Eurographics Symposium on Parallel Graphics and Visualization (2008)
J. Favre, K. - L. Ma, and D. Weiskopf (Editors)

High-Fidelity Rendering of Animations on the Grid: A Case Study

Vibhor Aggarwal[†], Alan Chalmers and Kurt Debattista

Warwick Digital Lab, University of Warwick, UK
[†]Vibhor.Aggarwal@warwick.ac.uk

Abstract
Generation of physically-based rendered animations is a computationally expensive process, often taking many hours to complete. Parallel rendering, on shared memory machines and small to medium clusters, is often employed to improve overall rendering times. Massive parallelism is possible using Grid computing. However, since the Grid is a multi-user environment with a large number of nodes potentially separated by substantial network distances; communication should be kept minimum. While for some rendering algorithms running animations on the Grid may be a simple task of assigning an individual frame for each processor, certain acceleration data structures, such as irradiance caching require different approaches. The irradiance cache, which caches the indirect diffuse samples for interpolation of indirect lighting calculations, may be used to significantly reduce the computational requirements when generating high-fidelity animations. Parallel solutions for irradiance caching using shared memory or message passing are not ideal for Grid computing due to the communication overhead and must be adapted for this highly parallel environment. This paper presents a case study on rendering of high-fidelity animations using a two-pass approach by adapting the irradiance cache algorithm for parallel rendering using Grid computing. This approach exploits the temporal coherence between animation frames to significantly gain speed-up and enhance visual quality. The key feature of our approach is that it does not use any additional data structure and can thus be used with any irradiance cache or similar acceleration mechanism for rendering on the Grid.

Categories and Subject Descriptors (according to ACM CCS): I.3.1 [Computer Graphics]: Hardware Architecture—Parallel processing; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation and Ray-tracing; I.3.2 [Computer Graphics]: Graphics Systems—Distributed/network graphics

1. Introduction

Nowadays, high-fidelity animations are being used in a predictive manner for applications such as those dealing with medical imaging, architectural walk-throughs, archaeological reconstructions and simulations. Rendering of such animations using physically-based methods takes a long time to complete on a single machine since the computation is typically performed using a Monte Carlo simulation to solve the rendering equation [Kaj86]. Parallel computing is frequently used to improve the rendering times for such animations. However, parallel rendering is usually limited to a few processing nodes, in the order of tens, typically dedicated and situated within the same location. Grid computing offers an inexpensive alternative to highly-parallel computation albeit without the same level of control offered by local dedicated resources.

A Grid can be defined as a distributively-owned multi-programmed large pool of heterogeneous computing resources interconnected by telecommunication channels. It is a multi-user, massively parallel shared resource system with interconnected clusters, databases and equipment spanning administrative and geographic boundaries. Grid computing has helped research in diverse fields such as Medical Research [SPCF*07], Earthquake Simulations [PKG*04], Astrophysics [BP07], Climate Modelling [BBB*05].

Significant amount of work has been done on parallel rendering but little has been done to enable it on the Grid. There are unique challenges while computing on the Grid, since there is a pressing need to minimise communication between the processes to achieve high efficiency. The algorithms used to render on the Grid should be designed such that there is minimum sharing of data between nodes to fully utilise the available resources.

In this paper, a case study on high-fidelity rendering of walk-through animations on the Grid is presented. The initial results of high-fidelity rendering on the Grid demonstrate its computational potential. A straightforward approach is used for adapting a physically-based distributed ray tracing renderer [CPC84] based around the irradiance cache [WRC88] for the Grid. The advantages of using irradiance caching are that it is independent of the geometry and as opposed to other methods such as photon mapping [Jen01] and radiosity [GTGB84] it is view-driven conforming to the more traditional ray tracing approach of computing from the point of view of the camera. With a view-driven approach, a walk-through animation path may visit only certain parts of the model being rendered and since this is known beforehand, only those values which are required are computed rather than computing the global illumination for the whole model. Moreover, irradiance caching is an established method used in film production [TL04, Her04] and there are many other algorithms based on irradiance caching [TL04, AFO05], adaptations used for dynamic scenes [TMD*04, SKDM05] and adaptive versions of it [YPG01, KBPv06, Deb06, DCG*07]. Yet others perform similar approaches to compute rendering features, such as participating media [JDZJ07], glossy interreflections [KGPB05] and subsurface scattering [KLC06]. Most of the above could benefit from our approach via a similar straightforward conversion.

