AI Taiwan VVM

Team 07

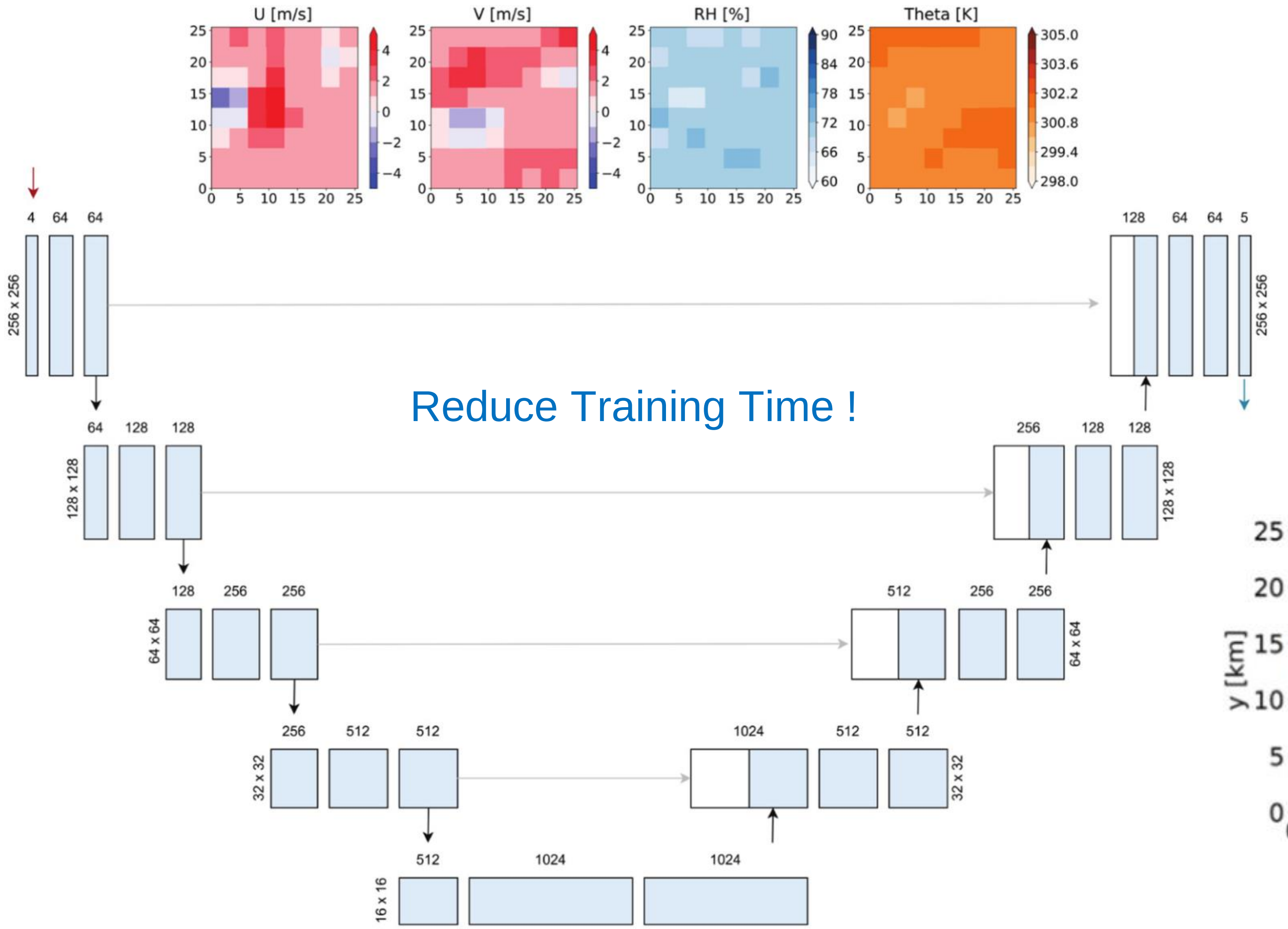
2025.11.12

OpenACC
More Science, Less Programming

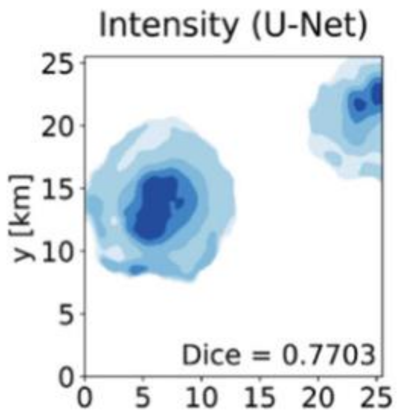
Team AI Taiwan VVM

Team Members are from Lab of Cloud Dynamic Modeling (LCDM), Department of Atmospheric Sciences, NTU
Mentors are from NVIDIA

Application: Accelerating a U-Net Model



Reduce Training Time !



LCDM Lab. for Cloud Dynamics and Modeling

OpenACC More Science, Less Programming · OPEN HACKATHONS · NCHC 國家實驗研究院 國家高速網路與計算中心 National Center for High-performance Computing · NVIDIA

Application: Accelerating a U-Net Model (Cont.)

- The current U-Net model is limited to training with a small dataset (~6.2 GB) due to slow training speed (2 hrs / 100 epochs, 35 samples/sec.)
- This limitation reduces the model's robustness and generalization.
- We have access to ~1 TB of high-resolution simulation data over Taiwan's complex terrain.
- Our goal in this hackathon is to **accelerate the U-Net training process** to better leverage the large dataset.
- By enhancing training efficiency, we seek to improve model robustness and realism for Taiwan's complex terrain

Goals

- Algorithm Acceleration:
