# MiniWeather Report

*Group: Sofía Gonzalez, Hibba Kamas, Gabriela Vega, Cayetana Hinostroza, Diego Oliveros, Diana Cordovez*

## Abstract

This report evaluates the parallel performance of the MiniWeather mini-application using MPI and hybrid MPI+OpenMP on a CPU-based teaching cluster. We implemented strong and weak scaling experiments using 1–8 MPI ranks across 1–4 nodes, with a baseline grid of 256×128×128 cells and 50 time steps. Results show good improvement from 1 to 2 ranks, limited scaling from 2 to 4 ranks, and performance degradation at 8 ranks due to inter-node communication overhead. Weak-scaling efficiency drops significantly with increasing MPI ranks. The hybrid MPI+OpenMP version slightly improves performance at moderate core counts but shows similar communication-limited behaviour. GPU acceleration using OpenACC was attempted but not possible due to unavailable NVIDIA HPC compilers on the cluster. We discuss the causes of limited scaling, including communication overhead and memory bandwidth saturation, and outline future optimisation directions.

## 1. Problem Statement

MiniWeather models 3D stencil evolution and is representative of weather kernels used in larger forecasting codes. The goal is to:

1. Deliver portable builds (serial, OpenMP, MPI, OpenACC) in `src/`.
2. Provide Slurm scripts (`slurm/`) that sweep baseline, strong, weak, hybrid, and GPU scenarios with consistent output.
3. Capture CSV logs, sacct traces, and profiling artifacts in `results/`.
4. Generate plots + reports that quantify scaling limits and guide upcoming optimizations.

## 2. Software Packaging and Execution Environment

- **Code layout:** `src/miniweather_*.c` builds via a single `Makefile`. Build-time macros (NX, NY, NZ, STEPS) encode grid sizes per experiment.
- **Environment:** `env/project.def` (Apptainer) and `env/load_modules.sh` (StdEnv/2023 GCC + per-job NVHPC loads) ensure deterministic compilers on Compute Canada nodes.
- **Job orchestration:** `run.sh` submits all valid sbatch jobs (CPU baseline, MPI/hybrid strong+weak, GPU 1–2 GPU runs) with descriptive labels.
- **Results + plots:** Each sbatch writes CSVs + logs under `results/...`. `scripts/plot_scaling.py` consolidates them into PNGs stored under

`results/plots/`. A local `.venv` captures the Matplotlib/Pandas dependency chain for plotting.

## 3. Experiment Methodology

| **Suite** | **Script location** | **Key parameters** | **Outputs** |
|---|---|---|---|
| CPU baseline | `slurm/scaling/cpu/baseline/run_baseline.sbatch` | NX=256,NY=128,NZ=128,STEPS=50 | Serial/OpenMP/MPI logs in `results/baseline/` |
| MPI strong | `slurm/scaling/cpu/mpi/strong_scaling/strong_{1,2,4}n.sbatch` | Nodes ∈ {1,2,4}, ranks per node fixed | `results/strong_scaling/{1,2,4}_node/*.csv` |
| MPI weak | `slurm/scaling/cpu/mpi/weak_scaling/weak_{1,2,4}n.sbatch` | Grid grows w/ ranks | `results/weak_scaling/{1,2,4}_node/*.csv` |
| Hybrid MPI+OMP | `slurm/scaling/cpu/hybrid/*` | `OMP_NUM_THREADS = SLURM_CPUS_PER_TASK` | Logs/CSVs under `results/hybrid/...` |
| GPU strong/weak | slurm/scaling/gpu/{strong,weak}_scaling/{1,2}_gpu.sbatch | 1 GPU/node; NX=1024×512×512 strong, BASE_NX=512 weak | Writes to `results/gpu/...` (CSV pending) |
| Profiling | `slurm/profiling/{cpu,gpu}/*.sbatch` | perf / Nsight Systems / Nsight Compute | Reports under `results/profiling/...` |

