J. Parallel Distrib. Comput. 118 (2018) 213–232

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

Let's HPC: A web-based platform to aid parallel, distributed and high performance computing education

Bhaskar Chaudhury [a,*], Akshar Varma [a], Yashwant Keswani [a], Yashodhan Bhatnagar [a], Samarth Parikh [b]

[a] *Group in Computational Science and HPC, DA-IICT, Gandhinagar, India*
[b] *DA-IICT, Gandhinagar, India*

HIGHLIGHTS

- An open platform to supplement conventional classroom oriented HPC and PDC education.
- Allows in-depth performance analysis from a system's perspective using over 20 parameters.
- Facilitates variability analysis for each parameter using highly customizable plots.
- Streamlines entire workflow using automated scripts and structured report generator.
- Data Repository with performance data of many implementations and architectures.

ARTICLE INFO

Article history:
Received 15 June 2017
Received in revised form 6 February 2018
Accepted 26 February 2018
Available online 20 March 2018

Keywords:
HPC education
Parallel & distributed programming
Performance analyzer
Multicore architecture
HPC database

ABSTRACT

Let's HPC (www.letshpc.org) is an evolving open-access web-based platform to supplement conventional classroom oriented High Performance Computing (HPC) and Parallel & Distributed Computing (PDC) education. This platform has been developed to allow users to learn, evaluate, teach and see the performance of parallel algorithms from a system's viewpoint. The *Let's HPC* platform's motivation comes from the experiences of teaching HPC/PDC courses and it is designed to help streamline the process of analyzing parallel programs.

At the heart of this platform is a database archiving the performance and execution environment related data of standard parallel algorithm implementations run on different computing architectures using different programming environments. The online plotting and analysis tools of our platform can be combined seamlessly with the database to aid self-learning, teaching, evaluation and discussion of different HPC related topics, with a particular focus on a holistic system's perspective. The user can quantitatively compare and understand the importance of numerous deterministic as well as non-deterministic factors of both the software and the hardware that impact the performance of parallel programs. Instructors of HPC/PDC related courses can use the platform's tools to illustrate the importance of proper data collection and analysis in understanding factors impacting performance as well as to encourage peer learning among students. Scripts are provided for automatically collecting performance related data, which can then be analyzed using the platform's tools. The platform also allows students to prepare a standard lab/project report aiding the instructor in uniform evaluation. The platform's modular design enables easy inclusion of performance related data from contributors as well as addition of new features in the future.

This paper summarizes the background and motivation behind the *Let's HPC* project, the design philosophy of the platform, the present capabilities of the platform, as well as the plans for future developments.

1. Introduction

Imparting effective Parallel and Distributed Computing (PDC) education to undergraduate students within a limited time-frame (e.g. one core course or an elective course) is always a challenging

* Corresponding author.
E-mail address: bhaskar_chaudhury@daiict.ac.in (B. Chaudhury).

https://doi.org/10.1016/j.jpdc.2018.03.001
0743-7315/

task because it involves teaching both the software as well as the hardware aspects of High Performance Computing. The main challenge is to make sure that after successfully completing an HPC/PDC focused course, a student understands the practicalities of doing HPC to achieve the maximum possible performance out of a particular system for a particular problem. This goal can be achieved only through well designed courses which simultaneously provides theoretical knowledge of parallel algorithms along with a set of well thought out programming assignments to impart the practical aspects of HPC/PDC. This need for keeping an eye on the practical aspects is best described by the quotation - "*In theory, there is no difference between theory and practice. But, in practice, there is.*" - attributed to Jan L. A. van de Snepscheut and/or Yogi Berra.

Most of the efforts towards the development of curricula in PDC for undergraduate courses and fostering HPC education have been supplemented by educators in the form of sample lectures, tutorials, modules, books, software, recommended assignments, sample exercises, problem sets etc. [14,24,33,37,40,42,49]. These initiatives are primarily focused on the important areas of algorithm design, programming and computer architecture, and aid in introducing key parallelism concepts such as concurrency, dependency, tasks, threads, problem decomposition, data parallelism, recursion, synchronization, race conditions, resource sharing etc. to undergraduates [22,38]. These efforts have significantly helped in popularizing PDC education at undergraduate level throughout the world. However, classroom based teaching that focuses mainly on theoretical aspects and book based concepts is not enough to fully convey the intricacies of HPC/PDC to students.

Theory of parallel algorithm design or theory of parallel computing is based on abstract concepts of time and memory which may ignore real life constraints for simplicity, and therefore not take into account non-deterministic and hardware factors [42,51]. The performance of a computer program depends on a wide range of factors like the nature of the algorithm, the machine (several hardware factors), compiler optimizations, the runtime environment, the input, the measurement methodology etc. and their mutual interaction [13,35,48]. These system dependent aspects are difficult to impart from a pedagogical point of view using solely lectures and material from books [4,36,39]. Even if one perfectly understands the behavior of a program and the properties of the targeted hardware system, one cannot confidently predict the behavior of the program on the system without actually evaluating the program in the given software environment–hardware setting. For example, theoretical concepts like Amdahl's law put a linear limit on the speedup, while certain implementations can achieve superlinear speedups due to factors like optimal cache utilization [3,26,27,35]. Therefore, there is a clear gap between theoretical learning, implementation, and understanding/analyzing the effect of non-deterministic factors on the performance of a program. A well-rounded PDC education should provide students with the required theoretical knowledge and at the same time help them deal with the various system dependent factors. It is important for a student to understand the HPC concepts from a system's perspective (as mentioned above, taking into account all of the behaviors of a system as a whole) because systems are in general non-terminating and non-deterministic, whereas the behavior of algorithms is terminating, deterministic and platform-independent [46].

Further efforts and new methodologies are required to educate competent undergraduate students to learn parallel programming from the whole system's perspective. We require effective student-centric tools and resources to help students in understanding how all components of the complex HPC ecosystem interact and function. There have been efforts in addressing these issues but they all tend to focus on a subset of the HPC ecosystem rather than providing students with a holistic perspective. For example, some of the recent efforts include works oriented towards providing students with better, more intuitive execution environments [16,25,34], libraries [17,23], interfaces [18], and simulators [7,53] for parallel and distributed programming. Other works focus on simplifying and explaining parallel computation [8,45] and concurrency [9,41] concepts to students. There have also been productivity oriented works, that aid students and instructors in the submission and evaluation of HPC programs [21,32,44,47]. While these are all important and useful, they have been made to address some specific issue, and none looks at the HPC ecosystem from the system's perspective. To the best of our knowledge, the web-based platform of *Let's HPC* is the first that provides tools for data-collection, plotting and analysis of performance of PDC programs while focusing on providing users with the whole system's perspective.

The need for resources that aid the process of analysis from a system perspective has been observed while imparting courses on HPC and Parallel Programming at our institute. It has been reported that evaluation in HPC/Parallel Programming courses need to focus on lab assignments and projects as much as on conventional exams that test students' theoretical understanding [10]. Lab assignments and projects are natural mechanisms for evaluation that allow students to analyze and realize the impact of various hardware/software/programming environment factors (system's perspective) on the performance of their own code. However, based on our experience, properly analyzing performance and learning to improve parallel implementations requires students to do tedious and mundane work (e.g. data collection and plotting) that does not directly contribute to HPC/PDC knowledge.

Our web-based platform (www.letshpc.org) aims to address these difficulties using an array of tools that streamline the process of analysis starting with automatic execution of parallel code, collection of required performance data, online plotting and analysis tools that help users to better understand various aspects impacting performance from the whole system viewpoint [30,50]. Our tool has been designed in a modular manner that allows addition of more advanced analysis tools and to improve existing tools without disturbing the platform's usage. While the *Let's HPC* platform has been built keeping in mind all the stakeholders in the HPC community, it has been motivated for streamlining the pedagogical process of HPC/PDC and hence is particularly useful for HPC course instructors and students. At present, we expect that a user has acquired a basic understanding of parallel algorithm design and can do some amount of "thinking in parallel", before the user starts using the tool.

The rest of this paper is structured as follows: Section 2 briefly describes our experiences of integrating PDC topics into our undergraduate curriculum and how that motivated the development of the *Let's HPC* platform. Section 3 describes the design philosophy of the *Let's HPC* platform and details how it can be useful for various stakeholders in the HPC community. Section 4 gives an overview of the website, the internal structure of the platform and the analysis tools. In Section 5 we demonstrate the platform's usefulness in performance analysis of problems using example case studies. The important findings regarding the effectiveness of the platform based on our course evaluation survey and user experiences are presented in Section 6. Section 7 summarizes all the features of the *Let's HPC* platform, and finally, Section 8 has concluding remarks including the current status, ongoing work and the future plans for the *Let's HPC* project.

2. Background and motivation

As an Early Adopter of the NSF/IEEE-TCPP curriculum initiative on Parallel and Distributed Computing [37], we have undertaken a long term multi-year effort to integrate PDC topics into core and elective courses throughout the undergraduate programs in

ICT (Information and Communication Technology) and ICT with minor in CS (Computational Science) at DA-IICT, India. ICT and CS focused undergraduate programs at DA-IICT have been thoughtfully designed to integrate disciplines such as Computer Science, Electronics & Communication, Information Technology, Natural Sciences, Humanities and Design, making it quite broad in scope. Therefore ICT/CS graduates have certain performance capabilities not found in conventional graduates from any of the individual disciplines.

During the first two years (semesters I–IV) of the undergraduate programs at DA-IICT, students have to take core courses in Programming, Calculus, Discrete Mathematics, Algebraic Structures, Computer Architecture and Organization, Object-Oriented Programming, Data Structures, Signals & Systems, Probability & Statistics, Algorithms, Systems Software, and various courses in Communication and Electronics. These courses provide a broad foundation and are helpful prerequisites to understand PDC topics as well as its applications. The course contents of most of the above mentioned courses are inline with the ACM Computer Science Curriculum guidelines [28] and several of these courses have significant lab components; however, the syllabi are focused on single processor and sequential programming. Since these technological courses are introduced from the very first semester, the students quickly gain the necessary background to specialize and explore specific sub-areas in the later part of the program via elective courses. Keeping in mind future technological shifts and to encourage the scope of specializing into specific sub-areas such as PDC, as a first step we introduced an elective course titled *Introduction to GPU Programming (IT477)* for B.Tech (ICT) students (Semester VII) in 2014. Next year, in 2015, we introduced a core course titled *High Performance Computing (CS301)* for the students (Semester V) of the B.Tech (Hons. in ICT with minor in CS) program.

In the following subsections, we briefly describe the design & implementation strategy of our PDC courses (particularly the core course CS301). This provides the background for some important findings as well as experiences gained after teaching the course consecutively for two years. The difficulties faced, and the lack of tools or resources present to address them inspired the development of an in-house platform to supplement teaching of HPC courses at our institute. The general nature of our findings and the difficulties faced provided motivation for the *Let's HPC* project, which we have designed in a general manner, as discussed in Section 3.

2.1. Course design and implementation

Before designing and finalizing the course contents of our PDC focused courses, we did an extensive study of the available literature on guidelines and strategies for integrating parallelism into CS undergraduate curriculum [1,11,12,15,37]. This helped us in understanding what are the most important HPC skills a student should learn at an undergraduate level, and the possible ways to provide opportunities to "think in parallel" by giving adequate lab assignments and projects. However, the limitation of including only one core course on PDC in the undergraduate curriculum made us realize that the best strategy is to tailor the course contents/implementation according to the background of the students.

Our main objectives from such a course are two-fold. First, to provide sufficient practical exposure in HPC and hands on development experience to students in order to strengthen their design and implementation skills. Second, to equip students to understand and analyze the performance of their parallel codes from a system's perspective. With these primary objectives in mind, the course categorizes [1,31] the major application areas and important parallel patterns (algorithms) from the domains of CS and ICT. Case studies in lectures and lab assignments are designed in a way which gives enough scope to the students to understand the concepts in more details. The most important parallel programming concepts necessary to implement these algorithms are covered early on. This enables students to start working on their own implementations and analyzing their performance as early as possible.

CS301 is a core course offered to a batch of 60 CS students in Semester V and consists of 3 h of lectures and 3 h of lab per week (Bloom levels for this course [37]: K/C/A). The course covers parallel computing of both shared memory and distributed memory nature, using OpenMP and MPI respectively. A module driven approach [14] is taken for this course, dividing the course into five important modules with each module consisting of about 7 lectures. These modules are focused on - (i) Modern Processors, parallel computers and basic optimization techniques, (ii) Design of Parallel Algorithms, (iii) Shared Memory parallel computing with OpenMP, (iv) Performance Analysis and (V) Distributed memory parallel programming with MPI.

The lectures in these modules and the corresponding lab assignments are designed and delivered in such a way that students acquire the ability to "think in parallel" starting at the hardware level, building up via parallel algorithms to various software environments. The first half of the course covers concepts and skills comprising the thinking in parallel mindset towards problems that is required to implement an optimized parallel algorithm/code. The second half of the course focuses more on case-studies, assignments & team projects from different areas of Computational Science and ICT. The fundamental idea is to teach the basic theoretical principles of parallel computing by actively writing codes/programs using well known case studies and examples as found in books [4,22,36,38–40,42] and material available from the internet [24,33]. The systems perspective is achieved by requiring rigorous performance analysis of these codes executed on HPC machines.

