Unplugged Activities to Introduce Parallel Computing in Introductory Programming Classes: An Experience Report

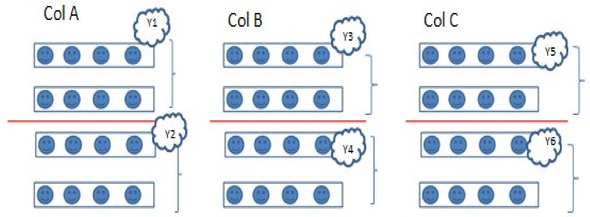
Sheikh Ghafoor, David Brown, Mike Rogers
[1]Department of Computer Science, Tennessee Tech University, USA

Overview

- Early programming courses present complex concepts to novice students
- Introducing parallel programming in early classes alongside traditional sequential programming is challenging
  - ACM recommends PDC as a required Knowledge Area for undergraduate CS curricula
- Unplugged activities, which are activities done without programs and which usually have physical activity with visual component have shown to engage students and convey concept in a way that enhances learning[3].
  - Unplugged activities may help alleviate some of difficulties for students
  - Unplugged activities have been shown to increase student interest, and to enhance student understanding of CS programming concepts [1]
  - We have used unplugged activities to teach PDC concepts before introducing parallel programming.

FINDING THE YOUNGEST IN CLASS

- Students are asked to find the youngest student in the class
- The student on the aisle side is in charge of finding the youngest in their row
  - Told to communicate column-wise and then row-wise to find the youngest in the class



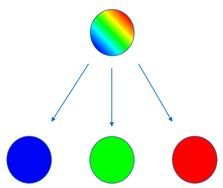
PDC Concept	Bloom Level
Distributed Computing Architecture	K
Communication Pattern	K
Parallel Overhead	K
Sequential Dependency	K
Load Imbalance	K

- Extensions
  - Let the class partition themselves
  - If there were 1,000 students, could they find the youngest faster than if there were only 100?

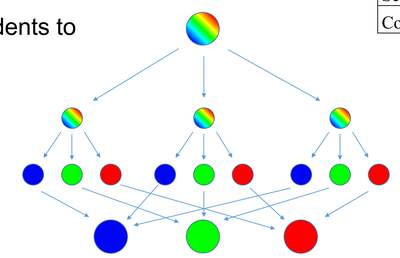
M&M SORTING

- Sorting M&Ms according to color
- Have one student sort M&Ms into separate bowls and have another student time it
- Have three students sort the M&Ms and have another time it
- Compare the time taken by three students to single student

PDC Concept	Bloom Level
Serial vs Parallel (Speed UP)	K
Decomposition	K
Parallel Overhead	K
Sequential Dependency	K
Concurrency	K



Serial (1 student)



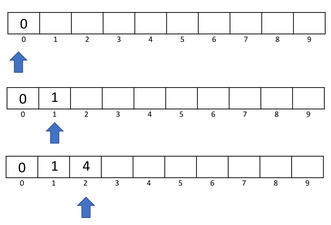
Parallel (3 students)

- Extensions
  - Add task of counting each color M&M after sorting is finished
  - Distribute M&Ms in different way

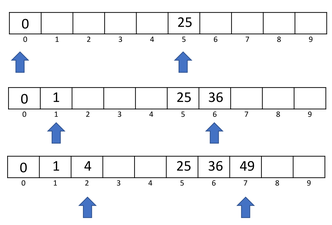
ARRAY IN PARALLEL

- Draw an array on the whiteboard with boxes
- Have a student populate the array using some function and time it
- Have one student start at beginning and another student start in the middle and time it
- Compare the two times

PDC Concept	Bloom Level
Serial vs Parallel (Speed UP)	K
Data Dependency	K/C
Concurrency	K/C



Serial (1 student)



Parallel (2 students)

- Extensions
  - Have more students work at once
  - At a critical point, adding students will actually result in a slower time

Evaluation

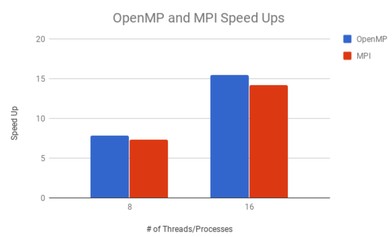


Fig:

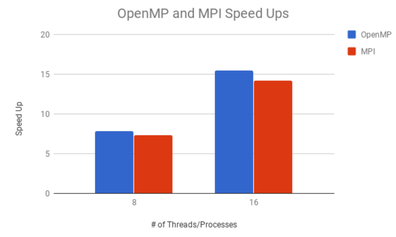


Fig:

CONCLUSION

- Our experiences show that using unplugged activities to introduce the PDC concepts reduce the barrier to learn parallel programming.
- Our preliminary evaluation indicates that students learns better when unplugged activities are used with traditional plugged activities compared to using traditional plugged activities only.

REFERENCES

[1] B. Rodriguez, C. Rader, and T. Camp, "Using Student Performance to Assess CS Unplugged Activities in a Classroom Environment," in Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education – ITiCSE 16, 2016, pp. 95 – 100
[2] C. Watson and F. W. B. Li, "Failure rates in introductory programming revisited," in Proceedings of the 2014 conference on Innovation & technology in computer science education - ITiCSE 14, 2014, pp. 39 – 44.
[3] T. Bell, J. Alexander, I. Freeman, and M. Grimley, "Computer science unplugged: School students doing real computing without computers," New Zeal. J. Appl. Comput. Inf. Technol., vol. 13, no. 1, pp. 20 – 29, 2009.
[4] "iPDC Modules", https://www.csc.tntech.edu/pdcincs/index.php/ipdc-modules/

ACKNOWLEDGEMENT

This project is supported by the U.S. Air Force (USAF) and Oak Ridge National Laboratory (ORNL) through the USAF-ORNL R&D collaboration. The research used resources of the Oak Ridge Leadership Computing Facility (OLCF) at ORNL. The authors also acknowledge the support by the Center of Management, Utilization and Protection of Water Resources at Tennessee Technological University (TTU).