A Hands-on Approach for Scalable Parallel Applications Development: From Testbed to Petascale

Deepak Aggarwal
Computer Division
Institute for Plasma Research
Gandhinagar, India
deepakagg@ipr.res.in

Arkaprava Bokshi
York Plasma Institute
University of York
York, UK
arka.bokshi@york.ac.uk

Deep Lad
Department of Computer Engineering
Dharmsinh Desai University
Nadiad, India
deep2468r@gmail.com

***Abstract*—HPC systems are ubiquitous yet inaccessible to college/university students as most universities in India still lack the infrastructure, experience, and budget for setting up an HPC facility. Therefore, it is highly desirable for academia to use effective pedagogy to train students about HPC concepts and develop skills to write parallel applications. In this paper, we present the hands-on approach of learning cluster architecture and the development of applications that can interact with HPC systems in a Parallel and Distributed Computing (PDC) environment. The applications have been developed on a testbed cluster and successfully scaled on the Institute for Plasma Research (IPR) Petascale cluster named *ANTYA*. These applications have been used as a tool for HPC outreach activities at IPR for training young college students about PDC along with a hands-on understanding of cluster architecture by building testbeds from scratch.**

***Index Terms*—Pedagogy, HPC, Cluster Architecture, Parallel Applications, PDC, Scaling Performance**

I. Introduction

The growing presence and popularity of HPC have driven the discovery of new insights and knowledge with the help of state-of-the-art computer simulations [1], [2]. An HPC cluster being a complex assemblage of interconnected compute systems requires optimized application software for achieving higher performance over the individual compute systems. With the accessibility of multicore processors in modern personal computers (PCs), it is necessary to have applications that can exploit these parallel architectures [3]. The nature of complexity in a cluster based HPC system makes it even more difficult to write applications that can interact with the system as a single unit. Also, there are challenges in porting the codes running on PC including the legacy codes [4] directly to the HPC system which requires building the code from scratch to utilize the capabilities of the system. HPC systems are ubiquitous yet intangible to many researchers for the development and testing of scientific applications. These systems are very expensive due to the costly hardware components along with the initial infrastructure required to house them as well as the operational cost of high energy consumption both for powering and cooling the components. Therefore, it is quite understandable that such costly HPC systems being used by many users simultaneously for their production jobs, cannot be allowed to use as a testbed platform for writing, testing, and developing efficient applications from scratch. The rapid increase in computing power (Moore's law [5]) means that software or computational methods that are highly scalable would be able to optimally exploit these advances in hardware capabilities. With the advent of small and highly affordable single-board computers like Raspberry Pi [6], [7], it is possible to make an HPC-like cluster environment by connecting several such boards. In an earlier work [8], we assembled a 4-node working cluster named *Pradyut* using Raspberry Pi boards to demonstrate an HPC-like environment. This dedicated system can serve as a low-cost (<$350) and green testbed (<50W) [9] for developing parallel applications as an alternative to expensive large petascale HPC systems like *ANTYA* [10] which uses high power (>150KW) for its operation and placed inside a state-of-the-art Data Center of IPR.

In the work presented in this paper, we have used *Pradyut* as a testbed for developing and testing applications which demonstrate the well-known behavior of parallel and distributed computing (PDC) systems [11]. With *Pradyut* being easy to maintain, it allowed us to do all experiments with no restriction while developing the applications from different domains. To start with, we had the serial *FORTRAN* codes for each application which could run only on a single CPU core. These codes were first converted into *Python*. Even with friendly syntax along with an enormous pool of free high-quality libraries, *Python* is considered to be inefficient for many-core architectures which do not perform as fast as lower-level languages like *FORTRAN* or *C*/*C++*. This case study demonstrates how we have used *Python* to overcome these challenges to run the codes in *Python* efficiently on multiple cores in both shared and distributed memory architectures. A number of *Python* libraries exist for doing parallel processing with different approaches [12], like *Just In Time* (*JIT*) Compilation which is based on *numba*, *cython* etc., symmetric multiprocessing which uses multiprocessing, *joblib* etc., and

distributed computing using *mpi4py*, *dask*, etc for cluster like environment. Since the majority of the HPC systems including *ANTYA* use Message Passing Interface (MPI) standard library for developing codes that can use thousands of cores simultaneously, we have written the *Python* codes using MPI for *Python* (*mpi4py*) library. This means that we have the same environment for the execution of codes on both *Pradyut* and *ANTYA*. The only difference in the execution of the codes was the use of open source batch scheduler SLURM on *Pradyut* versus PBS on *ANTYA* which understandably will not impact the results.

