A CACHE DESIGN METHOD FOR SPATIAL INFORMATION VISUALIZATION IN 3D REAL-TIME RENDERING ENGINE

Xuefeng Dai[a], Hanjiang Xiong[a], Xianwei Zheng[a]

[a] State Key Laboratory of Information Engineering in Surveying, Mapping and Remote Sensing, Wuhan University, 129 Luoyu Road, Wuhan, 430079, China – daixuefeng203@126.com

KEY WORDS: Memory cache, Disk cache, 3D Rendering Engine, Multi-thread, Replacement policy

ABSTRACT:

A well-designed cache system has positive impacts on the 3D real-time rendering engine. As the amount of visualization data getting larger, the effects become more obvious. They are the base of the 3D real-time rendering engine to smoothly browsing through the data, which is out of the core memory, or from the internet. In this article, a new kind of caches which are based on multi threads and large file are introduced. The memory cache consists of three parts, the rendering cache, the pre-rendering cache and the elimination cache. The rendering cache stores the data that is rendering in the engine; the data that is dispatched according to the position of the view point in the horizontal and vertical directions is stored in the pre-rendering cache; the data that is eliminated from the previous cache is stored in the eliminate cache and is going to write to the disk cache. Multi large files are used in the disk cache. When a disk cache file size reaches the limit length（128M is the top in the experiment）, no item will be eliminated from the file, but a new large cache file will be created. If the large file number is greater than the maximum number that is pre-set, the earliest file will be deleted from the disk. In this way, only one file is opened for writing and reading, and the rest are read-only so the disk cache can be used in a high asynchronous way. The size of the large file is limited in order to map to the core memory to save loading time. Multi-thread is used to update the cache data. The threads are used to load data to the rendering cache as soon as possible for rendering, to load data to the pre-rendering cache for rendering next few frames, and to load data to the elimination cache which is not necessary for the moment. In our experiment, two threads are designed. The first thread is to organize the memory cache according to the view point, and created two threads: the adding list and the deleting list, the adding list index the data that should be loaded to the pre-rendering cache immediately, the deleting list index the data that is no longer visible in the rendering scene and should be moved to the eliminate cache; the other thread is to move the data in the memory and disk cache according to the adding and the deleting list, and create the download requests when the data is indexed in the adding but cannot be found either in memory cache or disk cache, eliminate cache data is moved to the disk cache when the adding list and deleting are empty. The cache designed as described above in our experiment shows reliable and efficient, and the data loading time and files I/O time decreased sharply, especially when the rendering data getting larger.

1. INTRODUCTION

A well-designed cache system has positive impacts on the 3D real-time rendering engine, it can make the scene rendering smoothly, especially in the visualization of the mass geographic data. Data caching is an important technique for improving data availability and access latency [1]. A cache system can be complicated when considered with the real-time rendering engine, both memory and disk cache should be taken into account, and the replacement policy could differ. The main purpose of the cache system is to prepare the data that need most in the rendering engine.

The core of the cache system is the replacement policy. LRU[2] is one of the best-known replacement policies, This algorithm to choose the most long visit is not being replaced as a block by block, it takes into account the temporal locality[3] rather than spatial locality. O' Nell and others propose LRU-k[4],this algorithm to choose last but k visit is not being replaced as a block by block, actually, LRU is a special case of LRU-k when k equals 1.LRU-k has to store additional information, but how long it will be stored is not be solved well. Megiddo and others propose ARC[5]. This strategy use two LRU queues to manager the page cache, one queue is used to manage the pages which only be visited once, the other is used to manager the pages which are visited more than once, this strategy can adjust the size of the two queues according to the temporal or spatial locality. In [6] and [7] dynamic caching mechanisms are discussed, in [8–11] cooperative proxy caching is examined.

In the real-time rendering engine, cache replacement policy should be considered with the scene information. In our policy, each data item in the cache has a weigh value which is calculated dynamically, and the data item is replaced by the weigh value, it will be discussed in 2.1.4 and 2.2.2. Our cache system is consists of three parts, memory cache, disk cache and multi- threading mechanism.

2. CACHE SYSTEM

2.1 Memory cache

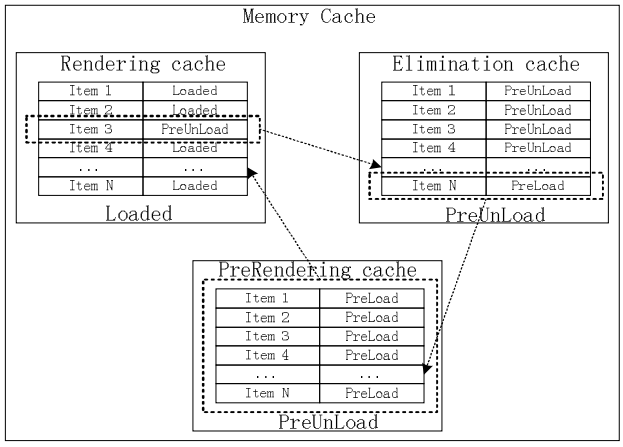

Figure 1. Data transfer in memory cache

2.1.1 Rendering Cache: The data stored in the rendering cache which has the status of Loaded should be copied to the video memory for the 3D scene rendering immediately. The status of the data items will be set to "PreUnLoad" after the copy is complete.

