The Design and Implement of Ultra-scale Data Parallel In-situ Visualization System

Liu Ning
Institute of Computing Technology
liuning01@ict.ac.cn

Zhang Yingping
Institute of Computing Technology
zhangyingping@ict.ac.cn

Gao Guoxian
Institute of Computing Technology
gaoguoxian@ict.ac.cn

Zhu Dengming
Institute of Computing Technology
mdzhu@ict.ac.cn

Abstract

nomena at a stunning degree of precision and sophistication. But it is more and more impractical to store the data to disk in real time because of the scale of the output data, and thus the conventional post-processing based technique no longer applies. In response to this challenge,we design and implement an in-situ visualization system based on MPI and CUDA. In this system, the simulation code and visualization code are coupled together and the visualization is performed locally as soon as the data is generated using a GPU visualization pipeline. In addition, we adopt a display wall to show the rendered image, which is capable of showing very fine details of the simulation output.

1. Introduction

Now the power of parallel computing system is growing dramatically, as a result, it is more and more impractical to store the output of scientific simulation to hard disk for the lack of sufficient disk space and network bandwidth. The conventional post-processing based visualization method usually requires the output to be dumped to the hard disk before it goes through further processing, so it is not appropriate to be adopted to visualize ultra-scale data generated by simulation program in real time. In this paper, we design and implement an ultra-scale data in-situ visualization system to tackle these obstacles.

Our system can fully exploit the parallel computing power of the cluster or super computers. The MPI (Message Passing Interface) is a commonly used standard to write parallel programs running on distributed memory systems. The nodes of these systems can communicate with each other according to

GPU cluster enable people to model and visualize scientific phe

the MPI interface. This is a relatively coarser grain of parallelism. In contrast with MPI, CUDA (Compute Unified Device Architecture) accomplishes parallelism using modern GPU. There may exist hundreds of cores in one GPU, and each core can serve as a powerful computing unit. If used properly, the CUDA program can achieve one or two orders of magnitude compared to traditional CPU computing. This is a finer grain of parallelism.

2. Related works

Molnar *et al.*[1] divide parallel rendering into three categories based on the mapping process from object space to screen space, that is, sort-first, sort-middle and sort-last. Sort-first is suitable for small data while sort-last scales well for large-scale data. If the data size is too large, sort-last is the only feasible option.

Ma[2] introduced Binary-Swap algorithm and it has been adopted in many applications because of its simplicity and good performance. But it restricts the number of nodes to be power of two, while in most cases the condition does not hold. Yu[3] extends this algorithm such that this restriction is released. Moreland *et al.*[4] proposed an image composition algorithm based on Binary-Swap that can support display wall.

Ma[5] made a good outline of ultra-scale data visualization, including load balancing, in-situ visualization and intellectual feature tracking. When the scale of the data reaches a certain threshold, the conventional post-processing based method no longer applies and the in-situ visualization becomes a promising approach. Ma[6] gave some key points of in-situ visualization.

3. System Design

The goal of the system is to visualize the simulation-generated data in real time and offer the users the ability to interact with the simulation and visualization process.

The system consists of a master node, a simulation & visualization cluster and a display wall. The nodes within the cluster cooperate with each other through the MPI interface and the master node uses the socket to communicate with the cluster.

