Predictive Model for Distributed Rendering using Dynamic Load Balancing

Dr. Ganesh V. Patil[a],

[a]*Associate Professor,DYPCET,Kolhapur, India, e-mail: pganeshv@gmail.com;*

Abstract: Rendering of animation is computational intensive errand. Rendering of complex scenes requires high computational power setup. In absence of high end computational setup rendering creates a big challenge for a computer animator to maintain quality of service. In present approach we solve this problem of complex scene rendering using prediction based dynamic load balancing mechanism. This model uses frame as a unit of distribution and does impartial distribution of animation frames among the computational nodes to achieve efficient rendering. Proposed prediction model utilizes Index, Glossiness, Light, width and height as a basic key attributes of rendering frame.

Keywords: *Distributed Rendering, Dynamic Load Balancing, Blender 3D.Blender, Rendering.*

1. INTRODUCTION

Modern era of animation based computer graphics demands use of imaginary effects to be inculcated in film industry, animation movies and many more simulation based scientific applications. It comes in reality with the process of animation. Animation is a process of adding imaginary effects in series of still images with respect to time. Core of animation process is rendering. Rendering is the process of generating a 2D image which comprises several phases such as modeling, setting materials and textures, placing virtual light sources. Rendering algorithms take a description of geometry, materials, textures, light sources, and the virtual camera as input in order to produce an image or a sequence of images (in the case of animations)as output [1].Rendering adds photorealistic effects in 3D geometric models. Unfortunately this process is computationally intensive and requires a lot of time when the rendering technique simulates global illumination issues [1].To achieve highly photo realistic images more computational power is required. In proposed work we experimented a task of 3D animation rendering using distributed rendering farm setup. The core part of current experimental work is use of dynamic load balancing in distributed rendering phenomena. The task of rendering is professionally carried out with the high end computing setup. Instead of this in current experimentation we are using low cost networked rendering setup which achieves efficient rendering of complex 3D scenes.

Distributed system gives coordinated network of computing resources to achieve efficient execution. Distributed system itself is a good solution for achieving good response time in rendering of complex animation scenes. Rendering adds photorealistic effects to geometrical images to have real time environmental effects. The process of rendering requires high computational setup as it involves most CPU intensive calculations. Load in these types of complex computationally intensive task is very high. Imbalance in load distribution causes an increase in response time. To achieve an impartial load distribution we are applying a load balancing concept in distributed rendering. Basically load balancing is categorized in two classes i.e. static and dynamic. Static load balancing requires prior information of computational resources. In static load balancing decision making is based on the static state information of resources involved in computation. As we are considering the process of web based distributed rendering static load balancing cannot give proper justice. Dynamic load balancing provides more suitable platform for distributed rendering as the task assignment is done at dynamic time. In dynamic load balancing decision of load balancing is based on the runtime information of computational resources. The run time state information is base of job scheduling in dynamic load balancing. Technique of dynamic load balancing in distributed rendering provides a more promising platform in real time processing. In current exhibit think we are using Blender 3D based rendering farm setup. Blender 3D is a open source software and provides high quality photorealistic effects at low cost.

This research paper is organized in five sections. Section two elaborates related work in the era of animation rendering. Section three illustrates methodology and architecture of current exhibit think. Section four contains results and discussions.

2. RELATED WORK

In today's world rendering becomes unavoidable part of computer graphics. A colossal amount of research is going in sector of computer graphics. Rendering attracts more and more researchers to achieve good response time and efficiency.

In [1] Carlos Gonzalez-Morcillo et. al. presented MAgArRO as a distributed, non centralized , multi agent rendering architecture to optimize the rendering process by

making use of expert knowledge of previous jobs. Experimental results prove optimization in rendering time.

In research think exhibition [2] Thu D. Nguyen et. al. described cluster based real time rendering toolkit i.e DDDDRRaW which supports a repeated, low-latency computation, the drawing of frames, which must take place on a time scale of 30-100 ms. DDDDRRaW applies Image Layer Decomposition as a rendering-specific work partitioning algorithm. It applies the concept of parallelism to evaluate the performance of rendering.

