 ICCES, vol.4, no.3, pp.151-157, 2007

PDSL and SDSL Parallel Visualization Algorithms for Large-scale Finite Element Analysis Data in Distributed Parallel Computing Environment

Jin Yeon Cho[1], Yun Hyuk Choi[2], You Me Song[3] and Chang Sik Kim[4]

Summary

In this work, PDSL(pre-detection sort last) and SDSL(strip-wise decomposition sort last) parallel visualization algorithms are proposed for efficient visualization of massive data generated from large-scale parallel finite element analysis through investigating the characteristics of distributed parallel finite element analysis procedure. The proposed parallel visualization algorithms are based on the sort last approach, and designed to be highly compatible with the characteristics of domain-wise computation in parallel finite element analysis. To investigate the performances of proposed algorithms, in-house software is developed by applying the binary tree network communication pattern along with the proposed sorting algorithms, and benchmarking test for parallel visualization is carried out.

Introduction

Currently, a considerable research effort has been given to the field of high performance computing (HPC). In HPC research, there are two major driving forces. One is the ever-increasing requirement for safety and preciseness in engineering design and analysis, and the other is the drastic reduction of computing cost caused by the splendid progress in computing hardware technologies such as parallel computing architectures. The representative nation-wide HPC research projects include US ASCI(Accelerated Strategic Computing Initiative), US ASC(Advanced Simulation and Computation), Japanese earth simulator project, Korean e-science project and so on. Through the projects, researchers are having solved the unprecedented problems which were never challenged before because of tremendous degrees of freedom.

As a result, extremely large sized data sets are inevitably produced from the high fidelity simulations, and the demands for highly efficient and affordable parallel visualization methodologies are being increased greatly in order to visualize the massive data [1, 2]. To meet the requirements for visualizing massive analysis

[1]Associate Professor, Department of Aerospace Engineering, INHA University, Incheon 402-751, Korea

[2]Research Assistant, Department of Aerospace Engineering, INHA University, Incheon 402-751, Korea

[3]Research Assistant, Department of Aerospace Engineering, INHA University, Currently Hyundai Motors

[4]Research Assistant, Department of Aerospace Engineering, INHA University, Currently Samsung Electronics

data, several high performance visualization tools have been developed. Among them, Chromium [3] is one of the most widely utilized general-purpose parallel visualization tools. However, it has some disadvantages in special purpose such as visualization of large sized data sets generated from distributed parallel finite element analysis, since it is designed for general purpose. Therefore this work aims to develop efficient parallel visualization framework and algorithms, which are highly compatible with distributed parallel finite element analysis procedure [4], with special consideration of domain-wise characteristics.

Parallel Rendering Framework

In distributed parallel finite element procedure, the whole domain of problem is decomposed into several sub-domains, and each decomposed sub-domain is allocated to each computer node in cluster system. After assigning the decomposed data, parallel finite element analysis is carried out through the network communication between computer nodes, and the final analysis results are stored in each computer node especially when the size of data is extremely large. Because the size of data is so large to handle in single conventional PC, traditional visualization system requires a high performance graphic server and expensive storage device such as RAID to store and retrieve the data. Further, one may redistribute the huge sized data into the visualization dedicated rendering cluster system. The situation is similar when the Chromium is utilized.

Here, one may easily notice that there is time-consuming duplicated work in this conventional visualization practice. It is collection and redistribution of huge sized data. Actually, if one changes the strategy in visualization of distributed parallel finite element analysis data, the time-consuming collection and redistribution of large data can be avoided because the data set resulted from distributed parallel finite element analysis procedure was already distributed domain-wisely in each computer node. Furthermore, one may avoid expensive hardware for visualization of massive data if one utilizes the computer nodes of cluster system for the purpose of data rendering as well as finite element computing. Based on the observation, we construct the following streamlined visualization strategy where parallel finite element analysis and parallel rendering are performed in the same computer nodes as shown in Fig. 1.