  - Distributed Data Parallelism (DDP) with multiple GPUs
  - Apply any possible techniques of acceleration
- Hardware Acceleration
  - P100 → A100

Evolution and Strategy

- Since we have p100 x 3 and not optimized by DDP, we planned to accelerate our training time to 3x
- Our initial strategy is modifying our code with the help from mentor to accelerate, and do not change the result of training .
- Based on the advice from mentors, some optimization beyond hardware acceleration (AMP, torch compile, TF32) can be done by changing P100 to A100

What problems have you encountered?

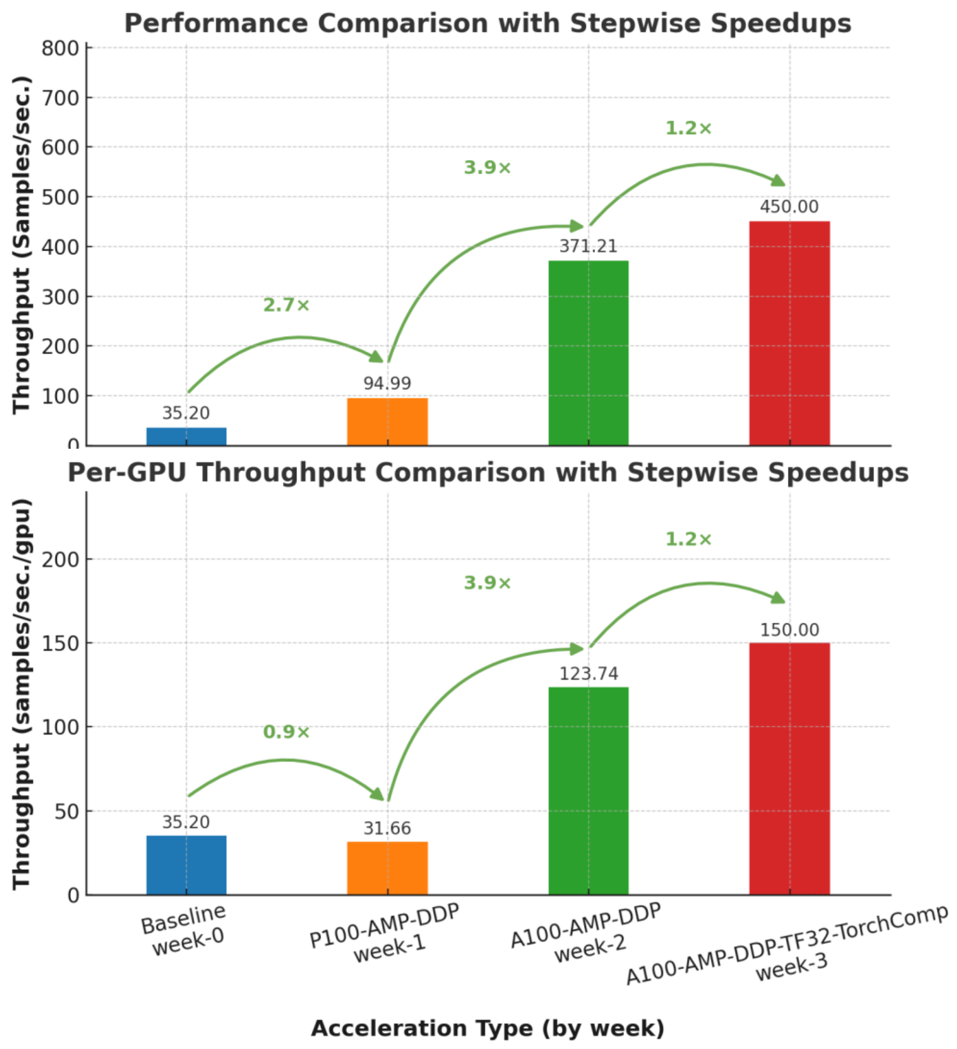
- Initial DDP setup on 3x P100 GPUs: We faced initial challenges in correctly configuring DDP.
- AMP Showed Limited Impact on P100: We observed that enabling AMP did not provide a significant performance boost on the P100 GPUs.
- Profiling with Nsight Systems: While Nsight Systems is a powerful tool, interpreting the detailed profiling data was a learning curve.
- Environment and Dependency Management: Setting up a consistent environment with the correct versions of PyTorch, CUDA for distributed training proved to be time-consuming.