Steven Molnar et. al. [3] presents classification based parallel rendering algorithm. Sorting of object coordinates to screen coordinates has done to carry out parallel processing of both geometry processing and rasterization. Analysis of both computational and communication cost of parallel rendering has exhibited using rendering pipeline.

Huabing Zhu et. al.[4] has given main focus to develop the rendering algorithm which splits screen space into tiles and assign them to rendering nodes to achieve more performance in rendering.

Risto Rangel-Kuoppa et. al. [5] has experimented multi agent based rendering platform. Each individual object is rendered first and then aggregation has done to achieve a realistic virtual environment. It mostly considers object movements and remote communication requirements.

Dynamic Pixel Bucket algorithm given by Huabing Zhu et. al. [6] mainly focus on task allocation in distributed rendering environment. Load balancing based rendering environment has experimented using computational grids.

Frederico Abraham et. al. [7] has presented multithreaded sort first distributed rendering algorithm which achieves load balancing in both geometry processing and rasterization.

Ali Sheharyar et. al.[8] presented a dynamic rendering environment to reduce overall resource utilization in university campus. It works well for system where computational load is more than rendering load without physical separation of actual resources.

Nicolae Tapus et.al.[9] puts light on incapability of a dedicated parallel computational set up as scalability and portability could not be obtained so easily on those architectures using rendering application.

Erick Irawadi Alwi et. al. [10] experimented MPICH cluster system using POV-Ray distributed processing software. According to author they had average speedup of 2.68 and average increase in efficiency rate by 92%.

Ruby Annette.Ja et. al. [11] provides knowledge about the RaaS (Rendering-as-a-service) Services business models and enables the animators to compare the popular RaaS services easily and effectively using the tree structured taxonomy.

Akira Takeuchi et. al. [12] presents an improvement on an efficient image compositing algorithm for sort-last parallel rendering i.e. binary-swap (BS) method. It uses three acceleration techniques, (1) the interleaved splitting, (2) multiple bounding rectangle, and (3) run-length encoding.

Jiaqi Wu et.al. [13] gives analysis of challenges for a animation rendering industry is given using aloud computing theory. It also establishes a private cloud platform for rendering process and had given key factor analysis for practical animation rendering problems.

Cristian F. Perez-Monte et. al. [14] focuses mainly on frame losses in multiple GPU-based heterogeneous nodes distributed graphics rendering system.

Stefan Eilemann et.al. [15] gives OpenGL based equalizer system i.e. toolkit for scalable parallel rendering. Equalizer provides API for development of scalable graphics applications for a wide range of systems ranging from large distributed visualization clusters and multiprocessor multipipe graphics systems to single-processor single-pipe desktop machines.

Ganesh Patil et.al. [16] presented a distributed Blender 3D frame based rendering system. This system considers run time parameters i.e. CPU and RAM utilization for task distribution. This is a self configured PXE booting based rendering system.

3. METHODOLOGY OF WORK-

Figure 1 shows the basic methodology of proposed work.

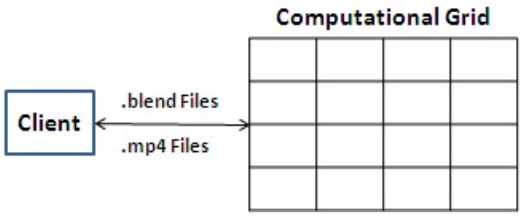

Figure 1: *Basic Methodology of the Work*

In this client submits .Blend file as a input to rendering farm based computational grid which returns final .mp4 file as a output.

