International Journal of Internet, Broadcasting and Communication Vol.14 No.5 30-35 (2022)
http://dx.doi.org/10.7236/IJIBC.2022.14.2.30

IJIBC 22-2-5

Version-Aware Cooperative Caching for Multi-Node Rendering

Kyungwoon Cho and Hyokyung Bahn

Department of Computer Engineering, Ewha University, Professor, Korea
bahn@ewha.ac.kr

Abstract

Rendering is widely used for visual effects in animations and movies. Although rendering is computing-intensive, we observe that it accompanies heavy I/O because of large input data. This becomes technical hurdles for multi-node rendering performed on public cloud nodes. To reduce the overhead of data transmission in multi-node rendering, this paper analyzes the characteristics of rendering workloads, and presents the cooperative caching scheme for multi-node rendering. Our caching scheme has the function of synchronization between original data in local storage and cached data in rendering nodes, and the cached data are shared between multiple rendering nodes. We perform measurement experiments in real system environments and show that the proposed cooperative caching scheme improves the conventional caching scheme used in the network file system by 27% on average.

Keywords: *Rendering, multi-node rendering, file system, cooperative caching, version-aware caching*

1. Introduction

Rendering is becoming increasingly important in animations and movies to generate high-resolution images [1, 2, 3]. Rendering is performed by dedicated software, which consists of long computation processes [3, 4]. Specifically, in high resolution rendering, generating a single image frame often takes a full day or more. So, rendering is usually executed in high-end server systems. Recently, multi-node rendering by making use of public cloud nodes is attracting attention as rendering of each frame is an independent task [5, 6]. Concurrent rendering on multiple nodes can speed up the total rendering process significantly.

Although rendering is computing-intensive, it uses large input data, which requires heavy I/O processes. Specifically rendering input data amount hundreds of GBs. Thus, if rendering is performed in cloud nodes, these large input data should be transmitted from local storage to the cloud rendering nodes first, which causes heavy network delay. Also, as rendering input data are updated frequently, this process should be performed continuously.

Caching can be an efficient solution to cope with the large overhead of rendering data transmission [7]. When caching is used, copies of delivered data are maintained and will be used again if the same data are requested [8, 9]. However, caching in multi-node rendering is different from traditional caching used in the

Manuscript Received: February. 9, 2022 / Revised: February. 13, 2022 / Accepted: February. 14, 2022
Corresponding Author: bahn@ewha.ac.kr
Tel: +82-2-3277-2368, Fax: +82-2-3277-2306
Professor, Department of Computer Engineering, Ewha University, Korea

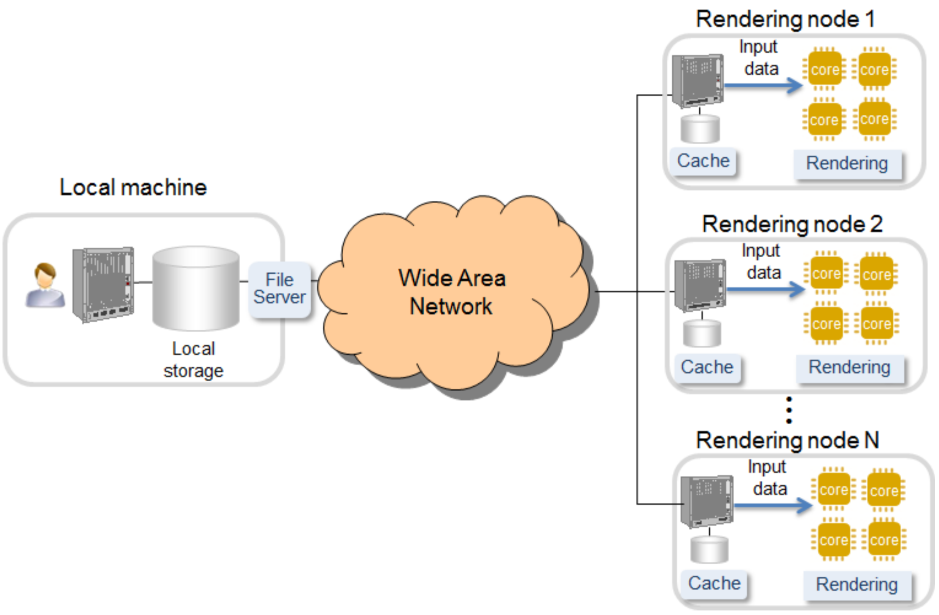


Figure 1. A multi-node rendering architecture for cooperative caching.