The HPC course CS301 is offered at such a stage of the program (after semester IV) when students have good understanding of programming, algorithms, software development environments as well as computer architecture, and are already skilled in UNIX shell scripting, using remote shell access, secure file transfer, building using Makefile, handling libraries etc., which makes it easy to teach the technical aspects very quickly and introduce parallel-programming/ code-optimization based lab assignments from the second week itself. This allows the course to expose the students to more case studies and lab assignments giving them more practical experience of programming and analyzing performance in the HPC/PDC paradigm.

In the lab, the students are provided with desktops each equipped with Quad core high-end Intel processor and necessary open-source software/libraries for parallel programming and analysis. In addition to this, we have also installed a HPC cluster facility at DA-IICT having 5 nodes each having 16 cores (total 80 CPU cores, 400 GB RAM) with an InfiniBand interconnect/ switch. Students mainly use this facility for MPI-based implementations. More details about the integration of PDC in ICT/CS curriculum along with a schematic of the course design strategy can be found in [10].

2.2. Evaluation

A continuous and comprehensive student evaluation strategy consisting of exams, assignments, projects and oral presentations provide very good feedback on how well the students have grasped the PDC/ HPC concepts and also help in assessing the teaching methodology. Compulsory projects drive student interest in research related applications and motivate students for some of the

Table 1
Selected questions from the two course evaluation surveys, weighted average of responses (10-point scale), percentage of favorable responses (≥6).

How familiar are you about High Performance Computing concepts? (Survey-1)	1.98	2.1
How familiar are you about High Performance Computing concepts? (Survey-2)	7.78	84.9
How much did you consider the impact of the architecture of the computer on the performance of your programs? (Survey-1)	4.83	41.7
How much did you consider the impact of the architecture of the computer on the performance of your programs? (Survey-2)	8.36	88.7
How much did you consider the impact of the memory hierarchy related factors of the computer on the performance of your programs? (Survey-1)	4.8	41.8
How much did you consider the impact of the memory hierarchy related factors of the computer on the performance of your programs? (Survey-2)	8.26	86.8
How good an idea do you have about the project topic that you are going to do as part of your course project? (Survey-1)	3.65	16.7
How good an idea have you got about HPC from the course project that you did? (Survey-2)	8.07	88.8

advanced topics. The evaluation is based on 2 mid-semester exams (60%), and lab assignments & final project (40%). The heavy focus on practical programming helps towards achieving the system's perspective and strengthens students' design and implementation skills.

2.3. Student projects

After 30 lecture sessions (10 weeks) and completion of 10 lab sessions (three-hour each), students are required to propose a PDC focused one month project of their choice which they implement in a team of two (teams must choose distinct topics). A project comprises of understanding the algorithm, writing serial and parallel implementations, followed by theoretical as well practical performance analysis. A project report and a short oral presentation of the work contributes towards the student's evaluation in the course. From Autumn 2017, we introduced a 2% credit towards their final grade by asking relevant questions in the presentations. This token incentive (2% towards final grade) significantly improved the student participation, and class interaction during the project presentations. The students looked at the results being presented critically, analyzing the presentation content (algorithm, pseudo-codes, data, plots) closely while putting forward their questions. Working in teams builds teamwork and these presentations (along with the Q & A session) help in peer learning. From a course instructor's point of view, project reports, oral presentations and the quality of Q & A session give a fair idea about the comprehension ability of the students and their understanding of PDC concepts. Depending on the diversity and breadth of the proposed projects, the students also get an opportunity to learn about different domains from their peers and get some exposure to research. A sample of reports of the projects successfully executed by the students of Autumn-2017 batch can be found under the "Repository" section of the *Let's HPC* platform (www.letshpc.org).

2.4. Course evaluation by students

Course evaluation through student feedback is very important in assessing the course outcome and determining to what extent the desired course objectives have been achieved. Student responses to well thought-out course specific questions helps in refining the curriculum, fine-tuning the focus of lectures and assignments, improving the effectiveness of course delivery as well as addressing student's needs and difficulties. We conducted two surveys in the CS301 course in the Autumn semester (July–November) 2017: one at the beginning of the course (Survey-1, during second week) and one at the end of the course (Survey-2, 14th (last) week). Around 50 students participated in the survey, responding to a set of 15 questions, revolving around different course aspects. An integer scale of 1 (negative/less/lower) to 10 (positive/high/greater) was used and we present some statistics related to some of the relevant questions from both surveys in Table 1; the second column is the weighted average of the responses (10-point scale) and the third column contains the percentage of responses that were at least 6, which we call the percentage of favorable responses. The full response data can be found in Table 6 in Appendix A.

The change in the responses from unfavorable to favorable clearly illustrate that the understanding of HPC concepts increased due to the course. In particular, the questions about architecture and memory hierarchy show that students realized the importance of the system components towards the end of the course. These course evaluation results provide justification for the design and implementation of the course.

2.5. Success stories

The success that students have achieved in international conferences provide further justification of the course design & implementation strategy. The students of the CS301 course were encouraged to do research work in the area of PDC/HPC and participate in various conferences to present their work. Students interested in research were able to perform commendably based on exposure to just one course. It is very encouraging to report about the following performances and achievements of the students from DA-IICT in the International Conference for High Performance Computing, Networking, Storage and Analysis (SC17; http://supercomputing.org/) and IEEE International Conference on High Performance Computing, Data, and Analytics (HiPC 2017, 2016 and 2015; www.hipc.org). Most of the following students received student travel grants to attend the conference to present their work, giving them exposure in the area of HPC. Name of students are mentioned in italics in the following list.

1. Research Poster titled "Parallelization of the Particle-In-Cell Monte Carlo Collision Algorithm (PIC-MCC) for Plasma Simulation on Intel MIC Xeon Phi Architecture" authored by *Keval Shah, Anusha Phadnis, Miral Shah* and Bhaskar Chaudhury got accepted for publication in SC17 Conference, USA, 2017. [Four reviews per submission, acceptance rate ≈ 58% (98 accepted out of 169 submissions)]

2. Paper titled "A Novel Implementation of 2D3V Particle-In-Cell (PIC) Algorithm for Kepler GPU Architectures" authored by *Harshil Shah, Siddharth Kamaria, Riddhesh Markandeya, Miral Shah* and Bhaskar Chaudhury got accepted for publication in HiPC 2017 Proceedings, 24th HiPC, India, 2016. [Four reviews per submission, acceptance rate ≈ 22% (42 accepted out of 184 submissions)]
3. Paper titled "Accelerated Fluid Simulation of Low Temperature Plasmas on Intel Xeon Phi MIC Architecture" authored by *Henil Shah, Anurag Gupta, Saumya Bhadani* and Bhaskar Chaudhury got accepted in HiPC Student Research Symposium, India, 2016. [Three reviews per submission, typical acceptance rate ≈ 60% (34 accepted out of 57 submissions)]
4. *Yashwant Keswani and Akshar Varma* secured second place in Student Parallel Programming Challenge supported by Intel and NVIDIA at 23rd HiPC, India, 2016.
5. *Keval Shah, Abhi Shah and Parshwa Shah* secured third place in Student Parallel Programming Challenge supported by Intel and NVIDIA at 23rd HiPC, India, 2016.
6. *Shaleen Kumar Gupta, Sishtla Chaitanya Prasad, Visharad Bansal* secured third place in Student Parallel Programming Challenge supported by Intel and NVIDIA at 22nd HiPC, India, 2015.
7. *Henil Shah, Anurag Gupta, Saumya Bhadani* and Bhaskar Chaudhury received the Best Poster Award (SRS) at the 23rd HiPC, India, 2016.
8. *Keval Shah* got accepted into the SC17 Experiencing HPC for Undergraduates Program and presented a research poster at SC Conference, Denver, USA, 2017.

2.6. Experiences and student feedback

Student feedback via surveys (formal and informal), as well as experience of the instructor and teaching assistants provided a very illustrative subjective assessment of the teaching methodology and students' learning perspectives. Summary of the important lessons learned are as follows:

1. Students need to learn and understand the importance of system's perspective in HPC.
2. Students face difficulties in co-relating all the deterministic and non-deterministic factors which affect performance.
3. Students face difficulty in data collection for proper performance analysis as well as in presentation of their results.
4. Requirement of Teaching Assistants (TAs) in the Lab who have good technical background in HPC and can guide students on the above mentioned aspects.
5. Allowing more time for project presentations and including question/answer sessions involving students, TAs and the instructor helps in peer learning.
6. Student get very limited opportunity for self-evaluation while completing the assignments, as well as for peer learning post assignments.
7. A lot of time that is spent in preparing reports for assignments and projects, does not actively contribute to students' HPC/PDC knowledge.
8. Additional lab time for more practice via access to a computational cluster during the weekdays and weekends provides more learning opportunities.
9. Students have access to only a limited variety of hardware architectures where the parallel codes can be executed.
10. Instructor and TAs face difficulties in evaluation of assignments and projects due to lack of uniformity.

The *Let's HPC* project was motivated by the aim of addressing as many of the above issues as possible. During the Autumn Semester of 2016, offline scripts were made to address some of these issues and used for the lab evaluation of the CS301-High Performance Computing Course at DA-IICT. The offline scripts used for the lab evaluation became the precursor of the tools provided by the *Let's HPC* platform, and the ongoing project is evolving to accommodate more advanced and user-friendly analysis tools which provide a better system's viewpoint to users.

3. Design philosophy of the *Let's HPC* Platform

The design of the *Let's HPC* platform has been made on the basis of two major guiding principles:

1. Build web-based tools that ease and streamline the process of analyzing the performance of a parallel program from a system's perspective [46] and thereby help all stakeholders in the HPC/Parallel Programming community to understand the barriers to higher performance.
2. Keep the design modular to allow easy storage, access, exchange and manipulation of data, and addition of more analysis tools without disturbing the platform.

Keeping in mind a need for a benchmark-like database for students to learn and for instructors to use to guide students, the central part of our platform is an archival database that contains all the necessary data from a given computational experiment to study performance on the basis of the software (problem solving approach, serial and parallel code) used and the execution environment (machine, OS, compiler, parallel framework etc.) used in the experiment.

Each of the tools interact with the database and use the data to construct plots that can be used to analyze performance. The tools have been designed in a way as to benefit instructors, students, self-learners and independent researchers in the learning and teaching of HPC. Used together these tools can make the study of HPC as smooth as possible, allowing students and instructors to focus on the important parts by removing mundane tasks such as generating plots and compiling a report of analysis. This should help in providing more time to the students to focus on improving their codes and their understanding rather than on tasks not directly adding to their HPC/PDC knowledge.

The four major components of the *Let's HPC* platform, schematically shown in Fig. 1, are as follows:

- The **archival database** contains software and hardware details of HPC experiments for various problems, approach, machines combinations which will act as a benchmark for the HPC community to study and teach from. Analogous to the theory contained in textbooks, this will contain the performance data (as detailed in Section 4.1) which can be analyzed to throw light on the effect of various (non-deterministic) factors that are difficult to quantify in theory.
- **Analyze Benchmark Data** (comparison tool) provides plotting features [20] that allow users to analyze and compare the data from various benchmark experiments that have been contributed by members of the HPC community. This is useful when users want to analyze, compare, or learn from the performance that others have achieved using various implementation approaches on various machines and programming environments.
- **Analyze custom data:** A tool that allows anyone in the HPC community to analyze their own data. This is useful when users want to analyze results that they achieved using their own implementation–machine–environment combination using the plotting and analysis tools available on our platform.

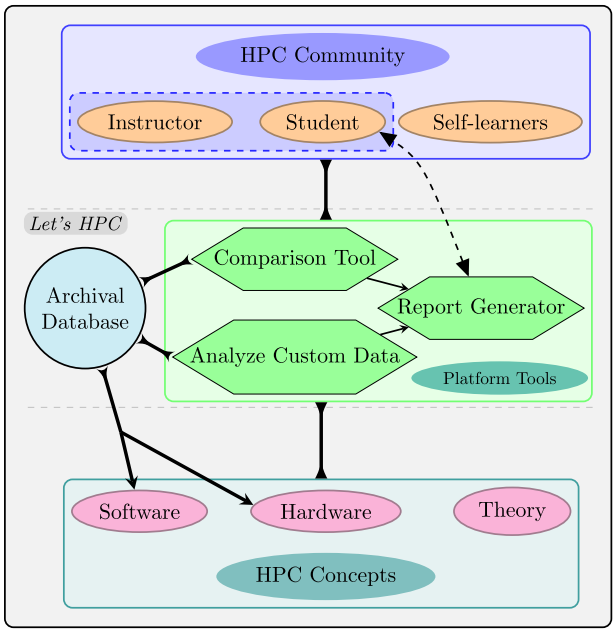


Fig. 1. Schematic of the conceptual design of the *Let's HPC* platform illustrating how the components interact with stakeholders of the HPC education community (platform users) and with important HPC/parallel programming concepts.

