In-Situ Visualization for 3D Agent-Based Vocal Fold Inflammation and Repair Simulation

***N. Seekhao*[1], *J. JaJa*[1], *L. Mongeau*[2], *N.Y.K. Li-Jessen*[2]**

A fast and insightful visualization is essential in modeling biological system behaviors and understanding underlying inter-cellular mechanisms. High fidelity models produce billions of data points per time step, making *in situ* visualization techniques extremely desirable as they mitigate I/O bottlenecks and provide computational steering capability. In this work, we present a novel high-performance scheme to couple *in situ* visualization with the simulation of the vocal fold inflammation and repair using little to no extra cost in execution time or computing resources. The visualization component is first optimized with an adaptive sampling scheme to accelerate the rendering process while maintaining the precision of the displayed visual results. Our software employs VirtualGL to perform visualization *in situ*. The scheme overlaps visualization and simulation, resulting in the optimal utilization of computing resources. This results in an *in situ* system biology simulation suite capable of remote simulation of 17 million biological cells and 1.2 billion chemical data points, remote visualization of the results, and delivery of visualized frames with aggregated statistics to remote clients in real-time.

Keywords: in situ, visualization, vocal fold, systems biology simulation, Agent Based Modeling, tissue inflammation and repair.

Introduction

Agent-based modeling (ABM) is a powerful and widely used approach to simulate a system consisting of interacting components or *agent*s. This form of modeling expresses a system at the microscale, and attempts to explain the emergence of higher order properties of the overall system [27]. As opposed to the analytical, equation-based approaches, the agent-based approach offers the ability to add complex behaviors to individual components or *agents*, making modeling of composite networks uncomplicated [8, 15]. In ABM, *agent*s are used to represent a wide spectrum of entities such as animals in ecosystems [13, 18, 23, 24], consumers and markets in economic models [3, 10, 34–37], and cells and proteins in biological systems [9, 11, 12, 17, 19–22, 29, 30, 32, 39]. These entities interact among themselves and with their environment (ABM *world*) in discrete time steps following a set of stochastic and/or deterministic rules. The simulation area or ABM *world* is discretized into 3D cubes called *patches*. In ABM, *Agent*s can be mobile and move from *patch* to *patch*. Each *patch* maintains its states, which affect the action decision of the residing and neighboring *agent*s.

In this paper, we use the ABM simulation approach to capture tissue injury and repair at the cellular level. More specifically, we focus on the vocal fold injuries. It is estimated that voice disorders afflict 1 in 13 adults [7], and nearly 1 in 12 children [1] in the the United State annually. During phonation, human vocal folds undergo continuous biomechanical stresses. Thus, voice overuse can lead to vocal fold mucosal tissue injury that triggers complex biological processes of inflammation and repair. Voice therapy treatments are usually prescribed to patients with voice problems [16, 25]. However, the healing outcomes of the treatment depends on the patient's initial condition and biological profile [21], making the treatment-planning process restraining and difficult for physicians and therapists. Personalized medicine is a promising candidate to

[1]University of Maryland, College Park, USA
[2]McGill University, Montreal, Québec, Canada

addressing this problem as it incorporates the patient's biological profile and initial conditions as parameters in determining an appropriate treatment. Thus, ABM offers a gateway to capturing inflammation and repair behavior to predict the outcomes of specific treatments for an individual.

The vocal fold (VF) ABM requires a high-resolution 3D grid to capture cell-cell and cell-substrate interactions with sufficient details in order to accurately predict the temporal tissue response to a provocation. Since the size of the *world* grid reflects the spatial resolution of the simulation, a high-resolution 3D VF ABM involves the generation and analysis of large amounts of data. Hence, a fast and low I/O load visualization is highly desirable, making *in situ* visualization an ideal candidate for understanding the model output.

In this work, we extend our previous work on a fast GPU implementation of VF ABM simulation to incorporate *in situ* visualization at no extra cost to the overall simulation. The proposed scheme combines a data reduction technique to gain optimal visualization performance with a CPU-GPU scheduling technique to overlap simulation and visualization while hiding the execution costs of the visualization component. More specifically, our main contributions can be stated as follows.

