The problem of physically based rendering in the cloud computing system

Dmitry Afonkin and Dmitry Zhdanov

ITMO University, St. Petersburg, Russia
daafonkin@corp.ifmo.ru, ddzhdanov@corp.ifmo.ru

Abstract. The objective of this investigation is the possibility of building a model of physically based rendering in a cloud computing system. Analyzing various algorithms for rendering process, and various ways to construct a distributed rendering system. There is an investigation of reaching cloud computing properties as high availability, fault tolerance, auto scaling, and flexibility in rendering domain. This paper describes various solutions of the problems of construction with the use of rendering algorithms in distribution context. The paper opens up possible ways to build a distributed rendering system and directions for future research.

Keywords: Physically based rendering, Distributed rendering, Cloud computing system, Distributed system.

1 Introduction

The cloud compute technology is a method of reaching high quality rendering system. The purpose is not related to the theory of quality and other formalization of computing software characteristics. High quality is a useful for end user process these days. The physically plausible 3D graphics has been in demand for the last years. Various fields use photo-realistic graphics for the solutions. For example, rendering of virtual furniture for demonstrations goals [1], rendering car paint [2], medical anatomic images [3] and also entertainment as game graphics [4], virtual and additional reality, cinema when computer-generated imagery is often used. The main method for reaching photo-realistic images is a physical based rendering. The method is consisting of complex processes as geometry computations, texturing and shading. The resource demands are depending on scale of 3D scene. The big scene for the cartoon or cinema cant render on one computer, except a supercomputer [5]. The problem to get 3D realistic image for 3D scene, is solved by a render farm [6]. The solution is related to distributed network renderer. A clear concept of render farm doesnt have a modern property of distributed system such as high availability, fault tolerance, flexibility for end-user process and auto-scale. The problems are from the actual rendering process. I solved these tasks for constructing SaaS system that is unified with collaboration tool for corporations, and decided these methods to allow to build

the modern high quality rendering system. Mathematical and engineering experience will enable to construct, consisting of predict and low-cost system for end user. The solution of the problem of distributed cloud rendering that consists of constructing of formal computation model and researches different rendering algorithms in that model, is to create a flexible and configurable distributed system. This provides new opportunities for rendering big scenes. The first one is, when rendering a big scene. This term doesnt have a formal definition. On this paper, a big scene is an object that occupies more than half of the computing resources of a host; as a result, it cannot be rendered on a single host. An example of such a scene is the city of San Fransokyo from the cartoon Big Hero 6. (Fig. 1 shows scene from cartoon).

Fig. 1. The city San Fransokyo from Disney cartoon Big Hero 6.

The city of San Fransokyo is illuminated by more than 216,000 street lights and it has 83,000 procedural-built buildings with an equal number of street props and trees. On this paper, some algorithms are considered from the render system for a big scene from such cartoon, Big Hero 6. Such system was developed by R&D, a department of Disney company, that shows many methods for out-core rendering. The system is monolith in-house solution and doesnt have cloud properties by design. Such characteristics as flexibility, reliability, availability are needed for efficiency of rendering a big scene for science and production tasks. It enables to construct end-user abstraction as SaaS or PaaS. The big problem is on how to construct computational model with such property for physical-based rendering pipeline. The paper contains review of existing experience and suggested novel methods, and directions for future researchers.

2 Render pipelines

There are three main rendering pipelines. A rasterized rendering pipeline, a ray tracing pipeline, and a hybrid pipeline.

2.1 Rasterization-based rendering

The first pipeline defines rasterization as an image synthesis system. The place of this pipeline is between the processing of primitives and the processing of fragments (in Figure 2, the rendering pipeline is based on rasterization). Rasterization is the projection and processing of simple geometric primitives, mainly given in the form of a triangular grid on a fragment of a raster screen. A fragment is a part of the image obtained as a result of rasterization of the primitive, and might not be an element of the light calculation. The method of rasterization is most effective for constructing simple images, but its a disadvantage in the impossibility of correctly calculating the indirect illumination. Rasterization is not a physically based solution and is usually used in the production of simple video games. It is usually not used to create cartoons and special effects in the film industry, and, especially for physically correct calculations. Therefore, the rendering pipeline, based on the principles of rasterization, does not meet the requirements.

