Using High Performance Computing To Model Cellular Embryogenesis

Gerard Vanloo[1]
gerard.vanloo0@gmail.com

Dr. Chung Ng[1]
chung.ng@morehouse.edu

Kison Osborne[1]
ozuboon95@gmail.com

Dr. Kwai Wong[2]
kwong@utk.edu

Ben Ramsey[2]
bramse10@vols.utk.edu

Dali Wang[3]
wangd@ornl.gov

Dr. Zhirong Bao[4]
baoz@mskcc.org

ABSTRACT

C. elegans is a primitive multicellular organism (worm) that shares many important biological characteristics that arise as complications within human beings. [1] It begins as a single cell and then undergoes a complex embryogenesis to form a complete animal. Using experimental data, the early stages of life of the cells are simulated by computers. The goal of this project is to use this simulation to compare the embryogenesis stage of *C. elegans* cells with that of human cells. Since the simulation involves the manipulation of many files and large amounts of data, the power provided by supercomputers and parallel programming is required.

CCS Concepts

• **Computing methodologies~Massively parallel and high-performance simulations** • *Computing methodologies~Agent / discrete models* • *Computing methodologies~Simulation languages* • Computing methodologies~Simulation by animation • *General and reference~Performance*

Keywords: RepastHPC; Parallel Programing; C. elegans; VisIt; NetLogo; Simulation

XSEDE16, July 17-21, 2016, Miami, Florida
 ISBN 978-1-4503-4755-6/16/07$15.00
DOI: http://dx.doi.org/10.1145/2949550.2949576

1.INTRODUCTION

A NetLogo [2] simulation was developed in order to quickly obtain experimental results involving *C. elegans*, a popular biological specimen. Given a time range and a file of initial cells, the program would calculate and record the paths of each microorganism as it moves and interacts with neighboring cells. These communications between the embryos are described through the "cell focusing" hypothesis proposed by Marcus Bischoff and Ralf Schnabel. [3] However, during the implementation process, it became apparent that the NetLogo simulation would not be sufficient for the problem at hand.

NetLogo, an agent based modeling software written in Scala and Java, was originally used to create the simulation. This type of application was developed for showing the results of agent-to-agent interactions over a period of time. While NetLogo is useful for these simulations, it is crippled by some limitations. One of these major drawbacks would be its lack of a parallelization component. Due to this, the entire program is executed on one core (serially). This causes slow results as the data set becomes larger. In trial tests, the simulation (which generated 292 cells) would take up to two hours to finish execution.

As research of this organism is vital, results from the program are required to be generated in a shorter amount of time.

RepastHPC [4], like NetLogo, is an agent based simulation toolkit that was written in C++. Unlike NetLogo, RepastHPC is specifically made to be used in a high performance computing environment by making use of the C++ parallelization library, Boost. This gives the toolkit an advantage over serial programs. Unlike NetLogo, it lacks of a visualization component. Therefore, one cannot see a visual animation of the RepastHPC simulation while it is running.

s

However, this effect of this limitation is lessened by the use of VisIt [5], a visualization program that can also make use of parallel programing. VisIt can read a variety of files and, though it is written in C/C++, can make use of other languages like Python to convert text files into a file format it can visualize.

2.METHOD

The *C. elegans* simulation begins with four initial cells and after over a 10,800 tick cycle — a tick is the unit of time in the

simulation — generates 292 cells. During this time, each cell, otherwise known as an agent for the purpose of the simulation, will move in a linear fashion towards a target point. In addition, the cells will divide and produce daughters at specific times. The parent will then become one of the newly created cells. After a period of time, the agents will stop dividing but will continue to move. This duration is specific to each cell.

These destinations and division times are determined by the experimental data gathered by the Memorial Sloan Kettering Cancer Center. [6] The files containing these values are read into the simulation. The program will then save the x, y, and z coordinates, name, size, and various other ids of each agent into a text file every tick.

The NetLogo simulation was ported into a C++ code so that it could interface with the RepastHPC toolkit.

The RepastHPC simulation was tested on University of Tennessee's cluster, RockFrog using four processes.

In order to visualize the data, Python scripts are used to convert the results generated by the simulation into a file that can be read and visualized by VisIt. VisIt is then uses this data in order to create a video of the simulation.

In addition to the *C. elegans* simulation, a simpler program was created in order to show RepastHPC's scalability. This code used a less complex algorithm in which the cells would randomly wander in a linear fashion. However, the agents would not attempt to divide and create more cells. Rather all the cells required in the simulation are created in the beginning of the execution. By doing this, the division of agents between processes can be as even as possible.

During the initialization phase, the first cell made by a process would be marked as infected. This cell would then pass on this trait to the closest neighboring one after the first 50 ticks. Then every 50 ticks, any infected agent would mark the closest neighboring cell to its location as infected. This "disease" simulation offers better results in terms of testing RepastHPC's strength as there is more cell-to-cell interaction across processes in comparison to the *C. elegans* model.

