Distributed Rendering Based on Fine-Grained and Coarse-Grained Strategy to Speed up Time and Increase Efficiency of Rendering Process

Melki Sadekh Mansuan[1] and Benfano Soewito[2]

[1]Computer Science Department, School of Computer Science, Bina Nusantara University
[2]Computer Science Department, BINUS Graduate Program - Master of Computer Science, Bina Nusantara University
Jln. K.H. Syahdan No. 9, Jakarta Barat 11480, Indonesia
[1]melky23i@binus.edu; [2]bsoewito@binus.edu

Received: 23rd October 2018/ **Revised:** 24th January 2019/ **Accepted:** 31st January 2019

How to Cite: Mansuan, M. S., & Soewito, B. (2019). Distributed Rendering Based on Fine-Grained and Coarse-Grained Strategy to Speed up Time and Increase Efficiency of Rendering Process. *ComTech: Computer, Mathematics and Engineering Applications, 10*(1), 15-22. https://doi.org/10.21512/comtech.v10i1.5067

***Abstract* -** The purpose of this research was to solve several problems in the rendering process such as slow rendering time and complex calculations, which caused inefficient rendering. This research analyzed the efficiency in the rendering process. This research was an experimental study by implementing a distributed rendering system with fine-grained and coarse-grained parallel decomposition in computer laboratory. The primary data used was the rendering time obtained from the rendering process of three scenes animation. Descriptive analysis method was used to compare performance using speedup and efficiency of parallel performance metrics. The results show that the distributed rendering method succeeds in increasing the rendering speed with speedup value of 9,43. Moreover, the efficiency of processor use is 94% when it is applied to solve the problem of slow rendering time in the rendering process.

Keywords: distributed rendering, fine-grained strategy, coarse-grained strategy, rendering process

I. INTRODUCTION

Today, three-dimensional (3D) animations are widely used in broadcasting, advertisement, and movie industry. The process of generating 3D animation movies in accordance with the needs of the industry requires an understanding of production pipeline (Cantor & Valencia, 2004). The animation is produced by sequent images. Each image is generated by rendering, which is the process of generating an idealistic image from a geometric 3D model and various graphics input data such as texture, material, color and light (Glez-Morcillo, Vallejo, Albusac, Jiménez, & Castro-Schez, 2011).

Rendering is the process of generating an image from a model or calling a scene, which is done through a computer program. A scene file contains an object of geometry information, viewpoints, textures, lighting, and shadows. Then, it becomes the description of the virtual scene. The data will be processed in the rendering program to a digital image or raster image. Although rendering techniques vary, in general, the challenge is to produce images of two-dimensional (2D) representations of 3D stored in the file scene. It is also depicted in a pipeline graphics on the device rendering, such as Graphics Processing Unit (GPU) (Akenine-Möller, Haines, & Hoffman, 2008).

The parallel technique is the right solution when rendering performance becomes an issue. The applications such as real-time simulation, animation, virtual reality, photorealistic images, and scientific visualization have leveraged the use of parallelism to improve rendering time. Parallel rendering has been applied in almost all image-making techniques used in computer graphics. It includes surface rendering and polygon, terrain, volume, ray-tracing, and radiosity. Even though the needs and approaches for each technique are different, there are some important concepts in understanding how parallelism applies in common rendering problems.

Parallel computing is a technology that has been developed. Its use ranges from the need for calculations in the laboratory of nuclear physics, the simulation of spacecraft, and weather forecasts. Parallel computing is the use of a collection of computer resources together to solve computing problems (Culler, Singh, & Gupta, 1999). Fundamentally, parallel computers divide the problem into small to be done by each Central Processing Unit (CPU) at the same time (concurrent). This principle is known as parallelism.

Decomposition or division of workload is fundamental to parallel algorithms. It is because the purpose of parallel computing is to improve the process of solving problems using shared computer resources. Based on the division of objects, decomposition is divided into two: data decomposition (domain) and function decomposition (Silva & Buyya, 1999).

Decomposition can customize the problem handled as in the repetition processing or large data input iteration. The used functions are very difficult to implement in parallel, so the data decomposition is better to be used. For the problems with various functions, function decomposition can be used. The determination of this decomposition is very influential on the performance or improvement of parallel computer processes.

Another important point in parallel algorithms is computation and communication comparisons. The computation means the process performed on each processor while communication is the process of information exchange that occurs between processors. A problem with simple functions generally provides a larger portion of computation than communication or granularity is called coarse grain. Meanwhile, a problem with many functions results in the number of linear communication with the amount of computation or Fine Grain Granularity.

Moreover, the parallel algorithm governs granularity. It enables the efficiency of the process in both computation and communication. The amounts of computation and communication also fit to be applied in parallel computer architecture.

Speeding up the rendering process is necessary since rendering is a process that requires intensive computing and time-consuming resources to create a 2D image (Sheharyar & Bouhali, 2014). For example, in James Cameron's film "Avatar", a frame takes an average of 40 hours of rendering to produce 2D images. The film is approximately 2 hours with a rendered frame over 216.000 frames and must be rendered twice to produce stereoscopy images. Overall, it is estimated that it takes 2.000 years if it is done using a single computer (Glez-Morcillo & Vallejo, 2011).

