Teaching High Performance Computing through Parallel Programming Marathons

Leandro A. J. Marzulo[†], Calebe P. Bianchini[‡], Leandro Santiago[*], Victor C. Ferreira[*],
Brunno F. Goldstein[*], Felipe M. G. França[*]
[†]Instituto de Matemática e Estatística, Universidade do Estado do Rio de Janeiro (UERJ), Brazil
Email: leandro@ime.uerj.br
[‡]Faculdade de Computação e Informática, Universidade Presbiteriana Mackenzie, São Paulo, SP, Brazil
Email: calebe.bianchini@mackenzie.br
[*]Programa de Engenharia de Sistemas e Computação - COPPE, Universidade Federal do Rio de Janeiro (UFRJ), Brazil
Email: {lsantiago, vcruz, bfgoldstein, felipe}@cos.ufrj.br

***Abstract*—Parallel and distributed programming is essential for exploiting the processing power of modern computing platforms. However, during the first years of a Computer Science course, students usually learn problem solving techniques, data structures and programming paradigms that are inherently sequential, hindering the transition to parallel architectures. Parallel Programming Marathons organized in Brazil are similar to other Programming Competitions around the world and have been used for teaching and stimulating undergraduate and graduate students into learning to "think in parallel" and to develop applications for different parallel architectures, including multicores, clusters and accelerators. This paper presents the structure of this Parallel Programming Marathon and an overview of how it supports regional and national contests. Also, this work presents use cases on Parallel and Distributed Computing course from two different Brazilian universities that use a challenge based learning approach and employ marathon problems as course assignments. This approach contributed to increase students' interest towards High Performance Computing.**

***Index Terms*—High Performance Computing, Marathon of Parallel Programming, Parallel and Distributed Computing**

I. Introduction

As multicore and manycore systems emerged as an industry standard either for personal computer or hyperscale data centers, Parallel and Distributed Computing (PDC) courses have been increasingly included in Computer Science curriculum of both graduate and undergraduate programs. Recent advances on data-intensive computation rely on High Performance Computing (HPC) to enable scientific research on academia and industry, in areas such as biology [1], finance [2], physics [3], chemistry [4] and engineering [5]. Thus, an education that includes PDC is essential for Computer Science programs seeking to prepare HPC professionals [6].

On the other hand, there is a wide range of advanced parallel hardware technology available, and although it is possible to notice that some architectures and approaches are currently thriving (such as GPU programming), a single platform that could be used to solve any problem has not been established as a standard so far. Therefore, teaching HPC is a challenge for educators where they need to select relevant topics according to their expertise and course duration. HPC is not just a theoretical subject and we argue that learning by programming non-trivial projects in adequate parallel computer systems could be a very good approach. It requires students to learn several topics, such as parallel programming models, profiling, OpenMP, MPI, GPU computing and so on [7]. Moreover, most of Computer Science programs introduce algorithms, programming languages and software design concepts through sequential semantics in the first year. A mindset focused on sequential design and semantics makes conceptualizing problems in parallel difficult.

In order to promote High Performance Computing and disseminate parallel and distributed programming among undergraduate and graduated students, the Brazilian HPC academic community organizes regional, national and international events that include tutorials, keynotes and paper presentation. Some regional and national events host Parallel Programming Marathons, in which students can compete in parallelizing different problems. Brazilian parallel programming contests are inspired by traditional programming competitions, such as ACM International Collegiate Programming Contest, but the ranking system is focused on total speedup rather than number of problems solved. The contest usually provides multiple HPC environments as large multicore machines and mini-clusters including GPU and Xeon Phi accelerators, so that competitors can choose where to submit their parallel solutions. This initiative motivates students to join in the HPC community, while providing a fun environment for their first contact with real technology available in the industry.

In this work, we provide comprehensive information on how those Parallel Programming Contests are organized. Moreover, we present a catalogue of all problems present in the national contest since 2006 following the 13 "Dwarfs" concept. We assembled a repository with all those problems and made it available on GitHub[1], so that faculty members of other institutions can use those problems in their courses. Finally, we discuss our approach to teaching PDC courses using those problems in two different universities. Our experience show that our students increased their participation in both regional

[1]https://github.com/bfgoldstein/ppc

and national contests and showed more interest and ease to learn HPC subjects. Furthermore, some of our students were highly ranked in the Parallel Programming Marathons.

The rest of the paper is organized as follows: Section II introduces the organization of Parallel Programming Marathons along the years, how Brazilian HPC community encourage these contests, provides information on the problems and how they can be used in a PDC course; Section III presents the report of two use cases where the Parallel Programming Contest problems have been used to teach PDC. Finally, Section IV concludes this paper.

II. The Brazilian Parallel Programming Contest

Most of the computer science community in Brazil is organized under the Brazilian Computer Society (*SBC* - the acronym is in Portuguese). SBC is the largest computer society in South America, counting with a board of directors, seven regional chapters and a network of several institutional representation offices in universities and research institutions. One of the main goal of SBC are promoting conferences and events that may be of interest for the scientific computer community. SBC also keep up to date a curriculum recommendations and guidelines for computer courses [8]. Figure 1 shows how SBC and HPC community is organized in Brazil.