- A **report generator** mainly aimed at students which allows them to upload their data and answer a basic list of well thought-out questions based on the plots that are generated automatically. This gives a uniform report that facilitates evaluation of labs/projects by instructors. It is also useful as a starting point for anyone who needs an HPC related report.

In the following subsections, we provide a brief review of the tools of our platform from the perspective of three major stakeholders (the instructor, the student, and the self-learner) in the HPC community that we perceive will benefit the most from this framework.

3.1. Course instructors

The primary mechanism for course instructors to use the *Let's HPC* platform is via the archival database containing data from numerous problems and solutions approaches. These can be used to supplement the theoretical knowledge that students receive with analysis of performance of various approaches and machine factors. These would provide students with example analysis that they can easily grasp and replicate during their own work. The platform also contains a comparison tool which allows users to upload and compare their custom data with any of the existing data available in the database. This provides an additional means by which instructors can teach students; instructors can use this to showcase how certain factors can affect performance using real examples that would be difficult to showcase using theory and textbook knowledge alone. Our tools' focus on analysis ensures that students quickly realize the importance of all the factors that impact the overall performance. Section 5 contains an example of how users could use the platform's archival database (and similarly the comparison tool) to study and analyze the performance of problems in an HPC setting.

The use of these two tools are analogous to how in more theoretical courses, textbooks can first be used to teach a concept and later examples can be solved to show how those concepts are used. Together these provide the means for instructors to provide students with an overall understanding of concepts in a manner similar to that available in more theoretical courses without losing the focus on the practical side of HPC.

Apart from bridging the theory–practice divide, one of the major difficulties faced by HPC course instructors is in the evaluation of students. HPC courses need to focus on practical aspects, in a manner which is different from a normal algorithm/programming course. HPC lab assignments are not as straightforward as other programming assignments; there is not necessarily any "single correct answer". From a pedagogical point of view, the focus is on understanding the reason for the performance achieved rather than on merely achieving optimal performance. An HPC course needs to be lab/project oriented and in such cases instructors need a method to evaluate the students. A simple and common solution comes in the form of a report to be submitted by the students containing their understanding and analysis of the results. Our platform's report generator (see Section 4.3) is an apt tool for such cases. The systematic manner in which questions have been divided into sections allows for the compilation of a proper, concise report which helps instructors to do a uniform evaluation.

3.2. Students

Students need to realize the importance of acquiring a holistic view of the system while learning parallel programming and they need resources via which they can learn how system factors beyond the code, such as the hardware and the programming environment play a role in performance. Generally students have access to only a couple of computing systems in the lab sessions of HPC courses, and even those may have similar configurations. Similarly, their lab assignments/projects may be restricted to merely focusing on getting correct answers and on the implementation approach due to a lack of resources as well as time. Our archival database provides students with an opportunity to analyze the performance of various HPC systems, from hardware differences to the effect of the programming environment. The platform also provides an opportunity for students to analyze and compare the performance of their implementations to those of others; this can happen at a course level or on a global level with students first learning from their peers and then from others in the HPC community. This will help students in easily learning more than one approach to parallelize a problem and also motivate them in improving the performance of their code.

From the point of view of students, making a polished report requires a lot of effort starting from measuring execution times, performing statistical pre-processing, generating plots, and finally performing various kinds of analyses to understand why a given performance is being achieved. Combined with the automated data collection scripts (see Section 5.1) our platform's "Report Generator" tool automates much of this process for students. Apart from automating mundane tasks and aiding in making reports, our tool brings to notice many minor points of analysis that students may miss out on. This ability to look at the whole system and being consciously aware of the fact that each component can impact performance can help in writing codes more attuned to the system.

3.3. Self-learners

Users who already have the required background to study HPC/PDC and having access to decent computational resources will find our framework quite useful to study HPC on their own. Our platform supplements theoretical, book-based knowledge very well and can act as a "textbook" to study non-deterministic factors. The available tools aid self-learners to get rid of mundane tasks and focus on understanding HPC from the whole system's perspective.

Apart from these stakeholders, independent researchers can also find use for the platform as a repository for benchmarked data, to contribute their own data and as a forum for discussion with other HPC educators/researchers.

4. Overview of the website and tools

Let's HPC has been developed using the standard MEAN stack web framework [52]. The MEAN (MongoDB, ExpressJS, Angular, Node) stack allows for modularity in the structure of the whole platform from back-end to front-end. Each piece works independently, with Node-MongoDB-Express forming the server side framework to serve database requests and Angular forming the client side scripting framework. Being a popular choice for many developer communities, the open source technologies used in the MEAN stack are continuously maintained and upgraded by their respective communities. This also implies that regular upgrading and addition of new features to the platform is fairly straightforward.

The plotting and analysis tools all use values present[1] in the database depending on the user's access permissions. *Let's HPC*'s archival database is *conceptually* structured into different databases based on access permissions of users. The conceptual division based on access permissions are:

- Public benchmark data accessible to everyone, even without an account. These would contain data that has been curated and made public by various contributors.
- Data of courses accessible to course instructors of registered institutes. Each batch of students may have (conceptually) different databases and the instructor would have access to and control over all of this data, with the ability to make data of a given assignment public as and when they want to.
- Data belonging to students registered in courses but not corresponding to course assignments, such as an extra projects taken up by the student. Students can directly upload their data and analyze it using the tools provided, even share it with peers and learn via discussions.
- Private data of individual contributors, only accessible to them. This will primarily be data pertaining to unpublished, ongoing research; collaboration among contributors would also be possible in this setting.

In Section 4.1, we provide details of the nature of the data that is stored in the database and how it is useful for analysis. Section 4.2 illustrates how we set filters for accessing data from the database and Section 4.3 discusses the report generator tool.

4.1. Nature of collected data

Inspired by the work presented in [6], we have arranged the algorithms in our database in separate categories, and the problems in a particular category share good amount of similarity in computation and data movement. However, unlike [6], we do not intend to keep the number of categories fixed. Each problem is categorized into broad problem categories (examples in Table 2; categories and problems within any category can be added/removed). Further, we store data about each machine's hardware (processor version, number of cores, cache sizes, clock speed, etc.) and the software/programming environment (OS, compiler, parallel libraries, API etc.) that is used for executing a particular problem. In addition to this, various theoretical details of the problem such as an algorithmic description of implementation approach used, the theoretical asymptotic computational complexity of serial and parallel codes etc. are stored in the database. All of this meta data provides the required background to start the analysis of a problem.

For each machine–implementation approach–programming environment combination, the serial and parallel versions of the code are run multiple times for a range of problem sizes and data of execution times is collected. Two kinds of time data have been collected which allow us to understand how costly I/O operations and other pre/post-processing is in any given problem.

- Algorithmic (ALG) time: This includes only the time for the core algorithm (memory access and operations), and does not include any time for the I/O part or other pre-processing.
- End-to-end (E2E) time: This counts the entire run-time of the program. In addition to ALG time, this includes the time taken for I/O (reading the input test-case files and writing the output data files as well) as well as any other pre/post-processing that is not a part of the core algorithm.

We run codes multiple times to allow later statistical processing, which allows us to not only use the mean of the raw data but also provides information about the variability[2] of the results and thus the nature of non-deterministic factors affecting the performance for that particular combination of machine and implementation. Based on this data we can then proceed to plot execution times (both ALG and E2E times), the relative speedup,[3] the efficiency, and the Karp–Flatt metric [2]; and these plots can be of statistics like the mean, median, standard deviation or range of all the runs [48].

The hardware specifications and OS, compiler versions are also collected. In our case, we used the following commands:

- `cat /proc/cpuinfo` (CPU)
- `lscpu` (CPU)
- `uname -a` (OS)
- `gcc --version` (Compiler)

All of the available plots, hardware specifications and OS/Compiler versions form the basis for preliminary and basic analysis of any parallel code and reveal a lot about the machine, the implementation approach, and the programming environment.

However this data is not fully sufficient to perform an analysis from a whole system's perspective, there are numerous non-deterministic factors regarding which we have no quantifiable data. Collecting only the execution times of codes leaves out a lot of information regarding many software–hardware interaction related factors, making analysis based on those factors difficult. We address this issue by also collecting data regarding various software–hardware interaction related parameters which quantify how they affect performance.

4.1.1. Collecting `perf` data

The *Let's HPC* framework uses the `perf` utility.[4] to collect data regarding software–hardware interaction related parameters using hardware counters, which is then presented to users similar to

Table 2
Illustrative set of categories and sample problems.

Category	Problems
Linear Algebra	Vector Dot Product Matrix Multiplication
Reduction, Scan, Sort	Array Sum Scan Quicksort
Image Processing	Grayscale conversion Median filtering
Divide and Conquer	Monte Carlo Pi using Series Sum

[1] Currently, the user module is under beta testing within our institute and therefore, only the public data has been made available.

[2] Based on suggestions in [48].

[3] The relative speedup $S = T_s/T_p$, where T_p is the time for parallel code and T_s is the time for serial code using the same approach on the same machine.

[4] https://perf.wiki.kernel.org/index.php/Main_Page.

Table 3
Software–hardware interaction related parameters calculated using `perf`.

Cycles	The number of cycles taken to execute a program.
Instructions	The number of instructions executed in a program.
Bus cycles	A count of the bus cycles occurring during the execution of a program.
Branches	The number of branches in the execution of a program.
Branch misses	The number of branch misses in the execution of a program.
Context switches	The number of context switches in the execution of a program.
CPU migrations	The number of CPU migrations in the execution of a program.
Page faults	The number of page faults in the execution of a program.
Cache references	A count of the cache-references occurring during the execution of a program.
Cache misses	Account of the cache-misses occurring during the execution of a program.
L1 data cache loads	A count of the L1 data cache loads occurring during the execution of a program.
L1 data cache load misses	A count of the L1 data cache load misses occurring during the execution of a program.
L1 data cache stores	A count of the L1 data cache stores occurring during the execution of a program.
LLC `loads`	A count of the Last Level Cache loads occurring during the execution of a program.
LLC load misses	A count of the Last Level Cache load misses occurring during the execution of a program.
LLC stores	A count of the Last Level Cache stores occurring during the execution of a program.
LLC Prefetches	A count of the Last Level Cache Prefetches occurring during the execution of a program.
dTLB loads	A count of the data translation lookaside buffer loads occurring during the execution of a program.
dTLB load misses	A count of the data translation lookaside buffer load misses occurring during the execution of a program.
dTLB prefetch misses	A count of the data translation lookaside buffer prefetch misses occurring during the execution of a program.

how the execution times (and derived metrics) are presented. This allow the users to analyze problems based on these parameters and this combined data (execution times and software–hardware interaction parameters) improves the analysis, making it more comprehensive by enabling users to back up their analysis with data related to the whole system especially how the code implementation interacts with the hardware resources available. The particular parameters calculated by `perf` are listed out in Table 3.

The parameters that `perf` calculates all directly affect the execution time of a program and hence the performance of parallel implementations. An approach may outperform another approach because, say one of them has less number of cache-misses. Having this data allows users to identify hardware resource use related issues in their implementation and improve it. When machines are being compared, these parameters can be easily used to identify the difference in the performance due to the hardware and thereby highlight the importance of considering the architecture while writing HPC programs.

As discussed earlier, any HPC system/model consists of 3 important blocks, namely the system hardware, the algorithm (specific implementation) and the programming environment. The `perf` data we have collected combines very well with the time related data to provide a holistic picture of any such HPC system and allow users to perform analysis from the whole system's perspective as we illustrate in Section 5.

Each of these blocks and their component parts have to be analyzed by segregating relevant components which affect the performance, and studying each of them as well as the correlation among them. We now discuss how this segregation has been naturally designed into our platform.

4.2. Data filtering process

Our platform's plotting and analysis tools get data for comparison from the database using a flexible filtering process that can be expanded with additional features and provides opportunities to compare a wide variety of factors that make up the HPC system. Fig. 2 contains a conceptual schematic that describes the flow of the tool when users filter data according to their requirements. A screenshot of how that is implemented on the platform is available in Fig. 3.

The user first selects a category of problems and a problem belonging to that category. Along with this, shared, distributed or heterogeneous architecture is selected. For the case study in Section 5, we would select "Linear Algebra" and "Matrix Multiplication" along with "Shared memory system".

The website then fetches data from the database and provides three options (compare either approaches, or machines or programming environments) for the primary basis of comparison. The user makes a choice here; in the example in Section 5 we illustrate using both "Approaches" and "Machines". Whichever basis is chosen, multiple instances can be chosen that the user wants to compare (for comparing "Approaches", users can select from many matrix multiplication approaches). The platform then provides a list of the remaining options in each of the remaining bases (users choose one each from available machines and programming environments for which the database has data for the selected approaches). The user selects one instance from each and data is fetched for all the problem sizes. The user can then proceed to adjust the plot settings according to their preference and study the plots. Users can also download the plots and include them in the report generator.

