MAUSAM, **77**, 3 (July 2026),1005-1018

MAUSAM

DOI: https://doi.org/10.54302/mausam.v77i3.7254
Homepage: https://mausamjournal.imd.gov.in/index.php/MAUSAM

UDC No.551.509.313:004.4.272

Performance profiling and optimization of the NCMRWF's unified model (NCUM) vn13.0 on the MIHIR cray XC40 HPC facility at NCMRWF

SHIVALI GANGWAR, ASHISH ROUTRAY*, SURYAKANTI DUTTA, PREVEEN KUMAR D,

V. S. PRASAD and B. ATHIYAMAN

National Centre for Medium Range Weather Forecasting (NCMRWF),

Ministry of Earth Sciences (MoES), A-50, Sector-62, Noida 201309, India

(*Received 25 August 2025, Accepted 13 January 2026*)

***Corresponding author's email: ashishroutray.iitd@gmail.com**

सार – मध्यम श्रेणी और मौसम पूर्वानुमान के राष्ट्रीय केंद्र (NCMRWF) ने MIHIR हाई-परफॉर्मेंस कंप्यूटिंग (HPC) सुविधा का संचालन किया है, जो संख्यात्मक मौसम पूर्वानुमान (NWP) मॉडल चलाने के लिए 2.8 पेटाफ्लॉप तक की प्रसंस्करण क्षमता प्रदान करता है, जिससे सटीक और समय पर मौसम पूर्वानुमान सक्षम होता है। यह अध्ययन MIHIR क्रे (Cray) XC40 HPC सुविधा पर NCMRWF यूनिफाइड मॉडल (NCUM) संस्करण 13.0 के व्यापक प्रदर्शन प्रोफाइलिंग और अनुकूलन विश्लेषण को प्रस्तुत करता है। इन मॉडलों को प्रति सेकंड पेटा फ्लोटिंग-पॉइंट ऑपरेशन्स (PFLOPS) के क्रम पर गणना की आवश्यकता होती है। इसके कंप्यूट नोड्स ड्रैगनफ्लाई टोपोलॉजी में क्रे एरीज़ (Cray Aries) उच्च बैंडविड्थ, कम विलंबता इंटरकनेक्ट द्वारा जुड़े हुए हैं। NCMRWF यूनिफाइड मॉडल (NCUM) संस्करण 13.0 को रनटाइम बाधाओं और संचार ओवरहेड की पहचान करने के लिए क्रे परफॉर्मेंस एनालिसिस टूल (CrayPAT) का उपयोग करके MIHIR पर दो क्षैतिज रिज़ॉल्यूशन n96e (~130 किमी) आधार रेखा के रूप में और n1280e (~10 किमी) उच्च रिज़ॉल्यूशन कॉन्फ़िगरेशन में प्रोफाइल किया गया है। प्रदर्शन का मूल्यांकन दो क्षैतिज प्रस्तावों पर किया गया था, और अनुकूलन प्रयोगों के प्रभावों, जो प्रभामंडल आकार में कमी (Halo size reduction), डोमेन अपघटन रणनीतियों (Domain decomposition strategies), OpenMP थ्रेडिंग, और MPI रैंक ऑर्डरिंग पर केंद्रित थे, का मूल्यांकन किया गया था। परिणाम प्रदर्शित करते हैं कि विस्तारित प्रभामंडल चौड़ाई के आकार को 10 से घटाकर 5 ग्रिड पॉइंट करने से निष्पादन समय में लगभग ~10 सेकंड का सुधार होता है और इस प्रकार संचार लागत कम हो जाती है, और प्रमुख रूटीन में असंतुलन 15.2% से घटकर 3.8% हो जाता है। डोमेन अपघटन में 4×8 से 3×12 में बदलाव ने MPI सामूहिक असंतुलन को और कम कर दिया, जबकि रैंक पुनर्रचना ने निष्पादन समय में 12.8% तक सुधार किया। ये निष्कर्ष क्रे XC40 सिस्टम और भविष्य के HPC आर्किटेक्चर पर उच्च-रिज़ॉल्यूशन NWP मॉडलों को कुशलतापूर्वक स्केल करने के लिए डोमेन और संचार अनुकूलन का मार्गदर्शन करने में मूल्यवान होंगे जिन्हें NCMRWF में तैनात किया जाएगा।

ABSTRACT. The National Centre for Medium Range and Weather Forecasting (NCMRWF) operated the MIHIR High-Performance Computing (HPC) Facility, delivering up to 2.8 petaflops of processing capacity to run Numerical Weather Prediction (NWP) models, thereby enabling accurate and timely weather forecasting. This study presents a comprehensive performance profiling and optimization analysis of the NCMRWF Unified Model (NCUM) version 13.0 on the MIHIR Cray XC40 HPC Facility. The model requires computations in the order of peta floating-point operations per second (PFLOPS). The NCUM is profiled at two horizontal resolutions n96e (~130 km) taken as baseline and n1280e (~10 km) using the Cray Performance Analysis Tool (CrayPAT) to identify runtime bottlenecks and communication overhead. The optimization experiments focused on halo size reduction, domain decomposition strategies, OpenMP threading, and MPI rank reordering. Results demonstrate that reducing the extended halo size from 10 to 5 grid points improves execution time by ~10 seconds and thus reduces communication costs and lowers imbalance in key routines from 15.2 % to 3.8 %. A change in domain decomposition from 4×8 to 3×12 further reduced MPI collective imbalance, while rank reordering improved execution time by up to 12.8%. These findings will be valuable in guiding domain and communication optimizations to efficiently scale high-resolution NWP models on the Cray XC40 system and future HPC architectures to be deployed at NCMRWF.

Key words – NCUM, Cray XC40, MPI load, Halo exchange, Domain decomposition, CrayPAT.

1. Introduction

Advances in numerical weather prediction (NWP) have emerged from decades of sustained scientific and technological progress rather than isolated physics breakthroughs. Despite this incremental evolution, NWP ranks among the most impactful achievements in the physical sciences due to the computational endeavour

TABLE 1

NCMRWF HPC Evolution

Year	System Name	Description/Key Features
1989	Cray X-MP/14	First operational supercomputer for Indian meteorology
1999	PARAM (C-DAC)	First indigenous HPC for weather modelling
2001	Cray SV1(24 Processors)	The major upgrade to its high-performance computing infrastructure. This air-cooled, vector processor supercomputer, capable of up to 500 MHz, ran on the UNICOS operating system and marked a significant step in NCMRWF's capacity for NWP.
2006	Cray C1e (64 processors)	It was part of a series of supercomputing upgrades at NCMRWF, contributing to the centre's high-performance computing capabilities for weather and climate research.
2010	IBM Power6	Cluster architecture, modernized large-scale simulations
2015	BHASKARA (IBM iDataPlex)	High-capacity IBM system, global top 100 ranking
2018	MIHIR (Cray XC40)	Modern Cray XC40, 2.8 petaflops for NCMRWF

(Das & Bose, 1958; Das, 1972; Bauer *et al*., 2015). The development of supercomputing facilities at the National Centre for Medium Range Weather Forecasting (NCMRWF) has played a critical role in advancing NWP capabilities in India. These advancements enabled improvements in forecast accuracy and timely weather forecasts in different spatial and temporal scales (Rajagopal *et al*., 2014; Prasad *et al*., 2014; Routray *et al*., 2017, 2019). The efforts commenced in 1989 with the commissioning of the Cray X-MP/14, the nation's first supercomputer dedicated to meteorological applications. This early acquisition of high-performance computing (HPC) positioned India among the early adopters of HPC for advanced atmospheric research and complex numerical simulations. In 1999, a significant milestone was achieved with the shift towards indigenous computing power through the deployment of the PARAM supercomputer developed by the Centre for Development of Advanced Computing (C-DAC). This move demonstrated India's growing self-reliance in HPC technology and ensured that the computing needs of weather forecasting could be met with customized, locally supported infrastructure. The PARAM system showcased the nation's commitment to build high-performance computing capabilities for meteorological research and operational forecasting.