To reduce the rendering time, each frame can be computed in parallel or distributed to a group of computers on the network. A type of parallel processing is called Distributed Rendering (DR). Currently, there are many researchers in addressing the problem of rendering with parallel and distributed processing approaches. Sheharyar and Bouhali (2014) implemented a framework for building DR on cluster computers using a rendering farm management software called Qube! and OpenPBS as cluster management. Moreover, Wang, Zhao, Xu, and Liu (2013) conducted research by implementing a sort-first algorithm with adaptive dynamic load-balancing. It was able to solve the load-unbalance problem in rendering by adjusting the rendering load dynamically on the computer node.

Then, Hong, Wang, and Shi (2014) used a performance-based approach for assigning work to DR based on rendering time performance index and providing a formula for evaluating rendering performance on nodes. Similarly, Kantert, Spiegelberg, Tomforde, Hähner, and Müller-Schloer (2015) introduced a Trusted Desktop Grid (TDG) technical approach with resource sharing to accelerate the rendering process among filmmaking companies. The built system was based on trust, so that from the security side, the grid could be isolated to avoid attacks with little impact on performance.

DR can guarantee to reduce rendering time significantly. However, it depends on the rendering task assignment strategy which all rendering processes are not delayed due to many complex and complicated rendering tasks. Referring to the distribution of 3D frames or animations that will be rendered, two main strategies can be distinguished. There are fine-grained and coarse-grained (Glez-Morcillo & Vallejo, 2011).

Fine-grained is a type of parallel decomposition in which the processors of multiple computers are connected in parallel to render on one frame only. Meanwhile, coarse-grained is also a type of parallel decomposition. It renders the animation frames in succession on a separate computer processor. Thus, each CPU will compute the entire frame of animation with the same configuration parameters on each engine render.

In this research, a proposed method has been presented to implement a distributed rendering method using fine-grained and coarse-grained strategy. This model consists of three components: manager, server, and submitter or monitor. The manager manages the entire process and distributes the work among all the computers in DR environment. The server is the computer on the network used to render frames of animation. It will contact the manager and notify if it is ready to render. Then, the submitter or monitor is a computer that must have a 3D application. It is the machine that initiates a rendering job.

There are currently many researchers in rendering farms for 3D animation. This concept has been implemented in the movie business to accomplish photo-realistic images approach by parallelism. Wald, Benthin, Dietrich, and Slusallek (2003) built a render farm by using personal computer cluster for interactive ray tracing system with custom optimized renderers. However, it is only for non-real-time rendering. In undergraduate research at the University of Wisconsin-Eau Claire, Bui, Boettcher, Jaeger, and Westphal (2013) utilized clusters for animation. In this project, an input animation scene file was processed to create a set of rendering jobs that generated a composed 2D images to form an animated movie. The system was Distributed System for Automated Blender Rendering (DSABR). DSABR rendered animations by sending a task to a group of computers using the Work Queue framework.

Render farm framework belongs to (Patoli *et al*., 2009). They adopted a Grid-based render farm "Condor" to use idle resources of the enterprises' machines when it was free. The user required to install a small Condor slave program to be part of a Grid.

Gooding, Arns, Smith, and Tillotson (2006) also offered a render farm. The design of the render farm was built upon the TerraGrid network. TerraGrid was a large grid computing architecture contributed mainly by universities. The problem faced was about the communication of master controllers and the worker computers. To reduce this problem, they used Condor. The other problem was the security of the system. To deal with this problem, a submission phase was designed having authentication and encryption.

On the other hand, there is also open source render farm such as DrQueue by adopting almost all renderers for distinct users, and mainly targeting the service quality. Moreover, there are commercial platforms including Qube, Deadline, and others. Those offer the system management to render farms that support managing user accounts and render fee (Fang, Zhao, & Wang, 2009).

On the other hand, remote render farm such as RenderRocket has exploit of a computing resource in a global online service environment. This method uses Grid or volunteer computing. Performing such remote rendering systems must consider design network interface and handle data access in this architecture (Anderson, 2004). The presented framework in this research claims that all render farm features can be provided at the same time. The preferred 3D software is open source Blender by the Script Generator feature. Theoretically, all 3D software can be integrated into this framework.

The purpose of this research is to implement the distributed rendering method with fine-grained and coarse-grained strategies to increase rendering time and efficiency of processor use. It is expected to become a solution in production process of an animation project. Thus, the product can be done on time.

II. METHODS

Figure 1 is the architecture of the distributed rendering system. In this system, the machine manager acts as a network manager. It is the manager's job to accommodate the efforts of all other machines in the distributed rendering environment. A rendering server is a machine on the network used to render frames of animation. It contacts manager and informs that this machine is ready to render. The rendering server starts 3ds Max when the manager sends a frame to be rendered. A submitter is a machine that must have an authorized copy of 3ds Max running. It initiates a rendering job. This machine also has a function as a monitor. It checks the current state of jobs that are rendering or that have been queued. Then, it can use schedule the rendering time.

