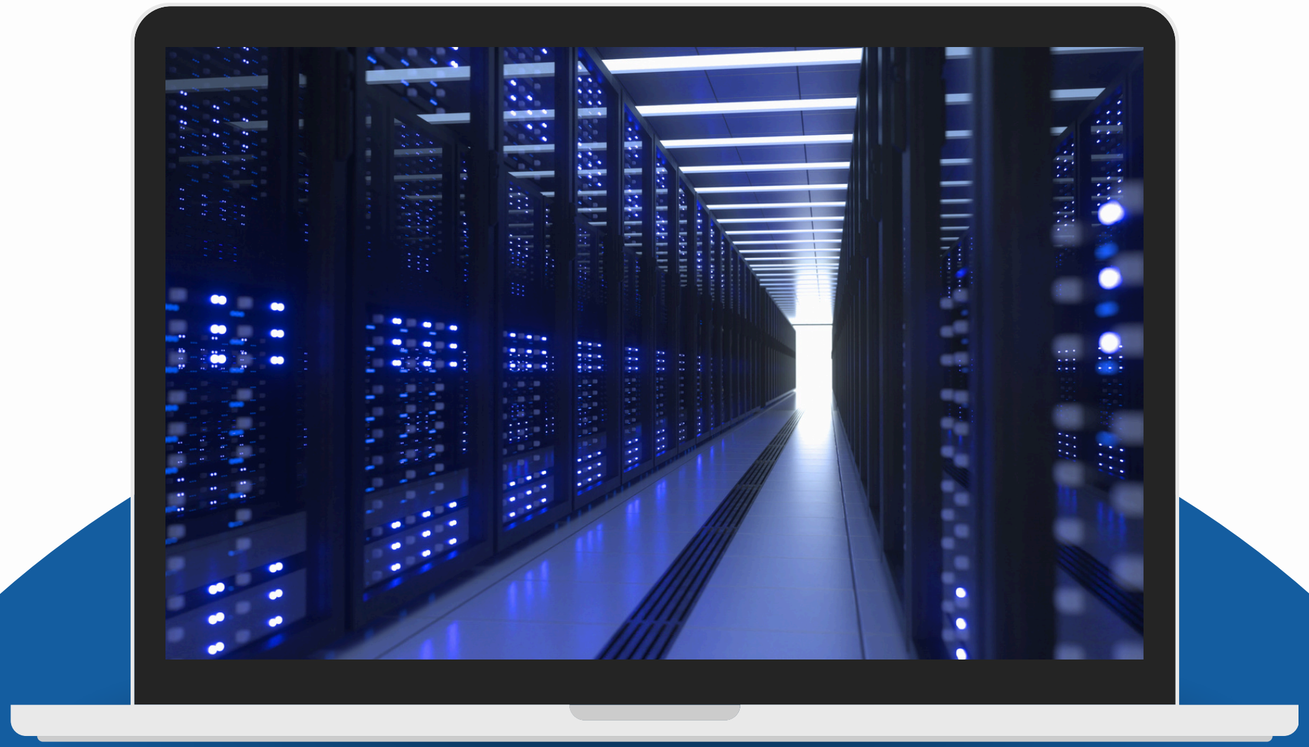
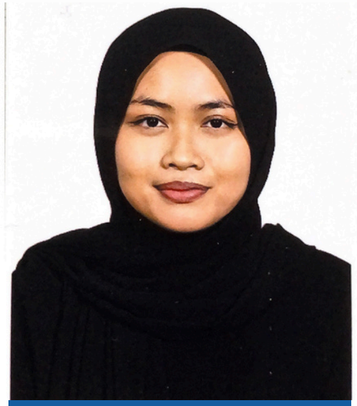
2024 APAC HPC-AI

Universiti Putra Malaysia
UPMTeam3

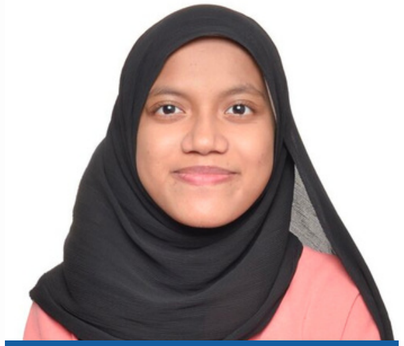


Our Team

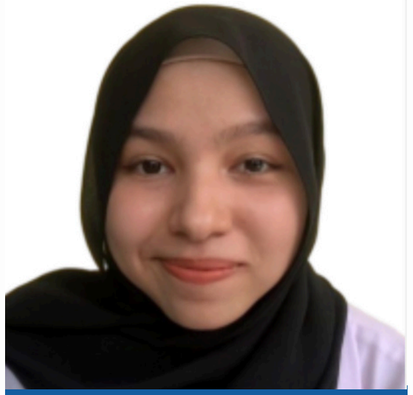
Anis Azman



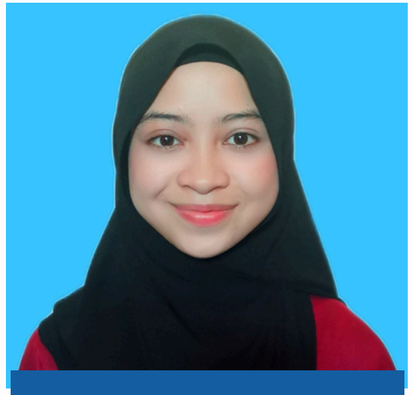
Nurul Farizatul Aina



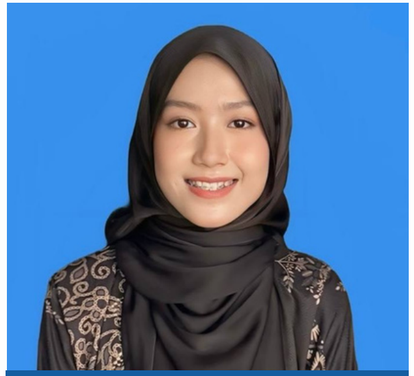
Maimunah Hosni



Wan Siti Aisyah



Azyan Syazwani



Siti Nurinsyirah

Universiti Putra Malaysia

Universiti Putra Malaysia (UPM) is a leading Malaysian university renowned for its expertise in agriculture and related fields. Founded in 1931, UPM has a long history of academic excellence and has grown to become a global leader in its area of expertise.

