Multi-frame Prediction Load Balancing Algorithm for Sort-first Parallel Rendering

Tianhan Gao
Software college
Northeastern University
Shenyang,China
86-13664118426
Gaoth@mail.neu.edu.cn

Xiaoyu Luo
Software college
Northeastern University
Shenyang,China
86-15524507796
1004100224@qq.com

Nan Guo
College of information science and engineering
Northeastern University
Shenyang,China
86-13664118426
Guonan@ise.neu.edu.cn

ABSTRACT

In order to solve the problem of the load unbalance in sort-first parallel rendering system, this paper proposes a multi-frame prediction algorithm derived from CSLB algorithm to improve performence in parallel rendering, which introduces multi-frame feedback prediction to achieve better practical results. The experiment is carried out on the Equalizer parallel rendering framework and the results show that the proposed algorithm can improve the system frame rate well compared with the CSLB algorithm in terms of sort-first parallel rendering.

Keywords

sort first parallel rendering; load balancing; CSLB; multi-frame prediction

1. INTRODUCTION

With the rapid development of science and technology, computer graphics has been deep into the realistic graphics, scientific computing visualization, virtual environment, multimedia technology, computer animation, computer aided engineering drawings, etc[1-3]. With the scene getting more and more complex, sort-first parallel rendering has become an important research direction due to the fact that GPU-based rendering technology is not only expensive but also poorly scalable.

Load balancing is a fundamantal issue for the efficiency of sort-first parallel rendering. At present, the load balancing algorithm for sort-first parallel rendering system is divided into two types: geometric data and space-time conversion[4-5]. The former takes the geometric primitive as the measure of the load. The algorithm divides the load more accurately. But suffers from the high computational complexity that may affect system rendering efficiency. The latter discards the large number of geometric data by adapting the time value as a load indicator to assign the task to save computing overhead. The CSLB (Cross-Segment Load Balancing) algorithm is a kind of space-time conversion load balancing algorithm, which only predicts the load of the next frame based on the drawing of the previous frame. This method is too simple and unstable when the load changes suddenly. Taking CSLB as a building block, this paper proposes an improved multi-frame prediction algorithm equiped with some classical mathematical models to promote the performence of sort-first parallel rendering system.

CGDIP '17, July 02-04, 2017, Prague, Czech Republic.
Copyright 2017 ACM 978-1-4503-5236-9/17/07…$15.00.
DOI: http://dx.doi.org/10.1145/3110224.3110240

2. RELATED WORK

2.1 Sort-First Parallel Rendering

In order to improve the efficiency of rendering system, the parallel system decompose the data or the function to give full play to the computing power of multiple nodes which is known as the data parallel and time parallel. Sort-first parallel rendering system complete the task distribution in the image space which belongs to the data parallel system. As shown in Figure 1, the primitive attribution judgment occurs prior to the geometrical pipeline when the geometric data is determined on the projection area of the display screen. The image space of the screen is divided into multiple non-overlapping sub-areas, every sub-area corresponds to a rendering pipeline that completes the rendering task by the corresponding rendering nodes. Finally all the sub-area image outputted by their rendering nodes are merged to the final display image.

Advantages of sort-first parallel rendering system:

1) Each pipeline is complete and independent to build a PC cluster parallel graphics rendering system;

2) low network traffic;

3) The synthesis method is simple;

4) good scalability.

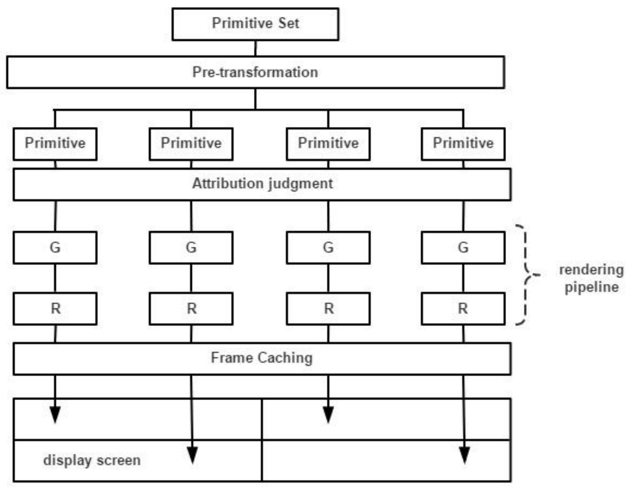

Figure 1. Sort-first parallel rendering system

2.2 Load Balancing[1]

In parallel rendering system, fine load balancing is the key to improve system performance. Load balancing issue in sort-first parallel rendering system is particularly prominent. Load balancing means that each drawing node is assigned a reasonable calculation task so that they can complete the rendering task equably. The efficiency of whole rendering system is thus improved.