When rendering animation frames in parallel, artefacts are generated due to temporal incoherence if the irradiance cache is not shared between the rendering processes. The artefacts are a result of interpolation of irradiance values from different irradiance caches on close parts of the model appearing in different frames of the animation. Our method uses a two-pass approach for rendering the animations as suggested by [LS98]. The irradiance at selected points along the animation is computed in the initial pass, while the second pass computes the animation in the traditional way. We have been able to eliminate any visual temporal noise in the animations without significantly increasing the communication costs and gaining speed-up due to saving on recomputation of data while rendering high-fidelity animations on the Grid.

This paper is divided as follows: In the next section 1.1, we give a short background on Grid computing. Section 2 presents some previous works in the fields of parallel rendering and irradiance cache. Section 3 discusses the issues related with rendering on the Grid and presents our two-pass approach to overcome these issues. Section 4 contains the results of our two-pass approach. The conclusion and future work are presented in section 5.

1.1. Background

The Grid is envisaged by Foster as a dependable, consistent, pervasive and inexpensive access to high-end computational resources [FK99]. Recently, the field of Grid computing has been given a boost by the standardisation of the Grid technologies under the Open Grid Services Architecture (OGSA) [FKNT02] and development of the Globus Toolkit [Fos06]. More researchers are now looking forward to tapping the enormous potential that lies within the Grid to help them in their research.

Using the Globus Toolkit, a user accesses the Grid and submits jobs to the Grid middleware via secure authentication; which in turn executes the user's job on the available resources.The Grid middleware communicates with local resource management systems such as Portable Batch System (PBS), Load Sharing Facility (LSF), Condor etc. for job scheduling on idle computing nodes. The Grid middleware along with the local resource management system is responsible for providing fault tolerance. There is a delay between when a job is submitted to the Grid and when it is run on a node. This is attributed to the middleware, which needs time to find a suitable match for the job and transfer the data needed for computation to the selected node. Since the Grid is a multi-user environment, there might be a significant delay in job execution due to unavailability of idle resources needed for its execution.

A batch of parallel jobs submitted to the Grid maybe executed on a set of nodes belonging to different clusters. Since jobs may run in parallel on different clusters it's not economical to share data between them in real time. Therefore while adapting algorithms to run on the Grid, care must be taken to make them latency tolerant and bandwidth minimising in order to extract the full potential from it as advised by Messina [Mes99]. Due to the overheads of job submission and retrieval, it is better to run jobs which take considerably longer than these overheads so that their effect can be minimised.

2. Related Work

2.1. Parallel Rendering

Since high-fidelity rendering is a computationally intensive process there have been many attempts at parallelising it. Reinhard *et al.* [RCJ98] and Chalmers *et al.* [CDR02] offer an extensive evaluation of the accepted techniques in the field of parallel rendering looking into issues regarding static and dynamic load balancing, data and task management, and more advanced approaches. Most of the parallel rendering algorithms use either a shared memory model or a distributed memory model to communicate between the parallel processes. Shared memory methods have become more common recently. Parker *et al.* [PMS*99] used shared memory and optimised code for raytracing simple scenes. The Manta interactive ray tracer [BSP06] is designed from the ground up for use with shared memory computation on current commodity hardware. Davis *et al.* [DD99] employed a special data structure storing information on frame coherence in animations, while rendering in a distributed computing environment using Parallel Virtual Machine (PVM). Wald *et al.* [WSBW01] used a master-slave model to compute tiles of the frames asynchronously and communicated between processes using their own TCP/IP based protocol. With this method they managed to achieve interactive ray tracing for moderately complex scenes using up to 24 processors. Wald *et al.* [WKB*02, BWS03] further enhanced their interactive renderer for global illumination using a modified version of instant radiosity [Kel97]. Guenther *et al.* [GWS04] enhanced this even further to fully account for caustics via photon mapping.

Chong *et al.* [CSL06] have provided a framework for rendering animations on the Grid and they look into providing an easy access of Grid services to the end user along with some data management issues. They have developed a lossless compression algorithm to compress the model geometry by hashing duplicate objects in the scene. However, their approach does not take advantage of the temporal coherence between animation frames to save on computation time.

2.2. Irradiance Cache

Traditionally, in a distributed ray tracing system, a large number of rays are shot at each ray intersection point to sample the hemisphere around that point. Ward *et al.* [WRC88], observed that the indirect diffuse component in a scene is a continuous function in space and is not subjected to high frequency variation as the specular component. To exploit this nature of the rendering computation, they proposed an accelerating data structure to store the irradiance values computed in the scene. Whenever a new irradiance sample is needed, the cache is consulted to check if there is another sample already calculated in the cache within the search radius defined by the user. If there exists such a sample in the cache, the sample to be calculated is interpolated from it providing an order of magnitude speed-up.