- Reducing the amount of data analyzed, while maintaining a high fidelity visual resolution using an adaptive sampling scheme.
- An *in situ* bio-simulation suite capable of processing 17 million biological cells and 1.2 billion chemical data points (a scale biologically representative of the human vocal fold at the cellular level) as well as collecting aggregated statistics, which:
  - Completely bypasses I/O.
  - Analyzes and renders results without causing an increase in the overall simulation time.

1. Background and Related Work

1.1. Agent-Based Modeling (ABM)

The basic components of ABMs are:

- ***Agents*** - Autonomous objects that perform actions and interact with other agents and the environment;
- ***Agent Rules*** - Behaviors of each type of agents;
- ***World*** - The environment in which all agents belong to.

Multiple types of agents can be modeled in a single ABM. Each type of agents behaves according to a set of predefined rules, which can be deterministic or stochastic. For example, a simulation related to tissue inflammation may have various biological cell types, such as neutrophils, macrophages and fibroblasts, as *agent*s. The predefined rules are determined using the best available knowledge in the literature of each component of the system. The autonomous agents are mobile and make decisions based on their states and their environment. *Patch* size is uniform across the world, and thus the resolution of the simulation environment is inversely proportional to the *patch* size. The temporal dimension of ABMs is discrete and the simulation progresses in a sequence of synchronous iterations (sometimes referred to as *ticks*).

1.2. Modeling Vocal Fold Inflammation and Repair with ABM

A Vocal Fold (VF) ABM simulating inflammation and repair was developed and partially verified against empirical data [21]. The biological cells were implemented as mobile ABM *agent*s. These *agent*s perform their functionality and make action decisions based on the states of their surrounding. Briefly, at the time of injury, the damaged mucosal tissue triggers platelet degranulation [19, 20]. Platelets secrete chemicals by modifying the states of their residing *patch*es. In the inflammatory phase, these chemical gradients stimulate vasodilation and attraction of inflammatory cells, namely, neutrophils and macrophages. These inflammatory cells then get activated once they reached the wound site. The activated cells then clean up cell debris and produce more chemicals to attract cells called fibroblasts, which are responsible for tissue structure maintenance. In the healing phase, fibroblasts, activated by tissue damage, synthesize and deposit extracellular matrix (ECM) proteins such as collagen, elastin, and hyaluronans to form building blocks in tissue repair [38]. These ECM proteins then form a scaffold for supporting fibroblasts and other cells migration and wound repair activities [5].

1.3. *In Situ* Agent-Based Modeling

In this section, we summarize the most related works. To the best of our knowledge, there has not been much *in situ* visualization work involving agent-based modeling (ABM). A quadtree-based ABM is proposed in [17] to reduce the amount of irrelevant data analyzed in-situ, where the work in [33] attempts to accomplish the same goal with a bitmap-based approach. There are tools available on Paraview [14], a popular visualization framework, which can be used for ABMs. Paraview Catalyst [4, 6] was developed to process simulation output data in-situ according to the user's co-processing script. An image-based approach built on top of Paraview Catalyst was presented in [2] to efficiently manage rendered images created in-situ by Paraview Catalyst. As much as all these works [2, 4, 6, 17, 33] reduce I/O loads, none completely by-passes I/O or can be used to achieve anything close to the desired performance for our problem.

2. A Novel Approach

2.1. Data Reduction Techniques

The processing of large amounts of data is inevitable in high-fidelity simulations. Expressive visualization enables the human eye to easily extract insightful information about the simulated system. Our model consists of approximately 17 million agents and produces 1.7 billion bio-marker data points in each iteration. The visualization component includes cell migration, chemical diffusion, and damage tracking. The most time consuming component is the chemical diffusion, which needs to access 154 million points of data during each iteration. To optimize the visualization of such large amounts of data, we employ sampling techniques and study their effects on the simulation visual output and corresponding performance enhancements.

2.1.1. Constant sampling

The first sampling approach is simply constant sampling. The world environment is divided into tiles of size $grid_x \times grid_y \times grid_z$, where the corresponding grid point represents the values of the tile centered at the point and within the radius of $grid_x/2$, $grid_y/2$ and $grid_z/2$ in the

x-, y-, and z-dimension, respectively. This naive approach was used to improve the visualization speed in our earlier work [31]. At the perspective of the entire simulated volume, the appearances of sampled simulation were indistinguishable for up to $(6 \times 6 \times 6)$-segment sampling. However, the output became pixelated when the view is magnified and the camera focuses on a smaller area. In particular, the wound area was not rendered with reasonable fidelity.