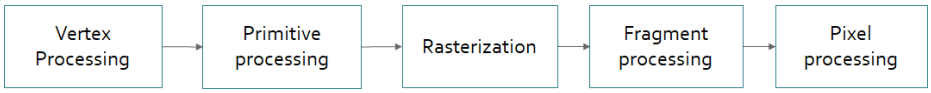

Fig. 2. rasterization-based rendering pipeline

2.2 Ray tracing-based pipeline

It is known that the architecture of a distributed rendering system places high demands on the efficiency and correctness of modeling. Therefore, the most universal methods of forming photorealistic images are the methods based on ray tracing. The methods have their advantages and disadvantages. The simplest variation of the ray tracing method is based on the approach of visualizing sources of primary and secondary scene brightness, i.e. ray tracing (or visualization paths) from the observer's eye to the geometric objects of the scene (Fig. 3. shows the main algorithm for backward ray tracing) [1]. The backward ray tracing algorithm sends rays from the observer's eye through the pixels of the screen into the scene. When the ray intersects with the first element of the scene, the brightness of its direct illumination is calculated. Brightness is depends on the material of the object wherein the ray is intersected and can divide the ray into several rays, for example, a reflection ray and a refraction ray. This process will continue until the specified depth of ray tracing is reached, it will not be absorbed on the object of the scene, or it will not go beyond the limits of the scene. This algorithm does not distinguish between how the beam interacts with the diffuse and specular objects of the scene and is sometimes called naive ray tracing. Ray tracing is a very resource-intensive procedure. For methods of backward ray tracing, time costs can be represented as O (whn), when n is the

number of objects in the scene, w and h is the screen resolution in width and height. There are many methods for optimizing ray tracing, such as adaptive control of the depth of the ray path, accelerating the search for the first ray intersection with the surface, linking objects, linking the hierarchy, and spatial consistency [1]. However, this optimization can also be used for other methods of radiation rendering. In addition, rendering that supports complex optical effects may be ineffective. For example, rendering scenes with objects containing complex patterns of scattering, such as "glossy" light scattering, polarization, fluorescence, subsurface and volumetric light scattering.

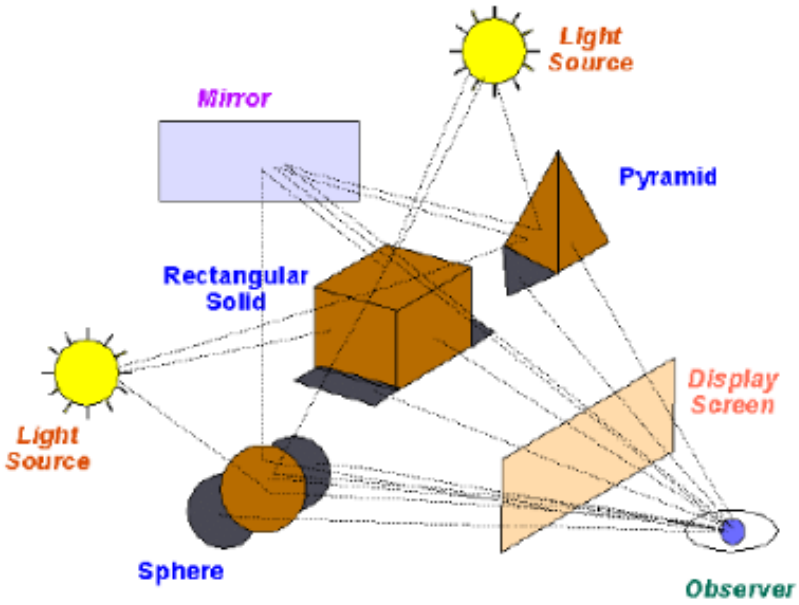

Fig. 3. Basic Ray Tracing algorithm

Another method - the path tracing method - is the generalized ray tracing, which is closest to the physical laws. Rays are traced until they are completely absorbed or leave the stage. In addition, for each scattering event, the direct brightness is calculated, which allows to significantly increase the overall rendering efficiency. However, the effectiveness of this method is not always optimal in scenes with complex, for example, caustic lighting and in the presence of a large number of primary light sources.

