Parallelized Traffic Simulation in C++

Zach Alaniz
Department of Computer Science
College of Science and Engineering
Texas Christian University
Fort Worth, Texas 76129
Email: z.a.alaniz@tcu.edu

Saby Sahoo
Department of Computer Science
College of Science and Engineering
Texas Christian University
Fort Worth, Texas 76129
Email: s.sahoo@tcu.edu

Bradley Schoeneweis
Department of Computer Science
College of Science and Engineering
Texas Christian University
Fort Worth, Texas 76129
Email: b.schoeneweis@tcu.edu

***Abstract*—This semester research project investigates the potential speedups and scalability of running a traffic simulation in parallel using C++ and OpenMP. These speedups are evaluated by comparing the sequential performance to the parallel performance when varying thread counts for a fixed-size, randomized input. While a large percentage of the program is inherently sequential due to computation, the process of transitioning from one state to another was parallelized.**

***Index Terms*—OpenMP, Cellular Automata, Parallel Simulation, C++**

I. Introduction

Extensive research and improvement has been made over the years in the field of study that is parallel computing. Falling within this domain of study is the topic of simulation, and more specifically, traffic simulation. The ability to optimize light scheduling and move cars from one point to another in an efficient, organized manner is paramount to day-to-day travel. Being able to apply parallel computing to simulate increasingly large traffic environments leads to many interesting research topics and advances in road safety, traffic capacity, economic impact, among many others. The topic of simulating traffic flow for increasingly large sizes of roads is the topic of interest in the following research. Real life traffic flow allows for movement when possible at all times, and by utilizing parallel computing topics and tools, the speedup of a traffic simulation should prove beneficial in seeing how a particular road and intersection works for an increasingly large amount of cars funneling in and out.

Traffic simulation naturally lends itself to parallel ideas. Decomposing the movement of cars into a sub-domain of problems and updating traffic flow in parallel, just as normal traffic would run, is one method to improved performance. Thus, a parallel solution to traffic flow needs to be able to parallelize any movement and update that could be happening concurrently. The dynamic nature of traffic also plays a large role in this type of simulation, but was handled to leave a larger emphasis on the updating of car movements. The following research being presented simulates traffic flow through a four-way stop, which includes two-lane roads from all directions. The simulation was built with C++ utilizing a cellular automaton approach, and parallel integration was made with OpenMP, an application programming interface (API) that supports shared memory multiprocessing. The following is an overview of the remainder of the paper: Background research on every tool and idea used as well as reasons for why they were chosen, sequential solution and design, parallel integration and results, and finally a summary with conclusions of our research.

II. Background and Related Research

Chances are if you have rode in a vehicle, you have come across a four-way stop being controlled by traffic lights. A four-way traffic stop has roads leading from all four directions (north, east, south, west) where each direction takes a turn moving towards its destination as directed by the controlling traffic light. The traffic light operates in three states: red, green, and yellow, where green means go, red means stop, and yellow means slow down. Assuming each direction is at least a two-lane road, any car driving on the outer most lane is capable of taking a right turn at the traffic light intersection if oncoming traffic is clear, regardless of the state of the light. Cars in the innermost lanes are also capable of making a left turn, given that their light is currently green. The light itself cycles on a timer allowing for each lane to have its own turn to proceed, which creates the flow of traffic. This four-way model is what our simulation is derived from, which will be explained in a later section.

The placement of cars along any road is relatable to a data model studied in computer science and related fields known as *cellular automata*. Formally, cellular automota

is defined as a collection of cells on a grid that evolves during a number discrete time steps according to a set of rules based on the states of neighboring cells. The rules are then applied iteratively for as many time steps as desired [1]. Cellular automata has been studied since the early 1950s and started as a model for biological systems. The grid and state like approach of cellular automata was taken and applied to our simulation too represent state of traffic flow and position of cars within a street.

III. Parallelization Technique

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

[2]

IV. Summary of Results

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

V. Conclusions, Lessons Learned, and Future Work

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

References

[1] M. J. Lazlo, *Computational Geometry and Computer Graphics in C++*. Prentice Hall, 1996.

[2] J. O'Rourke, *Computational Geometry in C*. Cambridge University Press, 2nd ed., 2005.