Research activities are fostered by 27 Special Interest Groups, among which is the Special Interest Group on Computer Architecture and High Performance Computing (*CE-ACPAD* - the acronym is in Portuguese). *CE-ACPAD* is responsible for overseeing and managing events and conferences on HPC and Computer Architecture on the National level. Two of these conferences are depicted at Figure 1, highlighting the Symposium on High Performance Computing Systems (*WSCAD* - the acronym is in Portuguese) that hosts the National Parallel Programming Contest that happens since 2006. WSCAD also hosts a Workshop on Computer architecture and HPC Education[2].

CE-ACPAD is also responsible for overseeing the 5 Brazilian Regional Committees in HPC called *CRAD*s. Each *CRAD* is responsible for expanding and promoting events on the regional level. The main event organized annually by each Regional Committees is the Regional Schools of HPC (*ERAD*s). These schools are usually 3-day events that focus more on student related papers (both on graduate and undergraduate levels). Most *ERAD*s also offer keynotes and mini courses to its participants. Moreover, 3 *ERAD*s (Rio de Janeiro, Rio Grande do Sul and São Paulo) feature Parallel Programming Contests in a similar format of the national contest. Those events have been successful in attracting students to the HPC community.

A. Environment Structure

Although the main structure of the WSCAD Parallel Programming Marathons has been the same since its first edition (2006), different environments have been used throughout the years. Usually the Marathons are organized in two day sessions. No internet access is allowed for the participants during the competition. A reward is usually given to the top 2 teams. The only requirement is that participants need to be students. [9]

The first day of competition is called Warmup day, in which students get to choose their teams and get to know the environment they will be competing on. Teams are composed of up to 3 members plus one substitute member. In the warmup phase, one problem is provided so that teams can practice and prepare for the main event. This approach enables for newcomers to become familiarized with the whole process and enables an initial experience where students from different Universities may interact.

From the year of 2015 and onward, students have been allowed to participate remotely, as long as they have previously registered. Although their score is updated live on the scoreboard, they are ineligible to receiving rewards.

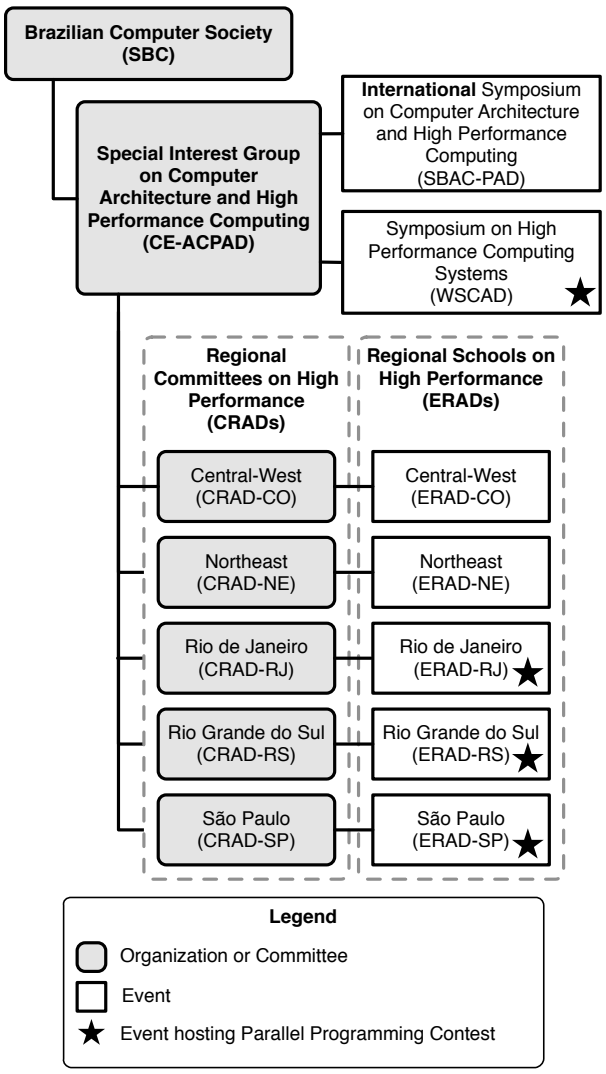


Fig. 1. HPC Community organization in Brazil, including events and parallel programming contests

[2]Site in Portuguese: http://wscad.sbc.org.br/current/programa-weac.html

Each team is provided with one computer and a configured environment. These local machines are used for testing solutions with smaller inputs before submitting a solution for evaluation. Sometimes a remote machine is also provided for testing distributed applications. When the team desires to submit a version for evaluation, it must select an environment (when multiple environments are available) and upload the solution. A team can submit solutions multiple times, but only the most recent correct solution for each problem is considered for scoring purposes. The submitted solution is compiled and executed with a large input in a Judge Machine, which is usually an HPC environment.

Brazil has a long tradition on competing at the ACM-ICPC International Collegiate Programming Contest, which is a regular (non-parallel) programming marathon. The Brazilian Online Contest Administration system (BOCA) [10]–[12] has been developed to help judges and students to communicate through the whole ACM-ICPC contest. It provides a stable interface so that teams can submit solutions and get notified about their results. A modified version of BOCA was developed to comply with the peculiarities of a parallel marathon. The basic modifications involve the scoring system (discussed in Section II-B), the submission page and the auto-judge.

Figure 2 depicts the submission page of Parallel BOCA for a certain team. In the first section of that page, all previous submission for that team are listed in a table, where each line represents a submission and each column provides a feedback containing an answer, speedup and output so that teams can manage their time and strategy during the contest. Submissions where the speedup is in fact a slowdown are not take into account. Correct answers are marked with a "YES" message followed by a colored balloon and wrong ones are marked with a "NO" followed by a debug message. Below the table, there is a submission form, were users can select the Problem, the Environment/Language and upload the *.zip* file with the source code and *Makefile*.