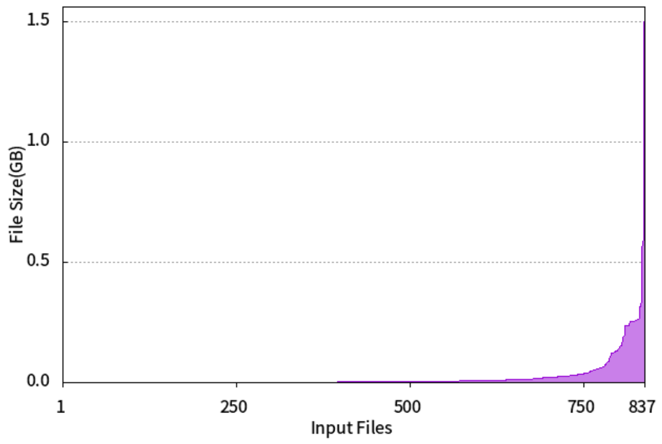
network file system [10], which performs block-based caching and does not support cooperative caching [11, 12]. Also, it is different from content caching or web caching that delivers the up-to-date version of data [16], as rendering should utilize the input data modified just before the current rendering process starts. This implies that cache consistency of rendering input data is more challenging than content caching as it depends on the start time of rendering.

In this paper, we analyze the characteristics of rendering input data, and propose a version-aware cooperative caching scheme for multi-node rendering. Our caching scheme is implemented on a user-level file system [13] as follows. In local storage, we have implemented a file server that manages versions of rendering input data and performs synchronization. For cloud rendering nodes, we have implemented the rendering cache module, which checks the versions of the input data and fetches data from multiple rendering nodes. This allows for the minimization of data transmission latency in multi-node rendering. Specifically, our rendering cache module fetches input data from one of the peer rendering nodes if a valid version exists, and caches the data with the version information for reuse of the data between rendering nodes. To validate the efficiency of the proposed cooperative caching scheme, we conducted measurement studies. Experimental results showed that the proposed cooperative caching scheme outperforms the caching performed by the network file system by 27% on average.

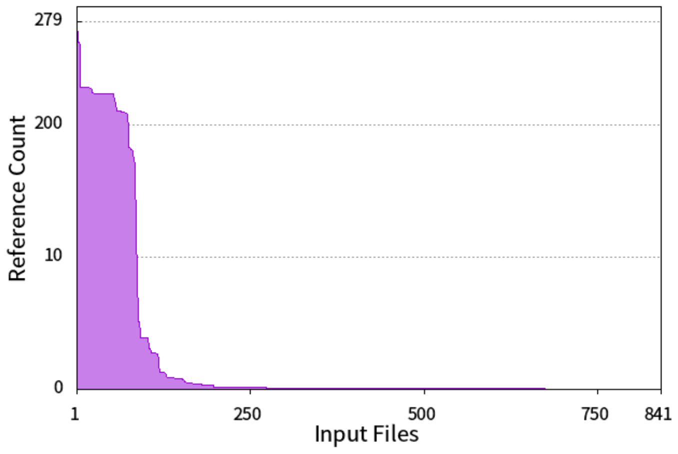
The remainder of this paper is organized as follows. Section 2 analyzes the workload characteristics of rendering input data and explains the proposed cooperative caching scheme for multi-node rendering. Section 3 describes the experimental results through measurement for assessing the effectiveness of the proposed caching scheme. Finally, we conclude this paper in Section 4.

2. The Cooperative Caching Scheme for Multi-Node Rendering

We analyze the rendering workload characteristics of open movie projects [14]. Figure 2(a) shows the distributions of rendering input data with respect to the file size sorted in an ascending order. As shown in the



(a) File size distribution (ascending order)



(b) Reference count distribution (descending order)

Figure 2. Distributions of rendering input data.

figure, most of input data are very small, but a limited number of large files exist, which are responsible for the most of the total input sizes. Figure 2(b) shows the distributions of rendering input data with respect to the reference count sorted in a descending order. As shown in the figure, the reference count of most input data is 1, but a limited number of hot input files exist, which are responsible for the most of the total reference count. That is, references in rendering input data are extremely skewed such that a certain limited data are responsible for most of reference counts and file sizes whereas a majority of data are referenced only once and their size is very small.

