EAPoster: Using BigData for learning about a slice of parallel computation in several courses

***Abstract*—An Early Adopter effort for the Parallel and Distributed Computation (PDC) curriculum emphasizing "big data" applications is described. The work in progress will use modules in several courses across the curriculum, starting with the first term of the first year. Students are asked to learn about and work with parallelism in distributed operation, using WebMapReduce, Hadoop, and Spark. Surveys are being developed to see changes in beliefs about PDC and in conceptual understanding. Content product will be made available through community repositories for PDC curriculum work.**

***Index Terms*—computer science education, parallel and distributed computation education, data science education**

I. Overview

Synopsis : Use "Big Data" parallelism in several modules in existing courses across the curriculum: lab work culminating in problem-solving with WebMapReduce in CCI 101 early in the first year (before a full course in programming), problem solving with a Hadoop in CS 283, a second year course on Systems Programming. More customized, efficient processing with Spark at the advanced undergraduate level in CS 385, a course on Evolutionary Computing.

Background: Drexel Computer Science has approximately 150 undergraduate majors per year (700-800 through the five year undergraduate program of cooperative education)t. There is an existing course on PDC on concurrent programming CS 361 an elective at the advanced undergraduate level which is using Java and Go covering standard synchronization concepts (threaded programming with shared variables, race conditions, mutual exclusion, semaphores and monitors, synchronized message passing, etc.).

Course design and development tasks : Introduce a slice of PDC to the mainstream through episodic, spiraling work several courses: 1. Introduction to WebMapReduce computation and concepts in CCI 101, a first year first term Introduction to Design for Informatics and Computer Science, taken by all majors, not just CS, in the College of Computing). 2. Lab activities in CS 283, a required course in Systems Programming for second year students: measuring parallel performance in WebMapReduce, and porting computation work from WebMapReduce to Hadoop. This is a shift from current CS 283 PDC course activities which explore multithreaded parallelism via PThreads. 3. Customized computation using Spark and Mahout machine learning libraries in CS 385 (Evolutionary Computing), an advanced undergraduate elective, new material for that course.

Rationale for doing only a "slice": In the crowded CS curriculum, mandating new topics in the mainstream of courses must be done with attention to both the intellectual and departmental-political issues involved altering topics. We believed that an incremental approach would work better at persuading our deparmental colleagues and our student clientele at further PDC into the mainstream. We decided that it would be more appropriate to first demonstrate PDC topic integration in courses we had direct control over. We defer an implementation of a integrated full PDC curriculum as recommended by ACM 2013 or the IEEE/TCPP model curricula, after we see how our design choices have worked out in the initial work.

II. Educational objectives

New content: Some of the programming patterns in PDC emphasize functional programming, which formerly was seen first as a concept in our third-year programming concepts course CS 360. Initially we see that the students will need work on the fundamentals – understanding how sample parallel computation works and getting them to correctly synthesize their own. However, a fundamental difference for PDC as contrasted with the other content of our curriculum is that much of it deals with concepts and programming patterns involved in practical computational efficiency rather than conceptual notions of efficiency or correctness. Since parallelism is most relevant where sequentialism does not deliver, it also tends to involve a much larger computational burden than most students typically face in introductory programming, where the complexity and cost of writing a correct program are typically the main concern. So the mission of parallelism – better performance – needs to be prominent at the introductory work, even if proficiency in efficiency is not an objective until second or later courses.

Relevance: Many of our students decide to go into careers in applications of computing to business. Familiarizing them with the opportunities and techniques of Data Science is consistent with their long-term career intentions. We believe that early introduction and frequent contact is the way to establish relevance for our students.

Practical effectiveness: A major distinctive feature of Drexel is its extensive co-op program. This can attract students who value coursework with hands-on experience in practical problem-solving, troubleshooting, and professionally appropriate tools. However, there can be tension between the "get down to it" mindset and situations where abstraction and extended conceptual reasoning are helpful. We are attempting to design our curriculum to showcase or preview the practical while spending enough time developing students' ability and

willingness to use the abstraction and high-level thinking where it would be meaningful.

Impact: We seek to have a long-term permanent effect on the CS education of our students. Our approach is designed for repeated hands-on encounters at rising levels of expectation and sophistication, to better provide retention and transference. For example, it is planned that the cs283 activities refer to earlier experience with MapReduce in CCI 101, but to write stand-alone programs invoking Hadoop using Python, or Java rather than in Javascript with a web interface to the computation. The conceptual framework about PDC that the students were introduced to in CCI 101 will be further elaborated, and detailed, with greater student responsibility expected to measure and improve performance through PDC. Practical usage of parallel algorithms is needed in cs385, where students often find that a program or program step takes hours to run, rather than seconds and must rethink their overall problem solving strategy as a result.