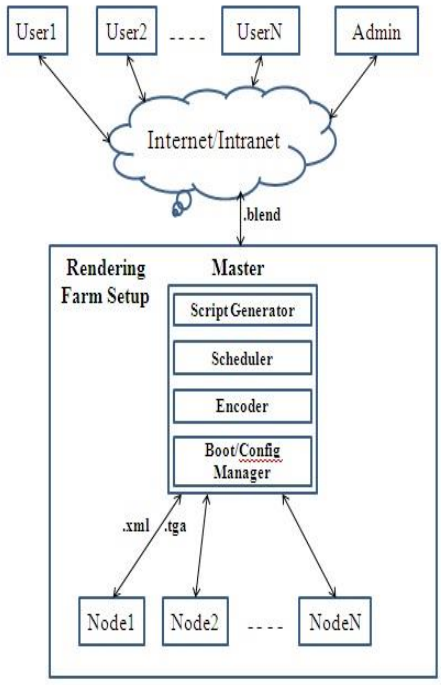

Figure 2: *Architecture of Proposed Work*

Figure 2 shows architecture of the proposed work. Users submit the .blend files as an input to the system which is collectively submitted to a master node.

Master node comprises of following components:

1. **Script generator**

Its task of script generator to convert the .blend file in .xml file. In our experimental work we are using YafaRay [18] rendering engine. YafaRay [18] accepts only .xml file as a input so we have converted .blend file to .xml files. Blend file

consists of multiple frame in sequence. Each frame is converted into a separate .xml file.

2. **Scheduler**

Scheduler periodically collects dynamic state information from the underlying networked resources i.e. CPU and RAM utilization and takes the scheduling decisions based on the dynamic state information [17]. We are using predictive dynamic load balancing mechanism. When input frame comes to scheduler it calculates the predictive time required for rendering and then schedule it for execution based on the periodical state information.

3. **Encoder**

Rendering node converts an .xml file in .tga file and sends back as an output to a master node. The responsibility of master node to collect all .tga files and convert those files collectively in final .mp4 files using Encoder.

4. **Boot/config Manager**

We are using PXE boot based auto configuration mechanism. As any computational node is added in a network a preconfigured operating system with rendering engine is installed on that node. This process of auto configuration is managed by Boot/Config manager.

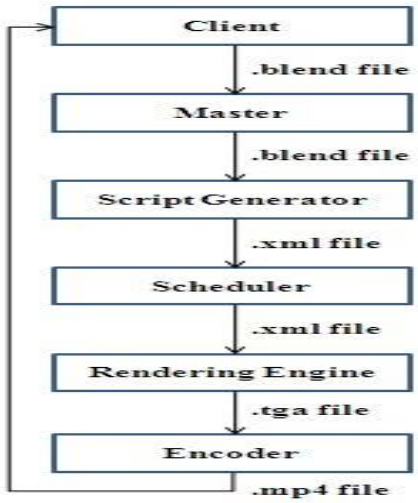

Figure 3: *Basic Flow Diagram*

Figure 3 shows flow of how input .blend file is converted into .mp4 file.

4. EXPERIMENTAL SETUP

Proposed Architecture is based on central coordinated system as shown in Figure 2. Details of node specification are given in Table 1. Details of software tools is given in Table 2.

Table 1: *Computational set up specifications*

Sr. No.	Node Details	Hardware specifications
1	Manager	Dell Vostro 1014,CPU-Intel Core 2 Duo,2.10GHz,RAM-2GB, OS- Ubuntu 11.10
2	Worker	Dell Optiplex 380,CPU-Intel Core 2 Duo,2.93GHz,RAM-2GB, OS-Windows 7(32 Bit)
3	Network	Giga Bit Network

Table 2: *Software tools used for computation*

Sr. No.	Software Name	Purpose
1	DnsMasq	Dynamic Host Configuration Protocol
2	Thrift-0.7.0	Library for distributed programming
3	Blender-Yafaray	Reendering Engine
4	TinyCoreLINUX	Kernel for PXE Booting
5	Ffmpeg	.tga to .mp4 converter