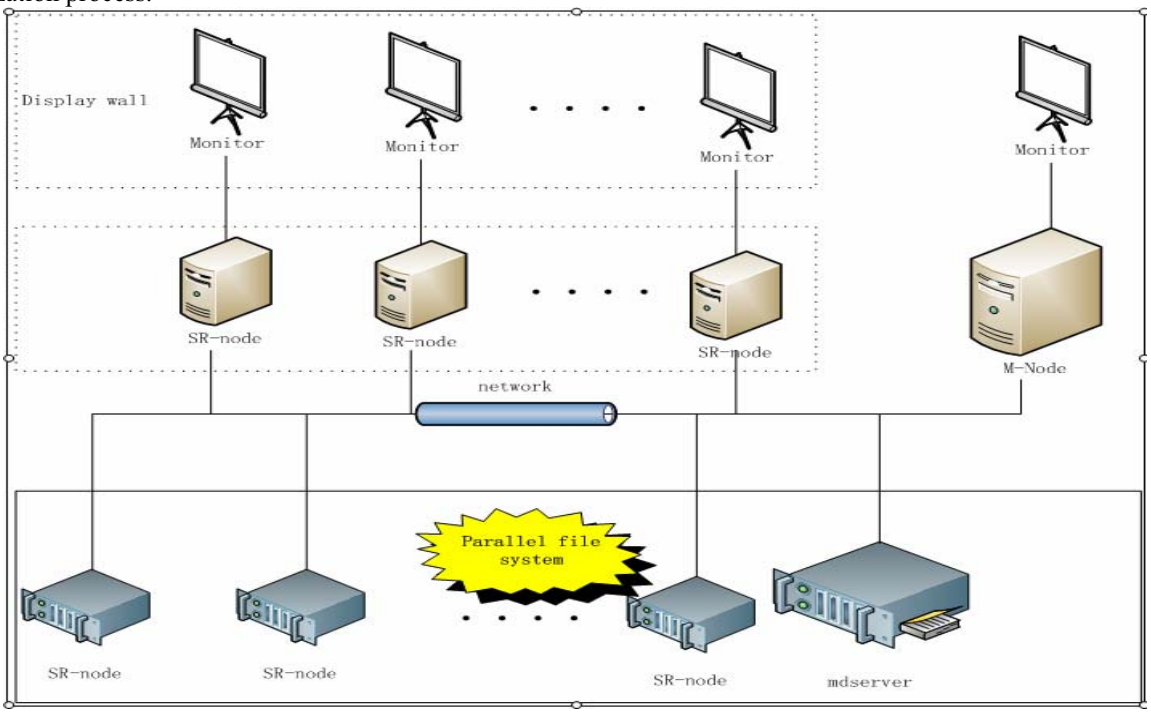

Fig. 1 Overall design of the system

3.1 Master Node (M-Node)

Users interact with the system through a QT GUI running on the master node. It is equipped with a monitor that is responsible to show the thumbnail delivered back from the cluster. Users request some certain services and the cluster gives back the requested result.

3.2 Simulation and Rendering Node (SR Node)

The simulation and rendering node performs scientific simulation and produces some elementary image results that are composed afterwards to get the whole image. The elementary results are also down-sampled and delivered to the master node such that the master node can fetch a thumbnail from them. Part of the SR nodes are also responsible to drive the display wall and thus require DVI port to be installed. All the SR nodes need to support OpenGL in order to perform visualization.

4. Parallel Visualization

Parallel visualization can be divided into four stages, that is, data distribution, local rendering, image composition and image output. We mainly focus on two problems: one is load balancing and the other is to minimize the communication delay within the cluster.

4.1 Data Distribution

In the in-situ visualization mode, the scientific simulation usually adopts uniform data partition technique, as a result, the data distribution is suitable for visualization. We just follow the simulation data partition way for simplicity. If the data partition of the simulation does not adapt to the visualization calculation, the data must be re-distributed.

4.2 Local Rendering

We use VTK(The Visualization Toolkit) to do local rendering. Because the simulation generated data reside

in the GPU memory, but the conventional VTK visualization pipeline can only handle the data that are in the memory, so we design a new CUDA based pipeline and integrate it into the existing VTK framework. The source of the pipeline is simulation data and it goes through Filter and Mapper stage and finally is visualized by the Render. Fig.3 is the GPU visualization pipeline.

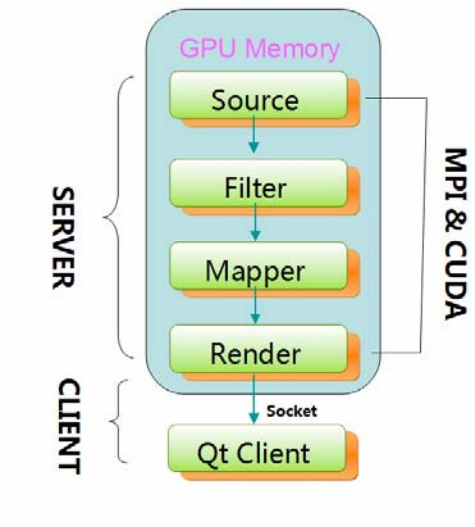

Fig. 2 The GPU visualization pipeline

4.3 Image Composition

Binary-Swap is a commonly used and has been proven to be the best image composition strategy if the number of nodes is power of two. Binary-Swap consists of logN stages and in each stage the nodes are divided into several pairs. A node only communicates with its pair in each stage. Finally, each node holds 1/N of the whole image and an image collection is performed to get the whole image.

4.4 Image Output

The composited image is displayed during the image output stage. Because of the scale of the data, if we would like to see the details, we need the resolution of the image to be large high enough. Thus we use a display wall to show the rendered result.