A new Global Forecast System (GFS) at T254L64 resolution has been implemented at NCMRWF, on the Param Padma (IBM P5 based) and Cray-X1E computer systems. On the Cray-X1E, one analysis cycle, followed by a seven-day forecast integration, took approximately three hours. The GFS has been running in real-time operational mode since 1st January 2007 (Rajagopal *et al*., 2007). In 2009, NCMRWF commissioned an IBM Power6 cluster, boosting its capability for large-scale simulations and shifting from vector-based to cluster-based architectures. This upgrade played a crucial role in preparing NCMRWF for the new generation of parallelized global forecast models. Thereafter in 2015, NCMRWF commissioned the IBM iDataPlex HPC (named, Bhaskara), ranked among the world's top 100 supercomputers and optimized for weather and climate applications (Rakhi *et al*., 2016). A major upgrade followed in 2018 with a 2.8-petaflop peak performance, Cray XC40 (named, MIHIR), which enabled high-resolution global and regional modelling, improved forecast accuracy, and better prediction of extreme weather events (Kumar *et al*., 2018). Its peak performance enabled extensive calculations for both near and extended-range forecasts using the NCMRWF Unified Model (NCUM), implemented across both global and regional domains (Kumar *et al*., 2018). The hybrid parallelization strategies have been adopted with this upgrade, which combines Message Passing Interface (MPI) and OpenMP threading on multicore HPC systems like MIHIR. MPI enables the model to scale efficiently across distributed nodes, while OpenMP enhances parallelism at the shared-memory level within each node, thereby improving data locality and reducing inter-process communication overheads. However, the effectiveness of these parallel methods depends on implementing optimization and scalability strategies for the models, considering hardware configurations such as NUMA mappings and the structure of model calculations and communication patterns. (Sivalingam *et al*., 2015). The detailed evolution of HPC at NCMRWF is provided in Table 1.

NCMRWF runs two global deterministic forecast models, the NCMRWF UM and the NCMRWF GFS. The NCUM is based on the UK Met Office Unified Model, configured at a horizontal resolution of ~12 km (n1024e) with 70 vertical levels. It provides 10-day forecasts twice daily, initialized at 00 & 12 UTC. The NGFS, on the other hand, is a spectral model configured at T574L64 (~12 km), also providing 10-day forecasts. While both models generate global medium-range forecasts, their physics packages differ; the NCUM uses semi-Lagrangian

advection and non-hydrostatic dynamics, whereas NGFS employs spectral methods. These two models are often run in parallel, providing independent guidance that helps us to evaluate model uncertainty and robustness (Rajagopal *et al*., 2007; Rajagopal *et al*., 2014). To strengthen probabilistic forecasting, NCMRWF developed the NCUM based Ensemble Prediction System (NEPS), adapted from the UK Met Office's ensemble framework, NEPS is configured at ~12 km horizontal resolution with 70 vertical levels with 22 members & ensemble forecasts up to 10 days. It provides valuable probabilistic guidance on extreme weather events, tropical cyclones, and heavy rainfall (Sarkar *et al*., 2016; Prasad *et al*., 2019).

High-resolution NWP models need robust computational and parallelization methods to meet operational deadlines. These models employ complex physics parameterizations and data assimilation techniques, which frequently result in performance bottlenecks and scaling challenges. These employ hybrid parallelization strategies that combine the Message Passing Interface (MPI) for distributed-memory parallelism and OpenMP for shared-memory parallelism. While MPI enables efficient scaling across distributed nodes and OpenMP enhances parallelism within nodes, their effectiveness depends critically on hardware alignment, NUMA mappings, and communication patterns (Sivalingam *et al*. 2015). Profiling of the NCUM model on MIHIR, helped design domain decomposition strategies and MPI/OpenMP optimizations (Bermous & Steinle, 2015) used to achieve efficient scalability and minimize runtime. Performance tuning tools such as CrayPAT help identify both code-level and system-level bottlenecks, thereby guiding optimization efforts that make high-resolution simulations more efficient (Sivalingam *et al*., 2015).

In large-scale operational runs, communication overheads, especially halo exchanges and collective MPI calls can dominate execution times. Load imbalance, caused by uneven work across MPI ranks or delays between nodes, can also reduce scalability (Chunduri *et al*., 2018; Gangwar *et al*., 2023). Approaches such as MPI rank reordering, designed to improve processor placement and reduce off-node communication, have been shown to significantly enhance performance (Mercier & Jeannot, 2011). The study examines the performance of NCUM version 13.0 on the MIHIR HPC at two distinct horizontal resolutions: n96e (~130 km), taken as the baseline, and n1280e (~10 km). Key areas of focus include evaluating the NCUM execution profiles at multiple resolutions and node counts, analysing the load imbalance and MPI communication overhead using the Cray Performance Analysis Tool (CrayPAT), evaluating the impact of halo size and domain decomposition on performance, and assessing the benefits of MPI rank reordering for high-resolution configurations. Insights from this work support both future code enhancements and more effective utilization of India's leading meteorological HPC assets, ultimately strengthening national weather forecasting capabilities, thereby preparing the site for profiling and optimization of the latest NWP model and higher-resolution configurations on the new commissioned HPC at NCMRWF. MIHIR continues to serve as a major computational resource supporting operational weather prediction and climate applications under the Ministry of Earth Sciences (MoES, 2023). Thus, strengthening NCMRWF's weather forecasting capabilities and meeting the growing demand for precise and timely weather forecasts globally and across the Indian subcontinent.

2. Data and methodology

2.1. *Model configuration and setup*

NCUM version 13.0 (released 25 July 2022, revision r107106) was compiled and executed on MIHIR using Cray Compiling Environment (CCE) version 8.7.7. The model initial conditions are generated by the NCMRWF's operational global deterministic hybrid 4DVAR atmospheric data assimilation (DA) system (George *et al*., 2016). Configuration management is handled through the Rose suite framework, while job scheduling and execution are coordinated using the Python-based Cylc workflow engine. NCUM configurations analysed were n1280e, ~10 km horizontal resolution, 70 vertical levels, 36, and 288 compute nodes. Each node used 12 MPI ranks × 3 OpenMP threads (36 cores), and the global MPI rank count varied by decomposition (e.g., 4×8=32, 3×12=36) for n96e.

2.2. *Hardware specifications*

The MIHIR Cray XC40 features 2,322 compute nodes, each equipped with two 18-core Intel Xeon Broadwell processors (2.1 GHz), 128 GB of DDR4 memory (64 GB per NUMA domain), and dual 24 MB L3 caches. Compute nodes are interconnected via Cray Aries with dragonfly topology. Peak performance is 2.8 petaflops (PF).

A detailed summary of the system's hardware specifications (Gangwar *et al*., 2023) is presented in Table 2.

2.3. *Compilation and compiler optimizations*

The NCUM compilation used Cray cc and Cray ftn (Fortran compiler driver) version 8.7.7, with optimization

TABLE 2

MIHIR hardware specifications (Gangwar *et al*., 2023)

Components	MIHIR
Processor Type	Intel Xeon Broadwell E5-2695
Number of Compute Nodes	2322
Total Number of Cores	83592
Interconnect	Cray Aries with Dragon Fly topology
Node Processor Cores	2 x (2.1 GHz, 36-Core)
Memory Size per Node	128 GB
Memory Type	DDR4
Node Cache Size	2 x 24 MB
Turbo Boost	OFF
Usage of Hyper Threading	ON
Peak Compute Power	2806 TF
Effective Distributed Memory	297

flags selected for vectorization, OpenMP parallelism, cache usage, branch prediction reduction, loop unrolling, and architecture-specific memory handling. These compiler directives were tuned to maximise floating-point performance and efficient hybrid MPI/OpenMP execution on Broadwell hardware.

