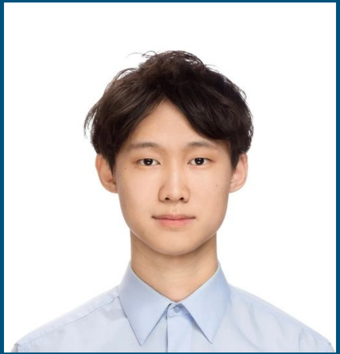
Team 2 – Usagi

Team Members

Organization of all team members: National Taiwan University

Tso-Fei Yen



Jui-Chien Tsou

Wei-Chin Wang

Kuan-Hsun Tu

Hsuan-Chi Liu

Chia-Yi Chin

Hsin-Lu Yeh

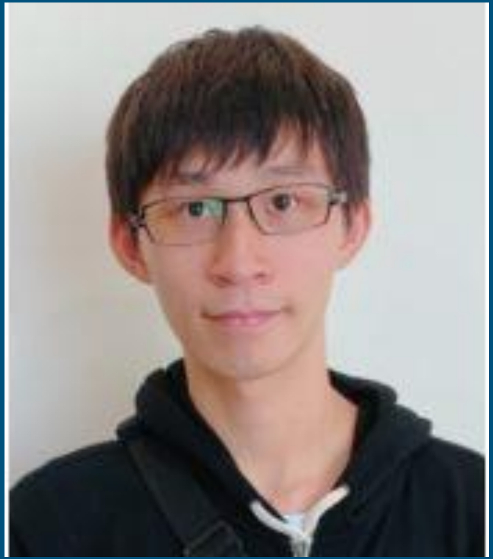
Fan-Shi Liu

Jui-En Lee

Prof. Chun-Yi Lee

Mentors

Reese Wang — NVidia

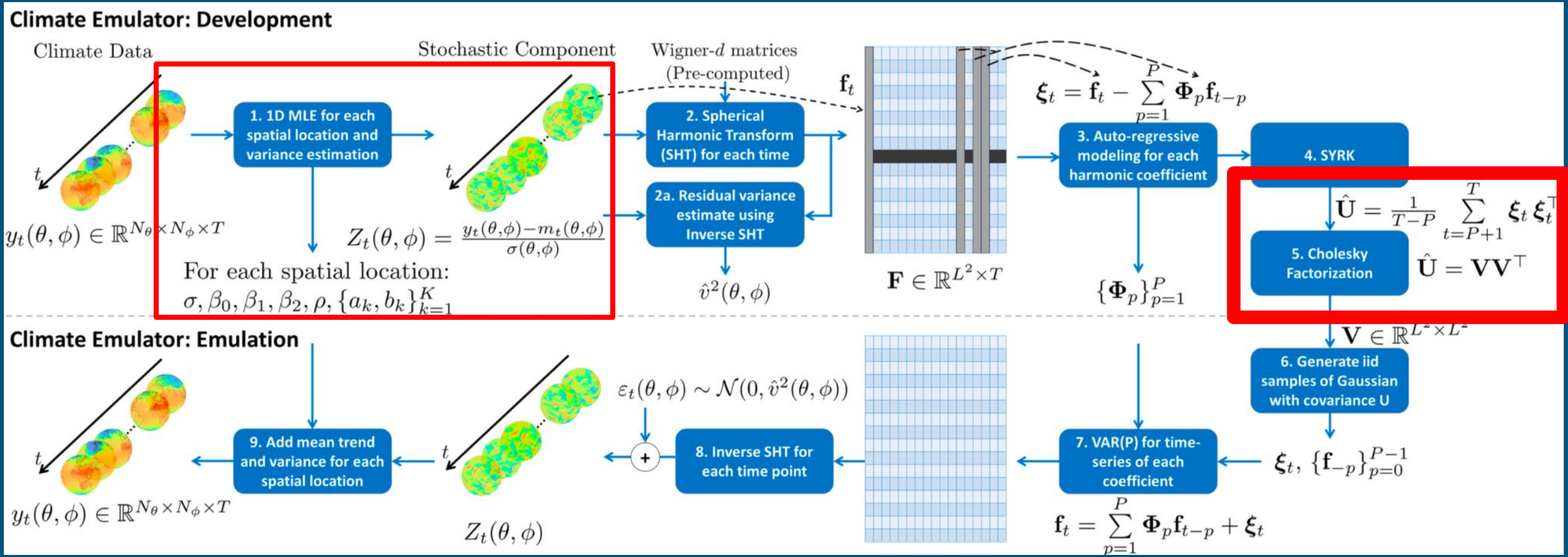


Johnson Sun — NVidia

Exascale Climate Emulator

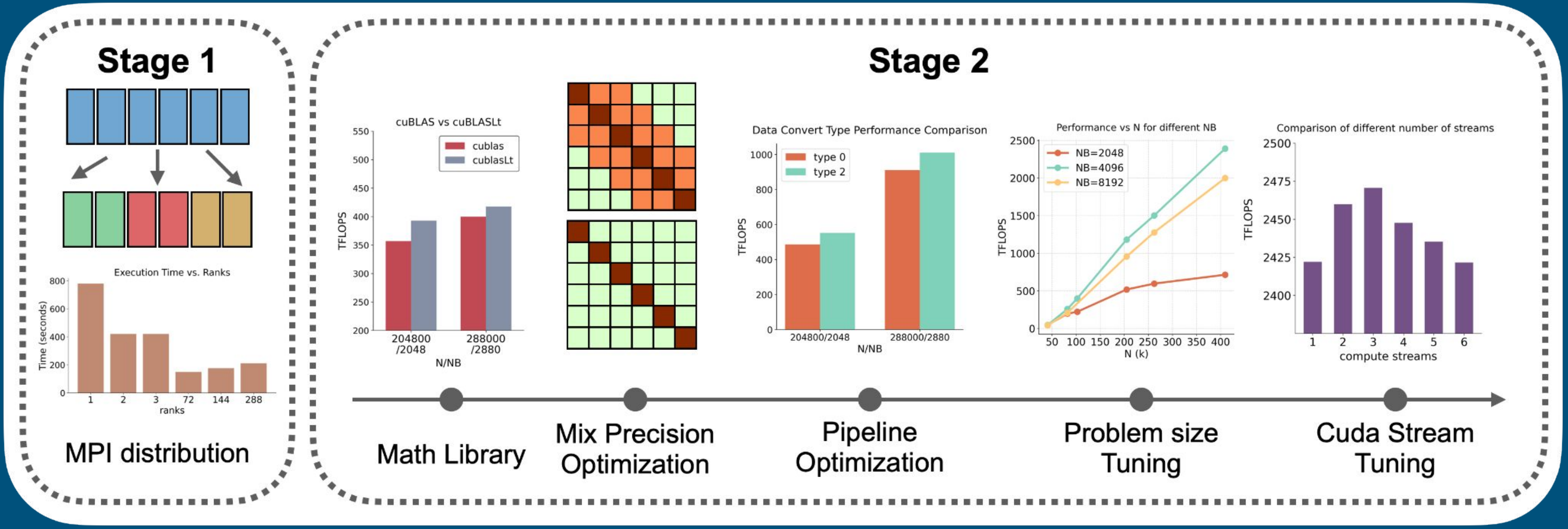
- Abdulah et al. (2024) introduced the **Exascale Climate Emulator (ECE)**, leveraging **SHT**(Spherical Harmonic Transform) and **Cholesky factorization** to boost emulating resolution and throughput.
- We build on this foundation to optimize and scale the climate emulator on our GPU cluster to improve its performance

Exascale Climate Emulator—Framework



Source: Abdulah, S., et al

Optimization Strategy



Result: Achieve 777.7x Speedup and 119.8x Energy Efficient

Stage 1:
Data preprocessing

Strategy – MPI in Data Pre-processing

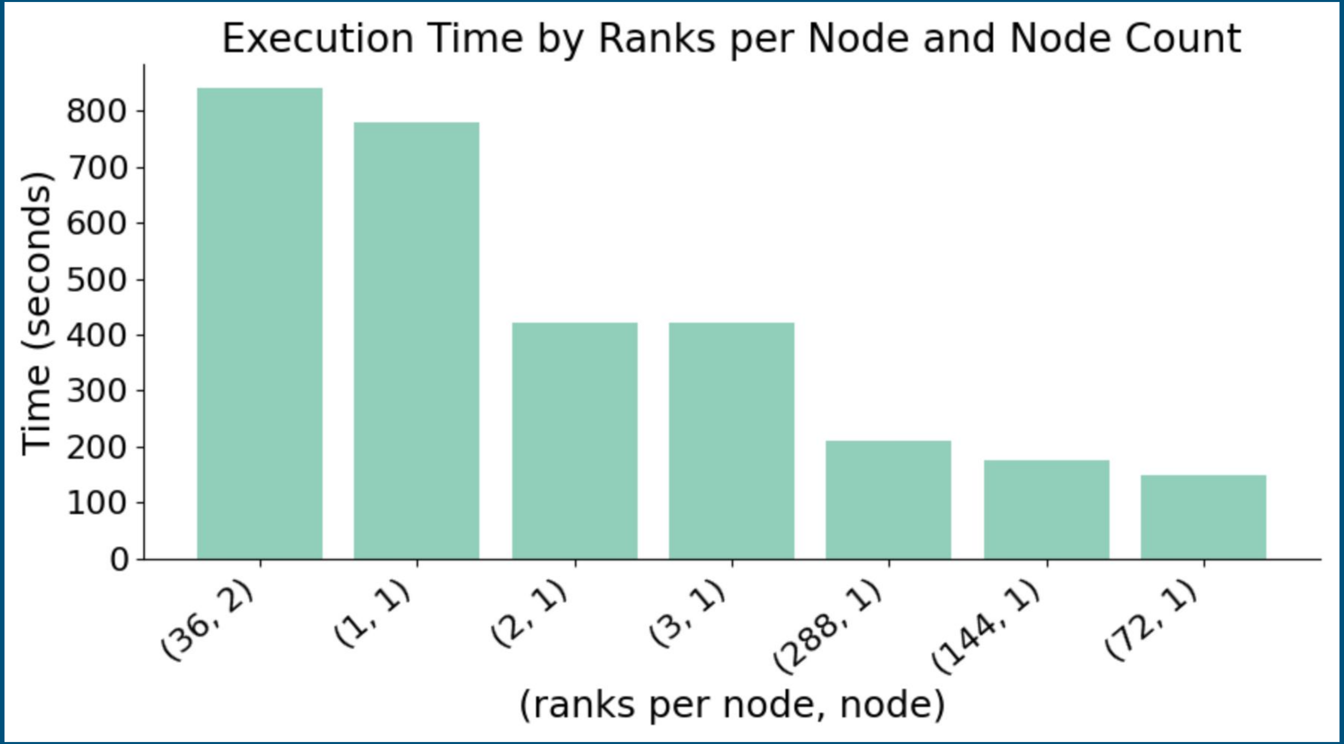
Steps

- Read Forcing Data
- Run NetCDF Files
- Run Mean Trend (Longest time!)

