# Introducing PDC concepts with spatial computing

Satish Puri
Marquette University

***Abstract*—From online maps, such as Google Maps, to consumer GPS devices, we immensely benefit from spatial computing. As a new early adopter (started in 2018) from Marquette University, I want to present a poster describing my project in which I plan to introduce parallel and distributed computing (PDC) concepts using topics from spatial computing domain. Line segment intersection reporting is chosen as an exemplar and the overall material is built around it. This project should be useful to the broader community under the umbrella of spatial computing that includes instructors who teach algorithms, computational geometry, computer graphics, spatial databases, geographic information system (GIS), etc.**

## I. Introduction

Spatial computing deals with shapes/geometries e.g. polygons. The material discussed here deals with data that is not in image format, rather as sequence of 2D coordinates. The data structures and algorithms involved are somewhat specialized, as such spatial computing is typically taught as a chapter in Algorithms/Computer Graphics/Computational Geometry courses. Other courses where this topic fits are Spatial databases and Geographic Information Systems.

The benefit of using examples from spatial domain is that it can be interesting to students because they can visualize the data on a map or a GUI when compared to standard material that covers matrices and arrays. Moreover, spatial algorithms involve irregular computations and exhibit load imbalance problem naturally. So, students can get exposed to real problems that are challenging to efficiently parallelize. Using the material mentioned here (source code and sample data sets), instructors may introduce PDC content in their related courses.

With the rise of Big Data, spatial domain also need parallel and distributed computing for efficiency and speedup. This has been addressed in the research community as evidenced by projects like CyberGIS and SpatialHadoop. Our project has educational emphasis around the broad area of spatial computing. With geo-spatial big data and data science as motivating factor, we observed an educational gap where spatial computing was not getting addressed in the regular coursework for undergraduate students at Marquette University. To address this gap in curriculum, we organized a three day Spatial Data Analytics Workshop for undergraduates in January, 2017. The last day of this workshop was dedicated to using PDC techniques in spatial computing for computer science, information systems, and engineering majors. Some of the material discussed here was a part of that workshop.

For homeworks, one problem is identified. Since, it is an undergraduate course, the problem needs to be simpler. Instructor discusses how to parallelize an algorithm to solve the chosen problem using a shared memory and a distributed memory implementation. Then, students are asked to tackle the same problem using different language extensions or frameworks. Doing the same exercise but with different implementation, students get exposed to different ways of computational thinking. I have chosen segment intersection reporting problem.

Segment intersection problem: Given a set of n lines, report the intersection points. There is an $O(n^2)$ algorithm that uses brute force technique and tests each line segment against the other line segments. For testing scalability, I share large publicly available spatial datasets from few gigabytes to 100 GBs. Parallelizing point in polygon problem builds on top of segment intersection.

**Challenges**: As students do assignments, they encounter few stumbling blocks like how to store the result efficiently for sparse output. Another challenge is to do data decomposition among different threads for very large datasets. Students will notice that partitioning point data is easier. However, partitioning Rivers and Road data leads to lot of duplication of input data. These lead to interesting discussion and reading assignments of current literature in this area. Due to the difference in density of spatial objects in a map, some partitions will have less work and others will end up with more work. Load-balancing among threads/processes can be then discussed. Round-robin, master slave task scheduling can be experimented.

For shared memory programming, OpenMP and OpenACC with parallel loops and reductions could be discussed. Then, data transfer clauses should be discussed for OpenACC. New assignments can be created for students where they try OpenACC implementations for reporting segment intersections. I chose OpenACC considering its simplicity when compared with CUDA. I decided not to use CUDA because it requires more programming effort. In order to familiarize students with GPU and OpenACC, I covered vector addition and Jacobi Iterative method that solves Laplace equation for heat transfer using OpenACC pragmas. I found chapter 19 of book "Programming Massively Parallel Processor" very useful for covering OpenACC.

**MPI assignment**: Define derived data types for shapes like points, line segments, rectangles, etc. Writing custom reduction operators for shape objects e.g. union of rectangles or min operation for a set of of points. I plan to cover parallelization of range query operation using MPI.

**Lessons learnt**: Some of the assignments were tried in graduate level classes. For example, a project involving line segment intersections using OpenMP and OpenACC was taken up by two students. They compared their implementations with sequential libraries and reported 10X speedup using GPU. For the MapReduce topic, word count, inverted index, page-rank implementations in Hadoop and Spark were discussed in the class. As a homework, segment intersection reporting problem was assigned where they had to write map and reduce functions to find all the segment intersections. A correct solution will choose the intermediate key-value pairs appropriately in the map phase. However, none of the students came up with the correct MapReduce implementation for segment intersection reporting problem. Typical mistakes were due to assuming that each mapper has access to the entire file and performing entire file reading in the map phase. Subsequent lecture focused on correcting the misunderstanding.

In the poster, instructional materials developed so far including sequential implementations using C/C++/Java and a few parallel implementations using threads will be presented.