5. RESULTS AND DISCUSSIONS

Table 3. *Results obtained by preprocessing the input rendering frame*

Sr. No.	Index	Gloss	VCount	Light Count	Width	Height	Total Time
1	1.5	0.349	64	1	50	50	19
2	1.5	0.349	64	1	100	100	60
3	1.5	0.349	64	1	200	200	139
4	1.5	0.349	4096	1	50	50	54
5	1.5	0.349	4096	1	100	100	210
6	1.5	0.349	4096	1	200	200	813
7	1.5	0.349	262144	1	50	50	237
8	1.5	0.349	262144	1	100	100	835
9	1.5	0.349	262144	1	200	200	3252
10	1.5	0.349	64	4	50	50	15
11	1.5	0.349	64	4	100	100	50
12	1.5	0.349	64	4	200	200	200
13	1.5	0.349	4096	4	50	50	84
14	1.5	0.349	4096	4	100	100	342
15	1.5	0.349	4096	4	200	200	1321
16	1.5	0.349	262144	4	50	50	335
17	1.5	0.349	262144	4	100	100	1371
18	1.5	0.349	262144	4	200	200	5688
19	1.5	0.349	64	8	50	50	33
20	1.5	0.349	64	8	100	100	69
21	1.5	0.349	64	8	200	200	256
22	1.5	0.349	4096	8	50	50	117
23	1.5	0.349	4096	8	100	100	465
24	1.5	0.349	4096	8	200	200	1864
25	1.5	0.349	262144	8	50	50	502
26	1.5	0.349	262144	8	100	100	2429
27	1.5	0.349	262144	8	200	200	7972
28	1	0.000	64	1	50	50	11
29	1	0.000	64	1	100	100	37
30	1	0.000	64	1	200	200	125
31	1	0.000	4096	1	50	50	56
32	1	0.000	4096	1	100	100	205
33	1	0.000	4096	1	200	200	784
34	1	0.000	262144	1	50	50	204
35	1	0.000	262144	1	100	100	803
36	1	0.000	262144	1	200	200	3212
37	1	0.000	64	4	50	50	14

38	1	0.000	64	4	100	100	48
39	1	0.000	64	4	200	200	183
40	1	0.000	4096	4	50	50	85
41	1	0.000	4096	4	100	100	322
42	1	0.000	4096	4	200	200	1276
43	1	0.000	262144	4	50	50	327
44	1	0.000	262144	4	100	100	1317
45	1	0.000	262144	4	200	200	5241
46	1	0.000	64	8	50	50	19
47	1	0.000	64	8	100	100	69
48	1	0.000	64	8	200	200	257
49	1	0.000	4096	8	50	50	114
50	1	0.000	4096	8	100	100	462
51	1	0.000	4096	8	200	200	1825
52	1	0.000	262144	8	50	50	486
53	1	0.000	262144	8	100	100	1962
54	1	0.000	262144	8	200	200	7922
55	1	0.262	64	1	50	50	9
56	1	0.262	64	1	100	100	26
57	1	0.262	64	1	200	200	90
58	1	0.262	4096	1	50	50	42
59	1	0.262	4096	1	100	100	159
60	1	0.262	4096	1	200	200	608
61	1	0.262	262144	1	50	50	166
62	1	0.262	262144	1	100	100	649
63	1	0.262	262144	1	200	200	2727
64	1	0.262	64	4	50	50	9
65	1	0.262	64	4	100	100	27
66	1	0.262	64	4	200	200	93
67	1	0.262	4096	4	50	50	44
68	1	0.262	4096	4	100	100	165
69	1	0.262	4096	4	200	200	727
70	1	0.262	262144	4	50	50	185
71	1	0.262	262144	4	100	100	652
72	1	0.262	262144	4	200	200	2629
73	1	0.262	64	8	50	50	9
74	1	0.262	64	8	100	100	27
75	1	0.262	64	8	200	200	92
76	1	0.262	4096	8	50	50	63
77	1	0.262	4096	8	100	100	169
78	1	0.262	4096	8	200	200	613
79	1	0.262	262144	8	50	50	165
80	1	0.262	262144	8	100	100	670
81	1	0.262	262144	8	200	200	2581