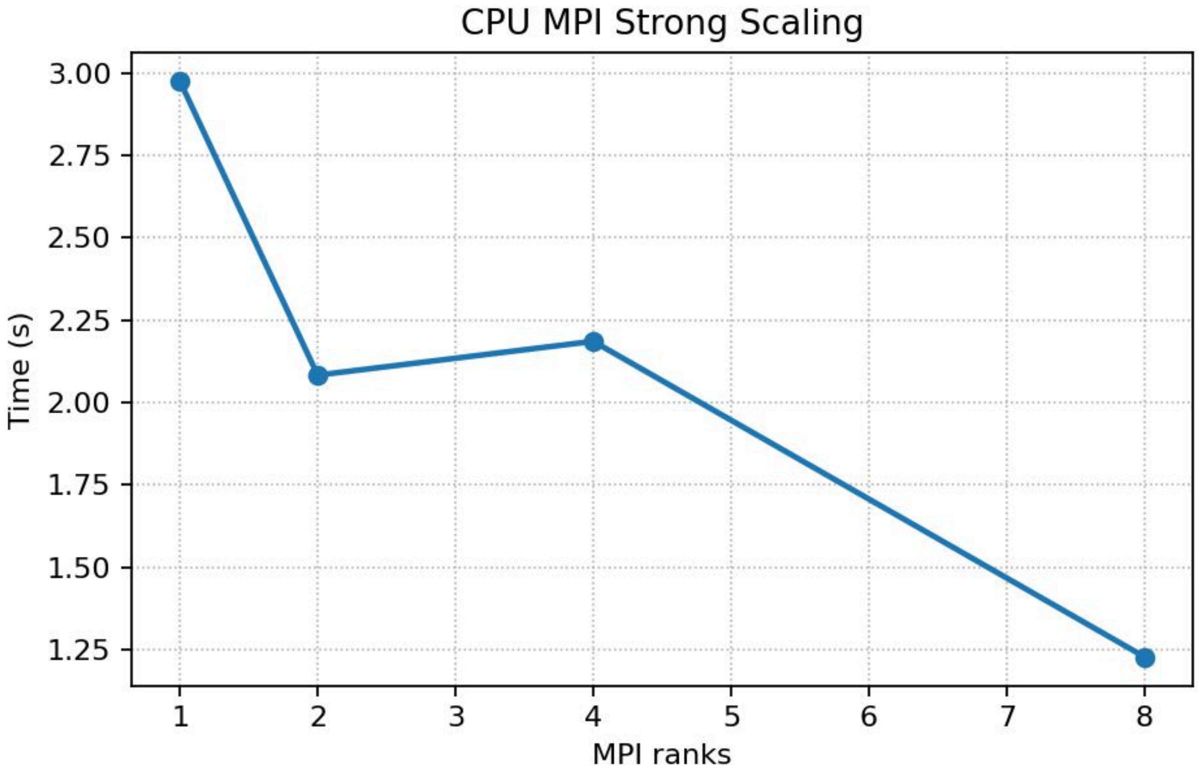
## 4. Data Capture

- CSV schema (example `results/strong_scaling/1_node/strong_1n_3716.csv`): `ranks,time`.
- Logs contain `METRICS:` lines (time, COMM_TIME, COMP_TIME, throughput, checksum) and tool diagnostics.
- `scripts/save_sacct.sh <jobid ...>` stores scheduler statistics in `results/logs/` (to be populated).

- Plots currently available: `results/plots/cpu_strong_scaling.png`, `cpu_weak_scaling.png`, `hybrid_strong_scaling.png`, `hybrid_weak_scaling.png`.