IPR has been training college students in the field of HPC with its HPC outreach program through various academic projects. The outreach program comprises two parts: firstly to provide a hands-on experience of building the cluster to understand the cluster architecture and second to learn PDC concepts by constructing small parallel applications with their scaling performances. The parallel applications discussed in this paper will be used to train students as a part of IPR HPC outreach program. The rest of this paper is organized as follows. Section II lists select related work and highlights how the perspective of our work is different. Section III gives the details of IPR's HPC outreach program. Section IV gives an overview of *Pradyut*. Section V introduces the parallel applications that have been developed and their PDC performances. Finally, concluding remarks and future work scope are given in Section VI.

II. RELATED WORK

With the emergence of ARM-based SBC [13], especially Raspberry Pi offering multiple cores, a large number of small-scale clusters have shown up on the landscape mostly for demonstration and educational purposes. In the literature survey, several papers were found using Raspberry Pi for implementing HPC clusters for many different applications. Ou et al. [14], and Pfalzgraf and Driscoll [15] argued about the low-cost, low power, and better price-to-performance ratio of Raspberry Pi clusters as compared to the traditional HPC systems. A well-known 64-node Raspberry Pi cluster, Iridis-pi [16], was implemented for the benchmarking of scientific applications whereas D'Amore et al. [17] highlighted the use of such systems for Big Data applications. Similar to our work, an 8-node cluster for calculating the value of pi (π) using the cores distributed across the 8 nodes [18]. Apart from the systems for pedagogy purposes, there are commercial implementations like BitScope [19] for testing scientific applications before moving to large-scale traditional HPC systems. The work that the authors present in this paper is a part of an initiative to have a portable compute cluster as a package for testing and developing various HPC-related capabilities which are otherwise not possible on the big production clusters. The cluster that we used in this work is made *portable* (see section IV) for hands-on outreach activities and, like others, supports traditional technologies such as MPI upon which many scientific applications are built. In the context of prioritizing student engagement, our work is one of few works that used three different parallel applications with one of the applications coming from a real plasma physics problem. We believe that our approach of democratizing physical access to portable cluster hardware components along with the three parallel and scalable applications from completely different domains is crucial to introducing students to the domain of computational plasma physics which is the thrust of the research happening at institutions such as IPR (India), University of York (UK) and more globally in support of, among other research frontiers, the wider fusion research initiative. The experience of running and scaling testing on a petascale production cluster completes the learning cycle from the testbed cluster to the petascale cluster.

III. HPC OUTREACH PROGRAM

HPC outreach program at IPR is an integrated, long-term program designed to address the huge demand for skilled professionals and students knowing PDC to utilize the cluster-based HPC facilities. It is important that every young student, whatever their background, be provided with a meaningful opportunity to learn how HPC Clusters work and how to use them. A number of students have been trained through academic projects of 2-3 months as a part of this program. To synergize our efforts with the Scientific Social Responsibility (SSR) [20] initiative of the Government of India for taking the HPC education to society at the grassroots levels the program now has evolved and will include a week-long onsite hands-on HPC workshop for students selected from various colleges without any restrictions of their fields. This hands-on HPC workshop will cover the followings:

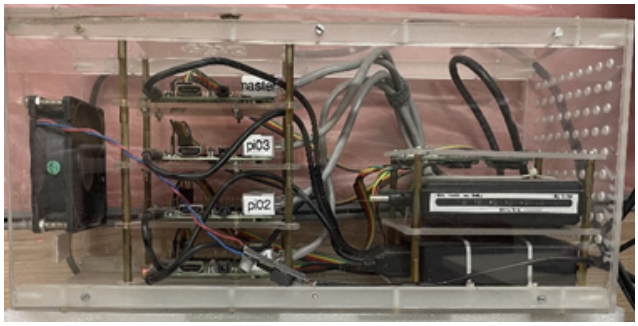
- The first phase will have training lectures by subject experts and hands-on exercises which would give an overview of the cluster components with its design, including the setting up of computing hardware, networking, storage, and installation of the required OS software.
- The second phase will be fully lab-oriented in which the students divided into multiple groups will be provided with the hardware components (Raspberry Pi-based single board computers, cables, power supplies, etc.) and each group will, under supervision, have to separately integrate these components into a cluster and test the assembled system for overall performance.
- Small parallel applications will be provided to each group to run on their own cluster and then optimize the python program for speedup/performance demonstration.