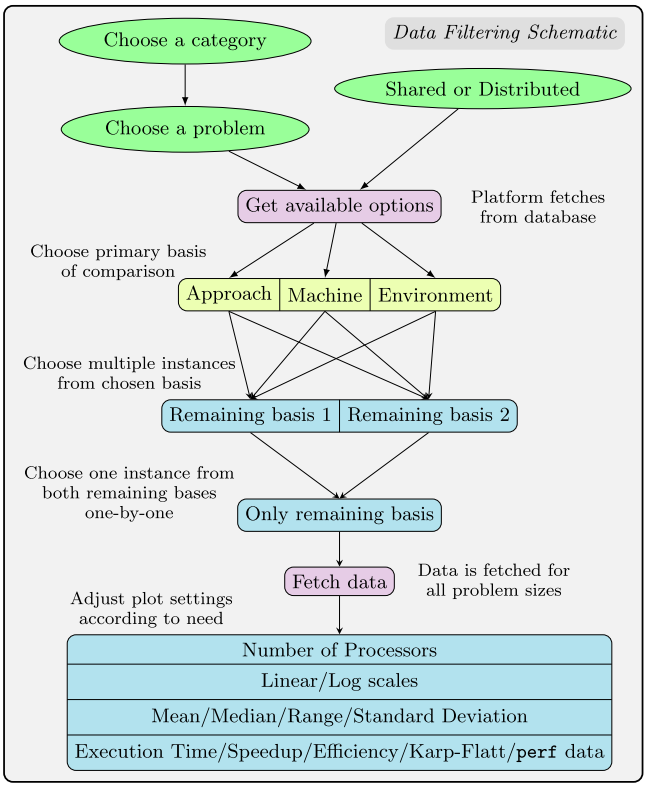


Fig. 2. Schematic explaining how the data filtering process works in the *Let's HPC* platform.

One salient aspect of our platform is the customizability of the options and settings of our tools. The plots that we provide can contain the mean, median, standard deviation and range of the data. The scales of the axes can be changed as needed between logarithmic and linear. And these customizations can be done for any of the metrics that are available, be it time, speedup, or any parameter calculated by `perf`. Further, depending on the metric (or otherwise) the minimum and maximum values on the axes can be set according to preference to zoom the plot to a desired portion. Thus, our platform allows ample customization options for the user to be chosen and used as required.

4.3. Report generator

As we have already discussed in Section 3.2, students need a mechanism to quickly and efficiently make reports based on their analysis that instructors can evaluate uniformly and easily. The report generator part of our platform has questions designed to significantly ease this process for students. Table 4 contains the broad idea of the division of questions in the report generator into various sections and a short description of the kind of questions. It allows students to upload their results, get plots automatically generated and a set of questions which students answer on the basis of their analysis. They get a `.tex` file along with the required plots that they can then modify as required and compile to get a PDF report. It forms the last step in the chain of any HPC analysis, and our platform automates all aspects starting with compiling and running of the codes and collecting performance data automatically to uploading execution results and finally getting a report containing the analysis.

5. How to use *Let's HPC* tools and case studies

In this section we discuss how the platform can be used and illustrate how various stakeholders are benefited from the use of the *Let's HPC* platform. We go through the normal workflow that someone working on an HPC/PDC problem would follow while using this platform. We have taken the **matrix multiplication** problem from Linear Algebra category for this illustration. The usage of the platform tools begin once the user has written the serial and parallel codes and ends in the analyses of various aspects that affect performance.

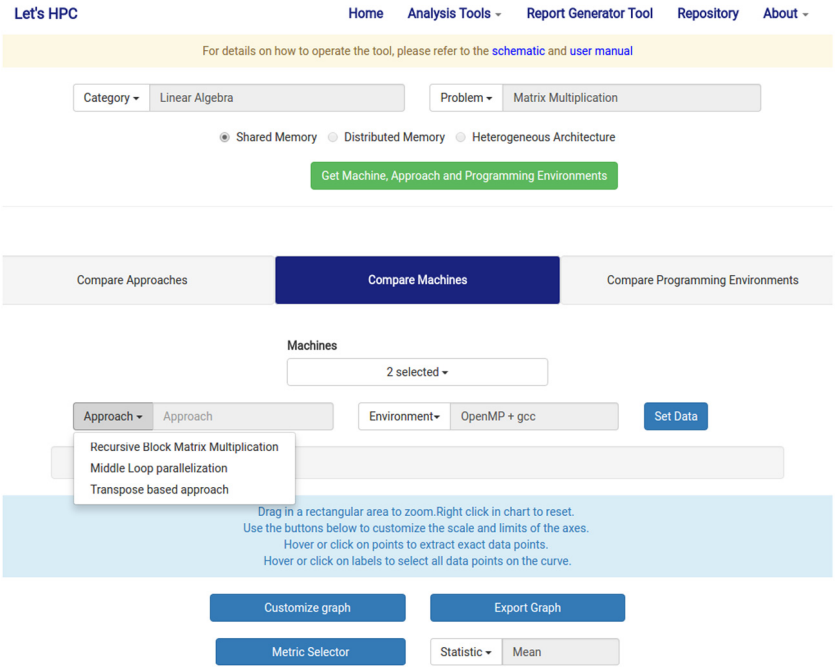


Fig. 3. A screenshot of how the data filtering process is implemented on the *Let's HPC* platform.

Table 4
Sample questions for the Report Generator.

Section	Question description
Basic description	Basic questions asking for description of the Serial and Parallel approaches used.
Complexity, analysis	Questions regarding complexity of the implementations, memory accesses, computations, theoretical speedup, etc.
Curve based analysis	Analysis of Execution time, Speedup, Efficiency and Karp–Flatt metric plots.
Further detailed analysis	Detailed analysis on the basis of concepts like cache coherence, false sharing, granularity, load balancing, etc.
Additional analysis	Analysis of various miscellaneous factors that impact performance; advantages/disadvantages and difficulties with respect to the implementation.

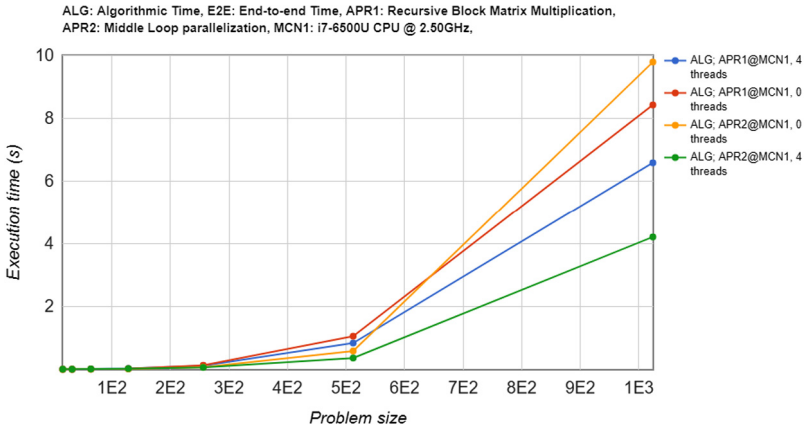


Fig. 4. Execution time (ALG: Algorithmic time in seconds) plot generated using the *Let's HPC* tool (www.letshpc.org) for comparing the two approaches of matrix multiplication.

5.1. Scripts for data collection

Once the codes have been written, Python scripts are used to automatically run the codes multiple times[5] to minimize anomalies which might arise due to the non-deterministic hardware dependencies (other processes, OS scheduling, etc.). Performance related data including execution times as well as hardware related parameters are gathered during each run and all this data is stored. These open sourced scripts (as well as sample OpemMP parallel codes) can be downloaded from the Github repository [29] and anyone can easily use them to automatically compile the codes, run them multiple times and collect data during each run.

The automated scripts ensure that data is collected in a format which can easily be uploaded onto the platform to create the required plots and to analyze them using the tools provided. The execution time data that is collected for each run of the code is used to plot the execution time, speedup, efficiency and the Karp–Flatt metric. Further, the scripts use the `perf` utility to collect various parameters detailed in Section 4.1.1. All of this data is then plotted as various aggregate statistical measures as discussed in Section 4.1.

5.2. Performance analysis using platform tools

In this section we show how once the data has been collected using these scripts and uploaded to the database, our plotting and analysis tools can be used to compare the performance given by an HPC system using some standard laws, ground rules and guidelines for interpretation of results [2,19,48]. There are numerous approaches possible for Matrix multiplication; we illustrate the usage of the tools of the *Let's HPC* platform by comparing two such approaches. Later we compare the performance of one of those approaches on two different machines.

Note: We have chosen the problem, matrix sizes, number of threads, number of figures to use in the analysis all with the aim of best illustrating the usefulness of the platform as a pedagogical tool from a system's perspective. We choose to use modest problem sizes and number of threads and instead use more figures to bring focus to the breadth of data that can be analyzed using the Let's HPC platform. The analysis we present is generic enough to be easily extended to more complex problems, performing deeper analysis as required. At the same time it is something that instructors can readily explain to students, enabling them to appreciate the need for proper data collection, plotting and analysis for understanding performance.

5.3. Comparing approaches

We compare recursive block multiplication approach (Approach 1) and parallelization of the middle loop using '`#pragma omp parallel for`' approach (Approach 2). We plot the curves for the parallel version for 4 threads and the serial version and then analyze those curves to understand the performance of the two approaches. We have considered square matrices (no. of rows=no. of columns), and the problem size in this case is represented by the number of rows (or columns). We analyze results (Figs. 4–14) of runs on an Intel® Core™ i7-6500U at 2.50 GHz.

5.3.1. Analysis based on the execution time graph

Based on Fig. 4, we can clearly see that Approach 2 is better than Approach 1 in terms of execution time for parallel case (4 threads). The number of recursive calls in the recursive block method is a function of the problem size, so as the problem size increases, more

[5] The script allows configuring the number of times the codes are run.

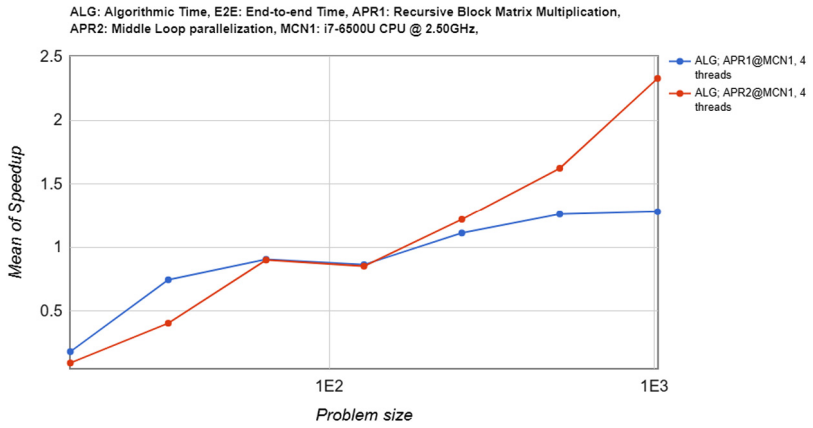


Fig. 5. Speedup plot generated using the *Let's HPC* tool (www.letshpc.org) for comparing the two approaches of matrix multiplication. ALG: Algorithmic time has been used to calculate the results.

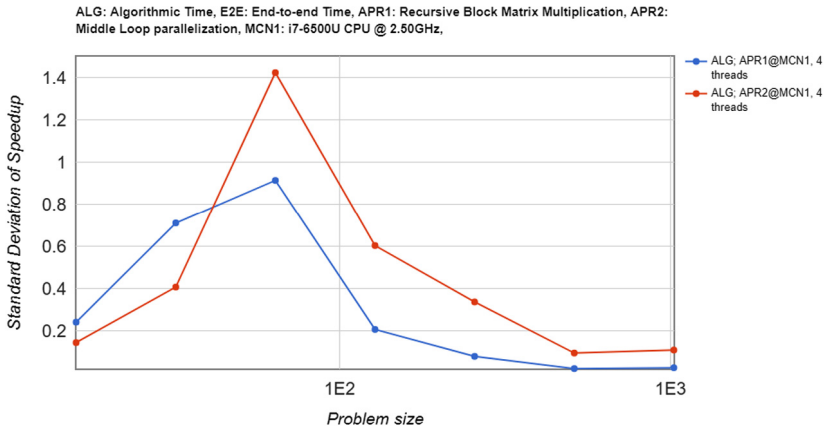


Fig. 6. Speedup plot (Standard Deviation) generated using the *Let's HPC* tool (www.letshpc.org) for comparing the two approaches of matrix multiplication. ALG: Algorithmic time has been used to calculate the results.

