Python: A Promising Common Platform for Parallel and Distributed Computing Education

Linh B. Ngo
School of Computing
Clemson University
Clemson, South Carolina
lngo@clemson.edu

Edward B. Duffy
Clemson Computing and Information Technology
Clemson University
Clemson, South Carolina
eduffy@clemson.edu

Amy W. Apon
School of Computing
Clemson University
Clemson, South Carolina
aapon@clemson.edu

***Abstract*—**

***Keywords*-MPI; MapReduce; Spark; CUDA; Python; distributed systems; parallel computing education**

I. Introduction

Parallel and distributed computing (PDC) terms such as multi-core processing, GPU processing, big data, data-intensive computting, etc have become the norms in aspects of IT across academic and industrial areas. It is critical that college graduates are properly introduced to PDC core concepts and technologiesfor their future careers. It is typical for PDC to be primarily taught as a single course within the entire undergraduate CS curriculum while having some PDC concepts embedded in other courses [1], [2].

The existing body of PDC knowledge spans across four different areas: Data Structures and Algorithm, Software Design, Software Environments, and Hardware [3]. This leads to an increase in the number of computing tools and platforms that need to be used to teach the corresponding PDC concepts. The relevancy of including various technologies in the curriculum is clear as PDC begins to move beyond the traditional high performance computing concepts and into the areas of data-intensive computing, big data analytics, and large-scale streaming systems. The students are faced with a significant challenge to balance between learning both new PDC concepts and the accompanying computing platforms.

In this paper, we describe the development of an educational approach to teach various PDC concepts using a common computing platform based on Python. Our approach and the accompanying teaching platform will offer the following advantages:

- The approach facilitates the delivery of most common PDC areas such as high performance computing, data-intensive computing, in-memory distributed computing, and GPU-based computing. These areas are originally designed for different computing platforms using different languages. The corresponding libraries in Python that can be leveraged to teach these areas are shown in Table I.
- Python is a user-friendly introductory programming language with a low barrier of entry to new users. It is also well supported and works across different operating systems (Linux, MacOS, and Windows).
- Python libraries to support PDC are highly unified and well-defined by the open source community. In many cases, the performance of these libraries are not as good as the original. On the other hand, it is more important for students to spend more time understanding PDC concepts and less time on the technical and syntax aspects of specific tools and languages, even if they provide the optimal performance.

The remainder of this paper is organized as follows. Section II describes in detail the development of our approach and the accompanying computing platform, and how it will be used to facilitate the teaching of the four PDC areas mentioned in Table I. Section III describes the preliminary responses from students as we utilize the platform to teach a one-week REU-site boot-camp workshop for data-intensive computing. Section IV concludes the paper and discusses future work.

II. A Common Python Platform for Parallel and Distributed Computing Education

A. Computing Environment

Jupyter Hub

Table I: PDC Areas, their traditional platform/language for educational purpose, and the new corresponding Python-platform/libraries

Area	Typical platform/language	Python-based equivalent
High Performance Computing	MPI/C	py4MPI
Data-Intensive Computing and Big Data	Hadoop MapReduce/Java	Python
In-memory Distributed Computing	Spark/Scala	Spark/pyspark
GPU-based Computing	...	...

B. Python for MPI

C. Python for Hadoop MapReduce

D. Python for Apache Spark

E. Python for CUDA

III. Preliminary Results

IV. Conclusion and Future Work

V. Acknowledgments

References

[1] G. Wolffe and C. Trefftz, "Teaching Parallel Computing: New Possibilities," *Journal of Computing Sciences in Colleges*, vol. 25, no. 1, pp. 21–28, 2009.

[2] R. M. Butler, R. E. Eggen, and S. R. Wallace, "Introducing Parallel Processing at the Undergraduate Level," in *ACM SIGCSE Bulletin*, vol. 20, no. 1. ACM, 1988, pp. 63–67.

[3] 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*, 2010.

[4] S. K. Prasad *et al.*, "NSF/IEEE-TCPP Curriculum initiative on parallel and distributed computing - Core topics for undergraduates,"
http://www.cs.gsu.edu/ tcpp/curriculum/sites/default/files/NSF-TCPP-curriculum-version1.pdf, 2012.

[5] Apache Hadoop 2.0: YARN (Yet Another Resource Negotiator), hadoop.apache.org/docs/current2/hadoop-yarn, 2013.

[6] Apache Spark, "Lightning-fast cluster computing," http://spark.incubator.apache.org, 2013.

[7] Apache Tez, "A framework for near real-time big data processing," http://hortonworks.com/hadoop/tez, 2013.

[8] Apache HBase, http://hbase.apache.org, 2013.

[9] Clemson University, "CPSC3620: Parallel and Cluster Computing,"
https://sites.google.com/site/clemsondagoba/teaching, 2014.