Computational Biology as a Compelling Pedagogical Tool in Computer Science Education

Vijayalakshmi Saravanan
Rochester Institute of Technology
Rochester, USA
vsavse@rit.edu

Anpalagan Alagan
Ryerson University
Toronto, Canada
alagan@ee.ryerson.ca

Kshirasagar Naik
University of Waterloo
Waterloo, Canada
snaik@uwaterloo.edu

ABSTRACT
High-performance computing (HPC), and parallel and distributed computing (PDC) are widely discussed topics in computer science (CS) and computer engineering (CE) education. In the past decade, high-performance computing has also contributed significantly to addressing complex problems in bio-engineering, healthcare and systems biology. Therefore, computational biology applications provide several compelling examples that can be potent pedagogical tools in teaching high-performance computing. In this paper, we introduce a novel course curriculum to teach high-performance, parallel and distributed computing to senior graduate students (PhD) in a hands-on setup through examples drawn from a wealth of areas in computational biology. We introduce the concepts of parallel programming, algorithms and architectures and implementations via carefully chosen examples from computational biology. We believe that this course curriculum will provide students an engaging and refreshing introduction to this well-established domain.

Keywords
Pedagogical Tools · High-Performance Computing (HPC) · Parallel and Distributed Computing (PDC) · Computational Biology.

1. INTRODUCTION
Over the last few years, computational biology has revolutionized medical research, bringing in novel analysis tools that accelerate diagnosis and drug discovery. The enormous amount of experimental data generated by the human genome project, proteomics, and clinical research has fostered this revolution by enabling extremely accurate, albeit complex models for various biological phenomena. The analysis of these models requires high processing power and time to find accurate solutions, making them attractive candidates for parallelization. For example, high-performance parallel computing has successfully contributed to the understanding of protein dynamics [1], ion channels and cellular reaction kinetics [2], resulting in several specialized high-throughput tools such as GROMACS, a parallelized molecular simulation toolkit [3]. Further, novel projects such as Folding@home [4] have enabled the pooling of distributed computing resources from around the world to analyze proteins. Recently, bioengineers have begun focusing on reverse engineering biological systems, by reconstructing gene and metabolic networks that describe the interactions between various genes and protein from experimental data. This relatively new area of research requires novel computational tools due to the vastly heterogeneous nature of the data involved [5]. While computer scientists have been able to contribute to improving the performance and accuracy of biological analysis, the striking applications found in the domain can also serve to provide a wealth of motivation for computer scientists. In addition, there are several different methods of implementing these applications, some more easily parallelized than others did (and the best implementation can depend on the application). Therefore, they provide an excellent opportunity for computer science students to gain insight into issues faced by programmers of parallel algorithms. For this reason, we believe that biomedical applications can be a powerful tool in teaching parallel and distributed computing. In this paper, we propose a novel course curriculum that introduces parallel and distributed computing to senior graduate students in a hands-on manner through a set of carefully chosen computational biology applications. We also propose several sample research term projects that can be carried out as a direct extension of the learning outcomes of this course.

1.1 Contribution and Related Work
The ACM and NSF/TCPP guidelines recommend that parallel computing is introduced in CS and CE courses from early stages [28][29]. As parallelism and multi-core computing becomes more accessible, academic institutions in India are exploring the introduction of interdisciplinary concepts in CS and CE education. In this context, several courses have been developed to teach the parallel computing programming concepts with real-world examples [30] [31] [32] [33]. The first author has also introduced a course teaching parallelism with hands-on experimental learning activities as a member of the Board of Studies (BoS)/Curriculum Design Committee at Amrita/VIT University, India in 2005-2009. In this course, the author piloted a new course introducing certain concepts in HPC and PDC using real-world applications, including those in computational biology. Drawing upon this experience, the key contribution of this paper is the design of an interdisciplinary course curriculum that uses problems in computational biology as educational tools in computer science education. Currently, several courses designed for biology majors focusing on the fundamentals of parallel and distributed computing [6] [7]. Recently, courses incorporating high-performance computing for medical applications have also been developed [8]. Advanced courses in computational biology have also been targeted towards CS graduate students specializing in biological computation [9]. However, there has been little attention to the pedagogical value that computer scientists can draw from the biological application domain. The curriculum proposed in this paper fills this gap, while bringing in several advantages. First, it serves to provide students with an insight on programming choices regarding how much parallelization is required, based on the application. Second, it promotes interdisciplinary thinking among computer science graduate students and will be particularly valuable to

computer scientists who wish to make a career transition into the computational biology domain. Finally, this course does not require expensive parallel computing resources, and can easily be taught using FPGAs and commonly available desktop/laptop GPUs. Further, the use of biological problems such as protein folding as examples to teach parallel computing enables the use of a wealth of tools that pool worldwide distributed computing resources such as the Folding@home project [4].

