Proceedings of the 2016 Winter Simulation Conference
T. M. K. Roeder, P. I. Frazier, R. Szechtman, E. Zhou, T. Huschka, and S. E. Chick, eds.

A QUANTITATIVE ANALYSIS OF LOCAL DATA MANAGEMENT FOR 3D CONTENT STREAMING

Elvis S. Liu
Aditi Rungta

School of Computer Science and Engineering
Nanyang Technological University
SINGAPORE

ABSTRACT

Content streaming is a mechanism used to distribute static geometry data to users of a virtual environment in real-time. Although most existing content streaming mechanisms have shown to meet their run-time performance requirements, they have an underlying constraint on the quality of 3D data that can be streamed. This restriction stems from the fact that most existing content streaming mechanisms require the server to transmit the same content every time the user encounters it. As a result, unnecessary bandwidth consumption would be induced. In this paper, we present a caching approach termed as the Local Data Management (LDM) framework to reduce bandwidth usage by maintaining local copies of geometry data in the client system. We also evaluate the performance of the LDM approach in conjunction with multiple multiresolution interest matching algorithms. Our goal is to compare the cost of strategies studied under various system conditions and scenarios.

1 INTRODUCTION

The increasing popularity of large-scale distributed virtual environments (DVEs) and massively multi-player online games (MMORGs) in the recent years, and their growing inclination towards user-generated content has necessitated the need for real-time streaming mechanisms for 3D models. The dynamic nature of the graphical content in virtual worlds which allow user-generated content, along with the ever-increasing character of such virtual worlds, makes it impractical to expect users to pre-install the latest content before entering the virtual world. Hence, it has become imperative to employ real-time content streaming mechanisms, or in other words to distribute static data along with dynamic data in real-time, in such situations. Moreover, it is important to note that given the scale of these virtual worlds, it is understandable that the users would not be of interest to the complete content of the virtual world at any moment in the simulation. This entails the use of interest management techniques (Liu and Theodoropoulos 2014) to identify the fraction of static content which would be of interest to any particular user. The use of content streaming with data filtering is, therefore, indispensable in large-scale virtual environments in order to enable real-time user interactions and eliminate the trouble of pre-installations and offline patch updates.

Throughout the years, a number of studies have been published on how to use fundamental features of client-server and peer-to-peer networks to the advantage of content streaming, in order to stream 3D content with an acceptable frame rate. However, the quality of 3D content in most existing content streaming mechanisms is limited, which is due to the fact that every frame is streamed from scratch. Therefore, it would be desirable for the clients to locally maintain a copy of most recent or most frequently encountered 3D content from a virtual world, and reuse them during the rendering of future frames. This mechanism is termed as local data management (LDM). By using the LDM, streaming performance would not only benefit with respect to improved frame rate but also in terms of reduced bandwidth consumption as objects need not be transmitted again during later re-visits.

The bulk of study done so far in this field has been focused on the performance of frameworks for streaming 3D content with specific data filtering mechanisms. This gives rise to the need for a universal framework for 3D content streaming which can be used in any large-scale distributed virtual environment irrespective of their visibility calculation and data filtering mechanisms. The LDM framework aims to bridge this gap by presenting a generic solution for streaming 3D content from large scale DVEs in real-time.

This paper presents an evaluation and comparison of the performance of this mechanism when used with different interest management algorithms. We base our study on the client-server model since it is the most common system model for MMOGs. We analyze the server-side computational cost (for calculating each user's visibility and determining the data set to be transmitted) and the server-side bandwidth consumption (in sharing the required 3D content with the client users). We focus our analysis on three widely used interest matching schemes, namely, aura-based, zone-based and zone-aura hybrid schemes, and present their corresponding multiresolution versions, which we refer to as multi-aura visibility tests. This paper also aims to understand the trade-offs between the server-side computational and transmission load and the client-side usage of local disk space.

The rest of the paper is organized as follows: Section 2 reviews related work in content streaming. Section 3 gives an overview of the design of the LDM framework. Section 4 delves into the nuances of the implementation of the framework with respect to the three interest matching algorithms being examined. Section 5 presents a quantitative evaluation of the performance and behavior of the LDM framework for the different interest matching algorithms. Finally, Section 6 concludes the paper with a discussion on the possible direction of future work.

2 RELATED WORK

Schmalstieg and Gervautz (1996) introduced the concept of demand-driven geometry transmission in distributed virtual environments in as early as 1996. The work focused on gaining significant savings on network bandwidth by employing a strategy that combined techniques including levels of detail, progressive refinement, and graceful degradation to enable just-in-time or real-time rendering in a client-server model. It also mentioned the use of the client's cache to maintain geometry data as long as they lie within the latter's area of interest. This forced the server to stream frequently encountered geometry data repeatedly.

