Abstract - The widespread deployments of multicore and GPU based systems in recent years have changed the computing landscape. However, most undergraduate computer science (CS) programs do not teach parallel and distributed computing concepts, and CS undergraduates are typically exclusively trained to think and program sequentially. To address the rapidly widening gap between emerging highly parallel computer architectures and the sequential programming approach taught in traditional CS courses, the Computer Science Department at Tennessee Technological University plans to systematically integrate PDC topics into the current undergraduate curriculum. We have begun integrating elements of the TCPP curriculum into multiple undergraduate CS classes over multiple semesters starting from fall 2014 with the support of an NSF-TCPP Early Adopter Award. This paper presents the implementation plan, current status and preliminary results.

I. OVERVIEW

Tennessee Technological University is a medium size public four-year engineering college with approximately twelve thousand students. The computer science department has approximately four hundred undergraduate majors and offers BS and MS degrees in computer science, and a PhD degree through the College of Engineering. The undergraduate curriculum contains a three course required programming sequence consisting of Introduction to Problem solving and Computer Programming, Data Structures and Algorithms, and Object Oriented Programming and Design. These introductory courses are offered every semester. To address the high DFW rates in the 1st and 2nd programming classes, a required principle of computing (CS0) class had been added from fall 2013 in the curriculum. Required upper division courses are typically offered once each year. Our required upper division courses for the traditional computer science degree include Assembly Language Programming, Operating Systems, Computer Networks, Computer Architecture, Database Systems, and a two-semester software engineering series.

II. IMPLEMENTATION PLAN

The objective of the present work is to integrate selected PDC topics in six undergraduate classes (CS0-Principles of Computing, CS1-Introduction to Problem Solving and Computer Programming, CS2-Data Structure and Algorithms, Operating Systems, Computer Networks, Computer Architecture) and add an upper division elective parallel programming class in the undergraduate program. The plan is to incorporate PDC topics starting with CS0 (Principles of Computing) and gradually incorporate PDC topics in subsequent classes in the following semesters and complete the implementation in two years. Our strategy is to repeat some of the PDC topics from the previous classes in greater depth and add new topics in subsequent classes in following semesters. We plan to develop self-contained pluggable modules (power point slides, assignments, quizzes, and exam questions) that can be easily adopted by faculty members for their classes.

III. CURRENT STATUS

In fall 2014 the upper division elective Parallel Programming class was added to the curriculum and was offered for the 1st time. The class was fairly popular and 30 students took the elective parallel programming class. The topics included different parallel programming models and architectures, design, development, and implementation of parallel algorithms using different parallel program tools. The bulk of the course was spent in writing parallel programs using pthread, OpenMP, MPI, and CUDA.
In fall 2014, PDC topics were integrated in the CS0 - Principles of Computing class. All incoming freshman students are required to take this class. No prior computing knowledge is expected for this class. There were 37 students in the fall 14 CS0 class. The following PDC concepts / terminology were introduced in Principles of Computing: parallel and distributed computing, concurrency, parallelism vs concurrency, speedup, and race conditions. These concepts were introduced through traditional lectures and slide handouts. Students were given two parallel programming examples using SNAP![1]. SNAP! is a popular graphical teaching tool used to teach beginning programmers. One example was parallel addition of integers and the other example was sorting a list in parallel. While demonstrating these examples, students were introduced to general methodologies of developing parallel programs. Figure 1 shows a graphic of the sort example. As homework assignment students were given two sorting programs: one parallel and one sequential. Students were asked to run and time the code with different size datasets and plot a speed-up graph and answer some questions. About 3 lecture hours out of total 42 hours were spent on PDC topics.

Assessment of students’ learning was done in two ways: subjective and objective. In the subjective assessment, the students’ perspectives on their learning were measured through pre and post surveys. Objective assessment was done through quiz and exam questions related to PDC topics and a homework assignment. In the pre and post surveys students were asked about their level of understanding of five PDC topics (Parallel Computing, Distributed Computing, Concurrency, Concurrency Vs Parallelism, and Race Condition) and answers were measured in the 1-5 point Likert scale (1-None at all, 2-Heard the terminology but don’t know what it is, 3-Understand somewhat, Understand, Very well). Figure 2 shows result of the pre and post surveys. From figure 2 it can be seen that students gained knowledge on all selected PDC topics. From the figure it can also be seen that students gain is least on the race condition topic, which was expected as that topic is difficult to comprehend for incoming freshman students. The modules for CS0 are available upon request (we will add it to CEDER repository and on our website soon).

In spring 2015 PDC topics are being integrated in CS1- Introduction to Problem solving and Computer Programming and the integration in CS0 is continuing. PDC topics will be integrated in Data Structures and Operating Systems in fall 2015 and in Computer Networks and Computer Architecture in spring 2016.

ACKNOWLEDGMENT

- NSF/IEEE-TCPP Curriculum Initiative
- Department of Computer Science, Tennessee Tech. University

REFERENCES