2.1.2 Pre-rendering Cache: The data items stored in the Pre-rendering cache should have the status of "PreLoad" and comes primarily from three sources. The first one is from the disk cache. This kind of data could have been used before, and should be reload to the memory cache from the disk cache; the second one is from the cache download thread. This kind of data is not existed in local machine, and should be run it off the server, new empty data items with status of "PreDownLoad" will be created and added to the download list, when the download is completed, data items should be decompressed and parsed as well as status resetting before loading to the Pre-rendering cache; the third one is from the elimination cache. The status of this kind of data is reset to "PreLoad" before writing to the disk cache by the cache dispatch thread, and should be transferred to Pre-rendering cache. In figure 1, data item 3 in the rendering cache has been loaded into the video memory, and its status has been set as "PreUnLoad", it will be transfer to the Elimination cache in the next few frames, the status of data item N in the Elimination cache has been set as "PreLoad", and it will be transfer to the Pre-Rendering cache, all the data item has the status "PreLoad" in the Pre-Rendering cache will be move to the Rendering cache after the Rendering cache is ready.

2.1.3 Elimination Cache: The data eliminated from the rendering cache is stored in the elimination cache which has the status of "PreUnLoad". Data in this cache will be written to the disk cache gradually if the status keeps unchanged. The data items in this cache should also be rapid retrieval because they may be reloaded to the Pre-rendering cache before written to the disk cache. The elimination cache should be locked when written data to the disk cache to control the possible conflicts between parallel threads. As we know, disk I/O operation is a very time-consuming process, if there is too much data to be written to the disk cache, the elimination cache will be locked for a long time, which may cause low efficiency of the rendering engine. Two different strategies have been introduced for this problem. The first one, control the number of the data item that written to the disk cache each time, we define a constant N as the maximum data item number, only N data items at top written to the disk cache each time, and the lock time can be controlled by the definition of N; the second one, instead of lock the whole elimination cache, only N data item is locked while the rest data still ready for retrieval or status resetting, the status of the locked data items will be set as "UnLoad".

2.1.4 Memory Cache Elimination Methods: There are two kinds of elimination in the memory cache. The first one, eliminate the data from rendering cache to the elimination cache, in the rendering engine, view-dependent frustum culling and clipping are performed, and an elimination list is created. The first kind of elimination is based on the elimination list. The status of the eliminated data will be set as "PreUnLoad", the eliminated data stay in the memory cache temporary and still has a chance to have the status reset in the next few frames; the second one, eliminate the data from the memory cache to the disk cache, this kind of elimination will erase the data items from the memory cache and written them into the disk cache files, and the status will be set as "UnLoad", the elimination is based on the weight of the data item which can be calculated by formula 1. In the formula, D means the distance between the data item geometry center and the view point, if the camera in the rendering engine moving in the horizontal direction, the D value should be recalculated; L means the difference value of the data item LOD level and scene LOD level, if the camera in the rendering engine moving in the vertical direction, L value should be recalculated; C means the render priority of different type of data, which is decided by rendering engine. The weight of the data item in the elimination cache should be recalculated each frame because the weight could change in every frame.

$$W = \frac{D + L}{M} + C \tag{1}$$

Where W = Elimination weight
D = View distance
L = Level difference
M = normalization parameter

2.2 Disk Cache

All the data eliminated from the memory cache is stored in the disk cache files, the status of all the data in disk cache is "UnLoad".

2.2.1 Index Files And Data Files: Disk cache consists of few index files and data files; there is a one-to-one relationship between the index file and data file. The index file records the index information of data items, which stored in the data file. Each data file has a constant size of S, which is 128M in our experiment system, so the data file can be loaded to the memory by memory mapping method in order to reduce I/O time consuming. A single data item should only exist in one of the data files. When eliminating data from the memory cache to the disk cache, the index file should be checked before written any data into the disk cache, if the data item already existed in the disk cache, the data item can be erased from the memory cache directly.