The monitor can get the valuable information pertaining to all the jobs in the render queue. The researchers use a file server to tell 3ds Max where it can find the information to render a scene. The 3ds Max must find the location of textures and other information and know where to put each frame that it renders.

The researchers implement a network model in client-server and use Transmission Control Protocol/Internet Protocol (TCP/IP) to handle network communication. Then, Server Message Block (SMB) protocol for the file server that acts as network file sharing.

A fine-grained strategy will be applied to the manager that will assign rendering servers to render one frame simultaneously. If there are ten rendering servers, one frame will be divided into sliced images, and the finished slice will be composed into one image. A fine-grained strategy workflow can be seen in Figure 2.

The workflow of fine-grained parallel decomposition strategy applied is as follows. First, the manager receives a render job submitted by the submitter with rendering settings using a fine-grained strategy. Second, the manager will assign tasks to render servers connected in the render farm to render by splitting the frames into chunks according to the number of render servers. Third, if the number of render servers is ten, the first stage of the ten render servers will render the frame fragments. Then, the second stage of the rendering server will be assigned to merge the finished pieces rendered into one rendered or intact image. If the number of frames to be rendered is more than one, the frames will be animated. Then, this process will be repeated until all the frames are finished in the rendering.

Moreover, a coarse-grained strategy will also be applied. The manager will assign each render server to render one frame simultaneously. If there are ten rendering servers, each server will render one frame. Thus, there are ten rendered frames simultaneously. If there are hundred frames in one animation, each server renders ten frames.

The workflow of coarse-grained decomposition strategy in Figure 3 is as follows. First, the manager receives a render job submitted by the submitter with rendering settings using a coarse-grained strategy. Second, the manager will assign tasks to render servers connected in render farm to render by dividing each frame according to the number of render servers. Third, if many render servers are n, the number of n servers will render many n frames. Moreover, if the number of frames to be rendered is more than one like frames of animation, the rendering process with many servers will run together until the number of rendered frames ends.

The researchers conduct performance analysis to compare time rendering between existing and proposed methods. Rendering time is evaluated by the speedup and efficiency function equation (Eager, Zahorjan, & Lazowska, 1989) as follows:

$$S(n) = \frac{T(1)}{T(n)} \tag{1}$$

Where, $S(n)$ is speedup, $T(1)$ is time rendering single computer, and $T(n)$ is time rendering (n) node. Meanwhile, the efficiency parameter is defined as the average use of n processors dedicated to rendering. The equation is as follows:

$$E(n) = \frac{S(n)}{n} \tag{2}$$

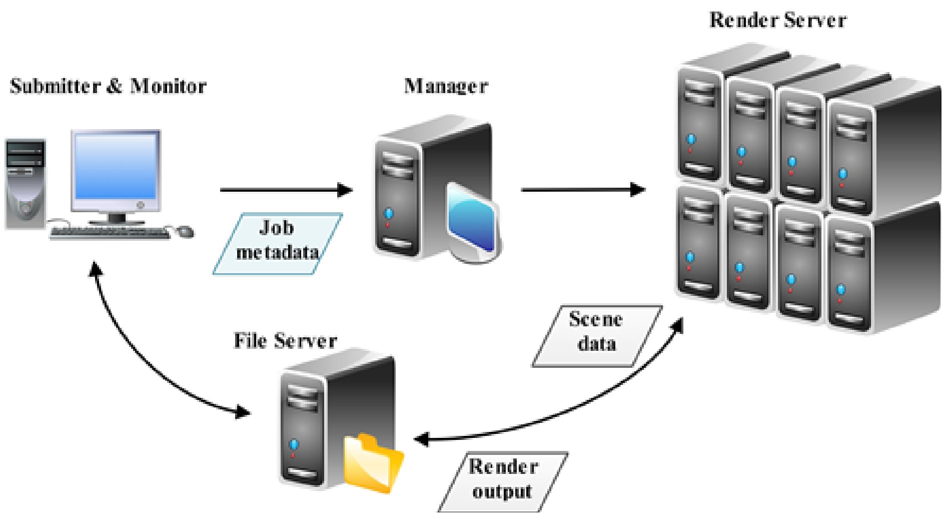


Figure 1 Architecture of the Distributed Rendering System

The $E(n)$ is the efficiency of (n) processor, $S(n)$ is speedup, and n is the number of n processor.

