Load Balancing a 2D Cellular Automata Over a Cluster

Jared DuPont - JDuPont13@winona.edu - Computer Science
Advisors: Dr. Sudharasan Iyengar, Dr. Mingrui Zhang

Introduction

Cellular Automata were first discovered by Stanislaw Ulam and John von Neumann in 1940 and are used in cryptography, modeling bacterial and population growth, and more. Cellular automata are typically a grid of cells that each hold a state. The state of each cell is updated every cycle by having each cell look at the cells that surround it and change depending on them. This is very computationally intensive when the size of the grid gets large which is why the job needs to be split up between many computers.

A common method of computational work distribution is using a cluster. A **Cluster** is typically many smaller computers, called nodes, connected to each other over a network in close proximity. A master computer is also connected to this network who's purpose is to issue commands and tasks to the nodes.

The goal of this research is to develop a system to share the work of a cellular automata across many computers in a cluster dynamically. The advantage of this research is that the cluster continuously adjusts how much of the grid each cell is responsible for so that the work load of each computer is the same.

Hypothesis

Dynamically adjusting how much area that each computer is responsible for in a cellular automata will decrease the total idle time of the simulation when compared to the same simulation without dynamic adjustment.

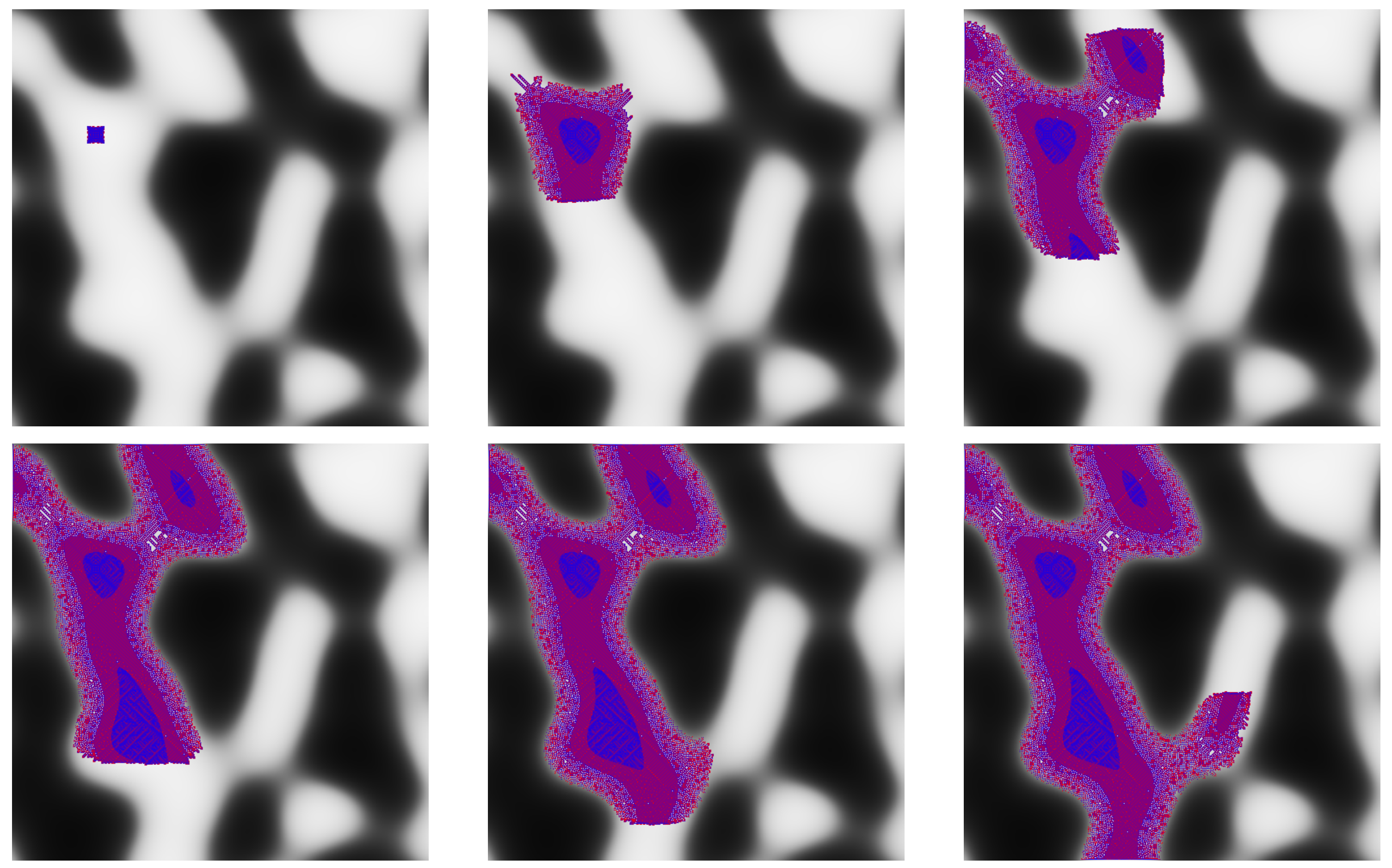
Method

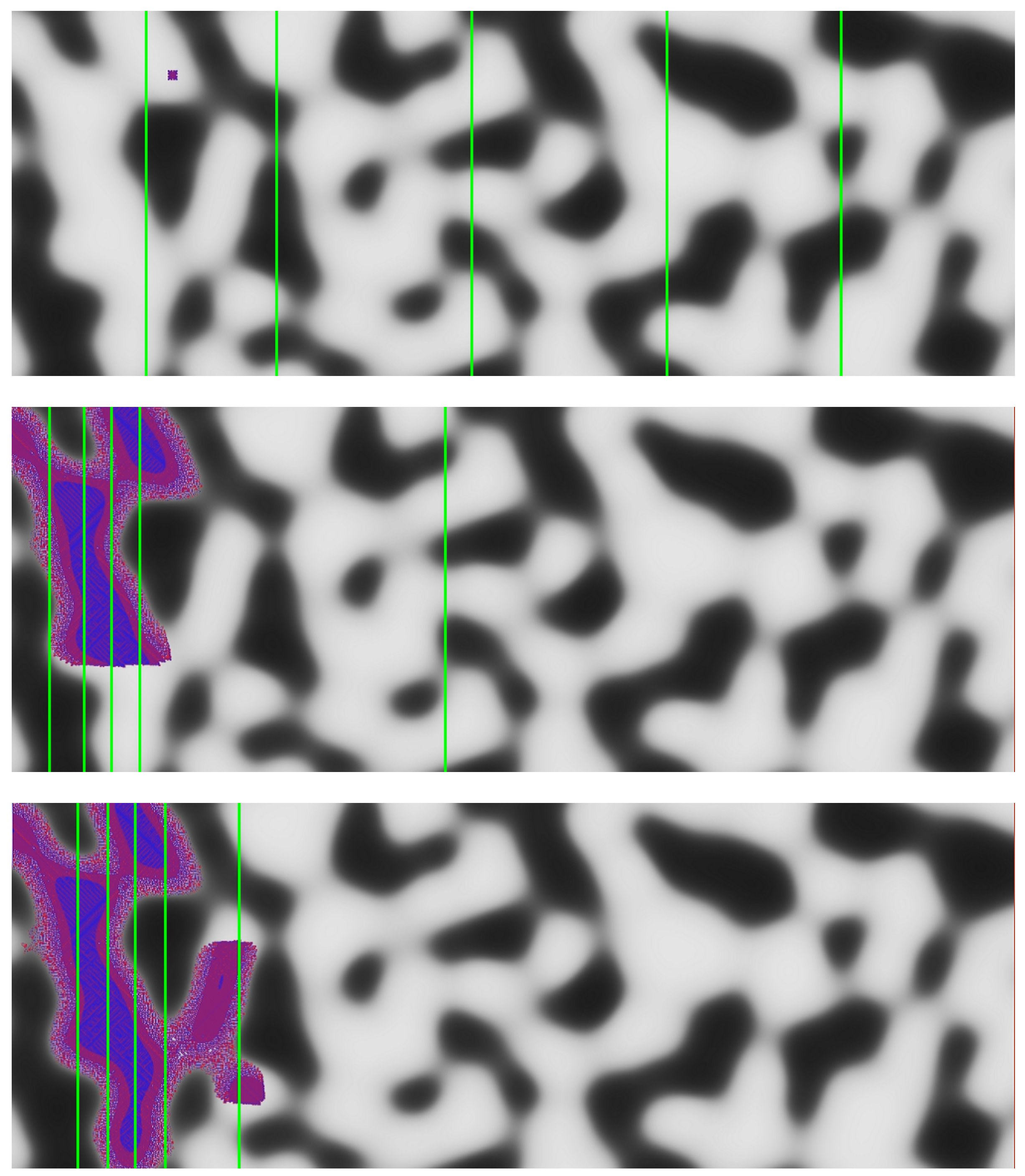
An example cellular automata is used that was designed to create localized areas of high intensity computations to exaggerate the effect of the load balancing.

The load balancing algorithm and a simpler, non load balancing version were deployed to a cluster of 6 nodes and the following timing data was recorded from them:

- Computation time: The time spent actually doing the simulation
- Idle time: The time spent waiting for other nodes to finish

This data was used to evaluate if load balancing had any effect on the total computation time or idle time.

