高效能巨量資料與人工智慧系統 - HW3

You are asked to work in *a group of 3 people* to parallelize the "Standard Scaled Dot-Product Attention" in C with Open MPI.

1. Introduction

Standard Scaled Dot-Product Attention

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}(\frac{\mathbf{Q}\mathbf{K}^{\text{T}}}{\sqrt{d_k}})\mathbf{V}$$

- $\mathbf{Q} \in \mathbb{R}^{m \times d_k}$, $\mathbf{K} \in \mathbb{R}^{n \times d_k}$, $\mathbf{V} \in \mathbb{R}^{n \times d_v}$
- The softmax function here means applying softmax independently to each row of its argument.
  - i.e., $\mathbf{A}_i$ becomes $\sigma(\mathbf{A}_i)$, where $\mathbf{A}_i$ is the row i of the matrix $\mathbf{A}$.

Reference: Wikipedia

Softmax

When applying the softmax function σ to a vector $\mathbf{z}$, each element z_i becomes $\sigma(\mathbf{z})_i$.

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

- $\mathbf{z} = (z_1, z_2, \dots, z_K) \in \mathbb{R}^K$

Reference: Wikipedia

> **Tip**
>
> When you are implementing the softmax function σ, we **highly** encourage you to subtract all elements by the maximum value of them. Otherwise, you may encounter overflow problem!
>
> $$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} = \frac{e^{z_i - \max(\mathbf{z})}}{\sum_{j=1}^{K} e^{z_j - \max(\mathbf{z})}}$$

2. Requirements

1. Implement a serial attention computation code.
2. Implement an Open MPI-parallelized version of attention computation code.
3. Report and analyze the results.

> **General Tips About Your Report**
>
> - How matrix size influences the performance?
> - How number of processes influences the performance?
> - How number of nodes influences the performance?
> - What is the speedup of your parallelized code compared to the serial one with different number of nodes/processes?
> - How is the scalability of your work?
>   - in terms of strong scalability / weak scalability
> - What optimization techniques you applied?
> - Anything you want to share.

> **Note**
>
> You may use AVX instructions to further optimize your code to out stand in the competition.

3. Environment

- **OS:** Ubuntu 24.04 Server
- **CPUs:** 144
  - Sockets: 2
  - Cores / socket: 36
  - Threads / core: 2
- **Networking:** all nodes are connected to the same switch via Mellanox ConnectX-5 Adapter

There are 8 nodes in total. To accommodate all classmates, you are split into 2 groups:

Group	Teams	Accessible Nodes
Group 1	Team 1 -- Team 9	Node 0 -- Node 3
Group 2	Team 10 -- Team 19	Node 4 -- Node 7

Besides, each team should **not** create more than **16 computing processes per node** to reduce the intervention among your work.
For fairness, we will run your code individually to score your ranks.

SSH

Machine	IP Address	SSH Port
MPI Nodes	172.16.179.50 -- 172.16.179.57	22
Jump Server	140.112.90.37	9037

You can directly log in the MPI nodes if your client is in the subnet `172.16.0.0/16`.

NTU CSIE Students

For CSIE students at NTU, you can connect to the CSIE VPN before logging in any MPI node.

Other Students

For other students, we set up a jump server for you. You can first log in the jump server, and then log in any MPI node.

> ⚠ **Warning**
>
> Please do not execute computing processes on the jump server!

4. Execution

MPI Setup

> ⚠ **Warning**
>
> You should finish this part to run your MPI program.

1. Generate an SSH key pair on one of the MPI node (leave the passphrase empty):
   ```
   ssh-keygen -t ed25519
   ```
2. Make all the nodes you are allowed to access recognize your public key:
   ```
   ssh-copy-id -i ~/.ssh/id_ed25519.pub rdmaX
   ```
   `rdmaX` is the host name of `172.16.179.5X` recognizable among the MPI nodes. Remember to change the `X` into some number.
3. Copy the private key to all the nodes you are allowed to access:
   ```
   scp ~/.ssh/id_ed25519 rdmaX:.ssh
   ```

> ⚠ **Warning**
>
> It is recommended to just leave your private key on the MPI nodes for security's sake. For accessing from outside, generate another key pair on your PC or the jump server and run `ssh-copy-id` to send your public key to the MPI nodes.

MPI Run

> 🔥 **Important**
>
> You should place your code and testing data at the same absolute path on all MPI nodes. Remember to copy the testing data on all MPI nodes beforehand.
> Besides, after compiling your code with `mpicc`, remember to copy your compiled code to all MPI nodes under the same path.
>
> By the way, though the testing data are placed on all nodes, only the rank 0 process will read the data, and you are asked to distribute them from the rank 0 process.

We will run your code like this:

```
# only C's math library is linked for your code
mpicc attention-mpi.c -o attention-mpi -lm -march=native
# copy the compiled code to other MPI nodes
for i in `seq 1 3`; do scp attention-mpi rdma$i:; done
# we will run your code on 4 nodes with 16 processes per node
UCX_TLS=rc,sm,self \
UCX_NET_DEVICES=mlx5_0:1 \
UCX_LOG_LEVEL=info \
mpirun \
  --hostfile hosts \
  -npernode 16 \
  --mca pml ucx \
  --mca btl ^tcp \
  ./attention-mpi <testing data>
```

Content of `hosts`:

```
rdma0
rdma1
rdma2
rdma3
```

For testing, you can try different configurations of the host file. Just remind you here that different teams are allowed to access different group of nodes.

5. Submission

- Deadline: 11/13 (Thu) 23:59
- Submit a zip file named `Team_<Team Number>_HW3.zip` by one of your team members with the following files:
  - `attention.c`
    - your serial code
  - `attention-mpi.c`
    - your final optimized code
  - `Team_<Team Number>_HW3_report.pdf`
- Please make sure your source codes can be compiled and executed without any problem.
  If some modifications are required to compile and run your code, there will be a small points deduction.

6. Grading Policy

- Correctness: 10%
  - The points you earned in this part is proportional to the number of test cases you passed.
  - We accept an error of ± 0.02.
- Scalability: 10%
  - You will get full points as long as your parallelized code can show good scalability.
  - Please justify the scalability of your program in the report.
- Competition Rank: 40%
  - $\text{score} = \max(10, 40 \times (100 - 5 \times (\text{rank} - 1))\%)$
  - However, you will get 0% in this part if your program gives wrong answers.
- Report: 40%
  - The more thorough, reasonable, and clearer your analysis is, the more points you will get in this part.

> ⚠ **Warning:**
>
> - Your score will be multiplied by 70% if you submit your homework after the deadline.
> - Please follow the instructions in the template code and **do not modify any code that should not be changed.**
> - **Don't use multi-threading**

7. References

- MPI Tutorial
- Open MPI v4.1 Document