Key Pair:

```
ssh-keygen -t rsa -b 2048 -f id_rsa -N ""
```

Explanation: Generates an RSA SSH key pair locally for passwordless authentication between MPI containers.

- `-t rsa`: Specifies RSA as the key type
- `-b 2048`: Sets key size to 2048 bits
- `-f id_rsa`: Names the private key file as `id_rsa` (public key automatically named `id_rsa.pub`)
- `-N ""`: Sets empty passphrase for automated SSH connections

This generates two files in the current directory: `id_rsa` (private key) and `id_rsa.pub` (public key), which will be copied into all containers during the Docker build process.

2.3 Prepare Required Files

Place the following files in the same directory:

- **Dataset:** `Books_rating.csv` - The dataset to be processed
- **MPI Scripts:** `q1_t3.py, q2_t3.py, q3_t3.py, q4_t3.py` - Python scripts for data analysis
- **SSH Keys:** `id_rsa, id_rsa.pub` - SSH key pair generated in previous step

2.4 Distribute Image to Worker Nodes

Save and Transfer Image:

```
# Save the image to a tar file
docker save -o mpi-node.tar mpi-node

# Transfer to worker machines
scp mpi-node.tar user@worker-machine:/path/to/directory

# Load on each worker machine
docker load -i mpi-node.tar
```

Explanation: Since the MPI image is built on the manager node, it must be distributed to all worker machines in the Docker Swarm cluster. The `docker save` command exports the image to a tar archive, which is then transferred via `scp` to each worker machine. On each worker, `docker load` imports the image, making it available for container deployment. This ensures all physical machines have the `mpi-node` image before launching worker containers.

2.5 Run Master Container

Launch Master Node:

```
docker run -d --name mpi-head --hostname mpi-head
    --network 6231-net -v "$(pwd):/app:ro" mpi-node
```

Explanation - Command Parameters:

- `-d`: Runs container in detached mode (background)
- `--name mpi-head`: Names the container for easy reference
- `--hostname mpi-head`: Sets hostname for network identification
- `--network 6231-net`: Connects to the overlay network created in Task 1
- `-v "$(pwd):/app:ro"`: Mounts current directory to /app as read-only, sharing dataset and scripts
- `mpi-node`: Uses the previously built image

2.6 Run Worker Containers on Master Node

Launch Worker 1:

```
docker run -d --name mpi-worker1 --hostname mpi-worker1
    --network 6231-net -v "$(pwd):/app:ro" mpi-node
```

Launch Worker 2:

```
docker run -d --name mpi-worker2 --hostname mpi-worker2
    --network 6231-net -v "$(pwd):/app:ro" mpi-node
```

Explanation: Creates two worker containers on the master physical machine. Each worker has access to the same shared data via the volume mount and can communicate with other nodes via SSH over the overlay network.

3 Worker Node Configuration

3.1 Second Physical Computer - Workers 3, 4, 5

Launch Worker 3:

```
docker run -d --name mpi-worker3 --hostname mpi-worker3
    --network 6231-net -v "$(pwd):/app:ro" mpi-node
```

Launch Worker 4:

```
docker run -d --name mpi-worker4 --hostname mpi-worker4
    --network 6231-net -v "$(pwd):/app:ro" mpi-node
```

Launch Worker 5:

```
docker run -d --name mpi-worker5 --hostname mpi-worker5
    --network 6231-net -v "$(pwd):/app:ro" mpi-node
```

Explanation: Workers run on the second physical machine in the cluster. The volume mount path `$(pwd)` should contain the same files (dataset and scripts) on this machine.

3.2 Third Physical Computer - Worker 6

Launch Worker 6:

```
docker run -d --name mpi-worker6 --hostname mpi-worker6
    --network 6231-net -v "$(pwd):/app:ro" mpi-node
```

Explanation: Single worker on the third physical machine, demonstrating the flexibility of distributing containers across different physical hosts.

3.3 Fourth Physical Computer - Workers 7, 8, 9, 10

Launch Worker 7:

```
docker run -d --name mpi-worker7 --hostname mpi-worker7
    --network 6231-net -v "$(pwd):/app:ro" mpi-node
```

Launch Worker 8:

```
docker run -d --name mpi-worker8 --hostname mpi-worker8
    --network 6231-net -v "$(pwd):/app:ro" mpi-node
```

Launch Worker 9:

```
docker run -d --name mpi-worker9 --hostname mpi-worker9
    --network 6231-net -v "$(pwd):/app:ro" mpi-node
```

Launch Worker 10:

```
docker run -d --name mpi-worker10 --hostname mpi-worker10
    --network 6231-net -v "$(pwd):/app:ro" mpi-node
```

Explanation: Four workers on the fourth physical machine, completing the 10-worker setup for scalability testing.

4 MPI Execution Commands

4.1 Access Master Container

Enter Master Container:

```
docker exec -it mpi-head bash
```

Explanation: Opens an interactive bash shell inside the running mpi-head container. The `-it` flags enable interactive terminal mode, allowing execution of MPI commands from within the master node.