The bidirectional ray tracing method combines two methods: path tracing and Forward Monte Carlo Ray Tracing methods. In the first case, the trace comes from the eye of the observer, and in the second case from the light source. By connecting the two paths with shadow rays on the diffuse surfaces of the

scenes, it is possible to estimate the brightness of the secondary illumination for the screen point for which the ray was emitted from the observer. The main problem of this method is the calculation of caustic lighting and the corresponding caustic brightness, since this method is based on the presence of at least two diffuse events on the paths of the direct and reverse rays. The main advantage of this method is its unbiased in assessing the brightness of images of screen dots. The method of photon maps is a fairly simple and effective method of photorealistic visualization. Its main disadvantage is a possible bias in the estimate of the average brightness of the image point. The main idea of this method is to build an irradiance map (photon map) by tracing photons from a light source and using the resulting map to calculate the brightness of the indirect and caustic lighting. When the ray crosses the photon map element, the brightness is calculated, which accumulates at the corresponding points of the image [4]. This is a universal method that allows one to physically correctly estimate the brightness of all components of the illumination, and its one disadvantage is the presence of a bias of the obtained brightness estimate. The presence of bias can adversely affect the initial stages of calculations, however, with long-term modeling using an adaptive photon map, the effect of bias is usually not manifested. On the other hand, it must be remembered that the visualization of the caustic effect can be effectively implemented only in the photon map method. The Metropolis method is an optimized physical simulation method for handling lighting. This method is based on the mutation of a single ray distribution, as in the Metropolis method in computational physics. The method consists of two stages. The first is warming up, which can calculate the distribution of rays (paths) for a future mutation. Various methods can be used for calculations, such as bidirectional ray tracing or photon mapping methods. The next steps are to generate distributions around the sources of local brightness in the scene in accordance with the Metropolis algorithm. Usually, the first step takes a short time with respect to the entire rendering time. The main element of the algorithm is the ray path mutation strategy. It is responsible for the mutations that will determine the overall computational efficiency. This method is universal and unbiased [3]. Of course, there are many modifications of considered methods such as importance sampling techniques and progressive techniques. They allow you to calculate the time and improve the quality of the synthesized image. In conclusion, the study should emphasize the Metropolis method, which can significantly improve the efficiency of image formation for a complex scene, with a large number of light sources, and objects with complex optical properties, creating effects, such as contact shadows, caustics from curvilinear mirror bodies, and secondary illumination from small scene objects size. For example, the light reflection of white tiles under the door and forming a bright strip along the back of the floor. In some cases, the Metropolis method provides better convergence in time and quality of rendering than most classical methods. However, methods based on the calculation and use of photon maps also meet the requirements of physically correct rendering and in most cases, make it possible to pipeline better the effect of caustic lighting. Errors caused by the bias of the photon mapping method are

in most cases not critical and do not violate the correctness of the calculations. For a more accurate assessment of the suitability of the considered methods, it is necessary to conduct a study of their performance on a distributed pipeline [3].

2.3 Hybrid pipeline

The hybrid pipeline combines two methods: rasterization and ray tracing. This allows you to improve graphics in real time and to add additional visual effects obtained by tracing a small number of rays. An example of this approach is to use rasterization to calculate direct illumination, that is, for the first generation of the rays. The small amount of traced rays affects the quality of the final image and cannot provide the physically correct modeling needed for filmmaking when 3D-scene images are built for weeks (for a small movie plot) on supercomputers, as what Disney has done. In conclusion, we can say that at the moment only the pipeline, built on the basis of ray tracing, allows us to solve the problems under consideration.

