Analyzing Communication Performance From Testbeds to HPC Systems: A Lab Course

Harasees Kaur, Sankalp Shashi, Umesh Konduru
Dept. of CSE, IIT Ropar
Rupnagar, India
{harasees.24csz0001, 2022csb1137, 2022csb1139}@iitrpr.ac.in

Sahil Kumar, Balwinder Sodhi, Jagpreet Singh
Dept. of CSE, IIT Ropar
Rupnagar, India
{sahil.22csz0018, sodhi, jagpreets}@iitrpr.ac.in

B.S. Sanjeev
Dept. of Applied Sciences, IIIT Allahabad
Prayagraj, India
sanjeev@iiita.ac.in

***Abstract*—High-Performance Computing (HPC) education is often limited by restricted access to large-scale systems, making it challenging for students to gain practical experience. To address this, we have developed a Low-cost Pi-based Instructional Testbed for Experimentation (LITE-Pi), that provides a hands-on environment for exploring HPC concepts such as parallel programming, communication performance, and scalability. Given that scalable performance depends not only on hardware but also on a deep understanding of parallel system behavior, it is essential to train students in practical aspects of HPC through accessible, small-scale platforms. Our experiments show that communication trends on LITE-Pi closely mirror those of a production HPC cluster, although optimal performance parameters differ, highlighting the need for scale-aware optimization. To extend this understanding, we integrate tools such as Netgauge, LLAMP, and LogGOPSim, enabling students to analyze and optimize performance at larger scales. Based on these insights, we propose a 14-week laboratory course focused on practical experimentation and optimization across system scales, helping students develop the analytical and technical skills required for real-world HPC environments. Upon successful completion of the course, students will acquire the expertise required to transfer the skills and insights gained from LITE-Pi to the production-scale system.**

***Index Terms*—HPC, Education, Netgauge, LogGOPSim.**

I. Introduction

High-Performance Computing (HPC) involves the execution of complex computations at extremely high speeds, playing a pivotal role in addressing major scientific and engineering challenges [1]. From exploring climate prediction and healthcare to enabling structural and aerodynamic designs, HPC continues to advance and is becoming increasingly widespread across multiple fields [2]. The significance of HPC is further underscored by market forecasts, which predict an annual growth rate of 8.2% over the next five years, driven by global investments in HPC infrastructure [3].

Although the areas of application for HPC are diverse, the extent of actual progress will depend on the availability and quality of skilled researchers. Despite huge advances in HPC, there is still a gap between HPC infrastructure and education. This calls for the necessity of structured educational programmes in HPC, accompanied by small-scale testbed clusters that can facilitate practical learning, as access to production-level systems remain restricted and is generally not suitable for instructional and experimental purposes.

While several recent studies have proposed strategies to bridge this gap, the majority have focused on application development and performance analysis. For instance, Aggarwal et al. [4] proposed a Raspberry Pi based cluster for training students in parallel and distributed computing. Their work focused on the development and scaling of parallel applications and scaling parallel applications from a small scale testbed to a petascale production system, including heat diffusion and pi estimation. While the focus of studies was on application driven scalability, communication specific overheads that fundamentally impact the performance of applications in distributed HPC systems also demands attention. Another work proposed by Zhang et al. [5] builds an inexpensive cluster with 5 Raspberry Pi 4 nodes for Parallel Computing Topology Optimization workloads. However, their approach remained narrowly focused on manufacturing optimization and did not extend to more general HPC applications or communications. Another cluster design proposed by Chen et al. [6], provides a low cost cluster of 81 Raspberry Pi 4 nodes for Numerical Weather Prediction which successfully completed high-resolution forecasts and reached over 200 GFLOPS in performance. However, this cluster was optimized only for a particular application. Other studies like Papakyriakou et al. [7], who utilized a cluster of 24 Raspberry Pi 4 nodes and Cloutier et al. [8], who employed a cluster of 25 Raspberry Pi 4 nodes, proposed a STREAM-based analysis of Raspberry Pi clusters. These studies demonstrated the educational potential of small-scale clusters by measuring bandwidth, execution time, scalability, and efficiency. However, further studies are required to implement significant optimizations in order to provide comparisons with production-scale HPC systems.