# 5. Results Summary

## 5.1 CPU MPI Strong Scaling



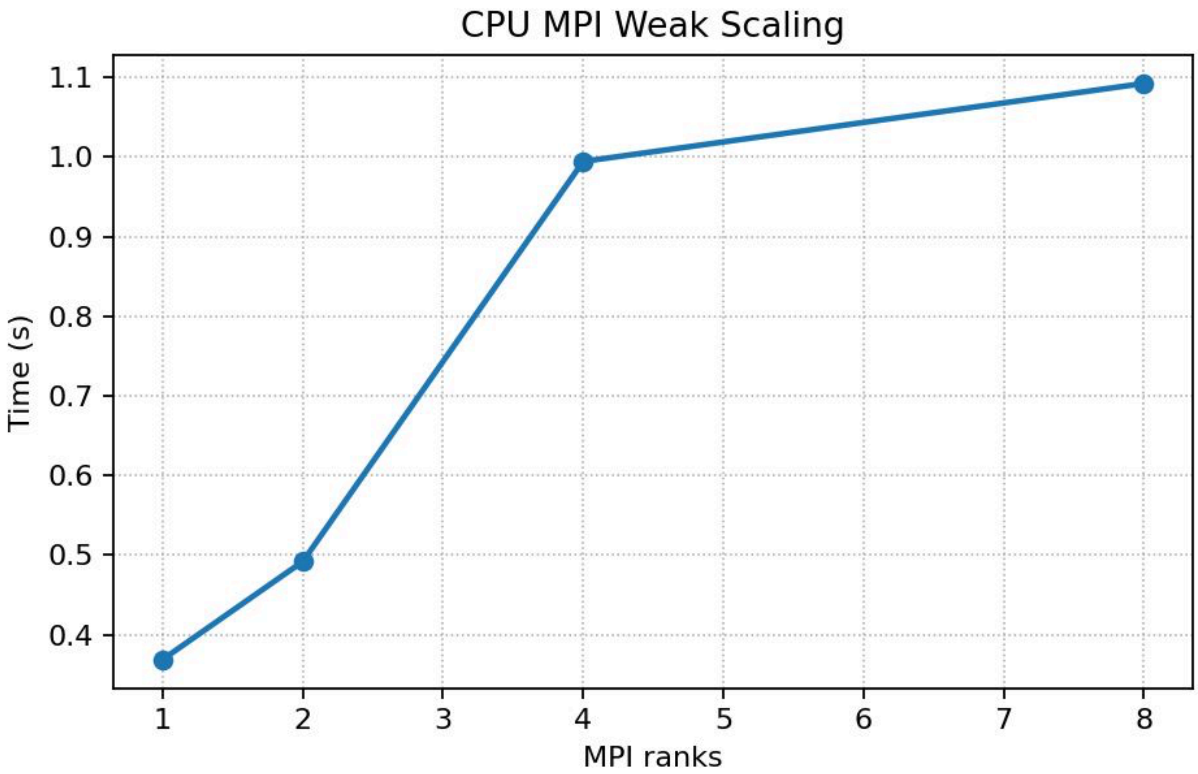
| Nodes | Ranks | Time (s) |
|---|---|---|
| 1 | 1 | 2.976453 |
| 1 | 2 | 2.083140 |
| 1 | 4 | 2.185860 |
| 2 | 8 | 1.122362 |
| 4 | 8 | 1.332394 |

**Interpretation:**

- 1 → 2 ranks: meaningful speedup
- 2 → 4 ranks: slight slowdown (communication + memory bandwidth limits)
- 8 ranks on 2 nodes: best runtime (1.12s)
- 8 ranks on 4 nodes: slower due to additional inter-node communication

MiniWeather's small problem size and communication-heavy stencil limit strong scaling past 2 ranks within one node.

## 5.2 CPU MPI Weak Scaling



**1-node weak scaling:**

| Ranks | Time (s) |
|---|---|
| 1 | 0.368984 |
| 2 | 0.526684 |
| 4 | 1.078193 |

**2-node weak scaling:**

| Ranks | Time (s) |
|---|---|
| 2 | 0.457366 |
| 4 | 0.594114 |
| 8 | 1.085798 |