3 Limitation classic distributed network pipeline

On this paper, classic distributed network pipeline means render farm. Such clean concept of network render architecture doesnt have such cloud property as SaaS or PaaS system for reliable and available rendering of big scene. The main questions here are related to fault tolerance, scaling, and configurable pipeline for rendering big scene. System needs to work after failure more than half of system resources and can full recovering process. Its important for rendering of big scene when computation time takes weeks. Scaling is about dynamic reconfiguration pipeline. To make it faster, simple scaling of resources is the solution and this wont restart the whole pipeline. onfigurable pipeline is important for economic profit. It about the ratio of cost to time and quality. The different task as cartoon or simulation for scientific research have different requirements and different cost with time. The next part reviews of ways to go out from distributed network pipeline limitations and solve considered problems. The first step for solution is decomposition of the system where the whole pipeline is on one host. The unit of decomposition is micro-service. Its a small program module that is independent and autonomous. Such module can replace any other module of corresponding class. Also, it supports horizontal scaling through increase count of micro-service. There are three main class of modules: pre-processing micro-service, ray tracing processing micro-service and post-processing microservice. Pre-processing is consisted of decomposition of scene on voxels and initialization of pipeline. Decomposition allows to distribute pipeline on micro-service, when one service per voxel. Algorithms of voxelization are many. One of corresponding methods is binary space partitioning. This is a method for recursively subdividing a space into voxels. Its process are continuing while voxels has big size for render on one micro-service. As a result, there is hierarchy of objects for ray

tracing processing. Each voxel has type as empty voxel for part of scene without object. It optimizes processing. Ray tracing processing includes one of methods for lighting processing like photon mapping or metropolis light transport. That service has unified interface and there is no meaning concrete method for tracing. Each service can send to other services batch of rays for processing. The process is called streaming. It is the key process for rendering pipeline. It enables the process of out-of-core scene also. The problem of fault tolerance is solved by replication of produced data. Produced data is stored to database server for reliable. When a database server is breaking, one of replica become to main and continuing processing (Fig 4. shows big picture of system).

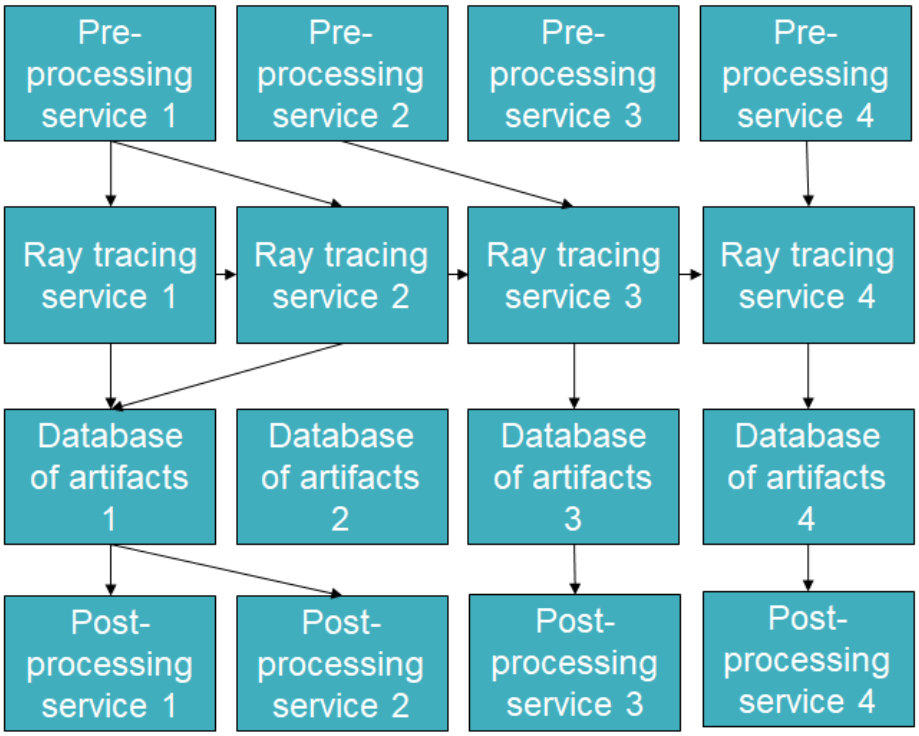

Fig. 4. Bit picture of system

That architecture pattern is called service mesh. Post-processing are forming image, estimate accuracy, construct predictions, doing filtering, and compressing for output artifacts. If that is cartoon then combining. The main issue here is the process of big data output to send an image and cartoon and video to user. There is distributed filtering and processing for solving. As considered important process is streaming between ray tracing services. In the next section reviews reliable solution.