2.1.2. Adaptive sampling

Adaptive sampling is used to optimize the data access while enhancing the resolution of the visual output in important areas. This helps keep the execution time low, and yet the details in the output presented are not compromised.

For models aiming to capture injury undergoing inflammation and repair processes, the wound site is the most active area. Therefore, the highest importance index was assigned to the wound site volume. The margin around the wound site, and the rest of the tissue are then respectively assigned less and least importance The adaptive sampling pipeline diagram in fig. 1 illustrates data processing steps used to sample and send data to the visualization pipeline.

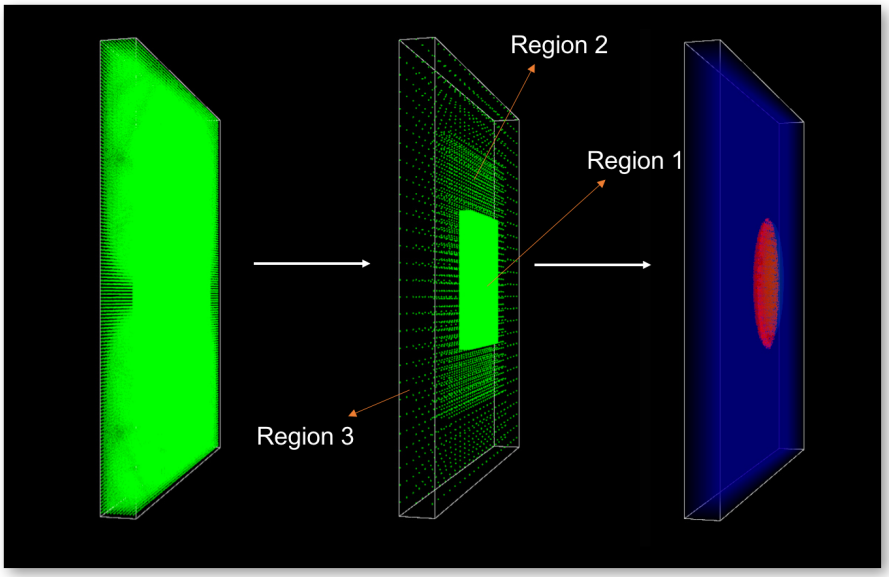

Figure 1. Adaptive sampling pipeline. From the whole set of data (left), the data are read with resolution conforming to the importance index (middle). The sampled data are then processed and sent to the visualization pipeline to be rendered (right).

2.2. Simulation and Visualization Scheduling

Given the amounts of data updated at each time step, our model requires a carefully planned scheduling mechanism for optimal performance. Accelerators such as GPUs need a CPU host, each of which has a number of cores that can be exploited using parallel programming techniques. However, whenever accelerated performance is the goal, the focus usually shifts towards GPUs due to their exceptional data parallel computation capabilities. Often, this results in idle CPU cores, since their only job is to transfer data and launch GPU tasks, while the GPUs perform all the computing work. We proposed a host-device computation overlap technique in our earlier work [30], which results in state-of-the-art performance for multi-scale 2D bio-simulation

ABMs. Since the 3D model is substantially more computationally demanding, we significantly extend the methods described there to achieve extremely high speed 3D simulation with *in situ* visualization.

Model operations are divided into sub-tasks and categorized as coarse or fine [30]. Since coarse-grain tasks (inflammatory cell and ECM function) are complex, but less data-intensive, they are deemed CPU-suitable, where simple data-intensive fine-grain (chemical diffusion) and rendering tasks run most effectively on GPUs. The blue boxes in fig. 2 illustrate the coarse-grain tasks that are executed in parallel on the CPU using OpenMP, where the green boxes denote the fine-grain tasks executed on the GPUs. The course-grain tasks execute on all CPU cores with the exception of $N_{GPU} + 1$ cores, where N_{GPU} denotes the number of GPUs. The rest of the N_{GPU} CPU cores are then used to dispatch fine-grain tasks and manage data to and from the GPUs, while the last core is spared to issue rendering calls to execute visualization on GPU. In [31], constant sampling was used, which was fast at the cost of lower resolution. Thus, placing the visualization execution in the GPU idle period was simple. However, in some circumstances, the user may want to focus on a smaller sections of the world, which means a sampling technique to enhance the resolution, while keeping the visualization execution time smaller than GPU idle period is required.