The key optimizations target the specific performance bottlenecks in the NCUM model. Vectorization using the -hvector3 flag addresses the extensive loops present in NCUM's atmospheric dynamics and physics routines, which are suited for SIMD parallelism. Level 3 vectorization maximizes computational throughput on Broadwell's AVX2 units (256-bit vectors), which is crucial for accelerating the model's grid-point computations. Cache optimization is enabled using the -hcache3 flag exploits the strong data locality in NCUM's halo exchange patterns and vertical column calculations. Level 3 cache optimizations improve L1/L2/L3 hit rates, reducing memory bandwidth pressure on the dual-socket NUMA architecture. Additionally, OpenMP support, enabled by using the -h omp flag, facilitates hybrid MPI+OpenMP parallelization, which is essential for MIHIR's multicore nodes. OpenMP directives enable thread-level parallelism within MPI domains, reducing MPI rank count while maintaining computational efficiency (12 MPI × 3 OpenMP threads per node).

Additional flags enhance floating-point efficiency (-hfp4), loop unrolling (-hunroll2), and scalar optimizations (-hscalar3). The complete compilation command is used, and the details are put in Table 3:

TABLE 3

NCUM compiler specifications

Flag	Description
-e m	Generates code optimised for the designated MIHIR Broadwell target architecture.
-s default64	Sets the default pointer size to 64 bits, ensuring correct addressing on the system.
-h flex_mp=default	Applies the Cray default memory model for parallel compilation in hybrid MPI/OpenMP.
-h omp	Enables OpenMP directives for shared-memory parallel execution within each MPI rank.
-h fp4	Improves the efficiency of floating-point operations while preserving numerical robustness.
-hvector3	Activates high-level SIMD vectorisation, optimising compute-intensive loops.
-hshortcircuit2	Enables level-2 short-circuit evaluation of logical expressions to skip unnecessary work.
-hscalar3	Enables level-3 scalar optimisations such as scalar replacement and constant propagation.
-O pattern	Detects and optimises common code patterns to improve instruction scheduling and throughput.
-hcache3	Enables level-3 cache optimisation, improving data locality and cache usage.
-O zeroinc	Optimises increment operations (e.g. loop counters) to near zero-cost where possible.
-hunroll2	Unrolls loops to reduce loop-control overhead and expose further optimisation opportunities.

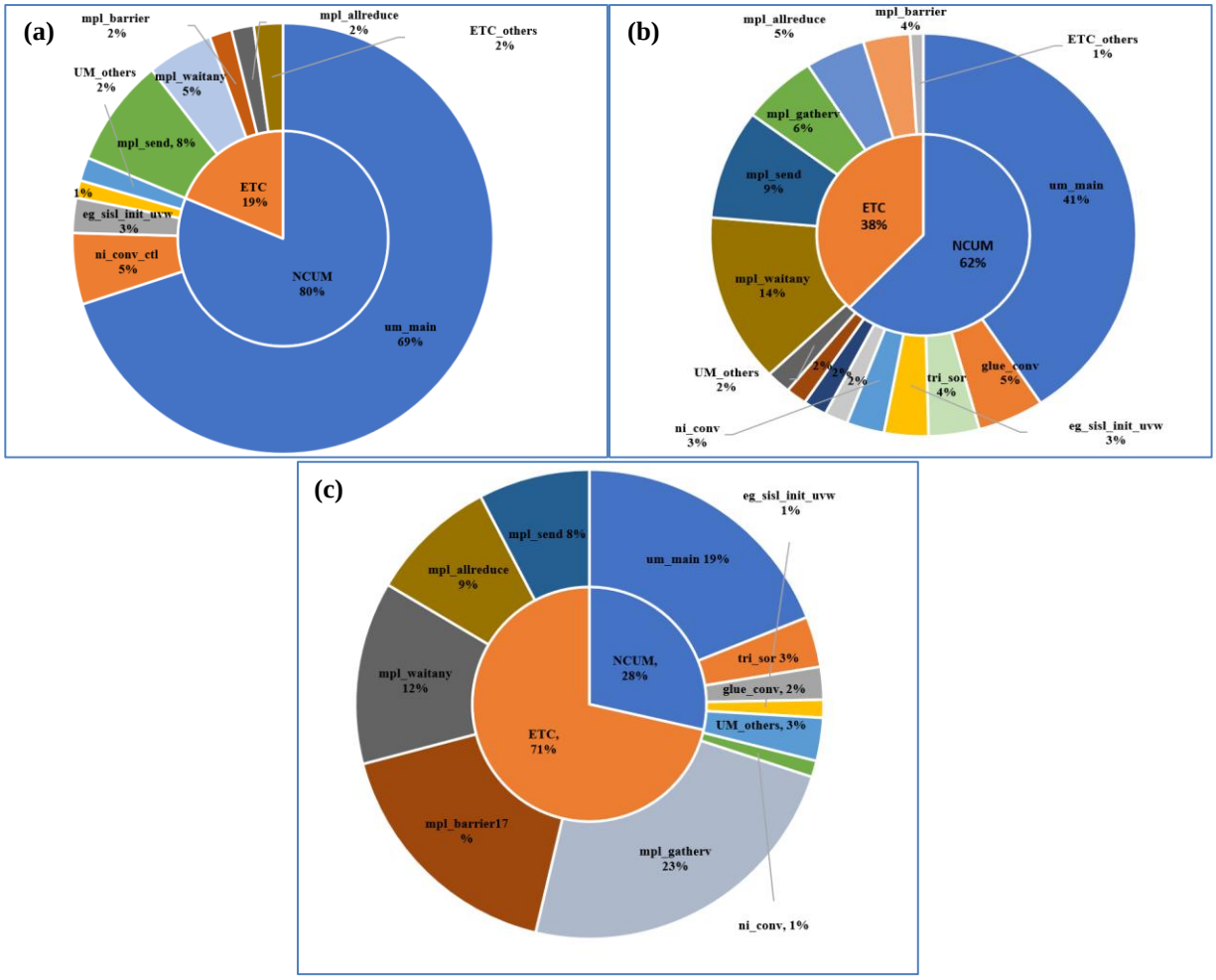

Fig. 1(a-c). (a) CrayPAT analysis for n96e: single node (b) CrayPAT analysis for n1280e: 36 nodes (c) CrayPAT analysis for n1280e: 288 nodes

2.4. *Performance metrics*

The performance metrics derived from CrayPAT include Total execution time (T_total) which is wall-clock time for complete model simulation, communication time (T_comm) that is the time spent in MPI communication routines, user execution time (T_user) which is the time spent in NCUM computation routines and load imbalance which is derived from CrayPAT reports and in percent gives the idle time in parallel routine relative to active processing elements. The MPI overhead in percent is calculated as follows:

$$MPI\ overhead(\%) = \left(\frac{T\_comm}{T\_total} x100\right) \qquad (1)$$

2.5. *Profiling methodology*

NCUM was profiled using CrayPAT in Automated Program Analysis (APA) (Cray Inc., 2024) mode. Each configuration was run as a continuous five-day simulation, and the wall-clock time was recorded. A baseline n96e single-node case was compared against n1280e runs on 36 and 288 nodes to assess resolution and scaling effects. Metrics capture both code-level routine contributions and system-level communication patterns.