4 Reliable out-of-core streaming

The worst case of rendering is when a scene needs more than one host. To do this, there are many methods as out-of-core geometry and out-of-core ray tracing or out-of-core texturing. In engineering, the method is usually making a stream of data when the data is too big. To reach cloud property, it is necessary to make flexibility, reliability, availability and scaling stream. The problem is solved in a system of big data streaming called Apache Kafka. That system that is a queue solved problems of reliable distributed streaming. It allows transmit a big data through network in reliable and scaling way. Usually Kafka used in pipeline for big data processing (Fig 5. shows common big data pipeline). That

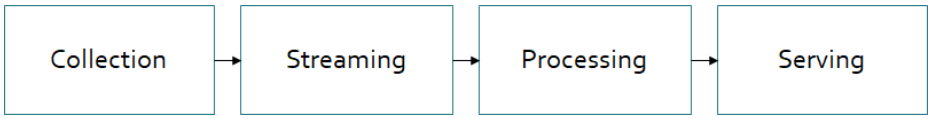

Fig. 5. Common big data pipeline

experience actual for distributed rendering system starting from persistent layer to streaming and processing. For persistence layer, there are variants such as Apache Ignite, PostgreSQL and Apache Cassandra. The estimation of efficiency on these systems in distributed rendering pipeline, is a separate subject for future researches.

Constructed computational model with service mesh pattern enables to estimate various algorithms of various parts of the pipeline. Distributed architecture of system enables to replace any algorithm thus the system will evaluate on time.

5 Conclusion

A study was conducted to investigate other possible ways of organizing cloud computing with distributed processing of the renderer, and the problem of separating a large process between components of a cloud distributed system was also considered. The study showed that the cloud rendering system was being developed will allow solving complex computational problems related to the assessment of visual discomfort caused by the mismatch between the vergence and the accommodation of human vision. With the help of the developed high-performance computing system, it will be possible to perform time-consuming and resource-intensive calculations necessary for synthesizing images generated by virtual prototypes of the human vision system and virtual and mixed reality systems, and for assessing the consistency of vergence and accommodation of human vision in the resulting images.

6 Acknowledgements

This work was partially funded by the RFBR grant No. 18-08-01484.

References

1. Photo-realistic Rendering of Metallic Car Paint from Image-Based Measurements, Rump M., Mller G., Sarlette R., Koch D. and Klein R., Institute for Computer Science II, University of Bonn (2008)
2. Shaping the Future through Innovations From Medical Imaging to Precision Medicine. Comaniciu D., Engel K., Georgesc B., Mans T.,(2016).
3. Photorealism - the future of video game visuals. Stuart K. URL: https://www.theguardian.com/technology/2015/feb/12/future-of-video-gaming-visuals-nvidia-rendering(2012)
4. Disney rendered its new animated film on a 55,000-core supercomputer. Volpe J. URL: https://www.engadget.com/2014/10/18/disney-big-hero-6(2014).
5. Entertainment Computing - ICEC 2009, 8th International Conference, Paris, France, September 3-5, Yao J., Pan Z., Zhang H. 2009. Proceedings (pp.264-269) A Distributed Render Farm System for Animation Production(2009).
6. Antonio H., Martinez V.: Accelerating algorithms for Ray Tracing Subsequences.J. Mol. Biol. 147, 195197(1981)
7. Veach E. and Guibas L. J. Metropolis light transport. Proceedings of SIGGRAPH 97, Computer Graphics, Vol. 31, No. 4, p.p. 6576(1997).