In this work, we discuss LITE-Pi, a Raspberry Pi–based cluster constructed to serve as an educational testbed for exploring high-performance computing concepts. We present the software environment of LITE-Pi and report the results of experiments conducted to analyze and compare the communication characteristics observed in our small-scale setup with those of an HPC cluster, Surya, at the Indian Institute of Information Technology, Allahabad (IIITA). To conduct this study, we first conducted experiments to measure how communication performance varies with message size on both

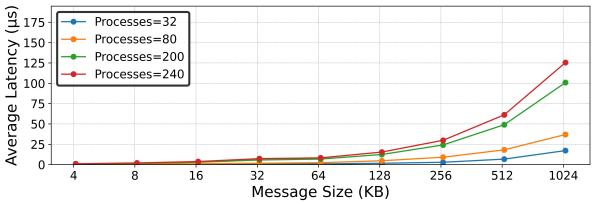

Fig. 1: Allgather Performance on Surya Cluster

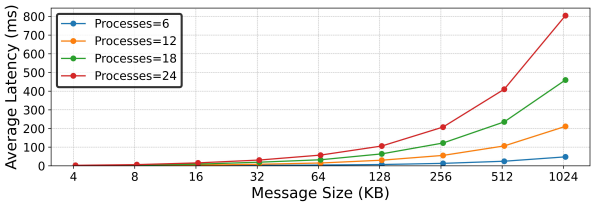

Fig. 2: Allgather Performance on LITE-Pi

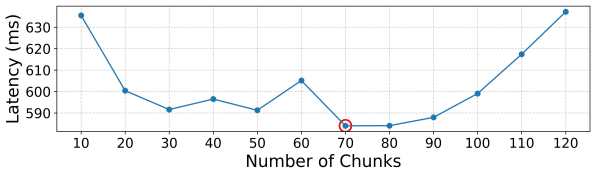

Fig. 3: Latency vs. Chunk Count for 8 MB Binary-Tree Broadcast on LITE-Pi

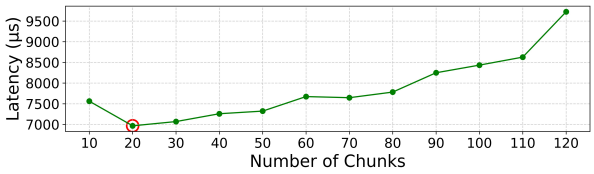

Fig. 4: Latency vs. Chunk Count for 8 MB Binary-Tree Broadcast on Surya

of the clusters. As shown in Figures 1 and 2, both Surya and LITE-Pi clusters display similar latency patterns, remaining low for smaller message sizes and rising sharply for larger ones due to increased communication overhead which shows the overall trends remain constant when we move from small cluster to the production cluster.

To further analyze the performance behavior, we implemented a broadcast topology and evaluated its performance by dividing each message into smaller parts to enable more efficient data transfer. These smaller parts, referred to as chunks, allow overlapping of communication and computation, thereby improving pipeline efficiency and network utilization. The optimal number of chunks required for LITE-Pi was observed to be greater than that of Surya. The results plotted in Figure 3 and 4 respectively, demonstrate that whereas small-scale systems are capable of replicating general communication patterns, their optimal performance parameters do not directly transfer to production scale systems.

These findings emphasize how crucial communication optimization and tuning are towards realization of good scaling in HPC applications. In this paper, we propose a lab course designed to help students understand and optimize the communication of applications, thereby optimizing performance, as they move from small-scale testbeds to large HPC systems. Since direct access to production clusters is often restricted, the LITE-Pi platform offers a useful foundation for performance prediction and experimentation when used in with tools like Netgauge [9], LLAMP [10], and LogGOPSim [11]. The course equips students with practical skills in analyzing communication patterns, evaluating scalability, and optimizing HPC applications, fostering a deeper understanding of performance tuning in real-world high-performance computing environments.

II. LITE-Pi Testbed: Architecture, Optimization Tools and Multicast Communication

A testbed that mimics the hardware and software architectures of modern HPC clusters is helpful for learning and experimenting from an educational perspective. Our LITE-Pi cluster shown in Figure 5 provides students with a compact, inexpensive, and energy efficient platform for testing distributed and parallel computing applications. This approach reduces the cost and resource demands of production clusters while providing practical exposure to real HPC techniques.