3. Results and discussion

3.1. *NCUM performance analysis*

The compute nodes on MIHIR HPC operate on SUSE Linux Enterprise Server 12 SP2 and are configured with the Cray MPI library, which is based on the MPICH3, along with Cray's LibSci scientific library. The NCUM codebase incorporates updated components including the Joint UK Land Environment Simulator (JULES), the Suite of Community Radiative Transfer codes based on Edwards and Slingo (SOCRATES), the Cloud Aerosol Interacting Microphysics scheme (CASIM), and the Shared UM Library (SHUMLIB).

TABLE 4

Top routines by fraction of total runtime

Routine	n96e (1 node) %	n1280e (36 nodes) %	n1280e (288 nodes) %
UM total	**80.40**	**61.80**	**28.20**
um_main	69.40	41.20	18.50
glue_conv_6a	<1	5.10	2.20
tri_sor_sp_sp	<1	3.90	3.40
eg_sisl_init_uvw	2.60	3.40	1.20
ni_conv_ctl	5.30	2.90	1.10
ETC total	**18.70**	**37.60**	**70.70**
mpl_waitany	5.00	13.70	12.30
mpl_send	8.10	8.90	7.50
mpl_gatherv	<1	5.80	23.10
mpl_allreduce	1.70	4.60	8.60
mpl_barrier	1.70	3.60	16.80

The General Communications (GCOM) library (version 7.8 compiled with -O3), supplied with NCUM source code, interfaces NCUM with Cray MPI communication libraries. The NCUM codebase consists primarily of Fortran 90 (with a substantial amount of legacy Fortran 77), supplemented by C preprocessor (CPP) directives for shared blocks and parameter lists. The codebase, comprising roughly 900,000 lines predominantly in legacy Fortran 77 with preprocessor directives for shared blocks and parameter lists, incorporates this structure as a core execution sequence.

NCUM's MPI domain decomposition divides the computational grid along east–west (longitude) and north–south (latitude) directions according to the specified number of processes. Iterative solvers are employed to resolve the Helmholtz equation, with each iteration requiring halo exchanges between neighbouring MPI processes. The extent of the halo is determined by the interpolation order used in semi-Lagrangian advection and the maximum wind speed in the east–west direction. For high-resolution configurations, an extended halo of up to 10 grid points is used, which in turn limits the maximum number of MPI processes allowable in each direction.

3.2. *Profile distribution results*

Performance profiling and runtime analysis using CrayPAT (Cray Inc., 2024) for the n96e (single node), n1280e (36 nodes), and n1280e (288 nodes) resolutions identified the major functions executed and their corresponding time contributions. For reference, an n96e configuration running on a single node was taken as the baseline case, and its profile was compared against high-resolution n1280e (10 km) runs on 36 and 288 nodes. This comparison highlighted how the execution profile shifts with increasing resolution and scaling and helped pinpoint the specific routines and communication patterns that change in performance behaviour as the model scales from 36 to 288 nodes. The CrayPAT recorded the time spent in model functions and MPI calls. In all cases, the runtime was evenly distributed among multiple routines.

Fig. 1(a-c) below presents the CrayPAT performance analysis for the NCUM model across different configurations. These figures highlight key scalability trends and optimization impacts as the computational domain scales from small to large core counts.

In the profile visualisations provided in Fig. 1(a-c) with different model resolution and nodes, the central section of each pie chart summarises the overall runtime distribution, while the outer ring provides a breakdown of the contributions from individual functions and procedures. The segments labelled NCUM in the inner ring correspond to user-level NCUM routines (*e.g.*, *um_main*, *tri_sor*), whereas the ETC (External Time Consumption, the profiling label used by CrayPAT) segments represent time spent in MPI communication calls (such as *mpi_barrier*, *mpi_send*). The categories *NCUM_Others* and ETC_Others group together all other instrumented routines within the NCUM and MPI libraries that individually account for only a small fraction of the total execution time. Each profile shown is based on a continuous five-day model run.

Analysis of Fig. 1(a–c) indicates that execution time in NCUM is distributed evenly among different functions. Apart from *um_main*, no single routine contributes more than 10% of the total runtime, indicating the model exhibits a flat profile as shown in Table 4 (Sivalingam *et al*., 2015). There is no single routine that dominates, suggesting that improving performance would require optimising multiple areas rather than addressing one major bottleneck. For the n96e case, ni_conv_ctl routine which serves as the control driver routine for

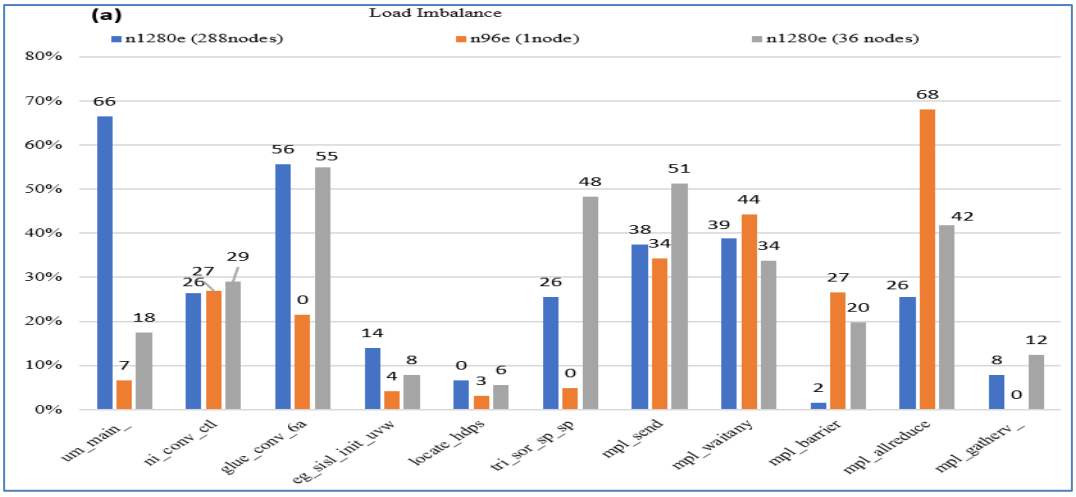

Fig. 2(a). Imbalance percentage of NCUM jobs: n96e job running on a single node; n1280e job running on 36 nodes and 288 nodes

NCUM's convection scheme and manages convection parameterization in NCUM's physics package, emerges as the largest contributor at around 5 % of execution time; for the n1280e configuration on 36 nodes, glue_conv which links the convection scheme back into the model state by transforming its tendencies into increments for prognostic variables (T, q, u, v), applying vertical limits and boundary conditions, enforcing conservation of mass, energy, and moisture, and passing updated fields to the physics or dynamics, also accounts for roughly 5 % of runtime and for n1280e configuration on 288 nodes, *tri_sor* which is a numerical solver for tridiagonal systems using SOR, widely used in NCUM's vertical implicit solvers, contributes about 3 %. Therefore, the functions that consume the most time change with both resolution and node count, indicating that performance-critical routines vary depending on the model configuration.

The distribution of computational work is spread across many functions and procedures, without being dominated by a small number of high-cost routines. This pattern is typical of what is referred to as a flat profile, where no single procedure is responsible for most of the runtime. In such cases, meaningful performance gains are likely to come from incremental optimisation across several components rather than from tuning a single dominant routine.

3.3. *Load imbalance analysis*

The CrayPAT reports provide detailed load-imbalance statistics for each processing element (PE) and thread, allowing examination of how evenly work is distributed across computational resources for different routines. In these reports, imbalance is expressed as a percentage relative to the active set of PEs or threads. This percentage quantifies the proportion of potential parallel resources that remain idle, *i.e.*, the fraction of time certain PEs or threads are not performing useful work for the function being measured. A perfectly balanced implementation would show an imbalance of 0 %, whereas a fully serial section would report 100 %.