Cyberwalk (Chim et al. 2003) used an on-demand transmission approach with a multiresolution caching mechanism to enable virtual walkthroughs in its web-based distributed system. The object models were cached in the local storage until space was exhausted. When the storage was exhausted, existing objects were replaced. Visibility calculations were performed by an interest matching algorithm based on overlaps between circular viewer and object scopes. However, the Cyberwalk model was simulated and evaluated for a virtual environment with static objects only. The FLoD framework (Hu et al. 2008) proposed the use of peer-to-peer networks for 3D streaming and introduced the possibility of content exchange between peers with similar interest regions in the virtual world. The geometry data was stored in the client's cache upon download and the view was rendered from the cache based on the location, orientation and scale of each object. The cache size was selected such that it stored approximately three times the expected amount of data within an area of interest (AOI). The model was evaluated for varying cache sizes with the maximum cache size set to 400KB.

Wong and Muntz (2000) addressed the content streaming problem in an interactive virtual world as a scheduling problem, with the focus on providing guaranteed on-time data delivery and sustaining a certain level of quality of service. The desired level of perceived quality in terms of frame rate and picture quality was ensured by having the required data arrive at the client before it falls in the client's view. Chu, Chan, and Wu (2008) suggested an incremental transmission approach of a 3D model from a coarse representation to a finer one by transmitting only the mesh difference between two level-of-detail (LoD) models. The approach rendered the streaming data in real-time upon the arrival of each LoD data, using multi-thread programming techniques. The massively multi-user virtual game environment of Second Life uses 3D

streaming techniques to allow real-time user interactions by progressively downloading and rendering game scenes. However, details related to the game's content streaming mechanism are unavailable.

This paper complements the existing work on content streaming with caching by quantitatively analyzing the impact of local caching on the computational and network resource usage at the server's end, for various interest management algorithms. The study also aims to understand the impact of the size of the virtual world in terms of object count and the frequency of local disk cleanups on the server's resource usage and the client's local space usage.

The LDM framework has several advantages over previous approaches to content streaming. Firstly, the caching mechanism used is different from the usual caching approaches, as instead of imposing a hard limit on the cache size, it adopts a flexible approach where disk space utilization is controlled by periodic cleanups of obsolete geometry data. Secondly, the importance of a virtual object from the perspective of any given viewer is based on the object's viewing distance. This is reflected in the objects being rendered at the lowest optimal resolution for the acceptable visual quality at the client's system. Finally, it works well in case of both static and dynamic virtual objects, allowing 3D content streaming of large-scale DVEs in real-time.

3 LOCAL DATA MANAGEMENT FRAMEWORK

In our framework, we consider a large DVE with 3D objects of varied shapes and sizes, where each user navigates through remote walkthrough (Chim et al. 2003). Each object is defined by polygonal meshes at three LoDs, with the highest resolution having approximately 3000 polygons and the lowest being a bitmap image. Object definitions also include the texture, light map, animation, and audio data associated with the polygonal meshes. This multi-resolution storage of virtual objects allows the server to transmit geometry data of optimal visual quality, thereby saving network bandwidth from transmitting unnecessary visual details for distant objects. The resolution of an object as viewed by a user at any instant of time is determined based on the viewing distance. The virtual world objects include both static objects as well as moving objects. Also, in order to enable the models of these virtual objects to be frequently updated by the users or the DVE administrators, a version number is assigned to each object.

All virtual world content is maintained at the server, which transmits it to the client users on-demand. Since the locations of the objects change dynamically in the DVE, the server also needs to transmit position data with the geometry data for these objects. For simplicity and uniformity, we have the server send position data along with geometry data for all static and dynamic objects. At the start of the walkthrough, the server sends data pertaining to all objects within the user's AOI to the user's system. The depth of vision for the user can be selected by the user or the system, depending on the application. The client systems use their local disk space to maintain local copies of the 3D models streamed by the server. This local caching mechanism lowers transmission delays and reduces network dependency. The server also maintains records of the DVE objects locally available at each client. After performing visibility computations, the server compares the list of objects currently visible with the list of objects locally available to determine the objects to be transmitted. Periodic cleanups of rarely encountered objects on each client's local disk are carried out to keep a check on the disk space used. The intervals at which the cleanup is carried out must be chosen carefully, as it impacts the amount of space occupied on each client's local disk by the accumulated geometry data. Frequent cleanups would defeat the purpose of local caching whereas large gaps between consecutive cleanups could result in space overflows. Also, it is important to note that the size of a user's AOI impacts the maximum attainable local disk usage on the client's system. Clients with very large areas of interest would stream huge volumes of geometry data as they walk around the virtual space eventually causing the maximum attainable local disk usage to exceed the available disk space. We refer to the mechanism of maintaining local copies of 3D models to enhance content streaming for DVEs as LDM. The framework's process loop running on the server can be summarized by Algorithm 1.