Our overriding aim with this dynamic, hands-on learning program is to empower students in the early stages of their learning to achieve their maximum potential: by providing them training in this emerging field in a format that is unique and will equip them to use computational thinking for solving the problems that can possibly change the world. The advent of methods based on Machine Learning/Deep Learning/Artificial Intelligence for analyzing the massive amounts of data has further necessitated the requirement for HPC. The Hands-on HPC Workshop will build capacity amongst students in HPC driven discovery of new insights and knowledge. The program

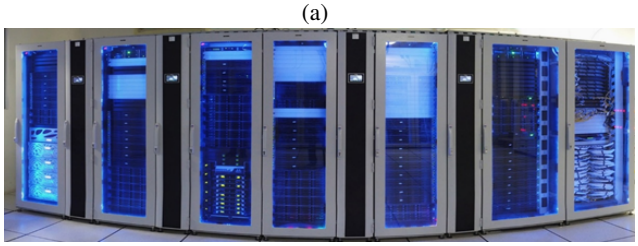
can be scaled to train teachers from government educational institutions at both school and college levels, who can then take the HPC education to their students.

IV. OVERVIEW OF TESTBED CLUSTER: *Pradyut*

The development of parallel applications which can run on large-scale HPC clusters like *ANTYA* requires that the testbed cluster should be representative of modern HPC systems in terms of hardware architecture and software. The testbed should be small in size for easy portability and assembled from low-cost hardware components which consume very less power to ensure economic feasibility. Our testbed cluster named *Pradyut* has been tested to highlight its parallel capabilities like large HPC clusters and demonstrated the well-known behavior of PDC systems with the performance results.



(a)



(b)

Fig. 1: (a) *Pradyut* housed in a portable acrylic box (30cm x 15cm x 15cm). (b) *ANTYA* with its components in the racks in Data Center occupying a large space more than the size of a large room.

It consists of 4 Raspberry Pi Model 3 B+ nodes where 1 node is configured as the master node (pi-master) and 3 nodes as compute nodes (pi01, pi02, and pi03). All the nodes are connected through a 5-port fast Ethernet switch and are powered by a 6-port USB 2A power hub including the Ethernet switch using an in-house Frankenstein cable. The cluster is also equipped with an active cooling fan. The power-on process is hassle-free and is through a single commodity power cable coming from the power hub to give power to the cluster. All the components are assembled in an acrylic transparent box having only two external connections: one for Ethernet and the other for the power cable. *Pradyut* offers a total of 16 cores with 4 cores from each node to ensure that it can run both the shared (OpenMP) and distributed (MPI) application workloads which are commonly used in HPC clusters. The pi-master node acts as the gateway for handling user login, code compilation and job submission. Fig. 1 shows both *Pradyut* and *ANTYA* which have been used for the development and testing of applications. A comparison of cluster components is given in table I. Like most HPC systems, all 4 nodes are running the Raspbian operating system which is an optimized version of the Debian GNU/Linux distribution for the Raspberry Pi. For storage, a 64 GB pen drive connected to the USB port of pi-master is used as a shared directory across the nodes having all user files and installed packages. All the established compilers that are common to larger HPC clusters such as GCC, OpenMPI, etc. are installed in this shared pen drive. For workload management, the Slurm [21] job scheduler has been configured.

TABLE I: Comparison of *Pradyut* and *ANTYA* cluster components

Component	***Pradyut***	***ANTYA***
No. of compute nodes	4	260
No. of cores per node	4	40
Processor(s)	ARMv8	2 × Intel Xeon Gold 6148
Processor clock speed	1.4 GHz	2.4 GHz
RAM per node	1 GB	384 GB
Storage	64 GB	2 PB
Networking	100 Mbps Ethernet switch	100 Gbps IB Mellanox Switch
Job Scheduler	Slurm	PBS Pro

The performance of Pradyut has been tested with High Performance Linpack (HPL) benchmark. The HPL benchmark is used to benchmark the integrated cluster performance and is arguably the most well-known benchmark used to rank the supercomputers. HPL-2.1 was installed in the shared directory and the HPL performance results are given in table II.

TABLE II: HPL Performance Results of *Pradyut* for different values of problem size (N).

N	**Gflops**
5120	2.52
6400	2.86
7680	3.16
8960	3.36
9600	3.51
10080	3.63
18000	3.80

The cluster performance results for several problem sizes (N) have been obtained after optimizing other parameters of the HPL.dat file. We ran the HPL benchmark on our cluster starting with 1 core (serial mode) on a single node of the cluster and to a total of 16 cores (MPI mode). The goal of this benchmark is to completely fill the RAM and maximize processor use for the duration of the test. For N = 18000, the HPL run consumed 85-90% of the RAM from each node with *Pradyut* delivering a peak performance of 3.8 Gflops. By increasing the value of N further, we experienced that one of the nodes crashed due to high RAM usage. Although the total

performance of *Pradyut* is almost negligible as compared to that of *ANTYA* (HPL performance ~0.63 Petaflops), *Pradyut* offers a similar testing environment which is required for the development of MPI applications.