As an illustration, if NCUM is executed with two OpenMP threads and an imbalance of 50 % is reported, it implies that one thread is performing all the computation while the other remains idle for half of the execution time of that function.

Fig. 2(a) presents the imbalance percentages for the NCUM runs at different resolutions and different node counts, n96e on a single node, and n1280e on 36 and 288 nodes.

Imbalance time is a valuable performance indicator derived from an application's execution profile. Analysis with CrayPAT revealed that load imbalance tends to increase when the model is run at higher resolutions and on fewer nodes. A key contributor to the imbalance load is mpl_allreduce, a collective routine. It synchronizes and reduces data across all processes. The CrayPAT recorded around 68% of imbalance time for mpl_allreduce in the n96e single-node configuration, 42% for the n1280e configuration on 36 nodes, and 26% for n1280e on 288 nodes, emphasizing that synchronization and collective communication become more inefficient at higher resolutions and lower node counts

For point-to-point communications performed by *mpl_send*, the imbalance remains high across configurations, with values of 34% in n96e on one node, 38% in n1280e on 288 nodes, & peaking at 51% in

TABLE 5

Impact of Halo Size and Decomposition on Load Imbalance Time % for n96e

UM/ETC routines	Decomposition (4x8) Halo EW/NS (10) Imbalance Time%	Decomposition (3x12) Halo EW/NS (5) Imbalance Time%
um_main	15.20%	3.80%
ni_conv_ctl	30.80%	28.10%
eg_sisl_init_uvw	14.90%	4.70%
locate_hdps	6.10%	2.50%
mpl_send	35.60%	35.80%
mpl_waitany	52.40%	40.40%
mpl_allreduce_	69.70%	56.80%
mpl_barrier_	26.90%	7.20%

TABLE 6

Impact of Halo Size on Load Imbalance Time % for n96e with 3x12 Decomposition

UM/ETC routines	Decomposition (3x12) Halo EW/NS (10) Imbalance Time%	Decomposition (3x12) Halo EW/NS (5) Imbalance Time%
um_main	6.60%	3.80%
ni_conv	27.00%	28.10%
eg_sisl_init_uvw	4.10%	4.70%
locate_hdps	3.10%	2.50%
mpl_send	35.80%	34.20%
mpl_waitany	44.20%	40.40%
mpl_allreduce	68.10%	56.80%
mpl_barrier	26.50%	7.20%

n1280e on 36 nodes. This indicates that, as the number of nodes is reduced for high-resolution jobs, message-passing between nodes becomes more costly & less balanced, which can limit overall efficiency. On a single node, *mpl_send* can still exhibit high imbalance because different MPI ranks do not all spend the same amount of time in mpl_send, even though the communication is intra-node. In the n96e single-node case, a 34% imbalance for mpl_send typically means that some ranks issue more sends, send larger messages, or reach the send points earlier, so they spend more time inside *mpl_send* while others are computing. The shared-memory MPI implementation may serialize access to internal buffers, so ranks with heavier communication patterns experience longer send times, which CrayPAT reports as load imbalance in *mpl_send*. A similar effect appears in *mpl_waitany*, where processes wait for incoming messages. Here, idle time increases when processes are blocked waiting for data exchanges. Imbalance values are 44 % for n96e on one node, 39 % for n1280e on 288 nodes, and 34 % for n1280e on 36 nodes, suggesting that message arrival delays contribute significantly to idle time especially at higher resolutions.

For synchronization barriers (*mpl_barrier*), the imbalance is more moderate, around 27 % for n96e, and it decreases as the number of participating nodes increases. while the routine mpl_gatherv exhibits no measurable imbalance, i.e., perfect balance, indicating all processes participate equally in data gathering, hence no detectable imbalance. It remains close to balance in all configurations, indicating minimal idle time and relatively efficient participation of all processes in this data-gathering phase. This trend highlights that the inefficiency in collective operations becomes more pronounced as model resolution increases and the job runs on fewer nodes, due to increased synchronization and communication delays.

3.4. *Halo size optimization*

The halo region is also a key factor in parallel performance because it governs the amount of data exchanged between neighboring processes. Using larger halo extents of 10, in both the extended_halo_size_ew (east–west) and extended_halo_size_ns (north–south) directions can influence the performance in several ways. A larger halo requires a higher communication volume, as more boundary data must be exchanged at each timestep, which increases execution time in communication -intensive routines such as *mpl_send*, *mpl_waitany*, and *mpl_allreduce*. Additionally, the memory footprint increases because each MPI rank must store larger ghost regions, resulting in less memory available for other

TABLE 7

Impact of Halo Size on Load Imbalance Time % for n96e with 4x8 Decomposition

UM/ETC routines	Decomposition (4x8) Halo EW/NS (10) Imbalance Time%	Decomposition (4x8) Halo EW/NS (5) Imbalance Time%
um_main	15.20%	14.90%
ni_conv	30.80%	33.00%
eg_sisl_init_uvw	14.90%	15.40%
locate_hdps	6.10%	4.10%
mpl_send	35.60%	50.60%
mpl_waitany	52.40%	38.10%
mpl_allreduce	69.70%	68.10%
mpl_barrier	26.90%	26.20%

computations. Larger halos can also contribute to greater load imbalance, since variability in send and receive times across processes may increase idle waiting during synchronisation, thereby reducing overall efficiency.

Communication overhead for each experiment is quantified as the fraction of total runtime spent in MPI routines. It is computed using

$$Communication\ Overhead\ (\%) = \frac{Total\ Time\ Spent\ in\ MPI\ Routines}{Total\ Execution\ Time} \times 100 \quad (2)$$

Hence, higher values indicate a larger proportion of time dominated by communication rather than useful computation, and therefore poorer parallel efficiency.

The most substantial benefit is obtained when halo reduction is combined with a more favourable domain decomposition. Table 5 shows that switching from a 4×8 decomposition with halo size 10 to a 3×12 decomposition with halo size 5 in the n96e configuration markedly improves load balance across both user-level and communication routines. The imbalance in um_main decreases from 15.2% to 3.8%, while ni_conv_ctl drops from 30.8% to 28.1% and eg_sisl_init_uvw decreases from 14.9% to 4.7%. Additionally, communication routines such as *mpl_waitany*, *mpl_allreduce*, and *mpl_barrier* also exhibit reductions in imbalance. Thus, the 3×12/halo-5 setup consistently achieves a total runtime of 105.86 s, with a communication overhead of ~19.1% of the execution time, indicating a more efficient and scalable configuration.

Total MPI time is 20.18s and total execution time is 105.86 s, so

Communication overhead (%) =

$$\frac{20.18}{105.86} \times 100 \approx 19.1\% \quad (3)$$

Taken together, as shown in Table 6 the 3×12 decomposition offers superior parallel performance by better managing halo boundary data exchanges and reducing communication-induced idle times, making it more suitable for large-scale, high-resolution NCUM simulations demanding frequent boundary communication. In the 3×12 configuration with halo size 5, the total execution time is 105.86 s, of which 84.66 s (about 80 %) is spent in user mode, with *um_main* alone taking 72.97 s (approximately 68.9 % of the user time). Communication routines collectively consume 20.18s (approximately 19.1% of the total runtime), and the load imbalance in um_main is limited to 3.8%, indicating an efficient distribution of computational work across MPI tasks. In the communication phase, *mpl_send* takes 7.28 s (35.8 % of communication time), *mpl_waitany*, 5.81 s (40.4 %), *mpl_allreduce*, 2.7 s (associated with a 56.8 % imbalance), and *mpl_barrier* 1.72 s (7.2 % imbalance), reflecting a well-balanced system.