There have been many attempts at parallelising the irradiance cache. The standard Radiance [LS98], supports a parallel renderer which uses the Network File Sharing (NFS) to manage simultaneous access to irradiance cache between parallel processes on a distributed system. This may lead to file contention while using inefficient file lock managers resulting in poor performance. Koholka *et al.* [KMG99] shared the irradiance values between processes on the slaves using MPI after every 50 irradiance samples were calculated at each slave. Robertson *et al.* [RCLL99] proposed a master slave model of Radiance where each slave calculates and deposits the irradiance cache values to the central master after a threshold, and then gathers the irradiance values calculated by other slaves from the master after a threshold. Debattista *et al.* [DSC06] followed the irradiance caching component-based philosophy by subdividing the computation of the indirect diffuse on a separate dedicated set of irradiance cache nodes, while the remainder of the nodes computed the rest of the rendering. This enabled cached samples to be shared quickly and efficiently amongst the dedicated irradiance cache nodes.

Gautron *et al.* [GBP06] presented an approach for caching of irradiance and radiance to remove artefacts due to temporal incoherence using temporal gradients for rendering animations. Whereas Smyk *et al.* [SKDM05] used a data structure to link irradiance caches to exploit temporal coherence between frames and gained performance boost and better visual quality in animations.

The algorithms discussed above are not suitable for use on the Grid since, as explained earlier in section 1.1, we need to minimise communication between parallel processes to increase efficiency.

3. Rendering on the Grid

3.1. Single-pass Approach

A straightforward approach for rendering on the Grid would be to submit each frame of the animation to the Grid as an independent job and once all frames have been rendered the animation could be compiled [CSL06]. A problem with this approach is the noise artefacts which are generated due to lack of sharing of data between the frames. This is seen as temporal noise (flickering and shimmering) in the compiled animation and may draw viewer's visual attention towards unimportant parts of the animation. Henceforth we will refer to this approach as the single-pass approach.

In the single-pass approach each frame is calculated independently on a different machine on the Grid. Therefore, there is no sharing of irradiance cache samples between machines calculating successive frames. This results in interpolation of irradiance values at same (or relatively close) points in the different frames using different irradiance cache samples (see Figure 1). Theoretically, this problem can be solved if the search radius for the cache is kept very small. But practically, it is not possible to keep the radius very small for rendering in reasonable times. Creating a system to share irradiance cache data between different nodes on the Grid while the frames are being computed would incur large communication cost as different nodes on the Grid may be on different clusters separated by large network distances. Since the Grid is a massively parallel system there would be many copies of the irradiance cache distributed among different nodes and it would be difficult to update each cache with the values calculated by the other nodes in real time without stalling the rendering process.

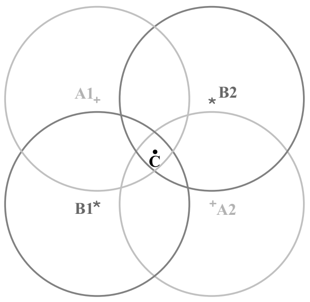


Figure 1: *The irradiance value at point C may be interpolated in one frame of the animation from points A1 and A2, while points B1 and B2 may be used for interpolation in second frame. As the two frames are rendered on different nodes on the Grid, irradiance value at C would be different in the two frames giving rise to temporal noise in the animation.*

Furthermore, as we shall show, the computation of frames is usually slower while rendering using a single-pass approach. The irradiance values calculated on one part of the model in one frame can be reused to interpolate irradiance in successive frames. But there is no sharing of the irradiance cache between the nodes, hence computations done on one node can not be reused for computation on other nodes resulting in recomputation of similar values. This naturally results in extra computation resulting in poorer execution times.

3.2. Two-pass Approach

To overcome the problems discussed in the previous section a two-pass approach is proposed in this paper for rendering on Grid systems. This approach considerably reduces visual temporal noise in animations and speeds up the process taking advantage of coherence between successive frames.The two passes can be summarised as:

1. A set of frames is selected from all the frames to be rendered, by giving equal weight to change in direction and position of camera. These selected frames are rendered in order to generate separate irradiance caches. The caches thus obtained are then merged together on one node.
2. The merged cache is distributed to the assigned computational nodes on the Grid. The scene is then rendered using the merged irradiance cache, queuing each frame as a job on the Grid.

