Using High Performance Computing To Model Cellular Embryogenesis

C. Elegans RepastHPC Simulation Manual

Written By: Gerard Vanloo

Last Updated: August 7, 2015

Table of Contents

Abstract and Overview

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.

Overview

The objective of this project is to replicate the data of the beginning stages (embryogenesis) of a *C. Elegans* worm cell from experimental data through computer simulation. A preliminary version of simulation was done using the agent based simulation software, NetLogo. However, NetLogo is unable to handle large data sets as the software is unable to work in parallel and thus would only be able to perform such a simulation given an impractical amount of time. Thus the simulation has been ported to RepastHPC to solve this issue. A visualization program, VisIt, is used to show to the results of the RepastHPC simulation. This manual will cover only the RepastHPC process. Consult the VisIt manual for instructions for visualizing results from RepastHPC.

[1]Caenorhabditis Genetics Center, College of Biological Sciences, University of Minnesota. “What is C. elegans?”. *College of Biological Sciences, University of Minnesota*. July 22, 2015. https://www.cbs.umn.edu/research/resources/cgc/what-c-elegans

Basic Algorithms

Every time step, all cells must:

- Wander
  - Move in a linear path
- Divide
  - If parent cell is ready to divide,
    - New cell becomes the first daughter
    - Parent cell becomes the second daughter
  - If cell has outlived division cycle,
    - Stop dividing
- Save progress(*)
  - X, Y, Z Coordinate
  - Name
  - Size
  - Various IDs

* = Frequency will change

Program Documentation

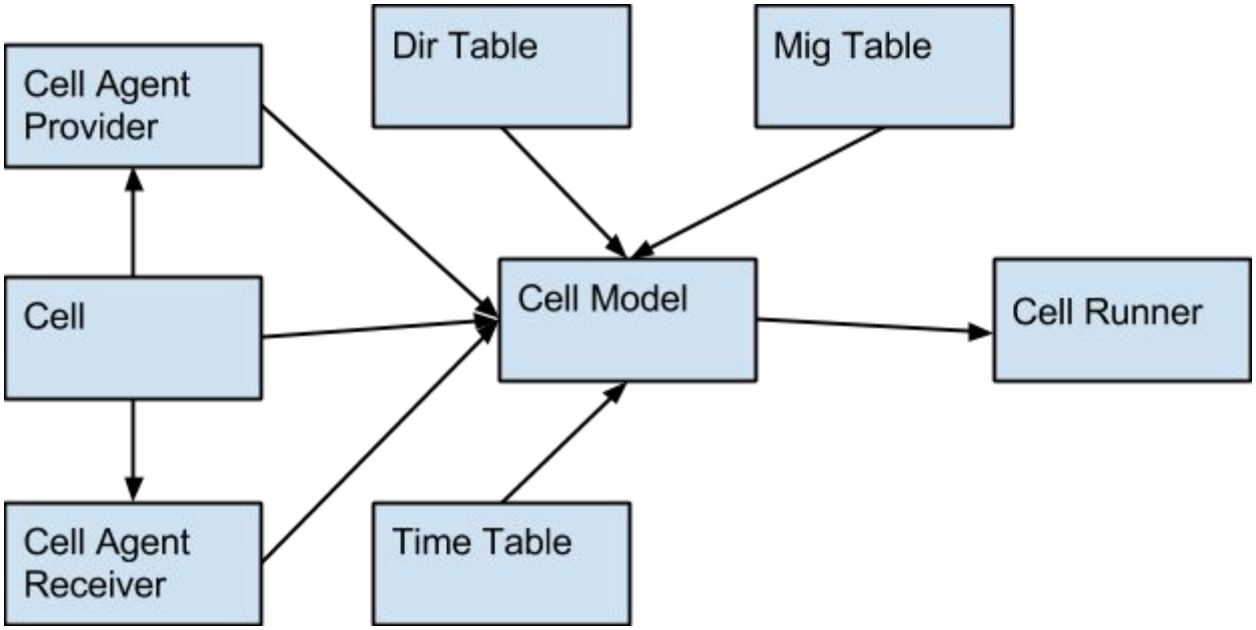
Note: Those with stars(*) have a more detailed instruction located in the RepastHPC tutorial: http://repast.sourceforge.net/hpc_tutorial/RepastHPC_Demo_01_Step_07.html and http://repast.sourceforge.net/hpc_tutorial/RepastHPC_Demo_01_Step_08.html.