When compared with the corresponding setup using halo size 10, the benefits of the reduced halo become more evident. The total execution time increases by about 10.04 s, from 105.86 s to 116.90 s, and user execution time grows by roughly 3.03 s, 84.66 s to 87.68 s, with *um_main* rising by approximately 0.5 s to 75.47 s. More importantly, communication time increases markedly by about 8.09 s to 28.27 s, and the imbalance in *um_main* worsens from 3.8 % to 15.2 %, This shows that larger halos increase communication overhead and load imbalance. while 3×12 decomposition with halo size 5 provides a more scalable and efficient operating setup for NCUM at this resolution.

In contrast, the 4×8 decomposition shows higher load imbalances, especially in communication intensive routines like *mpl_waitany* that exhibits a 52.4% imbalance and *mpl_allreduce* 69.7%. These increased imbalances point toward difficulties in balancing workloads and synchronizing inter-process communication within

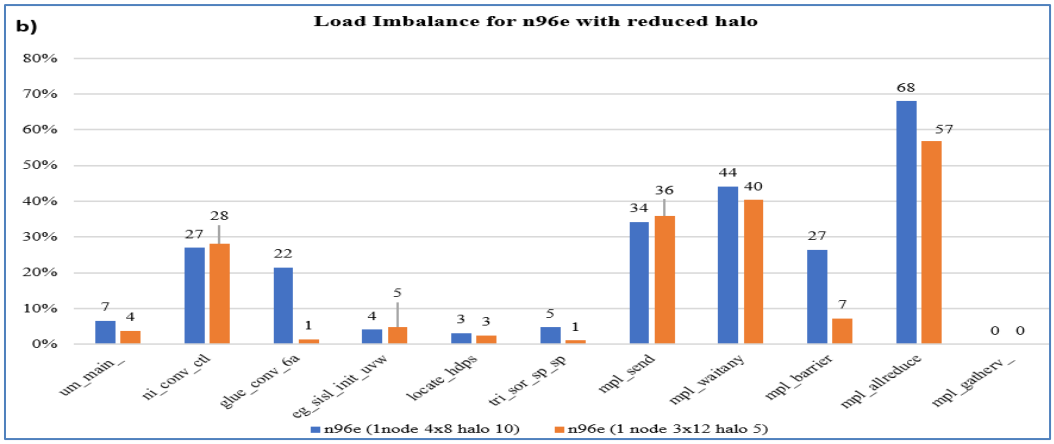

Fig. 2(b). Imbalance percentage of NCUM jobs: n96e job running on a single node with halo 10 and halo 5 values

TABLE 8

Routine-Level Load Imbalance Improvement with 3×12 (Halo 5) Configuration

Routine	4×8, Halo 5	3×12, Halo 5	Improvement
um_main imbalance	14.90%	3.8%	11.4 pp
ni_conv_ctl imbalance	33.00%	28.1%	2.7 pp
eg_sisl_init_uvw imbalance	15.40%	4.7%	10.2 pp
locate_hdps imbalance	4.10%	2.5%	3.6 pp
mpl_send imbalance	50.60%	35.8%	14.88 pp

smaller, more fragmented domain partitions. The increased number of partitions in the 4×8 setup may also lead to increase synchronization delays, thereby worsening data exchange efficiency.

Table 7 summarizes the effect of halo width for a fixed 4×8 decomposition. Reducing the extended halo size from 10 to 5 grid points for n96e leads to a modest runtime decrease from 116.90 s to 114.68 s (about 2 s over a five-day integration) and a small reduction in communication overhead from 24.2% to 23.2%, with modest gains in load balance for routines such as *um_main* and *mpl_waitany*.

For halo size 10, total runtime = 116.90 second MPI communication ≈ 28.27 second

$$\frac{28.27}{116.90} \times 100 \approx 24.2\% \quad (4)$$

For halo size 5, total runtime = 114.68 second

MPI communication ≈ 26.63 second

$$\frac{26.63}{114.68} \times 100 \approx 23.2\% \quad (5)$$

These results suggest that the 3×12 decomposition, combined with smaller halo widths, yields a more balanced load distribution across computational and communication routines, thereby contributing to improved parallel efficiency.

Fig. 2(b) illustrates the comparative load imbalances between the 3×12 and 4×8 decompositions with halo sizes of 5 and 10, respectively. The data clearly shows that the 3×12 decomposition with reduced halo sizes achieves substantially lower imbalance levels, particularly in UM/ETC routines such as *um_main* and *ni_conv_ctl*, whose imbalance times decrease from 15.2% to 3.8% and 30.8% to 28.1%, respectively.

Thus, the study shows that the halo region plays an important role in parallel computing performance, by controlling the exchange of boundary data between neighbouring MPI processes. While the larger 3×12 decomposition also increases inter-node communication, but it still demonstrates better overall balance in communication routines including *mpl_send*, *mpl_waitany*, and *mpl_allreduce*. Reduced imbalance in these functions is important for scaling in HPC environments, where excessive communication overhead can limit performance gains.

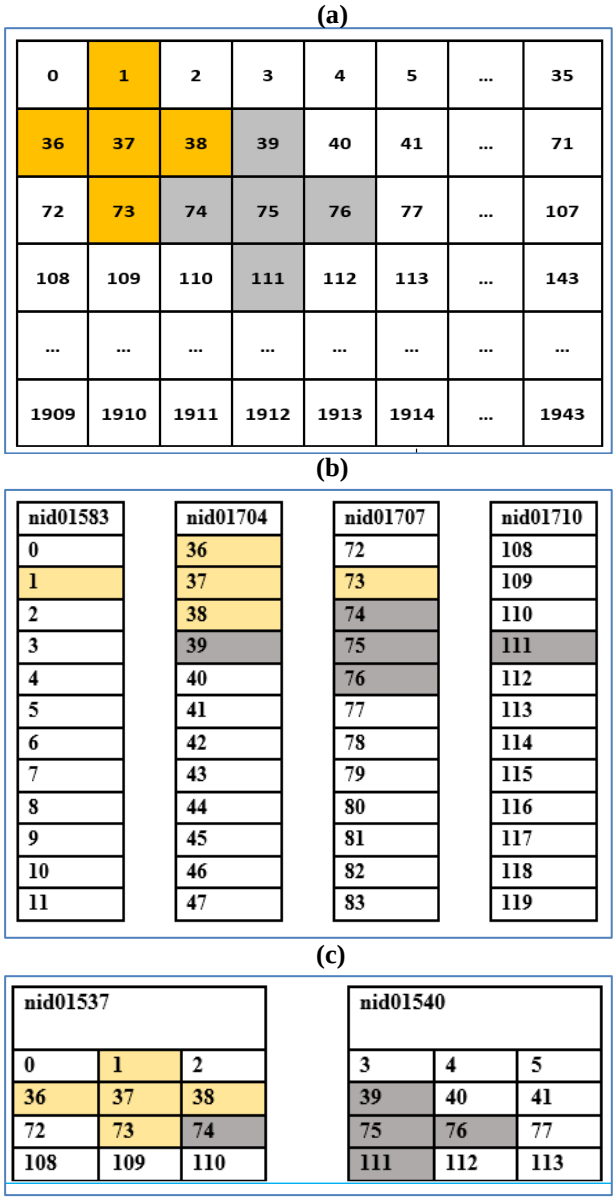

Fig. 3(a-c). (a) MPI ranks - PE configuration 36 x 54 (b) MPI ranks SMP style (c). MPI ranks GRID style

The extended halo size imposes limits on the number of MPI tasks due to increased memory use per task and extra communication overhead. This highlights the need of optimising thread-level parallelism and MPI task configuration to improve scalability and performance for high-resolution models.