2.3 CSLB Algorithm

The CSLB algorithm is a dynamic load balancing algorithm[6] embedded in the open source parallel rendering framework Equalizer[7]. The algorithm calculates the resource footprint of each task at the beginning of each task then reallocates the available render resources to the task evenly (using N nodes to drive M parallel rendering requests). One of the primary ideas in CSLB algorithm is the one-to-one correspondence between the fixed rendering node and the rendering request as shown in Figure 2.

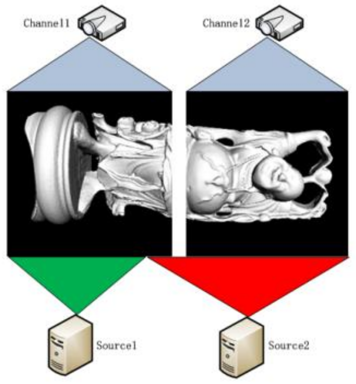

Figure 2. CSLB algorithm

The CSLB algorithm holds the feature of "inter-frame continuity" in rendering technology. It introduces a dynamic task partitioning strategy based on time feedback to analyze the rendering time of the previous frame and the nodes' status so as to obtain the corresponding allocation strategy. In the rendering system, the strategy introduces the concept of channel and source where each channel corresponds to a display area. The number of channels and the display area are set before the system start. The number of rendering nodes (GPUs) that control each channel is not one-to-one. As shown in Figure 2, the image on the left is larger than the image on the right. The server node analyzes the usage of the available resources by counting the time information of the previous frame, and then calculates the optimal task distribution scheme for each drawing node so that Source1 and Source2 may complete the rendering task of Channel1 together. While Source2 completes the rendering task of Channel2 alone.

3. PROPOSED LOAD BALANCING ALGORITHM

3.1 Parallel Rendering System Architecture

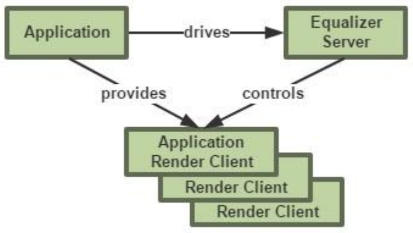

Figure 3. Equalizer process diagram

This paper applies a parallel rendering framework according to OpenGL real-time 3D graphics rendering Equalizer. Equalizer adopts Client-Server model as shown in Figure 3. The architecture is mainly composed of three parts: server nodes and , application nodes, rendering client nodes. The server node is the manager of the whole display system. It runs in the form of a separate process. It is responsible for the configuration management of the system, the automatic operation and management of the remote nodes, the distribution of the running and synchronization of running time. The application nodes are independent process, which establishes a connection with the server node and obtains configuration data. After that, the application node triggers the rendering task on the client nodes and responds to the events from the server node. The client nodes are in charge of the rendering task.

3.2 Algorithm Description

Although the load-balancing algorithm based on time-space conversion can predict the next frame, it may cause the failure of prediction due to the sudden change of the scene. Then the rendering node may have difficulty to respond quickly and affect the rendering performance. CSLB algorithm mainly predicts and changes the task allocation of the next frame based on the rendering situation of the previous frame. This method is less accurate and the stability is also very poor due to the small size of predicted samples.

In this paper, a multi-frame joint analysis method is proposed, which can use multi-frame feedback prediction to achieve better practical results. This method improves CSLB algorithm and presents a multi-frame prediction load balancing approach. The algorithm is mainly based on the known historical average model (Informed Historical Average[8]) as shown in equation (1):

$$t_{i+1} = \frac{t_i}{s_{x,i}} \times s_{x,i+1} \tag{1}$$

Where *i* denotes the number of drawn frames, *x* denotes the number of samples used for prediction, t_{i+1} is the prediction value for the next frame, t_i is the actual value for the current frame , $s_{x,i}$ is the average of the previous *x* frames, $s_{x,i+1}$ is the average of the previous *x* frames and the next predicted frame..