2. HPC IN COMPUTATIONAL BIOLOGY

While several fields can be used to supply examples for the application of HPC and PDC, computational biology provides a large variety of problems that are both complex and challenging even at a research level. This diversity in the availability of HPC and PDC applications in computational biology is the primary reason for choosing this domain. In addition, since the course is taught to senior graduate students, we hope that the large number of open problems in this emerging area will inspire students to explore this interdisciplinary area for their research.

Over the last two decades, there have been significant advances in biomedical sciences, computational biology, drug discovery and systems biology with the introduction of high-performance computing technologies. The tremendous increase in computational power of the desktop to high-end computing devices such as supercomputers, clusters, and grids due to the end of Dennard scaling and Moore's law has opened up great opportunities in the simulation of relevant biological systems for applications in bioinformatics and computational biology. In this section, we introduce the applications of high-performance systems in computational biology. Among them, we introduce relevant biomedical problems such as heterogeneous computing, GPU architecture and large-scale distributed computing has been successfully applied [11]. We also point out the pedagogical value associated with these problems for computer science students. We will later draw upon these areas to construct specific application examples to teach high-performance computing.

2.1 Big Data Analytics

Owing to advances in genome projects, proteomics, high-resolution imaging and the rapid digitization of patient clinical records, there has been an explosion in the volume of biological and medical data sets. The optimal use of this data is a major challenge in biomedical research, requiring the development of more sophisticated architectures and tools. A key question that remains in computational biology research is how to extract information, construct models and reverse-engineer biological networks from the massive amount of vastly heterogeneous data [5]. However, dealing with biological networks, while extremely effective, is also computationally intensive. Here, the integration of big data tools with high-performance and parallel processing techniques have been proposed [10]. From a pedagogical perspective, biomedical big data applications can be effective in teaching efficient parallelization and introducing massively parallel processing (MPP) tools.

2.2 GPU Computing

The recent explosion in the availability of cheap graphical processing units (GPU) from PCs to heterogeneous computing platforms where they perform massively parallel have introduced a new facet of high-performance computing. This fact has attracted many researchers to use GPU computing platforms for wider applications, in particular, biomedical engineering. Similarly, bio-inspired algorithms such as the genetic algorithm and ANT Colony optimization [12] [13] [14] have been effectively implemented on GPUs which drastically reduce the communication overhead between the CPU and GPU. Problems like the analysis of biological DNA sequences can be effective in teaching CUDA [15] [16] [17] as well as a good source for discussion of the concerns involved in GPU programming and parallel programming in general. They can also be used to motivate a framework for easily parallelizing genetic algorithms. CUDA makes it simple for programmers with only a basic understanding of genetic algorithms to code their own genetic algorithms to run on NVIDIA GPUs.

2.3 Distributed Computing

Distributed computing allows for the utilization of vast amounts of computational power to tackle challenges in medical and computational biology. In particular, the biggest challenge in computational biology is simulating proteins due to their great complexity. Analysis of the folding of complex proteins requires the fastest CPUs and supercomputers. Recently, idle computing resources worldwide have been pooled to carry out this analysis, creating a massively distributed computing setup [4]. The key challenge in this distributed computing setup is to exploit parallelism, where it is often difficult to subdivide jobs and extract work from all jobs. Similarly, another challenge is to have efficient algorithms to exploit the available computing power. We believe that the protein-folding problem can help introduce concepts such as large-scale distributed computing architectures and algorithms, as well as network security, novel simulation methods and client-server architectures.

3. COURSE DESCRIPTION

In this section, we outline the proposed course curriculum, "High Performance Computing through Computational Biology"(HPCCB), where students will be introduced to various concepts in high-performance computing in a hands-on manner using examples from computational biology. This course will allow students to learn high-performance computing through direct application as well as carry out design projects from concept to realization. First, we provide an outline of the learning objectives, teaching methodology and a brief description of the biological applications chosen, with particular emphasis on their value in illustrating specific HPC concepts. Second, we provide criteria for course evaluation, including laboratory work and research term projects. Finally, we outline the process of evaluating the success of the course via direct feedback from students.