V. APPLICATIONS FOR PDC DEMONSTRATIONS AND THEIR PERFORMANCE

In the case study presented here, we have used *Pradyut* as a testbed for developing and testing applications. With *Pradyut* being easy to maintain, it allowed us to do all experiments with no restriction while developing the applications. All the required packages were installed from scratch on the shared directory. The following 3 applications were developed in *Python* for PDC demonstration:

- ***Pi Estimation (PE)***: This application estimates the value of pi (π) using the Monte Carlo method. By parallelizing this application, we want to show the embarrassingly parallel nature of such problems having no data sharing between the process which slows down the overall calculation.
- ***Heat Diffusion (HD)***: This application calculates the transfer of heat from high temperature to low-temperature areas in a 2-D plane using a 2-D heat equation. This application has been chosen to highlight how data sharing can be a bottleneck in parallelization.
- ***Lorentz Force (LF)***: The application demonstrates the motion of independent charged particles in electric and magnetic fields. The charge particles can be launched in parallel on multiple cores across the nodes where each core calculates the particle motion independently. The scaling performance of LF depends on the number of particles available in the system being simulated.

Fig. 2 shows the network schematic of *Pradyut* used for the development of the PDC application. We started with legacy *FORTRAN* codes which ran only on a single CPU core (serial). These codes were first converted into *Python*. These applications have been successfully scaled on *ANTYA* and the results are shown in the next section. Fig. 3 shows the execution in serial and parallel modes on a single node as well as 4 nodes of *Pradyut*. The 4 cores in each node are numbered 1,2,3,4 in the column for representation purposes.

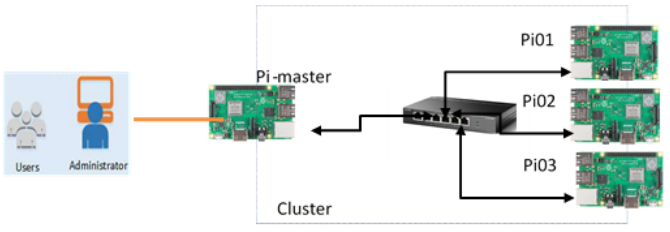


Fig. 2: Network layout of *Pradyut* showing accessibility for the users and administrator for application development.

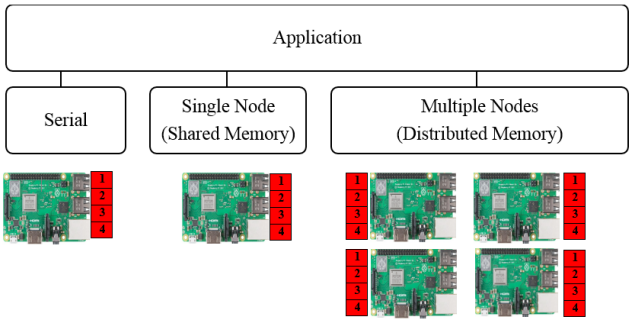
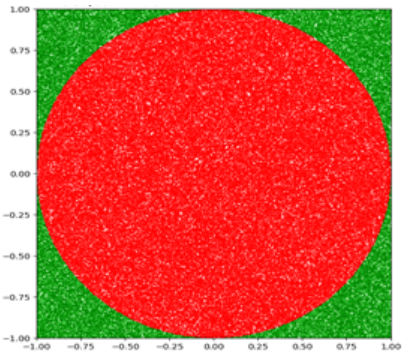


Fig. 3: Serial and parallel execution of application for PDC demonstration.

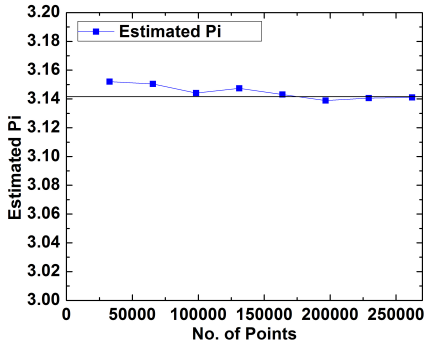
A. *Pi Estimation*

This application involves estimating the value of pi (π) using the Monte Carlo approximation method which is simple and can be easily parallelized. This approach involves generating a large number of random points in a unit square plane (Nt) and seeing how many fall in the circle (Nc).