Table 3 shows results obtained by preprocessing the incoming rendering frame submitted by the user. As mentioned in Table 3 we have considered Index, gloss, vCount, lightCount, width, height as the key factors for analysis of .blend rendering frame. We have applied a multiple regression on the results obtained and derived a relational equation of rendering time i.e. Equation 1.

$$\text{Total Rendering Time} = 54863.2 + 705 \times \text{Index} - 259.9 \times \text{Reflectivity} + 0.006 \times \text{vCount} + 65.87 \times \text{Light} + 12.41 \times \text{Width} + 1.72 \times \text{CPU} + 701 \times \text{RAM}$$

Equation (1)

By using the equation 1 master calculates the total rendering time required by each node and the assigns the rendering job to a node with minimum rendering time. To test the significance of the multivariate analysis we are using P test validation. Table 4 shows P values of the multivariate analysis.

We are testing the hypothesis at significance level $\alpha = 0.05$.

$\beta 1$ = Index, $\beta 2$ =gloss, $\beta 3$ =vCount, $\beta 4$ =lightCount, $\beta 5$ =width, $\beta 6$ =height.

H0 : $\beta 1 = \beta 2 = \beta 3 = \beta 4 = \beta 5 = \beta 6 = 0$.

Ha : $\beta 1 = \beta 2 = \beta 3 = \beta 4 = \beta 5 = \beta 6 \neq 0$.

Table 4: *P values of multivariate regression analysis*

Attribute	P-value
Intercept	0.000265
Index	0.037411
Gloss	0.029218
vCount	1.23E-09
lightCount	0.049341
Width	6.72E-09
Height	0.029341

As all P values are smaller than α we accept the alternate hypothesis. This fact proves that all these factors are significant in calculating the rendering time.

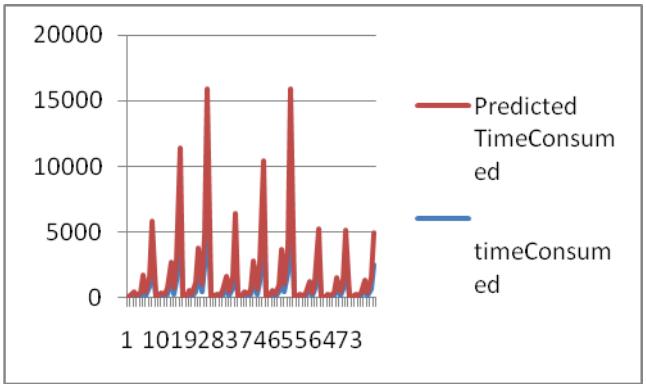

Figure 4: *Comparison of Predicted Rendering Time and Actual Rendering Time*

Figure 4 shows the comparison of predicted rendering time and actual rendering time. It is proven that our predictive model gives approximately same results as actual.

CONCLUSION

In this research work we have presented a predictive dynamic load balancing methodology for distributed rendering using lender 3D. We have considered CPU and RAM as run time attributes for dynamic load balancing. Analytical

regression method used in this work does well for predictive rendering mechanism. Dynamic load balancing mechanism best suits for the rendering work as it involves massive CPU and RAM intensive tasks.