The fundamental tasks to be carried out by the LDM framework can, therefore, be listed as follows:

Algorithm 1: Server-side process loop.

```
while true do
    loop ← loop + 1;
    Read position update from every client A_i;
    L_vis ← VisibilityCalculation();
    L_stream ← CompareObjects(L_vis , L_loc);
    foreach (A_i, o_k) in L_stream do
        Stream object o_k to client A_i;
    end
    if loop%ctime = 0 then
        LocalDiskCleanup();
    end
end
```

i. **Visibility Calculation:** Identifying all viewer-object pairs (A,o), where object o is visible to viewer A.

ii. **Object Set Comparison:** Computing a list of viewer-object pairs to be streamed L_{stream} by comparing a list of visible viewer-object pairs L_{vis} , with list of locally cached viewer-object pairs L_{loc} $\forall$ $(A_i,o_k) \in L_{vis}$.

iii. **Content Streaming and Tracking:** After streaming object o_k to viewer A_i, $\forall$ $(A_i,o_k) \in L_{stream}$, update L_{loc} to reflect the recently streamed objects. New viewer-object pairs are added to L_{loc} and existing pairs in L_{loc} are updated based on the object's streamed version and resolution. This step also maintains a count f reflecting the frequency of encounters for every viewer-object pair in L_{loc}.

iv. **Periodic Local Disk Cleanup:** Based on the count f for every viewer-object pair (A_i,o_k) in L_{loc}, least frequently encountering viewer-object pairs (i.e., pairs with very low f value) are removed from L_{loc} and client A_i is notified to delete the locally cached geometry and position data for object o_k.

During the simulation, the users may frequently update the server on their current location. The server is responsible for performing simulations, updating the virtual world states, and performing visibility computations for each user. After determining the set of visibility objects, $L_{vis} = \{(A,o,v,r) : \text{object } o \text{ is visible to } A \text{ at resolution } r \text{ and version } v\}$, for every viewer A_i the server compares it with the set of locally available objects, $L_{loc} = \{(A,o,v,r) : \text{version } v \text{ and resolution } r \text{ of object } o \text{ is locally cached at } A\}$, and streams the required new content. New content is required to be streamed only in the three situations listed below.

- For $(A_k,o_k,v_k,r_k) \in L_{vis}$, $\nexists\ (A,o,v,r) \in L_{loc} : A = A_k,\ o = o_k$ (i.e., object is not locally available)
- For $(A_k,o_k,v_k,r_k) \in L_{vis}$, $\exists (A,o,v,r) \in L_{loc} : A = A_k,\ o = o_k,\ v < v_k$ (i.e., older version of object is available)
- For $(A_k,o_k,v_k,r_k) \in L_{vis}$, $\exists (A,o,v,r) \in L_{loc} : A = A_k,\ o = o_k,\ v = v_k,\ r < r_k$ (i.e., lower resolution of object is available)

When an older version or lower resolution is available, they are overwritten by the latest geometry data, therefore if $(A,o_m,v_m,r_m) \in L_{loc}$ and $(A,o_n,v_n,r_n) \in L_{loc}$, then $o_m = o_n$, $v_m = v_n$ and $r_m = r_n$ must always hold. Since the server internally maintains a dynamic record of the locally available objects at each client, it also updates its records before moving to the next time step. It is also important to note that in scenarios when a vast majority of the objects are frequently re-encountered or a vast majority of the objects are rarely re-encountered (both with similar frequencies i.e., count f for most pairs (A_i,o_k) in L_{loc} is the same),

the periodic local disk cleanup process tends to become the removal of a randomly selected viewer-object pair rather than the least frequently encountered pair. Therefore, in such scenarios, the framework fails to benefit much from the periodic local disk cleanup process. Moreover, in some cases, the periodic local disk cleanup process could have a negative impact on the performance of the framework, by increasing the amount of content to be streamed each time.

The prime focus of the LDM framework is to avoid streaming redundancy by maintaining local copies of frequently encountered virtual objects for every client. As a result, the bandwidth usage of streaming duplicate 3D models can be reduced. However, as the clients constantly receive the missing content from the server, the geometry data accumulated on the local disk can become too large in volume and often obsolete with time. Obsolete data could include older versions of DVE objects as well as objects that have not been recently encountered by the user. To solve this problem, the proposed LDM framework carries out a periodic clean-up process to purge such obsolete content. This process enables the clients to obtain up-to-date content from the server and reduces the consumption of the local storage space.

4 MULTI-AURA VISIBILITY FILTERING

In this section, we describe the details of the visibility filtering approaches used in the LDM framework, namely, aura-based, zone-based, and zone-aura hybrid schemes.