2.2.2 Disk Cache Eliminate Method: Since disk I/O operation is a very time-consuming process, data transfer in the disk cache should be avoided. In the experiment system, a single data item is not allowed to transfer or delete. Since disk space is much larger than the memory space, disk cache elimination occurs infrequently, so eliminate method is much simpler than the memory cache. When the disk cache elimination happens, the earliest data file and index file is deleted from the disk. Figure 2 shows how data stored in the disk cache, and data stored in the disk cache has the status of "UnLoad". Disk cache consists of

several block files, each block has an index file and a data file, and all the blocks have the same size.

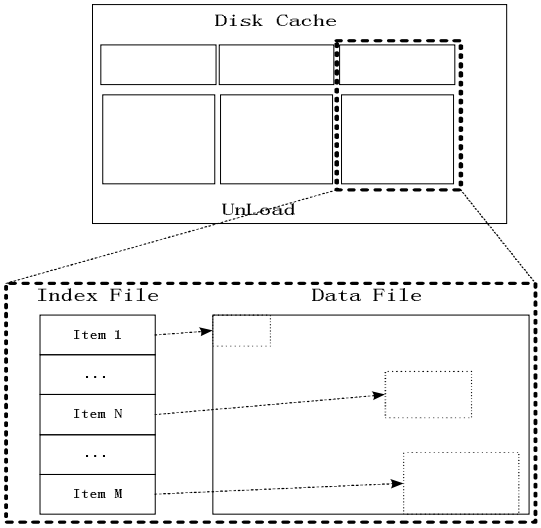

Figure 2. Data stored in disk cache

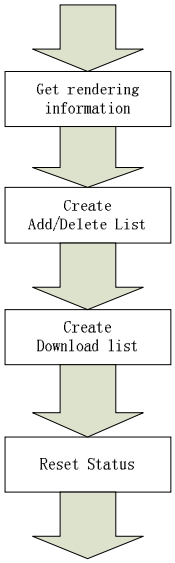

Figure 3. The Dispatch thread

2.3 Multi-threading mechanisM

The cache system is managed by multi- threading mechanism, all the threads, which managed the cache system are divided into three types in the experiment system: dispatch threads, data threads and download threads.

2.3.1 The Dispatch Thread: The main task of the dispatch thread is to reset the status of data items and create a download list by some logical calculations. Dispatch threads get rendering information from the rendering engine, reset data items' status for data threads and create a download list for download threads. The rendering information includes camera position, view-dependent frustum culling and clipping results and so on. In the figure 3,we can see how the dispatch thread works. In the first step, it gets rendering information from the real-time rendering engine, in the second step, according to the rendering information, the add and delete list can be create, the data item that no longer needed will be add to the delete list, while the data is going to be used in the next frames but not in the rendering cache is add to the add list, in the third step, if a data item in the add list cannot be found in the disk cache or memory cache, then it will be move to the download list, and wait for the download thread download, this step is not necessary if all the data items have been stored in the cache, in the last step, the status of all the data items recorded in the add, delete and download list will be reset.

2.3.2 The Data Thread: The main task of data thread is transferring the data items according to the status identification. In the memory cache, the data thread move data items whose status is "PreUnLoad" from rendering cache to the elimination cache; move data items whose status are "PreLoad" from the elimination cache to the Pre-rendering cache; move the data items whose status is "UnLoad" to the disk cache. In the disk cache, data thread deletes the data file and index file if they are eliminated by the disk cache; create a memory-mapped file from the data file if necessary. In figure 4, the data thread also has some steps to do, first step, according to the data status, move the data which has the status of "PreLoad" from PreLoading cache to the Rendering cache, in the second step, move the data which has the status of "PreUnLoad" from the Rendering cache to the Elimination cache, in the third step, the data items in the Elimination cache could have the chance to be reused, visit the Elimination cache, move data items which have the status of "PreLoad" to the Pre-Rendering cache, in the fourth step, if the Elimination cache is running out, data items will be move to the disk cache by the weight value (section 2.1.4), and the status will be set as "UnLoad", in the last step, if the disk cache is larger than pre-set, the disk cache file will be deleted according to the disk cache elimination method (section 2.2.2).

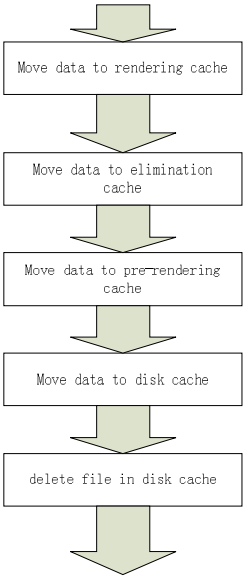

Figure 4. The data thread

2.3.3 The Download Thread

Download threads are managed by the thread pool. As shown in figure 5, download threads have two tasks; the first one is to download the data according to the download list, and the second one is decompressing or parsing the data, which is downloaded from the server.