3.5. *Impact of domain decomposition*

Tables 5–8 compare the 4×8 decomposition (32 MPI processes) with the 3×12 decomposition (36 MPI processes) for the n96e configuration under different halo settings and summarize the resulting routine-level imbalance and communication costs.

The 3×12 decomposition (halo 5) reduces imbalance in key user-level and collective routines. In particular, *mpl_barrier* imbalance decreases from 26.9% to 7.2% (a reduction of 19.7 percentage points; Table 5), indicating improved synchronization efficiency. Thus, the total execution time for 3x12 decomposition is 105.86 s, compared with 114.68 s for the 4×8 decomposition with halo size 5, giving a runtime reduction of about 9 s, or roughly 8%. User execution time is 84.66 seconds, which corresponds to approximately 80% of the total, while communication accounts for 20.18 seconds, or about 19% of the runtime. Relative to the 4×8, halo-5 case, this represents a reduction in communication time of approximately 8.1 s, corresponding to an improvement of about 28.6%. The imbalance in *um_main* decreases from 14.9% to 3.8%, a reduction of about 11.1 percentage points, proving that 3×12, halo-5 configuration runs more balanced and communication efficient.

3.6. *NUMA performance considerations*

The MIHIR HPC uses dual-socket compute nodes with each socket providing an 18-core NUMA nodes. Each NUMA node has 64 GB of dedicated DDR4 memory attached. In this architecture, the best combination for the HPC is 12 MPI + 3 OpenMP. The 12 MPI processes efficiently map to two NUMA nodes, and 3 OpenMP threads per MPI process help keep most data within the local memory of each NUMA node. This configuration minimizes the cost involved in remote NUMA memory accesses for n96e (single-node case), thus MPI overhead comes around 19%. For n1280e, overhead increases to 28.7% (288 nodes) and 37.6% (36 nodes), that clearly indicates inter-node communication has been the primary bottleneck at higher resolutions.

3.7. *MPI rank reordering optimization*

MPI rank reordering (Mercier *et al*., 2011) plays an important role in improving communication efficiency and scaling performance, especially for high-resolution runs across multiple nodes of HPC systems like MIHIR. The NCUM n96e configuration runs on a single compute node; thus, there is no inter-node communication. In contrast, the n1280e runs across 36 nodes, introducing extensive inter-node message passing, which is far more expensive than intra-node communication. Therefore, arranging MPI ranks carefully becomes crucial for good scaling.

Profiling with the CrayPAT shows an increase in MPI overhead from 19.06% in the single node n96e case to 28.7% and 37.59% in the n1280e runs on 288 and 36 nodes, respectively. When scaling the n1280e job from 36 to 288 nodes, the MPI overhead decreases by about

TABLE 9

MPI Rank Placement for NCUM Jobs (36 x 54 PE) and % increase in Execution Time on MIHIR using SMP and GRID Method (MPICH_RANK_REORDER=3)

EW	NS	node	n_{core}	T_{model} (Rank Reorder)	T_{hour} (SMP)	Percentage reduction in execution time
12	30	33	1188	14856.76	14921.61	0.43%
12	42	45	1620	10778.55	11087.38	2.8%
24	42	87	3132	5650.33	6021.23	6.2%
24	54	111	3996	4133.56	4462.65	7.4%
36	54	165	5940	3251.8	3729.41	12.8%

9.42%, and the consumed CPU time is reduced nearly threefold, indicating that inter-node message passing is the primary bottleneck for scaling in high-resolution simulations.

CrayPAT can analyse the MPI communication topology and suggest alternative rank orderings to reduce off-node messaging and improve MPI bandwidth (Cray Inc., 2024). On the MIHIR system, MPI rank reordering can be enabled by setting the environment variable MPICH_RANK_REORDER_METHOD to 3. When this method is selected, a custom rank order is specified in the MPICH_RANK_ORDER file; the default setting for this variable is 1, which corresponds to the symmetric multiprocessing (SMP) style rank placement.

For the n1280e job configured with 36 (east-west) by 54 (north-south) processor elements (PEs), CrayPAT recommends a rank order generated by the *grid_order* utility, which clusters nearest-neighbour ranks onto the same node to minimize expensive off-node communication. The command used is:

$$grid\_order - R - Z - c\ 12{,}3 - g\ 54{,}36 \qquad (6)$$

This GRID rank reordering helps effectively place processes such that it reduces inter-node communication compared to SMP (Symmetric Multiprocessing) placement, when multiple nodes are involved. For instance, under the SMP, the rank placement method is illustrated in Fig. 3(b), where rank 37 is assigned to compute node nid01704, rank 1 to nid01583, and rank 73 to nid01707, respectively. Consequently, message passing between these nodes involves inter-node communication, resulting in a higher communication cost. With the GRID style rank reordering method, the nearest neighbours of rank 37, including 1 and 73, are on the same node, which roughly halves the inter-node communication among these nodes.

Table 9 compares execution times for NCUM jobs under SMP style and GRID style rank reordering techniques, showing that for all test cases, the GRID style rank reordering shortens model runtime relative to the SMP style, with percentage reductions in execution time ranging from 0.43% to 12.8%. This clearly reflects differences in processor allocation and communication efficiency. Overall, the MPI rank reordering techniques prove how thoughtfully the MPI rank placement strategy can reduce communication overhead and improve the scalability of high-resolution NCUM simulations on MIHIR.

4. Conclusions

This study evaluated the performance of NCUM on the MIHIR Cray XC40 at two horizontal resolutions (n96e and n1280e) using CrayPAT for detailed profiling analysis. The analysis focused on identifying the main factors like load imbalance and MPI communication overhead that limit scalability as resolution increases and node counts vary.

The profiling reveals that load imbalance in collective MPI operations, and overall MPI communication overhead are the main performance bottlenecks. Collective routines show an imbalance level of up to 69.7%, while MPI communication accounts for up to 37.6% of the total runtime. The imbalance increases sharply from 0% in balanced n96e runs to approximately 68% for n96e on a single node and 51% for n1280e on 36 nodes, highlighting the need for improved communication and workload balancing, especially at higher resolutions and with fewer nodes.

Halo exchange configuration also has an impact on model runtime performance. Reducing the extended halo size from 10 to 5 grid points in both east–west and north–south directions improve runtime by about 8.9% (from 116.90 s to 105.86 s) for the 3×12 decomposition. This change lowers the imbalance in *um_main* from 15.2 % to 3.8 % and reduces communication time from 28.27 seconds to 20.18 seconds, mainly through improved behaviour in routines such as *mpl_waitany*. Although a halo size of 2 produced comparable runtimes, it introduced slightly higher imbalance and message-passing delays, so the halo-5 configuration provides the best overall trade-off.

Domain decomposition also plays a key role. The 3×12 layout with halo size 5 outperforms the baseline 4×8 configuration, reducing *um_main* imbalance from 15.2% to 3.8% and *mpl_barrier* imbalance from 26.9% to 7.2%, resulting in an overall runtime improvement of approximately 8.9%.

MPI rank reordering offers an additional optimisation level. A GRID-style rank placement applied to the n1280e configuration achieved up to 12.8 % faster execution by reducing expensive off-node communications. Further gains are attainable by parallelising additional routines (e.g., *glue_conv*, *tri_sor*) with OpenMP to lower thread-level imbalance within nodes.

Taken together, halo size optimisation, tuned domain decomposition, strategic MPI rank placement, and targeted OpenMP parallelisation constitute an effective pathway toward maximising NCUM scalability and reducing time to solution for high-resolution operational forecasts on MIHIR.

