Clothes Simulation using Sequential and Parallel processing

CSCI8205/EE8367–Parallel Machine Organization
(Spring 2023)

By: Ace Kaung

Abstract

This paper presents a study on the performance analysis of cloth simulation using parallel programming and different decomposition methods. The goal of the study is to identify the best approach to achieve optimal performance on three different sizes of clothes with varying qualities. Three types of clothes were used in the study, including a small cloth with a dimension of 100x100, a medium cloth with a dimension of 200x200, and a large cloth with a dimension of 300x300. The decomposition methods used in the study included row, column, and block decomposition. The performance of each method was evaluated using different numbers of threads. The results of the study show that the row decomposition method consistently performs well across all three sizes of clothes, while the performance of the column and block decomposition methods varies depending on the size of the cloth and the number of threads used. Whereas the normal sequential method produces the lowest performance among all four methods. The findings suggest that parallel programming with appropriate decomposition methods and thread configurations can greatly improve the performance of cloth simulation.

Introduction

Cloth simulation is an important application in computer graphics and animation. The simulation involves modeling the physical properties of cloth, such as texture, flexibility, and smoothness, to create realistic animations. However, the simulation is computationally intensive and requires significant processing power. Parallel computing techniques, such as domain decomposition, can be used to optimize the performance of cloth simulation. This paper presents an analysis of the performance of cloth simulation using parallel computing techniques compared to the normal sequential technique.

Methodology

In order to simulate the behavior of cloth, it is necessary to have a thorough understanding of its underlying structure. Cloth is composed of interconnected nodes, as shown in Figure 1.0. Each node is linked to adjacent nodes in the up, down, previous, and next directions, as appropriate. Next, physics forces must be calculated for each individual node. These forces include the Hooke's Law force, aerodynamic force, Eulerian integration force, and collision detection force (*Tuur Stuyck, Cloth Simulation for Computer Graphics*).

The Hooke's Law and aerodynamic forces are calculated by obtaining the forces of neighboring nodes and storing them as acceleration values, without updating the nodes themselves until the Eulerian integration stage. During Eulerian integration, additional forces are calculated using the data obtained from the Hooke's Law and aerodynamic forces. Only after this stage are the node positions finally updated. Therefore, the Hooke's Law and aerodynamic forces can be parallelized together beforehand, while the Eulerian integration stage must be parallelized separately. Collision detection, on the other hand, can only be calculated after Eulerian integration has been performed. As such, it must be parallelized separately as well.

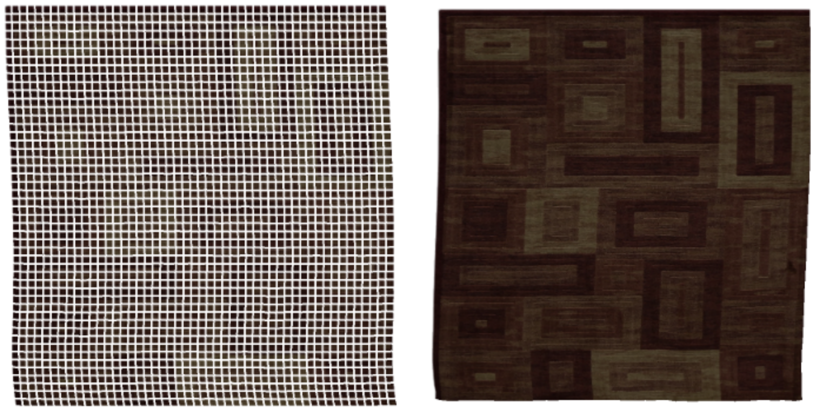
Figure 1.0 (Cloth Nodes)

Sequential

The sequential method for cloth simulation involves calculating the forces node by node, starting from the top left corner and proceeding towards the bottom right node. It calculates the first row's first column node and proceeds towards the last column of the first row until the end of the bottom row.

Parallel - Row

The parallel - row method for cloth simulation involves dividing the work into multiple threads. The first thread will perform the same computation as the sequential method, starting from the top left corner and proceeding towards the specified row. The second thread will start computing from where the first thread ends, and it will continue until the next specified row. Then the rest of the thread will follow the same logic until the end of the cloth.

Parallel - Col

The parallel - col (column) method for cloth simulation also divides the work into multiple threads. The first thread will start computing from the top left corner and proceed towards the specified column of the first row, then calculate the next row with the same logic until the last row. The other thread will start from where the previous thread ends and will continue computing towards the specified column of the last row.

Parallel - Block

The parallel - block method is a combination of the parallel - row and col methods for cloth simulation. It divides the work into multiple threads as well, similar to the previous methods. The first thread will compute the forces node by node, starting from the top left corner and proceeding towards the specified column of the first row, then calculate the next row until the specified column and will do this until the specified row. The other thread will start from where the previous thread ends and continue computing towards the specified row and column.