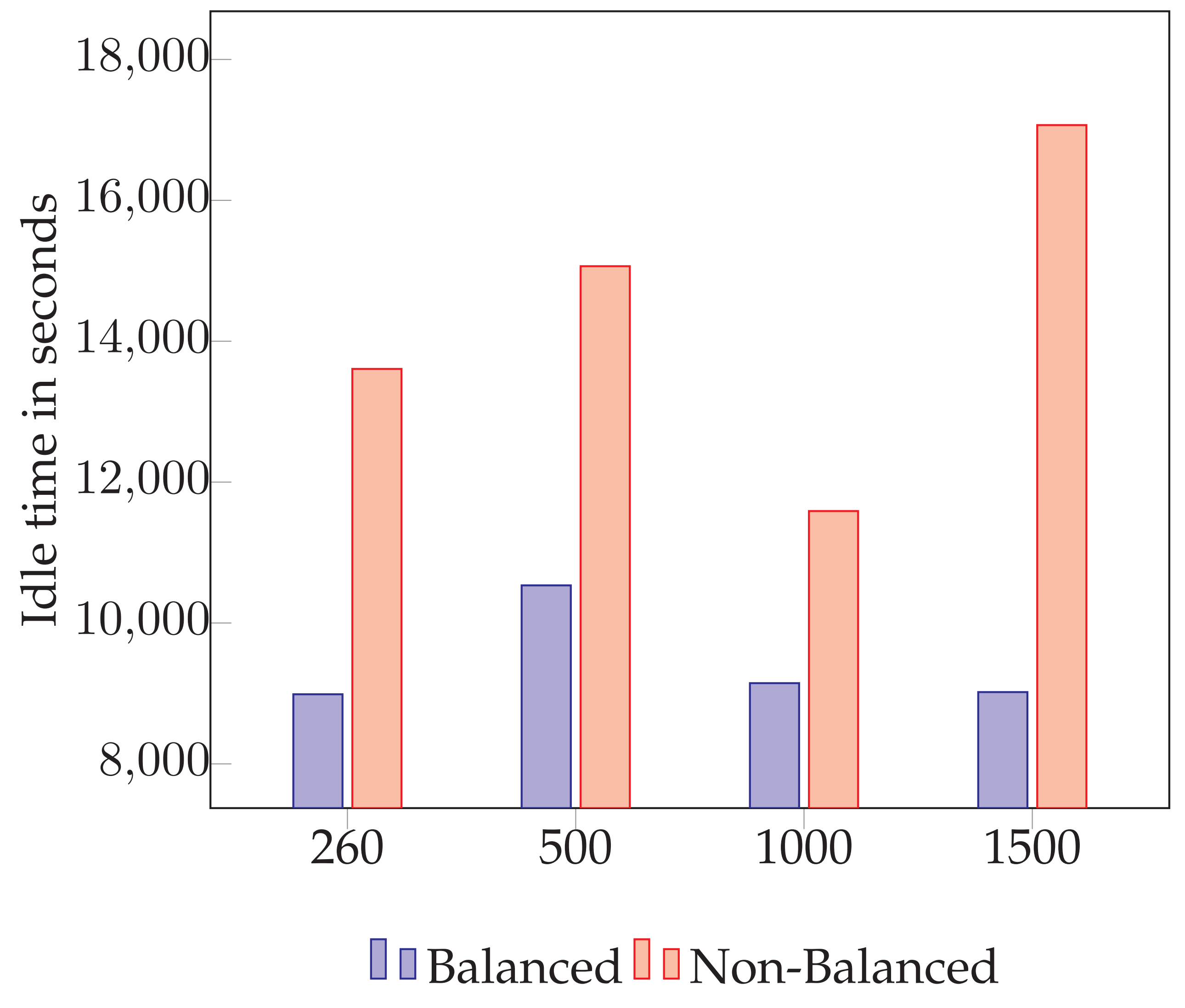

Step 250, 2250, 4250, 6500, 8750, and 11250 of a simulation.


Boundaries of the nodes adjusting themselves as the simulation progresses.

Results

Four different sized simulations were done using a balancing and a non-balancing algorithm for a total of 8 simulations. When the simulation starts, each node is assigned a starting width and a height. After the simulation starts, the width may change but the height always remains the same. The four starting widths chosen are 260, 500, 1000, and 1500. During the simulations every 50 steps worth of computation and idle times were added up and logged.

Total time spent idle for each of the simulations.



Conclusion

In both versions of the algorithm, the entire simulation is held back by the slowest node which presents a problem. In the balancing version of the algorithm, the slower nodes shrink and the faster nodes grow until all nodes take roughly the same time to simulate. In the non balancing version, only one or two nodes actually simulate the growth while the rest wait idle for them to finish. Balancing the simulation drastically reduces the amount of time spent idle as seen in the bar graph.

WINONA
STATE UNIVERSITY
Computer Science Department
http://cs.winona.edu