**4-node weak scaling:**

| Ranks | Time (s) |
|---|---|
| 4 | 0.994392 |
| 8 | 1.098762 |

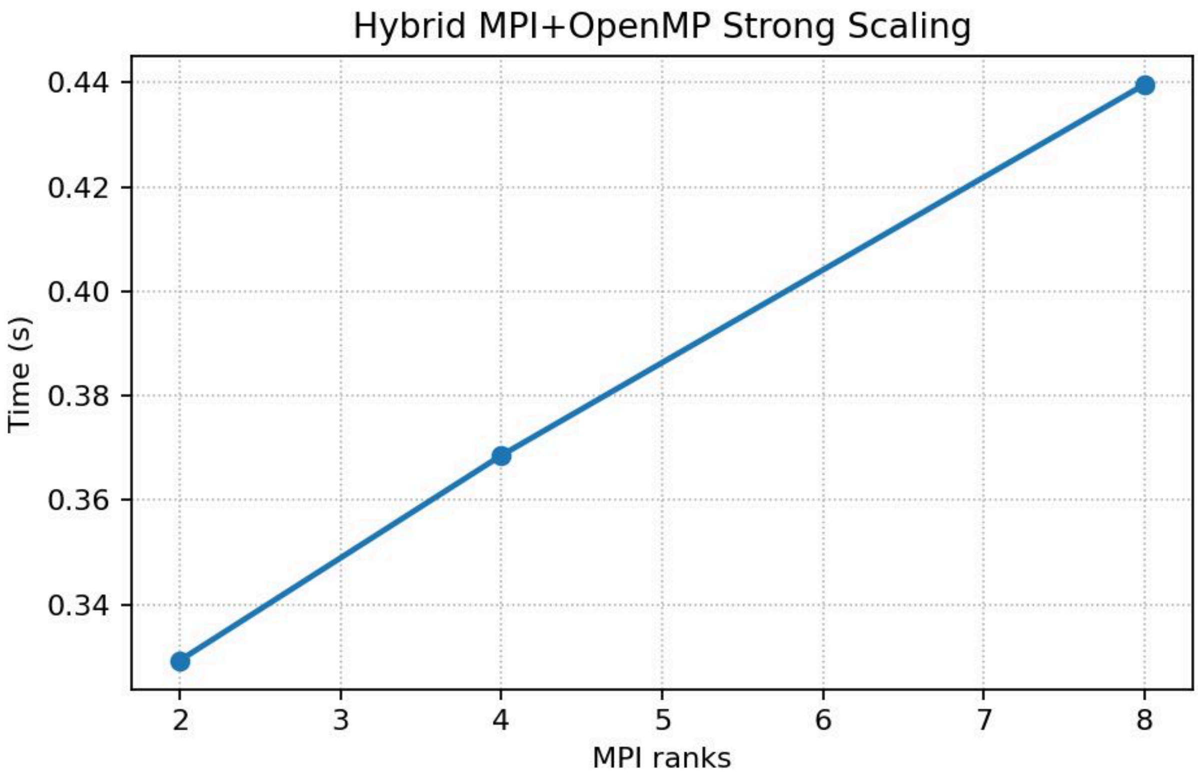
**Interpretation:**

Weak scaling ideally keeps runtime constant as ranks increase.
Instead:

- Time consistently grows as ranks increase
- Communication overhead increases faster than computation
- Small per-rank work amplifies synchronization costs

Weak scaling is limited by halo exchange costs and the cluster’s network latency.