This paper assumes that the *m* frames are the number of samples. The historical data is used to predict the load of the next frame. If $i < m$, then let $x = i$, the samples are the data from the first frame to the *i*-th frame. If $i \geq m$, then let $x = m$, we choose the *m* frames from the current frame back to front. (2) and (3) can be obtained by applying the above two cases to (1) :

[1] Supported by National Natural Science Foundation of China under Grant No. 61300196, Major National Scientific & Technological Projects of China underGrant No. 2013ZX03002006, China Fundamental Research Funds for the Central Universities under Grant No. N120417003, N120404010, and China NaturalScience Foundation of Liaoning Province under Grant No. 201202069.

$$t_{i+1} = \frac{t_i}{\frac{1}{x}(t_1+t_2+...+t_i)} \times \frac{1}{x+1}(t_1+t_2+...+t_i+t_{i+1}) \tag{2}$$

$$t_{i+1} = \frac{t_i}{\frac{1}{x}(t_{i-x}+t_{i-x+1}+...+t_i)} \times \frac{1}{x+1}(t_{i-x}+t_{i-x+1}+......+t_i+t_{i+1}) \tag{3}$$

The improved algorithm proposed in this paper is based on CSLB to change its single frame load forecast to multi-frame load forecast. The CSLB algorithm is only predicted by the drawing time of the current frame, $t_{dest} = t_{source\_average} \times \sqrt{n_{source}}$, where t_{dest} indicates the channel's prediction time, $t_{source\_average}$ represents the average rendering time of source, n_{source} indicates the number of sources. We need to set the sample frames *m* and store these data into the time queue in our improved algorithm beforehand. After that, the multiframePre () method is utilized to calculate the predicted time of the channel. The detailed steps are shown in Algorithm1.

Algorithm 1. Multi - frame Prediction Load Balancing Algorithm

```
Input: Samle frames m, Render modle (Happy,Lucy,Dragon)
1:Initialization
2:Set the load listener
3:Calculate the total number of available GPUs $n_{GPU}$
4:for each frame do
5:Computetime of the next frame to draw each destination
6:     for each channel do
7:Compute the  next prediction time
8:          Call multiframePre () method: find out $T_{dest}$
               $T_{dest} = \frac{1}{m+1}(\sum_{i-m}^{i} T_{source\_average} + T_{dest}) \times \sqrt{n_{source}}$
9:     end for
10:    $T_{total} = \sum T_{dest}$ //Compute total drawing time of a frame
11:    $T_{GPU} = \frac{T_{total}}{n_{GPU}}$ //Compute average drawing time of the GPU
12:    for each channel do
13:         Compute the number of CPUs needed:
14:         $n_{dest\_GPU} = \frac{T_{dest}}{T_{GPU}}$
15:         Usage of the destination's own GPU::
16:         $n_{self\_usage} = \min(1.0, n_{dest\_GPU})$
17:         $n_{dest\_remaining} = n_{dest\_GPU} - n_{self\_usage}$
18:         if $n_{dest\_remaining} > 0$ then
19:       According to the allocation of the last frame of the GPU to compensate for $n_{dest\_remaining}$
20:         $n_{dest\_remaining} = n_{dest\_remaining} - n_{dest\_lastframe}$
21:         end if
22:         if $n_{dest\_remaining} > 0$ then
23:              Add other available GPUs to $n_{dest\_remaining}$
24:         end if
25:    end for
26:end for
Output:Frame rate
```

4. EXPERIMENTS AND ANALYSIS

4.1 Experimental Environment

According to the parallel rendering system architecture proposed in this paper, the drawing cluster in the experimental environment is composed of two drawing computers. The hardware configurations are shown in Table 1. The network bandwidth is 500MB. All the drawing nodes are installed Ubutun14.04 operating system. In this paper, a large number of parallel rendering experiments are done to show the efficiency between CSLB and our propose algorithm(MPLB). The source data of the experiment is from Stanford Scan repository as shown in Figure 4 and Table 2.

