Diffusion Limited Aggregation

Simulating the Formation of Biological Crystal Growth in Flud Systems

Jarred Parr

October 2018

1 Introduction

The following writeup outlines the process an conclusions gathered from studying diffusion limited aggregation in biological systems. Its goal was to simulate the formation of crystals in any number of naturally occuring situations like gout, battery corrosion, and even snowflakes.

2 Program Architecture

All simulation code was abstracted from the main file in order to facilitate additional functionality as needed. Classes were also used to better group functionality. An initial attempt at writing this simulation led me to consider a class based approach of particle management. In it I had things like internal state management and point tracking, but that was in the idea of implementing a more fully featured and robust simulation. Due to time constraints, this unfortunately didn't get too far off of the ground.

I took two main considerations in my code design:

- Function
- Form

The code needed to work before significant refactoring could take place to remove issues like redundant code and to handle some of the more intricate features of the code. In the idea of function, I took as strongly of a functional programming approach to this problem as I could. This allowed me the flexibility to write my code in a more loosely coupled fashion while also being able to preserve the speed gains of using C++. The parallelism was

largely applied in the use of the main loops in the code. When used elsewhere it made for an extreme amount of issues and bottlenecks when accessing the values due to the neeed for crical and atomic pragma uses. That being said, I now firmly know that there is such a thing as too much parallelism.

3 Results

The code strongly exihibited the law of diminishing marginal returns. As shown below, as the number of cores increased, the linear speedup began to level off and saw only marginal gains as the number of cores increased. This has led me to hypothesize that had the number of parallel threads increased, the speed of exection would have continued to see smaller and smaller improvements from run-to-run.

- Sequential: 2496s
- 2 Core: 1325s
- 16 Core: 1719s