III. Plans for implementation and evaluation

In Fall 2015 a module with WebMapReduce in CCI 101 saw pilot usage. In the next two years, we will roll out modules for the three courses. We describe briefly the pilot version of the module developed for CCI 101. The hands-on work for the one-week module is designed to be performed in a lab classroom with small groups of students in a two hour session. Instructional staffs' during the lab is to circulate around the room to facilitate progress. Pre-lab activities ask the students to read a JavaScript WebMapReduce tutorial and to watch a video on it. Students have written programs in JavaScript in earlier course activity.

The lab materials direct the students through four activities: 1. Students apply "human-powered parallelism" to classifying playing cards by suit, and to explain why dividing up the task might get the work done faster than with one person doing all the work. 2. Each student group is given a WebMapReduce program and observes what happens using the system's execution visualization. They are asked to discuss: "what does this program do and how does it do it?" Students construct their own explanation of what the map and reduce operations for the example (which counts words from text) do, rather than recalling the explanation presented in lecture or written materials. 3. Groups are asked to modify the program from (2) to compute student grade averages from lists of data. Students are shown Python coding of a similar problem and asked to transfer the JavaScript programming for (2) into Python coding. The switch of language and problem emphasizes the value of the abstract knowledge of the programming pattern over the language- or problem-specific programming. 4. Each group is asked to create/develop on their own the solution to a another problem, computing enrollment and drop out figures from Philadelphia School District data. Work is expected to be turned in online at the end of the lab for grading. Incomplete submissions can be finished and turned in later in the weekly lab cycle.

We have developed pre- and post-surveys to evaluate the impact and reception of the modules on student learning and beliefs about PDC. Figure 1 shows a few of the questions in the draft survey which aim to do this. After studying prior pdc assessments (e.g. [1]) we have developed a few specific technical questions about the effects of parallelism and properties and application of map-reduce. Figure 2 shows a question from a draft of this section of the survey.

IV. Dissemination

We plan to make the content of our curriculum modules and our survey questionnaires available through the CSInParallel repository and TCPP repositories when complete.

V. Acknowledgments

The authors would like to thank Dick Brown and his CSInParallel colleagues for help with WebMapReduce, and Sushil Prasad and the staff of TCPP for their assistance in becoming familiar with the materials available through TDCC. This work was supported in part by an Early Adopter grant from TCPP.

References

[1] Rague, B. Measuring CS1 perceptions of parallelism. In *Frontiers in Education Conference (FIE), 2011* (2011), IEEE, pp. S3E–1.

Appendix A: questions from student surveys

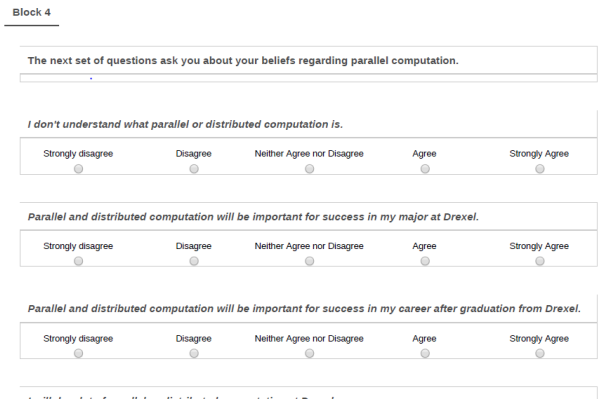

Fig. 1. Survey excerpt on beliefs about PDC (pre-survey). The five page survey takes about ten minutes to complete.

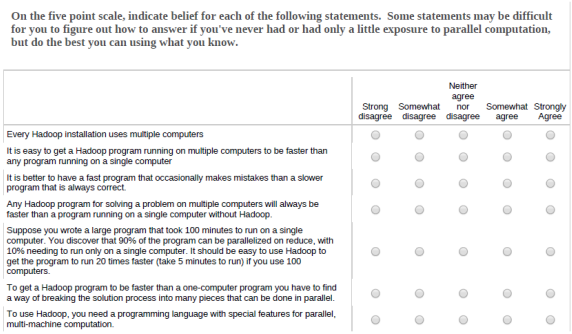

Fig. 2. Sample question assessing PDC and MapReduce knowledge and skill. The seven page survey takes about ten minutes to complete.