3.2.1. The First Pass

The first pass of our approach consists of rendering selected frames so as to generate the irradiance cache. While selecting the frames we need to keep in mind that the scene geometry should be sampled in such a way that most of the irradiance values in the second pass are interpolated from the irradiance cache generated in the first pass, rather than calculated. If the frames are not selected wisely, scene geometry may not be sampled properly leading to recomputation of irradiance cache samples on different nodes on the Grid. This would result in temporal noise and reduce the performance that can be achieved with a properly sampled geometry. It is possible to sample the geometry directly, but it is simpler to choose a frame from a set of frames.

Knowledge of the camera path allows us to employ a simple heuristic whereby we identify the frames, most viable to generate separate caches. The normalised change of angles (α) and normalised distance travelled by the camera (β) between successive frames is summed (γ). The normalisation ensures an equal importance between α and β. The value of γ helps in determining the greatest level of change between frames. This is used to select N equally spaced frames based on cumulative change of γ where N is the number of idle resources available on the Grid. The value of N is an approximate value observed at the start of the rendering process and differs considerably at and during the runtime since the Grid is a dynamic environment with multiple users. The frame selection process is depicted in Figure 2.

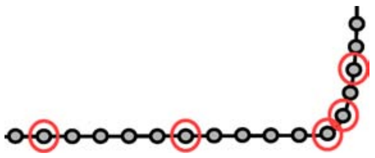


Figure 2: *The encircled frames are selected for the first pass, chosen by giving equal weight to both change in direction and position of the camera.*

Instead of calculating the complete frames a number of pixels are rendered until an irradiance cache hit-miss cache threshold is reached. Since the number of rays needed to be shot is not known until execution, a Sobol sequence [KK02] is used to generate the pixels from which to calculate the rays. This pseudo random sampling helps in shooting rays such that they are reasonably well distributed over the parts of the geometry required by the animation. The rendering of each of the selected frames is queued up as a job on the Grid. The computation is stopped once the number of cache hits to cache misses is 10 to 1. This threshold is checked only after a user-defined base amount of rays (typically 128, but this could be modified based on scene complexity) has been calculated. The various irradiance cache data obtained at each node is then merged by a single node. This merged cache is distributed to the nodes on the Grid in the second pass to compute the complete animation.

3.2.2. The Second Pass

In this pass, each frame of the animation being rendered is submitted to the Grid as a different job. The frames are calculated now with the help of the merged irradiance cache.

Animation Name	Frames	Resolution	Rendered on	Average Computation Time per frame (hr:min)	Makespan (hr:min)		Speed-up comparing Two-pass and Single-Pass Approach
					Two-pass Approach	Single-pass Approach	
Kalabsha Walk-through	91	1024×768	Intel Xeon 3.0 Ghz	00:06	02:03	09:37	4.69
Artgallery	192	2048×1536	NGS	00:37	04:35	05:49	1.26
Corridor	361	2048×2048	NGS	03:43	05:21	11:41	2.19
Kalabsha Rotation	1441	2048×1536	NGS	00:09	01:48	02:06	1.16

Table 1: *Animation Description and Timings*

Newly created samples are still cached but are not shared. The resulting frames can be stored on a data server using the data service of the Grid or they can be sent back to the user's machine. Job failure is detected by the underlying Grid technologies and resubmitted. Once every frame has been computed, the animation is compiled.

4. Results

Both the single-pass and two-pass approach described in the previous section have been implemented and tested on the National Grid Service (NGS) [NGS] in the United Kingdom. The Radiance Light Simulation package [LS98] has been modified to adapt to the two-pass approach. The modified binaries were transferred to the nodes for rendering frames where the animations were rendered using 2 to 3 indirect diffuse bounces as suggested by McNamara [McN05] and high quality parameters for the other settings.

Approximately two hundred processors were used while rendering animations on the NGS Grid, which has a total of about a thousand processors at its four core sites. The number of processors used for rendering was restricted by the multi-user environment on the Grid.

4.1. Visual Quality

We use the Brightness Flickering Metric (BFM) [BFM] to compare the visual quality of animations generated by a single-pass approach and our two-pass approach. BFM is measured by calculating the modulus of difference of average brightness values between successive frames. The average BFM is a poor indicator of visual quality in comparing temporal noise since it considers the average brightness of the complete frame while the temporal noise is generally restricted to some areas of the animations. The flickering can be observed in the portions of the animations which are predominantly lit by indirect diffuse lighting. The BFM graphs shown in Figure 3 compare selected portions of the animations where the model is mainly illuminated by indirect diffuse lighting.