Method

- Add MPI, process each location independently → communication overhead
- CHAMELEON → BLAS/LAPACKE, 7x speedup!

Result – MPI in Data Pre-processing



Stage 2:
Cholesky Decomposition

CPU Baseline

We first test out application a CPU node using the following configuration:

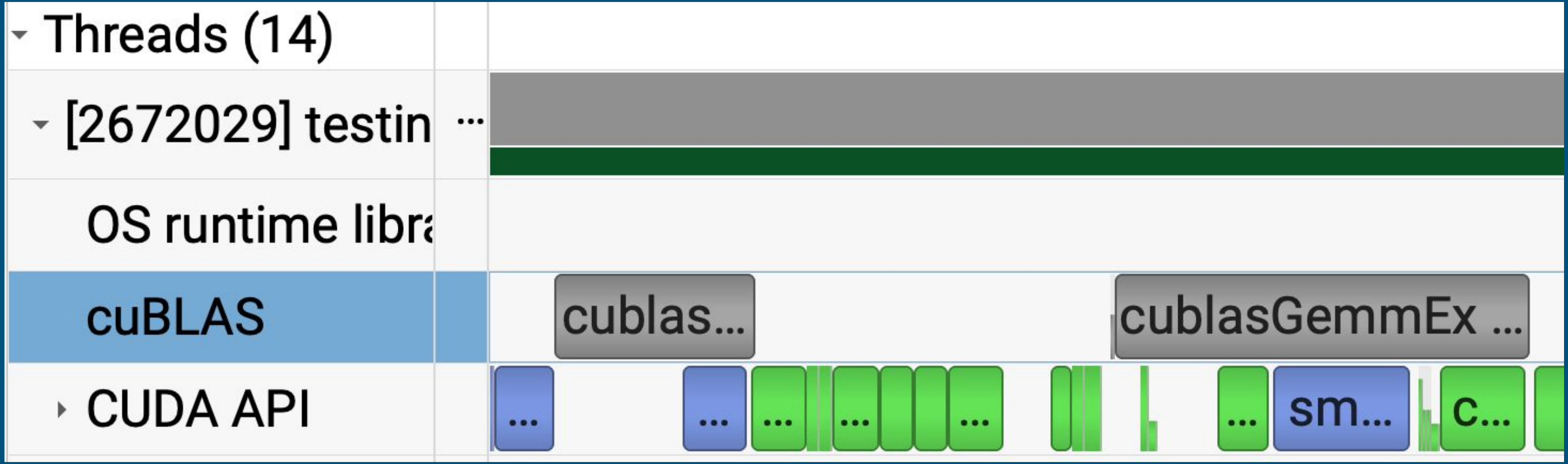
Setup: 112 CPU cores. 28 MPI ranks × 4 threads

Goal: establish a reference point before GPU/MP optimizations.

Result: 3264 GFLOPS

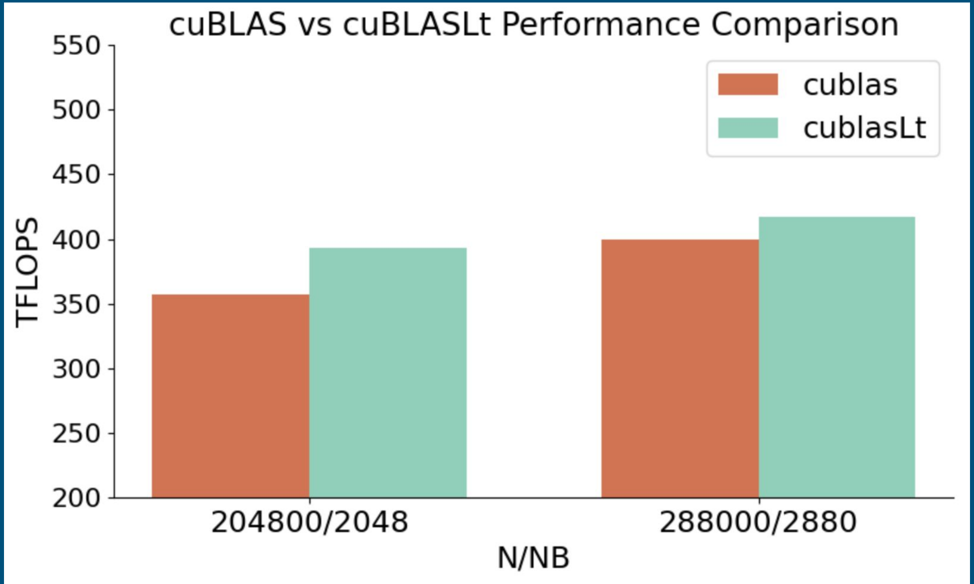
Strategy 1 – Cublas to CublasLt

- Nsight Systems shows frequent `cublasGemmEx` calls.
- **Idea:** Replaced it with `cublasLtMatmul` (cuBLASLt) .