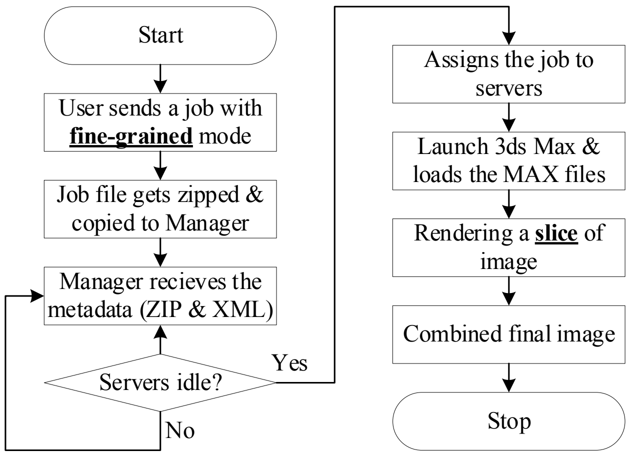

Start
User sends a job with **fine-grained** mode
Job file gets zipped & copied to Manager
Manager recieves the metadata (ZIP & XML)
Servers idle?
Yes
No
Assigns the job to servers
Launch 3ds Max & loads the MAX files
Rendering a **slice** of image
Combined final image
Stop

Figure 2 DR Fine-Grained Procedure

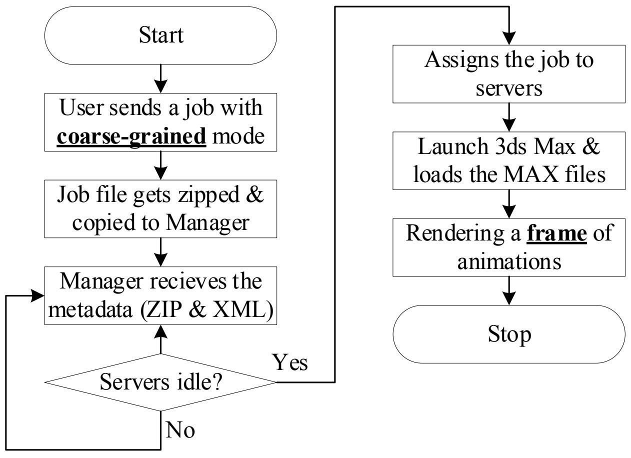
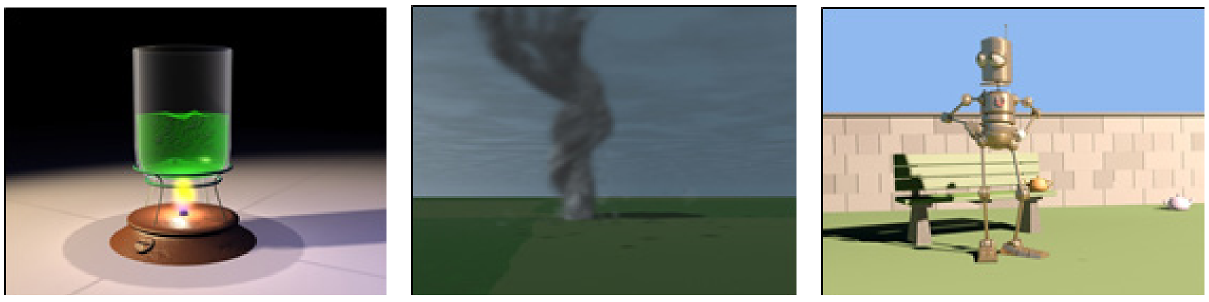

Start
User sends a job with **coarse-grained** mode
Job file gets zipped & copied to Manager
Manager recieves the metadata (ZIP & XML)
Servers idle?
Yes
No
Assigns the job to servers
Launch 3ds Max & loads the MAX files
Rendering a **frame** of animations
Stop

Figure 3 DR Coarse-Grained Procedure

Table 1 Specification of Animations

	Simple	**Medium**	**Complex**
Scene Name	Wiring-Breaker	Vortex-Tornado	Robby *vs* Fly
Number of Frames	301	301	341
Animation Speed	30	30	30
Resolution	800 x 600	800 x 600	800 x 600
Polygon	23914	306	112899
Vertex	13531	287	57188
Light	3	1	2
Particle System	-	Vortex Space Wrap	-
Number of Particles	-	100000	-
Camera	1	1	1
Compression	JPEG	JPEG	JPEG
Renderer	Scanline	Scanline	Mental Ray



(a) (b) (c)

Figure 4 The Sample Frames of Animations
(a)Wiring-Beaker, (b) Vortex-Tornado, and (c) Robby vs Fly

III. RESULTS AND DISCUSSIONS

The experiments are conducted in a computer laboratory using a computing platform based on Intel Xeon W3520 2.67 GHz CPU, 4 GB RAM, and Microsoft Windows 7 Professional 64-bit with SP1 operating system. The distributed rendering is performed using Autodesk Backburner since it is a standard tool for handling rendering and compiling tasks for a range of software tools developed by Autodesk.

In this experiment, three scenes of animation that comes with Autodesk 3ds Max are given in Table 1 and Figure 4. These scenes have different complexity including simple, medium, and complex animation.

The researchers conduct experiments on three scenes by rendering using Local, Distributed Rendering with Fine-Grained (DR FG), and Distributed Rendering with Coarse-Grained (DR CG) method. The results of rendering time are reported in Table 2. The first rendering is conducted using Local (single computer). Then, DR FG with one additional server is registered. For DR CG process is the same with DR FG.