LoD is the concept used in computer graphics, which involves decreasing the complexity of a 3D object representation based on parameters like viewer's distance from the object. The content streaming mechanism where the server determines and transmits visible 3D objects at varied LoDs to clients was first introduced by Schmalstieg and Gervautz (1996). We allow each DVE entity to be represented at three levels of details and enable the server to transmit their 3D models based on visual quality estimates. The visual quality is estimated by adopting a focus and nimbus approach similar to that introduced in the MASSIVE system (Greenhalgh and Benford 1995).

We define the viewability scope of each object proportional to the object's size. Specifically, large objects have a large viewability scope and small objects have a small viewability scope. In this manner, a viewability scope, N_o, is defined for each virtual world object. This allows large objects to be streamed to the client user well in advance of becoming visible and being rendered. Next, the viewing scope is defined for every user of the virtual world. The bigger is the viewing scope of a user the longer is the depth of their sight. The size of the viewing scope may vary for different users based on their virtual world profiles. In order to enable multiresolution filtering at the three available level of details, each user has their visibility limited to three concentric viewing scopes of varying sizes, as seen in Figure 1. Hence, each user has three viewing scopes, namely, F_1, F_2 and F_3 and one viewability scope, N_v, associated with them. The smallest viewing scope identifies objects closest to the users (i.e., objects that need to be rendered in their highest visual quality), and vice versa. Therefore, the resolution or visual quality of the object is directly related to the visual importance of the object, which in turn is based on the distance between the object and the viewer. It is important to note that here by degrading the visual quality of the object we are speeding up the rendering process without compromising on the perceptual impact. Similar to the viewability scope of an object, the viewability scope of the user defines how far they can be seen by another user in the virtual world. As the users walkthrough, the virtual world they send updates on their position coordinates to the server, which in turn determines their updated area of visibility based on the new positions.

The server performs visibility calculations and object set comparisons to identify the set of geometry data to be streamed to a client. Visibility calculations are carried out using interest matching mechanisms based on a scalable sort-based interest matching algorithm (Liu, Yip, and Yu 2005; Liu, Yip, and Yu 2006). Unlike the commonly used circular auras or AOIs, we use rectangular AOIs to represent our user's interests, in order to harness the superior performance of the scalable interest matching algorithm.

In the case of universal multi-aura visibility tests or in other words, multiresolution aura-based interest matching, each user is associated with four axis-aligned bounding boxes in increasing order of size representing its viewability scope N and three multiresolution viewing scopes or multi-auras, F_1,F_2 and F_3,

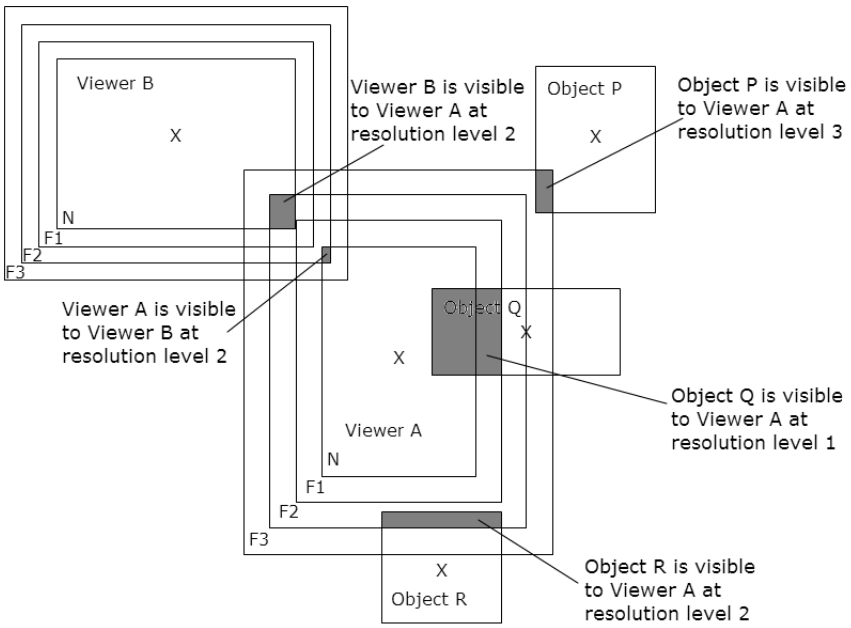


Figure 1: Multi-aura axis-aligned AOIs.

respectively. The viewing scope represents the observer's AOI, whereas the viewability scope describes the observed object's exposure. Each object, on the other hand, has only one viewability scope associated with it. Unlike existing aura-based schemes (Benford and Fahlén 1993; Greenhalgh and Benford 1995; Carlsson and Hagsand 1993), we use multiple levels of AOIs or viewing scopes for each user. The three viewing scopes defined for each user enable multiresolution filtering by allowing the matching process to identify the user's level of awareness for any given object and determine the corresponding resolution of the object model to be rendered. The objects overlapping the user's smallest viewing scope are rendered using their highest quality model, and vice versa. The aura-based interest matching process is captured well in Algorithm 2.