BOCA provides an auto-judge functionality that queues and executes submissions. The system automatically outputs an answer on the scoreboard for each submission leaving the responsibility of reviewing potential execution errors to Judges (Figure 3). The auto-judge is basically a daemon running on each Judge Machine that continually fetches new submissions from BOCA, execute them and sends the result back to BOCA. Parallel BOCA auto-judge had to be adapted to each environment used in the competition.

B. Score

The score in a traditional programming contest considers the number of problems solved and the time-stamp of submission. In case two teams solve the same number of problems, the time-stamp is used as a tiebreak (the team that solved problems first wins). On the other hand, a parallel programming marathon is all about performance, meaning speedups are used to calculate scores.

The score of each team is calculated by the Equation 1

$$Score = \sum_{i=1}^{P} \frac{Prob_i}{Sub_i} \quad (1)$$

where P is equals the number of problems, Sub_i and $Prob_i$ are the time it takes to run a parallel and a sequential version of the problem i respectively. Since students already have the sequential version of each problem, all teams start with *Score* equals to the number of problems.

After a submission, the resulting speedup is calculated and the live scoreboard updated. Figure 4 shows an example of BOCA's scoreboard where each row represents a team and each column shows which team solved a certain problem. Notice that teams do not need to solve all problems to win allowing them to use different strategies during the contest. One team may use naive parallelization techniques to solve more problems with speedups per problem, while others can try to implement a complex GPU kernel in order to obtain a high speedup in just one application. The problem type may also be taken into consideration, since some applications might have a higher speedup potential than others. Thus, a trade-off between coding effort and maximum speedup needs to be taken into consideration by the team strategy.

C. Problem Structure

During each day of competition, a problem set with more than 4 problems is posed to the students. Different from usual Programming Contests, the Brazilian Parallel Programming Marathon provides the sequential code for each problem making teams responsible only for their parallelization. As seen from Figure 5, each problem has a description, an input sample and the expected output. That helps students to understand the problem and source code. It is important to mention that major changes to the main code, like switching the given sequential algorithm for a faster sequential implementation, are usually not allowed unless it is necessary for a better parallelization of the same problem.

D. Statistics

Parallel programming has evolved over the past years with the emergence of new accelerators such as GPU and Intel® Xeon Phi™. Contests followed the same path by introducing these new technologies in their environment and enabling teams to achieve higher speedups in their solutions. Table I points out those environment changes by showing the number nodes, CPU cores, CUDA cores and Xeon Phi cores over the years. In an early stage (2006-2011) the number of nodes and cores are high due to no extra technologies other than multicore nodes. From 2012 to 2015, the number of cores and nodes decreased since most contests have GPUs and Intel® Xeon Phi™ available as submission targets. The boom happened in 2016 where the number of (cores, nodes) went from (2, 32) to (130, 5080). Table I also summarizes the number of problems, attendees and from which universities the winner teams are. It is important to mention that some winner

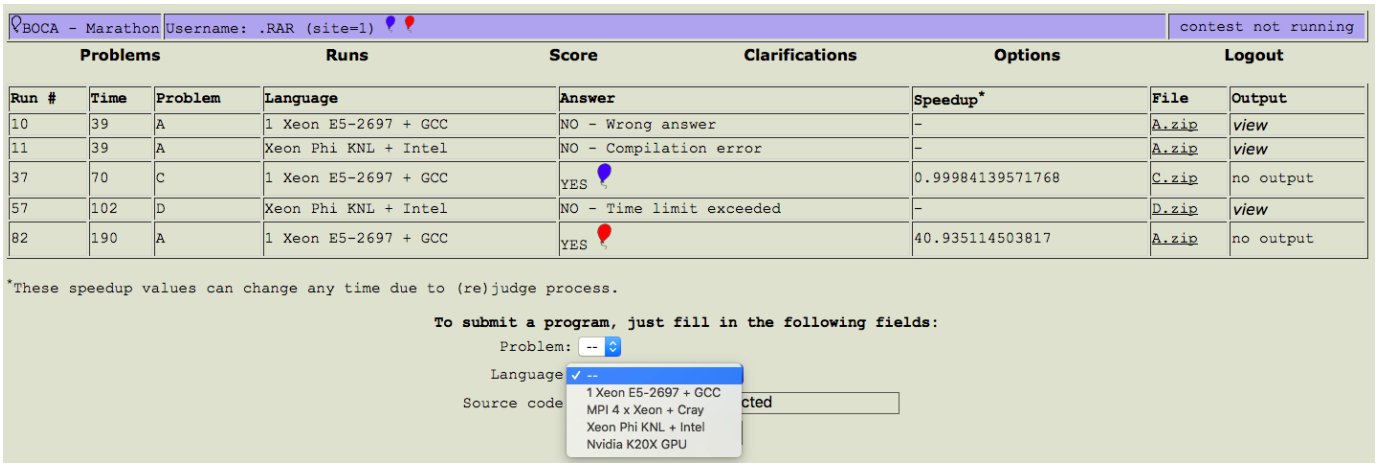


Fig. 2. Submission Page (Team View) at the Parallel BOCA System.