To verify a significant difference, the results of all methods are compared. The researchers perform the speedup and efficiency comparison on each method. Moreover, the researchers calculate the speedup using equation 1 and efficiency using equation 2. Then, the researchers compare the results as shown in Tables 3 and 4.

Table 2 The Results of Rendering Time

Method	**Rendering Time (sec)**		
	Wiring-Beaker	**Vortex-Tornado**	**Robby vs Fly**
Local	716	2612	7054
DR FG 2	857	1575	4262
DR FG 3	692	1150	3109
DR FG 4	611	961	2441
DR FG 5	540	880	2038
DR FG 6	522	794	1864
DR FG 7	509	742	1734
DR FG 8	503	684	1586
DR FG 9	497	661	1490
DR FG 10	455	635	1387
DR CG 2	321	1394	3531
DR CG 3	251	922	2362
DR CG 4	178	713	1792
DR CG 5	158	572	1453
DR CG 6	135	477	1218
DR CG 7	120	421	1051
DR CG 8	112	409	922
DR CG 9	110	413	824
DR CG 10	105	378	748

Table 3 Speedup Comparisons

Method	**Speedup**		
	Wiring-Beaker	**Vortex-Tornado**	**Robby vs Fly**
Local	1,00	1,00	1,00
DR FG 2	0,84	1,66	1,66
DR FG 3	1,03	2,27	2,27
DR FG 4	1,17	2,72	2,89
DR FG 5	1,33	2,97	3,46
DR FG 6	1,37	3,29	3,78
DR FG 7	1,41	3,52	4,07
DR FG 8	1,42	3,82	4,45
DR FG 9	1,44	3,95	4,73
DR FG 10	1,57	4,11	5,09
DR CG 2	2,23	1,87	2,00
DR CG 3	2,85	2,83	2,99
DR CG 4	4,02	3,66	3,94
DR CG 5	4,53	4,57	4,85
DR CG 6	5,30	5,48	5,79
DR CG 7	5,97	6,20	6,71
DR CG 8	6,39	6,39	7,65
DR CG 9	6,51	6,32	8,56
DR CG 10	6,82	6,91	9,43

Table 4 Efficiency Comparisons

Method	**Efficiency (%)**		
	Wiring-Beaker	**Vortex-Tornado**	**Robby vs Fly**
Local	100	100	100
DR FG 2	42	83	83
DR FG 3	34	76	76
DR FG 4	29	68	72
DR FG 5	27	59	69
DR FG 6	23	55	63
DR FG 7	20	50	58
DR FG 8	18	48	56
DR FG 9	16	44	53
DR FG 10	16	41	51
DR CG 2	112	94	100
DR CG 3	95	94	100
DR CG 4	101	92	98
DR CG 5	91	91	97
DR CG 6	88	91	97
DR CG 7	85	89	96
DR CG 8	80	80	96
DR CG 9	72	70	95
DR CG 10	68	69	94

In the first calculation, the researchers compare the speedup value for Wiring-Beaker scenes. DR FG method has good speedup compared to Local method (one server). The lowest value of speedup is 1,06 (two servers), and the highest is 1,86 (ten servers). For DR CG, the lowest speedup value is 2,23 (two servers), and the highest one is 6,82 (ten servers).

From the results, it can be seen that DR FG for the process of rendering simple animations (Wiring-Beaker) on many different servers can improve rendering performance by utilizing the processor even though it is not maximized. It is explicitly seen that this animation sample is very simple. However, the large numbers of rendering processes cause a large amount of communication occurring between master and server rendering. If the scenario uses DR FG 5, the master assigns each frame to be rendered by five servers simultaneously. This means that there are five communications in one rendered frame. Thus, for this simple animation that has 301 frames, there will be 1.505 communications on five servers and 3.010 communications on ten servers.

The next analysis is the efficiency of processor usage on each server. The calculation results can be seen in Table 4. For DR FG method, the highest efficiency value is 42% (two servers), and the lowest value is 16% (nine and ten servers). For DR CG the highest efficiency value is 112% (two servers), and the lowest value is 68% (ten servers).

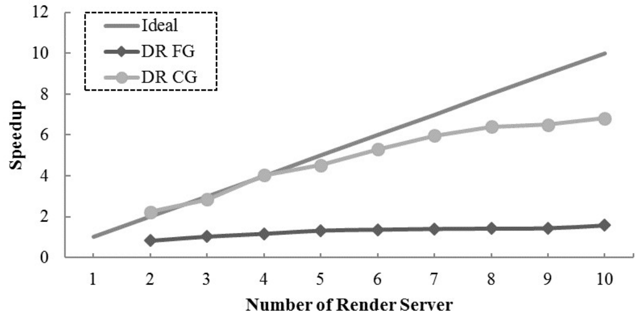


Figure 5 Speedup Comparison of Wiring Beaker Animation

The speedup comparison graph of DR FG and CG method for simple animation (Wiring-Beaker) is shown in Figure 5. From the graph, it is known that speedup of DR FG has changed stagnantly in every increase in the number of servers. Unlike DR CG which has an increase in speedup proportional to the increase in the number of servers, there is a saturation starting from the number of servers from seven to ten.