3.1 Prerequisites

This course is mainly designed for senior graduate students at the PhD level. Postgraduate students pursuing Master degrees may also register for this course if they have an individual study plan where this particular course is relevant. This course will require basic Linux competence and advanced programming skills. Basic

knowledge and/or exposure to biological, high data-intensive and computationally intensive applications will be useful.

3.2 Course Development and Learning Outcomes

Fig. 1 shows the design and development cycle of the HPCCB course. This course can be taught at the senior graduate (PhD) level as well as at the Masters level, with slightly different approaches. We begin by analyzing the preparedness of the

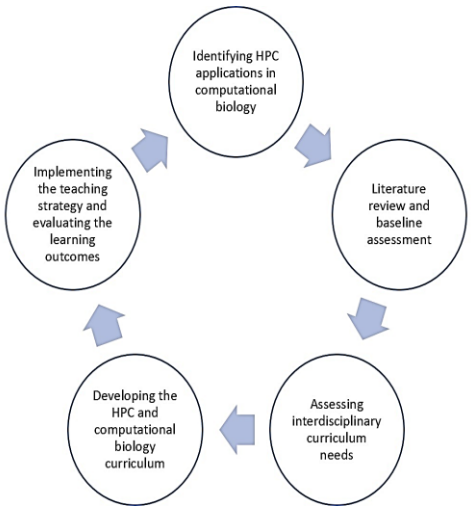

Fig. 1 HPC through computational biology: course design cycle

class, in terms of prior course work in CS as well as exposure to interdisciplinary domains. For graduate classes with a strong fundamental background in CS, the course can be tailored to be predominantly application-inspired (focusing on interdisciplinary research problems) to provide a different flavor to traditional concepts. For classes that are composed primarily of students at the Masters level, bridging material needs to be provided to supplement the application-based teaching of concepts with traditional CS problems and examples. Based on the above assessment, we define the learning outcomes of the course. While specific learning outcomes may vary, a general baseline can be as follows:

Students should be able to:

- Define different kinds of parallel architectures, like processor arrays, shared and distributed memory multiprocessors, reconfigurable computing processors and supercomputer architecture,
- Analyze an application for parallelization potential,
- Design/select algorithms for the high-performance computing requirements of the application,
- Assess the scale of bio-specific tools and libraries for parallel and distributed computing,
- Implement commonly used HPC platforms and parallel programming models using appropriate programming languages,
- Measure, assess and analyze the performance of the designed HPC solution and optimize HPC codes, and,
- Perceive the larger role of HPC in computational biology, through examples and detailed term research projects.

Other learning outcomes for application-oriented classes may include:

Students should be able to:

- Effectively utilize the visualization techniques to present the results,
- Provide experience in technical communication (both oral and written),
- Design prototype for computational biology software or bioengineering devices,
- Evaluate the economic considerations related to HPC-based bioengineering designs, such as market analysis and budgeting,
- Apply the regulatory rules important in biomedical devices such as FDA regulations etc., and,
- Value the ethical considerations of biological research, devices, and treatments on individuals, industries and society, as well as ethical considerations involved in collecting and processing big data.

At this stage, interdisciplinary programs may choose to provide an increased weightage to biological applications, while traditional CS programs may want to balance out the application examples with core CS examples. As a baseline, we recommend that at least 60% of the examples chosen in the course be based on biological applications, to exploit the full pedagogical value of application-based teaching. We also recommend that interdisciplinary HPC and CS instructors are trained in the pedagogical perspectives as shown in Fig. 2. In particular, instructors must be exposed to laboratory-based hands-on teaching techniques, which they must employ to guide students towards developing parallel thinking by working on specific application studies in the laboratory. Further, the instructors must obtain exposure to the state-of-the-art computational techniques used in biological applications prior to teaching the course. Once HPC is introduced to the class through the suggested computational biology applications, research areas that can be extended into term projects are selected. In the final stage, the proposed curriculum is implemented with regular feedback to evaluate learning outcomes to further fine-tune the course.

3.3 Infrastructure

An important aspect of course development involves assessing the computational infrastructure necessary to conduct the course laboratory and research term projects. With the widespread availability of desktop and laptop GPUs, the infrastructure requirements for this course are minimal. The following are the minimum infrastructure requirements for the course.

- LAM/MPI, OpenMP, OpenCL, and CUDA
- 16 dual 450Mhz Pentium II Linux PCs

We recommend that computers/clusters with higher specifications be made available when possible.