A. Cluster Architecture and Hardware Specification

The LITE-Pi cluster is designed with a centralized architecture, where a single master node manages job scheduling, resource allocation, and overall coordination, while six dedicated compute nodes are utilized for executing parallel workloads and running user applications. At present, the cluster comprises six compute nodes, but its design allows for easy scalability and extension to a larger number of nodes. Raspberry Pi 5 boards have been used with 16 GB RAM and an external 256 GB SSD. A dedicated 700 GB Network File System (NFS), which can ve extended, has been set up to manage shared storage, providing a unified workspace for all user files, application binaries, and datasets. Communication between nodes is handled by a 12-port Gigabit Cisco Catalyst 2960-CX network switch. Every node is assigned a static IP address to efficiently manage the network configuration. The master node boots up from the attached 32 GB micro SD card, whereas the worker nodes use the attached 256 GB external SSDs for booting up the OS. The nodes are loaded with the headless version of Raspbian 12 (bookworm) OS, and utilize OpenMPI for distributed computation with a SLURM setup for job scheduling [12].

Different approaches can be employed for shared storage on HPC cluster. In LITE-Pi, a centralized NFS share has been exported, where a single dedicated server provides a shared filesystem mounted by all cluster nodes. This design is simple to deploy, easy to back up and reproduce, making it well-suited for small educational testbeds. However, it also introduces a single point of failure and can become a

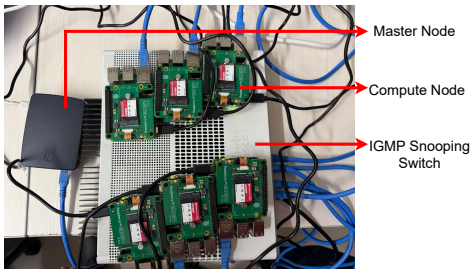

Fig. 5: LITE-Pi Cluster

network or disk I/O bottleneck under heavy workloads. An alternative is a distributed NFS design, where storage is spread across multiple machines or implemented using a distributed filesystem with an NFS frontend. This approach improves fault tolerance and throughput via replication or striping but adds complexity and overhead, making it better suited for large I/O-intensive clusters than small Raspberry Pi testbeds.

B. Performance Modeling and Optimization Toolchain

We equip LITE-Pi with research-grade tools so students can measure, analyze, and extrapolate communication behaviour. Netgauge [9] is a high-precision microbenchmark framework for measuring latency, bandwidth and other network parameters and for building statistically robust timing experiments. LLAMP [10] is an analytical toolchain that records MPI traces, converts them via the LogGP model into execution graphs, and uses linear programming to assess an application's latency sensitivity and to forecast performance under different latency scenarios. LogGOPSim [11] is a scalable LogGP-based simulator that can replay and extrapolate collective and point-to-point communication patterns to very large process counts, letting us reproduce and study scale-dependent behaviour that we observe on the LITE-Pi.

C. Leveraging Multicast for Collective Communication

A primary objective of this course is to equip students with the skills to analyze and optimize communication performance at the network level. Traditional MPI collectives like MPI_Bcast or MPI_Allreduce are often implemented using sequences of unicast messages, which can lead to significant network contention, which is a major performance bottleneck for large-scale systems. The LITE-Pi cluster supports multicast-based collectives for efficient one-to-many data delivery. Multicast-based collective operations can deliver substantially improved performance and lower latency compared to traditional unicast trees by minimizing network traffic and eliminating redundant data transmissions across the switch fabric [13]. Multicast groups are configured on the IGMP-snooping switch, ensuring that packets are forwarded exclusively to ports with active subscribers. This capability on the LITE-Pi allows students to experiment directly with advanced network optimization strategies that are often inaccessible on large production clusters. Figure 6 plots a comparative analysis of Latency against Message Size for

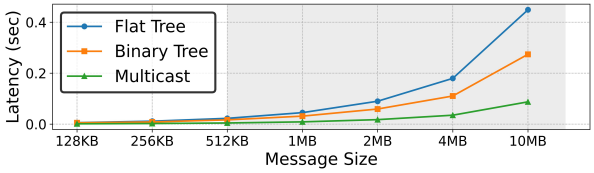

Fig. 6: Multicast vs. Broadcast on LITE-Pi for 60 KB chunks.