Algorithm 2: Universal multi-aura visibility test.

Function *UMAVisibilityCalculation()*

foreach *user* A_i *in virtual_world* **do**
update $A_i.viewing\_scope$ and $A_i.viewability\_scope$;
end
foreach *moving virtual object* C_i *in virtual_world* **do**
update $C_i.viewing\_scope$ and $C_i.viewability\_scope$;
end
pair_list ← `InterestMatching() ; // works on viewing and viewability scopes`
foreach *viewing and viewability scope pair* (box_1, box_2) *in pair_list* **do**
$A_k \leftarrow box_1.viewer$;
$o_k \leftarrow box_2.object$;
$o_k.resolution \leftarrow box_1.index$; add ($A_k$, o_k) to L_{vis};
end

In zonal multi-aura visibility tests, we perform multiresolution zone-based interest management by using axis-aligned bounding boxes to represent the user's area of interest (AOI) as well as the virtual world zones. This new approach to zone-based interest matching ensures that the zones are seamless allowing the AOI of an object to overlap with more than one zone at an instant. As a result, the sweep and prune algorithm offers not only a fast but also very smooth approach to zone-based interest matching. During remote walkthroughs, the server calculates the currently overlapping zones of every moving DVE object and transmits full zone geometry data to all zone members, as shown in the Algorithm 3. The sweep and prune algorithm makes this task fairly simple which keeps the computational overhead on the server to a minimum in this approach. Also, users experience better migration transparency, without any lag hampering their illusion of presence of virtual reality, in DVEs using seamless zones.

Algorithm 3: Zonal multi-aura visibility test.

Function *ZMAVisibilityCalculation()*

foreach *user* A_i *in virtual_world* **do**
  $A_i.zone \leftarrow$ `FindCurrentZone(`A_i`);`
  update $A_i.viewing\_scope$ and $A_i.viewability\_scope$;
end
foreach *moving virtual object* C_i *in virtual_world* **do**
  $C_i.zone \leftarrow$ `FindCurrentZone(`C_i`);`
  update $C_i.viewing\_scope$ and $C_i.viewability\_scope$;
end
pair_list $\leftarrow$ `InterestMatching() ; // works on zone and viewing bounding boxes`
foreach *zone and viewability scope pair* (box_1, box_2) *in pair_list* **do**
  foreach o_k *in* $box_1.zone$ **do**
    $A_i \leftarrow box_2.viewer$;
    add (A_i, o_k) to L_{vis};
    $o_k.resolution \leftarrow box_2.index$;
  end
end

The hybrid multi-aura scheme or the zone-aura hybrid approach is very similar to the aura-based approach as it also utilizes three viewing scopes along with a viewability scope to achieve multi-resolution filtering. It differs from the pure aura-based approach as it restricts the aura overlap tests to the user's current zone only. Also, unlike the pure zone-based approach the zone-hybrid approach uses disjoint zones allowing each virtual object to belong to only one zone at a time. As the users navigate through the DVE and update the server with their current positional status, the server determines the user's current zone and consequently, performs aura overlap tests within that zone to calculate the user's visibility. Algorithm 4 summarizes the zone-aura hybrid visibility calculations.

We demonstrate the performance of the LDM framework by implementing a prototype and simulating a remote walkthrough environment. This remote walkthrough environment comprises of a massive virtual world with static and moving objects and users. Each virtual world object is represented by three sets of geometry data of varying polygonal meshes complexities. Although the virtual world comprises of 3D content, for simplicity the terrain of the world is considered to be flat. The object models can be updated at any given time with a newer version for all three sets of representations. The version of the object is updated on the local disk of the users only if the object becomes relevant to the user again in the virtual world. This avoids unnecessary streaming of irrelevant content each time an object undergoes version update. The periodic cleanup of client's local disks is carried out at regular intervals based on time steps

Algorithm 4: Hybrid multi-aura visibility test.

Function *HMAVisibilityCalculation()*

foreach *user* A_i *in virtual_world* **do**
  $A_i.zone \leftarrow$ `FindCurrentZone(`A_i`)`;
  update $A_i.viewing\_scope$ and $A_i.viewability\_scope$;
end
foreach *moving virtual object* C_i *in virtual_world* **do**
  $C_i.zone \leftarrow$ `FindCurrentZone(`C_i`)`;
  update $C_i.viewing\_scope$ and $C_i.viewability\_scope$;
end
foreach *zone in virtual_world* **do**
  pair_list $\leftarrow$ `InterestMatching()` ;
  foreach *viewing and viewability scope pair* (box_1, box_2) *in pair_list* **do**
    $A_k \leftarrow box_1.viewer$;
    $o_k \leftarrow box_2.object$;
    $o_k.resolution \leftarrow box_1.index$; add ($A_k$, o_k) to L_{vis};
  end
end

and the objects that have been encountered for less than three times till then, are purged. Geometry data for virtual objects currently within the viewer's area of interest are not deleted irrespective of their encounter frequency, during the periodic cleanup.