Results and Final Profile

- Under the guidance of our mentors, we implemented various optimization techniques, including DDP, AMP, torch.compile, and other configuration adjustments that impact performance.

- **Achieved a 2.69x speedup** on our original P100 machine.

- **Achieved a 12.78x speedup (4.26x per-GPU speedup)** on the A100 GPUs
  - ~2.69x with DDP
  - ~2.95x with P100 -> A100
  - ~1.33x with AMP
  - ~1.01x with TF32
  - ~1.20x with torch.compile



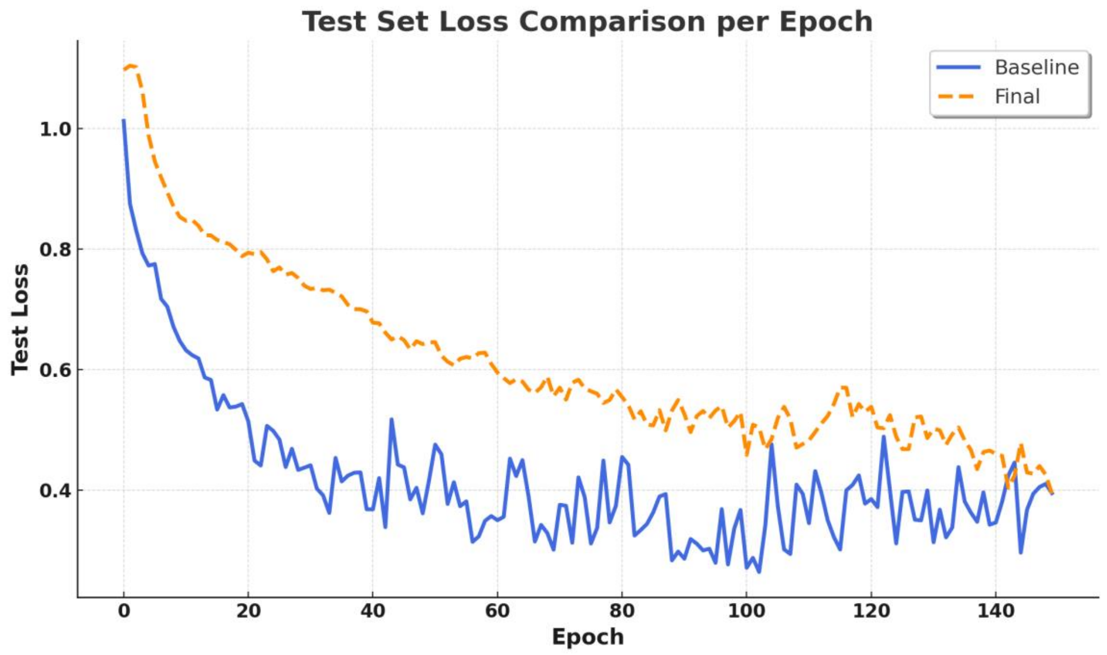
OpenACC OPEN HACKATHONS NCHC 國家實驗研究院 國家高速網路與計算中心 National Center for High-performance Computing NVIDIA