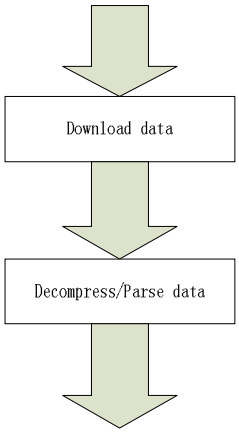

Figure 5. The download thread

3. EXPERIMENTS AND RESULTS

According to the strategy discussed above, an experiment cache system is implemented. Figure 6 shows the over view of the experiment system, the rendering engine includes serveral moduals as follows: "CommonObject" modual, "SystemObject" modual, "SceneObject" modual, "CameraObject" modual, "TerrainObject" modual, "AnnoObject" modual, and "Viewer" modual, the cache system get rendering information from the CameraObject modual, and prepare data for the SceneObject modual.

The experiment shows rendering engine based on this cache system can load data faster and view data smoother compare to the same engine based on the cache system which uses small disk cache files, especially as the cache data become larger and larger.

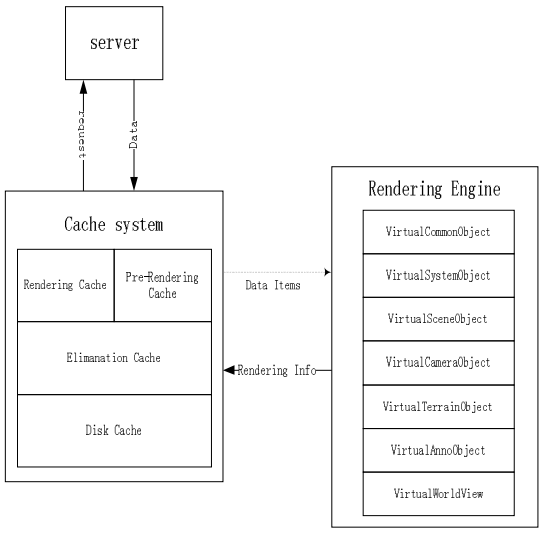

Figure 6. experiment system over view

4. CONCLUSIONS AND FUTURE WORK

This paper proposes a cache system tightly integrated in the real-time rendering engine. Take advantage of the scene rendering information, the cache system could precise control the data item status, prepare the data most needed. The results presents in section 3 confirm the effect replacement policy.

As future work, we plan make further improvement on the data download control and the optimization of thread policy

5. REFERENCES

[1]. M. J. Franklin, Caching and Memory Management in Client-Server Database Systems, Ph.d. Thesis, Dept. of Computer Science, University of Wisconsin, 1993.

[2]Mattson R L, Gecsei J, Slutz D R. Evaluation techniques for storage hierarchies. IBM System Journal, 9(2), pp. 78-117.

[3]Denning P J The working set model for program behavior. Communications of the ACM, 1l(5), pp. 323-333.

[4]O'Neil E J. O'Neil P E，Weikum G．The LRU-K page replacement algorithm for database disk buffering [C] Proc of the ACM SIGMOD 1993．New York：ACM, pp. 1-10.

[5] Megiddo N, Modha D S. ARC: A self-tuning,low overhead replacement cache[c] Proc of the 2nd USENIX Corf on File and Storage Technology．San Francisco, CA: USENlX, pp. 115-130

[6]. Wu, K.-L., Yu, P.S., Wolf, J.L.: Segment-based proxy caching of multimedia streams. In: Proceedings of the 2001 WWW Conference, Hong Kong, pages 36–44. An extended version titled Segmentation of multimedia streams for proxy caching will appear in IEEE Transactions on Multimedia.

[7]. Wu, K.-L., Yu, P.S., Wolf, J.L.: Segmentation of multimedia streams for proxy caching. IEEE Trans. Multimedia. **6**(5), pp. 770–780.

[8]. Acharya, S., Smith, B.C.: Middleman: A video caching proxy server. In: Proceedings of the NOSSDAV'00.

[9]. Paknicar, S., Kankanhalli, M., Ramakrishnan, K.R., Srinivasan, S.H.,Ngoh, L.H.:Acaching and streaming framework for multimedia. In: Proceedings of ACMMultimedia, California, pp. 13–20.

[10]. Chae, Y., Guo, K., Buddhikot, M.M., Suri, S., Zegura, E.W Silo, rainbow, and caching token: schemes for scalable, fault tolerant stream caching. IEEE J. Select. Areas Comm. 20(7),1328–1344 (2002)

[11]. Tewari, R., Dahlin, M., Vin, H., Kay, J.: Design considerations for distributed caching on the Internet. In: Proceedings of the 19th IEEE International Conference on Distributed Computing Systems.