One can see that collection and redistribution of large data are avoided, and each computer node, which is utilized for finite element computing, is re-utilized for the purpose of rendering of each analysis data stored in each computer node. Through the strategy, one can reduce hardware cost as well as the duplicated cumbersome job.

Parallel Visualization Compatible with Parallel FE Analysis

In 1991, Molnar [5] classified parallel rendering (visualization) algorithms into

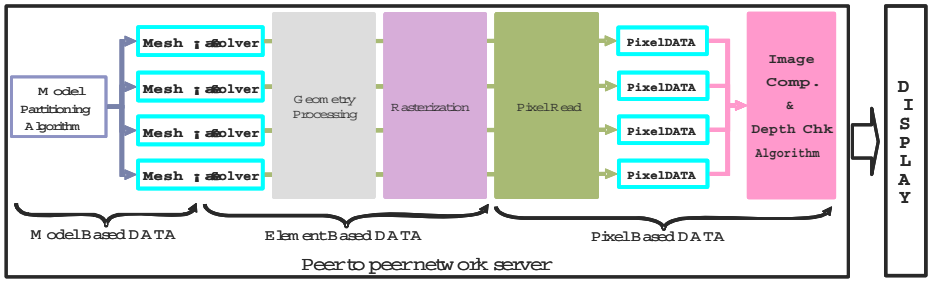

Figure 1: Parallel rendering system dedicated to distributed parallel finite element analysis

three categories based on the position of sorting procedure in the rendering pipeline. Those are sort-first (sort before geometry processing), sort-middle (sort between geometry processing and rasterization), and sort-last (sort after rasterization).

In sort-first approach, screen space is partitioned into non-overlapping two dimensional tiles or regions which will be rendered independently, and the geometric primitives (such as elements or meshes) are sorted for each screen space. From the reason, it is indispensable to collect the domain-wisely distributed analysis data and redistribute the massive data according to screen spaces. Therefore, the sort-first approach is not appropriate for visualization of distributed parallel finite element analysis data. In case of sort-middle approach, it is difficult to utilize in software level, because conventional commercial graphic accelerators do not provide API (application program interface) which makes it possible to access the graphic pipeline stage between geometry processing and rasterization.

In Sort-last approach, each computer (rendering) node makes a separated sub-image independently, and the local sub-images are combined into a single final image through depth comparison in order to make an entire image. Thus, the domain-wisely decomposed data stored in each computer node for distributed parallel finite element analysis procedure can be directly utilized without any collection and redistribution in this approach. Therefore, we can notice that the sort-last approach is compatible with domain-wise characteristics of distributed parallel finite element analysis.

Sort-last approach may be classified into sort-last-full and sort-last-sparse [6] according to the usage of frame buffer. Generally, in sort-last full approach, full frame pixel data set is utilized to make local sub-images, and the full frame pixel data set is transferred to other computer (rendering) node for image composition as shown in Fig. 2. Therefore the transfer of full frame pixel data may increase the network communication. (If one has display device with 1024×768 resolution, then full frame means 1024×768 pixel data.) To reduce the network communication, the

sort-last sparse approach utilizes only the bounding box pixel area containing the sub-image as shown in Fig. 2. After network communication between computer rendering nodes, image composition is performed through the depth comparison [7] for the entire region of transferred bounding box sub-image area.

PDSL and SDSL Algorithms

As noted before, in conventional sort-last sparse approach, depth comparison is carried out for the whole bounding box sub-image area which is transferred from the other computer nodes, even though the actual overlapped data set, for which depth test should be performed, is much smaller than the whole transferred bounding box sub-image area. Furthermore, since the sort-last sparse approach utilizes rectangular region of bounding box sub-image area, large amount of non-effective blank pixel data is still incorporated in the network communication.