5 EVALUATION

We conducted extensive simulations and evaluations to quantitatively analyze the performance of the LDM framework with respect to different interest management algorithms. The aim of the simulation was to understand the behavior of the LDM framework in various scenarios. In this section, we present the experimental results to illustrate the general behavior of the LDM framework with universal, zonal and hybrid multi-aura visibility filtering schemes, followed by a more detailed analysis of the performance of each scheme under diverse situations.

5.1 Performance Metrics

We utilize the following performance metrics in order to evaluate the LDM framework.

i. **Computational Effort:** The efficiency of the DVE system is dependent on its available computational resources. This entails ensuring that the processing power required at the server does not culminate into a potential bottleneck. We measure the computational effort in terms of the average execution time required per simulation time-step, by the LDM framework to identify the virtual objects to be streamed.

ii. **Bandwidth Usage:** For a large-scale DVE to be scalable it is crucial that the bandwidth requirements at its nodes do not exceed their respective capacity. Therefore, we evaluate the bandwidth usage of our framework to estimate its scalability.

iii. **Disk Space Utilization:** The LDM framework largely depends on the local disk space of client systems, purging geometry data periodically to limit the space used. The disk space utilization metric is used to observe the upper bound on the disk space used in the case of LDM frameworks with periodic cleanups.

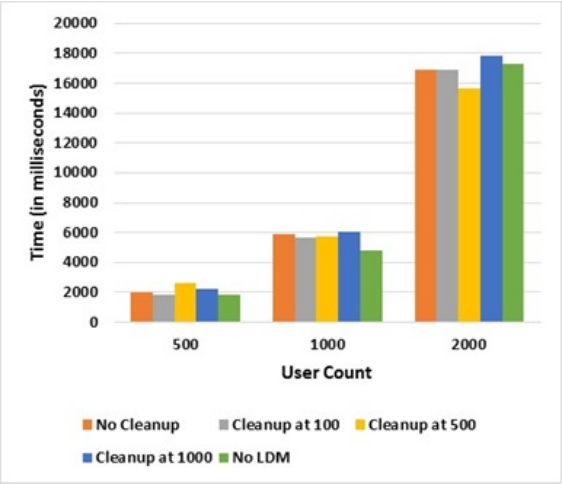
5.2 Experimental Settings

We focus on a single server and multiple clients network in this study. The size of the virtual world is set to $100,000 \times 100,000$ square units, and 10,000 virtual objects and 1,000 users are initially randomly distributed in the virtual world. Around 40% of the virtual objects are dynamic in nature (i.e., they move around in the virtual space). The lengths and breadths of the viewability scope of every virtual object are set to values ranging from 100 units to 200 units. The heights of their viewability scope vary between 0 units to 1000 units. The length and breadth of the users' viewability scope are allowed to lie between 50 units to 100 units, whereas the height of varies between 100 units to 150 units. The smallest viewing scope exceeds the size of the viewability scope by 100 units along all dimensions. Each successive viewing scope differs by 100 units in size when compared to the previous viewing scope. The probability of a model being updated at any given time is set to 33%. The virtual world is divided into 10×10 zones in case of zone-based and zone-hybrid schemes. The simulations were designed to model real-world scenarios as closely as possible. Some avatars walked in groups towards a common destination whereas some avatars followed other avatars in the virtual world. However, a large majority of avatars explored the virtual world while wandering aimlessly. The walking of avatars was designed to model the realistic behavior of users of a virtual environment rather than having them loiter randomly.

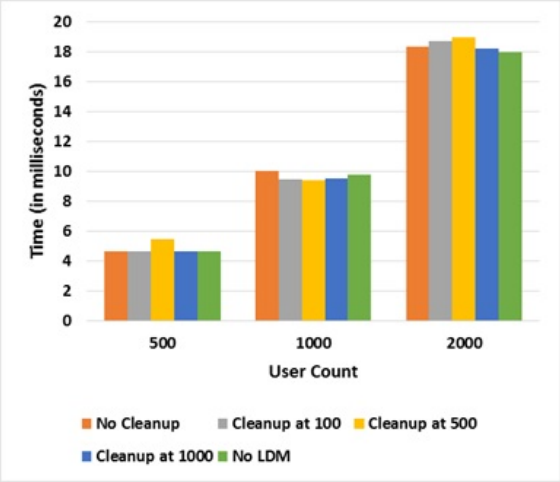
We experimented with three different schemes, namely, LDM with periodic cleanup, LDM without periodic cleanup, and without local caching at all (No LDM), for every interest matching algorithm. No LDM streams geometry data every time a viewer encounters a virtual object. No LDM forms the base for comparison. We use an interval of 500 timesteps for the cleanup process of the LDM framework with periodic cleanup. During the cleanup, we purge the data corresponding to objects with the lowest three values for encounter frequency.