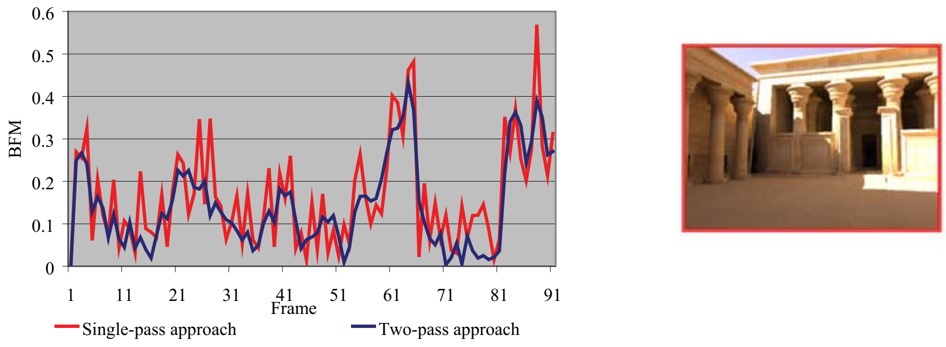
As can be seen from the BFM graphs in Figure 3, the change of brightness in the single-pass approach follows the trend of the change in brightness in animation rendered using the two-pass approach, but the BFM graphs of animations from the single-pass approach have more overshoots, indicating flickering. The coloured portion of the frame besides the graph shows the portion of the animation chosen for obtaining the BFM graphs. For the Kalabsha Walk-through animation (see Figure 3) most portions are lit by indirect diffuse lighting, hence the BFM graph is plotted for the complete animation.

4.2. Timing and Speed-up

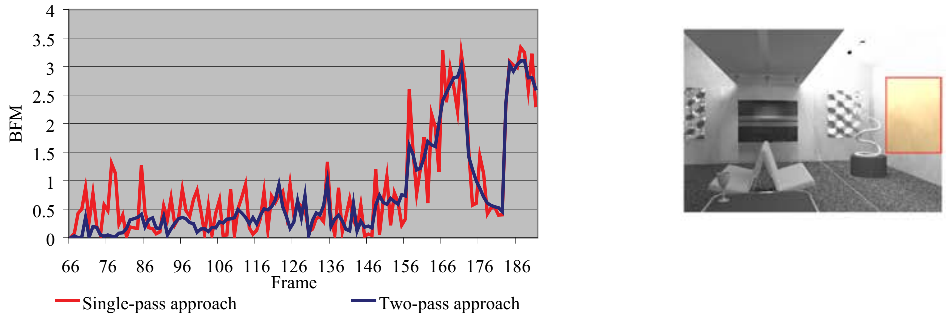
The timing results illustrating the speed-up of our two-pass approach over the single-pass approach along with the description of the animations have been summarised in Table 1. The makespan is the time taken from the moment a script is submitted to the Grid which submits the jobs till the time every frame has been rendered. Specifically, for the two-pass approach it includes the time taken for both the passes which are run successively on the Grid by a script which first submits jobs for the first pass, then checks for the completion of the first pass, merges the cache and finally launches jobs for the second pass.

We wanted to compare the timings while rendering on the Grid with the timings on a single processor, but the amount of time it takes to render the animation on a single processor makes this comparison infeasible. We present average computation time of a single frame as an overall indication. It can be seen from the Table 1, that we were able to render Kalabsha Rotation animation in less than two hours on the Grid, which is a high resolution animation with 1441 frames. It would have taken more than two hundred hours approximately (average computation time per frame multiplied by number of frames), if the animation was rendered on a single processor. This was achieved only because of the massive parallelism offered by the Grid. Similar results can be observed for the Art Gallery and Corridor animations.

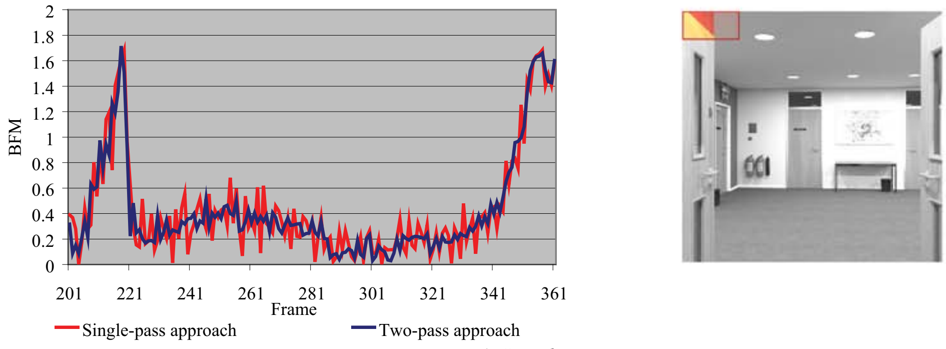
The speed-up comparing the two-pass approach and the single-pass approach on a Grid system is affected by many factors such as number of other users using the Grid services, the network load on the Grid interconnects, the number of nodes running on the Grid. Furthermore, communi-