Based on the aforementioned observation about the drawbacks of sort last sparse method, PDSL (pre-detection sort last) and SDSL (strip-wise decomposition sort last) algorithms are proposed in this work. PDSL (pre-detection sort last) algorithm is devised in order to reduce the depth comparison area, and SDSL (strip-wise sort last) algorithm is developed with the aim of reducing pixel data size for communication.

In PDSL algorithm, the bounding rectangular area for overlapped region is detected by simple communication of information about the vertices of bounding box for each sub-image before communicating valid pixel data as denoted in Fig. 2. By this pre-detection procedure for overlapped region, one can greatly reduce workload for depth comparison in image composition stage. In SDSL algorithm, the region of sub-image is decomposed into strip-wise rectangular pieces in order to approximate the shape of valid pixel, and the decomposed strip-wise rectangular pieces are utilized instead of the bounding box of sub-image area for data communication as shown in Fig. 2.

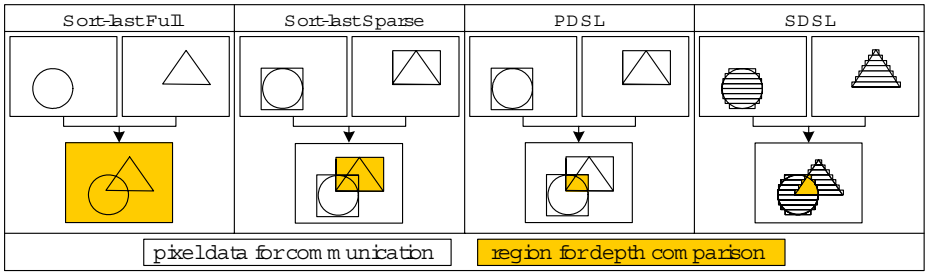

Figure 2: Pixel data for communication and depth comparison in sorting algorithms

Therefore, one can avoid the non-valid blank pixel data in network communication, and considerably minimize network burden which is one of the major rea-

sons for degradation of parallel visualization performance. Further, similar to the PDSL algorithm, the overlapped region is detected before depth comparison for each decomposed strip, and the depth test is carried out only for the pre-detected overlapped region. Therefore, one can also reduce workload for depth comparison in image compositing stage like the PDSL algorithm. The pre-detection time for SDSL algorithm is larger than that for PDSL algorithm because pre-detection is carried out for each decomposed strips in SDSL algorithm. However, it is noted that the increased time is negligible compared with the time saving resulted from the reduction of network communication in SDSL algorithm. Especially, it is noted that the SDSL algorithm is much more effective in case of complicated model where the portion of blank pixel in the bounding box of sub-image area is large.

Benchmarking Test

Benchmarking test is carried out to investigate the performances of the proposed PDSL and SDSL parallel visualization algorithms through the developed in-house software. In benchmarking test, self-made 16 node PC cluster system is utilized as a test bed. The cluster system is designed in order to utilize for parallel visualization as well as for parallel finite element analysis. The computer nodes of cluster system are interconnected by 3Com 1Gbit switching hub, and each computer node of cluster system is composed of Intel Pentium IV 2.8GHz CPU, Samsung 1Gbyte RAM, Intel 1Gbit network card, and GeForce FX5700 commercial graphic accelerator.

To observe the parallel visualization performances of the proposed PDSL and SDSL algorithms, parallel visualization benchmarking test is carried out for the finite element model composed of 1,329,027 tetrahedral elements as shown in Fig. 3, and those performances are compared with that of conventional sort-last sparse algorithm. The parallel visualization performance is measured by the time per revolution required to visualize the finite element model in rotating motion. For data communication between the computer rendering nodes, binary tree communication pattern [8] is utilized along with sorting algorithms in order to precisely measure the performances of each sorting algorithms concerning the depth comparison and communication overhead. It is noted that one can reduce the image composition time by the binary-tree communication pattern, since the network communication time of binary-tree structure is proportional to the number of computer rendering nodes logarithmically. In Fig. 3, one can observe that both of SDSL and PDSL algorithms are more efficient than sort last sparse algorithm. Especially, it is noted that the parallel visualization performance is increased around 13% in case of SDSL algorithm, compared with sort last sparse algorithm.