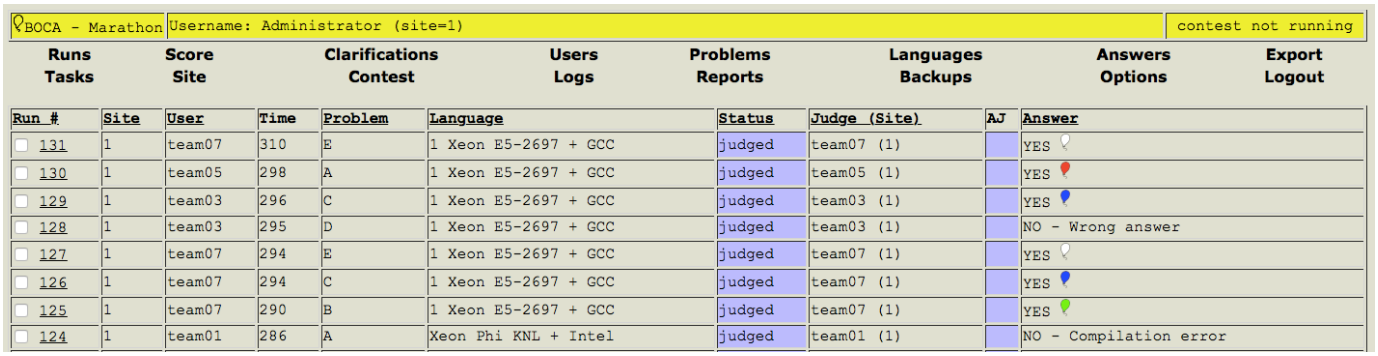


Fig. 3. Judge View at the Parallel BOCA System.

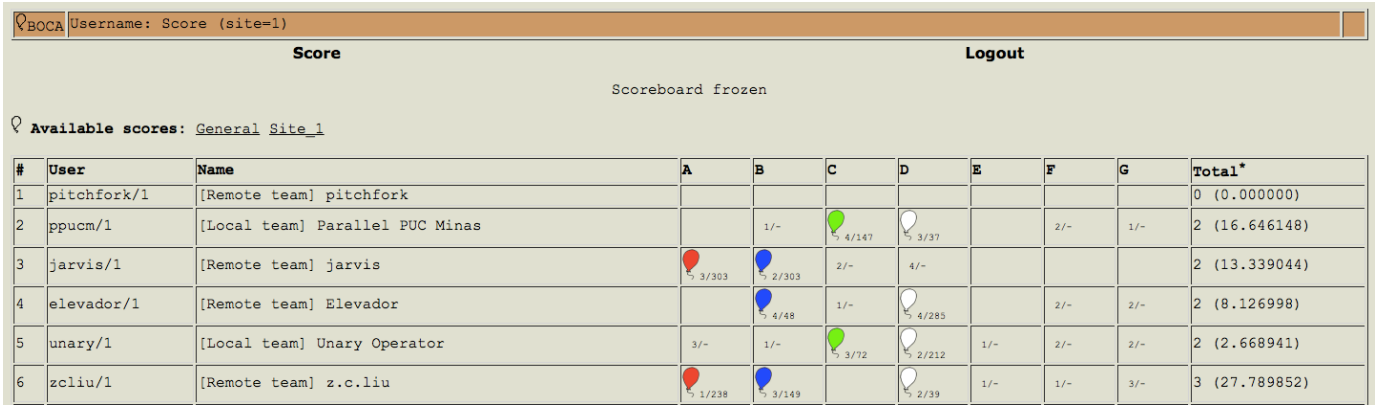


Fig. 4. BOCA System Scoreboard.

teams are composed by students from different universities showing that contests help students to interact with each other.

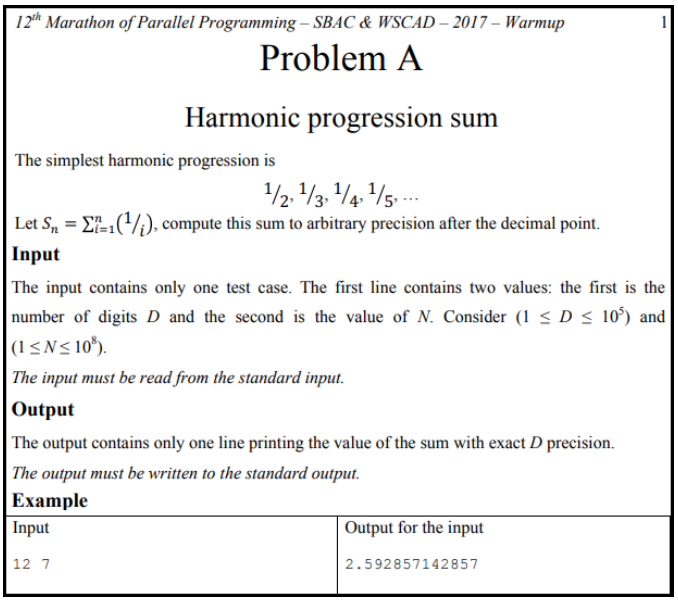
E. Problems Classification

Contests were created mostly to encourage and hone students skills on parallel programming by applying well known sequential problems that usually require different approaches to be solved in parallel. To understand how those problems demand a good notion of parallel programming models we classify each one into a "Dwarf" class. Asanovi **et al** [13] propose a set of classes called "Dwarfs" that captures a pattern of computation and communication of a parallel application. There are up to 13 "Dwarfs" listed in Table II.