TABLE I: Comparison of LITE-Pi and Surya clusters

Component	LITE-Pi	Surya
No. of Compute Nodes	6	20
No. of cores per node	4	40
CPU Architecture	ARMv8-A	x86_64
Processor clock speed	2.4GHz	2.4GHz
RAM per node	16 GB	384 GB
Networking	1 Gigabit	100 Gigabit
Job Scheduler	Slurm	PBS
MPI	OpenMPI	Intel

three broadcast approaches: Flat Tree broadcast, Binary Tree broadcast, and Multicast-based broadcast. In multicast-based broadcast, the sender transmits a single message to the switch, which replicates and delivers it to all subscribed recipients, reducing network load and enhancing efficiency. As shown in Figure 6, Multicast-based broadcast typically results in lower latency and better performance which is even more evident for larger message sizes, as compared to both Flat Tree and Binary Tree broadcasts. This demonstrates the benefits of leveraging multicast for collective communication in large-scale distributed systems. However, since multicast is supported only over Unreliable Datagram (UD) transport, it does not guarantee reliable delivery and therefore requires additional mechanisms to ensure data consistency and fault tolerance.

III. From Testbed to a Production System

To evaluate the performance of our educational testbed, we performed benchmarking experiments on Surya. The comparison between both clusters on the basis of architecture and performance can be found in Table I. The architectural comparison gives a general overview, but in distributed systems communication efficiency has a direct impact on scalability. To test the communication, OSU micro benchmarks were run on both clusters which are used to quantify the latency on different communication patterns including collectives, by varying the message size. Figures 1 and 2 show the plots of average latency against message size for Allgather collective of OSU mirco benchmarks [14] for Surya and LITE-Pi respectively, for various number of processes. In both the plots, latency remains low for smaller message sizes and increases as the message size increases. Overall performance trends are comparable for both the cluters with LITE-Pi cluster showcasing a significantly higher latency. This shows that the trends in communication behaviour of large scale HPC cluster can be successfully replicated on even a small scale educational testbed, which makes it an invaluable tool for teaching and training in parallel computing.

To examine how application performance depends on the underlying network characteristics, we executed the broadcast collective using a pipelined binary tree topology with an identical logical configuration (24 processes, 4 processes per node across 6 nodes) on both the LITE-Pi testbed and Surya. Figures 3 and 4 show that both systems exhibit a characteristic U-shaped latency curve when varying the number of chunks; however, the optimal chunk size differs between them. This variation indicates that communication performance cannot be directly replicated across systems of different scales. Even with identical logical configurations, parameters such as the optimal chunk size must be carefully re-evaluated. Hence, applications and collectives often require system-specific tuning rather than a one-size-fits-all configuration when moving from small-scale clusters to large HPC systems.

To teach students how to analyse and optimise communication patterns in production clusters without exclusive access, we introduce performance analysis and simulation toolchains that let them measure, model, and simulate scale-dependent behaviour. The first step in the workflow discussed above is to measure the cluster's communication characteristics, which provide the foundation for accurate performance analysis. Network measurements collected using Netgauge [9] provide these essential parameters, such as latency and bandwidth used to characterise the target system's interconnect. The obtained parameters, together with the application traces, which can be collected using the libAllprof library, are then fed into analysis and simulation frameworks such as LLAMP [10] and LogGOPSim [11] for deeper exploration of communication behaviour and scalability. LLAMP is an analytical toolchain that takes MPI traces as input, constructs execution graphs from traces, and formulates linear programmes. The solutions of these linear program yield predicted runtimes, critical-path metrics, and explicit latency-sensitivity/tolerance values. Another tool employed for trace-based analysis is LogGOPSim, a fast, trace-driven simulator that uses MPI communication traces to simulate both point-to-point and collective operations, thereby enabling accurate prediction of algorithm runtime and performance behavior using execution traces.

Traces collected from small-scale experimental clusters like LITE-Pi can be leveraged to predict application performance on larger production systems. This approach enables students and researchers to analyze performance behavior, optimize and adapt code for different cluster environments, and ensure portability and scalability across diverse HPC platforms.

IV. Course Modules

Module 1: Introduction to HPC and Parallel Programming
Introduction to HPC Clusters, Setup of Raspberry Pi Cluster, Point-to-Point and Collective Communication in MPI, Basic Parallel Programming Exercises (e.g., Matrix Multiplication and Fast Fourier Transform).

Module 2: Collective Communication Algorithms
Introduction to LogGP Communication Model, Virtual Topologies for Collectives, Implementation and Comparative Analysis of Collectives under Different Topologies.