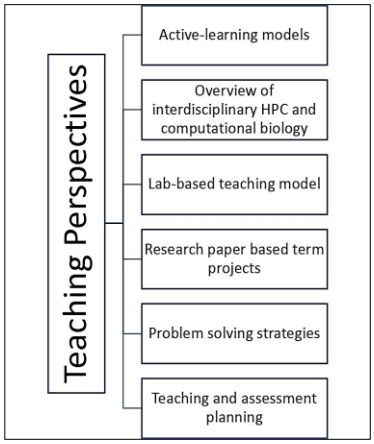

Fig. 2 Teaching perspectives for HPC and Computational Biology course

4. SAMPLE COURSE OUTLINE AND DESCRIPTION

We now provide a sample course outline that details the HPC concepts that can be covered in a one-semester course, along with the suggested bioengineering applications to introduce them.

4.1 Parallel Computation in Biological Applications

This section will comprise of a literature review on the basics of high-performance and PDC applications. Here we will introduce the real-world applications of HPC, with specific focus on computational biology and bioengineering, with surveys and extracts from texts like [18]. We will also introduce the basic concepts and terminology in HPC and parallel processing.

Section Summary

- High-performance computing and biological networks
- HPC architecture and Parallel processing

4.2 Parallel Implementation

In this section, we will introduce the problem of analyzing DNA sequences collected from large genome projects. One problem in the analysis of biological DNA sequences is the alignment of long sequences to identify regions that are matched and mismatched [19] [20]. This is accomplished by implementing a dynamic programming problem that provides a pair-wise comparison of sequences. The major problem when processing a long sequence of the whole genome is to meet the requirements of computer memory and execution time i.e. memory and CPU intensive application. Therefore, parallel implementations of this dynamic programming algorithm are necessary, as described in [19]. Through this problem, we introduce the concepts of parallel architectures and programming, along with data parallelization like SIMD and MIMD.

Section Summary

- Biological sequence analysis algorithm
- Parallel sequencing analysis methods: SIMD, MIMD
- Parallel biological tools
- References: [19], [20]

4.3 Massively/Embarrassingly Parallel Solutions to Computational Biology Problems

In this section, we use the problem of molecular sequence analysis to introduce the concept of hybrid SIMD-MIMD architectures. We attempt to provide insight into the process by which a programmer must determine the extent and scale of parallelization required for an application. We also introduce several mapping techniques to reduce the complexity of the sequence alignment. Through this example, we also briefly introduce the concept of performance evaluation.

Section Summary

- Hybrid SIMD-MIMD architecture
- Levels of parallelization fine and coarse-grained, and implementation choices
- Mapping techniques,
- Reference paper: [21]

4.4 Multithreaded/Multi-core Parallel Implementation

In this section, we return to [19] motivate the idea of multithreaded parallel programming implementations. We further use the problem in [22], where a protein threading problem is formulated as a Mixed Integer Program (MIP) to describe the advantages of multithreading. In this example, students will learn multithreading by decomposing the MIP into sub-tasks and implementing them in a multi-core parallelized setting. Further, this MIP can also be viewed as the shortest path problem. Through this problem, students will gain an in-depth insight into parallel multithreading/multi-core parallel implementations as well as a flavor for optimization problems such as linear programming (LP), MIP and shortest path problems.

Section Summary

- A multithreaded parallel implementation, parallel execution model
- Solving protein threading problem in parallel [22]
- Reference paper: [19], [22]

4.5 Performance Improvement using Distributed Computing Environments

In this segment, we introduce performance optimization for parallel codes by considering the example of the Clustal W algorithm for multiple sequence alignment. This algorithm involves a pairwise comparison stage followed by the construction of a guided tree, which is then used for progressive alignment. Each stage of this algorithm needs to be optimized to reduce computational complexity. Through this example, students will learn to measure the performance of their parallel implementations and optimize the code to improve performance. Further, parallel clustering algorithms will also be introduced. This problem will also use OpenMP, OpenCL and CUDA programming, thereby providing a succinct overview of these techniques.

Section Summary

– Measure and assess the performance of parallel implementations
– Parallel code optimization, parallel clustering
– OpenMP, OpenCL, and CUDA programming techniques
– Reference paper: [23]

4.6 Big Data and Parallel Computing using Genomics and Computational Biology

In this section, a large-scale search problem involving big-data will be introduced with the help of genomics data sets. Various search algorithms like GeneWise and GeneMatcher [24] will be introduced, with emphasis on performance analysis for parallel big-data implementations.