UPM offers a diverse range of programs, including food crops, animal husbandry, veterinary medicine, and forestry. The university also has expanded its scope to include science, technology, and other disciplines relevant to society and the national needs, ensuring graduates are well-prepared for the challenges of the modern world. UPM is committed to environmental sustainability and has consistently ranked highly in green university rankings.

OUR SUPERVISORS

MRS. Y.M. RAJA AZLINA BINTI RAJA MAHMOOD

PROF. MADYA DR. NOR ASILAH WATI BINTI ABDUL HAMID

Faculty of Computer Science and Information Technology (FCSIT)

In the 1980s, UPM started offering computer science courses to meet growing demand. By 1992, a dedicated department was established, offering various courses, including postgraduate programs.

In 1998, the Faculty of Computer Science and Information Technology was created to meet the increasing demand for computer education and keep up with technological advancements. Today, the faculty has four departments: Multimedia, Computer Science, Software Engineering & Information System, and Communication Technology & Networks.

Definitions

HOOMD-BLUE

HOOMD-blue is a molecular dynamics simulation tool used to study soft matter systems like polymers and colloids. It's designed to be easy to use, efficient, and flexible, making it a valuable resource for researchers in many fields

LITGPT LLAMA2

Lit-GPT is an open-source project that provides a lightweight and optimized version of large language models, like LLaMA 2, developed by Meta AI. LLaMA 2 is a powerful model used for tasks such as text generation and summarization.

Lit-GPT can run on regular computers, allowing researchers and developers to use them without needing powerful hardware.

Our Methodology

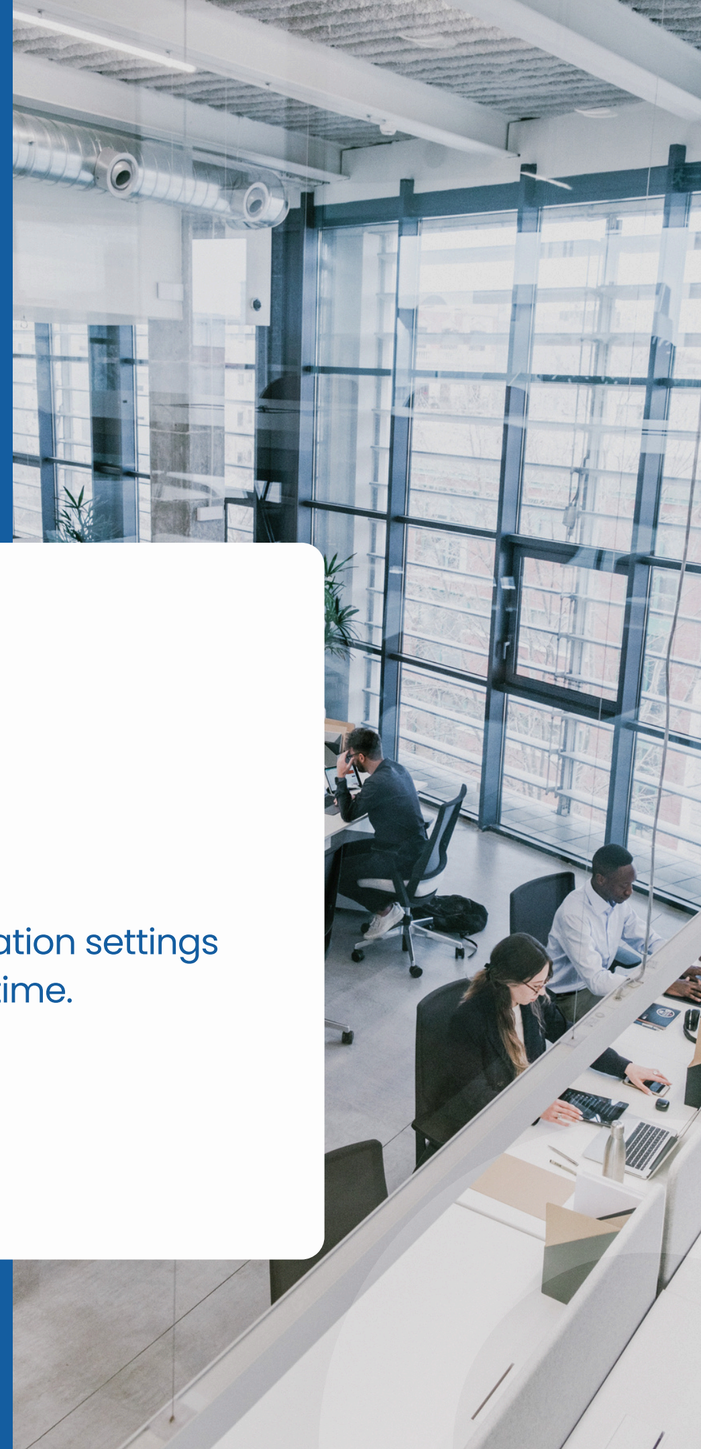
Improve performance, ideas, and tuning methods.

HOOMDBLUE

- Calculate efficiency using total cores and speedup.
- Try allocating different cores and nodes to increase the number of steps.
- Result: Comparison on different number of nodes.

LLAMA2

- Optimization on MPI communication settings
- Result: Comparison on training time.
- Finetune using LoRA



TASK 1
HOOMD-BLUE USING GADI

Performance Metrics

01

Steps per second – This measures how fast the system can simulate particle movements over time.

02

Execution time – This shows how long the task takes to complete.

03

Speedup – This measures how much faster a task completes when multiple processors are used compared to using a single processor.

04

Efficiency – This measures how effectively the processors are being used in parallel.