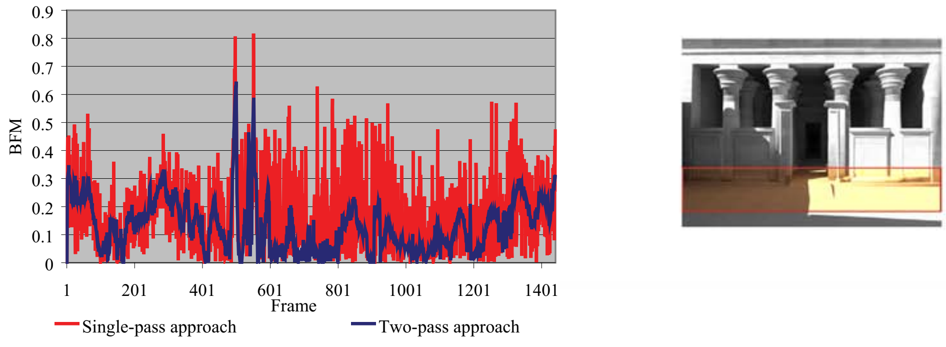
a) Kalabsha Walk-through



b) Artgallery



c) Corridor



d) Kalabsha Rotation

Figure 3: *The BFM Graphs: Coloured portion of the frame besides each graph indicates the part of animation chosen for calculating the BFM graph.*

cation costs on the Grid are high. These factors prevent us from measuring the complete speed-up achieved by computing animations using the two-pass approach over the single-pass approach. Hence, we present a result of a small animation rendered on a single Intel Xeon 3.0 GHz processor (see Table 1) as an indicator of speed-up of the two-pass approach over single-pass approach. This is obtained due to saving on recomputation time. Also, it is greater than the speed-up achieved on the Grid since additional factors are introduced while computing on the Grid. In particular, the communication overhead of job submission and the time taken by the Grid middleware are the major factors affecting the speed-up on the Grid.

Even though by using a two-pass approach we are increasing the overheads of job submission and communication, we have been able to achieve speed-up in all cases over the single-pass approach. This can be attributed mostly due to the saving of computation time by avoiding recomputation on different nodes of the Grid.

5. Conclusion and Future Work

In this paper, we have shown how the use of Grid computing can enable us to render high-fidelity animations in reasonable time. We have used a two-pass irradiance caching algorithm for indirect light calculations exploiting temporal coherence between animation frames. Our results indicate there is no visible temporal noise in animations rendered using two-pass approach, enhancing their visual quality. Using the two-pass approach, speed-up over the single-pass approach has been achieved despite the fact that there are more communication overheads on the Grid. This speed-up is attributed to significant time saving by avoiding recomputation of data.

Since no additional data structures have been used while adapting the irradiance cache algorithm for rendering of animations on the Grid, this approach can be applied to most renderers which use irradiance caching and other similar algorithms based on it (such as those mentioned in the Section 1).

As of now, communication costs on the Grid make it unsuitable for real-time rendering and small rendering jobs. But the Grid is a very useful resource for rendering high-fidelity and high quality animations for which the job computation time is high in comparison to the communication costs.

The future work will look at removal of bottleneck at the end of the first pass while merging the irradiance cache. It can be solved by distributing the merge process. Also the second pass can be launched for the part of the animation as soon as the computation of irradiance values for that part is completed.

The preliminary approach of rendering on the Grid provides a glimpse of the possible capabilities, but many challenges still remain in order to fully harness the power of Grid computing for rendering. The high cost of communication and the unpredictability of computing across the Grid may require careful data management, load balancing and the design of algorithms that can handle fault tolerance.

6. Acknowledgements

This work was supported by the UK-EPSRC grant EP/D069874/2. Thank you to the University of Bristol for the use of the Corridor and Kalabsha models.

References

[AFO05] ARIKAN O., FORSYTH D. A., O'BRIEN J. F.: Fast and detailed approximate global illumination by irradiance decomposition. In *Proceedings of ACM SIGGRAPH 2005* (2005), ACM Press.

[BBB*05] BERNHOLDT D., BHARATHI S., BROWN D., CHANCHIO K., CHEN M., CHERVENAK A., CINQUINI L., DRACH B., FOSTER I., FOX P., GARCIA J., KESSELMAN C., MARKEL R., MIDDLETON D., NEFEDOVA V., POUCHARD L., SHOSHANI A., SIM A., STRAND G., WILLIAMS D.: The earth system grid: Supporting the next generation of climate modeling research. *Proceedings of the IEEE 93*, 3 (March 2005), 485–495.