5.3 Observations

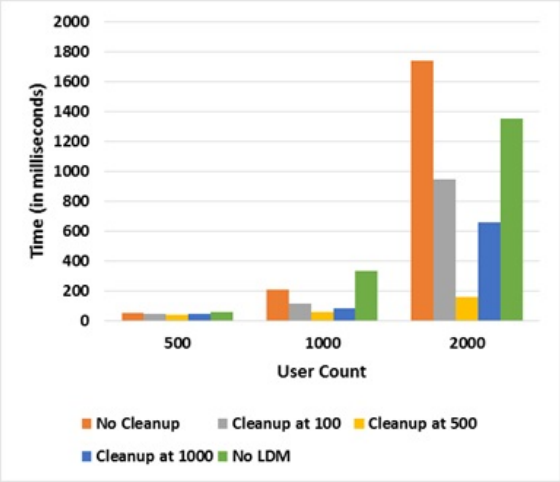
We studied the effect of user count and cleanup period on the computational effort required at the server. The average execution time required by the universal multi-aura scheme, as seen in Figure 2(a), linearly increases with an increase in user count, as a result of the increased number of overlapping scopes. However, the computational effort varies very slightly in relation to the cleanup interval used. This is because the considerably high computational overhead involved in the universal multi-aura scheme (due to its high filtering precision) tends to overshadow the effects of varying cleanup intervals on the overall performance of the system. Similarly, we observed that in the case of the zonal multi-aura scheme the average execution time required per time-step was proportional to the user count but independent of the cleanup period, as seen in Figure 2(b). It is simply because the average number of scope overlaps (i.e., the size of L_{vis}), as well as the average number of objects to be streamed (i.e., the size of L_{stream}) in the zonal multi-aura scheme, is bounded by the average number of objects in a zone. On the other hand, the average execution time for the hybrid scheme is proportional to the user count and also dependent on the periodic cleanup. As seen in Figure 2(c), the best performance is achieved in the case of the LDM framework with cleanup intervals of 500 time-steps. The hybrid scheme experiences a trade-off between the count of locally cached objects to be looked up during object set comparison and the count of objects to be streamed in a time-step. In other words, lesser the number of locally cached objects, more will be the number of objects to be streamed and vice versa, (i.e., the sizes of L_{stream} and L_{loc} are inversely proportional to each other). This trade-off balances well at the cleanup interval of 500, outperforming cleanup intervals of 100 as well as 1000. Also, in the case of LDM with a cleanup interval of 500 time-steps, the slope of the linear increase in average computation time with respect to user count is minimal. In addition, we also observe that the LDM framework with periodic cleanup performs better than the LDM without cleanup and the No LDM baseline frameworks, which again can be accredited to the large size of L_{loc} and L_{stream} in these cases, respectively.



(a) Universal multi-aura



(b) Zonal multi-aura



(c) Hybrid multi-aura

Figure 2: Average execution time per time-step.

Table 1: Maximum disk space used.

Case	User Count	Cleanup Period (time-steps)	Universal (in MB)	Zonal (in MB)	Hybrid (in MB)
1	500	500	1.53	117.50	31.00
2	500	1000	1.55	117.50	51.50
3	1000	500	1.88	127.50	61.58
4	1000	1000	2.35	127.50	104.50
5	2000	500	2.73	132.60	115.08
6	2000	1000	3.38	132.60	175.50

Table 2: Average bandwidth usage per time step (user count: 1000).

Framework Type	Universal (MB per time step)	Zonal (MB per time step)	Hybrid (MB per time step)
LDM with cleanup	0.22	18.33	4.91
No LDM	0.23	24.75	10.21

Since the LDM framework adopts periodic cleanups of the local disk of client systems in order to keep a check on the space used by the DVE objects, it is important to ensure that the maximum utilized disk space at any instance of time is bounded. We observed the maximum usage values of the disk space for the three multi-aura approaches under varying user counts and cleanup periods, as shown in Table 1. The disk space utilization was seen to be increasing with an increase in user count, for each of the three methods. This behavior is intuitive as a rise in user count results in a rise in DVE object count which in turn results in an increased chance of scope overlaps for users and virtual objects. The upper bound on disk space usage increases as the frequency of disk cleanups is reduced for the universal and hybrid schemes. However, in the case of the zonal approach, the maximum reached disk usage is independent of the cleanup frequency, as the data volume streamed in this case is bounded by the maximum object count for the virtual world zones. In addition, the average maximum utilized disk space for each of these methods reflects their respective filtering precision in relation to each other, as we see that the universal multi-aura approach requires the least space on the client system as opposed to the zonal multi-aura scheme which needs the most. Therefore, while implementing the LDM framework it is important to carefully select the cleanup interval based on the user count as well as the maximum available disk space on client systems. Lastly, we also observed (as shown in Table 2) that the average bandwidth requirements for the LDM framework were lesser than the No LDM baseline framework for all three schemes, as expected.