Note: A makefile and configuration for compiling and executing the program has been included.

Source Code - (.cpp and .h files)

The following program uses seven classes (eight cpp files and seven headers).
Note: The .cpp files for each program have all been commented and have a more indepth coverage of each function.

- Cell_Runner.cpp
  - Simulation driver
  - Uses the config.props file to configure RepastHPC
- Cell_Model Class (.cpp and .h)
  - Contains all methods that the agents need to accomplish during the time of the simulation such as dividing and wandering
    - Includes Wander, Division, Save_To_File, and various initialization methods
- Cell_Agent_Provider Class* (.cpp and .h)
  - Contains methods for agents to travel between processors (mainly providing agent packages)
- Cell_Agent_Receiever Class* (.cpp and .h)
  - Contains methods for agents to travel between processes (updating and creating new agents)
- Cell Class* (.cpp and .h)
  - Contains methods for interacting with a cell (agent)
  - Contains agent package struct
    - Includes the serialization function used by Boost
- Mig_Table Class (.cpp and .h)
  - Contains methods for initializing and searching through the migration table
- Dir_Table Class (.cpp and .h)
  - Contains methods for initializing and searching through the direction table
- Time_Table Class (.cpp and .h)
  - Contains methods for initializing and searching through the time table



Class Interaction Diagram

Other Files

Note: (*) denotes that the location of the file or folder must be provided by the user in the model.props file.

- Text Files
  - Input
    - (*) Migration Table (alignedAndNormalizedPositionsNoHeaderWQ.txt)
    - (*) Direction Table (DivisionDirWQ.txt)
    - (*) Time Table (ABM_time_table_WQ.txt)
    - (*) Initialization File (initV<**version #**>.txt)
  - Output
    - (*) nuclei [Folder]
- Props Files
  - Config.props
    - Contains RepastHPC's configuration
  - Model.props
    - Contains variables needed for the simulation to be properly initialized
      - **Various File Locations**
      - Cell Focus Bool
        - For now, leave this turned off (= 0)
        - The Complex Wander method has not been created
      - Random Seed
        - Leave as is

- **Time Resolution**
  - Determines how fast the cell information is saved to file
- **Time Limit**
  - How long the simulation needs to run
- **Init File Number**
  - The number of the initialization file
- **Div Cycle Time**
  - The time needed before cells can execute a different **wandering path**
- **Num of Cells**
  - Expected number of cells that will be made by the end of the simulation

■ These variables (bolded) should be set through the GUI once it has been implemented

Current Results – Conclusion

Simulation	NetLogo (normal speed)	RepastHPC (serial)	RepastHPC (parallel without Spatial Updates)	RepastHPC (parallel with Spatial Updates)
Cells Created	192	192	190	190
Time Taken (estimated in min.)	6	<1	<1	34

The following ABM simulation of the *C. Elegans* organism has been implemented in both NetLogo and RepastHPC.

To define the parameters of the simulation type:

Normal Speed: NetLogo offers to run a simulation under many different speeds. If the current simulation was under a fast speed, it would complete in about the same time as the RepastHPC serial timing.

Serial: RepastHPC is run with only one process under this setting. It does not share agents with any other process.

Parallel without Spatial Updates: RepastHPC is run with four processes. It does not update the spatial network created, share agents, but does split the work of the simulation between processes.

Parallel with Spatial Updates: RepastHPC is run with eight processes. It does update the spatial network created, share agents, and split the work of the simulation between processes.

Justification for Parallel with Spatial Updates (Speed)

The reason for such vast differences between simulations is the functions calls from Wander(). Using the current initialization file, not every process receives a cell (the file incorporates four cells and in order to run the simulation, eight processes are required). Therefore, in the beginning of the simulation, there is a lot of wasted time since at least half of the processes involved are not actively doing anything.

As described in the Basic Algorithms section, every cell wanders and divides (attempts) every time step. When the simulation starts, a grid is created and split between the processes. This grid is the spatial network that RepastHPC uses to determine which cell belongs to which process. As the cells begin to move out of a process's section of the grid, it is either moved or

copied to the adjacent process. This is determined by how far out it is from the process's portion of the grid.

Before a cell begins to wander the buffer area (a boundary line per sa, between grid sections) must be copied so that each process is aware of cells that are approaching their part of the grid. After this, all copies of agents from other processes must be synchronized—updated—with the most current information. This data is gathered from the copied agent's home process. Then cells then begin to move.