Module 3: Profiling and Benchmarking
Usage of Netgauge, OSU Benchmarks, HPL Benchmarks, NAS Parallel Benchmarks, Stream Benchmarks, etc.

Module 4: Analysis, Simulation, and Optimization
Trace Collection and Analysis using SST Dumpi and liballprof, Simulation Analysis using LogGOPSim and LLAMP Toolchain, Communication Optimization of HPC Applications for Scaling from Testbed Cluster to Production Scale Cluster.

Module 5: Advanced HPC Topics
Exposure to Recent Developments in HPC e.g., Multicast-based Collectives, Optimal Message Matching in MPI, HPC Training Pipelines for Machine Learning Workloads.

The detailed information regarding the Lab course can be accessed using [15].

References

[1] P. Bouvry, M. Brorsson, R. Canal, A. Eftekhari, S. Höfinger, D. Smets, H. Köstler, T. Kozubek, E. Krishnasamy, J. Llosa *et al.*, "The european master for hpc curriculum," *Journal of Parallel and Distributed Computing*, vol. 201, p. 105081, 2025.

[2] R. K. Raj, C. J. Romanowski, J. Impagliazzo, S. G. Aly, B. A. Becker, J. Chen, S. Ghafoor, N. Giacaman, S. I. Gordon, C. Izu *et al.*, "High performance computing education: Current challenges and future directions," in *Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education*, 2020, pp. 51–74.

[3] "ISC24 Presentations Hyperion Research hyperionresearch.com," https://hyperionresearch.com/hpc-market-update-briefing-during-isc24-virtual, [Accessed 13-10-2025].

[4] D. Aggarwal, A. Bokshi, and D. Lad, "A hands-on approach for scalable parallel applications development: From testbed to petascale," in *2022 IEEE 29th International Conference on High Performance Computing, Data and Analytics Workshop (HiPCW)*. IEEE, 2022, pp. 36–43.

[5] Z.-D. Zhang, D.-Y. Yu, O. Ibhadode, L. Meng, T. Gao, J.-H. Zhu, and W.-H. Zhang, "Topaddpi: An affordable and sustainable raspberry pi cluster for parallel-computing topology optimization," *Processes*, vol. 13, no. 3, p. 633, 2025.

[6] C.-Y. Chen, N.-C. Yeh, Y.-C. Chuang, and C.-Y. Lin, "Development of a low-cost portable cluster for numerical weather prediction," *Electronics*, vol. 11, no. 17, p. 2769, 2022.

[7] D. Papakyriakou and I. S. Barbounakis, "Performance analysis of raspberry pi 4b (8gb) beowulf cluster: Stream benchmarking," *International Journal of Computer Applications*, vol. 975, p. 8887.

[8] M. F. Cloutier, C. Paradis, and V. M. Weaver, "A raspberry pi cluster instrumented for fine-grained power measurement," *Electronics*, vol. 5, no. 4, p. 61, 2016.

[9] T. Hoefler, T. Mehlan, A. Lumsdaine, and W. Rehm, "Netgauge: A network performance measurement framework," in *International Conference on High Performance Computing and Communications*. Springer, 2007, pp. 659–671.

[10] S. Shen, L. Huang, M. Chrapek, T. Schneider, J. Dayal, M. Gajbe, R. Wisniewski, and T. Hoefler, "Llamp: Assessing network latency tolerance of hpc applications with linear programming," in *SC24: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 2024, pp. 1–18.

[11] T. Hoefler, T. Schneider, and A. Lumsdaine, "Loggopsim: simulating large-scale applications in the loggops model," in *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, 2010, pp. 597–604.

[12] G. Mills, "Building a Raspberry Pi Cluster glmdev.medium.com," https://glmdev.medium.com/building-a-Raspberry-pi-cluster-784f0df9afbd, [Accessed 13-10-2025].

[13] "Multicast in a Campus Network: CGMP and IGMP Snooping cisco.com," https://www.cisco.com/c/en/us/support/docs/switches/catalyst-6500-series-switches/10559-22.html, [Accessed 13-10-2025].

[14] https://ulhpc-tutorials.readthedocs.io/en/latest/parallel/mpi/%OSU_MicroBenchmarks/, [Accessed 13-10-2025].

[15] "GitHub haraseesKaur12/HPC-Lab-Course: LAB COURSE github.com," https://github.com/haraseesKaur12/HPC-Lab-Course, [Accessed 13-10-2025].