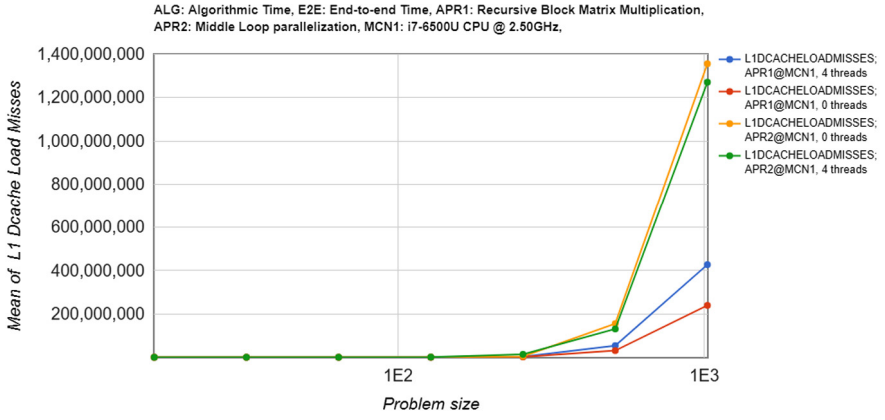


Fig. 7. L1d Cache Load Misses plot generated using the *Let's HPC* tool (www.letshpc.org) for comparing the two approaches of matrix multiplication.

number of recursive calls are made, and hence the time is much more than that for middle loop parallelization approach. However, for serial case (0 thread), Approach 2 is better than Approach 1 till problem size 512, but for problem size 1024 Approach 1 is better. This can be explained by looking at the machine details. For problem size 512, total storage required for storing A, B and C matrix is around 3MB which is less than 4MB L3 Cache. Therefore, even in the case of Approach 2 the whole arrays fit into the Cache memory and there is no obvious gain from block implementation. However, for problem size 1024, total storage required for storing A, B and C matrix is around 12MB which is more than 4MB L3 Cache and the whole arrays does not fit into the Cache memory. In this case (serial run for problem size 1024 and higher), Approach 1 (block matrix) aimed at utilizing the Cache memory does much better compared to Approach 2.

5.3.2. Analysis based on the (relative) speedup graph

The first thing we notice is that relative speedup (Fig. 5) only occurs for problem size $n >= 64$ (multiplication of two $64*64$ matrices) for both the approaches. This is because for smaller

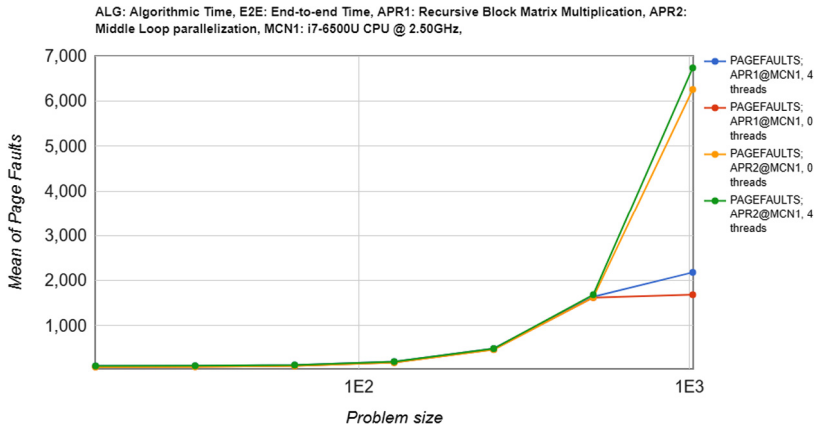


Fig. 8. Page Faults plot generated using the *Let's HPC* tool (www.letshpc.org) for comparing the two approaches of matrix multiplication.

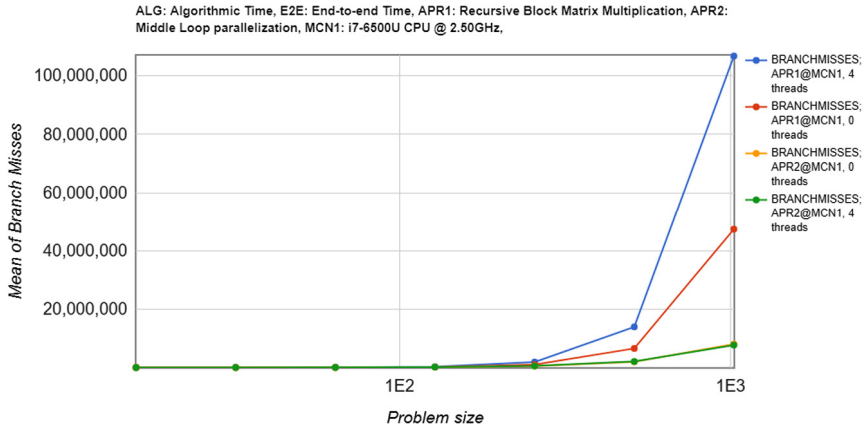


Fig. 9. Branch Misses plot generated using the *Let's HPC* tool (www.letshpc.org) for comparing the two approaches of matrix multiplication.

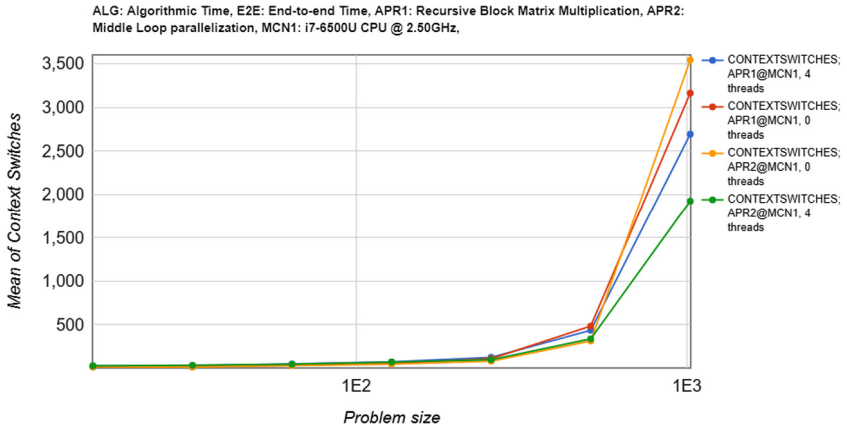


Fig. 10. Context Switches plot generated using the *Let's HPC* tool (www.letshpc.org) for comparing the two approaches of matrix multiplication.

problem sizes, the parallelization overhead in terms of initialization of threads and scheduling is significant in comparison to the parallelization achieved. The fact that the speedup data might be unreliable for smaller problem sizes can also be seen by looking at the Standard Deviation instead of the Mean for this, as in Fig. 6. For larger problem sizes, the computational part of the code is significantly higher than the overhead and we get reliable data and also notice a speedup.

In the recursive block approach, we see that the speedup tends to saturate at around 1.25. Knowing that the recursive block approach has been implemented using recursive calls suggests that the increasing number of recursive calls may be causing this saturation of speedup but this is unlikely to be the major cause since the speedup is relative to the same serial approach. While conventional analysis may resort to blaming scheduling overheads or memory access overheads as potential causes for this, we can use the `perf`

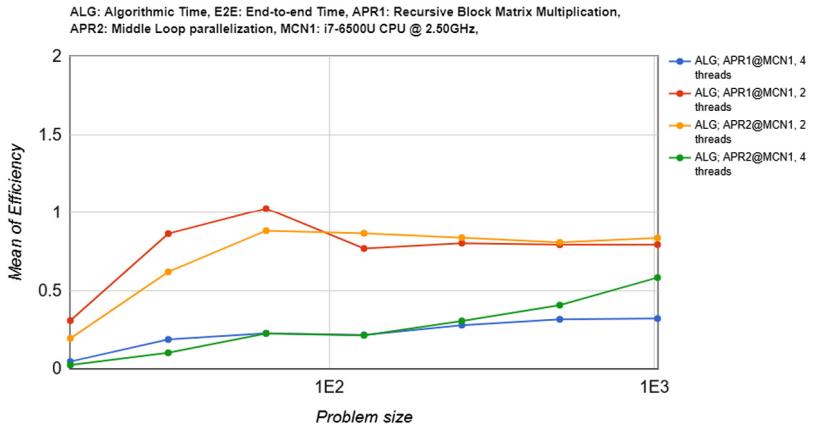


Fig. 11. Efficiency plot generated using the *Let's HPC* tool (www.letshpc.org) for comparing the two approaches of matrix multiplication. ALG: Algorithmic time has been used to calculate the results.

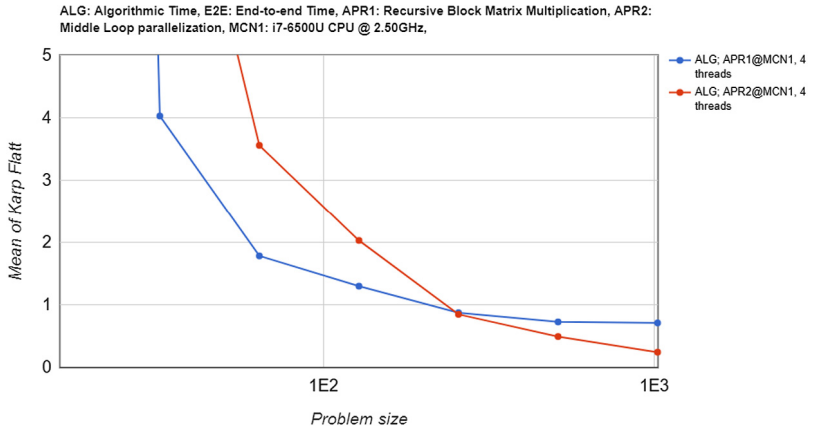


Fig. 12. Karp–Flatt metric plot generated using the *Let's HPC* tool (www.letshpc.org) for comparing the two approaches of matrix multiplication. ALG: Algorithmic time has been used to calculate the results.

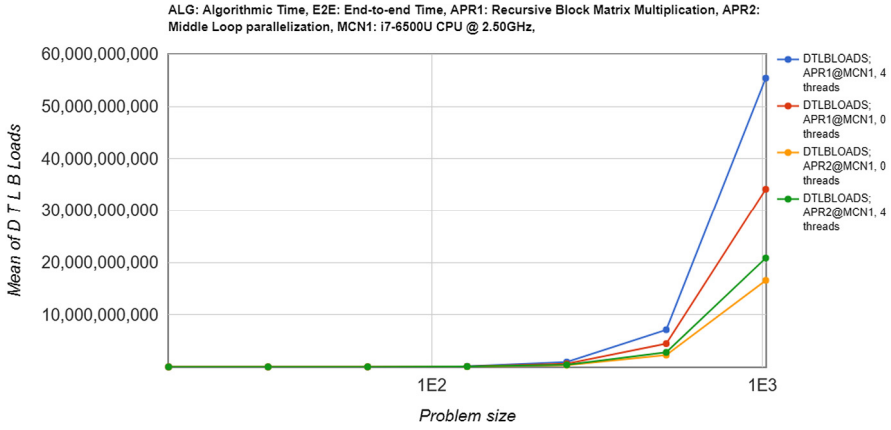


Fig. 13. dTLB loads plot generated using the *Let's HPC* tool (www.letshpc.org) for comparing the two approaches of matrix multiplication.

data to study this in more detail. Looking at the data we see that cache misses (Fig. 7) and page faults (Fig. 8) are reduced by the block matrix method compared to the middle loop approach. This is expected as the block multiplication method is used with the aim of improving memory accesses. However, we also see that branch misses (Fig. 9) and context switches (Fig. 10) are much more prevalent in the block multiplication method. This illustrates how having actual data allows us to better understand the causes for lack of performance without guessing.

In comparison, the steady slope of the speedup curve for the middle loop parallelization approach show that it is a better approach in terms of scalability as the problem size increases. This is to be expected since this approach does not have the overhead of the high number of recursive calls that the block method has. Additionally, a predictable pattern of memory access reduces the number of branch misses which also helps in making approach 2 more scalable. We can say that this approach provides better results in terms of speedup, but that should not be the final answer

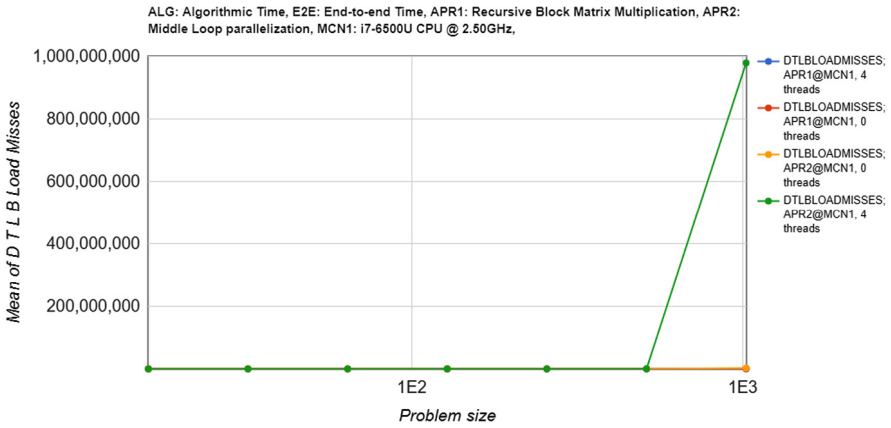


Fig. 14. dTLB load misses plot generated using the *Let's HPC* tool (www.letshpc.org) for comparing the two approaches of matrix multiplication.

and we continue our analysis to further understand the reasons for the achieved performance.