$$\pi = 4Nc/Nt$$



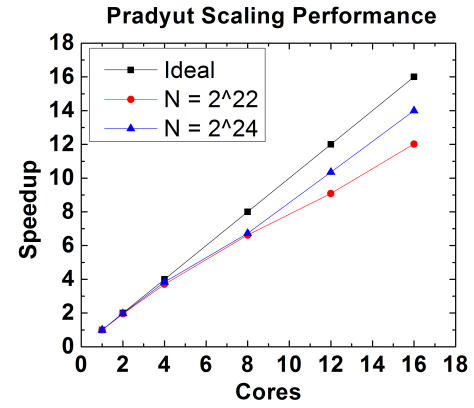
(a)



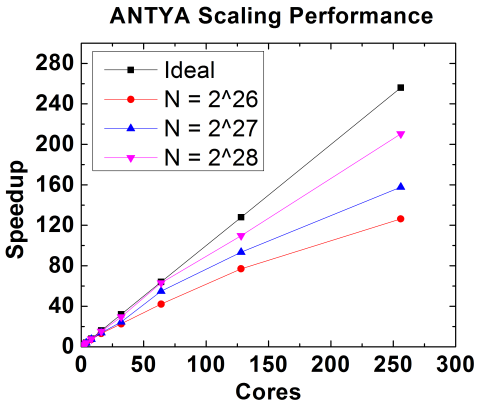
(b)

Fig. 4: PE application serial simulation performed on *Pradyut*, (a) shows the graphical representation of the points hitting inside and outside the circle. (b) The π value error plot shows with the increase in the value of Nt, the predicted value comes close to the actual value of π.

For an accurate calculation, it is necessary that the number of random points should be sufficiently large. The code is developed in *Python* and initially, the serial version of PE is written which involves a simple algorithm of iterating Nt times for calculating the value of Nc. Fig. 4 shows the results of the serial version of PE performed on 1 core of *Pradyut*. A number of runs were performed with Nt values (Nt = 2^{13} to 2^{18}) until a reasonably close estimate of the value of pi (π) was obtained. The simplicity of the algorithm allowed us to parallelize the iterative (sampling) part and perform the calculation in batches, dividing the no. of points equally in each batch. All the batches are completely independent and they can be launched on the available cores simultaneously. Since *Pradyut* has 4 nodes, we have used the *mpi4py* library for the distributed implementation across the nodes and have used the OpenMPI library for the parallel execution on the cluster. For Nt = 2^{18} points, on 1 core it took 22.77s, on 4 cores 6.6s, and on 16 cores it took 2.2s. Fig. 5 shows the scaling performance of PE obtained on *Pradyut* and *ANTYA*. The application developed on *Pradyut* shows good scaling performance on both the testbed cluster and production cluster and is a powerful exposition of both the parallel programming (MPI) concept and the Monte Carlo algorithm.



(a)



(b)

Fig. 5: Scaling performance of PC application on (a) *Pradyut* and (b) *ANTYA*, with different value of N = Nt.

B. *Heat Diffusion*

HD application involves visualizing the transfer of heat in a 2-D slab/plate from a high temperature zone to a low temperature zone by solving an ad-hoc form of the diffusion equation – the updated temperature value is a simple average over neighbouring cells at the previous time step. For visualizing the heat map of the 2-D slab, it is necessary to know the temperature throughout the slab, starting with heat sources and sinks defined at the domain edges. To get the value of temperature numerically, it is necessary that the 2-D slab is divided into a large number of cells, and temperature is calculated in each cell by distributing the data among multiple processes and processing them simultaneously using *Python* utilizing the *NumPy* and *mpi4py* for computation and communication as well as *matplotlib* for visualizations. This problem serves to introduce students to the concept of MPI and the importance of distributed memory computing as either problem becomes memory intensive or requires communication among processors.