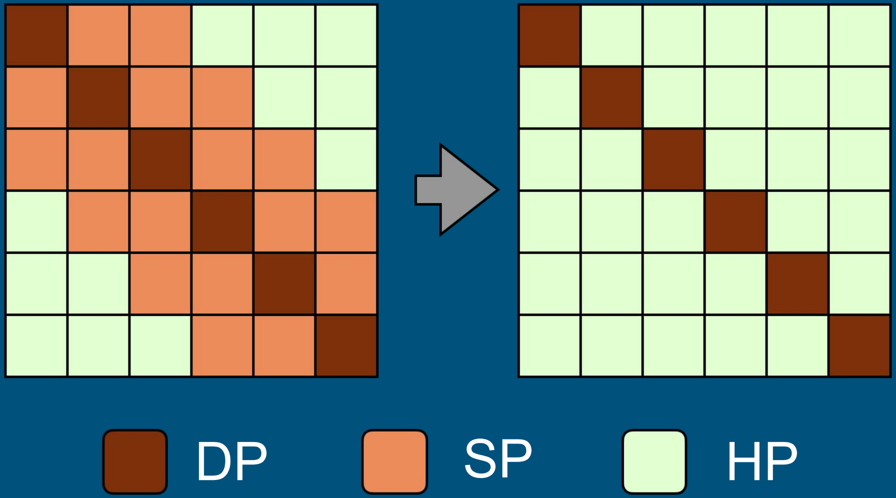
Result – Cublas to CublasLt

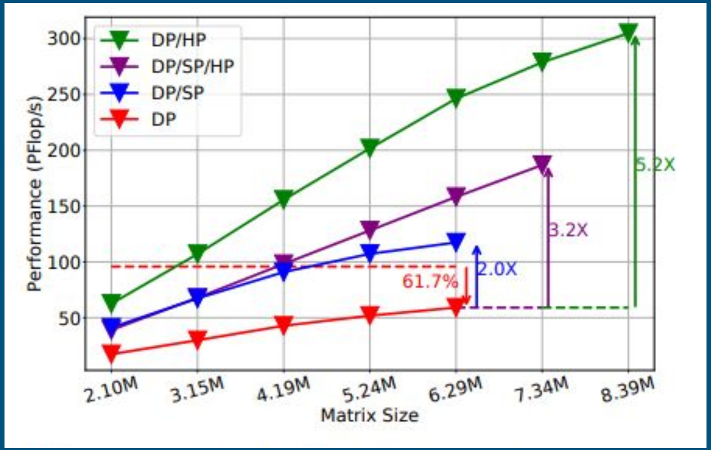
cublasLt delivered a **4.5~10%** improvement.



Strategy 2 – Mixed Precision

Following Abdulah et al. (2024), we introduce better tile-level mixed precision.