Section Summary

– Genomics introduction
– Various genome algorithms
– Performance analysis
– Reference paper: [24]

4.7 High-performance algorithms in Computational Biology

The examples suggested in the above segments will be revisited to understand SMP and CMP deployments and introduce advanced concepts in algorithm complexity analysis.

Section Summary

– SMP and CMP machine deployment
– Complexity algorithm analysis
–

4.8 Parallel and Distributed Memory Architecture

In this section, we use the problem of gene linkage analysis to introduce the concept of distributed memory architectures. Linkage analysis software like Genehunter [25] efficiently distribute both computation and memory. Students will learn these parallelization approaches, including assessment of memory requirements and memory architectures through this problem.

Section Summary

– Memory architecture for parallel bioengineering
– Memory requirements
– Parallelization approach and methods
– Reference paper: [25]

5. RESEARCH PROJECT

This advanced elective course focuses on interdisciplinary teaching approach similar to other existing courses. Our unique approach is to satisfy the current computing demand and train our students in the research and scientific-orientation. In this context, we provide the final project based on the research problems with the perspectives of computational biology. The references provided in each of the above sections can be extended into excellent implementation projects in HPC as well as computational biology. The design project component is intentionally kept open-ended, with no predefined solution. Project outcomes can be hardware, software or review-based.

For example, Fig. 3 represents the basic flow of genetic information and how it gets manifested at the population level in bacteria. At each stage we need high-throughput computational programs to simulate various processes involved.

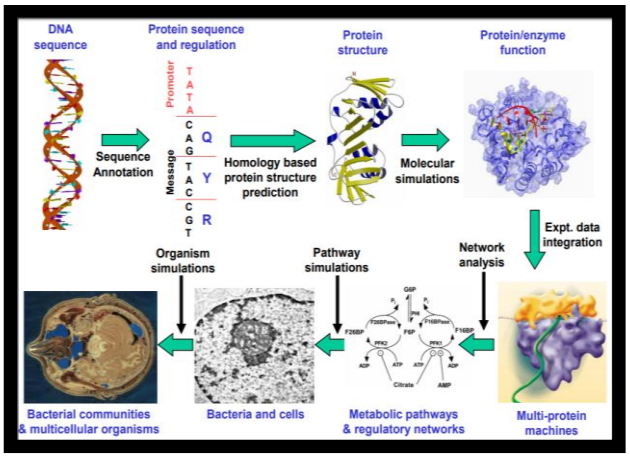

Fig. 3 Basic flow of genetic information in bacteria [34]

One of the main challenges in implementing biological high-performance computing is to develop platforms for efficient analysis by choosing the right architectures and implementations. The implementation project will provide students with hands-on experience in applying the learning outcomes of this course to real-world computational biology problems. Students may, in consultation with the course instructor, choose research projects related to various topics in course outline or topics that are of interest to their doctoral research, in the case of students at the PhD level. Depending on the size of the class, research projects may be carried out in teams or individually. A research design project typically includes collecting data/information, using tools to analyze the data, using various methods and algorithms, and quantifying the effectiveness of the proposed solution. Potential research topics will be listed during first two weeks of a term, considering the preparedness and specific research interests of the graduate students taking the course. This research project will help students in the following activities:

– Apply theoretical knowledge to identify, formulate and solve real-world biological problems,
– Design devices, software, or experimental apparatus related to biomedical applications or research,
– Experience the end-to-end design process in real-world applications,
– Test solutions by designing suitable experiments,
– Plan and manage a research project, including building teamwork strategies,
– Gain experience in working independently or as part of interdisciplinary teams, and
– Effectively communicate and document research progress and outcomes.

6. ASSESSMENT METHODOLOGY

Students are evaluated on their progress towards the course learning outcomes based on the following criteria:

– Demonstration of a computational biology software, device-prototype or study as part of a term research project,
– Submission of assignments and laboratory reports,
– Submission of a final project report that includes theoretical modelling and related observations,
– Short oral presentation of the outcomes of the term project, and
– Final examination.

In Table 1, we summarize the weightage assigned to different facets of the course in the evaluation process.