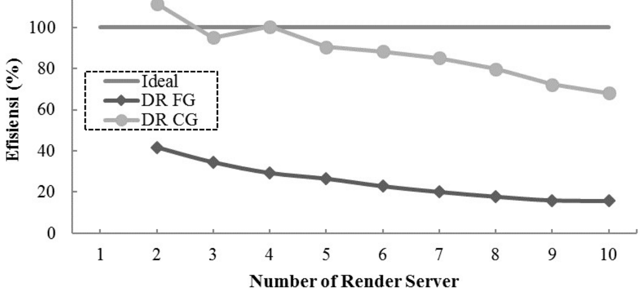


Figure 6 The Efficiency Comparison of Wiring-Beaker Animation

Comparison of efficiency for DR method with FG and CG strategies in the process of rendering simple animations is shown graphically in Figure 6. It shows that the efficiency value obtained by DR CG method is greater than DR FG method. This is an important note that DR FG method has not been able to maximize the distributed rendering process even with a large number of servers. Meanwhile, DR CG method can optimally perform the distributed rendering processes. It is because the utilization of processors on each used server has an average value of efficiency above 50%.

In the next calculation, the researchers compare the speedup value for Vortex-Tornado scenes. The implementation of DR FG method has a good speedup compared to Local method (one server). The lowest speedup value is 1,87 (two servers), and the highest value is 4,60 (ten servers). For DR CG method, the lowest speedup value is 1,87 (two servers). It is equal to the speedup value of the DR FG. Then, the highest value is 6,91 (ten servers).

From speedup results, it is found that the implementation of DR FG for medium animation rendering process on many different servers can improve the rendering performance by utilizing the processor on the server even though it is not maximal. Although this animation has a small number of polygons, there are 100.000 particles. Thus, it causes the rendering process to be computationally intensive. There is a large amount of communication occurring between master computer and servers. With scenario using DR FG 2, the master will assign each frame to be rendered by two servers simultaneously. This means that two communications occur in one rendered frame. For this medium animation that has 301 frames rendered, 602 communications will occur for two servers. Then, 3.010 communications are for ten servers. This condition proves that DR FG has a communication ratio that is greater than the computational process or overhead.

Meanwhile, the implementation of DR CG for the same animation can have better speedup than DR FG. The maximum speedup value is 6,91 (ten servers), and the minimum is 1,87 (two servers). However, the speedup is saturated on the number of servers 8, 9, and 10. So, speedup value obtained cannot reach its linear value. For DR CG 2 scenario, two frames will be rendered simultaneously for one-time communication between master and two servers. Thus, only 151 communications occur for rendering 301 frames. In 31 communications for DR CG 10 scenario, the number of servers is ten. This proves that DR CG has a ratio that is more computational than communication.

The next analysis of each DR method is to calculate the efficiency of processor usage on each node, the results can be seen in Table 4. For DR FG method, the highest efficiency value is 83% (two servers), and the lowest value is 41% (ten servers). For DR CG, the highest efficiency value is 94% (2 servers), and the lowest value is 69% (10 servers).

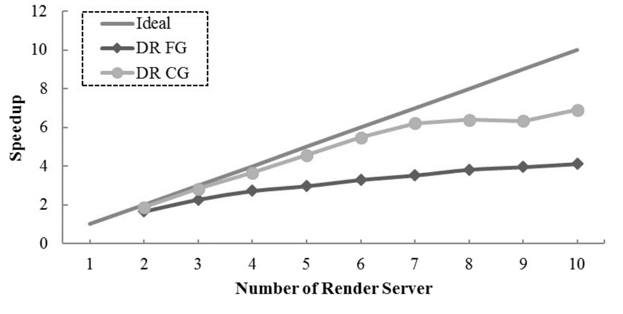


Figure 7 Speedup Comparison of Vortex-Tornado Animation

Speedup comparison of the DR FG and CG method for the graphical animation rendering process (Vortex-Tornado) is graphically shown in Figure 7. From the graph, it is known that speedup from the implementation of DR FG tends to increase in every increase in the number of servers. However, the increase is not directly proportional to the increase in the number of servers. In contrast to the implementation of DR CG which has a trend of increasing speedup linearly to the increase in the number of servers, there is a saturation starting from eight to ten servers.