After this, the grid must be synchronized to determine which cells have moved out of their process's grid. Any cell that has left the process's portion of the grid is then moved to the appropriate process which is responsible for the section they have moved into. Currently, the grid has a wrap around feature such that should a cell leave a boundary, it will then appear on the other side of the projection. Since each cell keeps track of the their own coordinates, the grid is mainly used for future implementations in determining which cell is closer.

The amount of necessary synchronization calls cause the simulation to be much slower than any other version of the simulation. In order to decrease this time, different ways of handling the synchronization calls will need to be looked into. Or a different algorithm must be implemented to determine how to find cells in a certain range very quickly across processes.

Justification for Parallel (Cells Produced)

Rather than using RepastHPC, Boost's all_reduce() function is used to determine the number of times each process accesses the time table (after the time table has been completed searched through for the first time, all cells are to stop no longer allowed to undergo the same division process). Since the call to this function must be placed in a manner in which all processes will execute it, there is a moment where some processes allow cells to dividing too many times.

For example, the time table has been accessed 188 times and should only be accessed 190 times. Three process access the time file. Since each process, during this cycle, accessed the file twice, all three processes will access the time table at the same time and thus the number of accesses increases to 194 rather than stopping at 190. This is the primary difference between the serial execution and parallel and more than likely the cause of the number of cells produced being different to the serial versions of this simulation.

Bug Log

Legend: <type of problem> - <cause> - <solved?>

Known Problems

- **Segmentation Fault - Requesting Agents - Solved**
    - Processes throw a segmentation fault when trying to request agents.
    - How Solved
        - Corrected Cell Package's serialization function.
            - Each archive variable must be separated with a semicolon.
        - Create agents before the simulation starts.
            - An AgentId is formulated as such (<some number>, <process the agent was created on>, <type>, <process the agent is currently on>). The AgentId is composed of only integers.
            - When requesting an agent, the agent requested must exist on the process it is being requested from. For example, if an agent with an AgentId of (0, 0, 0, 0) is requested, RepastHPC expects that that the process of an agent has a similar AgentId already in that process's context. So, if process 1 asked for the agent mentioned above (created by process 0), RepastHPC expects that an agent had an AgentId of (0, 1, 0, 1) on process 1. If the toolkit cannot find such an agent, a segmentation fault is the result.
- **Segmentation Fault - Balancing Discrete Space - Solved**
    - Processes throw a segmentation fault when attempting to balance the discrete space.
    - How Solved
        - Run the simulation with 8 processes.
        - As a note, 3D simulations must be run with (x * x * x) where x > 1 processes; x does not have to be the same number.
- **Segmentation Fault - Search For Closest Cell During Wander - Solved**
    - Processes throw a segmentation fault while moving.
    - How Solved
        - Before trying to do any operations involving the actual movement of moving a cell away, check to make sure that there is actually a cell to move to.
- **Boost Uniform Int Distribution Error (min_arg <= max_arg) - Solved**
    - Processes throw an exception when getting agents
    - How Solved
        - Ensure that every process's context has at least one agent before trying to get an agent.

Future Problems

Note: These problems may not even arise, but should an issue ever come up in the future. The following cases should be looked into.

- **Coordinate Comparison Rate with MatLab Code**
    - The Matlab code indicates that cells are moving at a different delta every time.
        - Currently, the cell's path is linear where it appears that it should not be.
    - Possible corrections can include…
        - Creating a mathematical expression that accurately describes a cell's movement or changing the way the cell's move by adding a degree variable so that the cell does not move linearly.
        - Moving a cell based on the distance between its' neighbors.
- **Rate of Division**
    - Some discrepancies between the Matlab code and the RepastHPC code have appeared between some cells dividing early and creating cells. One cell in particular seems to be created six files writes before it appears in the Matlab code.
    - Finding a better algorithm to using Boost's all_reduce() function to keep a better update on the flagging variable while still guaranteeing that every process will execute that function.
- **Speeding Up Simulation**
    - Using Boost/MPI directly would all increase the speed of the simulation rather than using some of RepastHPC's functions.
- **Rate of Saving Data**
    - The frequency of how often the processors write data to a file will need to be adjusted in order to improve efficiency with larger data sets.
- **Incorporating GUI System**
    - Since the model.props file will need to be the only thing changed, setting that information from the GUI system (as of Summer 2015: currently under development) will be the only requirement other than calling RepastHPC and the files needed to run the program that will need to be tackled by both the GUI and RepastHPC teams.