Table 1: Grading Breakdown

Item	**Weightage**
Assignments	10%
Research Project	40%
Laboratory Modules	20%
Final Exam	30%

Table 2: Laboratories and Assignments

Topic	**No. of Hours**	**Assignments/Labs**
Parallel architecture and system models	2	Reference [27]
Introduction to HPC in Biological applications	5	Paper review
Teaching parallel programmingtechniques MPI, Pthreads and OpenMP, Biotools/libraries	6	Lab 1-Lab 4
Analyze data intensive and computationally intensive applications	3	Lab 5
Data visualization	3	Lab 6
Big Data Infrastructure	4	Hadoop Eco sy stem, HDFS
HPC in Big Data	4	Lab 7
Statistical modeling for data	2	Linear, logical regression, Bayesian models
Research project	40	Chosen by student in consultation with instructor

In Table 2, we list various laboratory modules along with the number of hours assigned to them. Note that students are expected to spend at least 40 hours over the semester working on their chosen research project. We also provide the rubrics for assessment of the course research project in Appendix A.

7. REFERENCE MATERIAL

To accomplish the learning outcomes in Section 3.2, we choose a set of examples, drawn primarily from current literature in computational biology, and supplemented from traditional textbooks like [26], to cover the essentials of parallel and distributed high-performance computing. We note that specific references pertaining to the chosen biological applications and their HPC implementations have been provided under each subsection of the course outline in Section 4. This material can be replaced or augmented with current literature relevant to the research interests of the students taking the course. Considering the practical and demonstrative nature of the course, the use of online interactive books is worth mentioning. While we do not recommend any specific online textbooks, some interactive programming books can be used as a good source of practice material [27]. With several iterations of this course, we will work towards the design of an interactive online book where we will introduce parallel and distributed computing through examples and exercises from biological applications. Students will then be able to access this material both on-campus as well as remotely and will be able to work on the exercises and projects designed for them online.

8. FEEDBACK MODEL

The level of student engagement and satisfaction with the learning outcomes of the course can be analyzed by conducting two surveys during the term, and one survey after the end of the term. The first survey can be conducted in one month of the course and the second survey in three months of the course. The first two surveys may be replaced by a single survey for universities following the quarter system with shorter terms. Students can be asked to rate the course on a scale of 0 to10 on a variety of topics, including but not limited to, their satisfaction with the course instructor, laboratories, assignments, choice of research projects, amount of time invested into the course, difficulty level of the course and usefulness of the assigned textbooks and reference material. In addition, students may also be asked to provide suggestions on how the course can be improved both in the short term that is, during the same semester, and in the long term, that is, in subsequent iterations. This feedback will then be used as a pivot to improve the course structure to cater the students' needs in the particular term, as well as to refine the course development process as laid out in Fig. 1 in the subsequent terms.

9. COURSE EXPERIENCES AND EVALUATION

We now detail the experiences and evaluation results of the first author in introducing a similar curriculum, and describe how these experiences have contributed to the shaping of the curriculum presented in this paper. The first author introduced an interdisciplinary course on HPC and PDC when she was a faculty

and member of the Board of Studies (BoS)/curriculum design committee at Amrita/VIT, India in 2005-2009. In this course, senior undergraduate engineering students in their final year were introduced to HPC and PDC concepts through real-world examples drawn from multiple domains, including computational biology. As the course involves learning HPC and PDC concepts directly via implementation of problems at the research-level, the feedback received indicated that this course was too involved at the undergraduate level. Therefore, the author reintroduced this curriculum in 2010 as an advanced elective for graduate students (PhD) who have taken prior courses on the foundations of HPC, and could draw value from the interdisciplinary flavor of this course. This course, comprising of 9 students, was well-received with an average rating of 95%. Several students attending the course also chose to pursue interdisciplinary HPC-based term and thesis projects supervised by the first author. Initially, the author faced various difficulties such as getting proper training in HPC, lack of proper textbooks and lack of HPC tools to support the computational needs at the university. However, now the author's institute has received the funding from Intel to support the infrastructure needs and the management streamlined the other issues as well. We also taught the project based proposed course, which is mentioned in Section.5, was well received by Ph.D. students with an average rating of 90%.

10. CONCLUSION

In this paper, we present a novel course designed to teach high performance parallel and distributed computing to graduate students directly via hands-on applications drawn from computational biology. While HPC successfully contributed to solving several biological problems, we believe that computer scientists can conversely draw enormous pedagogical value from biological examples and problems. Motivated by this, we presented a course curriculum where HPC is introduced almost entirely via hands-on application examples. We first summarize the main trends in HPC applied to biomedical engineering and computational biology. We demonstrate several successful stories and application fields, in which relevant biological problems have been solved (or are being targeted) using the computational power available in current processors. We then provide a detailed description of the course and suggested examples to be used in the course. We would also like to point out some potential implementation challenges in terms of the high learning curve in emerging programming models. We believe that this course can inspire students to undertake investigations into improving the performance on HPC systems motivated by computational biology problems. This will be of high technical interest in the computer science community as biological applications have novel computational patterns that can lead the next generation of high-performance heterogeneous computing systems.