Table 1. Hardware configuration

Number	Configuration	RAM
1	Intel(R) Core(TM)i7-4790 CPU	12GB
2	Intel(R) Core(TM)i7-4790 CPU	8GB

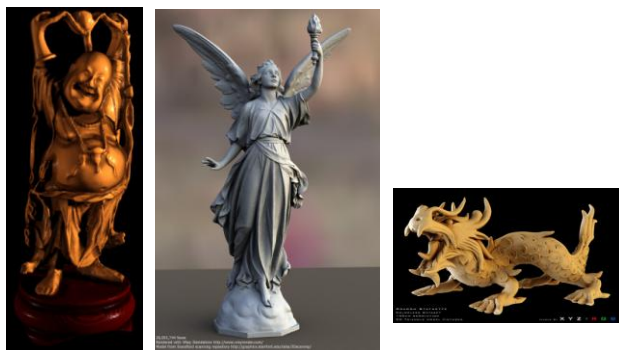
(a) Happy model (b) Lucy model (c) Dragon model

Figure 4. Rendering model

Table 2. Model data information

Model	Model Size / MB	Number of vertices	number of triangles
Happy	40.6	543,652	1,087,716
Dragon	130	3,609,455	7,218,906
Lucy	508	14,027,872	28,055,742

4.2 Experimental results

The experiment manually controls the size of the sample *m* and obtains the appropriate sample capacity by several experiments. Three tests are carried out with sample size *m* = 5, *m* = 10 and *m* = 20, respectively.

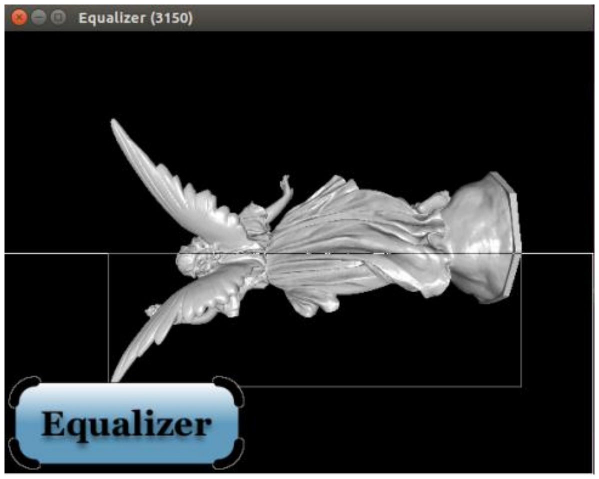

Figure 5. Rendering results of Lucy model

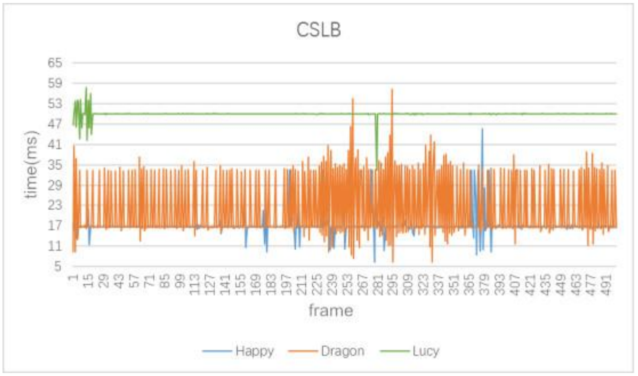

Figure 6. Rendering results of CSLB

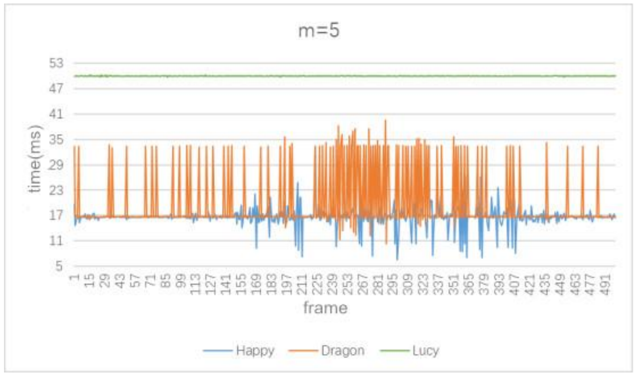

Figure 7. Rendering results of MPLB (m=5)