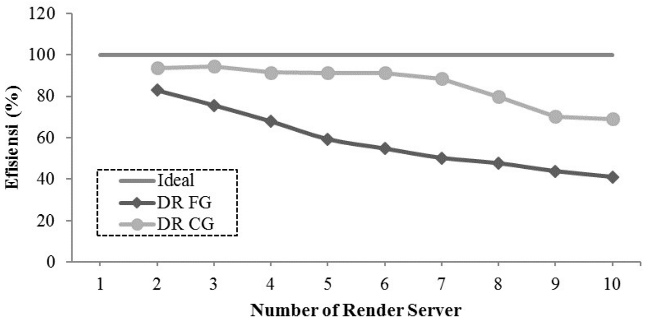


Figure 8 The Efficiency Comparison of Vortex-Tornado Animation

Efficiency comparison chart for implementation DR FG and CG in the rendering process of the animation medium (Vortex-Tornado) is shown in Figure 8. From the graph efficiency value obtained by DR CG method, it shows it is better than DR FG. It should be noted that the implementation of DR FG method has not been able to maximize the distributed rendering process even with a large number of servers. Although the efficiency looks good with the value of 83% in two servers, compared to CG DR with the same number of servers, the result is better with the value of 94%. In general, DR CG method can perform a maximally distributed rendering process, because the use of processors used has an average value above 60%.

Last, the researchers compare speedup value for Robby vs Fly scenes. The implementation of the DR FG method has a good speedup compared to Local method (one server). The lowest value is 1,77 (two servers), and the highest value is 5,38 (ten servers). For the implementation of DR CG method, the lowest speedup value is 2,00 (two servers), and the highest speed is 9,43 (ten servers). The speedup and efficiency values for all rendering processes can be seen in Tables 3 and 4.

From the speedup results, it shows that DR FG for complex animation in the rendering process on a number of different servers can improve rendering performance by utilizing the processor. However, it has not been utilized maximally as it is the same as the previous animation samples. For complex animation samples with 341 frames, it causes a large amount of communication that occurs between master and servers. If the number of the server is two, the master will assign each frame to be rendered by two servers simultaneously. This means that two communications occur in one rendered frame. So, for this sample that has 341 frames, 682 communication will occur for the number of servers is 2 and 3.410 communications for the number of servers is 10. This means that communication is greater than the computing process.

Meanwhile, the implementation of DR CG has a better speedup than DR FG. The average of all values almost reaches a linear value according to the addition of the number of servers. This is because the DR CG can perform an efficient rendering process with less communication between the master computer and the server. With DR CG 2 scenario, two frames will be rendered simultaneously for one-time communication between the master computer and two servers. Thus, there are only 171 communications for the rendering process in 341 frames. Moreover, it is only 35 communications with ten servers. This proves that DR CG has a greater computational process than its communication.

Next, the analysis of each DR method is to calculate the efficiency of processor usage on each node. The results can be seen in Table 4. For the DR FG method, the highest efficiency value is 88% (two servers), and the lowest value is 19% (ten servers). For DR CG, the highest efficiency value is 100% (two servers), and the lowest value is 94% (ten servers).

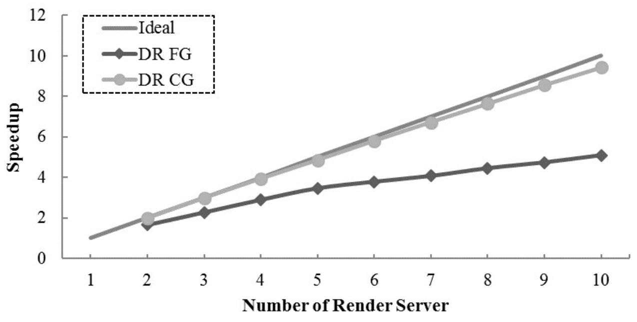


Figure 9 Speedup Comparison of Robby vs Fly Animation

The speedup comparison graph of DR FG and CG method for the complex animation (Robby vs Fly) is shown in Figure 9. From Figure 9, it is known that speedup of DR FG tends to increase in every addition the number of servers, although it is not significant. In contrast, in DR CG, almost all speedup values are close to linear values.

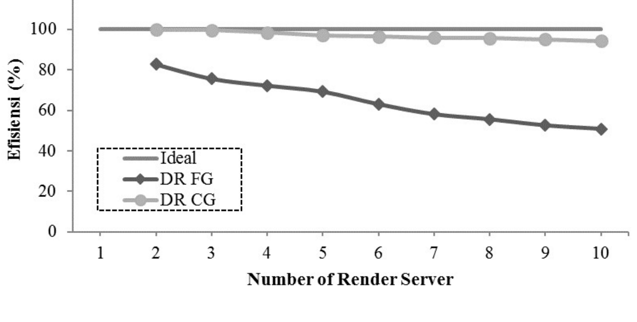


Figure 10 Efficiency Comparison of Robby vs Fly Animation

The efficiency comparison chart for implementation DR FG and CG in the rendering process of the animation complex (Robby vs Fly) is shown in Figure 10. It can be seen that the efficiency value of DR CG method is greater than DR FG. This chart proves that the implementation of

DR FG method has not been able to maximize the processor in the distributed rendering process although there are many numbers of servers. Meanwhile, DR CG method can maximize the processor for distributed rendering processes and have an average value above 90% in this animation.