5.3.3. Analysis based on efficiency

Looking at a cross section ($x = c$ line) of the efficiency plot (Fig. 11) with multiple threads plotted, we note that the efficiency decreases as number of threads increases. This suggests that neither approach is scalable. However, we can still compare between the two approaches and say that the middle loop approach is more scalable than the block method as the decrease in efficiency in the former is not as sharp as that in the latter approach. Further, looking at the trend of the efficiency of 4 threads in the middle loop approach, we can see that there is a slight increase visible as the problem sizes increase. This suggests that the method may be scalable in terms of the problem size.

A similar analysis for the recursive block based approach shows that the efficiency for 2 threads is more than that for 4 threads, which indicates that the problem does not scale as we increase the number of processors. This is similar to the lack of scalability with respect to problem size.

5.3.4. Analysis based on Karp–Flatt metric

We use the graphs of the Karp–Flatt metric (Fig. 12) to corroborate the analysis we performed using the other plots. These plots represent the experimentally determined serial fraction of the code. For the recursive block approach, we see that the serial fraction starts to saturate towards the higher problem sizes, while for the middle loop approach the serial fraction continues to decrease. This provides further evidence for our earlier analysis.

5.3.5. Other analysis

Looking at other data, we can find further evidence for the efficiency of using block method, but of the inefficiency introduced due to implementing it using recursion. The `perf` data contains the data translation lookaside buffer (dTLB) loads and the data translation lookaside buffer (dTLB) load misses (Figs. 13 and 14). A Translation lookaside buffer (TLB) is a memory cache that is used to reduce the time taken to access a user memory location [5]. As defined a dTLB acts like a memory cache, so in the event of a dTLB load miss, it takes the system an increased number of cycles to fetch the data required. Therefore, a large number of dTLB misses affects the performance adversely. The percentage of load misses in the block method are only 0.0028% while that for the middle loop is 4.5%. However looking at the number of loads, we see that it is much higher for the recursive block approach compared to the middle loop approach which shows that the recursive implementation harms the performance significantly.

Overall, using all these plots, one can analyze the two approaches for solving the matrix multiplication problem and conclude that the middle loop parallelization approach is better and more scalable than the recursive block matrix approach. Both the approaches suffer from parallelization overheads, but the middle loop approach is comparatively better from a parallelism point of view.

5.4. Comparing machines

We present a similar analysis while comparing machines but here we focus only on the machine specific aspects that our analysis tool helps in isolating. We compare an Intel® Core™ i7-6500U at 2.50 GHz (Machine 1) with an Intel® Core™ Intel® Core™ i5-4200M at 2.50 GHz (Machine 2), both having been used to run the recursive block matrix multiplication approach. The analysis is done using the plots for the parameters (Figs. 15–21), primarily those that the `perf` utility calculates.

The reason for looking primarily at `perf` calculated data becomes clear when we look at the execution time (Fig. 15). The only conclusion we can get out of this plot is that Machine 1 seems to function better, which is to be expected as it is a newer processor (by 2 generations) and would have improvements over the older processor. As speedup, efficiency and the Karp–Flatt metrics are all derived from the execution time, even those would not be significantly helpful in further analysis. This prompts us to look at the `perf` data.

If we look at the LLC load misses graph (Fig. 16), we see that the misses incurred by Machine 1 is significantly lower than those incurred by Machine 2. Looking at the specifications of these machines, we can see that the L3 cache size for Machine 1 is much higher at 4096 KB while that for Machine 2 it is only 3072 KB. This extra 1 MB of cache explains the plot and provides a concrete reason to confirm the earlier hypothesis that Machine 1 is a better processor than Machine 2.

The LLC stores graph (Fig. 17) is also quite interesting, which shows that the stores that Machine 2 does is lower than the stores that Machine 1 does. This provides further evidence for our earlier assertion that the extra 1 MB of cache is helping Machine 1 be better than Machine 2. More stores implies that 1 MB is actually being used and hence it is improving performance. One can also look at this from the implementation perspective. We have three matrices of floats of size 1024×1024 (8MB), and hence require 24MB for all the data that is used during the whole matrix multiplication. Since we have 1 MB of extra cache space in Machine 1, it provides enough space for 4% more data than Machine 2 does. Looking at the plots for L1d cache stores and loads (Figs. 19 and 18), we can see

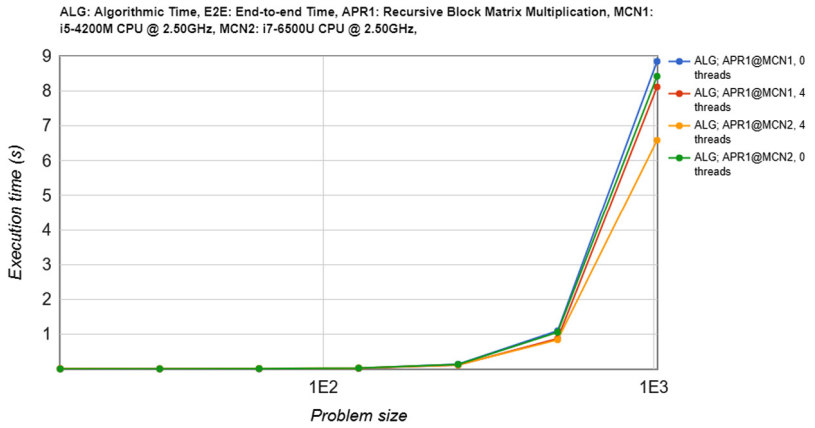


Fig. 15. Execution time (in seconds) plot generated using the *Let's HPC* tool when comparing the two machines. ALG: Algorithmic time has been used to calculate the results.

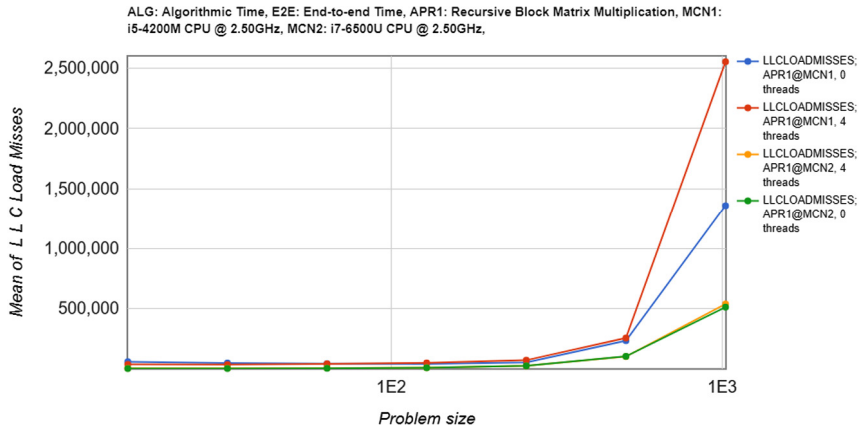


Fig. 16. Last level cache loads plot generated using the *Let's HPC* tool when comparing the two machines.

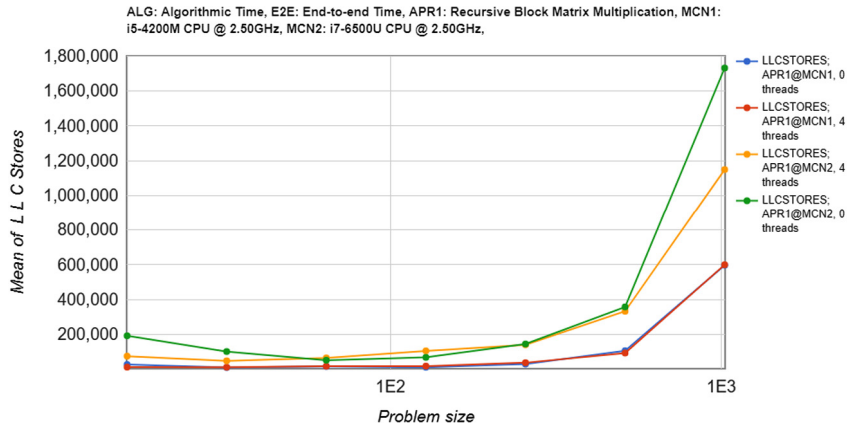


Fig. 17. Last level Cache Stores plot generated using the *Let's HPC* tool when comparing the two machines.

that both the processors have almost equal numbers. Since the L1d cache sizes are the same, this supports our earlier reasoning.

Our overall analysis that Machine 1 is better than Machine 2 is also supported by the plots for CPU Migrations (Fig. 20) and Context Switches (Fig. 21). Both show that Machine 1 needs many more CPU migrations and context switches both of which hurt performance. Apart from this, even going through the specifications of the two processors reveals more reasons why Machine 1 would be better than Machine 2. For example, the Maximum Memory Bandwidth of Machine 1 is 34.1 GB/s while that of Machine 2 is 25.6 GB/s; and considering that matrix multiplication is a memory intensive problem, this difference in speed also plays a role in the overall performance of the system.

Both our case studies illustrate how having data related to the whole system allows users to analyze performance from a holistic system perspective. The analysis presented show how it is important to have data about other interacting components of the system to analyze any particular component. The analysis we present for simple case-studies can be readily extended to more complex problems and systems. The tools provided by the *Let's HPC* platform are all designed to allow such analysis, combining analyses based on execution time, speedup, etc. with that of system

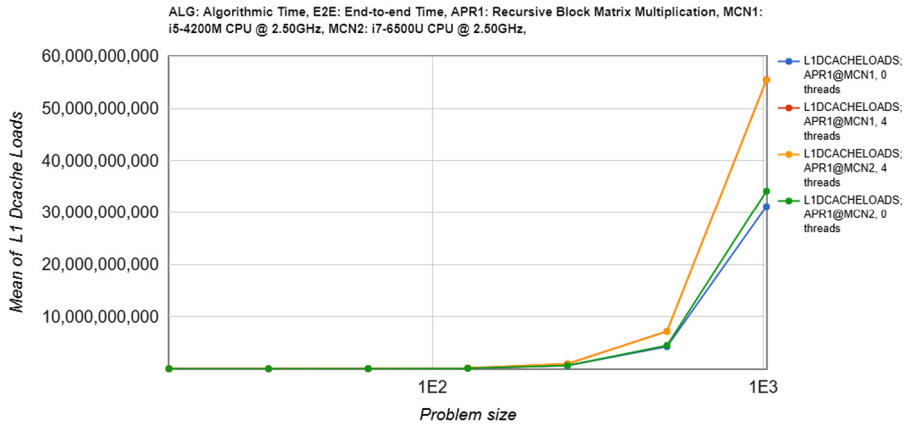


Fig. 18. L1d Cache Loads plot generated using the *Let's HPC* tool when comparing the two machines.

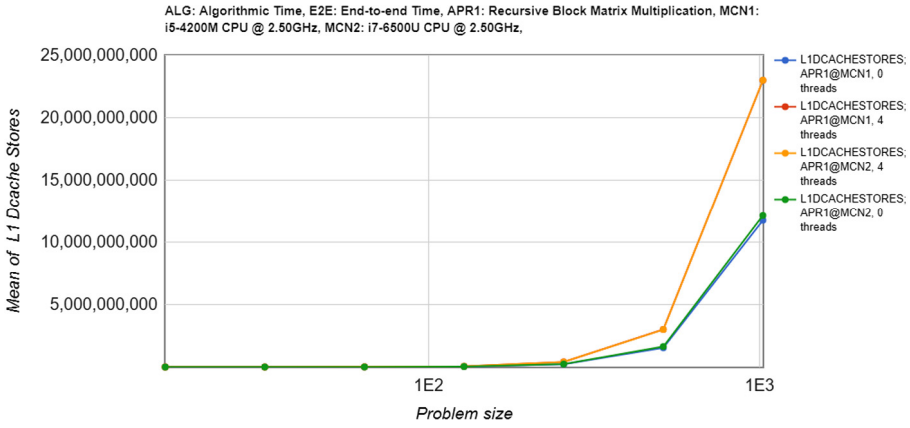


Fig. 19. L1d Cache Stores plot generated using the *Let's HPC* tool when comparing the two machines.

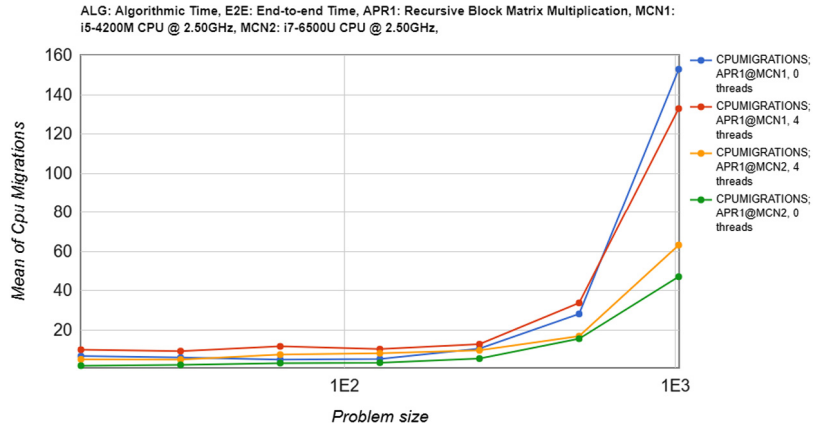


Fig. 20. CPU Migrations plot generated using the *Let's HPC* tool when comparing the two machines.

specific factors and helps in significantly improving the overall analyses.