Table III summarizes the set of problems with their names, the year of contest it was used and dwarf classes. Since some of them can fit into more than one class we extend it in two columns (D#1 and D#2). This classification helps which one of the 65 problems can be used when adopting topics

TABLE I
WSCAD MARATHON ENVIRONMENTS AND RESULTS ALONG THE YEARS.

Year	Problems	Attendees	1st	2nd	3rd	nodes	CPU cores	CUDA cores	Xeon Phi cores
2006	8	-	-	-	-	-	-	-	-
2007	-	24	UFMG	PUCRS, UFMG	UFJF	-	-	-	-
2008	8	-	UFMG	UFRGS	UFMS	-	-	-	-
2009	6	23	UFRGS	USP	UFPE	10	80	0	0
2010	5	37	USP	UFRGS	UFV	30	240	0	0
2011	5	13	UFRGS	UFV	UFRGS	30	120	0	0
2012	5	13	UFF	UNIPAMPA	UNIPAMPA	4	64	5376	-
2013	6	11	PUCMG, UFMG	PUC Minas	UNIPAMPA, UFRGS	4	64	2496	117
2014	6	15	PUCMG	PUCMG	UNIFESP	4	64	2496	117
2015	7	23	PUCMG	PUCMG	UFRGS	2	32	0	117
2016	4	35	UERJ	UFF	UNIPAMPA, UFRGS	130	5080	21504	122
2017	5	27	UFSM, UNIJUI, UNIPAMPA	UFSC	UERJ	14	560	5568	64



12th Marathon of Parallel Programming – SBAC & WSCAD – 2017 – Warmup 1

Problem A

Harmonic progression sum

The simplest harmonic progression is

$$1/2, 1/3, 1/4, 1/5, \cdots$$

Let $S_n = \sum_{i=1}^{n}(1/i)$, compute this sum to arbitrary precision after the decimal point.

Input

The input contains only one test case. The first line contains two values: the first is the number of digits D and the second is the value of N. Consider $(1 \leq D \leq 10^5)$ and $(1 \leq N \leq 10^8)$.

The input must be read from the standard input.

Output

The output contains only one line printing the value of the sum with exact D precision.

The output must be written to the standard output.

Example

Input	Output for the input
12 7	2.592857142857

Fig. 5. A PDF page from the WSCAD 2017 Warmup, presenting the Harmonic progression sum problem.

TABLE II
DWARFS NAMES AND IDS.

Dwarf	ID
Dense Linear Algebra	1
Sparse Linear Algebra	2
Spectral Methods	3
N-Body Methods	4
Structured Grids	5
Unstructured Grids	6
MapReduce	7
Combinational Logic	8
Graph Traversal	9
Dynamic Programming	10
Back-track and Branch Bound	11
Graphical Models	12
Finite State Machine	13

from IEEE/TPPC curriculum [14] in a problem-based learning course.

Figure 6 also shows the percentage of each class considering all problems since the first contest edition. Today, the Marathon of Parallel Programming database [15] has 23.3% *Back-track and Branch Bound* problems followed by *Graph Traversal* and *Dense Linear Algebra* with 21.7% and 15% respectively. *Sparse Linear Algebra*, *Unstructured Grids* and *Graphical Models* have 1.7% each, being the least frequent problem classes in contests.

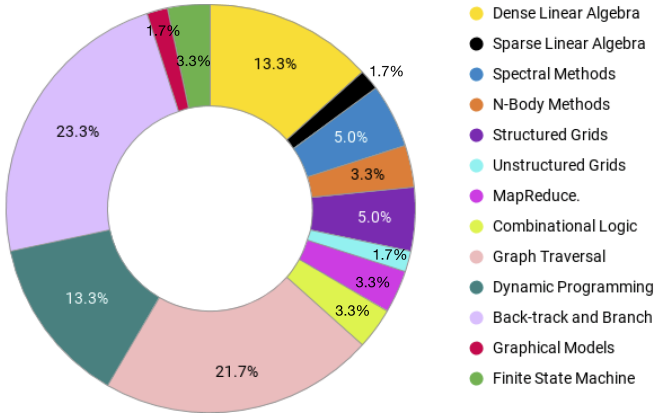


Fig. 6. Percentage of the 13 "Dwarfs" at the contest database.

III. USE CASES

In this section we present use cases of Parallel and Distributed Computing Courses offered by two Brazilian universities. Moreover, we discuss how Parallel Programming Marathons influenced our teaching approach and how it motivated students to engage the HPC community.

A. State University of Rio de Janeiro

The Parallel and Distributed Programming course is offered to both undergraduate (elective) and graduate students, usually

TABLE III
CONTESTS PROBLEMS ALONG THE YEARS AND THEIR DWARF CLASSES.

