Parallelization of a Three-Dimensional Full Multigrid Algorithm to Simulate Tumor Growth

Dylan Goodin[1], Chin F. Ng[1], Hermann B. Frieboes[1,2,3]

[1]Dept. of Bioengineering, University of Louisville; [2]Dept. of Pharmacology/Toxicology, University of Louisville; [1]James Graham Brown Cancer Center, University of Louisville;

Abstract

We present the performance gains of an openMP implementation of a fully adaptive nonlinear full multigrid (FMG) algorithm to simulate three-dimensional multispecies desmoplastic tumor growth on computer systems of varying processing capabilities. The FMG algorithm is applied to solve a recently published thermodynamic mixture model that uses a diffuse interface approach with fourth-order reaction-advection-diffusion PDEs (Cahn-Hilliard-type equations) that are coupled, nonlinear, and numerically stiff. The model includes multiple cell species and extracellular matrix (ECM), with adhesive and elastic energy contributions in chemical potential terms, as well as including blood and lymphatic vessels represented as continuous vasculatures. Advection-reaction-diffusion PDEs are employed for the cell-ECM components, whereas reaction-diffusion/advection-reaction-diffusion PDEs are used for the cell substrate and vessel species. This desmoplastic tumor model exhibits an extracellular matrix rich tumor microenvironment and may be beneficial when applied to studying fibrotic tumors such as pancreatic adenocarcinoma.

After adding openMP to the FMG code, the program was run for a single time step on a i7-4600U processor in single core and dual code configurations. Timing macros were used to determine how effective openMP improved the performance of the program from the single core to dual code execution. The model was then timed on a “FAT Node” of the Cardinal Research Cluster (CRC) for 1, 2, 4, 6, 8, 16, and 32 cores.

The resulting data indicate that, relative to a single core system, openMP applied to the FMG algorithm renders the initial time step 1.7 times faster on a dual core system and approximately 3 times faster on a quad core system. However, overhead between processing cores overtakes the benefits of using openMP on CPUs with more than 8 cores. Although using openMP demonstrates modest improvement in performance, this study indicates that further parallelization is required to achieve model performance that will yield practical benefit.

Background

A three-dimensional nonlinear tumor growth model composed of heterogeneous cell types in a multicomponent-multispecies system, including viable, dead, healthy host, and extra-cellular matrix (ECM) tissue species was recently presented (Ng & Frieboes, *J Theor Biol* 2017). The model includes the capability for abnormal ECM dynamics noted in tumor development, as exemplified by pancreatic ductal adenocarcinoma, including dense desmoplasia typically characterized by a significant increase of interstitial connective tissue. An elastic energy is implemented to provide elasticity to the connective tissue. Cancer-associated fibroblasts (myofibroblasts) are modeled as key contributors to this ECM remodeling. The tumor growth is driven by growth factors released by these stromal cells as well as by oxygen and glucose provided by blood vasculature which along with lymphatics are stimulated to proliferate in and around the tumor based on pro-angiogenic factors released by hypoxic tissue regions.

Background

Cellular metabolic processes are simulated, including respiration and glycolysis with lactate fermentation. The bicarbonate buffering system is included for cellular pH regulation. This model system may be of use to simulate the complex interactions between tumor and stromal cells as well as the associated ECM and vascular remodeling that typically characterize malignant cancers notorious for poor therapeutic response.

A fully adaptive nonlinear full multigrid (FMG) algorithm was implemented (Ng & Frieboes, in review) to simulate the 3D multispecies desmoplastic tumor growth. The algorithm was applied to solve the thermodynamic mixture model that uses a diffuse interface approach with Cahn-Hilliard-type fourth-order equations that are coupled, nonlinear, and numerically stiff. The model employs fourth-order nonlinear advection-reaction-diffusion PDEs for the cell-ECM components, as well as reaction-diffusion/advection-reaction-diffusion PDEs for the cell substrate and vessel species.

Overview of Work

This poster describes work to fully parallelize the algorithms used in order to speed up the computational burden of the modeling system. While the V-cycle of the FMG algorithm is inherently serial, many portions of the FMG algorithm execution, such as RB-Gauss-Seidel relaxation, contain repetitive looping; such locations are ideal for openMP implementation. After adding openMP to the FMG code, the program was run for a single time step on a i7-4600U processor in single core and dual code configurations. Timing macros were used to determine how effective openMP improved the performance of the program from the single core to dual code execution. The model was then timed on a “FAT Node” of the Cardinal Research Cluster (CRC) for 1, 2, 4, 6, 8, 16, and 32 cores.

Results

According to figure 1, 8 cores in parallel were over a minute faster than 32 cores. Even with 16 cores allocated, 8 cores remained the better option by 10 seconds, indicating there is no advantage to using 16 or 32 cores for openMP. Ideally, for well parallelized systems, the code would not maximize performance at 8 cores, but would continue to benefit from increased core count. The runtime data depicted on figure 2 indicates that functions with openMP ran 2.04 times faster when a dual core runtime was used vs. a single core runtime on an Intel i7-4600U. Furthermore, in multicore situations, functions that were improved by parallelization consume less of the overall time required to perform a single time step. This can be seen in figure 3, where 0.94% less time, on average, was spent on each of the four select parallelized functions. Functions that were not parallelized comprised a larger share of the overall runtime, indicating that serial portions of the code will prevent ideal scaling.