Results and Final Profile (Cont.)

Preserved Model Convergence and Quality

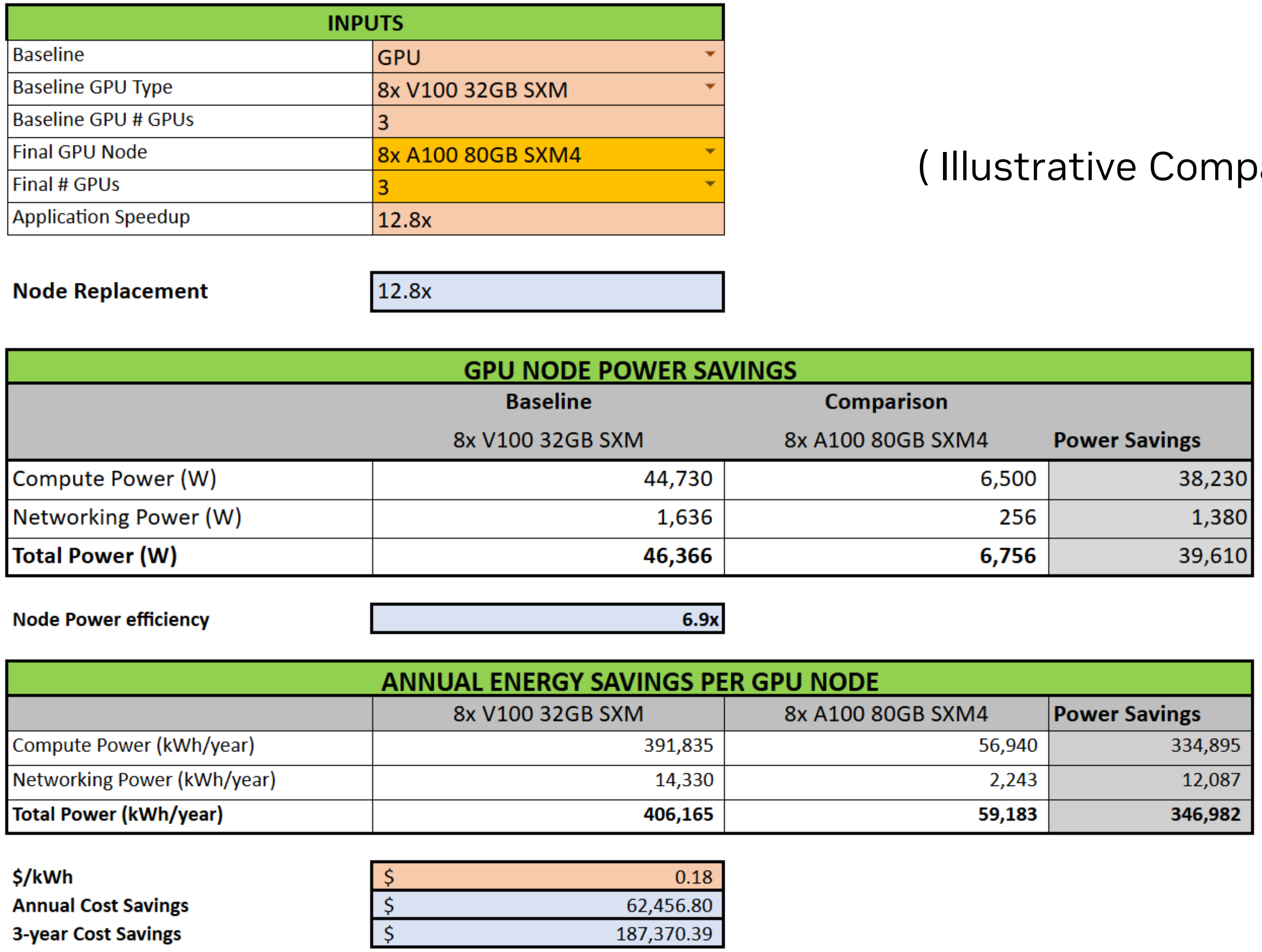
The validation loss demonstrates that our accelerated model achieves the same desired convergence as the baseline.