Problem	Year	D#1	D#2
0-1 knapsack	2006	11	-
3SAT	2009	8	-
A Graph's Maximal Independent Set	2014	5	-
Binary Heaps and Priority Queues	2008	9	-
Binary Search Tree	2006	9	-
Bitonic Sort	2015	11	-
Black-Scholes	2013	7	-
Blum Blum Shub Random Num. Gen.	2013	7	-
Bucketsort	2012	11	-
Clustering Coefficient for a Graph	2008	9	11
Color Histogram	2016	12	-
Component Labeling	2006	1	6
Convex Hull	2006	10	-
Dijkstra	2014	9	-
DNA Subsequences	2009	11	-
Enumeration Sort	2014	10	-
Eternity II	2017	5	-
Fibonacci Numbers	2009	10	11
Finding the Roots of a Forest	2008	9	-
Game of Life	2006, 2016	4	-
Gauss Elimination	2010, 2013	1	-
Haar Wavelets	2012	1	-
Harmonic Progression Sum	2006	10	-
K-Means Clustering	2015, 2016	1	11
Karatsuba Multiplication	2015	11	-
LCS	2013	1	10
Leibniz's PI	2011	3	-
Lossless Text Compression	2015	3	9
Loyd's tile Puzzle	2008	11	-
LU Decomposition	2015	1	-
Machin's	2009	3	-
Mandelbrot	2011, 2017	10	-
Matrix Multiplication	2010	1	-
Maximum Clique	2006	9	-
Median of a Set	2008	10	-
Minimum Spanning Tree	2011	9	-
Minimum Weight Polygon Decomp.	2014	11	-
MSD Sorting	2009	9	11
Mutually Friendly Numbers	2012	10	-
N-Body	2015	4	-
PageRank	2008	11	-
Permutation Flowshop Scheduling	2011	11	-
Prefix Sum	2008	9	-
Quicksort	2010	9	-
Raytracer	2015	6	-
Selection	2013	11	-
Shellsort	2011	11	-
Shortest Superstring	2017	13	-
Smooth	2010	1	-
Software Testing	2009	9	-
Sparse Matrix Multiplication	2006	2	-
String Parsing	2016	13	-
Sudokount	2016	5	-
The Energy Min. Cooking Problem	2014	11	-
The Knight's Tour Problem	2013	11	-
The Sieve of Eratosthenes	2008	10	-
The Traveling-Salesman Problem	2010, 2014	9	11
Transitive Closure	2017	9	-
Unbounded Knapsack Problem	2012	11	-
We're Back: 3SAT	2012	8	-

in separate classes. The goal of this course is to teach tools and techniques used to devise and implement parallel and distributed applications. Moreover we focus on practical aspects related to execution and performance evaluation of such applications. This course covers Threads and Processes, synchronization and deadlocks, Amdahl's and Gustafson's laws, OpenMP, Pthreads, MPI and also introduce CUDA and Xeon Phi programming. For the undergraduate course, the prerequisites are Operating Systems, Computer Architecture and C Programming. For the graduate course, since students might have different backgrounds, we offer review classes on the first two weeks.

In 2014 professor Leandro Marzulo started teaching this course and decided to use problems from the *WSCAD* Marathon as assignments and using a classic textbook [16], along with a set of textbooks that addopt a hands on approach [17]–[19]. We start with a toy application that has a set of input parameters to the amount of computation per task, which is used to make the parallelization overhead more evident to students. This application can also become unbalanced depending on its inputs, which is interesting to discuss the need of load balancing or scheduling techniques (in the case of OpenMP we use this example to discuss the `schedule` clause).

The first assignment is to implement a naive prime number counter (just a loop calling a primality test function for each number in the interval). This application is straightforward to parallelize, but allows students to play with the problem size (which influences task grain). This application also has a natural unbalance that needs to be tackled by students, since primality test can take more time to run for larger numbers on the interval. Students have to profile the given sequential solution and implement parallel solutions with different balancing mechanisms for shared memory (OpenMP and Pthreads). Finally, they have to run all their parallel versions and produce a report with the results, including speedup charts.

After students are familiar with the basics of parallel programming we select a set of Marathon problems (usually 3) and discuss the sequential solutions in the classroom. We also start discussing possible parallel solutions and ask them to implement different versions. Students should present a speedup report comparing all solutions for each problem.

One assignment example is the "Fibonacci numbers" (2009 Marathon) that calculates a desired term (N) of the sequence, N being very large. The sequential solution uses vectors to represent a term of the sequence, since a large term that would not fit a `long long` variable. Therefore, this solution implements an addition algorithm to use that data structure. We inform students that they can't use a different method to calculate the Fibonacci numbers (just the classic dynamic programming solution) and they should find a way to build a parallel solution for this classic method. A possible path would be to try to parallelize the addition algorithm, using a carry look ahead technique. A different and more challenging solution would be to use a Residue Numeral System (RNS) that would allow us to break the numbers in parts (modules) and make real parallel adders. All those assignments are made in groups of up to 3 students.

On the next step, students learn MPI, also employing a set of toy examples, and then will need to solve the same set of

problems from the previous phase with message passing. It is important noticing that our classroom has 40 computers (one per student) and those computers can be arranged in multiple clusters. Since we usually have up to 20 students, each group can have about 6 computers to develop their MPI solutions in class, interacting with the Professor. If students need more time, they can finish it at home or arrange access to the classroom during office hours.

Since we do not have access to GPU or Xeon Phi accelerators to all students, we just give them an overview of those technology and the Professor solves some problems in class, running the solutions on machines that belong to other partner institutions. Although it would be better if students had access to such systems so we could have assignments focused on accelerators, we think this first contact with those technologies can motivate students to seek for more information on their own or to engage in research projects in HPC.

The effects of our approach were positive from the very first year. In 2014 a group of undergraduate students published the paper [20] on *WSCAD* based on one of their assignments. They also engaged in research projects related to HPC. One of this students is now a PhD student on our group (and a co-author of this work). In 2015 a group of students from this course attended the first *ERAD-RJ* and our teams got 2nd and 3rd place on the *ERAD-RJ* Marathon. One of those students is now a master student in our group and another one is currently a PhD student and also an co-author of this work. In 2016, besides publishing on *ERAD-RJ* and *WSCAD* our students got the 1st place in both Marathons. In 2017 our students got the $3rd$ place at the *WSCAD* Marathon as shown in Table I.