Results

The fact that decreased core count is favored was seen in the performance improvement when openMP was set to dynamic core allocation. For 16 and 32 cores, runtime data indicated the program performed better on dynamic systems, than non-dynamic systems. Dynamic systems decrease core count automatically to maximize parallel benefit, thus a decrease of parallel processors was both recorded and yielded improved performance for the time step oeverall. Finally, dynamic allocation of 8 cores was 0.340074 seconds slower than 8 cores static allocation. Since runtime data indicated 8 cores were used by the program throughout, the difference in performance is not significant.

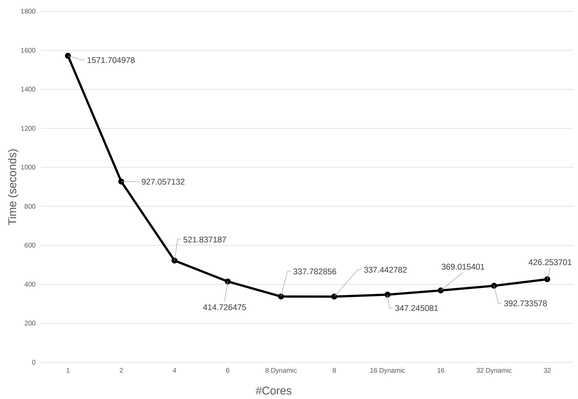

Figure 1 shows the time taken for the tumor code to process a single time step when running on 1, 2, 4, 6, 8, 16, and 32 cores.

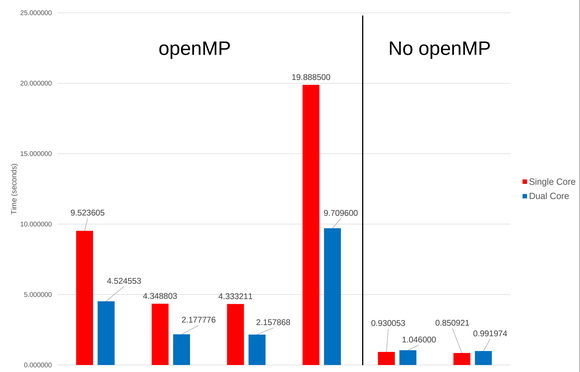

Figure 2 shows six functions in the tumor code timed in seconds on an Intel i7-4600U. The times given are the total amount of time spent on the function throughout a single iteration of the program. The left four functions were improved by openMP while the two functions on the right did not receive openMP improvements.

Results

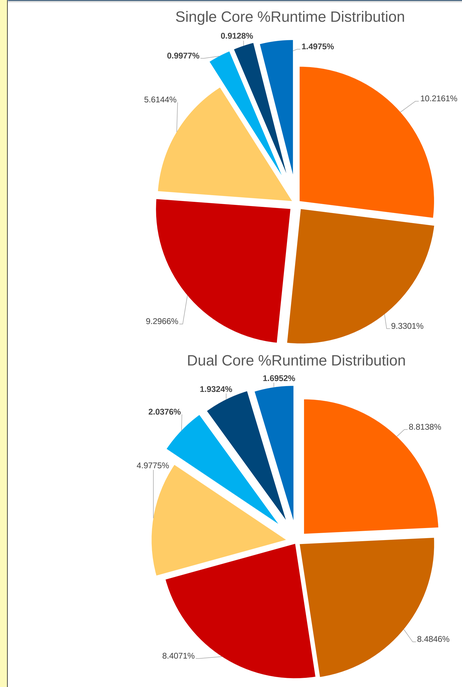

Figure 3 shows the distribution of the runtime across one time step, excluding some non-computational functions. Functions that were not parallelized are emphasized in blue.

Conclusions

It was concluded that openMP based implementations provide a substantial increase in performance. However, the overall serial nature of the current algorithm limits gains on performance. Given that the problem workload was the same across all models, there are two likely sources for the performance of the model. First, there was insufficient work to justify an increase of core count, leading to overhead. This was seen in the improved performance on 16 and 32 core options when dynamic nature was activated. Second, it is likely that memory conflicts between CPU cores were the major contributor to the decreased performance of the model. Since the amount of work increases at a cubic rate, openMP will not likely have the core count to achieve the scaling required.

Acknowledgements and References

Funding by U54CA143907, R01CA180149, and R15CA203605.

1. **CF Ng, HB Frieboes. Model of Vascular Desmoplastic Multispecies Tumor Growth. J Theor Biol 2017; 430:245-282.**
2. **CF Ng, HB Frieboes. A Fully Adaptive Nonlinear Full Multigrid Algorithm to Simulate Desmoplastic Multispecies Tumor Growth. 2017 (in review).**

This work was computed, in part, using the Cardinal Research Cluster (CRC) at the University of Louisville.