REFERENCES

1. Sanbonmatsu, K. Y., and C-S. Tung. High performance computing in biology: multimillionatom simulations of nanoscale systems. Journal of structural biology 157.3 (2007): 470-480.
2. Stevens, R. (2002, September). Biology and High-Performance Computing. In UK-HPCUsers Meeting. http://www.cels.anl.gov/about/people/files/UK-BIO-HPC-final1.pdf
3. Pronk, Sander, et al. GROMACS 4.5: a high-throughput and highly parallel open sourcemolecular simulation toolkit. Bioinformatics (2013).
4. Pande, Vijay S., et al. (2003). Atomistic protein folding simulations on the submillisecondtime scale using worldwide distributed computing. Biopolymers 68.1 91-109.
5. Lee, W.P., Tzou, W.S. (2009). Computational methods for discovering gene networks from expression data. Briefings in Bioinformatics 10 (4):408-423.
6. http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-047computational-biology-fall-2015/.
7. https://biology.ufl.edu/files/ZOO4926-6927-CompBio-Basic-Res-Comp-Skills.pdf.
8. Fundamentals of High-Performance Computing for Public Health. Columbia University School of Public Health. https://www.mailman.columbia.edu/public-healthnow/news/big-data-academy-public-health-supercomputing.
9. CMSC 838T: Advanced Topics in Programming Languages - Systems Software for High-performance Computing, Emphasis on Bioinformatics Applications. University of Maryland. http://www.cs.umd.edu/class/spring2003/cmsc838t/.
10. Marx, Vivien. Biology: The big challenges of big data. Nature 498.7453 (2013): 255-260.
11. Garland, M., Kirk, D.B.: Understanding throughput-oriented Architectures. Communications of the ACM 53, 5866 (2010)
12. Wong, M. L., Wong, T. T., and Fok, K. L. (2005, September). Parallel evolutionary algorithms on graphics processing unit. In 2005 IEEE Congress on Evolutionary Computation (Vol. 3, pp. 2286-2293). IEEE.
13. Manfrin, M., Birattari, M., Stutzle, T., Dorigo, M.: Parallel ant colony optimization forthe traveling salesman problem. In: Dorigo, M., Gambardella, L.M., Birattari, M., Martinoli, A., Poli, R., Stutzle, T. (eds.) ANTS 2006. LNCS, vol. 4150, pp. 224234. Springer, Heidelberg (2006)
14. Cecilia, J.M., Garcia, J.M., Ujaldon, M., Nisbet, A., Amos, M.: Parallelization strategiesfor ant colony optimization on GPUs. In: NIDISC 2011: 14th International Workshop on Nature Inspired Distributed Computing. Proc. 25th International Parallel and Distributed Processing Symposium (IPDPS 2011), Anchorage, Alaska, USA (May 2011)
15. Wong, M. L., and Wong, T. T. (2009). Implementation of parallel genetic algorithms ongraphics processing units. In Intelligent and Evolutionary Systems (pp. 197-216). Springer Berlin Heidelberg.
16. Pospichal, P., Jaros, J., and Schwarz, J. (2010, April). Parallel genetic algorithm on thecuda architecture. In European Conference on the Applications of Evolutionary Computation (pp. 442-451). Springer Berlin Heidelberg.
17. Garland, M., Le Grand, S., Nickolls, J., Anderson, J., Hardwick, J., Morton, S., Phillips, E., Zhang, Y., Volkov, V.: Parallel Computing Experiences with CUDA. IEEE Micro 28, 1327 (2008)
18. Aluru, Srinivas, ed. Handbook of computational molecular biology. CRC Press, 2005.
19. Martins, Wellington S., et al. "Whole genome alignment using a multithreaded parallel implementation." Symposium on Computer Architecture and High Performance Computing. 2001.
20. Yap, Tieng K., Ophir Frieder, and Robert L. Martino. "Parallel computation in biologicalsequence analysis." IEEE