REFERENCES

- **[1]** Carlos Gonzalez-Morcillo; Gerhard Weiss; David Vallejo; Luis Jimenez-Linares; Jose Jesus Castro-Schez, A Multiagent Architecture For 3D Rendering Optimization, Applied Artificial Intelligence: An International Journal,**2010**, 24:4, pp.313-349.
- **[2]** Thu D. Nguyen; Christopher Peery; and John Zahorjan; DDDDRRaW: A Prototype Toolkit for Distributed Real-Time Rendering on Commodity Clusters, Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS), **2001**, pp. 1-13.
- **[3]** Steven Molnar, Michael Cox; David Ellsworth, Henry Fuchs; A Sorting Classification of Parallel Rendering, IEEE Computer Graphics and Applications, **1994**, 14(4), 23-32.
- **[4]** S. See; H. Zhu; K. Y. Chan; L. Wang; W. Cai; DPBP: A Sort-First Parallel Rendering Algorithm for Distributed Rendering Environments, In: Proceedings. International Conference on Cyberworlds(CW), **2003**, pp. 214, 2003
- **[5]** R. Rangel-Kuoppa; C. Aviles-Cruz; D. Mould; Distributed 3D Rendering System in a Multi-agent Platform Proceedings of the Fourth Mexican In: International Conference on Computer Science, **2003**.
- **[6]** Huabing Zhu; Lizhe Wang; Chan Kai Yun; Wentong Cai; A Distributed Rendering Environment for Massive Data on Computational Grids, In: Proceedings of the Third International Conference on Peer-to-Peer Computing, **2003**.
- **[7]** Frederico Abraham; Waldemar Celes; Renato Cerqueira; Joao Luiz Campos; A load-balancing strategy for sort-first distributed rendering, In: Proceedings of the XVII Brazilian Symposium on Computer Graphics and Image Processing, **2004**.
- **[8]** Ali Sheharyar; Othmane Bouhali; A Framework for Creating a Distributed Rendering Environment on the Compute Clusters, International Journal of Advanced Computer Science and Applications, **2013**, Vol. 4, No. 6, pp. 117-123.
- **[9]** Tapus N.; Slusanschi E.; Popescu T.; (2002) Distributed Rendering Engine. In: Grigoras D.; Nicolau A.; Toursel B.; Folliot B. (eds); Advanced Environments, Tools, and Applications for Cluster Computing. IWCC. Lecture Notes in Computer Science, **2001**, vol 2326. Springer, Berlin, Heidelberg, pp.207-215.
- **[10]** Erick Irawadi Alwi; Teguh Bharata Adji; Sujoko Sumaryono; Implementation and Performance Analysis of MPI Cluster System in Distributed Rendering, International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences,**2012,** pp.225-229.
- **[11]** Ruby Annette.J*;* Dr. Aisha Banu.W; Subash Chandran. P; Rendering-as-a-Service: Taxonomy and Comparison, In: 2nd International Symposium on Big Data and Cloud Computing, Procedia Computer Science 50, **2015,** pp. 276 – 281.
- **[12]** Akira Takeuchi; Fumihiko Ino; Kenichi Hagihara; An improved binary-swap compositing for sort-last parallel rendering on distributed memory multiprocessors, Parallel Computing 29 **2003**, pp.1745–1762.
- **[13]** Jiaqi Wu; Huahu Xu; Design of 3D Rendering Platform based on cloud Computing, In: 4th International Conference on Enterprise Systems, pp. 153-159, **2016**.
- **[14]** Cristian F. Perez-Monte; Mauricio D. Perez; Silvio Rizzi; Fabiana Piccoli; Cristian Luciano; Modeling Frame Losses in a Parallel Alternate Frame Rendering System with a Computational Best-effort Scheme, Computers and Graphics, **2016**.
- **[15]** Stefan Eilemann; Maxim Makhinya; Renato Pajarola; Equalizer: A Scalable Parallel Rendering Framework, IEEE Transactions On Visualisation And Computer Graphics, **2009**, Vol. 15, No. 3.
- **[16]** Ganesh V. Patil; Dr. Santosh L. Deshpande; Distributed rendering system for 3D animations with Blender, In: IEEE International Conference on Advances in Electronics, Communication and Computer Technology,pp. 91-98, **2016**.
- **[17]** Ganesh V. Patil; Santosh L. Deshpande; Image optimisation using dynamic load balancing, Int. J. Knowledge Engineering and Data Mining, **2018**, Vol. 5, Nos. 1/2,pp. 68-89.
- **[18]** Yafaray at www.yafaray.org (Accessed January 22, 2018).