IV. CONCLUSIONS

Distributed rendering based on fine-grained and coarse-grained strategy is proposed to improve the performance of rendering in 3D animation. DR method with fine-grained is good for animation rendering processes at all levels of complexity. It can improve rendering performance. However, it does not maximally utilize the number of available servers. In this research, there are ten servers. It can be seen from the maximum speedup value obtained in complex animation samples. It is only 5,38, and the efficiency of processor usage is 88% in 10 servers. Meanwhile, the DR method with coarse-grained for the entire rendering process in all three animations can improve the rendering performance by almost reaching the number of servers used. In the complex animation sample, the maximum speedup value is 9,43, and the efficiency of processor usage is 100% in 10 servers. For future research, it is necessary to experiment on larger network coverage such as Grid and Cloud networks. Moreover, the future researchers can optimize the modeling process for animation that can speed up rendering time.

REFERENCES

Akenine-Möller, T., Haines, E., & Hoffman, N. (2008). *Real-time rendering* (Third edition). A K Peters/CRC Press.

Anderson, D. P. (2004). Boinc: A system for public-resource computing and storage. In *Fifth IEEE/ACM International Workshop on Grid Computing* (pp. 4-10). https://doi.org/10.1109/GRID.2004.14

Bui, P., Boettcher, T., Jaeger, N., & Westphal, J. (2013, September). Using clusters in undergraduate research: Distributed animation rendering, photo processing, and image transcoding. In *2013 IEEE International Conference on Cluster Computing (CLUSTER)* (pp. 1-8). https://doi.org/10.1109/CLUSTER.2013.6702634

Cantor, J., & Valencia, P. (2004). *Inspired 3D short film production*. Course Technology Press.

Culler, D., Singh, J. P., & Gupta, A. (1999). *Parallel computer architecture: A hardware/software approach*. Gulf Professional Publishing.

Eager, D. L., Zahorjan, J., & Lazowska, E. D. (1989). Speedup versus efficiency in parallel systems. *IEEE Transactions on Computers, 38*(3), 408-423. https://doi.org/10.1109/12.21127

Fang, C., Zhao, Y., & Wang, Z. (2009). Research and design of a service management system for deadline render farm. In *2009 International Conference on Environmental Science and Information Application Technology* (Vol. 2, pp. 542-545). https://doi.org/10.1109/ESIAT.2009.123

Glez-Morcillo, C., & Vallejo, D. (2011). Using expert knowledge for distributed rendering optimization. In *International Conference on Computer Vision, Imaging and Computer Graphics* (pp. 3-16). https://doi.org/10.1007/978-3-642-32350-8_1

Glez-Morcillo, C., Vallejo, D., Albusac, J., Jiménez, L., & Castro-Schez, J. J. (2011, October). A new approach to grid computing for distributed rendering. In *2011 International Conference on P2P, Parallel, Grid, Cloud and Internet Computing* (pp. 9-16). IEEE. https://doi.org/10.1109/3PGCIC.2011.12

Gooding, S. L., Arns, L., Smith, P., & Tillotson, J. (2006). Implementation of a distributed rendering environment for the TeraGrid. In *2006 IEEE Challenges of Large Applications in Distributed Environments* (pp. 13-21). https://doi.org/10.1109/CLADE.2006.1652049

Hong, Z. G., Wang, Y. B., & Shi, M. Y. (2014). A performance-based policy for job assignment under distributed rendering environments. In *Applied Mechanics and Materials* (Vol. 543, pp. 2949-2952). https://doi.org/10.4028/www.scientific.net/AMM.543-547.2949

Kantert, J., Spiegelberg, H., Tomforde, S., Hähner, J., & Müller-Schloer, C. (2015). Distributed rendering in an open self-organised trusted desktop grid. In *2015 IEEE International Conference on Autonomic Computing* (pp. 267-272). https://doi.org/10.1109/ICAC.2015.66

Patoli, M. Z., Gkion, M., Al-Barakati, A., Zhang, W., Newbury, P., & White, M. (2009). An open source grid based render farm for blender 3D. In *2009 IEEE/PES Power Systems Conference and Exposition* (pp. 1-6). https://doi.org/10.1109/PSCE.2009.4839978

Sheharyar, A., & Bouhali, O. (2014). A framework for creating a distributed rendering environment on the compute clusters. *arXiv preprint arXiv:1401.0608*.

Silva, L. M., & Buyya, R. (1999). Parallel programming models and paradigms. *High Performance Cluster Computing: Architectures and Systems, 2*, 4-27.

Wald, I., Benthin, C., Dietrich, A., & Slusallek, P. (2003). Interactive ray tracing on commodity pc clusters. In *European Conference on Parallel Processing* (pp. 499-508). https://doi.org/10.1007/978-3-540-45209-6_72

Wang, W., Zhao, Z. X., Xu, Q., & Liu, T. (2013). Design and implementation of adaptive dynamic load balancing distributed parallel rendering system based on sort-first. In *Advanced Materials Research* (Vol. 798, pp. 693-698). https://doi.org/10.4028/www.scientific.net/AMR.798-799.693