B. Mackenzie Presbyterian University

Professor Calebe Bianchini has been using marathon problems in his undergraduate courses on PDC at Mackenzie Presbyterian University, São Paulo since 2016. This approach has been showing interesting results regarding the engagement of student on the HPC community in Brazil. In general, during his lectures, he presents several problems and categorizes them according to the adopted textbooks [21]–[24]. Note that the categories are not the same as the 13 "Dwarfs". Those categories represent the following algorithm classes to be studied:

- Matrix Multiplication
- System of Equations Solver
- Sorting
- Graph Search
- Search and Optimization
- Dynamic Programming
- Numeric Methods & FFT

The PDC course at Mackenzie University is organized in 15 weeks. Each week has 1 lecture and 1 practice at the lab. Besides that, the student must do some extra exercises at home (based on one of the textbooks). This course has 2 tests (worth 5.0); 15 practice labs (worth 1.5) and 1 final project (worth 3.5).

There is an initial lecture when an introduction to parallel computing is discussed. There are more 2 lectures that discusses parallel computer architectures and parallel programming models. At this point, it is possible to show more complex examples, such as matrix multiplication with memory hierarchy and blocking. All these lectures have an corresponding practical lab, that allow students to understand details associated with code compilation on current architectures, and a homework allowing students to experiment different solutions to the same problem. The next 5 lectures discuss about how to decompose an algorithm to a parallel architecture. Based on Flynn's taxonomy, we discuss how to use SIMD/SIMT and MIMD (shared memory) to achieve better performance, using Amdahl's and Gustafson's laws to plot some graphs. We also use OpenMP at the practice labs and homeworks.

The last 7 lectures follow the algorithm classes presented before. On each lecture, after understanding one class, a set of basic (sequential) solutions are discussed along with their space and time complexities. Notice that most of those problems are seen in previous programming courses and this first part can be also considered as a big review. However, we take this opportunity to dive into important concepts related to performance, such as memory usage (in-/out-of-place, cache hierarchy, memory alignment, etc) and efficient processor usage (pipelining, data dependencies, etc). The Parallel Programming concepts are also presented on the next step, where the goal is to redesign the sequential algorithms in a parallel fashion and understand the complexities. Students then proceed to implement the new solutions to efficiently run on top of parallel architectures.

The final course project is based on those studies, where students form groups and use the BOCA system to incrementally develop and test their solutions. Students must handle a detailed report containing tables showing the execution times and charts for the speedups and efficiency, according to Amdahl's e Gustafson's laws for strong and weak scalability. Students must also provide a discussion on their parallel algorithm, including implementations aspects related to the adopted technology. At the first term of 2018 we had 45 students, 27 groups (each having at most 3 students), 1136 submissions (443 correct answers) and the maximum obtained speedup was 106.77 (for an Intel®Xeon Phi™ Processor 7250).

All that effort to improve our Parallel Programming course stimulated our students to engage the HPC community and attend to the main HPC events in Brazil and the State of São Paulo. Since 2016, when we first tried that approach, the number of attendees at *ERAD-SP* has been constantly increasing and Mackenzie students were the second largest group of participants at 2018. However, the participation on the National Marathon (*WSCAD*) is still small and it is usually through "remote teams".

IV. Conclusion

In this paper we have presented an overview on how the Brazilian HPC community is structured, as well as, an introduction to Brazilian Parallel Programming Marathons. With the support provided by the SBC and its instances, a competitive environment that incentives students of multiple universities to study HPC has been created though events like WSCAD and ERADs.

Parallel Programming Marathons have been happening throughout the years on multiple Brazilian conferences. We have depicted the problems used during each WSCAD events and classified them according to the 13 Dwarfs taxonomy. This allowed for a better understanding on which class of problems Marathons usually focus on and also show it's diversity.

Professors from two different universities also presented their view on how the Parallel Programming Marathons affects their way of teach HPC. For UERJ, Professor Leandro Marzulo has implemented a Marathon problem-based approach that has increased student interest as well as great ranking in several Parallel Programming Marathons. In Mackenzie University, Professor Calebe Bianchini has also used multiple Marathon related problems, along with the BOCA system, resulting on an increase in student participation on multiple Brazilian conferences.

This work raises a series of opportunities of future contributions: (*i*) we intend to use the Git repository[3] to upload the best solutions devised by our students in future editions of our parallel programming courses; (*ii*) UERJ intends to start using the parallel version of BOCA for our assignments, so we can collect statistics and provide better information on the progress of our students; (*iii*) we intend to get in touch with Professors from other Institutions involved in the organization of the ERADs to obtain more use cases and get a national panorama on the educational efforts on HPC; and (*iv*) we intend to use the repository of problems and the BOCA system to promote local Marathons and create a training program for universities.

Acknowledgment

The authors would like to thank CAPES, CNPq and FAPERJ for the financial support to this work.

References

[1] B. Langmead, "Practical software for big genomics data," in *2013 IEEE 3rd International Conference on Computational Advances in Bio and medical Sciences (ICCABS)*, June 2013, pp. 1–1.