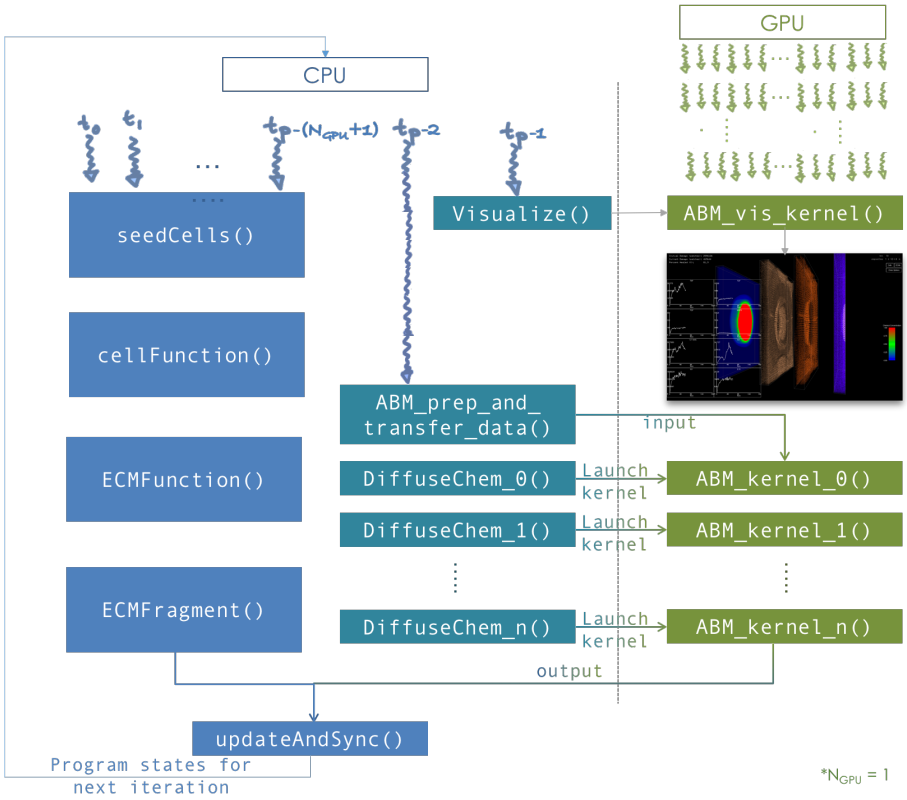

Figure 2. Diagram illustrating the scheduling and coordination of CPU-GPU computation and visualization overlap. For simplicity, this diagram is depicting a system with a single GPU. For multi-GPU, diffusion kernels launched are simply divided up and dispatched to multiple GPUs.

2.3. Remote *In Situ* Visualization

To fully utilize the powerful server, while allowing the user to steer the computation in real-time, a client-server *in situ* remote visualization [26] protocol was exploited using VirtualGL and

an X Proxy. The 3D ABM implementation was developed and tested on a system configured with VirtualGL and an X Proxy, TurboVNC. VirtualGL is an open source software which allows any Unix or Linux remote display software to run OpenGL applications by using the server's powerful 3D accelerator to perform rendering calls and send only rendered images to the client [28]. fig. 3 shows the X11 transport with an X proxy diagram. The application uses Xlib to communicate with the 3D X server to request for an openGL context. Once the context is created, the application can then interact with the rendering hardware directly via libGL. An X proxy, in our case TurboVNC, acts as a virtual X server. The X11 rendering is then performed to a virtual framebuffer in main memory rather than a real framebuffer on the graphics card. This allows the X proxy to compress and transmit the buffer content to end user without the need to provide any X server capabilities. Since the server is now responsible for most of the rendering, only the visualized simulation image frame (shown in fig. 4) is sent over network. Thus the remote user only needs to install a very light client package to see visualized result frames.