PBS Script Directives & Modules Loaded

```
#!/bin/bash
#PBS -j oe
#PBS -M 215014@student.upm.edu.my,214511@student.upm.edu.my
#PBS -m abe
#PBS -P zd64
#PBS -l ngpus=0
#PBS -l walltime=00:00::60
#PBS -l other=hyperthread
#-report-bindings \

module purge
module load ${HOME}/hpcx-v2.20-gcc-mlnx_ofed-redhat8-cuda12-x86_64/modulefiles/hpcx-ompi
```

hoomd.sh

PBS Commands Used

- qstat - To display queue information

```
(/home/552/sp1115/scratch/workdir/hoomd/hoomd.py312) [sp1115@gadi-login-05 run]$ qstat
Job id                  Name             User             Time Use S Queue
----------------------  ---------------- ---------------- -------- - -----
126480181.gadi-pbs      hoomd.nodes1.WS* sp1115           00:10:27 R normal-exec
```

- qdel - To delete a job

```
[sp1115@gadi-login-05 run]$ qdel 126480181.gadi-pbs
```

- cat <job file> - To display the contents of a file

```
(base) [sp1115@gadi-login-07 run]$ cat hoomd.nodes4.WS40000.BS80000.o126485280
time mpirun -host gadi-cpu-clx-2378.gadi.nci.org.au,gadi-cpu-clx-2379.gadi.nci.org.au,gadi-cpu-clx-2380.gadi.nci.or
adi-cpu-clx-2391.gadi.nci.org.au,gadi-cpu-clx-2396.gadi.nci.org.au,gadi-cpu-clx-2400.gadi.nci.org.au,gadi-cpu-clx-2
di.nci.org.au,gadi-cpu-clx-2411.gadi.nci.org.au -wdir /home/552/sp1115/scratch/workdir/hoomd -output-filename /home
p-by ppr:48:node -oversubscribe -use-hwthread-cpus -x PYTHONPATH=/home/552/sp1115/scratch/workdir/hoomd/build/hoomd
e/552/sp1115/scratch/workdir/hoomd/hoomd.py312/bin/python -m hoomd_benchmarks.md_pair_wca --device CPU -v -N 200000
Using existing initial_configuration_cache/hard_sphere_200000_1.0_3.gsd
notice(2): Using domain decomposition: n_x = 8 n_y = 8 n_z = 9.
Running MDPairWCA benchmark
.. warming up for 40000 steps
.. running for 80000 steps 1 time(s)
.. 4240.476353911216 time steps per second
4240.476353911216
1243.40user 278.41system 0:37.44elapsed 4063%CPU (0avgtext+0avgdata 441320maxresident)k
347488inputs+16840outputs (39780major+8426785minor)pagefaults 0swaps

================================================================================
                Resource Usage on 2024-10-09 19:06:01:
   Job Id:             126485280.gadi-pbs
   Project:            zd64
   Exit Status:        0
   Service Units:      12.80
   NCPUs Requested:    576                    NCPUs Used: 576
                                           CPU Time Used: 05:03:03
   Memory Requested:   576.0GB              Memory Used: 187.65GB
   Walltime requested: 00:10:00            Walltime Used: 00:00:40
   JobFS requested:    1.17GB                 JobFS used: 0B
================================================================================
```

PBS Value Initialization

Number of nodes	Number of cores used	Warmup/ Benchmark	Walltime Requested	Memory Requested
1 x 2	48 x 1	40,000/80,000	10 mins	48GB
2 x 2	48 x 2	40,000/80,000	10 mins	96GB
4 x 2	48 x 4	40,000/80,000	10 mins	192GB
8 x 2	48 x 8	40,000/80,000	10 mins	384GB
16 x 2	48 x 16	10,000/160,000	10 mins	768GB
32 x 2	48 x 32	10,000/320,000	10 mins	1536GB

Execution Script

```
module purge
module load ${HOME}/hpcx-v2.20-gcc-mlnx_ofed-redhat8-cuda12-x86_64/modulefiles/hpcx-ompi

hosts=$(sort -u ${PBS_NODEFILE} | paste -sd ',')

cmd="time mpirun \
    -host ${hosts} \
    -wdir ${HOME}/scratch/workdir/hoomd \
    -output-filename ${HOME}/run/output/${PBS_JOBNAME}.${PBS_JOBID} \
    -map-by ppr:$((1*${NCPUS})):node \
    -oversubscribe -use-hwthread-cpus \
    -x PYTHONPATH=${HOME}/scratch/workdir/hoomd/build/hoomd-openmpi4.1.5:${HOME}/scratch/workdir/hoomd/hoomd-
benchmarks \
    ${HOME}/scratch/workdir/hoomd/hoomd.py312/bin/python \
    -m hoomd_benchmarks.md_pair_wca \
    --device CPU -v \
    -N ${N} --repeat ${repeat} \
    --warmup_steps ${warmup_steps} --benchmark_steps ${benchmark_steps}"


echo ${cmd}

exec ${cmd}
~
```

hoomd.sh

Submitting Job Commands

Example:

```
(base) [sp1115@gadi-login-03 run]$ nodes=32 walltime=00:10:00 \
> warmup_steps=10000 benchmark_steps=8000 repeat=1 N=200000 \
 \
'qsub -V \
-l> bash -c \
> 'qsub -V \
> -l walltime=${walltime},ncpus=$((48*nodes)),mem=$((48*nodes*1))gb \
> -N hoomd.nodes${nodes}.WS${warmup_steps}.BS${benchmark_steps} \
> hoomd.sh'
```