In the future, optimisation gains can be obtained through targeted code changes, such as glue_conv and tri_sor, which will be rewritten to utilise OpenMP more effectively. This should spread the work more evenly across threads and reduce waiting time within a node. High-resolution model optimization can be made possible by profiling the model. The best possible settings can be achieved by tuning halo size, selecting an optimal domain decomposition, analysing MPI imbalance, and reordering ranks. The same approach can be extended to new platforms, including GPU-based systems and future CPU clusters. The aim is to develop a set of simple, reusable guidelines that can help accelerate NCUM and other weather models as resolutions and ensemble sizes continue to increase. The techniques already integrated into NCMRWF operations have delivered consistent performance improvements for current configurations and provide a solid foundation for tuning forthcoming future high-resolution model versions and additional applications on forthcoming HPC systems.

Acknowledgements

The experiments described in this study were carried out on the MIHIR supercomputing system at the National Centre for Medium Range Weather Forecasting (NCMRWF). The authors sincerely thank their colleagues and the scientific staff at NCMRWF for their valuable assistance and guidance during the course of this work. We also extend our appreciation to the Head of NCMRWF for providing encouragement and support throughout the project.

Funding

The authors declare that no funds, grants, or other support were received during the preparation of this manuscript.

Data Availability

The weather data supporting the findings of this study is available upon request, and the datasets will be shared accordingly.

Conflicts of Interest

The authors have no relevant financial or non-financial, personal or other relationships with other people or organizations that could inappropriately influence, or be perceived to influence, their work.

Authors' Contributions

Shivali Gangwar: Installation, configuration, execution analysis, and Original manuscript writing. (*email: shivalig@ncmrwf.gov.in)*
Ashish Routray: Technical guidance, HPC infrastructure support, manuscript editing and reviewing. (*email: ashish.routray@gov.in)*
Suryakanti Dutta: Technical guidance, manuscript revision and review. (*email: suryakanti.dutta@gov.in*)
Preveen Kumar D.: Technical guidance, Facilitator. (*email:* preveen@ncmrwf.gov.in)
V. S. Prasad: Facilitator. (*email: ps.vijapurapu@nic.in)*
B. Athiyaman: Technical guidance, HPC infrastructure support, Facilitator. (*email: athiyab@gmail.com)*

Disclaimer: The contents and views presented in this research paper are the views of the authors and do not necessarily reflect the views of the organisations they belong to.

References

Bauer, P., Thorpe, A. & Brunet, G., 2015, "The quiet revolution of numerical weather prediction", *Nature*, **525**, 47–55. doi: https://doi.org/10.1038/nature14956.

Bermous, I., & Steinle, P., 2015, "Efficient performance of the Met Office Unified Model v8.2 on Intel Xeon partially used nodes", Geoscientific Model Development, **8**, 769–779. https://doi.org /10.5194/gmd-8-769-2015.

Chunduri, S., Parker, S., Balaji, P., Harms, K., & Kumaran, K., 2018, "Characterization of MPI Usage on a Production Supercomputer", 386–400 doi: https://doi.org/10.1109/SC. 2018.00033.

Cray Inc., 2024, "Cray Performance Analysis Tool (CrayPAT) User Guide", Cray Documentation. Available at: https://cpe.ext. hpe.com/docs/24.11/performance-tools/index. html.

George, J. P., Rani, S. I., Jayakumar, A., Mohandas, S., Mallick, S., Lodh, A., Rakhi, R. and Sreevathsa, M. N. R., 2016, *NCUM Data Assimilation System*, NCMRWF Technical Report, NMRF/TR/01/2016, National Centre for Medium Range Weather Forecasting (NCMRWF), Noida, India. Available at: ResearchGate version of NCUM Data Assimilation System.

Gangwar, S., Preveen Kumar D., & Athiyaman, B., 2023, "Scalability Analysis of the NCMRWF Unified Model (NCUM)", NCMRWF Technical Report nmrf.tr.7.2023, Available at: https://nwp.ncmrwf.gov.in/publications/tr/10.64349/nmrf.tr.7.20 23.

Kumar, S., Jayakumar, A., Bushair, M. T., & Rajagopal, E. N., 2018, "Implementation of New High Resolution NCUM Analysis-Forecast System in Mihir HPCS. NCMRWF Technical Report, NMRF/TR/01/2018, https://doi.org/10.64349/nmrf.tr.1.2018.

Mercier, G., & Jeannot, E., 2011, "Improving MPI Applications Performance on Multicore Clusters with Rank Reordering", Lecture Notes in Computer Science, 6960, 39–49. https://doi. org/10.1007/978-3-642-24449-0_7.

MoES, 2023, Annual Report 2022–23. Ministry of Earth Sciences, Government of India.

NCUM Global Model Verification: Monsoon (JJAS) 2019, 2020, 2021, 2022.

Prasad VS, Saji Mohandas, SK. Dutta, M. Das Gupta, GR. Iyengar EN, Rajagopal, Basu S., 2014, "Improvements in medium range weather forecasting system of India", *J Earth Syst Sci* **123**, 2, 247–258, https://doi.org/10.1007/s12040-014-0404-5.

Prasad, S.K. A. Sarkar, A. Mamgain and E. N. Rajagopal, 2019, "Implementation of NCMRWF Regional Ensemble Prediction System (NEPS-R), NCMRWF Technical Report, nmrf.tr.**9**.2019, https://doi.org/10.64349/nmrf.tr.9.2019 .

Rajagopal, E. N., Iyenger, G. R., Das Gupta, M., Mohandas, S., Siddharth, R., Gupta, A., Chourasia, M., Prasad, V. S., Sharma, K., and Ashish, A., 2012, "*Implementation of Unified Model Based Analysis-Forecast System at NCMRWF*. NMRF Technical Report, TR-2.https://doi.org/10.64349/nmrf.tr.2.2012

Rajagopal, E.N., Munmun Das Gupta, Saji Mohandas, V.S. Prasad, John P. George, G.R. Iyengar and D. Preveen Kumar, 2007, "Implementation of T254L64 Global Forecast System at NCMRWF", NCMRWF Technical Report, nmrf.tr.1.2007, National Centre for Medium Range Weather Forecasting (NCMRWF), Noida, India.

Rakhi, J.R., A. Jayakumar, M. N. R. Sreevasthsa and E. N. Rajagopal, et. al., 2016, "Implementation and Up-gradation of NCUM in Bhaskara HPC", Technical Report, **3**, 1-22, nmrf.tr.3.2016, https://doi.org/10.64349/nmrf.tr.3.2016 .

Routray, A., Devajyoti Dutta, and John P. George, 2019, "Evaluation of Track and Intensity Prediction of Tropical Cyclones Over North Indian Ocean Using NCUM Global Model, Pure Appl. Geophys. **176**, 421–440, https://doi.org/10.1007/s00024-018-1924-8.

Routray, A., Vivek Singh, Harvir Singh, Devajyoti Dutta, John P. George, R. Rakhi, 2017, "Evaluation of different versions of NCUM global model for simulation of track and intensity of tropical cyclones over Bay of Bengal, Dynamics of Atmospheres and Oceans", **78**, 71–88, https://doi.org/10.1016/ j.dynatmoce.2017.04.001.

Sarkar, A., Paromita Chakraborty, John P. George and E. N. Rajagopal, 2016, **"**Implementation of Unified Model based Ensemble Prediction System at NCMRWF (NEPS), NCMRWF Technical Report, nmrf.tr.2.2016, https://doi.org/10.64349/nmrf.tr.2.2016 .

Sivalingam, K., Lister, G., & Lawrence, B., 2015, "Performance analysis and Optimisation of the Met Unified Model on a Cray XC30", arXiv:1511.03885. https://doi.org/10.48550/arXiv.1511.03885.