6 CONCLUSIONS AND FUTURE WORK

In this paper, we have presented the design of LDM framework with Multi-Aura Visibility Filtering for 3D content streaming. We have demonstrated the feasibility of the LDM framework and also proved the superior performance of the LDM framework when compared to content streaming frameworks without local caching. Our simulation experiments have reiterated the trade-off between filtering precision and computational overhead in visibility filtering mechanisms. Although universal multi-aura filtering is extremely precise, it induces a high load on the computational resources. On the other hand, the zonal multi-aura scheme gives the best performance computationally but fails to offer sufficient filtering precision. The hybrid approach offers the best of both worlds, from the perspective of both filtering precision and computational overhead.

This study can be extended along several directions. One direction is to explore pre-fetching of objects to reduce latency and prioritization of objects based on their visual importance. Another interesting research

would be in the direction of data partitioning in highly parallel multithread servers and peer-to-peer networks. In subsequent studies, we would also like to extend the application of the LDM framework to deliver 3D content with depth information for 3D viewing on head-mounted devices.

REFERENCES

Benford, S., and L. Fahlén. 1993. "A Spatial Model Of Interaction In Large Virtual Environments". In *Proceedings of the Third European Conference on Computer-Supported Cooperative Work*, 109–124. Springer.

Carlsson, C., and O. Hagsand. 1993. "DIVE–A Platform For Multi-user Virtual Environments". *Computers & Graphics* 17 (6): 663–669.

Chim, J., R. W. Lau, H. V. Leong, and A. Si. 2003. "CyberWalk: A Web-based Distributed Virtual Walkthrough Environment". *IEEE Transactions on Multimedia* 5 (4): 503–515.

Chu, C.-H., Y.-H. Chan, and P. H. Wu. 2008. "3D Streaming Based On Multi-LOD Models For Networked Collaborative Design". *Computers in Industry* 59 (9): 863–872.

Greenhalgh, C., and S. Benford. 1995. "MASSIVE: A Collaborative Virtual Environment For Teleconferencing". *ACM Transactions on Computer-Human Interaction (TOCHI)* 2 (3): 239–261.

Hu, S. Y., T. H. Huang, S. C. Chang, W. L. Sung, J. R. Jiang, and B. Y. Chen. 2008. "FLoD: A Framework For Peer-to-Peer 3D Streaming". In *INFOCOM 2008. The 27th Conference On Computer Communications. IEEE*, 1373–1381. IEEE.

Liu, E. S., and G. K. Theodoropoulos. 2014. "Interest Management For Distributed Virtual Environments: A Survey". *ACM Computing Surveys (CSUR)* 46 (4): 51:1–51:42.

Liu, E. S., M. K. Yip, and G. Yu. 2005. "Scalable Interest Management For Multidimensional Routing Space". In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, 82–85.

Liu, E. S., M. K. Yip, and G. Yu. 2006. "Lucid Platform: Applying HLA DDM To Multiplayer Online Game Middleware". *Computers in Entertainment (CIE)* 4 (4).

Schmalstieg, D., and M. Gervautz. 1996. "Demand-Driven Geometry Transmission For Distributed Virtual Environments". In *Proceedings of the Computer Graphics Forum*, Volume 15, 421–432. Wiley Online Library.

Wong, W.-M. R., and R. R. Muntz. 2000. "Providing Guaranteed Quality Of Service For Interactive Visualization Applications (Poster Session)". In *Proceedings of the 2000 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, 104–105. New York, NY, USA: ACM.

AUTHOR BIOGRAPHIES

ELVIS S. LIU is an Assistant Professor in the School of Computer Science and Engineering at Nanyang Technological University, Singapore. He holds a Ph.D. degree in Computer Science from the University of Birmingham, UK. His research interests include distributed virtual environments, parallel and distributed simulations, and computer games. His email address is elvisliu@ntu.edu.sg.

ADITI RUNGTA is a graduate student at the School of Computer Engineering, Nanyang Technological University. She is pursuing her M.Sc. in Digital Media Technology. Her research interests lie in interest management and content streaming in large-scale virtual environments and massively multi-player online games. Her email address is aditi015@e.ntu.edu.sg.