Transactions on Parallel and Distributed Systems 9.3 (1998): 283-294.
21. Schmidt, Bertil, Heiko Schrder, and Manfred Schimmler. ”Massively Parallel Solutions forMolecular Sequence Analysis.” ipdps. 2002.
22. Yanev, Nicola, and Rumen Andonov. ”Solving the protein threading problem in parallel.”Parallel and Distributed Processing Symposium, 2003. Proceedings. International. IEEE, 2003.
23. Mikhailov, Dmitri, Haruna Cofer, and Roberto Gomperts. ”Performance optimization ofclustal w: Parallel clustal w, ht clustal, and multiclustal.” SGI ChemBio (2001).
24. Mo, Yi, Moira Regelson, and Mike Sievers. ”A Study of GeneWise with the DrosophilaAdh Region.” 13th Annual Genome Sequencing and Analysis Conference. 2001.
25. Conant, Gavin C., et al. ”Parallel genehunter: Implementation of a linkage analysis packagefor distributed-memory architectures.” Journal of Parallel and Distributed Computing 63.7 (2003): 674-682.
26. Jeffers, Jim, and James Reinders. High Performance Parallelism Pearls Volume Two: Multicore and Many-core Programming Approaches. Morgan Kaufmann, 2015.
27. K. Hwang, G. C. Fox, and J. J. Dongarra, Distributed and Cloud Computing: From ParallelProcessing to the Internet of Things, Elsevier, 2012.
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. New York, NY, USA: ACM, 2013.
29. S. K. Prasad, A. Chtchelkanova, S. Das, F. Dehne, M. Gouda, A. Gupta, J. Jaja, K. Kant, A. La Salle, R. LeBlanc, M. 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, in Proceedings of the 42Nd ACM Technical Symposium on Computer Science Education, ser. SIGCSE 11. New York, NY, USA: ACM, 2011, pp. 617618.
30. R. Brown, E. Shoop, J. Adams, C. Clifton, M. Gardner, M. Haupt, and P. Hinsbeeck, Strategies for preparing computer science students for the multicore world, in Proceedings of the 2010 ITiCSE Working Group Reports, ser. ITiCSE-WGR 10. New York, NY, USA: ACM, 2010, pp. 97115.
31. A. Fitz Gibbon, D. A. Joiner, H. Neeman, C. Peck, and S. Thompson, Teaching high performance computing to undergraduate faculty and undergraduate students, in Proceedings of the 2010 TeraGrid Conference, ser. TG 10. New York, NY, USA: ACM, 2010, pp. 7:17:7.
32. J. Adams, R. Brown, and E. Shoop, Patterns and exemplars: Compelling strategies for teaching parallel and distributed computing to CS undergraduates, in IEEE International Parallel and Distributed Processing Symposium Workshops PhD Forum (IPDPSW), May 2013, pp. 12441251.
33. Eduardo Csar, Ana Corts, Antonio Espinosa, Toms Margalef, Juan Carlos Moure, AnnaSikora, Remo Suppi, “Introducing computational thinking, parallel programming and performance engineering in interdisciplinary studies,” J. Parallel Distrib. Comput. 105, 2017, pp. 116-126.
34. Rick Stevens, Biology and High-Performance computing https://pdfs.semanticscholar.org/presentation/cd70/536ba6011018539eda74dddaede95cca893b.pdf

Appendix A: Rubric for Research Project Evaluation

Criteria	Poor	Average	Excellent
Abstract	Not precise or succinct, one or more elements missing	Contains the topic under investigation, but does not clearly summarize the conclusion and/or methodology of investigation.	Clearly states the topic of investigation, how the investigation was undertaken and the conclusions.
Introduction and relevance of application, device, prototype or study	Little or no description, little or no attempt to explain the significance of the application, device, prototype or study	Some description of device, some attempt to explain the significance of the application, device, prototype or study	application, device, prototype or study is clearly described, clearly explains the significance and application of developed application, device, prototype or study, places the problem in the context of relevant literature
Blue prints and Block Diagrams	Unreasonably sized and spaced, either incorrectly captioned or not captioned at all	Most are appropriately sized and spaced, most are properly captioned	All block diagrams have a specific purpose, all appropriately sized and spaced, all properly captioned
Procedure	Method not described clearly, omits crucial details	Method described fairly clearly, some important details omitted	Provides a detailed and comprehensive description of the implementation
Demonstration	Unclear demo n-stration/questions not answered	Better description/some questions answered	Excellent and clear demonstration/ most questions answered