This simulation was executed on the University of Tennessee's supercomputer, Darter.

3.RESULTS

3.1.*C. elegans Serial and Parallel Comparison*

On average, the simulation currently produces about 292 cells.

NetLogo takes six minutes to fully execute the *C. elegans* simulation without the "cell focusing" implementation.

By comparison with NetLogo, the RepastHPC simulation (also without "cell focusing") completed in under a minute when executing on four processors of a small cluster.

Through the use of the Python scripts, the data is visualized through VisIt in two ways: a general animation and a cell tracking simulation.

Figure 1a shows a frame of this general video. The colors are arbitrary; however, they show the daughter cells that have been generated by the parent cell. There are four starting cells in the simulation, hence the four color animation.

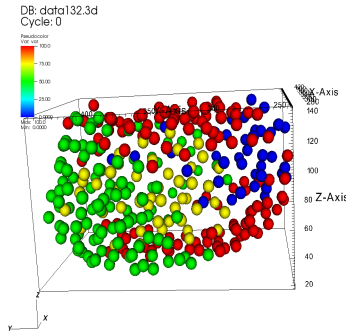

Figure 1a: VisIt Visualization of RepastHPC Simulation– General Animation

Figure 1b is a frame from the cell tracking animation. The Python script takes the name of a cell in the simulation and highlights it and all of its daughters green. Every other agent is colored blue.

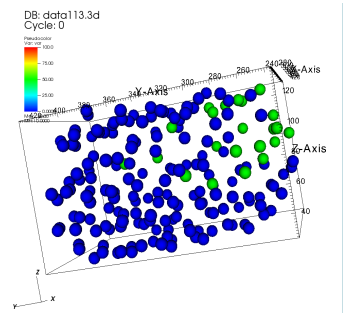

Figure 1b: VisIt Visualization of RepastHPC Simulation – Cell Tracking

The axes shown in both frames are the XYZ axis that VisIt displays while running the animation.

Despite this decrease in time, the program was still limited by the power of the cluster that the program was run on.

3.2.*RepastHPC Simulation Scaling*

With the change in execution medium (from RockFrog to Darter), the "disease" model could be used to determine RepastHPC's ability with bigger data sets.

First, the simulation was executed multiple times with 8 processes. Each time, the number of cells was incremented by either 500 or 1,000 cells. Depicted in Figure 2, as the number of cells becomes larger, the time taken by the eight processes does increase steadily by an average of roughly 11 seconds per 500 cells added.

The reason for this could be due to the number of synchronization calls that RepastHPC must make in order to keep the simulation stable. In order to do this, each process must make the following calls: 1) mark all cells that have moved out of that process's portion of the three dimensional grid, 2) move those agents to the appropriate adjacent process, 3) create copies of agents close to the boundary line to the adjacent process so that it is aware of it, and 4) update copies with the current information from their home process.

Since these function calls are made every tick cycle (after the agents have moved), the processes are constantly communicating

with each other. In addition, there is the synchronization between ticks for all processes that will also cause more time to be put into this process. As the number of agents grows, the amount of information passed between the processes increases. Thus the execution time becomes larger.

Figure 2 shows this relationship to be largely linear.

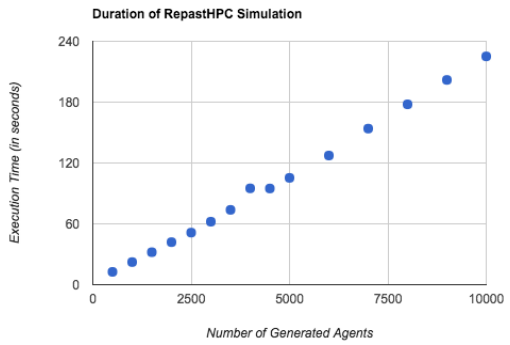

Figure 2: Scaling of RepastHPC (Fixed Number of Processes – 16)

Next, the simulation was run with 1,000,000 agents. Each time, the number of processes used was decreased by a power of 2 (32, 64, 128, 256, 512, 1024 respectively to the data shown in Figure 3).

As shown, a large data set requires an adequate amount of processes in order to run the simulation in a timely fashion. As mentioned previously, there are four function calls that are used in order to keep the program stable in addition to synchronizing between tick cycles.

For the same reasons mentioned, the execution time for simulations using a small number of processes is greatly affected by these synchronization times such as a simulation using 32 processes (which lasts for about 7.23 hours) since these processes have many more cells to scan through in the synchronization step than a simulation that has the agents divided among 1024 processes (which completes in roughly 9 minutes). While the number of processes increases by a factor of two, the time taken decreases on average by a factor of 2.88 minutes.