[BFM] Moscow State University Video Quality Tool/BFM Metric. `http://www.compression.ru/video/quality_measure/video_measurement_tool_en.html`.

[BP07] BENACCHIO L., PASIAN F. (Eds.):. *Grid-enabled Astrophysics* (2007).

[BSP06] BIGLER J., STEPHENS A., PARKER S.: Design for parallel interactive ray tracing systems. *Interactive Ray Tracing 2006, IEEE Symposium on* (Sept. 2006), 187–196.

[BWS03] BENTHIN C., WALD I., SLUSALLEK P.: A Scalable Approach to Interactive Global Illumination. *Computer Graphics Forum 22*, 3 (2003), 621–630. (Proceedings of Eurographics).

[CDR02] CHALMERS A., DAVIS T., REINHARD E. (Eds.): *Practical Parallel Rendering*. A. K. Peters, Ltd., Natick, MA, USA, 2002.

[CPC84] COOK R. L., PORTER T., CARPENTER L.: Distributed ray tracing. In *SIGGRAPH '84* (1984), ACM Press, pp. 137–145.

[CSL06] CHONG A., SOURIN A., LEVINSKI K.: Grid-based computer animation rendering. In *GRAPHITE '06* (2006), ACM, pp. 39–47.

[DCG*07] DEBATTISTA K., CHALMERS A., GILLIBRAND R., LONGHURST P., MASTOROPOULOU G., SUNDSTEDT V.: Parallel selective rendering of high-fidelity virtual environments. *Parallel Computing 33*, 6 (2007), 361–376.

[DD99] DAVIS T. A., DAVIS E. W.: Exploiting frame coherence with the temporal depth buffer in a distributed computing environment. In *PVGS '99: Proceedings of the 1999 IEEE symposium on Parallel visualization and graphics* (1999), pp. 29–38.

[Deb06] DEBATTISTA K.: *Selective Rendering for High-Fidelity Graphcs*. PhD Thesis, University of Bristol, 2006.

[DSC06] DEBATTISTA K., SANTOS L. P., CHALMERS A.: Accelerating the irradiance cache through parallel component-based rendering. In *EGPGV 2006* (May 2006), Eurographics, pp. 27–34.

[FK99] FOSTER I., KESSELMAN C. (Eds.): *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers Inc., 1999.

[FKNT02] FOSTER I., KESSELMAN C., NICK J., TUECKE S.: The physiology of the grid: An open grid services architecture for distributed systems integration, 2002.

[Fos06] FOSTER I. T.: Globus toolkit version 4: Software for service-oriented systems. *Journal of Computer Science and Technology 21*, 4 (2006), 513–520.

[GBP06] GAUTRON P., BOUATOUCH K., PATTANAIK S.: Temporal radiance caching. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Sketches* (2006), ACM, p. 171.

[GTGB84] GORAL C. M., TORRANCE K. E., GREENBERG D. P., BATTAILE B.: Modeling the interaction of light between diffuse surfaces. In *SIGGRAPH '84* (1984), ACM Press, pp. 213–222.

[GWS04] GUENTHER J., WALD I., SLUSALLEK P.: Realtime Caustics using Distributed Photon Mapping. In *EGSR'04* (2004).

[Her04] HERY C.: Rendering evolution at industrial light & magic. In *Rendering Techniques* (2004), pp. 19–22.

[JDZJ07] JAROSZ W., DONNER C., ZWICKER M., JENSEN H. W.: Radiance caching for participating media. In *SIGGRAPH '07: ACM SIGGRAPH 2007 sketches* (2007), ACM, p. 56.

[Jen01] JENSEN H. W.: *Realistic Image Synthesis Using Photon Mapping*. AK Peters, 2001.

[Kaj86] KAJIYA J. T.: The rendering equation. In *SIGGRAPH '86* (1986), ACM, pp. 143–150.

[KBPv06] KŘIVÁNEK J., BOUATOUCH K., PATTANAIK S. N., ŽÁRA J.: Making radiance and irradiance caching practical: Adaptive caching and neighbor clamping. In *Rendering Techniques 2006, Eurographics Symposium on Rendering* (Nicosia, Cyprus, June 2006), Eurographics Association, pp. 127–138.

[Kel97] KELLER A.: Instant radiosity. In *SIGGRAPH '97* (1997), ACM Press/Addison-Wesley Publishing Co., pp. 49–56.