5 Launching and Executing MPI Applications

5.1 General MPI Launch Command

Launch Application with MPI:

```
mpirun --allow-run-as-root --mca btl_tcp_if_include eth0
    --mca oob_tcp_if_include eth0 -n <processes>
    --host <hostlist> -x PATH_DATASET="/app/Books_rating.csv"
    python <script.py>
```

Explanation: This command launches MPI applications across the distributed cluster. The parameters control process distribution, network configuration, and environment variables.

- `--allow-run-as-root`: Permits MPI to run as root user (necessary in containers)
- `--mca btl_tcp_if_include eth0`: Configures MPI to use eth0 network interface for byte transfer layer
- `--mca oob_tcp_if_include eth0`: Configures MPI to use eth0 for out-of-band communication
- `-n <processes>`: Specifies the total number of MPI processes to launch (varies from 5 to 11 in experiments)

- `--host <hostlist>`: Comma-separated list of container hostnames where processes will run
- `-x PATH_DATASET`: Exports environment variable to all MPI processes
- `python <script.py>`: The Python script to execute in parallel (q1_t3.py, q2_t3.py, q3_t3.py, or q4_t3.py)

Example - Running with 6 Processes:

```
mpirun --allow-run-as-root --mca btl_tcp_if_include eth0
    --mca oob_tcp_if_include eth0 -n 6
    --host mpi-head,mpi-worker1,mpi-worker2,mpi-worker3,mpi-worker6,mpi-worker9
    -x PATH_DATASET="/app/Books_rating.csv" python q1_t3.py
```

Explanation: This example launches 6 MPI processes distributed across 6 containers (master and 5 workers from different physical machines). The dataset is processed in parallel by all 6 processes, with results aggregated on the master node.

5.2 Executing Different Analysis Scripts

Execute Analysis Scripts:

```
# Question 1: Top 5 ratings distribution
mpirun --allow-run-as-root --mca btl_tcp_if_include eth0
    --mca oob_tcp_if_include eth0 -n <processes>
    --host <hostlist> -x PATH_DATASET="/app/Books_rating.csv"
    python q1_t3.py

# Question 2: Maximum positive review
mpirun --allow-run-as-root --mca btl_tcp_if_include eth0
    --mca oob_tcp_if_include eth0 -n <processes>
    --host <hostlist> -x PATH_DATASET="/app/Books_rating.csv"
    python q2_t3.py

# Question 3: Minimum positive review
mpirun --allow-run-as-root --mca btl_tcp_if_include eth0
    --mca oob_tcp_if_include eth0 -n <processes>
    --host <hostlist> -x PATH_DATASET="/app/Books_rating.csv"
    python q3_t3.py

# Question 4: Average ratings and reviews
mpirun --allow-run-as-root --mca btl_tcp_if_include eth0
    --mca oob_tcp_if_include eth0 -n <processes>
    --host <hostlist> -x PATH_DATASET="/app/Books_rating.csv"
    python q4_t3.py
```

Explanation: All scripts are executed using the full mpirun command with MPI parameters. To execute different analyses, simply change the script name in the final parameter (q1_t3.py, q2_t3.py, q3_t3.py, or q4_t3.py).
For scalability testing, experiments were conducted with varying container counts (4-10 containers) by adjusting:

- **Process count (`-n`)**: Set to match the number of containers (4, 5, 6, 7, 8, 9, or 10 processes)
- **Host list (`--host`)**: Comma-separated container names to include in execution (e.g., mpi-head,mpi-worker1,mpi-worker2,...)

6 Network and Storage Configuration

6.1 Network Configuration

- **Network Type:** Docker overlay network (6231-net)
- **Communication Protocol:** SSH over TCP/IP, MPI over TCP/IP
- **Network Interface:** eth0 (configured in MPI commands)
- **Ports Used:**
  - SSH: Port 22 (for inter-container communication)
  - MPI: Dynamic ports assigned by OpenMPI
  - Docker Swarm: Port 2377 (management), 7946 (node communication), 4789 (overlay network)

6.2 Storage Configuration

- **Shared Storage:** Each physical machine has its own copy of the dataset and scripts
- **Volume Mount:** `-v "$(pwd):/app:ro"` mounts host directory to container in read-only mode
- **Dataset Location:** `/app/Books_rating.csv` (consistent across all containers)
- **Results Storage:** Results are aggregated on the master node (mpi-head)

6.3 Cluster Distribution Summary

Physical Machine	Containers	Role
Manager	mpi-head, mpi-worker1, mpi-worker2	Master + 2 Workers
Worker Computer 1	mpi-worker3, mpi-worker4, mpi-worker5	3 Workers
Worker Computer 2	mpi-worker6	1 Worker
Worker Computer 3	mpi-worker7, mpi-worker8, mpi-worker9, mpi-worker10	4 Workers

Table 1: Cluster container distribution across physical machines

6.4 Container IP Address Mapping

Container Name	IP Address
mpi-head	10.0.1.76
mpi-worker1	10.0.1.78
mpi-worker2	10.0.1.79
mpi-worker3	10.0.1.72
mpi-worker4	10.0.1.74
mpi-worker5	10.0.1.75
mpi-worker6	10.0.1.80
mpi-worker7	10.0.1.85
mpi-worker8	10.0.1.84
mpi-worker9	10.0.1.65
mpi-worker10	10.0.1.67

Table 2: IP address assignments for all containers in the MPI cluster