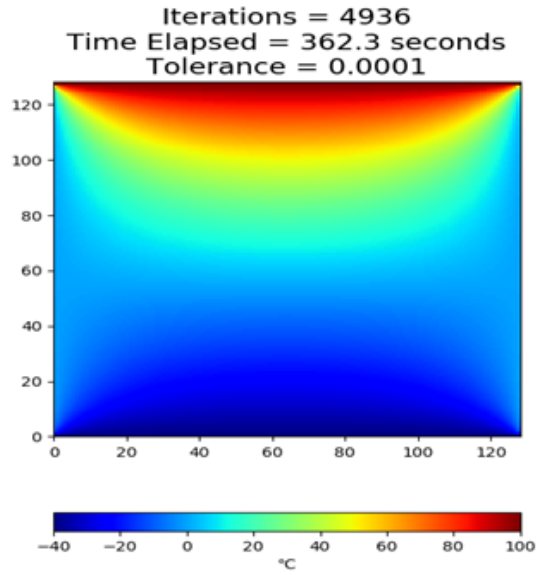
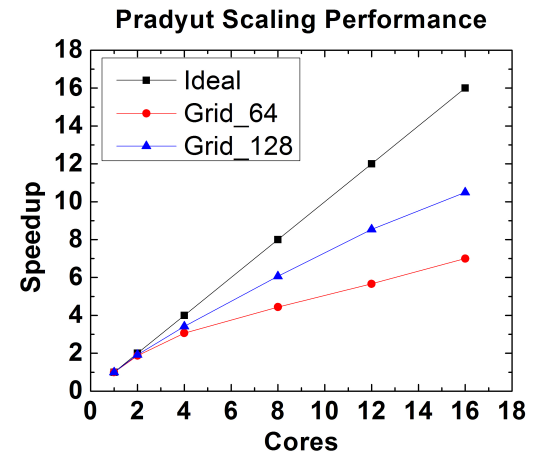
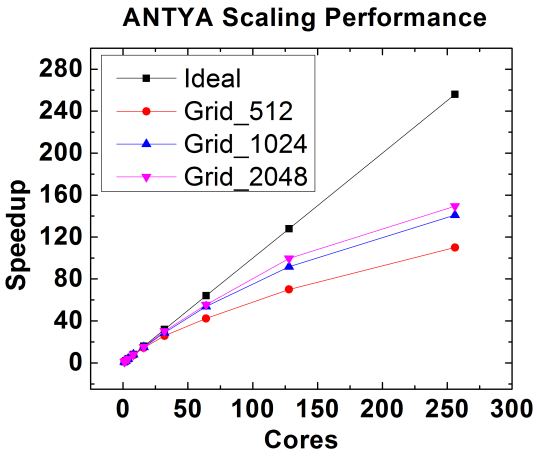


Fig. 6: HD application visualization of temperature distribution across the 2-D slab performed serially on *Pradyut*.

The important consideration while writing the parallel algorithm was to consider the contribution of immediate neighboring cells of a cell for calculating its temperature. The *Python* implementation starts with an n x n cell plane that evolves over time. The slab is initialized to zero temperature. The top edge is set as a heat source with a specific positive temperature along the top edge and the bottom edge is set as a heat sink with a specific negative temperature along the bottom edge. Each cell is updated every time step based on the average values of itself and its four neighboring cells. It is iterated until the temperature change in a cell becomes less than the tolerance value set. Fig. 6 shows the visualization of the distribution of temperature across the 2-D slab obtained on *Pradyut*.



(a)



(b)

Fig. 7: Scaling performance of HD application on (a) *Pradyut* and (b) *ANTYA*, with different grid sizes.

The Nt x Nt grid is decomposed across the Nproc processes in one dimension only. Each subdomain, therefore, has the dimension (Nt +2) x (Nt/Nproc + 2), wherein each dimension 2 rows/columns halo cells have been added to store the edge value of the adjacent subdomain required in the averaging process. To run on the 4 nodes of *Prdayut*, we have used the *mpi4py* library for the distributed implementation across the nodes and have used the OpenMPI library for the parallel execution on the cluster. Fig. 7 shows the scaling performance of HD obtained on *Pradyut* and *ANTYA*. The application developed on *Pradyut* shows decent scaling performance on both the testbed cluster and production cluster. However, as compared to the PE problem, the scaling performance is not so good since, at the end of each averaging cycle, the updated values must be communicated to the halo cells of the adjacent subdomain. Therefore, as the simulation is scaled across nodes, the memory communications start dominating the total computation time, negating gains made from having the additional computation power.

C. *Lorentz Force*

When a particle with charge q moves with velocity v through an environment having electric field E and magnetic field B, it experiences a force, F = q (E + v x B), known as Lorentz force. This force is responsible for the motion of charged particles in different trajectories inside a system.