Source: Abdulah, S., et al

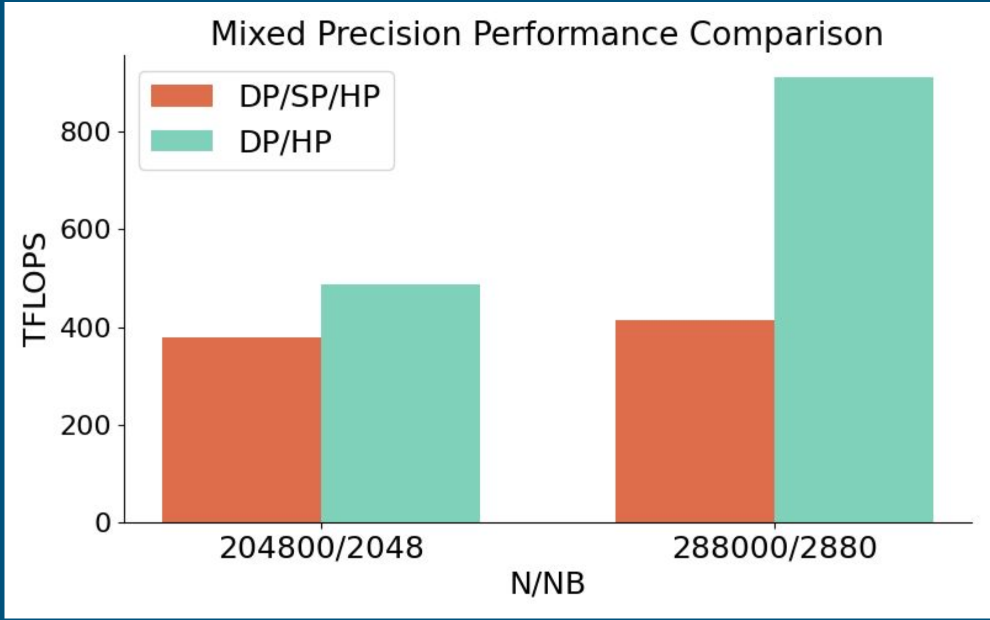
Results – Mixed Precision

Achieve more than **2x** improvement

L1 loss accuracy check

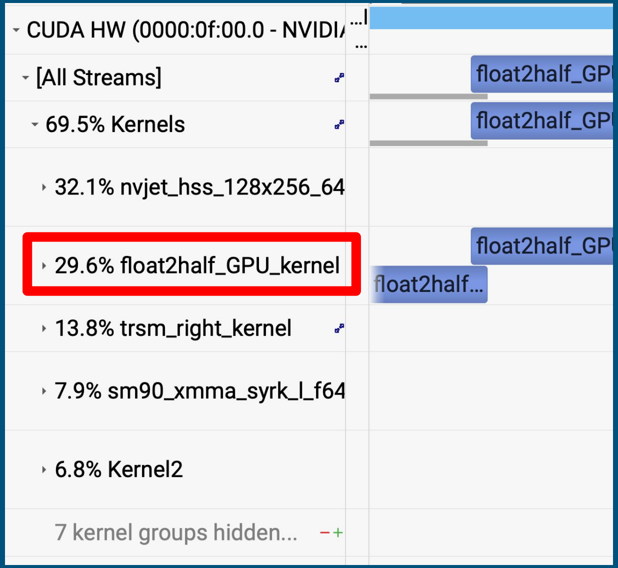
- DP/SP/HP : 5.3e-08
- DP/HP : 9.6e-06

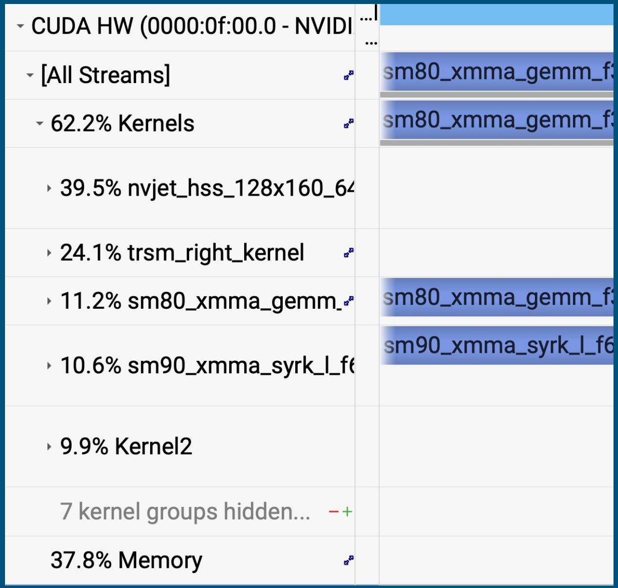
→ **Still remain accurate**



Strategy 3 – Data Conversion Overhead

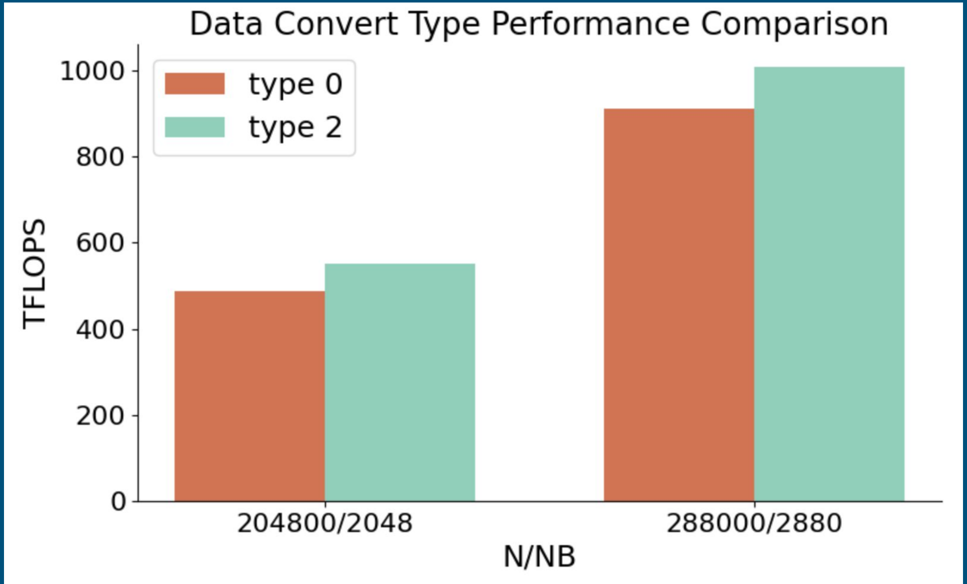
Use cublasLt's internal FP conversions instead of explicit float2half





Result – Data Convertion Overhead

Yielded **11~14%** improvement.



Strategy 4 – Adjust N/NB Size

N: matrix order

- Larger **N** increases work, keeps GPUs busier, and improves throughput.
- But too large **N** may overwhelm CPU RAM

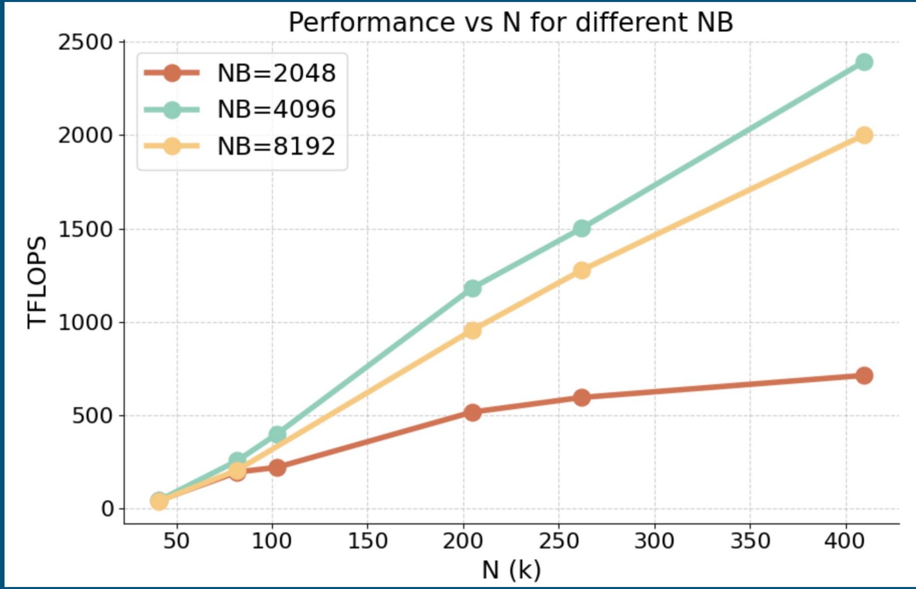
NB: tile size

- Larger **NB** boost higher arithmetic intensity
- But too large **NB** may increase DP percentage and harm performance