Examples of Output

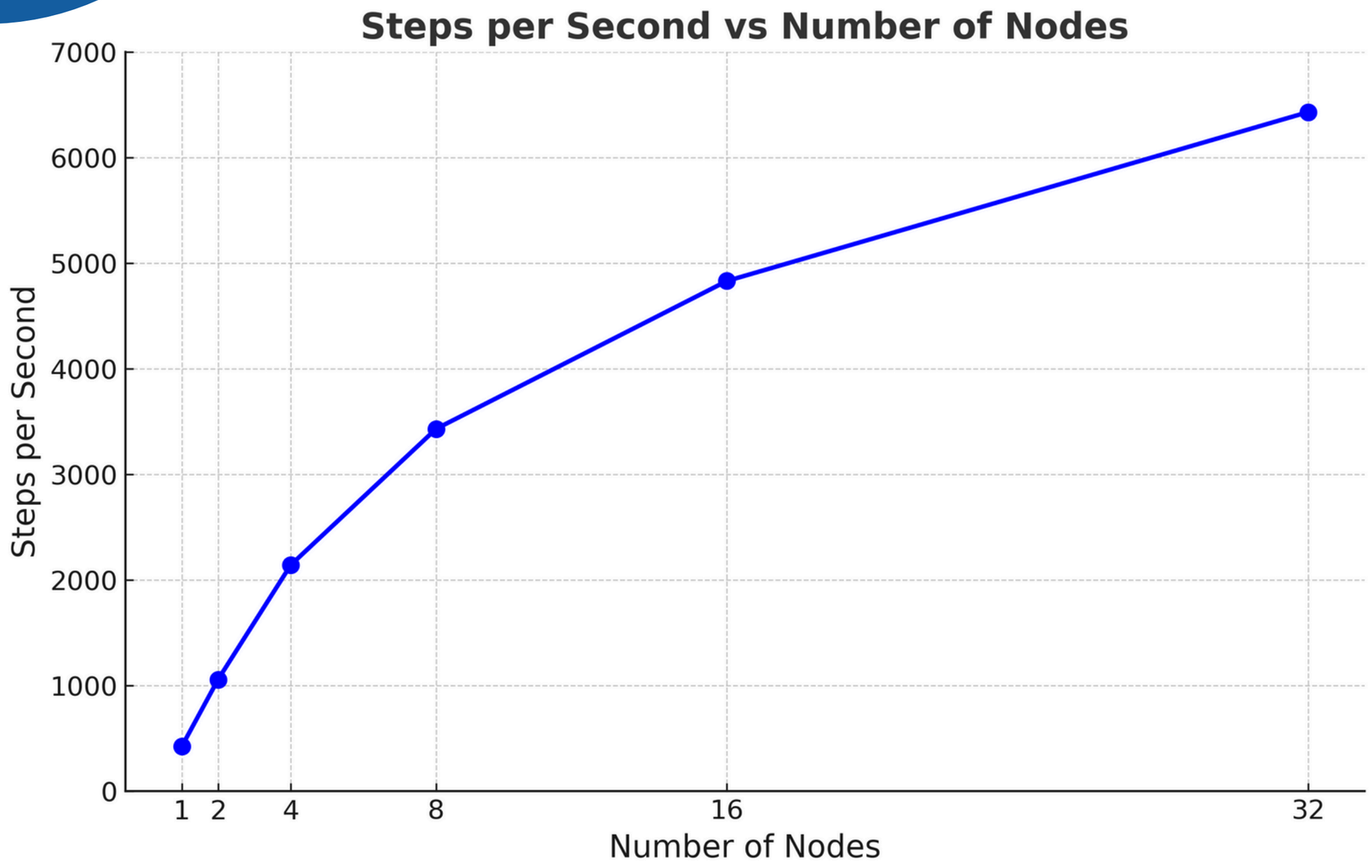
```
Using existing initial_configuration_cache/hard_sphere_200000_1.0_3.gsd
notice(2): Using domain decomposition: n_x = 4 n_y = 4 n_z = 6.
Running MDPairWCA benchmark
.. warming up for 40000 steps
.. running for 80000 steps 1 time(s)
.. 999.3391245451993 time steps per second
999.3391245451993
5598.51user 308.30system 2:08.09elapsed 4611%CPU (0avgtext+0avgdata 431800maxresident)k
225936inputs+16848outputs (28009major+12530102minor)pagefaults 0swaps

======================================================================================
                  Resource Usage on 2024-10-10 01:31:28:
   Job Id:             126506621.gadi-pbs
   Project:            zd64
   Exit Status:        0
   Service Units:      6.93
   NCPUs Requested:    96                     NCPUs Used: 96
                                           CPU Time Used: 03:16:35
   Memory Requested:   48.0GB                Memory Used: 31.53GB
   Walltime requested: 00:10:00            Walltime Used: 00:02:10
   JobFS requested:    200.0MB                JobFS used: 0B
======================================================================================
```

hoomd.sh output file

HOOMDBLUE Result

Number of nodes	Number of cores used	Total Cores	Memory requested (GB)	Walltime Used	Memory Used (GB)	Steps per second
1 x 2	48 x 1	48	48	0:55	21.54	423
2 x 2	48 x 2	96	96	0:27	31.68	1058
4 x 2	48 x 4	192	192	0:18	62.11	2142
8 x 2	48 x 8	384	384	0:15	123.58	3431
16 x 2	48 x 16	768	768	0:14	249.8	4831
32 x 2	48 x 32	1536	1536	0:14	494.79	6431