We utilize Moreland's algorithm to accomplish our goal. The algorithm is executed as follows: The nodes are divided into several groups according to the number of the tiles in the display wall, then each group is responsible for its corresponding tile. At first, the groups exchange image data with each other such that in each group the nodes only hold the data that will be displayed in its corresponding tile. After that, Binary-Swap is performed within each group. In each group, there exists a drawing node that collects the composited image in that group and eventually shows the image on the display wall.

5. In-situ Visualization

Conventional visualization technique is commonly based on post-processing model, that is , the simulation program stores the output data to file system and then some other dedicated visualization machines will fetch the data and do corresponding processing. The co-processing model steps a little forward. In this model, the data is transfered directly to the visualization machines after it is generated. If the visualization and the simulation calculations are done on the same machine, then we get the in-situ model.

In-situ visualization can greatly reduce the demand for disk I/O and network bandwidth. Furthermore, because the result of in-situ visualization is continuous and so we can observe some information that is hard or impossible to get from post-processing visualization.

In our system, the simulation code acts as a module of the visualization program and it uses the MPI communicator of the visualization side. The simulation side sends the generated data to the visualization side through a proxy. The design is quite flexible and all we need to do is to design a proxy and fill the data structure of the visualization side with the generated data from the simulation side.

6. Results

We carried out an experiment on a GPU cluster consisting of 32 nodes, and the nodes use InfiniBand to communicate with each other. Each node is equipped with four GeForce GTX 295 video cards, one Xeon 5430 2.66GHz CPU, one 1.5TB hard disk. The final frame rate is about 12fps and in each frame about 30MB data are generated in each node, so the overall data that the system needs to process in one second is around 10GB. We showed the results in the 50^{th}, 100^{th}, 200^{th} and 300^{th} frame.

Fig. 3 Experiment results

7. Conclusion

We design and implement an ultra-scale in-situ visualization system in this paper. Our work mainly focuses on the design of in-situ visualization and the GPU visualization pipeline. The in-situ visualization technique can process the data as soon as they are generated and greatly reduce the demand for disk space and network bandwidth. The GPU visualization pipeline is capable of handling the data residing in the GPU memory, without the need of transferring the data between GPU memory and main memory.

We just implement some basic visualization techniques, such as contour and volume rendering, so our future work will stress on the functions of system.

Acknowledgements

This work is supported and funded by National Natural Science Foundation of China Grant No. 60703019, National Key Technology R&D Program 2008BAI50B07, and NSFC-Guangdong Joint Fund (U0935003). National 863 Foundation of China Grant No. 2009AA01Z312.

References

[1] Steven Molnar, Michael Cox, David Ellsworth, Henry Fuchs, "A Sorting Classification of Parallel Rendering", IEEE Computer Graphics and Applications , 1994

[2] Kwan Liu Ma, "Parallel Volume Rendering Using Binary Swap Image Composition", IEEE Computer Graphics and Applications , 1994

[3] H Yu, C Wang, KL Ma, "Massively Parallel Volume Rendering Using 2-3 Swap Image Compositing", ACM SIGGRAPH ASIA, 2008

[4] Kenneth Moreland, Brian Wylie, Constantine Pavlakos" Sort-Last Parallel Rendering for Viewing Extremely Large Data Sets on Tile Displays "Proceedings of the IEEE symposium on parallel and large-data visualization and graphics , 2001

[5] Kwan-Liu Ma, "Ultra-Scale Visualization", Third International Symposium on ransdisciplinary Fluid Integration , 2006

[6] Kwan-Liu Ma, Chaoli Wang, Hongfeng Yu, Anna Tikhonova, "In-Situ Processing and Visualization for Ultrascale Simulations", Journal of Physics, 2007

[7] Hongfeng Yu, Chaoli Wang, Ray W. Grout, "A Study of In-Situ Visualization for Petascale Combustion Simulations", tech. report CSE-2009-9, Dept. of Computer Science, Univ. California, Davis，2009

[8] Christiaan P. Gribble, Abraham J. Stephens, James E. Guilkey, Steven G. Parker, "Visualizing Particle-Based Simulation Datasets on the Desktop", British HCI 2006 Workshop on Combining Visualization and Interaction to Facilitate Scientific Exploration and Discovery , 2006

[9] J Meredith, KL Ma, "Multiresolution View-Dependent Splat Based Volume Rendering of Large Irregular Data, IEEE Symposium on Parallel and Large-Data Visualization and Graphics", 2001

[10] Kenneth Moreland, Lisa Avila, and Lee Ann Fisk, "Parallel Unstructured Volume Rendering in ParaView" In Visualization and Data Analysis 2007， Proceedings of SPIE-IS&T Electronic Imaging, 2007