Need to strike a balance to maximize occupancy and intensity

Results – Adjust N/NB Size

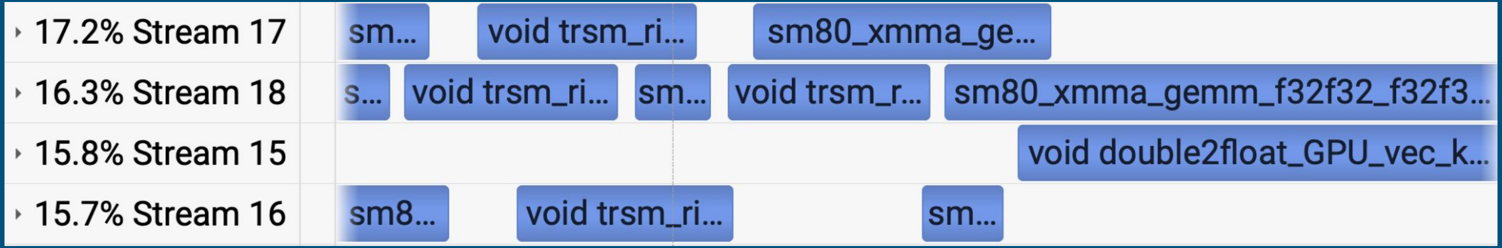
NB = 4096 with N = 409600 performs the best and reached **2450 TFLOPS!**



Strategy 5 – Cuda Streams number tuning

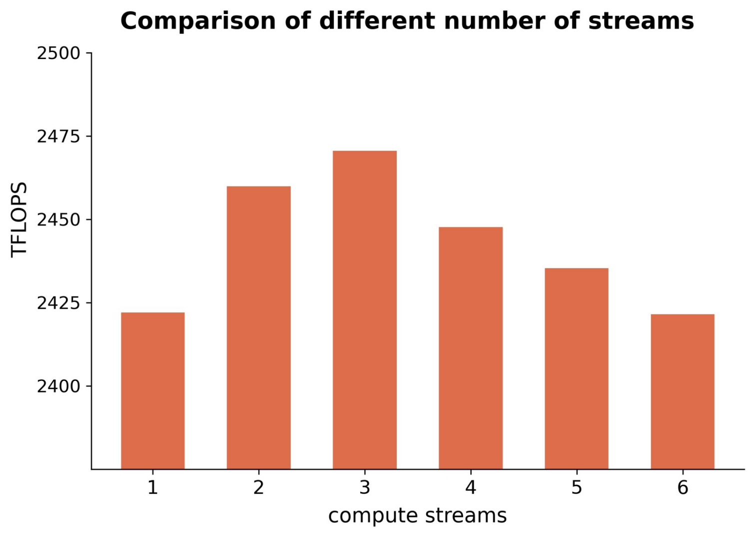
- **1 stream:** serialized; low overlap/SM util.

- **4+ streams:** concurrent & higher overlap but may contend for SM resources.



Results – Cuda Streams number tuning

After tuning, **3 cuda streams** strike the best balance between concurrency and resource contention. Achieving **2.47 PFLOPs** throughput.



Experiments are tested under problem size: N/NB=409600/4096

Final Performance

- Implement cuBLASLt
- Mix precision optimization
- Pipeline simplify
- Problem size tuning
- Multi-stream adjustment

Combined above optimizations, we achieve 2.47 PFLOPs, 777x speedup

Energy Efficiency

INPUTS	
# CPU Cores	112
# GPUs (H100)	8
Application Speedup	777.0x

Node Replacement	777.0x

GPU NODE POWER SAVINGS			
	Intel Platium 8480+	8x H100 80GB	Power Savings
Compute Power (W)	784,770	6,760	778,010
Networking Power (W)	36,081	93	35,988
Total Power (W)	**820,851**	**6,853**	813,998

Node Power efficiency	**119.8x**

ANNUAL ENERGY SAVINGS PER GPU NODE			
	Intel Platium 8480+	8x H100 80GB	Power Savings
Compute Power (kWh/year)	6,874,585	59,218	6,815,368
Networking Power (kWh/year)	316,073	814	315,259
Total Power (kWh/year)	**7,190,658**	**60,031**	**7,130,627**

$/kWh	$	0.18
Annual Cost Savings	$	1,283,512.78
3-year Cost Savings	$	3,850,538.34

Metric Tons of CO2	5,056
Gasoline Cars Driven for 1 year	1,091
Seedlings Trees grown for 10 years	83,571

(source: Link)

POWER ASSUMPTIONS		
Node Configurations	**Baseline Node** Intel Platium 8480+	**Alternative** 8x H100 80GB
CPU SKU	Intel Platium 8480+	Intel Platium 8480+
# CPU	2	2
# CPU Cores	112	112
CPU Power (W)	650	650
GPU SKU	0	H100 80GB SXM4
# GPU	0	8
GPU Power (W)	0	5600
Network Type	IB EDR	IB EDR
# Network Ports	2	2
Network Card Power (W)	60	60
RBoM Power (W)	300	450
Total Compute Node Power (W)	1010	6760
Core Network Power / Node	46	93
Total Power / Node	1056	6853

ASSUMPTIONS

(1) The workload being input will run 24/7/365 on the node in question

(2) When the workload runs on a fraction of a CPU or GPU server, no other bottlenecks occur to stop it from scaling up to occupy the full server

(3) The calculations use TDP for both CPU and GPU. In reality, neither server will run full time at TDP. The comparison here is "worst case CPU" vs. "worst case GPU"

(4) Annual cost savings are operational for electricity only. Capital, personnel, etc are not included