Steps per second

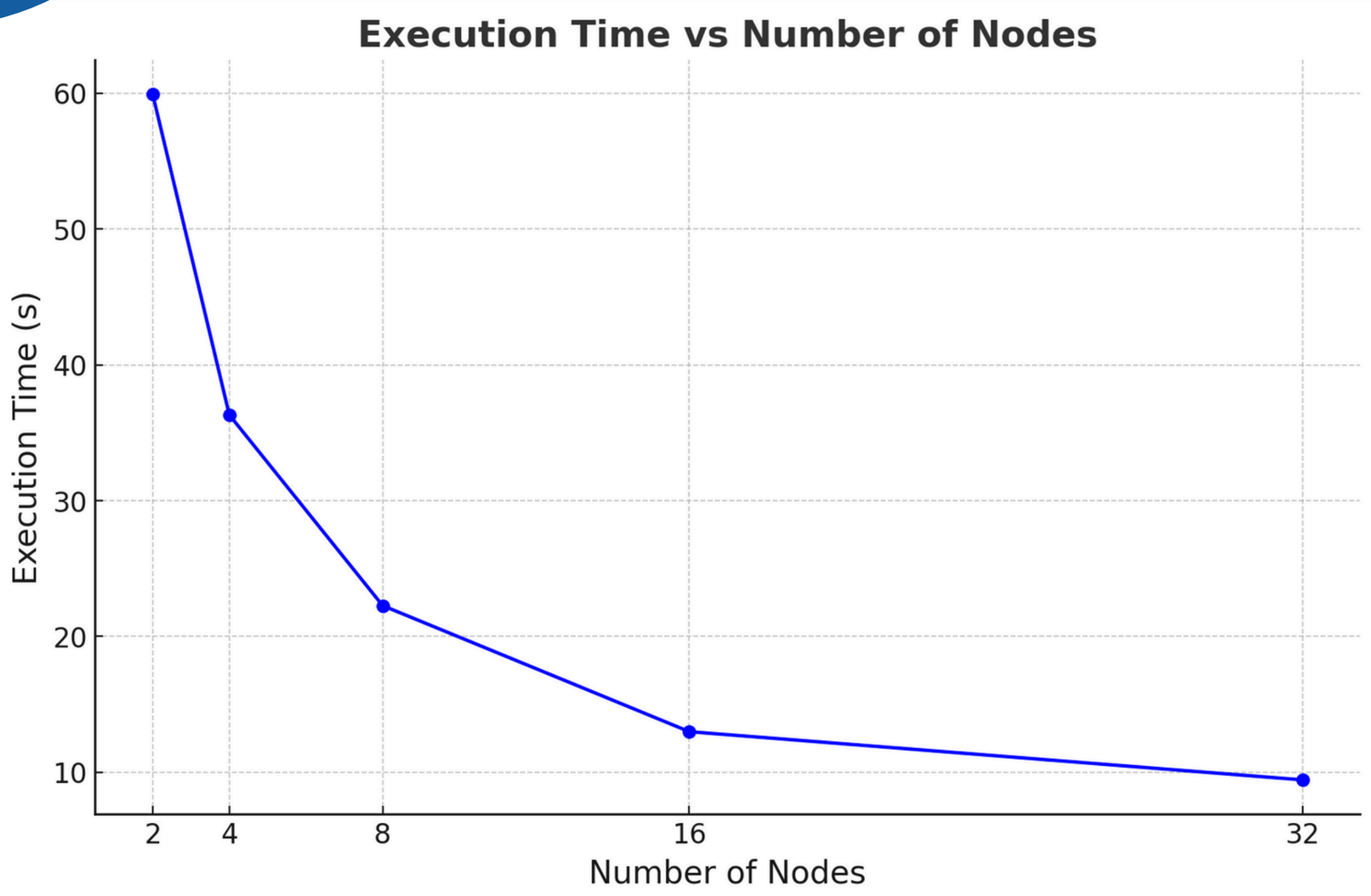


The steps per second increase as more nodes and cores are used, reflecting improved performance with additional computational resources.

However, while the performance improves significantly as nodes increase, the performance gain from adding more nodes diminishes slightly after 16 nodes.

Beyond this point, further node increases result in only marginal improvements, likely due to communication overheads and resource bottlenecks. This pattern suggests that adding nodes improves performance but with limited efficiency gains at higher node counts, which can guide optimal resource allocation.

Execution Time(s)

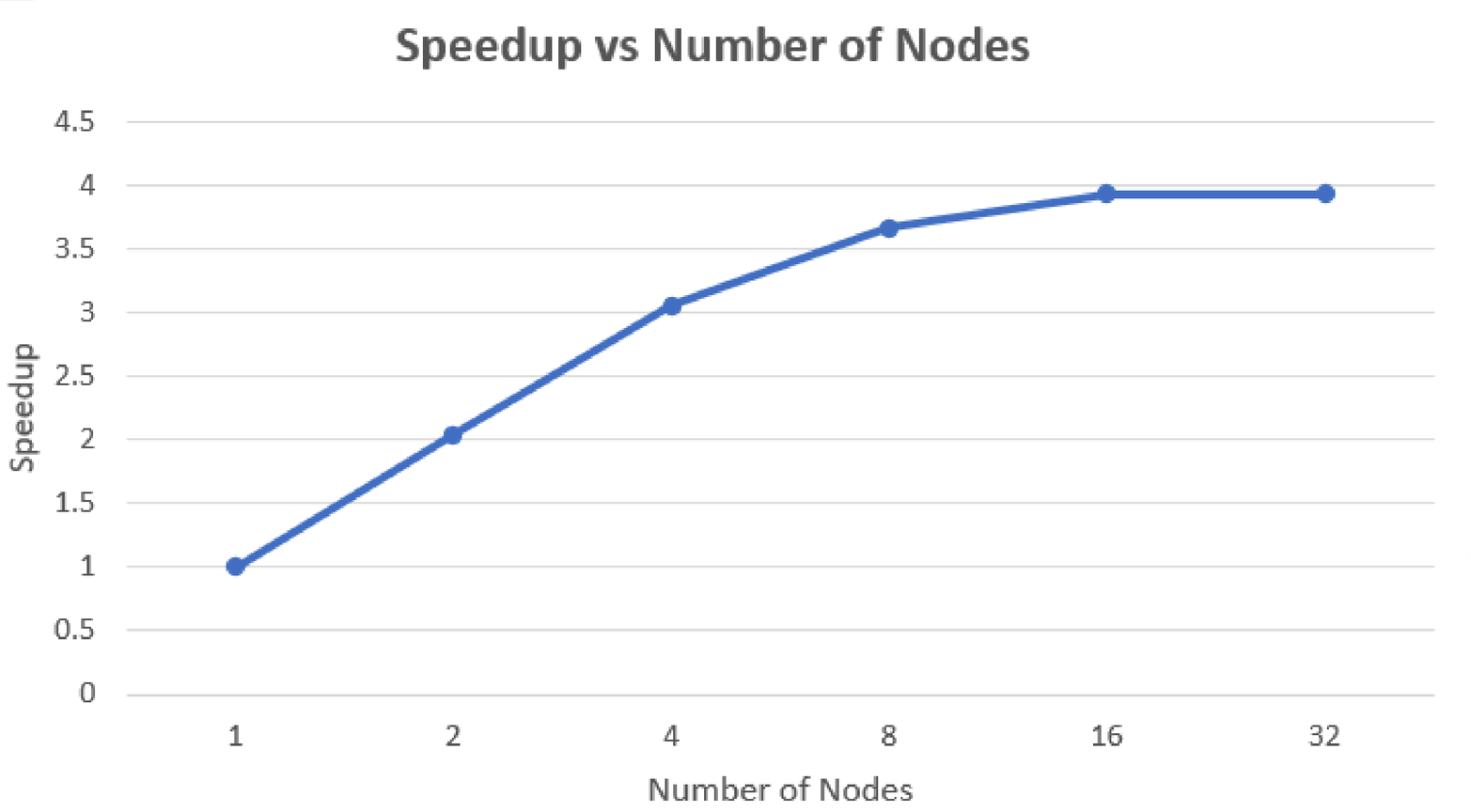


This graph shows that as the number of nodes increases, the walltime (execution time) decreases.