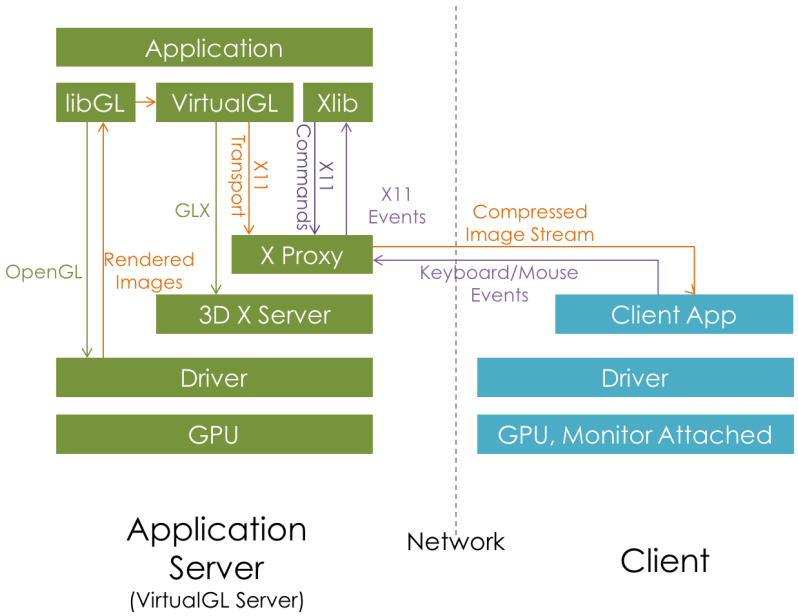

Figure 3. Diagram depicting different components and and work flow of in-situ visualization using VirtualGL and X Proxy (TurboVNC).

3. Results

The subsequent results discussed were benchmarked with 32 threads on a single compute node with 44-core Intel(R) Xeon(R) CPU E5-2699 v4 @ 2.20GHz host and two attached accelerators, NVIDIA Tesla M40. The size of the world grid is 110 x 1390 x 1006.

3.1. Visualization Component Performance

For the ABM simulation, the CPU tasks (excluding updates) take about 4.7 seconds while the GPU tasks only take 2.5 seconds for each iteration. Thus, there is an idle period on the

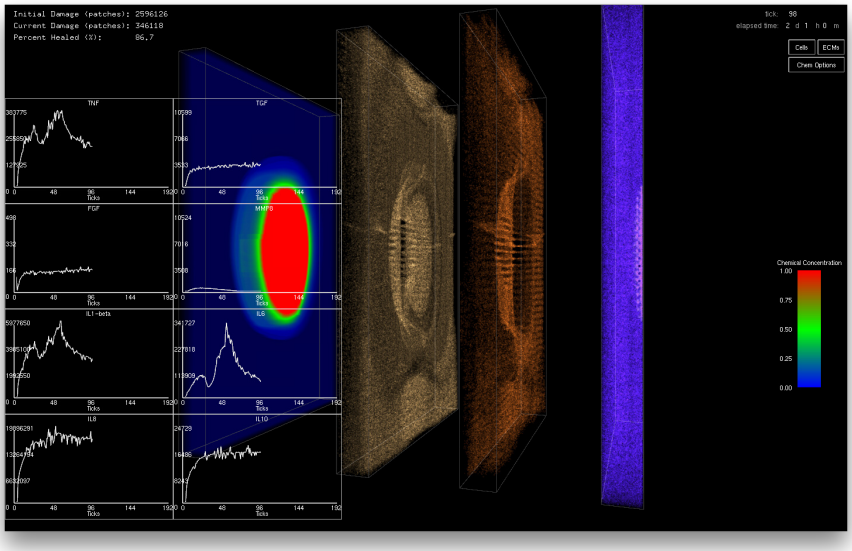

Figure 4. A screenshot of *in situ* visualization of the simulation captured at the client side. The 2D charts plot total concentration for each type of chemical. The left most 3D volume displays the distribution of one of the eight chemicals specified by the user. The second and third volumes show macrophage (brown) and neutrophil (red) distributions, respectively. The last volume on the right displays current damage (pink) and the distribution of fibroblasts (blue).

