Accelerating Worm Segmentation through inter-node parallelism

Vineeth Remanan Pillai[+], Daniela Stan Raicu[*], Ioan Raicu[+]

[+]Department of Computer Science, Illinois Institute of Technology Chicago, IL, USA

[*]School of Computing, DePaul University, Chicago, IL, USA

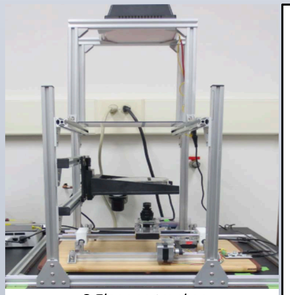
Abstract

- Nematode Caenorhabditis (C.) elegans is a roundworm which is one of the researcher's choice for neuronal development studies and molecular and developmental biology studies
- Locomotory behavior of C.Elegens is studied by recording the worm movement and then the recorded video data is processed to determine the path, speed, and trajectory of the worm, in order to identify higher level functions and behaviors.
- This work aimed to accelerate the processing of the C.Elegans tracking data through inter-node parallelism
- Achieved two to three orders of magnitude performance improvement when comparing to the original Java-based processing system

Motivation

- The C.Elegans is widely used for unraveling the principles underlying functional neural circuits
- C.Elegans was the first multicellular organism to have its whole genome sequenced
- The worms have a simple neural network with exactly 302 neurons and approximately 7000 synaptic connections
- The worm moves in a sinusoidal fashion and the locomotion relies on muscular contraction and nuero-muscular transmissions

C. elegans



C.Elegans tracker

The goal was to understand and study the locomotory behavior of the worm at various environments like a food deprived situation for example, for a long duration of time. The locomotory behavior has also been has been studied under various conditions like different mutations to the genetic structure of the worm. Knowledge gained from these studies could be applied to complex nervous systems like those of mammals.

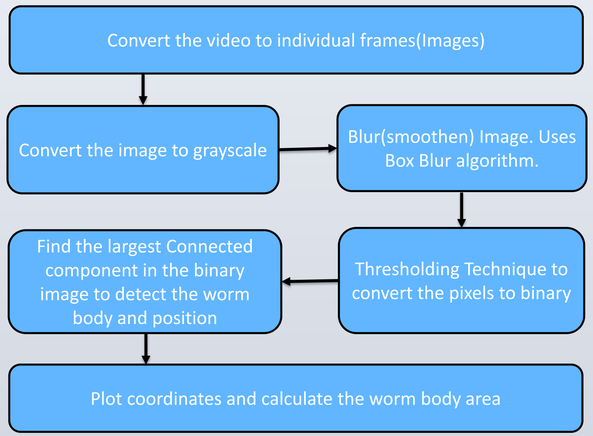
Experimental Setup

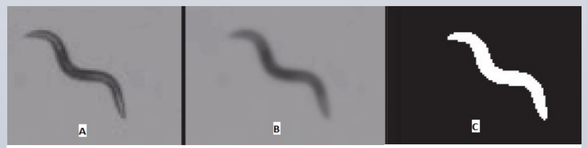
A setup consisting of both hardware and software was designed to study C.Elegans locomotory behavior. The setup was designed specifically to track one worm at a time.

- Hardware consisted of a base to hold the agar plate with the worm, a three axis movable camera mechanism and a small computer to control the camera.
- Tracking software is responsible for capturing the raw video (webcam capture API Sarxos) of the worm movement in the agar plate
- The software triggers camera movement to keep track and follow the worm in the agar plate

Analysis Design and Implementation

- Initial implementation in Java
- Ported Java code to different implementations in C and C++
  - C++ with OpenCV
  - C with PThreads
  - C with MPI
  - C with Swift/T





A: Grayscale image B: Smoothened image C: Binary Image

To track the worm location during its movement, centroid or center of mass of worm in each frame is calculated. Centroid is calculated as the average of x and y coordinated of all M pixels on the worm body:

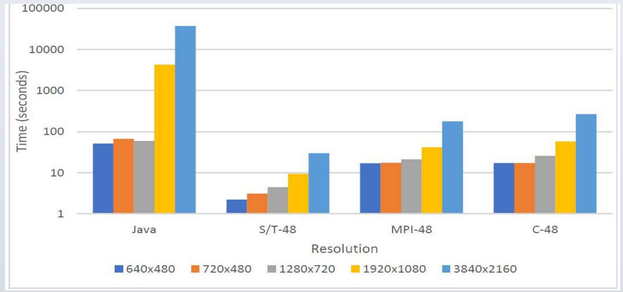
$$(C_x, C_y) = \left(\frac{\sum_{i=1}^{M} x_i}{M}, \frac{\sum_{i=1}^{M} y_i}{M}\right)$$

This work aims to improve the performance of worm segmentation by leveraging the power of multi-core processors and high-bandwidth low-latency interconnected nodes

- Implemented in C++ and OpenCV for ease of prototyping
- Implemented in C without any external library dependency for good support in HPC environments with light-weight OSes and poor support for third party libraries
- Extended the C implementation to use MPI and SWIFT/T to allow future evaluations of multi-node parallelism with the application spanning multiple inter-connected HPC nodes

Performance Evaluations

- Compared and Evaluated all implementation
  - S/T-48: Swift/T implementation with 48 threads.
  - MPI-48: MPI implementation running on a single system with 48 threads
  - C-48: C implementation with 48 threads
- Dataset consisted of 47000 images extracted at various resolutions
- Testbed : Chameleon cloud with 24-core Intel Haswell processors and 128GB RAM



Conclusion

- Java implementation has significant overhead compared to C/C++
- At higher resolutions, performance degradation in Java is even more pronounced (speedups of 2 to 3 orders of magnitude are common)
  - Low resolution dataset computation cost reduced from 51 seconds to 2.2 seconds
  - High resolution dataset computation cost reduced from over 10 hours down to under 30 seconds
- Swift/T and MPI performance are comparable to C with Pthreads; we will explore Swift/T and MPI in multi-node environments in future work as there are no code changes needed

References

[1] K. Moy, et al. "Computational Methods for Tracking, Quantitative Assessment, and Visualization of C. elegans Locomotory Behavior", PloS one 10, e0145870. 2015

[2] David R. Butenhof, "Programming with POSIX Threads", 1997

[3] William Gropp, et al. "A high-performance, portable implementation of the MPI message passing interface standard", 1996

[4] Michael Wilde, et al. "Extreme-scale scripting: Opportunities for large task-parallel applications on petascale computers", SciDAC 2009

[5] OpenCV (Open Source Computer Vision Library), https://opencv.org, 2018

[6] Stroustrup, Bjarne (1997). "1". The C++ Programming Language (Third ed.). ISBN 0-201-88954-4. OCLC 59193992.

[7] Gosling, James; McGilton, Henry (May 1996). "The Java Language Environment"

Acknowledgement

This work was supported in part by the NSF awards OCI-1054974 and NSF-1461260, as well as the Chameleon Testbed (NSF award 1743358 and 1419141). We also want to thank Kyle Moy for the original Java implementation codebase, and early exploratory work done by Adelina Voukadinova and Tejus Prasad.