Samples

Three different types of clothes (figure 2.0) with varying sizes and qualities are used in this project. The first cloth is small, with a dimension of 100x100 and 2500 nodes to build the cloth. The medium cloth has a dimension of 200x200 and 6400 nodes, while the large cloth has a dimension of 300x300 and 10,000 nodes in it. In all these three clothes, the stiffness, airflow, and texture are all the same to make sure that it does not affect the results.

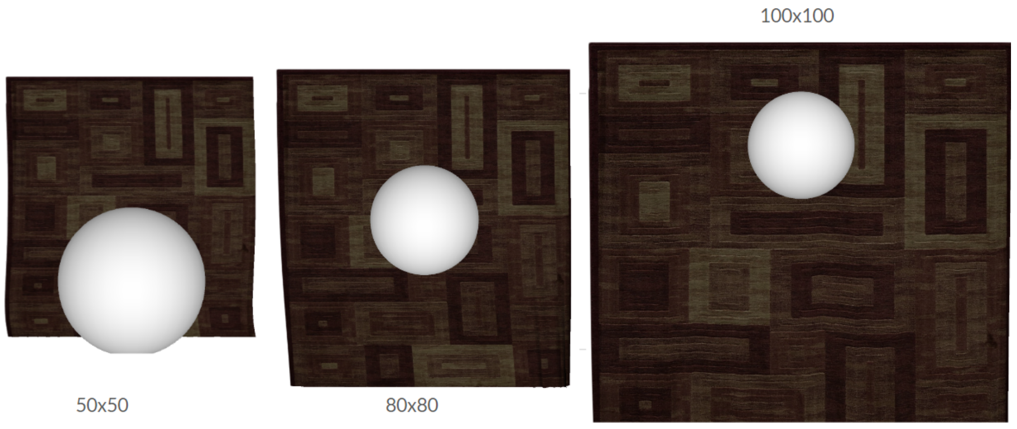

Figure 2.0 (3 different type of cloth samples
with sphere in front of it)

Thread Analysis

The performance of each method is measured using the FPS metric. To determine the best performance for each cloth size, we first find the optimal number of threads for each method. This has been done by measuring the FPS of each method in different types of clothes using different amounts of threads over a period of 10 seconds and taking the average of it to use as the performance, for this measurement we use it purely just the cloth without any collision to it. Table 3.0 shows the full details on the performance on different methods using different numbers of threads.

	Small			Medium			Large		
Threads	Row	Col	Block	Row	Col	Block	Row	Col	Block
2	92	91	86	43	40	39	28	28	26
3	99	100	104	49	47	45	33	31	28
5	109	102	111	55	52	46	36	36	30
10	109	105	97	56	54	45	38	36	29
15	99	102	91	55	53	45	35	34	38
20	89	87	74	48	46	44	38	31	38

Table 3.0 (Thread analysis on different methods
for different clothes measured in FPS)

From table 2.0, we can see that for the small cloth, row decomposition performs the best using 5 or 10 threads with the performance of 109 FPS, while column decomposition performs the best using 10 threads with the performance of 105, and block decomposition performs the best using 6 threads with the performance of 111 FPS. For the medium cloth, row decomposition performs the best using 10 threads with the performance of 56 FPS, column decomposition performs the best using 10 threads with the performance of 54 FPS, and block decomposition performs the best using 6 threads with the performance of 46 FPS. For the large cloth, row decomposition performs the best using 10 or 20 threads with the performance of 38 FPS, column decomposition performs the best using 5 or 10 threads with the performance of 36 FPS, and block decomposition performs the best using 15 or 20 threads with the performance of 38 FPS. These findings of the amount of threads will be used to measure the actual performance for each clothes along with the sequential method below.

Performance

Using the optimal number of threads that we found above for each method, we compare the performance of sequential computation with domain decomposition methods (table 4.0). The results show that sequential computation performs the worst for all three cloth sizes. For the small cloth, sequential computation performs at 84 FPS, while row decomposition performs at 110 FPS, column decomposition performs at 95 FPS, and block decomposition performs at 104 FPS. For the medium cloth, sequential computation performs at 33 FPS, while row decomposition performs at 57 FPS, column decomposition performs at 55 FPS, and block decomposition performs at 57 FPS. For the large cloth, sequential computation performs at 21 FPS, while row decomposition performs at 37 FPS, column decomposition performs at 34 FPS, and block decomposition performs at 37 FPS.