Adding more nodes allows the system to distribute the task across multiple nodes and process them simultaneously, which reduces the overall time required to complete the task.

However, beyond 16 nodes, we notice a slight increase in execution time, which could be due to the communication overhead between nodes, leading to delays.

Simulation speed



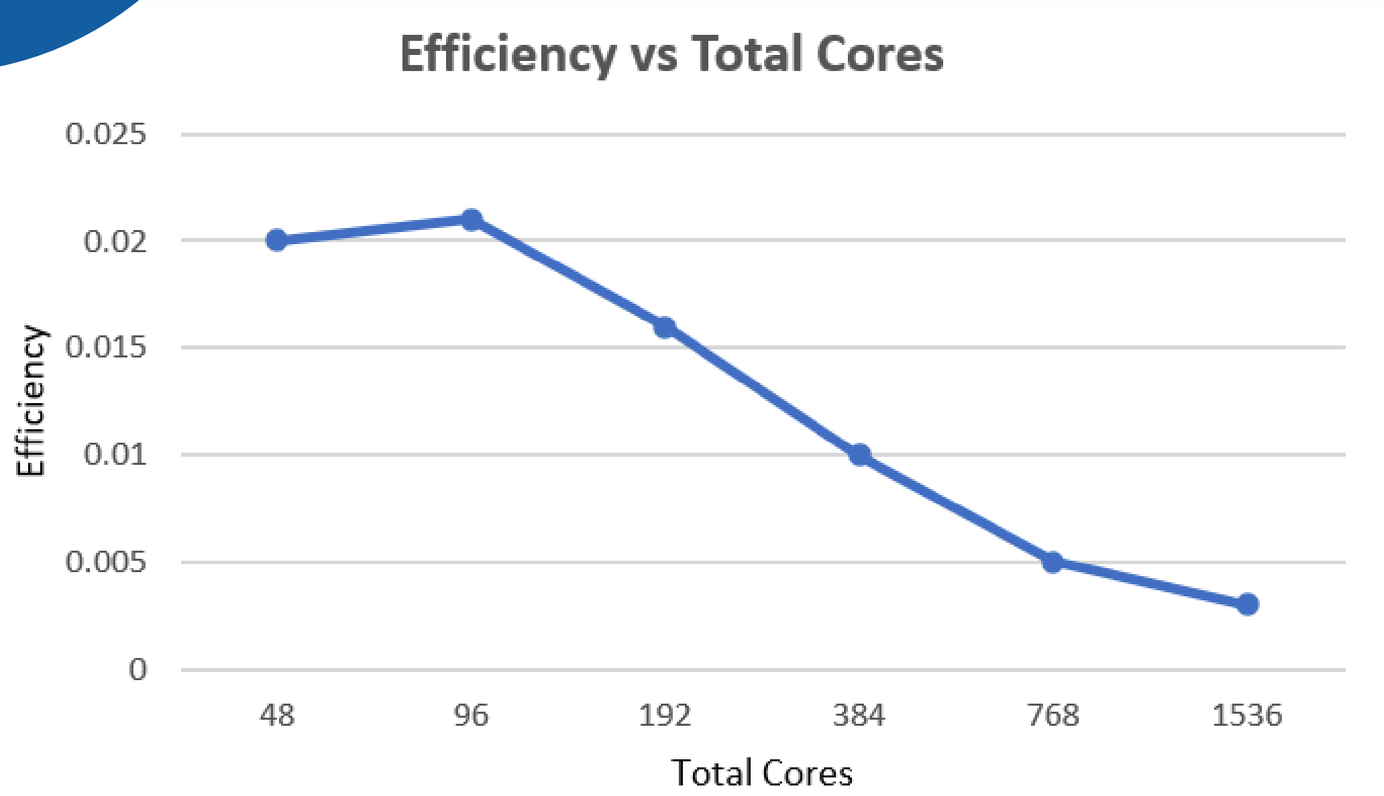
As the total cores increases, the speedup of the system tends to increase because tasks are divided and processed in parallel, reducing overall time. This indicates that adding more cores can improve performance.

However, after a certain point, adding more cores offers diminishing returns due to communication overhead.

Number of nodes	1	2	4	8	16	32
Total cores	48	96	192	384	768	1536
Speedup ***[Sequential time (s) ÷ Parallel Time (s)]***	55/55 = 1.00	55/27 = 2.04	55/18 = 3.06	55/15 = 3.67	55/14 = 3.93	55/14 = 3.93

Sequential Time : 55 seconds

Efficiency



The graph shows that as more cores are added, the system becomes less efficient. Each additional cores contributes less to overall performance due to increased coordination and communication overhead.

While more cores can speed up the system, they do not necessarily improve efficiency, as more resources are spent on managing the extra cores rather than productive work.

Number of nodes	1	2	4	8	16	32
Total cores	48	96	192	384	768	1536
Efficiency *[Speedup/ Total cores]*	1.00/48 = 0.02	2.037/96 = 0.021	3.056/192 = 0.016	3.667/384 = 0.010	3.929/768 = 0.005	3.929/1536 = 0.003

Challenges

- We aim to run simulations across multiple nodes, up to 32 in total. By scaling, the system resources like CPU, memory, and communication between nodes becomes difficult to handle.

- Initially, the performance increases with increasing cores. However, beyond a threshold, efficiency is reduced due to the dominating communication overhead between the different nodes.