6. Platform evaluation and improvement

At the end of the Autumn semester 2017, a comprehensive survey was used to gather feedback from the 53 students who used the platform throughout the CS301 course for their assignments and projects. The survey asked students to respond on a 5-point integer scale with 1 representing negative/less and 5 representing positive/more and we use the results to measure the effectiveness of the platform. Table 5 contains statistics of the survey (refer to Table 7 for the raw data); the second column contains the weighted average of responses (5-point scale) and the third column contains the percentage of responses that were at least 4, which we call the percentage of favorable responses.

The majority of the responses were favorable, suggesting that the platform was indeed useful in helping students learn HPC better. Some of the questions received overwhelmingly favorable responses and they are worth noting to understand the strengths of the platform. In particular, the platform eased the process of conducting HPC experiments and analysis and the report generator

Table 5
Questions asked at the end of the course, weighted average of responses (5-point scale), percentage of favorable responses (≥4).

Tools, features and information are clear and easily understandable.	4.0	77.4
Learning to use the platform is easy.	3.62	58.5
24 × 7 access to the platform and its tools helps in learning.	4.1	77.4
Using the platform significantly eased the task of performing experiments and analysis.	4.24	86.8
The platform's automation of mundane tasks helped you to focus more on learning HPC concepts better.	3.96	71.7
The questions in the report generator brought focus to HPC concepts and aspects of the assignments that may have been missed otherwise.	4.11	79.2
The platform helps in "thinking-in-parallel" and analyzing performance keeping in mind the whole systems perspective (architecture, algorithm and programming environment).	3.62	58.5
Overall, the tool enhanced understanding about HPC concepts and the importance of having a systems perspective.	3.79	67.9
The platform was useful for the HPC course project.	3.72	66.0
You will use the platform for any future HPC related projects	3.81	64.1
The Let's HPC platform is relevant to the CS301 course structure and its use was helpful in comprehension of the content.	4.23	86.8

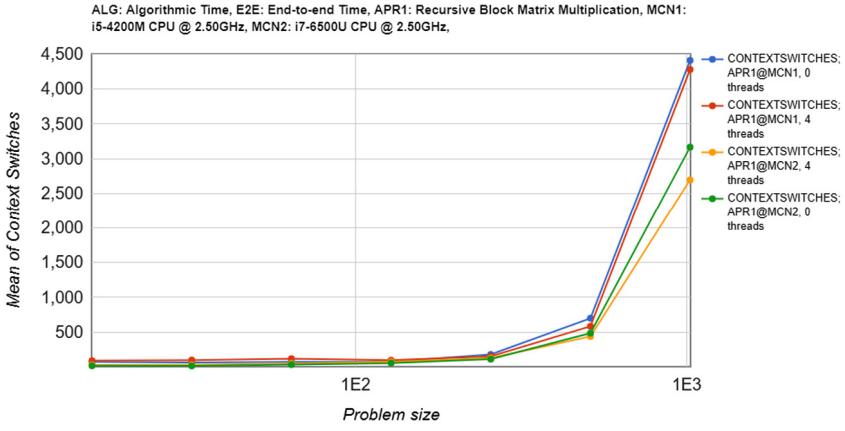


Fig. 21. Context Switches plot generated using the *Let's HPC* tool when comparing the two machines.

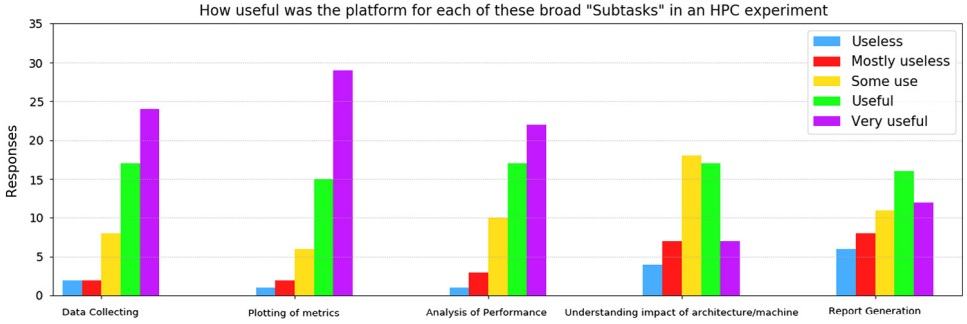


Fig. 22. Summary of the usefulness of the features of the *Let's HPC* framework from the survey. The *y*-axis is the number of absolute responses, 53 people responded in all.

brought focus to aspects of analysis that may otherwise have been missed. These along with the favorable responses to questions regarding gaining a system's perspective show that the platform definitely helps students to get a more holistic understanding of HPC/PDC concepts. Further, students found that the platform was relevant to the CS301 course structure and it was helpful in the comprehension of the content of the course. That observation combined with the course evaluations of the course (Section 2.4) and success of students who have taken the course (Section 2.5) provide ample evidence of the success of the platform in providing students a holistic system's perspective of HPC/PDC concepts.

To get a better understanding of which part of an HPC experiment was most facilitated by using the *Let's HPC* platform, we asked students to rate the platform tools in terms of the usefulness of the tools available on the platform for each "subtask" involved in an HPC experiment (on a range from "Useless" to "Very Useful"). Fig. 22 shows a bar chart of responses to this question, and it clearly shows that the tools provided by the *Let's HPC* platform are useful for all aspects of an HPC experiment. The platform was especially useful for data collecting, plotting of metrics, and analysis of performance; thus streamlining the process of conducting an HPC experiment for students.

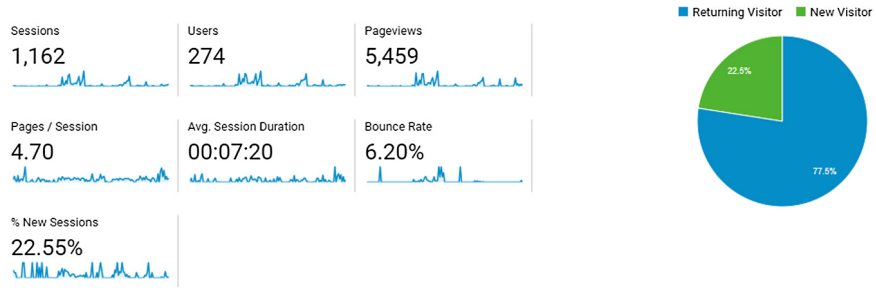


Fig. 23. Summary of the site usage patterns (collected using Google Analytics) for the *Let's HPC* tool in the period August 2017–December 2017 (roughly the duration of the CS301 course).

To get direct feedback from the students regarding aspects of the platform that could be improved, we looked at some of the written, long-form feedback provided in the "additional comments" question of the survey. A very frequent feedback provided by the students was that the report generator lacked customizability and this hindered the flow in writing up their analysis in reports. This explains why students believe that the report generator helped them realize aspects that they may have missed otherwise (Table 5) while at the same time finding the report generator only moderately useful (Fig. 22). Realizing the difficulties faced by the students, we are working on customizing the report generator. Another aspect mentioned multiple times in the written feedback was an initial difficulty in learning how to use the platform, and getting used to the various tools. To address this, we are creating a tutorial page for using the *Let's HPC* platform wherein we would be using inputs from the students in making the platform more intuitive. Thus the platform evaluation surveys provide a direct feedback mechanism for students which helps us improve the platform, making it more user-friendly and intuitive.

The *Let's HPC* platform also uses Google Analytics on the website in order to estimate the usage patterns on the platform. Fig. 23 summarizes the results of the platform for the duration of August 2017–December 2017, which roughly corresponds to the duration of the Autumn 2017 CS301 course offering. We make a couple of observations based on the Google Analytics data.

- Roughly 77% of the users are returning users, which indicates that the students found the *Let's HPC* platform to be helpful throughout their lab assignments and projects.
- The Average Session Duration reported is only 7 min and 20 s. This suggests that the *Let's HPC* platform allows users to quickly plot, summarize, and analyze data about their HPC experiments.

These survey results and the Google Analytics data have enabled us to understand how students use this platform, where the strengths of the platform lie and what avenues are available for improvement. We use these inputs to make better plans for future work (elaborated in Section 8), aiming to build on existing strengths and to enhance tools that can be improved.

7. Summary of platform features

The *Let's HPC* platform has been designed with the aim of easing the process of analysis of HPC/PDC programs from a holistic systems perspective. All the tools that the platform provides are targeted towards making the process as smooth as possible. The tools of the platform can be used starting at the stage where codes have been written. The automated scripts for compiling and running codes, and for data collection ensure that tasks not directly helpful for analysis are automated. The scripts also run the codes multiple times to account for any anomalies that may arise due to non-deterministic hardware factors. The *Let's HPC* platform's online analysis tools provide a comprehensive set of parameters on the basis of which to perform analysis. The plotting tools can be used to plot over 20 parameters which includes `perf` parameters along with the conventional metrics like execution time, relative speedup, efficiency, and the Karp–Flatt Metric. The customizability of the plotting tool ensures that users can plot the mean, median, mode or the standard deviation of the parameters (across multiple runs) and configuring the plots themselves by choosing ranges for the *X*/*Y* axes as well as changing the scales between logarithmic and linear. As demonstrated in Section 5 these parameters and plotting tools when combined together are very handy and facilitate the user to pinpoint the reasons for the superior/inferior performance of an approach, machine or programming environment in comparison to other approach(es), machine(s), or programming environment(s).

The platform also has a Report Generator tool which allows the students to analyze their approaches and write reports in a highly structured manner using the plots generated by the platform's tools based on the data submitted after running the scripts provided by the *Let's HPC* platform. Apart from empowering the students to focus on the analysis and not the presentation, this also allows the Instructors/TAs in the grading process. The instructors can also use the repository of data (codes, data logs and plots) that is made publicly available, to teach students more practical concepts that cannot be done using merely textbooks. This further enhances the platform as a lot of novel approaches are made available to the users of the *Let's HPC* platform, who may not have access to this data otherwise.

Overall our platform provides numerous tools that simplify the workflow of all users of the HPC community. The modular design allows our platform to expand in the future to incorporate more advanced features.

8. Concluding remarks, ongoing work, and future scope

Our framework is designed in a highly modular manner, providing numerous features that are useful to various stakeholders in the HPC community. The archival database is apt for acting as a benchmark for the HPC community particularly from a pedagogical perspective. At the same time, it is a tool that acts as a record of the HPC capabilities of various machines across years and can become a resource later for studying the gradual evolution of multi-core architecture and performance of parallel algorithms on such architecture. The plotting and analysis tools built on top of it, including the comparison tool and the tool allowing analysis of custom data both fill a gap in current approaches to HPC education. Our platform supplements theoretical and algorithm oriented education by showcasing the importance of keeping a holistic, system's perspective to the study and analysis of performance of HPC systems. The automated scripts and the report generator tool makes HPC education easier for both instructors and students by

bringing the focus to concepts that matter and allowing automation of trivial, mundane tasks.

The modular design of our platform allows for numerous, easy ways to improve the framework and to add features in the future. We are in the process of developing discussion forums allowing comments for each approach–machine–environment configuration in the database. Currently, the involvement of the HPC community on this platform is a passive involvement with most users only studying the data provided. The introduction of discussion forums will allow users to participate more interactively with other users. Extending this with HPC courses in mind, we will also introduce more refined forum abilities for HPC courses, with peer learning oriented features for students, and teaching and evaluation related features for instructors. This would not only bring in very fine access permissions which can enable all course assignments and projects to be done and evaluated via the platform itself but also facilitate active collaboration. Further features along with discussion forums can make the whole process of collaboration more streamlined.

While the inclusion of `perf` data allows a holistic system perspective analysis, we plan to include more advanced tools and models in the existing framework in the future which will offer additional insights to educators and students on how to improve performance (e.g. [43]). Along with catering to more advanced users, we also hope to incorporate features that aid beginning self-learners to understand HPC/PDC concepts. Apart from additional tools and features we also plan to extend and customize existing ones, making them more intuitive and user-friendly in the process. We will be adding more problems and categories to the set of benchmark data including those that use accelerators. We would also be allowing customized report generation that users can tune based on the particular problem at hand. These changes address various feedback that were received from students during the platform evaluation surveys.

As a long term goal, we envision the *Let's HPC* project to become a platform satisfying all requirements for HPC education, providing tools that combine to give an end-to-end solution starting with the scripts to automate data collection and statistical tools allowing analysis of variability of performance data to the final report generation (for students) and report evaluation (for instructors). Eventually, with active collaboration from the HPC community, the platform will include tools for online comparison of experiments run on various heterogeneous architectures and accelerators. As a more generic service, we want to eventually allow users to upload just their codes and then automate the whole end-to-end process using servers to run codes and allow users to directly move to analyzing performance results. This would truly allow HPC to be adopted by those without high end HPC resources.