	Small				Medium				Large			
Seconds	Sequential	Row	Col	Block	Sequential	Row	Col	Block	Sequential	Row	Col	Block
1	79	102	98	106	34	59	56	58	22	39	38	41
2	83	109	106	109	33	56	54	55	21	38	37	40
3	82	109	107	108	33	56	54	57	21	37	36	38
4	85	110	107	105	33	57	53	57	21	37	36	38
5	85	110	94	100	33	56	54	57	21	37	36	38
6	85	110	88	105	33	57	55	57	21	37	36	36
7	85	112	88	104	33	57	54	57	21	37	31	34
8	85	111	87	99	33	57	55	57	21	37	31	34
9	86	112	85	101	33	57	55	57	21	37	31	34
10	85	112	88	101	33	57	55	57	21	37	31	34
Average	84	110	95	104	33	57	55	57	21	37	34	37

Table 4.0 (Performance results on different methods for different clothes)

Results

From the performance analysis that we have seen above, for the small cloth the row decomposition produced the best performance by using 10 threads and produced 110 FPS. For the medium cloth the row and block decomposition came to tie by producing the best performance with 57 FPS by using 10 threads for row and 6 threads for block. For the large cloth the row and block decomposition continue to place tie performance results by producing 37 FPS with 10 threads on row method while 16 threads on block decomposition. From these results, we found that the sequential performs the worst among all of the provided methods and to get better results we should be using the parallel domain decomposition.

Conclusion

In conclusion, this study has shown that parallel programming with appropriate decomposition methods and thread configurations can greatly improve the performance of cloth simulation. The performance analysis revealed that the row decomposition method consistently produced the best performance across all three sizes of clothes, while the column and block decomposition methods varied depending on the size of the cloth and the number of threads used. Additionally, the normal sequential method produced the lowest performance among all four methods. These findings suggest that parallel programming techniques can significantly enhance the performance of cloth simulation, leading to more realistic animations in computer graphics and animation.

References

Baraff, David, and Andrew Witkin. "Large Steps in Cloth Simulation." *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '98)*, 1998, pp. 43–54. https://doi.org/10.1145/280814.280821.

Bridson, Robert, Sebastian Marino, and Ronald Fedkiw. "Simulation of Clothing with Folds and Wrinkles." *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2003, pp. 28–36. https://doi.org/10.2312/SCA/SCA03/028-036.

Stuyck, Tuur. *Cloth Simulation for Computer Graphics*. Springer International Publishing, 2018.

Volino, Pascal, and Nadia Magnenat-Thalmann. "Implementing Fast Cloth Simulation with Collision Response." *Computer Graphics International, 2000. Proceedings*, IEEE, 2000, pp. 257–266. https://doi.org/10.1109/CGI.2000.852333.

NVIDIA Corporation. *CUDA C Programming Guide*. NVIDIA, 2013. https://docs.nvidia.com/cuda/cuda-c-programming-guide/.

OpenMP Architecture Review Board. *OpenMP Application Programming Interface Version 5.2*. OpenMP.org, 2023. https://www.openmp.org/wp-content/uploads/OpenMP-API-Specification-5-2.pdf.

Fang, Shaojie, and Ming C. Lin. "Parallel Simulation of Large-Scale Cloth." *ACM SIGGRAPH 2011 Talks*, 2011, p. 1. https://doi.org/10.1145/2018436.2018467.

Eberly, David. *Game Physics*. Morgan Kaufmann, 2003.

Ng, H. W., and R. L. Grimsdale. "Computer Graphics Techniques for Modeling Cloth." *IEEE Computer Graphics and Applications*, vol. 16, no. 5, 1996, pp. 28–41. https://doi.org/10.1109/38.536275.

Provot, Xavier. "Deformation Constraints in a Mass-Spring Model to Describe Rigid Cloth Behavior." *Proceedings of Graphics Interface*, 1995, pp. 147–154.

Lenoir, Jean, and Gilles Debunne. "Efficient Parallel Cloth Simulation on Modern Multicore Architectures." *Journal of Graphics, GPU, and Game Tools*, vol. 11, no. 1, 2006, pp. 51–64. https://doi.org/10.1080/2151237X.2006.10129253.

Choi, Kwang-Jin, and Hyeong-Seok Ko. "Stable but Responsive Cloth." *ACM Transactions on Graphics (TOG)*, vol. 24, no. 3, 2005, pp. 604–611. https://doi.org/10.1145/1073204.1073241.

Teschner, Matthias, et al. "Collision Detection for Deformable Objects." *Computer Graphics Forum*, vol. 24, no. 1, 2005, pp. 61–81. https://doi.org/10.1111/j.1467-8659.2005.00829.x.

Nguyen, Binh, Ronald Fedkiw, and Henrik Wann Jensen. "Physically Based Modeling and Animation of Fire." *ACM Transactions on Graphics (TOG)*, vol. 22, no. 3, 2003, pp. 721–728.

https://doi.org/10.1145/882262.882334.

Lin, Ming C., and Stefan Gottschalk. "Collision Detection Between Geometric Models: A Survey." *Proceedings of IMA Conference on Mathematics of Surfaces*, vol. 1, 1998, pp. 37–56.