Conclusion

The results obtained from simulations demonstrate that an increased number of CPUs can significantly improve system performance by reducing execution time and enhancing simulation speed, especially when tasks are divided and processed in parallel. However, there are diminishing returns beyond a certain point, as coordination and communication overhead between nodes start to limit further improvements. This highlights the importance of an optimized balance between hardware scaling and system efficiency for maximum performance in high-performance computing environments.

TASK 2
LITGPT LLAMA2 USING ASPIRE2A NSCC

Performance Metrics

01

Training time (seconds) - This measures how long it takes to complete the training.

PBS Script Directives and Modules Loaded

```
#!/bin/bash
#PBS -P 50000032
#PBS -l walltime=00:01:00
#PBS -j oe
#PBS -M 214928@student.upm.edu.my,216638@student.upm.edu.my
#PBS -m abe

date
module purge
module load openmpi/4.1.2-hpe
module load libfabric/1.11.0.4.125
```

PBS Value Initialization

Number of nodes	Number of GPUs	Number of CPUs	Number of Epochs	Global Batch Size	Micro Batch Size	Max steps
2	8	128	1	128	32	20

PBS Commands Used

- qsub tuningllama.pbs - To submit job script (tuningllama.pbs)
- qstat - To display queue information
- qstat -q - To display current queue information

```
(base) [aa5323@gadi-login-03 run]$ qsub tuningllama.pbs
126609921.gadi-pbs
```

```
(base) [aa5323@gadi-login-03 run]$ qstat
Job id                  Name             User              Time Use S Queue
----------------------  ---------------- ----------------  -------- - -----
126609921.gadi-pbs      tuningllama.pbs  aa5323                   0 Q gpuvolta-exec
```

```
(base) [aa5323@gadi-login-03 run]$ qstat -q

server: gadi-pb

Queue            Memory CPU Time Walltime Node   Run   Que   Lm  State
---------------- ------ -------- -------- ---- ----- ----- ----  -----
normal             --      --       --     --      0    56   --   E R
normal-exec        --      --       --     --   1232   357   --   E R
express            --      --       --     --      0     0   --   E R
express-exec       --      --       --     --      2     1   --   E R
copyq              --      --       --     --      0    43   --   E R
copyq-exec         --      --       --     --     12     1   --   E R
gpuvolta           --      --       --     --      0   361   --   E R
gpuvolta-exec      --      --       --     --    339   424   --   E R
hugemem-exec       --      --       --     --     85   164   --   E R
hugemem            --      --       --     --      0    44   --   E R
```

Execution Script

```
env
cat $PBS_NODEFILE

export NCCL_DEBUG=INFO
export NCCL_IB_DISABLE=1
export NCCL_NET_GDR_LEVEL=0
export NCCL_SHM_DISABLE=1

nvidia-smi

cmd="mpirun \
-wdir ${HOME}/scratch/workdir/llama \
-output-filename ${HOME}/run/output/${PBS_JOBNAME}.${PBS_JOBID} \
-map-by ppr:4:node -oversubscribe \
-report-bindings \
-x mpirun -mca btl ^ucx \
-mca coll_hcoll_enable 1 -mca coll_basic_priority 10 \
${HOME}/scratch/workdir/llama/litgpt.py312/bin/litgpt \
finetune_full \
${HOME}/scratch/workdir/llama/model/litgpt/meta-llama/Llama-2-7b-hf \
--data JSON --data.json_path ${HOME}/scratch/workdir/llama/dataset/alpaca1024 \
--out_dir ${HOME}/scratch/workdir/llama/out/finetune/full
--config ${HOME}/scratch/workdir/llama/full.yaml \
--eval.final_validation=false \
--train.epochs=1 \
--devices=4 --num_nodes=2 \
--train.max_steps=${max_steps} \
```

tuningllama.pbs

- **NVIDIA-SMI**
  Track gpu's utilization and performance

- **-x mpirun -mca btl ^ucx**

  OpenMPI for communication backend, using Byte Transfer Layer (BTL) and exclude UCX

- **-mca coll_hcoll_enable 1 -mca coll_basic_priority 10**
  Set up collective algorithms such as enabling HCOLL and prioritize basic collective operations

Submitting Job Commands

Example:

```
[apacsc32@asp2a-login-nscc02 run]$ nodes=2 walltime=7201 \
> global_batch_size=128 micro_batch_size=32 max_steps=20 \
> bash -c \
> 'qsub -V \
> -l walltime=${walltime},select=${nodes}:ngpus=4 \
> -N llama.nodes${nodes}.GBS${global_batch_size}.MBS${micro_batch_size} \
> llama.sh'
8616103.pbs101
```

Example of Output

```
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 535.154.05             Driver Version: 535.154.05   CUDA Version: 12.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |         Memory-Usage | GPU-Util  Compute M. |
|                                         |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA A100-SXM4-40GB           On | 00000000:03:00.0 Off |                    0 |
| N/A   43C    P0              52W / 400W |      0MiB / 40960MiB |      0%      Default |
|                                         |                      |             Disabled |
+-------------------------------+----------------------+----------------------+
|   1  NVIDIA A100-SXM4-40GB           On | 00000000:41:00.0 Off |                    0 |
| N/A   42C    P0              51W / 400W |      0MiB / 40960MiB |      0%      Default |
|                                         |                      |             Disabled |
+-------------------------------+----------------------+----------------------+
|   2  NVIDIA A100-SXM4-40GB           On | 00000000:81:00.0 Off |                    0 |
| N/A   43C    P0              51W / 400W |      0MiB / 40960MiB |      0%      Default |
|                                         |                      |             Disabled |
+-------------------------------+----------------------+----------------------+
|   3  NVIDIA A100-SXM4-40GB           On | 00000000:C1:00.0 Off |                    0 |
| N/A   43C    P0              54W / 400W |      0MiB / 40960MiB |      0%      Default |
|                                         |                      |             Disabled |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

llama.sh --> nvidia-smi output

Example of Output