[KGPB05] KŘIVÁNEK J., GAUTRON P., PATTANAIK S., BOUATOUCH K.: Radiance caching for efficient global illumination computation. *IEEE Transactions on Visualization and Computer Graphics 11*, 5 (September/October 2005).

[KK02] KOLLIG T., KELLER A.: Efficient multidimensional sampling. *Computer Graphics Forum 21*, 3 (2002), 557–563.

[KLC06] KENG S.-L., LEE W.-Y., CHUANG J.-H.: An efficient caching-based rendering of translucent materials. *The Visual Computer 23*, 1 (2006), 59–69.

[KMG99] KOHOLKA R., MAYER H., GOLLER A.: Mpi-parallelized radiance on sgi cow and smp. In *ParNum '99: Proceedings of the 4th International ACPC Conference* (1999), Springer-Verlag, pp. 549–558.

[LS98] LARSON G. W., SHAKESPEARE R.: *Rendering with Radiance: The Art and Science of Lighting Visualization*. Morgan Kaufmann Publishers Inc., 1998.

[McN05] MCNAMARA A. M.: Exploring perceptual equivalence between real and simulated imagery. In *APGV '05* (2005), ACM, pp. 123–128.

[Mes99] MESSINA P.: Distributed supercomputing applications. *The Grid: Blueprint for a New Computing Infrastructure* (1999), 55–73.

[NGS] National Grid Service, United Kingdom. `http://grid-support.ac.uk`.

[PKG*04] PEARLMAN L., KESSELMAN C., GULLAPALLI S., B. F. SPENCER J., FUTRELLE J., RICKER K., FOSTER I., HUBBARD P., SEVERANCE C.: Distributed hybrid earthquake engineering experiments: Experiences with a ground-shaking grid application. *High Performance Distributed Computing* (2004), 14–23.

[PMS*99] PARKER S., MARTIN W., SLOAN P.-P. J., SHIRLEY P., SMITS B., HANSEN C.: Interactive ray tracing. In *I3D '99: Proceedings of the 1999 symposium on Interactive 3D graphics* (1999), ACM, pp. 119–126.

[RCJ98] REINHARD E., CHALMERS A., JANSEN F. W.: *Overview of Parallel Photo-realistic Graphics*. Tech. rep., Bristol, UK, UK, 1998.

[RCLL99] ROBERTSON D., CAMPBELL K., LAU S., LIGOCKI T.: Parallelization of radiance for real time interactive lighting visualization walkthroughs. In *Supercomputing '99* (1999), ACM.

[SKDM05] SMYK M., KINUWAKI S., DURIKOVIC R., MYSZKOWSKI K.: Temporally coherent irradiance caching for high quality animation rendering. In *EUROGRAPHICS 2005* (2005), Alexa M., Marks J., (Eds.), vol. 24 of *Computer Graphics Forum*, Blackwell, pp. 401–412.

[SPCF*07] STEF-PRAUN, CLIFFORD T., FOSTER B., HASSON I., HATEGAN U., SMALL M., WILDE S. L., M. ZHAO Y.: Accelerating medical research using the swift workflow system. *Studies in Health Technologies and Informatics 126* (2007), 207–218.

[TL04] TABELLION E., LAMORLETTE A.: An approximate global illumination system for computer generated films. In *SIGGRAPH '04* (2004), ACM, pp. 469–476.

[TMD*04] TAWARA T., MYSZKOWSKI K., DMITRIEV K., HAVRAN V., DAMEZ C., SEIDEL H.-P.: Exploiting temporal coherence in global illumination. In *SCCG '04: Proceedings of the 20th Spring Conference on Computer graphics* (2004), ACM Press, pp. 23–33.

[WKB*02] WALD I., KOLLIG T., BENTHIN C., KELLER A., SLUSALLEK P.: Interactive Global Illumination using Fast Ray Tracing. In *13th EUROGRAPHICS Workshop on Rendering* (2002).

[WRC88] WARD G. J., RUBINSTEIN F. M., CLEAR R. D.: A ray tracing solution for diffuse interreflection. In *SIGGRAPH '88* (1988), ACM, pp. 85–92.

[WSBW01] WALD I., SLUSALLEK P., BENTHIN C., WAGNER M.: Interactive rendering with coherent ray tracing. In *Eurographics 2001 Proceedings*, vol. 20(3). Blackwell Publishing, 2001, pp. 153–164.

[YPG01] YEE H., PATTANAIK S., GREENBERG D.: Spatiotemporal sensitivity and Visual Attention for efficient rendering of dynamic Environments. In *ACM Transactions on Computer Graphics* (2001), vol. 20, pp. 39–65.