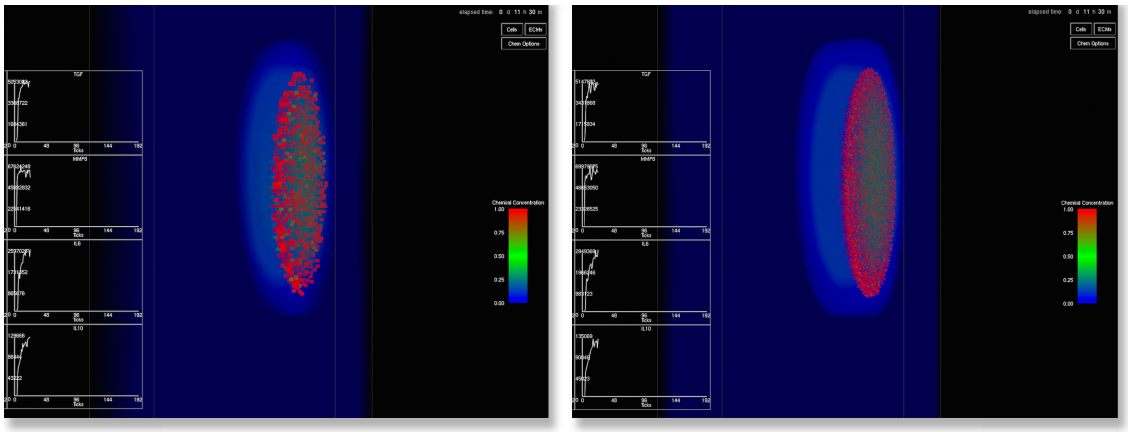

Figure 5. Screenshots comparison of 6^3 sampling windows (left) and $2-4-6$-sampling resolution (right) when zoomed in to high-activity area.

GPUs waiting for the CPU to finish the coarse-grain tasks. Our goal is to make the visualization component fast enough so that the 2.2-second window gap would allow us to integrate visualization with computation on the GPUs without increasing the total execution time. As discussed earlier, our work in [31] used constant sampling, which was fast enough but did not achieve good enough resolution when zooming in areas of interest such as the wound site. However, given that for the entire simulated volume, the appearances of sampled simulation were indistinguishable for up to $(6 \times 6 \times 6)$-segment sampling, the least important areas in adaptive sampling are set to use 6^3 sampling windows. The medium and most important areas are then sampled with 4^3 and 2^3 windows, respectively. With $2-4-6$-sampling resolution, as shown in fig. 5, the resolution of the visualization improved significantly, while the execution time of the visualization component only increased to 1.9 seconds, which is still below our visualization time budget.

3.2. Coupled Simulation and Visualization Performance

The performance of the simulation suite is shown in fig. 6. Without sampling, visualization took 23 second to complete. With adaptive sampling, visualization of chemical diffusion decreased to 1.9 seconds. The visualization execution was performed during the idle period on one of the GPUs, keeping the total execution time unchanged at 6.2 seconds per iteration on average. This results in the ability to run the simulation from the start of the iteration, remote computation, remote visualization to the moment the frame gets rendered on the client's machine in under 7 seconds/frame. To the best of our knowledge, this is by far the fastest known complex ABM simulation and visualization of a problem of that scale.

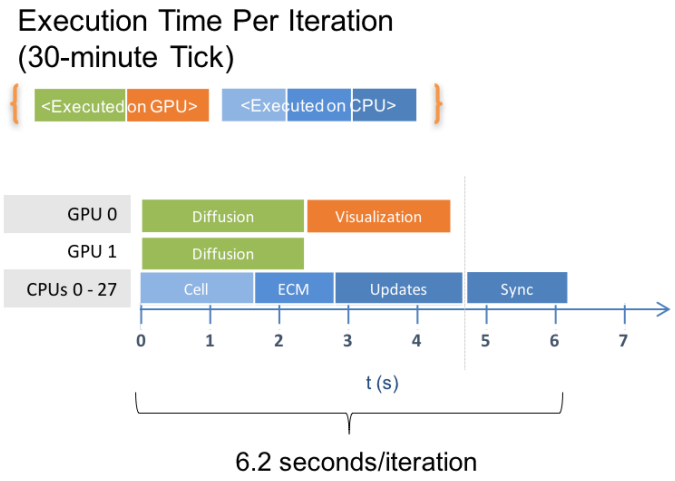

Figure 6. Simulation suite performance. Chart demonstrating overlapped visualization and computation executions on GPUs and CPUs.

4. Conclusion

We presented in this paper novel techniques to achieve *in situ* 3D ABM visualization at almost no cost to the overall simulation. As a result, we can simulate and render the VF inflammation and repair in real time. An effective task scheduling and management approach was used to orchestrate the execution of coarse-grain cellular functions, which are parallelized on the multi-core CPU, with the execution of fine-grain GPU tasks, including the overlapping with the in-situ visualization component. This results in optimal concurrent utilization of both multi-core CPU and GPU, including the fact that the execution time of the GPU visualization component is completely hidden behind the CPU tasks. Overall we are able to simulate 17 million inflammatory cells and 1.7 billion bio-marker data points, as well as analyze/render the same number of cells and 154 million bio-marker data points on the server and send result frames to the remote user in under 7 seconds per iteration.