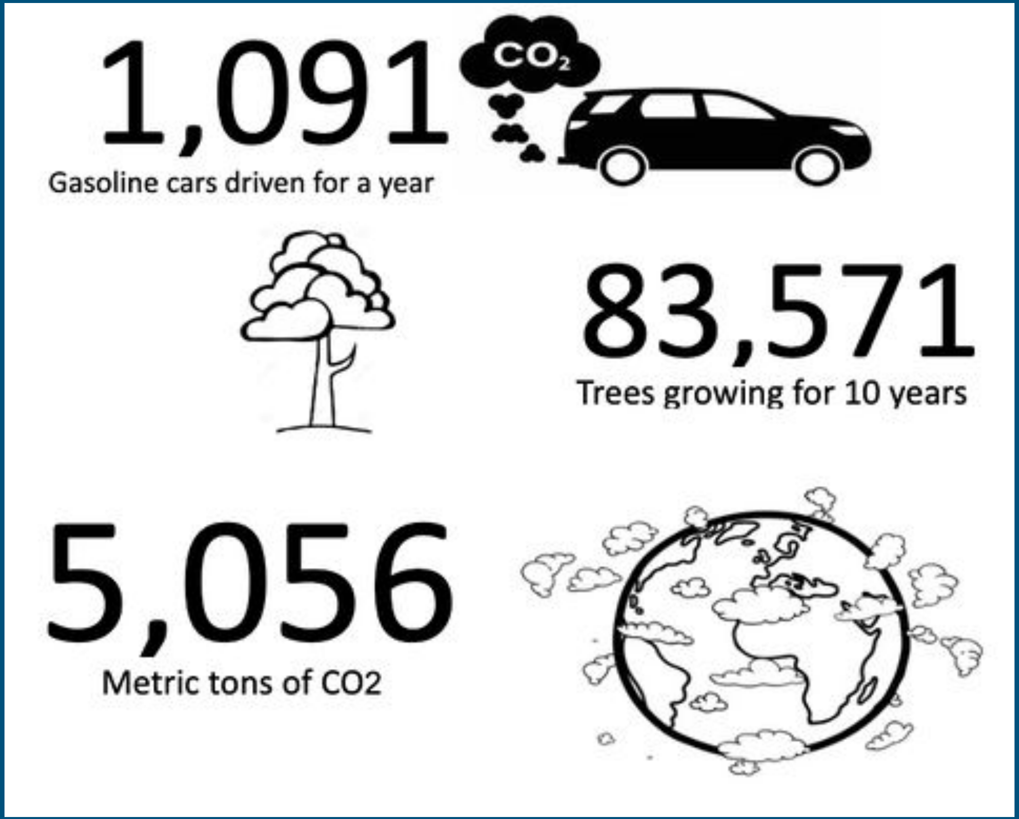
(5) Perfect scaling of the workload to multiple nodes for CPUs

(6) Fractional workload scaling for both CPU and GPU nodes

(7) The GPU machine runs the CPUs a full speed, full power draw

Energy Efficiency

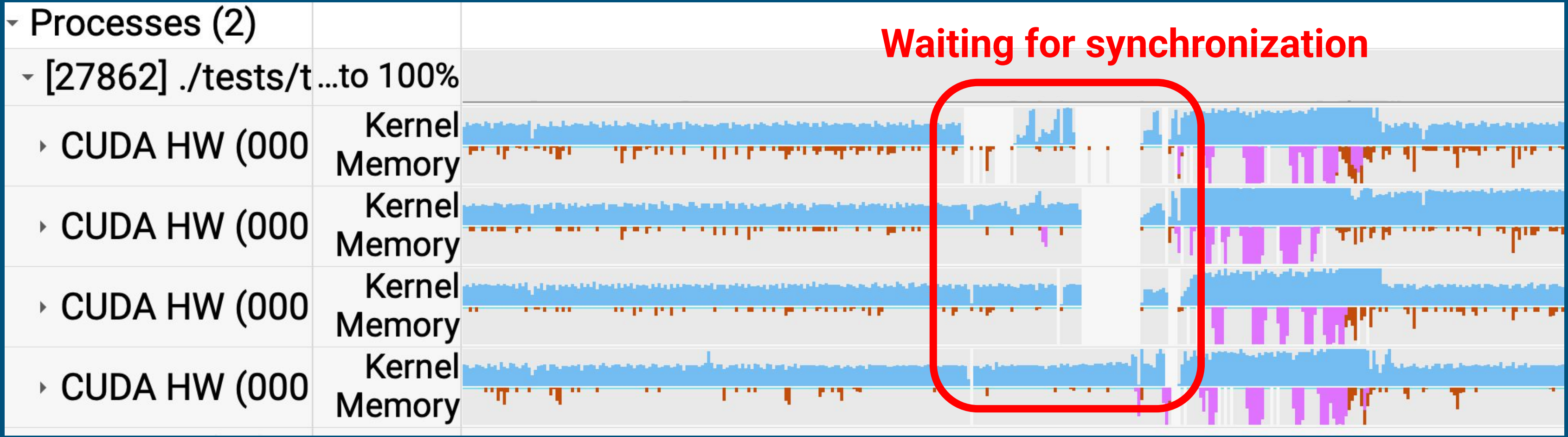
119x



Future Works

Workload imbalance between processes

- CUDA Graphs, matrix fusion



Problems Encountered During Optimization

Cublas operation does not append to different cuda streams

- Solution: Use `cublasGetStream()` to bind cublas on cuda streams

Problem size is bottlenecked by host RAM.