Based on this observation, we present a cooperative caching scheme for multi-node rendering. In our cooperative caching, local storage maintains the original rendering input data, and they are synchronized to rendering nodes when the rendering process starts. Since the local storage should check the modification of data before sending them, our file server at local storage manages rendering input data with their version information. In particular, a new version is created whenever input data are updated. The version information is snapshotted and delivered together to the rendering nodes when the rendering process starts. For data synchronization, instead of using the well-known `rsync` (remote sync) protocol, our caching scheme makes use of the version information of the data transferred. Note that `rsync` performs data synchronization to the

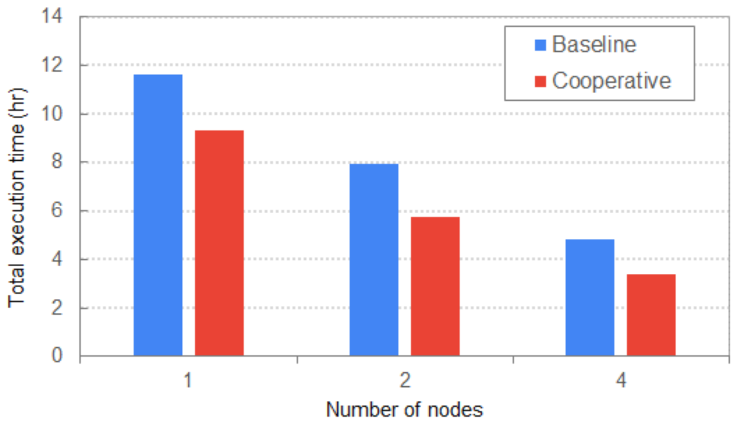


Figure 3. Comparison of cooperative caching and baseline as the number of nodes is varied.

up-to-date version by comparing the time stamp of the data, whereas our synchronization is just performed by checking the version information of the data transferred upon the beginning of the rendering.

To reduce the transmission overhead of rendering input data, a cache module is located at the rendering nodes. It will be a performance bottleneck if all rendering nodes get the input data from the local storage. Thus, our cache module performs cooperative caching between multiple rendering nodes, which supports modification-aware caching.

When rendering process begins and input data are necessary, our cache module checks if the data exist in the cache. If then, the cache module checks whether the current version is valid. If the cached version is valid, it is used. Otherwise, the cache module tries to fetch valid input data from peer rendering nodes. The cache module fetches the input data from local storage if it fails to get the valid input data from peer nodes. Then, the fetched input data are maintained in the cache with its version information. Note that our cache module maintains cache data in a file unit to make use of version information of rendering input data efficiently.

3. Experimental Results

To assess the efficiency of the proposed caching scheme, we conduct measurement studies under real system environments. We run the open source rendering software Blender on AWS with the open rendering project introduced in Section 2 [14, 15]. For each rendering node, we use Intel Xeon E5-2620 v4 2.1GHz Eight-Core with 4GB main memory. The size of the storage cache for each node is set to 20GB. The network bandwidth between local storage and rendering nodes is 1.8MB/s on average. For comparison, we additionally measure the performance of the baseline scheme, which uses the conventional caching used in the network file system.

Figure 3 compares the total execution time of the cooperative caching scheme and baseline as the number of rendering nodes increases. As we see in this figure, the execution time is reduced significantly as the number of rendering nodes increases. This is because rendering of each frame is an independent job, and thus concurrent executions can save the total execution time in proportion to the number of rendering nodes used. When comparing our caching scheme and baseline, our scheme outperforms baseline in all cases and the improved performance is 26% on average and up to 30%. Although baseline also has the caching ability, our caching scheme exhibits even better results as it performs cooperative caching among rendering nodes, which reduces the network transmission between local storage and rendering nodes connected by wide area networks. Also, our caching scheme saves the I/O operations in local storage, which may cause the performance

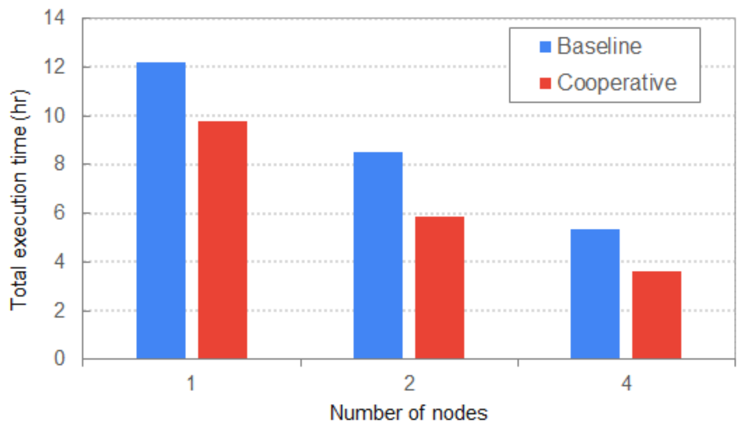


Figure 4. Comparison of cooperative caching and baseline as rendering input data are modified.

bottleneck in the overall rendering architecture. Another interesting observation is that the performance gap between the proposed scheme and baseline becomes large as the number of rendering nodes increases. This is because rendering nodes in our cooperative caching have more chances to retrieve the input data from peer nodes rather than the local storage in this case. It is also noteworthy that our cooperative caching performs better than the baseline scheme even though the number of nodes is one. This is because baseline performs instant per-block cache validation before the cached data are actually used, whereas our scheme does not perform such validations as it utilizes the version information of the files.