Acknowledgments

The author Bhaskar Chaudhury would like to acknowledge the support received through NSF/TCPP CDER Center Early Adopter Awards Program (Fall 2014). This work has been carried out using the computing resources/HPC facilities at DA-IICT and we would like to thank DA-IICT for the invaluable support in carrying out this project. We thank students of CS301 High Performance Computing course (Autumn 2016 and Autumn 2017) at DA-IICT for their feedback and for providing some of the codes on which this tool was tested. We thank Omkar Damle and Rajdeep Pinge for their contribution towards the project. Many thanks to all the data (benchmark-database) contributors. We thank Dhruv Thakker for his very useful contributions towards the web development and maintenance of this website. We would also like to express our gratitude to all the reviewers for their comments which have helped us improve the quality of the paper immensely.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.jpdc.2018.03.001.

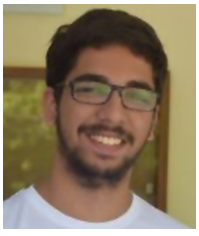
References

[1] J. Adams, R. Brown, E. Shoop, Patterns and exemplars: Compelling strategies for teaching parallel and distributed computing to CS undergraduates, in: 27th International Parallel and Distributed Processing Symposium Workshops and Ph.D. Forum IPDPSW, 2013.
[2] Alan H. Karp, Horace P. Flatt, Measuring parallel processor performance, Commun. ACM 33 (1990) 539–543.
[3] G.M Amdahl, Validity of the single processor approach to achieving large scale computing capabilities, in: AFIPS '67 (Spring) Proceedings Of the Spring Joint Computer Conference, 1967 New Jersey, pp. 483–485.
[4] Ananth Grama, George Karypis, Vipin Kumar, Anshul Gupta, Introduction to Parallel Computing, Addison-Wesley, 2003.
[5] Remzi H. Arpaci-Dusseau, Andrea C. Arpaci-Dusseau, Operating Systems: Three Easy Pieces v0.91, Arpaci-Dusseau Books, 2015.
[6] K. Asanovic, R. Bodik, J. Demmel, T. Keaveny, K. Keutzer, J. Kubiatowicz, N. Morgan, D. Patterson, K. Sen, J. Wawrzynek, D. Wessel, K. Yelick, A view of the parallel computing landscape, Commun. ACM 52 (10) (2009) 56–67.
[7] B. Atanasovski, S. Ristov, M. Gusev, N. Anchev, Educache simulator for teaching computer architecture and organization, in: Global Engineering Education Conference, EDUCON, 2013 IEEE, IEEE, 2013, pp. 1015–1022.
[8] E. Ayguade, R.M. Badia, D. Jimenez, J. Herrero, J. Labarta, V. Subotic, G. Utrera, Tareador: A tool to unveil parallelization strategies at undergraduate level, in: Workshop on Computer Architecture Education, 42nd ISCA, 2015, Portland, USA.
[9] M. Ben-Ari, A suite of tools for teaching concurrency, ACM SIGCSE Bull. 36 (3) (2004) 251–251.
[10] Bhaskar Chaudhury, Integrating parallel computing courses into the undergraduate programs in ICT and Computational Science, EduPar-16, 31st IEEE-IPDPS , Chicago, IL, USA, 2016 http://grid.cs.gsu.edu/~tcpp/.
[11] StevenA. Bogaerts, Limited time and experience: Parallelism in CS1, in: Parallel & Distributed Processing Symposium Workshops (IPDPSW), 2014 IEEE International, 2014.
[12] R. Brown, E. Shoop, J. Adams, C. Clifton, M. Gardner, M. Haupt, P. Hinsbeeck, Strategies for preparing computer science students for the multicore world, in: Proceedings of the 2010 ITiCSE Working Group Reports, ITiCSE-WGR 10, ACM, New York, NY, USA, 2010.
[13] M. Cosnard, Denis Trystram, Parallel Algorithms and Architectures, Thomson Learning, 1994.
[14] CSinParallel Project, (Retrieved 2016) Science Education Resource Center (SERC) at Carleton College http://csinparallel.org.
[15] A. Danner, T. Newhall, Integrating parallel and distributed computing topics into an undergraduate cs curriculum, in: Proc. Workshop on Parallel and Distributed Computing Education, EduPar13, 2013.
[16] C.T. Delistavrou, K.G. Margaritis, Towards an integrated teaching environment for parallel programming, in: Informatics (PCI), 2011 15th Panhellenic Conference on, IEEE, 2011, pp. 3–7.
[17] W.B. Gardner, J.D. Carter, Using the pilot library to teach message-passing programming, in: Education for High Performance Computing (EduHPC), 2014 Workshop on, IEEE, 2014, pp. 1–8.
[18] P. Garrity, T. Yates, R. Brown, E. Shoop, Webmapreduce: An accessible and adaptable tool for teaching map-reduce computing, in: Proceedings of the 42nd ACM Technical Symposium on Computer Science Education, SIGCSE11, ACM, NY, USA, 2011, p. 183188.
[19] A.Y. Grama, A. Gupta, V. Kumar, Isoefficiency: Measuring the scalability of parallel algorithms and architecture, IEEE Parallel Distrib. Technol. (1993) 12–21.
[20] Google Charts API (Retrieved 2017) http://developers.google.com/chart/.
[21] M. Gusev, S. Ristov, G. Velkoski, B. Ivanovska, E-learning and benchmarking platform for parallel and distributed computing, iJET 9 (2) (2014) 17–21.
[22] G. Hager, G. Wellein, Introduction to High Performance Computing for Scientists and Engineers, CRC Press, 2011.
[23] Shams Imam, Vivek Sarkar, Habanero-Java library: A Java 8 framework for multicore programming, in: Proceedings of the 2014 International Conference on Principles and Practices of Programming on the Java platform: Virtual machines, Languages, and Tools, 2014.
[24] Intel Courseware : High Performance Computing (Parallel Programming) Retrieved 2017, Intel Software Academic Program https://software.intel.com/en-us/courseware/hpc.
[25] C. Ivica, J.T. Riley, C. Shubert, Starhpcteaching parallel programming within elastic compute cloud, in: Information Technology Interfaces, 2009. ITI'09.

Proceedings of the ITI 2009 31st International Conference on, IEEE, 2009, pp. 353–356.
[26] L. John Gustafson, Reevaluating Amdahl's law, Commun. ACM 31 (1988) 532–533.
[27] L. John Gustafson, Fixed time, tiered memory, and superlinear speedup, in: Proceedings of the Fifth Distributed Memory Computing Conference, DMCC5, 1990, p. 1255.
[28] Joint Task Force on Computing Curricula, Association for Computing Machinery(ACM) and IEEE Computer Society, Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science, ACM, New York, USA.
[29] Let's HPC Team, Let's HPC Automated Scripts and Sample Code https://github.com/letshpcorg/letshpcsample, 2017.
[30] Let's HPC Project DA-IICT (Retrieved 2017) http://letshpc.org, http://letshpc.herokuapp.com.
[31] Timothy Mattson, Beverly Sanders, Berna Massingill, Patterns for Parallel Programming, Addison-Wesley, 2004.
[32] M. Nowicki, M. Marchwiany, M. Szpindler, P. Bała, On-line service for teaching parallel programming, in: European Conference on Parallel Processing, Springer, 2015, pp. 78–89.
[33] nVIDIA Educator Resources (Retrieved 2017) https://developer.nvidia.com/educators.
[34] OnRamp to Parallel Computing - an educational web portal for learning to use parallel systems (Retrieved 2017) OnRamp to Parallel Computing project https://github.com/OnRampOrg/onramp.
[35] D.A. Patterson, J.L. Hennessy, Computer Architecture A quantitative Approach, fifth ed., Morgan Kaufmann, 2011.
[36] Peter S. Pacheco, An Introduction to Parallel Programming, Elsevier, 2011.
[37] S.K. Prasad, A. Chtchelkanova, F. Dehne, M. Gouda, A. Gupta, J. Jaja, K. Kant, A. La Salle, R. LeBlanc, A. Lumsdaine, D. Padua, M. Parashar, V. Prasanna, Y. Robert, A. Rosenberg, S. Sahni, B. Shirazi, A. Sussman, C. Weems, J. Wu, NSF/IEEE-TCPP Curriculum Initiative on Parallel and Distributed Computing - Core Topics for Undergraduates, Version I, 2012 55 pages, www.cs.gsu.edu/~tcpp/curriculum/index.php.
[38] S. Prasad, A. Gupta, L., A. Rosenberg, A. Sussman, C.C. Weems, Topics in parallel and distributed computing: introducing concurrency in undergraduate courses, Elsevier Morgan Kaufmann, USA, 2015.
[39] Michael J. Quinn, Parallel Computing Theory and Practice, McGraw Hill Education, 2002.
[40] Michael J. Quinn, Parallel Programming in C with MPI and OpenMP, McGraw Hill Education, 2003.
[41] B.W. Rague, Exploring concurrency using the parallel analysis tool, in: Proceedings of the 43rd ACM technical symposium on Computer Science Education, ACM, 2012, pp. 511–516.
[42] Miller Russ, Algorithms Sequential & Parallel: A Unified Approach, Cengage Learning, 2013.
[43] Williams Samuel, Waterman Andrew, Patterson David, Roofline: An insightful visual performance model for multicore architectures, Commun. ACM 52 (2009) 65–76.
[44] Moritz Schlarb, Christian Hundt, Bertil Schmidt, SAUCE: A web-based automated assessment tool for teaching parallel programming, in: Euro-Par 2015: Euro-Par 2015: Parallel Processing Workshops, 2015, pp. 54-65.
[45] SeeMore project (Retrieved 2017) Understanding Parallel computing through Art http://www.seemoreproject.com/.
[46] Joseph Sifakis, A vision for computer science the system perspective, Cent. Eur. J. Comp. Sci. 1 (2011) 108–116.
[47] O. Sukhoroslov, Using everest platform for teaching parallel and distributed computing, in: European Conference on Parallel Processing, Springer, 2016, pp. 16–27.
[48] Hoefler Torsten, Belli Roberto, Scientific benchmarking of parallel computing systems: twelve ways to tell the masses when reporting performance results, in: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, 2015, p. 73.
[49] Undergraduate petascale modules (Retrieved 2017) Shodor Education foundation http://shodor.org/petascale/materials/modules/.
[50] A. Varma, Y. Keswani, Y. Bhatnagar, B. Chaudhury, Let's HPC: A web-based interactive platform to aid High Performance Computing education, 2017, arXiv preprint arXiv:1701.06356v1.
[51] Vipin Kumar, Ananth Grama, Anshul Gupta, George Karypis, Introduction to Parallel Computing: Design Analysis of Parallel Algorithms, Benjamin-Cummings Publishing Co., 1994.
[52] Wikipedia, MEAN (software bundle) —wikipedia, The Free Encyclopedia, https://en.wikipedia.org/w/index.php?title=MEAN_(software_bundle)&oldid=760144286 (Online; accessed 15-January-2017)..
[53] Y. Zhi, Y. Liu, L. Jiao, P. Zhang, A parallel simulator for large-scale parallel computers, in: Grid and Cooperative Computing (GCC), 2010 9th International Conference on, IEEE, 2010, pp. 196–200.

Bhaskar Chaudhury is currently an associate professor of Computational Science at DA-IICT, India. He completed his Ph.D. in Computational Physics at IPR, India in 2008. From 2008–2013, he worked as a researcher at LAPLACE Laboratory, CNRS, Toulouse, France. His research interests include Computational Science, High Performance Computing, Computational Electromagnetics, Computational Plasma Physics, Parallel programming models, Scientific Data analysis and Visualization. He has published more than forty research papers in world class journals/conference proceedings and delivered various talks/presentations worldwide. He is a reviewer of various prestigious international journals and the recipient of several international/national awards in the area of Computational Science.

Akshar Varma is a Ph.D. student in the College of Computer and Information Science at Northeastern University, focusing on Theoretical Computer Science. He got his bachelor's degree in Information and Communication Technology with a minor in Computational Science from DA-IICT, India. His primary research interests are in theoretical aspects of computer science, and in the past he has been interested in HPC and Computational Sciences.



Yashwant Keswani completed his B.Tech. Honours in Information and Communication Technology with a minor in Computational Science from DA-IICT, India and has research interests in Parallel and Distributed Computing, Semantic Web and Data Science.

Yashodhan Mohan Bhatnagar received the B.Tech. (Honours) in Information and Communication Technology with minors in Computational Science from DA-IICT, Gandhinagar in 2017. His research interests include applications of artificial neural networks on real-life optimization problems, computational systems biology, scalable software architecture analysis and signals systems analysis.

Samarth Parikh is pursuing his B.Tech. in Information and Communication Technology at DA-IICT and has research interests in Parallel and Distributed Computing and Data Science.