```
The longest sequence length in the train data is 512, the model's maximum sequence length is 512 and context length is 4096
Verifying settings ...
Epoch 1 | iter 1 step 1 | loss train: 1.417, val: n/a | iter time: 5379.64 ms (step)
Epoch 1 | iter 1 step 1 | loss train: 1.428, val: n/a | iter time: 5390.40 ms (step)
Epoch 1 | iter 2 step 2 | loss train: 1.627, val: n/a | iter time: 5081.73 ms (step)
Epoch 1 | iter 2 step 2 | loss train: 1.435, val: n/a | iter time: 5086.31 ms (step)
Epoch 1 | iter 3 step 3 | loss train: 1.331, val: n/a | iter time: 5092.07 ms (step)
Epoch 1 | iter 3 step 3 | loss train: 1.288, val: n/a | iter time: 5086.58 ms (step)
Epoch 1 | iter 4 step 4 | loss train: 1.010, val: n/a | iter time: 5083.41 ms (step)
Epoch 1 | iter 4 step 4 | loss train: 0.962, val: n/a | iter time: 5085.00 ms (step)
Epoch 2 | iter 5 step 5 | loss train: 0.880, val: n/a | iter time: 5258.94 ms (step)
Epoch 2 | iter 5 step 5 | loss train: 0.834, val: n/a | iter time: 5258.90 ms (step)
Training time: 28.07s
Memory used: 17.73 GB
Training time: 28.10s
Memory used: 17.73 GB
x1000c3s4b0n0:2898258:2898878 [3] NCCL INFO [Service thread] Connection closed by localRank 3
x1000c3s4b0n0:2898256:2898876 [1] NCCL INFO [Service thread] Connection closed by localRank 1
x1000c3s3b0n0:828731:829309 [3] NCCL INFO [Service thread] Connection closed by localRank 3
x1000c3s4b0n0:2898257:2898875 [2] NCCL INFO [Service thread] Connection closed by localRank 2
x1000c3s4b0n0:2898255:2898877 [0] NCCL INFO [Service thread] Connection closed by localRank 0
x1000c3s3b0n0:828729:829308 [1] NCCL INFO [Service thread] Connection closed by localRank 1
x1000c3s3b0n0:828730:829307 [2] NCCL INFO [Service thread] Connection closed by localRank 2
x1000c3s3b0n0:828728:829311 [0] NCCL INFO [Service thread] Connection closed by localRank 0
x1000c3s3b0n0:828731:832787 [3] NCCL INFO comm 0x14117290 rank 3 nranks 8 cudaDev 3 busId 81000 - Abort COMPLETE
x1000c3s4b0n0:2898258:2902251 [3] NCCL INFO comm 0x14a1e7b0 rank 7 nranks 8 cudaDev 3 busId 81000 - Abort COMPLETE
x1000c3s3b0n0:828730:832789 [2] NCCL INFO comm 0x1558c8c0 rank 2 nranks 8 cudaDev 2 busId 3000 - Abort COMPLETE
x1000c3s3b0n0:828729:832788 [1] NCCL INFO comm 0x1413d660 rank 1 nranks 8 cudaDev 1 busId c1000 - Abort COMPLETE
x1000c3s4b0n0:2898256:2902252 [1] NCCL INFO comm 0x152e6ff0 rank 5 nranks 8 cudaDev 1 busId c1000 - Abort COMPLETE
x1000c3s4b0n0:2898257:2902254 [2] NCCL INFO comm 0x143701b0 rank 6 nranks 8 cudaDev 2 busId 3000 - Abort COMPLETE
x1000c3s4b0n0:2898255:2902253 [0] NCCL INFO comm 0x14f65fa0 rank 4 nranks 8 cudaDev 0 busId 41000 - Abort COMPLETE
x1000c3s3b0n0:828728:832790 [0] NCCL INFO comm 0x161490e0 rank 0 nranks 8 cudaDev 0 busId 41000 - Abort COMPLETE
```

llama.sh output file

LLAMA2 Results

Baseline

Number of nodes	Number of CPUs	Number of GPUs	Memory Used	Average Training Time (s)
2	128	8	17.73GB	41.57

LLAMA2 Results

Improved script

```
cmd="mpirun \
-wdir ${HOME}/scratch/workdir/llama \
-output-filename ${HOME}/run/output/${PBS_JOBNAME}.${PBS_JOBID} \
-map-by ppr:4:node -oversubscribe \
-report-bindings \
-x mpirun -mca btl ^ucx \
-mca coll_hcoll_enable 1 -mca coll_basic_priority 10 \
```

Number of nodes	Number of CPUs	Number of GPUs	Memory Used	Average Training Time (s)
2	128	8	17.73GB	28.09

LLAMA2 Results
Improved script

```
export NCCL_DEBUG=INFO
export NCCL_IB_DISABLE=1
export NCCL_NET_GDR_LEVEL=0
export NCCL_SHM_DISABLE=1
```

- Enables detailed NCCL logging
- Disables InfiniBand, uses TCP/IP
- Disables GPU Direct RDMA, uses CPU/system memory.

Number of nodes	Number of CPUs	Number of GPUs	Memory Used	Average Training Time (s)
2	128	8	17.73GB	29.04

LLAMA2 Results
Fine tuning using LoRA

LoRA (Low-Rank Adaptation) is a technique that adapts pre-trained neural networks by introducing a small set of trainable low-rank matrices into the network. This approach reduces the number of parameters that need to be fine-tuned, making the adaptation process more efficient in terms of both computation and memory.

Observation:

- Reduced training time
- Lower memory usage
- Maintained performance
- Flexibility and scalability

Challenges

- In Gadi, we ran into some issues when trying to execute the job, which stopped us from getting any results from Gadi. It might be due to our building environment process. The disks ran out of memory while we were trying to install modules, thus we were not able to run the job successfully.

- In Aspire2A, achieving efficient resource utilization requires precise parameter adjustments (like batch size and learning rate) and an great understanding of MPI communication, which we are learning to navigate as we work on this project. As for the fine tuning using LoRA, we were not able to set it up on time due to our limited knowledge.

Conclusion

This task that demanded in-depth understanding of HPC environments, parallel processing, and distributed machine learning that involves multi-faceted approach such as algorithmic advancements. Fortunately, this project successfully implemented optimizations that slightly reduced training time, such as adjustments to the communication backend.

THANK YOU!

UPM Team 3
Universiti Putra Malaysia