Abstract—This paper is focused on describing methodology for teaching Parallel and Distributed Computing topics to undergraduate students. The incorporation plan consists of three core courses of Data Structures, Operating Systems, and Algorithms. These courses have been selected with the consideration that they provide a diverse platform for teaching PDC topics related to programming, algorithms, and systems related knowledge. The incorporation plan has been implemented through programming-based skills, interactive teaching style, and student assessment and evaluations. The paper describes our methodology of the courses and explains mechanism for their implementation. We also provide detailed course feedback and lessons learned.

Keywords- Early Adapter Plan, Data Structures, Operating Systems, Algorithms.
of ‘C’. In addition, a 2 hour per week of laboratory was also included.

Evaluation
For assessment, students were given an assignment in which they would have to provide possibilities of parallelism in different sorting algorithms. This strengthened students’ intellectual and thinking capabilities.

Operating Systems
The course of OS involved many PDC topics. The rationale was to provide a broad understanding to students. The focus is mainly on programming topics related to synchronization using mutex and semaphores, shared memory programming using pthreads, distributed memory, inter process communication using pipes, client/server models, and process communication. In addition, the course covers topics related to consistency and scheduling. The course also has a few architecture related discussion on SIMD and MIMD. We also introduced the cross cutting topic of security, as we considered it important to introduced the topic of systems security. In addition to the theory component, a lab component of 2 hours per week was also included. The lab supplemented the theory part with programming exercises including thread programming, process programming, Inter process communication, shell programming. The course was taught in the Linux environment.

Evaluation
For assessment of PDC topics in OS, students were evaluated using assignments, home works, projects, and mid and final exams. In addition, weekly labs were also scheduled to teach students. A comprehensive evaluation was made including theory as well as programming portions. For instance, a programming assignment including producer consumer portion was included. Similarly, for theory part concepts like memory and process scheduling were considered very important for assessment. Concepts such as fork and pipes were included in the lab portion. The broad evaluation pattern allowed students enough opportunities for learning, peer-collaboration, assessment, and improvement.

Algorithms Analysis and Design
The Algorithms course is focused on explaining, enhancing, and imparting techniques, methods, and approaches for analyzing and designing sequential and parallel algorithms. The students have limited prior knowledge of sequential algorithms. So a major portion of the course was focused on the sequential algorithms. This also provides an opportunity to explain benefits for parallelization for computationally expensive algorithms. For PDC topics, the coverage was mainly focused on the ‘Algorithms’ topics. These include dependencies, Task graphs, recursion, and Asymptotic notations. The concepts were introduced using examples. For instance, divide and conquer strategies in merge sort helped in achieving speedup and reducing cost. Similarly, Fibonacci series was explained using parallel acyclic graph.

Evaluation
For evaluation of PDC topics, students were given assignment exercises related to divide and conquer strategies, brute force, and minimum spanning trees. A programming assignment related to graph traversal problem was also assigned. For examination, evaluation questions included computation of NxN matrix in parallel and task graph and work stealing. The broad coverage of theory and programming based topics allowed extensive evaluation.

IV. STUDENT FEEDBACK AND LESSONS LEARNED

Feedback
A mixed response was observed from the students in the evaluation.
- For the Data Structures course, based on class feedback during lecture, it was observed that the students had difficulty in comprehending parallelism in the third semester. One of the reasons was that the students were learning new language Java in this course for their assignments and project and they were trying to cope up with that. However, the performance of students was satisfactory in evaluations.
- For the Operating Systems course, we observed that students performed relatively lower as compared to non-PDC topics. However, this difference was not significant.
- For the Algorithms course, the students liked the parallelism topics. Their performance was also better in the PDC topics.

We hope that student satisfaction and evaluation would improve in the subsequent offering of these courses.

Lessons Learned
Following are the important lessons learned.
- Programming provides an effective method for learning. Theoretical concepts should be aided through programming examples and exercises.
- For early semester courses, an incremental approach of PDC topics is recommended.
- TCPP recommended curriculum is broad and multiple courses are needed in order to comprehensively cover the course.

ACKNOWLEDGMENTS
The authors are thankful to TCPP for providing recommended topics. In addition, the authors are thankful to Intel and NSF for facilitating grant through early adapter plan.