5. Acknowledgment

The authors would like to thank Caroline Shung, Yun (Yvonna) Li and Alireza Najafi Yazdi for their contributions to the development of the initial and base sequential model. Sujal Bista for guidance in developing the visualization component. And UMIACS staff for assistance with VirtualGL and TurboVNC configuration. Research reported in this publication was supported by National Institute of Deafness and other Communication Disorder of the National Institutes of Health under award number R03DC012112 and R01DC005788.

The authors gratefully acknowledge the support provided by NSF, Contract Number CNS-1429404 MRI project.

References

1. Voice, speech, and language quick statistics. `http://www.nidcd.nih.gov/health/statistics/vsl/Pages/stats.aspx` (2012), accessed: 2012-10-3

2. Ahrens, J., Jourdain, S., O'Leary, P., Patchett, J., Rogers, D.H., Petersen, M.: An image-based approach to extreme scale in situ visualization and analysis. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. pp. 424–434. IEEE Press (2014)

3. Arthur, W.B.: Out-of-equilibrium economics and agent-based modeling. Handbook of computational economics 2, 1551–1564 (2006)

4. Ayachit, U., Bauer, A., Geveci, B., OLeary, P., Moreland, K., Fabian, N., Mauldin, J.: Paraview catalyst: Enabling in situ data analysis and visualization. In: Proceedings of the First Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization. pp. 25–29. ACM (2015)

5. Bainbridge, P., et al.: Wound healing and the role of fibroblasts (2013)

6. Bauer, A.C., Geveci, B., Schroeder, W.: The paraview catalyst users guide (2013)

7. Bhattacharyya, N.: The prevalence of voice problems among adults in the united states. The Laryngoscope 124(10), 2359–2362 (2014)

8. Bonabeau, E.: Agent-based modeling: Methods and techniques for simulating human systems. Proceedings of the National Academy of Sciences 99(suppl 3), 7280–7287 (2002)

9. Brown, B.N., Price, I.M., Toapanta, F.R., DeAlmeida, D.R., Wiley, C.A., Ross, T.M., Oury, T.D., Vodovotz, Y.: An agent-based model of inflammation and fibrosis following particulate exposure in the lung. Mathematical biosciences 231(2), 186–196 (2011)

10. Caiani, A., Russo, A., Palestrini, A., Gallegati, M.: Economics with Heterogeneous Interacting Agents: A Practical Guide to Agent-Based Modeling. Springer (2016)

11. Cilfone, N.A., Kirschner, D.E., Linderman, J.J.: Strategies for efficient numerical implementation of hybrid multi-scale agent-based models to describe biological systems. Cellular and Molecular Bioengineering 8(1), 119–136 (2014)

12. D'Souza, R.M., Lysenko, M., Marino, S., Kirschner, D.: Data-parallel algorithms for agent-based model simulation of tuberculosis on graphics processing units. In: Proceedings of the 2009 Spring Simulation Multiconference. p. 21. Society for Computer Simulation International (2009)

13. Gras, R., Devaurs, D., Wozniak, A., Aspinall, A.: An individual-based evolving predator-prey ecosystem simulation using a fuzzy cognitive map as the behavior model. Artificial life 15(4), 423–463 (2009)

14. Henderson, A., Ahrens, J., Law, C., et al.: The ParaView Guide. Kitware Clifton Park, NY (2004)

15. Janssen, M.A.: Agent-based modelling. Modelling in ecological economics pp. 155–172 (2005)

16. Johns, M.M.: Update on the etiology, diagnosis, and treatment of vocal fold nodules, polyps, and cysts. Current opinion in otolaryngology & head and neck surgery 11(6), 456–461 (2003)

17. Krekhov, A., Grüninger, J., Schlönvoigt, R., Krüger, J.: Towards in situ visualization of extreme-scale, agent-based, worldwide disease-spreading simulations. In: SIGGRAPH Asia 2015 Visualization in High Performance Computing. p. 7. ACM (2015)

18. Lampert, A., Hastings, A.: Stability and distribution of predator–prey systems: local and regional mechanisms and patterns. Ecology letters 19(3), 279–288 (2016)