[2] M. Smelyanskiy, J. Sewall, D. D. Kalamkar, N. Satish, P. Dubey, N. Astafiev, I. Burylov, A. Nikolaev, S. Maidanov, S. Li, S. Kulkarni, C. H. Finan, and E. Gonina, "Analysis and optimization of financial analytics benchmark on modern multi- and many-core ia-based architectures," in *2012 SC Companion: High Performance Computing, Networking Storage and Analysis*, Nov 2012, pp. 1154–1162.

[3] M. J. Abraham, T. Murtola, R. Schulz, S. Pll, J. C. Smith, B. Hess, and E. Lindahl, "Gromacs: High performance molecular simulations through multi-level parallelism from laptops to supercomputers," *SoftwareX*, vol. 1-2, pp. 19 – 25, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2352711015000059

[4] W. A. de Jong, E. Bylaska, N. Govind, C. L. Janssen, K. Kowalski, T. Mller, I. M. B. Nielsen, H. J. J. van Dam, V. Veryazov, and R. Lindh, "Utilizing high performance computing for chemistry: parallel computational chemistry," *Phys. Chem. Chem. Phys.*, vol. 12, pp. 6896–6920, 2010. [Online]. Available: http://dx.doi.org/10.1039/C002859B

[5] T. C. Martins", J. M. Kian", D. K. Yabuki", and M. S. G. Tsuzki", "Gpu accelerated reconstruction of electrical impedance tomography images through simulated annealing," *Blucher Mechanical Engineering Proceedings*, vol. 1, no. 1, pp. 762 – 779, 2014.

[6] J. Wang, M. K. Gobbert, Z. bo Zhang, A. Gangopadhyay, and G. G. Page, "Multidisciplinary education on big data + hpc + atmospheric sciences," in *2017 Workshop on Education for High-Performance Computing (EduHPC)*, 2017, p. 8.

[7] A. Asaduzzaman, R. Asmatulu, and M. Rahman, "Article: Teaching parallel programming for time-efficient computer applications," *International Journal of Computer Applications*, vol. 90, no. 7, pp. 18–25, March 2014, full text available.

[8] A. F. Zorzo, D. Nunes, E. Matos, I. Steinmacher, J. Leite, R. M. Araujo, R. Correia, and S. Martins, *Referenciais de Formao para os Cursos de Graduao em Computao*, 1st ed. Porto Alegre, RS, Brazil: Sociedade Brasileira de Computao, 2017.

[9] C. P. Bianchini, "Wscad 2017 marathon rules." [Online]. Available: http://lspd.mackenzie.br/marathon/17/rules.html

[10] C. P. De Campos, C. E. Ferreira, and R. Anido, "Brazilian contest infrastructure: Boca and maratona linux," 2010.

[11] C. P. De Campos and C. E. Ferreira, "Boca: um sistema de apoio a competições de programação," in *Workshop de Educação em Computação*, 2004, pp. 885–895.

[12] C. P. De Campos and B. Cesar Ribas, "Boca github." [Online]. Available: https://github.com/maratona-linux

[13] K. Asanovi, R. Bodik, B. C. Catanzaro, J. J. Gebis, P. Husbands, K. Keutzer, D. A. Patterson, W. L. Plishker, J. Shalf, S. W. Williams, and K. A. Yelick, "The landscape of parallel computing research: A view from berkeley," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2006-183, Dec 2006. [Online]. Available: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-183.html

[14] 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, and J. Wu, "Nsf/ieee-tcpp curriculum initiative on parallel and distributed computing - core topics for undergraduates," IEEE/Technical Committee on Parallel Processing, Tech. Rep. I, 2012. [Online]. Available: http://www.cs.gsu.edu/~tcpp/curriculum/

[15] L. Marzulo, C. Bianchini, V. Ferreira, L. Santiago, B. Goldstein, and F. França, "Brazilian parallel programming contest github." [Online]. Available: https://github.com/bfgoldstein/ppc

[16] M. Herlihy and N. Shavit, *The Art of Multiprocessor Programming*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2008.

[17] D. B. Kirk and W.-m. W. Hwu, *Programming Massively Parallel Processors, Third Edition: A Hands-on Approach*, 3rd ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2016.

[18] J. Jeffers and J. Reinders, *High Performance Parallelism Pearls Volume One: Multicore and Many-core Programming Approaches*, 1st ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2014.

[19] ——, *High Performance Parallelism Pearls Volume Two: Multicore and Many-core Programming Approaches*, 1st ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2015.

[20] A. M. Lauredo, J. S. de Mesquita, L. Santiago, M. C. Castro, A. Sena, and L. Marzulo, "Implementação e avaliação de desempenho do lcs paralelo em cluster multicore," in *WSCAD 2014 - XV Simpósio em Sistemas Computacionais de Alto Desempenho*. Editora UFMS, 2014, pp. 27–38.

[21] V. Kumar, *Introduction to Parallel Computing*, 2nd ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2002.

[22] T. Rauber and G. Rnger, *Parallel Programming: For Multicore and Cluster Systems*, 2nd ed. Springer Publishing Company, Incorporated, 2013.

[23] C. Breshears, *The Art of Concurrency: A Thread Monkey's Guide to Writing Parallel Applications*. O'Reilly Media, Inc., 2009.

[24] M. McCool, J. Reinders, and A. Robison, *Structured Parallel Programming: Patterns for Efficient Computation*, 1st ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2012.

[3] https://github.com/bfgoldstein/ppc