Both training pipelines reach the target in 150 epochs, confirming that our 12.78x speedup is achieved without any sacrifice in model accuracy or quality.



Energy Efficiency

(Illustrative Comparison against V100 Baseline)



INPUTS	
Baseline	GPU
Baseline GPU Type	8x V100 32GB SXM
Baseline GPU # GPUs	3
Final GPU Node	8x A100 80GB SXM4
Final # GPUs	3
Application Speedup	12.8x

Node Replacement	12.8x

GPU NODE POWER SAVINGS			
	Baseline	**Comparison**	
	8x V100 32GB SXM	8x A100 80GB SXM4	**Power Savings**
Compute Power (W)	44,730	6,500	38,230
Networking Power (W)	1,636	256	1,380
Total Power (W)	**46,366**	**6,756**	39,610

Node Power efficiency	**6.9x**

ANNUAL ENERGY SAVINGS PER GPU NODE			
	8x V100 32GB SXM	8x A100 80GB SXM4	**Power Savings**
Compute Power (kWh/year)	391,835	56,940	334,895
Networking Power (kWh/year)	14,330	2,243	12,087
Total Power (kWh/year)	**406,165**	**59,183**	**346,982**

$/kWh	$	0.18
Annual Cost Savings	$	62,456.80
3-year Cost Savings	$	187,370.39

Wishlist

- Smarter Profiling Tools: A profiler highlights bottlenecks and may automatically suggests specific code optimizations or library replacements

- GPU Direct Storage (GDS): This would enable our GPUs to directly stream data from storage, bypassing the CPU and system memory. This would reduce I/O latency and maximize GPU utilization, which is critical for efficiently training on TB-scale datasets.

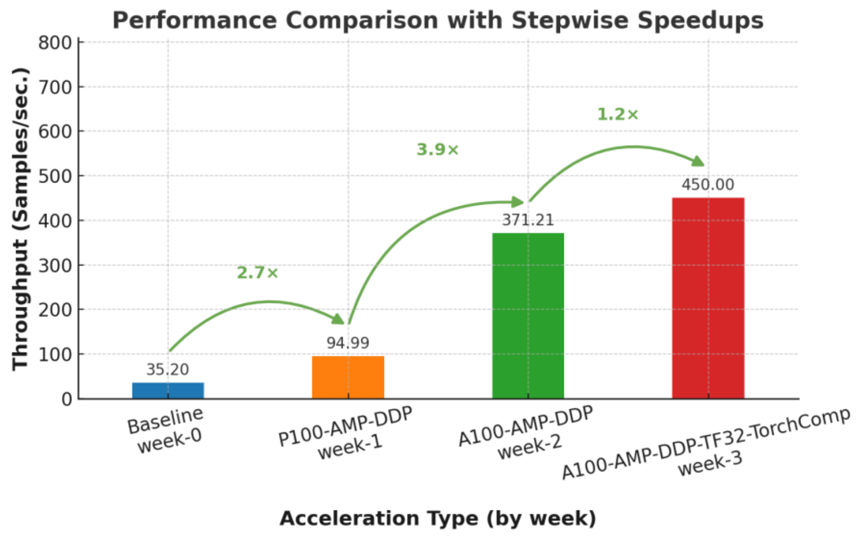
- Continued Collaboration with Mentors: We plan to maintain contact with our mentors for their expert guidance as we address the next critical bottleneck—optimizing the data loading pipeline.