Figure 3 shows the relationship between the execution time and the number of processes used is similar to that of the (a / x) equation where a is some positive constant number.

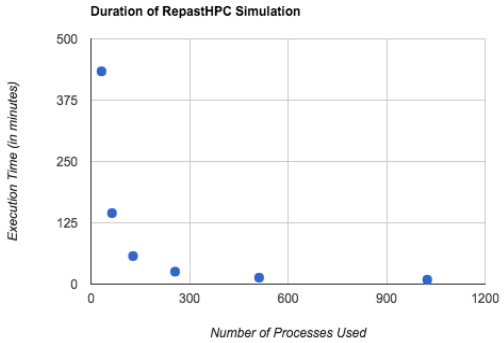

Figure 3: Scaling of RepastHPC (Fixed Number of Agents – 1,000,000)

4.CONCLUSION

RepastHPC shows definite promise of being the faster medium of the *C. elegans* simulation. While running a variation of the "cell focusing" algorithm (which only generated 292 cells), NetLogo did not finish until a little over two hours later. Meanwhile, although a much less complex program, RepastHPC was able to complete a 1,000,000 agent simulation in about 9 minutes. This shows that RepastHPC can handle a much larger capacity, but it also produces the results several times faster than NetLogo.

Although the goal is to use the "cell focusing" hypothesis as the guide of cell movement, due to time limitations, the agents moved independently to the other in the RepastHPC and NetLogo comparison. The "cell focusing" algorithm is a rather complex movement process and requires more time to be throughly tested. In addition to this, the entire simulation will then be verified with the experimental data to prove its correctness.

Time limitations prohibited more extensive tests to show RepastHPC's scalability. There are some other methods that could be run to further show RepastHPC's power in agent based modeling. The toolkit has one other method known as network projection which can be used independently or cooperatively with its spatial projection functions. In addition, more tests can be done to see whether or not the gradual increase in agents over the course of the simulation (dynamic creation of cells) causes significant changes in time.

Currently, there have not been any plans to release the RepastHPC nor NetLogo code; however, the VisIt Python scripts are available. [7]

5.ACKNOWLEDGEMENTS

Our thanks to National Science Foundation, Joint Institute For Computational Sciences, Memorial Sloan Kettering Cancer Center, Kison Osborne (Morehouse College), Scott Simmerman (Oak Ridge National Laboratory), and John Murphy (Argonne National Laboratory).

This material is based upon work performed using computational resources supported by the University of Tennessee and Oak Ridge National Laboratory's Joint Institute for Computational Sciences (http://www.jics.utk.edu). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the University of Tennessee (UTK), Oak Ridge National Laboratory, or the Joint Institute for Computational Sciences.

This work is also sponsored by the National Science Foundation through REU award #1262937. with additional support from the National Institute of Computational Sciences at UTK.

6.AUTHOR INFORMATION

1. Morehouse College, 830 Westview Dr SW, Atlanta, GA 30314
2. University of Tennessee, Knoxville, TN 37996-2030
3. Oak Ridge National Laboratory, 1 Bethel Valley Rd, Oak Ridge, TN 37831
4. Memorial Sloan Kettering Cancer Center, 1275 York Avenue, New York, NY 10065

7.REFERENCES

[1] Anon. College of Biological Sciences. Retrieved July 22, 2015 from https://www.cbs.umn.edu/research/resources/cgc/what-c-elegans.

[2] Marcus Bischoff and Ralf Schnabel. 2006. Global cell sorting is mediated by local cell–cell interactions in the C. elegans embryo. Developmental Biology 294, 2 (2006), 432–444. DOI:http://dx.doi.org/10.1016/j.ydbio.2006.03.005

[3] Wilensky, U. (1999). NetLogo. http://ccl.northwestern.edu/netlogo/. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.

[4] N. Collier and M. North. 2012. Parallel agent-based simulation with Repast for High Performance Computing. *Simulation* 89, 10 (June 2012), 1215–1235. DOI:http://dx.doi.org/10.1177/0037549712462620

[5] Hank Childs et al. 2012. VisIt: An End-User Tool for Visualization and Analyzing Very Large Data. *High Performance Visualization Chapman & Hall/CRC Computational Science Enabling Extreme-Scale Scientific Insight* 1 (2012). DOI:http://dx.doi.org/10.1201/b12985-21

[6] J.L. Moore, Z. Du, and Z. Bao. 2013. Systematic quantification of developmental phenotypes at single-cell resolution during embryogenesis. *Development* 140, 15 (2013), 3266–3274. DOI:http://dx.doi.org/10.1242/dev.096040

[7] Kison Osborne. 2015. Agent Based Visualization of C. Elegans Embryogenesis at Cellular Resolution. (August 2015). https://www.jics.tennessee.edu/files/images/csure-reu/2015/kison-gerard/report1.pdf