To see the effectiveness of the proposed caching scheme under more realistic rendering processes, we perform rendering multiple times as the input data are updated and measure the total execution time. Figure 4 shows the execution time of the proposed caching scheme compared to baseline. As we see in the figure, the proposed caching scheme outperforms baseline irrespective of the number of rendering nodes. The performance improvement is 28% on average and up to 33%. The performance improvement is slightly larger than the results of Figure 3, implying that our cooperative caching scheme is more effective as the input data are modified. This is because our caching scheme utilizes all cached data across multiple rendering nodes, which has the effect of reducing data transmission between local storage and rendering nodes further when the input data are modified.

4. Conclusion

Rendering is a computing-intensive process but we observed that it accompanies heavy I/O because of large input data, which becomes technical hurdles for multi-node rendering. In particular, sending large input data to rendering nodes increases the startup latency of rendering seriously. To reduce the overhead of data transmission in multi-node rendering, we analyzed the characteristics of rendering workloads, and presented the cooperative caching scheme for multi-node rendering. Our caching scheme has the function of synchronization between original data in local storage and the cached data in rendering nodes, and the cached data are shared between rendering nodes. We performed measurement experiments in real system environments and showed that the proposed cooperative caching scheme improves the conventional caching scheme of the network file system by 27% on average. As a future study, we plan to analyze the energy-saving effect of the proposed multi-node rendering with cooperative caching, which was not considered in this paper.

Acknowledgement

This work was partly supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2021-0-02068, Artificial Intelligence Innovation Hub) and the ICT R&D program of MSIT/IITP (2020-0-00121, development of data improvement and dataset correction technology based on data quality assessment). A proir version of this paper was presented at the 4th International Conference on Cloud Computing and Internet of Things, Tokyo, Japan, 2019.

References

[1] G.V. Patil and S.L. Deshpande, "Distributed rendering system for 3D animations with Blender," Proc. IEEE Conf. on Advances in Electronics, Communication and Computer Technology, pp. 91-98, 2016.

[2] D. Shin, K. Cho, and H. Bahn, "File type and access pattern aware buffer cache management for rendering systems," Electronics, vol. 9, no. 1, paper 164, 2020.
DOI: https://doi.org/10.3390/electronics9010164

[3] C. Rossl and L. Kobbelt, "Line-art rendering of 3D-models," Proc. IEEE Pacific Conf. on Computer Graphics and Applications, pp. 87-96, 2000.

[4] J. Annette, W. Banu, and P. Chandran. "Rendering-as-a-Service: Taxonomy and comparison," Procedia Computer Science, vol. 50, pp. 276-281, 2015.
DOI: https://doi.org/10.1016/j.procs.2015.04.048

[5] Microsoft Azure, https://azure.microsoft.com/

[6] Amazon Web Services, https://aws.amazon.com/

[7] H. Bahn, H. Lee, S. Noh, S. Min, K. Koh, "Replica-aware caching for web proxies," Computer Communications, vol. 25, no. 3, pp. 183-188, 2002.
DOI: https://doi.org/10.1016/S0140-3664(01)00365-6

[8] O. Kwon, H. Bahn, and K. Koh, "Popularity and prefix aware interval caching for multimedia streaming servers," Proc. IEEE CIT Conference, pp. 555-560, 2008.
DOI: http://doi.org/10.1109/CIT.2008.4594735

[9] T. Kim and H. Bahn, "Implementation of the Storage Manager for an IPTV Set-top Box," IEEE Trans. Consumer Electronics, Vol. 54, No. 4, pp.1770-1775, 2008.
DOI: http://doi.org/10.1109/TCE.2008.4711233

[10] B. Nowicki, "NFS: Network file system protocol specification," Technical Report RFC1094, 1989.

[11] M. D. Dahlin, R. Y. Wang, T. E. Anderson, and D. A. Patterson, "Cooperative caching: using remote client memory to improve file system performance," Proc. USENIX OSDI Conference, pp. 1-14, 1994.

[12] Y. Shin, H. Bahn, "A scalable web cache sharing scheme," Information Processing Letters, vol. 91, no. 5, pp. 227-232, 2004.
DOI: https://doi.org/10.1016/j.ipl.2004.05.009

[13] FUSE Filesystem in Userspace, http://fuse.sourceforge.net/

[14] Open Movie Projects, https://www.blender.org/about/projects/

[15] Blender, https://www.blender.org/

[16] A. Kabir, "Cooperative content caching and distribution in dense networks," KSII Trans. Internet and Information Systems. vol. 12, no. 11, 2018.
DOI: https://doi.org/10.3837/tiis.2018.11.009