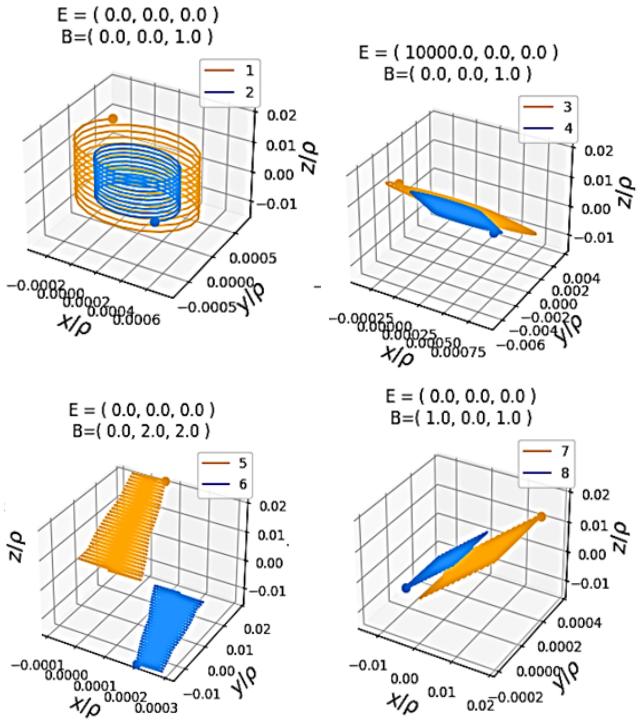


Fig. 8: LF application showing the motion of two charged particles under the influence of varying background fields (see plot title).

There are many physics problems, especially in Plasma Physics in which the collective behavior of the system is decided by the motion of charged particles inside it. Therefore, in a system having charged particles with negligible/weak interactions, the force on each charged particle can be independently calculated to simulate the collective behavior of the system. Here we have developed an LF application that does this work in parallel using independent processes across the nodes. Under various spatio-temporal structures of the background electromagnetic fields governed by E and B, charged particles exhibit characteristic "drifts" (e.g. due to the magnetic curvature) which are both fundamental and foundational to the understanding of particle dynamics in complex plasma systems, such as tokamaks used in fusion energy research. This application allows multiple such cases to be studied simultaneously. The implementation in *Python* starts with initializing charged particle parameters and the background E and B fields and updating the position-velocity vectors using a simple RK4 method. Using the *mpi4py* library, the LF carries out simulations of multiple particles with different values and in different environment settings simultaneously by dividing

particles among the available processes. Fig. 8 shows the motion of charged particles using the LF application. Each subplot has its own electric and magnetic fields with the trajectories of two particles (one electron and one ion) shown over time. This application is currently being extended to incorporate several interesting cases relevant to plasma physics and shall be communicated in future proceedings along with the implementation of a web-based interface that allows users to remotely access and run these applications on the IPR clusters.

VI. CONCLUSION

This paper presents a feasibility study of developing parallel applications using small-scale clusters of single-board computers which can be scaled directly to a production cluster. This approach allows the developers to learn about cluster architecture, prioritizes engagement during development to empower them in the early stages of their introduction to the HPC environment and parallel programming to achieve their maximum potential. The applications discussed demonstrates the concepts of parallelism in an HPC cluster and show how a small-scale testbed cluster like *Pradyut* can be a very attractive platform for parallel application development. The applications developed can be used to demonstrate how parallel computing affects the problem's efficiency and its computation time and also serve to introduce several key plasma physics concepts and computational methods in a methodical way. The applications PE and HD show good scaling from *Pradyut* to *ANTYA* and highlight that with increasing workloads the scaling performance improves. The PE application does not rely on communication between the processes whereas in the HD problem the neighboring values need to be communicated between the processes (calculating averages) which is reflected in its scaling behavior. LF application demonstrates the applicability of PDC to simulate Plasma Physics systems. As outlined, our perspective in this work is not to develop optimized applications but applications that can be scaled directly from a testbed cluster to large production clusters. Moreover, we believe that *Pradyut* being modular, portable, and affordable will be very useful for outreach activities to give students an insight into the world of parallel computing and its benefits.

We also look forward to future work that will explore the use of *Pradyut* with these applications to increase student engagement through our HPC outreach program. The hands-on HPC workshop with parallel applications discussed in this paper will help in the practical demonstrations to fulfill the following learning objectives for the students:

- Theoretical knowledge of cluster architecture.
- Building a typical cluster testbed with provided components.
- Constructing parallel processing schemes from sequential code using MPI and OpenMP.
- Analyzing the performance and scalability of parallel applications.
- Small parallel applications will be provided to each group to run on their own cluster and then optimize the python program for speedup/performance demonstration.
- Selected teams will be provided access to IPR's Petascale cluster '*ANTYA*' for developing and testing applications.

We hope the program may eventually also lead to an online HPC community where HPC Educators from colleges can share their knowledge and innovative ideas for teaching HPC/PDC. Furthermore, we intend to increase the number of nodes and finish the deployment of a web-based dashboard for *Pradyut* which will allow a wider engagement of students with IPR's R&D.

The authors ensure that the work presented here is reproducible. The codes used in work are available for use and can be downloaded from the bitbucket repository: https://bitbucket.org/deep2468r/pdc_applications/src/master/.

ACKNOWLEDGMENTS

The authors warmly thank Dr. Ravi for believing in the project and providing unconditional support. The authors would also like to acknowledge the Computer Center of IPR for providing the necessary resources for the project.