Final Thoughts

- With the nearly 13x speedup we've achieved, we can now confidently train our high-resolution dataset for Taiwan's complex terrain.
- Need to optimize our data loading: Our acceleration tests used a small dataset. We anticipate that when we switch to the dataset of Taiwan's complex terrain, optimizing this data loading process will be of critical importance.
- The hackathon provided a clear goal, GPU resources, and enthusiastic guidance from experienced mentors. Having access to these resources and support greatly advanced our project, making this a productive event!

Application Background

Our project use a U-Net deep learning model to analyze and predict high-resolution meteorological phenomena from low-resolution simulation data. The primary application is to improve weather forecasting realism over Taiwan's complex terrain. However, the model's effectiveness is limited by extremely slow training speeds, which prevents us from using our complete 1-TB dataset. This limitation reduce the model's robustness and generalization capabilities.



Hackathon Objectives and Approach

Accelerate the U-Net model's training pipeline to make training on large dataset feasible. This may involve a multi-pronged strategy in the hackathon:

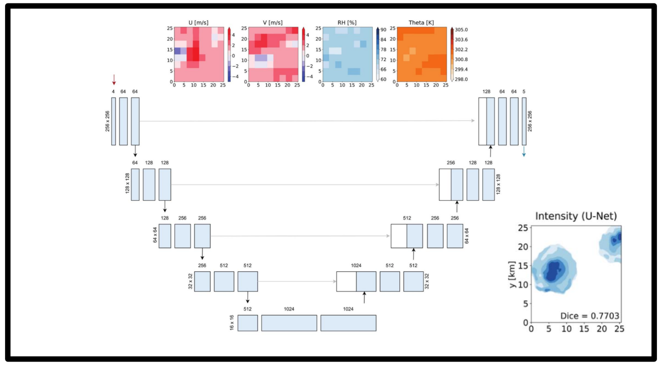
1. Implement Distributed Data Parallelism (DDP) to train across multiple GPUs, Automatic Mixed Precision (AMP) with TF32 training, and torch.compile to optimize CUDA kernel execution.
2. Used NVIDIA Nsight Systems to identify computational and data movement bottlenecks.
3. Migrated from NVIDIA P100 to A100 GPUs

Technical Accomplishments and Impact

- Achieved a **12.78x training speedup** compared to our original baseline.
- Stack of optimizations: multi-GPU scaling with DDP, enabling mixed-precision with AMP, JIT-compiling the model with torch.compile, and upgrading our hardware to A100 GPUs.
- This huge speedup makes it practical to use our 1-TB Taiwan's complex terrain dataset, enabling us to train a more accurate, robust, and realistic weather model for Taiwan.

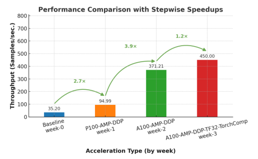
A storyline for publication on NCHC's website.