Conclusions

In this work, PDSL(pre-detection sort last) and SDSL(strip-wise decomposi-

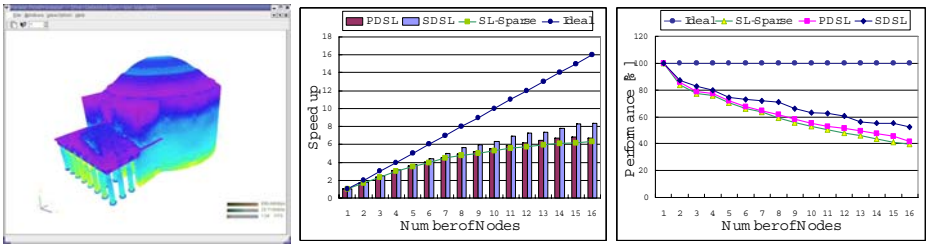

Figure 3: Comparison of parallel visualization performances of sorting algorithms

tion sort last) parallel visualization algorithms are proposed for efficient parallel visualization of massive data generated from large-scale parallel finite element analysis through investigating the characteristics of distributed parallel finite element analysis procedure. From the benchmarking test, it is known that both of PDSL and SDSL algorithms have better performances compared with the sort last sparse algorithm. Especially, the SDSL algorithm yields 13% improvement in parallel performance compared with the sort last sparse algorithm. From the result, it is expected that the proposed visualization algorithms can be efficiently utilized in high performance visualization of massive data generated from the distributed parallel finite element analysis in distributed parallel computing environment with affordable cost.

Acknowledgement

Authors would like to acknowledge the financial support from Agency for Defense Development through VT-41 project of Flight Vehicle Research Center.

References

1. Kenneth Moreland, Brian Wylie, Constantine Pavlakos, Vasily Lewis, (2001): "Scalable rendering on PC clusters", *IEEE Computer Graphics and Applications*, Vol. 21(4), pp. 85-154.

2. Kenneth Moreland, Brian Wylie, Constantine Pavlakos, (2001): "Sort-Last Parallel Rendering for Viewing Extremely Large Data Sets on Tile Displays", *IEEE Symposium on parallel and large data visualization & graphics*, pp. 85-154.

3. Greg Humphreys, Mike Houston, Ren Ng, Sean Arhen, Randall Frank, Peter Kirchner, James T Klosowski, (2002): "Chromium : A Stream processing framework for interactive on clusters", *Proceedings of SIGGRAPH 2002*, Vol. 21(3), pp. 693-702.

4. Kim S.J., Lee C.S., Kim J.H., Joh M.S., Lee S.S., (2003): "IPSAP : A High-performance Parallel Element Code for Large-scale Structural Analysis Based on Domain-wise Multifrontal Technique", *Proceedings of the*

ACM/IEEE SC2003 Conference, pp. 32.

5. Steve Molnar, Michael Cox, David Ellsworth, Henry Fuchs, (1991): "A Sorting classification of parallel rendering", *IEEE Computer Graphics and Applications*, Vol. 14(4), pp. 23-32.
6. Don-Lin Yang, Jen-Chih Yu, Yeh-Ching Chung, (2001): "A Sorting classification of parallel rendering", *The Journal of supercomputing*, Vol. 18(2), pp. 201-220.
7. Foley J., Dam A., Feiner S., Huges J., (1997): "Computer Graphics : Principles and practice", *ADDISON WESLEY*, Second Edition, pp. 668-672.
8. Steve Molnar, (1991): "Image Composition Architectures for Real-time Image Generation", *Univ. of North Carolina at Chapel Hill , Technical Report TR91-04*.