REFERENCES

[1] T. Liu, D. Lu, H. Zhang, M. Zheng, H. Yang, Y. Xu, C. Luo, W. Zhu, K. Yu, and H. Jiang, "Applying high-performance computing in drug discovery and molecular simulation," *National Science Review*, vol. 3, no. 1, pp. 49–63, 01 2016.

[2] J. Mengte, A. Raghunathan, S. Chakradhar, and S. Byna, "Exploiting the forgiving nature of applications for scalable parallel execution," in *2010 IEEE International Symposium on Parallel & Distributed Processing (IPDPS)*, 2010, pp. 1–12.

[3] S. Williams, A. Waterman, and D. Patterson, "Roofline: An insightful visual performance model for multicore architectures," vol. 52, no. 4, 2009.

[4] D. Aggarwal and A. Shingala, "Design and development of user-friendly interface environment for accelerating scientific research process: A case study for nuclear fusion applications," in *2020 7th International Conference on Computing for Sustainable Global Development (INDIACom)*, 2020, pp. 102–107.

[5] R. Schaller, "Moore's law: past, present and future," *IEEE Spectrum*, vol. 34, no. 6, pp. 52–59, 1997.

[6] Raspberry pi foundation. [Online]. Available: https://www.raspberrypi.org/about/

[7] J. C. Adams, J. Caswell, S. J. Matthews, C. Peck, E. Shoop, D. Toth, and J. Wolfer, "The micro-cluster showcase: 7 inexpensive beowulf clusters for teaching pdc." New York, NY, USA: Association for Computing Machinery, 2016. [Online]. Available: https://doi.org/10.1145/2839509.2844670

[8] D. Aggarwal, F. Cao, H. Charan, D. Deb, D. Ding, T. Dragon, M. Fuad, P. Kumar, H. Joshi, A. Moore, J. Shi, M. Zhu, M. Barnas, and N. Rodriguez, "Edupar posters," in *2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, 2019, pp. 347–349.

[9] S. Mollova, R. Simionov, and K. Seymenliyski, "A study of the energy efficiency of a computer cluster." New York, NY, USA: Association for Computing Machinery, 2018.

[10] Antya. [Online]. Available: https://www.ipr.res.in/ANTYA/Gananam_HPCNewsletter_IPR_Issue22_Sept2022.pdf

[11] Teaching pdc using the raspberry pi - csinparallel. [Online]. Available: https://csinparallel.org/csinparallel/raspberry_pi.html

[12] Parallel processing and multiprocessing in python. [Online]. Available: https://wiki.python.org/moin/ParallelProcessing

[13] S. J. Johnston, P. J. Basford, C. S. Perkins, H. Herry, F. P. Tso, D. Pezaros, R. D. Mullins, E. Yoneki, S. J. Cox, and J. Singer, "Commodity single board computer clusters and their applications," *Future Generation Computer Systems*, vol. 89, pp. 201–212, 2018.

[14] Z. Ou, B. Pang, Y. Deng, J. K. Nurminen, A. Ylä-Jääski, and P. Hui, "Energy- and cost-efficiency analysis of arm-based clusters," in *2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012)*, 2012, pp. 115–123.

[15] A. M. Pfalzgraf and J. A. Driscoll, "A low-cost computer cluster for high-performance computing education," in *IEEE International Conference on Electro/Information Technology*, 2014, pp. 362–366.

[16] S. J. Cox, J. T. Cox, R. P. Boardman, S. J. Johnston, M. Scott, and N. S. O'brien, "Iridis-pi: A low-cost, compact demonstration cluster," *Cluster Computing*, vol. 17, no. 2, 2014. [Online]. Available: https://doi.org/10.1007/s10586-013-0282-7

[17] M. d'Amore, R. Baggio, and E. Valdani, "A Practical Approach to Big Data in Tourism: A Low Cost Raspberry Pi Cluster," in *Information and Communication Technologies in Tourism 2015*, ser. Springer Books, I. Tussyadiah and A. Inversini, Eds. Springer, September 2015, pp. 169–181.

[18] G. M. Dorr, D. Hagen, B. Laskowski, D. E. Steinmetz, and D. Vo, "Introduction to parallel processing with eight node raspberry pi cluster," in *Midwest Instruction and Computing Symposium (MICS2017)*, 2017.

[19] Scalable clusters make hpc r&d easy as raspberry pi. [Online]. Available: Bitscope.com/cluster

[20] Scientific social responsibility. [Online]. Available: https://dst.gov.in/document/guidelines/scientific-social-responsibility-ssr-guidelines-2022

[21] Slurm scheduler. [Online]. Available: https://slurm.schedmd.com/documentation.html