- We tune **N and NB** to pack GPUs with tiles, without exceeding **CPU RAM**.

Unstable execution time and throughput

- CPU bind to reduce resource contention.

Wishlist

What do you wish existed to make your life easier?

48 hrs in a day, and more cores to run the process faster!

Event

More opportunities to get to know other groups better

Systems

Each member has their own account instead of a team account

Final Thoughts

Was this Open Hackathon worth it?

Absolutely, pushed us to make progress every week with tight deadlines and meetings.

Future plans

Connect every stage, try running the full pipeline, and identify bottlenecks for further improvements.

What resources/support will be critical for your work after the event?

- H100 machine, back to V100 now :(
- Mentor support

Application Background

Exascale Climate Emulator addresses the escalating computational and storage requirements of traditional Earth System Model simulations:

- Ultra-high spatial resolution of approximately 3.5 km in space
- Very low memory/computation cost.

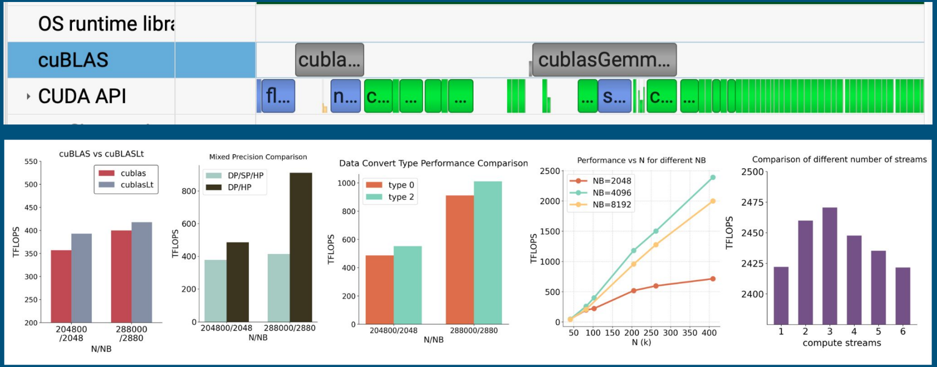


Fig.1 Profiling and Experiment Results. This application is CUDA-API-heavy, but can be improve by sequence of optimization.

Hackathon Objectives and Approach

Main Objective:
Accelerate Cholesky Factorization

- Cublas and CublasLt
- Mixed-precision computation
- Data conversion overhead
- Different N/NB sizes
- Cuda streams number tuning

Technical Accomplishments and Impact

Speedup: 777x!

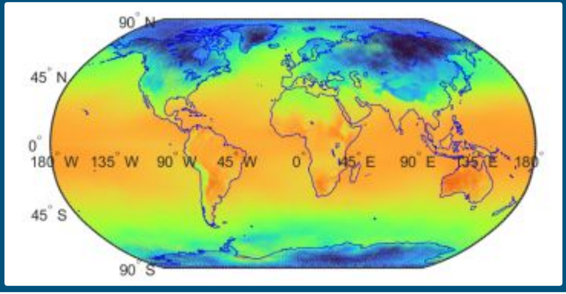
- Reduces months of computation to just days
- Saves energy equivalent to the emissions of 1,000 cars/year or the absorption of 80,000 trees over a decade.
- Faster climate prediction. Marks a major milestone in climate science and high performance computing.

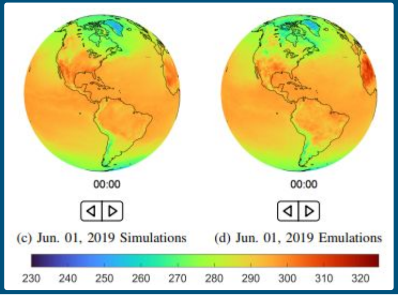
Appendix

A storyline for publication on NCHC's website.

高解析度大氣模擬計算器

研究領域示意圖

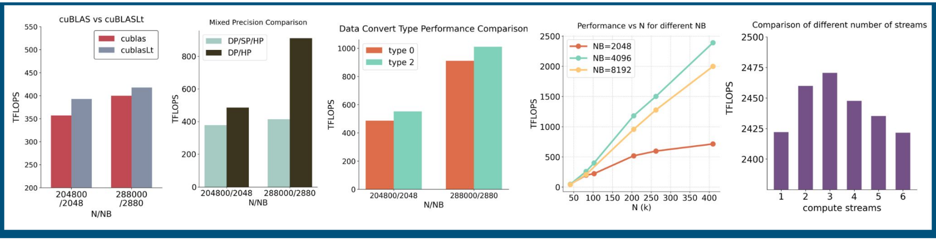




台大 Usagi 團隊來自李濬屹老師帶領的 ElsaLab 實驗室，將大氣模擬計算器加速了 777 倍！！

Exascale climate emulator 是新一代氣候模擬技術，它讓我們能以接近真實地球規模的速度與精度模擬氣候變化。傳統的高解析度氣候模型需要龐大的計算資源與時間，常常一個模擬要跑上數月甚至數年。而我們的系統透過 GPU 混合精度運算與分散式架構，大幅縮短模擬時間，達到 exascale（百億億次運算）級效能。這樣的加速代表著原本需要幾個月的運算現在可能只需幾天，節省的能量相當於 5,056 公噸二氧化碳，也就是一千多輛汽車一年的排放或八萬多棵樹十年的吸碳量。這樣的突破讓研究者能更快預測極端氣候、模擬不同政策下的地球變化，並推動永續與減碳決策，是氣候科學與高效能運算領域的重要里程碑。

實驗結果



報告投影片連結(由國網上傳到github)