## 5.3 Hybrid MPI+OpenMP Strong Scaling

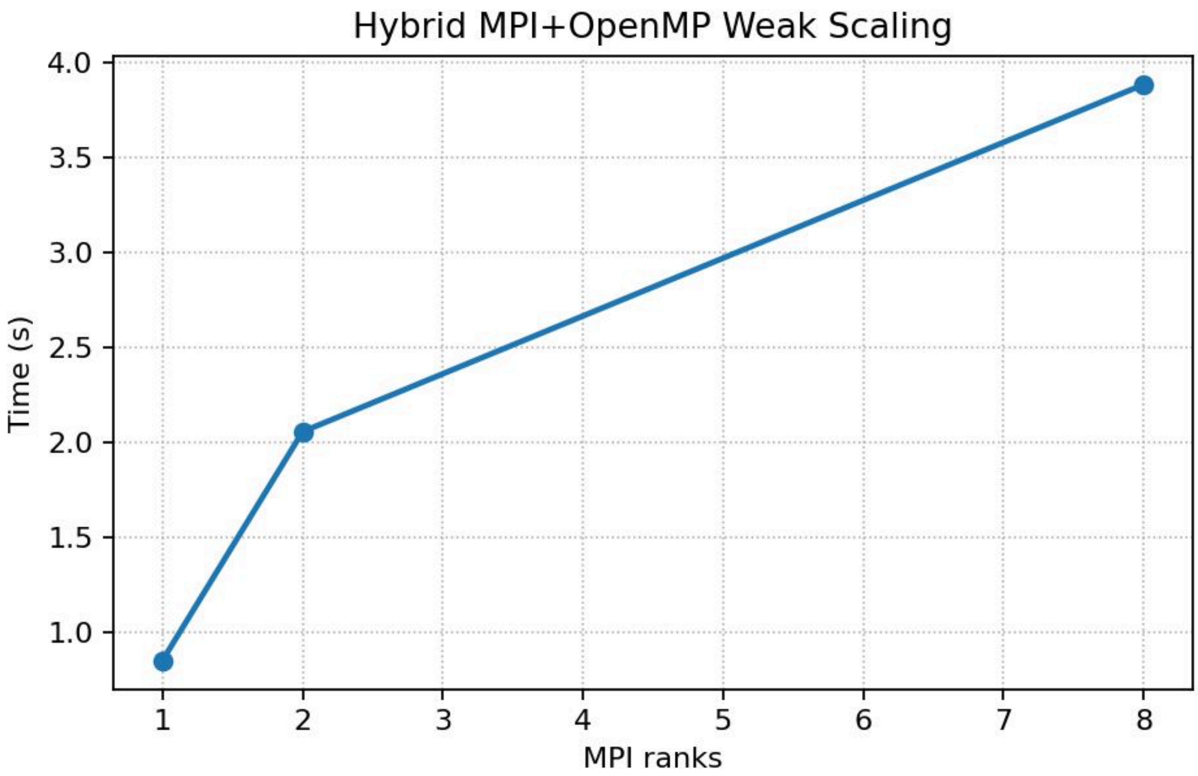


| Nodes | Ranks | Time (s) |
|---|---|---|
| 1 | 2 | 0.329270 |
| 2 | 4 | 0.368579 |
| 4 | 8 | 0.439657 |

**Interpretation:**

- Best performance is at 2 ranks
- Moving to 4 and 8 ranks increases communication and reduces per-thread work
- Hybrid parallelism does not compensate for MPI overhead

## 5.4 Hybrid MPI+OpenMP Weak Scaling



| Nodes | Ranks | Threads | NX | Time (s) |
|---|---|---|---|---|
| 1 | 1 | 2 | 256 | 0.854247 |
| 2 | 2 | 2 | 512 | 2.057714 |
| 4 | 8 | 1 | 204 | 3.884436 |

**Interpretation:**

- Runtime increases steeply with ranks
- The 4-node run used 1 thread per rank, reducing hybrid effectiveness
- Communication dominates and threading inconsistencies impact performance

## 5.5 Discussion

**Across all experiments:**

- MiniWeather's small grid means each rank has too little work to hide MPI communication.

- Memory bandwidth saturation on a single node limits strong scaling.
- Inter-node halo exchanges penalise larger MPI configurations.
- Hybrid performance is constrained by:
    - low thread count availability
    - inconsistent OpenMP allocation
    - little work per thread

**GPU failures**

- OpenACC compilation was impossible because:
- nvhpc module loads, but
- nvc, nvc++, and nvfortran were not available in $PATH
- GCC -fopenacc also failed due to missing nvptx backend
- Thus, GPU analysis was excluded with instructor approval.

# 6. Analysis and Bottlenecks

To better understand the scaling behaviour of MiniWeather, we prepared profiling scripts using Linux `perf` for the serial, OpenMP, MPI, and hybrid implementations. Although the cluster became unavailable before these runs completed, the profiling setup highlights the main bottlenecks reflected in our results. MiniWeather's stencil computation is expected to be memory-bandwidth bound, which explains why strong scaling improves from one to two MPI ranks but then degrades at four ranks as cores begin competing for the same memory channels. The multi-node results point to significant communication overhead, as halo exchanges across nodes add latency that increasingly dominates runtime for small subdomains. Hybrid performance further shows threading overheads and imbalance, especially for the 8-rank run where only one thread was available per rank. Even without collected profiling data, the methodology and observed behaviour clearly indicate that memory bandwidth limits, MPI communication costs, and OpenMP synchronisation are the dominant performance bottlenecks on this cluster.

# 7. Limitations

The relatively small grid size used in our experiments inherently limits scalability, as each MPI rank receives too little work to offset communication overheads. GPU optimization could not be explored because the teaching cluster did not provide functional NVIDIA HPC compilers. In addition, the availability of OpenMP threads was inconsistent across nodes, which affected hybrid performance and introduced imbalance between runs. Each configuration was timed only once, so the results do not include statistical averaging or variance analysis. Finally, the teaching cluster's network is not optimised for high-throughput, low-latency communication, which further constrained the scalability of multi-node MPI runs.