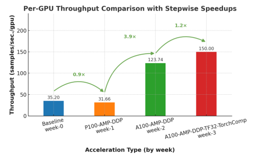
U-Net 高解析度對流預測模型



AI Taiwan VVM 團隊來自臺灣大學大氣科學系，吳健銘老師帶領的雲動力模擬實驗室，將 U-Net 高解析度對流預測模型的訓練流程，加速了近 13 倍！！

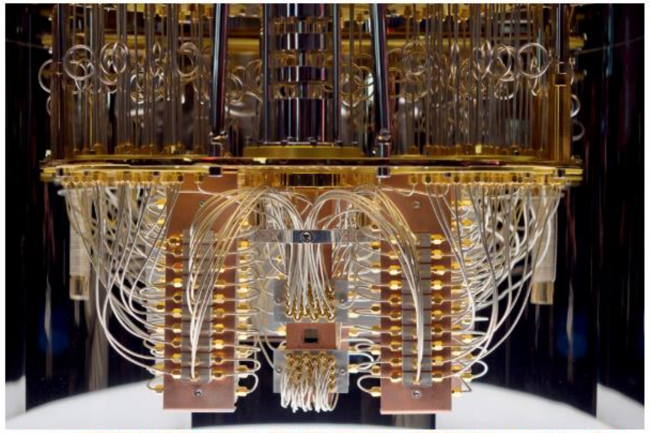
對於地形複雜、天氣多變的臺灣，高解析度的氣象預測模型非常重要，可以說是防災決策的關鍵。近年來，AI 的預測模型為高解析度的預報準確度帶來了新的切入點，但其巨大的潛力卻受限於漫長的模型訓練時間。本團隊原有的預測模型，在面對龐大的高解析度氣象模擬資料（~1 TB）時，訓練速度極其緩慢，使得無法涵蓋各種不同情境下的模擬資料來訓練模型，也限制了模型的穩健性及預報能力。
在本次的 Hackathon 中，團隊鎖定此一瓶頸，透過 NVIDIA A100 GPU 的強大算力，並結合分散式資料平行（DDP）、自動混合精度（AMP）及 torch.compile 等多項先進優化技術，成功將 U-Net 模型的訓練速度提升了 12.78 倍。這項突破性的加速成果，將原本需要數週的訓練時程大幅縮短至幾日內，使得團隊得以利用 TB 級完整的高解析度資料集來進行模型訓練。這不僅能顯著提升臺灣複雜地形天氣預報的準確度，更為發展下一代臺灣氣南 AI 模型提供一個紮實科學基礎。





報告投影片連結 (由國網上傳到 github)

量子算法模擬



haofan2023團隊成員來自臺灣大學資工系「洪士灝老師實驗室」，將量子演算法QAOA加速468倍！
— NVIDIA Mentors: Tian Zheng, Frank Lin, Yun-Yuan Wang

量子技術正以驚人的速度發展，預示著我們即將進入量子計算的時代。在這個過程中，量子電路模擬成為一個關鍵工具，它在量子硬體和軟體的開發中扮演著重要的角色，特別是在處理量子程式的編寫和驗證方面。傳統電腦的強模擬能夠獲得完整的量子狀態信息。這使得傳統電腦在構建量子系統方面變得不可或缺，尤其是在當前噪聲較多的中等規模量子（NISQ）時代。

量子近似優化算法（QAOA）是一種常用的量子算法，用於通過近似解來解決組合優化問題。然而，在虛擬量子計算機上執行QAOA對於解決需要大規模量子電路模擬的組合優化問題而言，會遇到模擬速度較慢的問題。團隊使用數學優化來壓縮量子操作，並結合有效的位元操作進一步降低計算複雜性，透過GPU加速最高獲取468倍的加速效果！

Table 1: The elapsed time of 5-level QAOA (unit: second, double).

Qubit	CPU_{Single}	$CPU_{Mutiple}$	CPU_{Cache}	GPU_{Cache}	GPU_{All}
23	29.80	1.28 (23x)	1.28 (63x)	0.24 (120x)	0.06 (**341x**)
24	68.00	3.46 (20x)	3.46 (43x)	0.55 (123x)	0.12 (**382x**)
25	152.52	15.32 (10x)	15.31 (45x)	1.19 (127x)	0.23 (**404x**)
26	330.69	33.83 (10x)	33.83 (56x)	2.60 (126x)	0.56 (**417x**)
27	712.26	72.66 (10x)	72.66 (54x)	5.59 (127x)	1.08 (**427x**)
28	1556.87	156.52 (10x)	156.52 (54x)	11.96 (130x)	2.17 (**445x**)
29	3325.55	335.09 (10x)	335.09 (49x)	25.73 (129x)	4.45 (**451x**)
30	7226.46	718.33 (10x)	718.33 (47x)	55.20 (130x)	9.22 (**468x**)

更多資訊請看：https://github.com/nqobu/nvidia/raw/main/20231207/Team02.pdf

Thank You

OpenACC
More Science, Less Programming