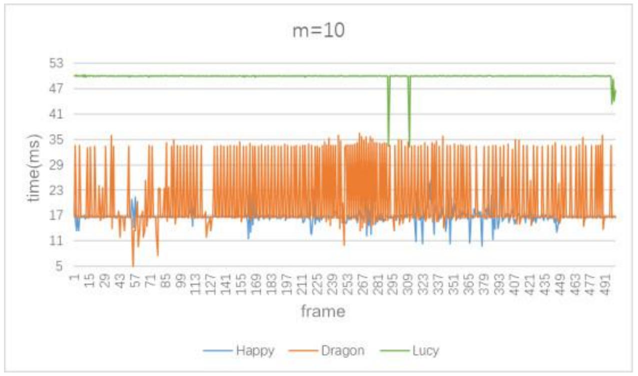

Figure 8. Rendering results of MPLB (m=10)

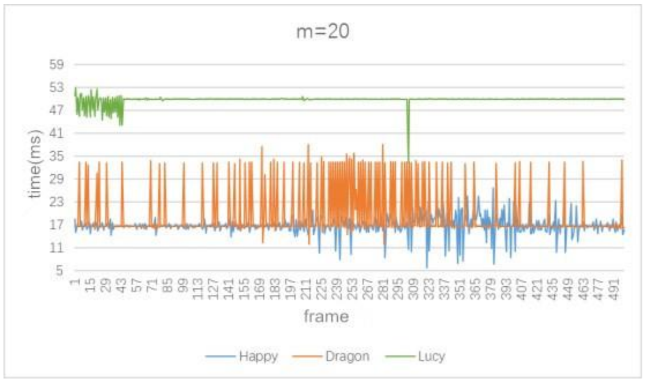

Figure 9. Rendering results of MPLB (m=20)

Table 3. Average frame rate of different algorithms

Model	Average frame rate /fps			
		MPLB		
	CSLB	M=5	M=10	M=20
Happy	60.1717	61.6561	62.9115	60.9124
Dragon	52.0894	55.1120	57.6272	55.0123
Lucy	20.0275	19.9987	21.0537	20.0764

Each test extracts the rendering time of 500 images as experimental data. The average frame rates of CSLB and MPLB, when rendering the 3 models, are shown as Table 3.Figure 5 shows the rendering results of Lucy model. The whole results are shown in Figure 6-9.

As shown in Table 3, we can see the average frame rate of MPLB is higher than that of CSLB mostly in the same model. While only when *m*=5, the frame rate drops by 0.0288 fps, which may due to the large amount of task and the small samples.

5. CONCLUSION

In this paper, according to the idea of time-space conversion, the CSLB algorithm is improved and a multi-frame prediction load balancing algorithm is proposed. The algorithm can address the inaccurate and unstable issues caused by single frame prediction. The experimental results show that the proposed algorithm owes higher average frame rate than CSLB and is relatively stable when selecting the appropriate sample capacity.

In the future research we will do the following works: (1) Collect a certain number of drawing historical data to optimize the prediction algorithm. (2) Combine the algorithm with geometric data-based algorithm to gain high prediction accuracy.

6. REFERENCES

[1] Minfeng Peng. Research on the Key Technology of Parallel Mapping System Architecture[D]. National University of Defense Science and Technology, 2006.

[2] Shi S, Hsu C H. A Survey of Interactive Remote Rendering Systems[J]. Acm Computing Surveys, 2015, 47(4):57.

[3] Rizzi S, Hereld M, Insley J, et al. Large-Scale Parallel Visualization of Particle-Based Simulations using Point Sprites and Level-Of-Detail[J]. 2015.

[4] Minfeng Peng, Liang Zeng, Sikun Li. Research on Dynamic Load Balancing Algorithm for Parallel Rendering[J]. computer application, 2007, 27(1):166-168.

[5] Rontteng Wu. Load Balancing Algorithm and Parallel Execution Time Prediction of Parallel Computing System[D]. Tianjin University, 2008.

[6] Erol F, Eilemann S, Pajarola R. Cross-segment load balancing in parallel rendering[C]// Eurographics Conference on Parallel Graphics and Visualization. Eurographics Association, 2011:41-50.

[7] Eilemann S. Equalizer Programming and User Guide: Version 1.2[M]. CreateSpace, 2012.

[8] Wenxin Qiao, Ali Haghani, Masoud Hamedi. A Nonparametric Model for Short-Term Travel Time Prediction Using Bluetooth Data[J]. Journal of Intelligent Transportation Systems, 2013, 17(2):165-175.