19. Li, N.Y., Vodovotz, Y., Hebda, P.A., Abbott, K.V.: Biosimulation of inflammation and healing in surgically injured vocal folds. The Annals of otology, rhinology, and laryngology 119(6), 412 (2010)

20. Li, N.Y., Vodovotz, Y., Kim, K.H., Mi, Q., Hebda, P.A., Abbott, K.V.: Biosimulation of acute phonotrauma: an extended model. The Laryngoscope 121(11), 2418–2428 (2011)

21. Li, N., Verdolini, K., Clermont, G., Mi, Q., Rubinstein, E.N., Hebda, P.A., Vodovotz, Y.: A patient-specific in silico model of inflammation and healing tested in acute vocal fold injury. PloS one 3(7), e2789 (2008)

22. Martin, K.S., Blemker, S.S., Peirce, S.M.: Agent-based computational model investigates muscle-specific responses to disuse-induced atrophy. Journal of Applied Physiology 118(10), 1299–1309 (2015)

23. McLane, A.J., Semeniuk, C., McDermid, G.J., Marceau, D.J.: The role of agent-based models in wildlife ecology and management. Ecological Modelling 222(8), 1544–1556 (2011)

24. McLane, A.J., Semeniuk, C., McDermid, G.J., Tomback, D.F., Lorenz, T., Marceau, D.: Energetic behavioural-strategy prioritization of clarks nutcrackers in whitebark pine communities: An agent-based modeling approach. Ecological Modelling 354, 123–139 (2017)

25. Misono, S., Marmor, S., Roy, N., Mau, T., Cohen, S.M.: Multi-institutional study of voice disorders and voice therapy referral: report from the cheer network. Otolaryngology–Head and Neck Surgery 155(1), 33–41 (2016)

26. Nvidia, C.: Compute unified device architecture programming guide (2014)

27. Page, S.E.: Agent based models. The New Palgrave Dictionary of Economics. Palgrave MacMillan, New York (2005)

28. Project, T.V.: VirtualGL background. `http://www.virtualgl.org/About/Background` (2015)

29. Richmond, P., Walker, D., Coakley, S., Romano, D.: High performance cellular level agent-based simulation with flame for the gpu. Briefings in bioinformatics 11(3), 334–347 (2010)

30. Seekhao, N., Shung, C., JaJa, J., Mongeau, L., Li-Jessen, N.Y.: Real-time agent-based modeling simulation with in-situ visualization of complex biological systems a case study on vocal fold inflammation and healing. IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW) (2016)

31. Seekhao, N., Shung, C., JaJa, J., Mongeau, L., Li-Jessen, N.Y.: High-resolution 3d vocal fold repair simulation using highly-parallelized agent-based modeling. submitted to PloS one (2017)

32. Shi, Z., Chapes, S.K., Ben-Arieh, D., Wu, C.H.: An agent-based model of a hepatic inflammatory response to salmonella: A computational study under a large set of experimental data. PloS one 11(8), e0161131 (2016)

33. Su, Y., Wang, Y., Agrawal, G.: In-situ bitmaps generation and efficient data analysis based on bitmaps. In: Proceedings of the 24th International Symposium on High-Performance Parallel and Distributed Computing. pp. 61–72. ACM (2015)

34. Tesfatsion, L.: Agent-based computational economics: Growing economies from the bottom up. Artificial life 8(1), 55–82 (2002)

35. Tesfatsion, L.: Agent-based computational economics: modeling economies as complex adaptive systems. Information Sciences 149(4), 262–268 (2003)

36. Tesfatsion, L.: Agent-based computational economics: A constructive approach to economic theory. Handbook of computational economics 2, 831–880 (2006)

37. Tesfatsion, L., Judd, K.L.: Handbook of computational economics: agent-based computational economics, vol. 2. Elsevier (2006)

38. Velnar, T., Bailey, T., Smrkolj, V.: The wound healing process: an overview of the cellular and molecular mechanisms. Journal of International Medical Research 37(5), 1528–1542 (2009)

39. Wang, Z., Butner, J.D., Kerketta, R., Cristini, V., Deisboeck, T.S.: Simulating cancer growth with multiscale agent-based modeling. In: Seminars in cancer biology. vol. 30, pp. 70–78. Elsevier (2015)