A Large Scale Distributed Virtual Environment Architecture

Elfizar[1,2], **Mohd Sapiyan BABA**[3], **Tutut HERAWAN**[1,*]

[1] Faculty of Computer Science and Information Technology, University of Malaya,
Pantai Valley, Kuala Lumpur, 50603, Malaysia
elfizarmd@gmail.com; tutut@um.edu.my

[2] Information System Department, University of Riau,
Kampus Binawidya, Pekanbaru, 28293, Indonesia

[3] Gulf University of Science and Technology,
Kuwait City, Kuwait, Kuwait
sapiyan.m@gust.edu.kw

* *corresponding author*

Abstract: Virtual Environment (VE) is a simulation application that is widely used for the development of computer generated synthetic environments. A Distributed VE (DVE) allows many users to access a VE concurrently from different locations. Most current DVEs are still using simulator-centric architecture that views VE operations as a set of homogenous simulators, each aggregating data structure and all the actors operating on the data structure. This architecture limits the number of users involved in the DVE. It reduces users' experiences because the area of VE is restricted. Also, when the number of objects increases, the VE runs more slowly. Although other architectures such as *Distributed Scene Graph* and *Sirikata*, have become available, the simulator still manages many objects in the simulation. It also restricts the number of objects and users involved in the VE. This paper proposes a new architecture to enable large scale distributed virtual environment. A simulator separation method is developed based on objects consisting of one process for one object (1P1O). The 1P1O architecture has a core component that comprises several simulators. In order to maintain the object, each simulator has two engines: physics engine and scripts engine. To maintain the consistency of the simulation, we introduce *Universe* that stores all objects state generated by simulators. *Universe* is responsible to store the state updates and disseminate them to interested simulators. There are two aspects used to evaluate the scalability of the 1P1O model, i.e. the number of objects and the number of concurrent users involved. Parameters such as CPU usages and memory allocation are used to analyze and evaluate the performance of the model. Experiments are conducted with varying number of objects and users. Compared with current architecture, the 1P1O model scalability and performance are better than current existing models in P2P network. The experiment results also comply with the mathematical model of the simulator and *universe*.

Keywords: Distributed virtual environment, 1P1O model, Large scale DVE, Architecture.

1. Introduction

A Virtual Environment (VE) imitates a certain real environment. It should make user feel as residing in the real environment. Hence, VE should meet some requirements occurred in the real world. To involve many users in a VE, Distributed Virtual Environment (DVE) is required. Many users in separated places can come together to collaborate in a VE. DVEs have many applications used in games, education, war simulation, medical simulation, etc.

Virtual world is one of the most popular applications of DVE. For instance, Second Life [1] is state of the art of virtual worlds. On May 2012, the world of Second Life was made up of thousands of regions, which if they are linked together will spread over 1,962.93 km^2 of virtual lands [2]. The world consists of avatars, terrains, trees, buildings, and other objects.

DVEs may have a very large number of objects and users at a time and that can easily overload a fast network, and impose huge processing requirements at the server and client computers. As computing resources are limited, there are obvious problems that arise once the number of objects and users in a simulation reach a certain limit. If no special mechanisms are provided, one may expect a DVE to produce undesirable effects such as choppy rendering, and loss of interactivity.

This paper focuses on this scalability issue. Scaling a DVE depends on two aspects, i.e. scaling the number of concurrent users interacting with each other, or scaling the scene complexity (number of objects and the complexity of their behaviours and appearances).

Several methods have been generated to scale DVEs such as dividing simulation workload [3] [4], using dynamic load balancing among servers [5], and creating alternative architectures [6][7][8][9]. Scaling the DVEs can be done at the server's side (using cluster or cloud computing) or the client's side (using peer-to-peer model). Unfortunately, those techniques are not enough for DVEs with huge number of objects and thousands of concurrent users. Some limitations still occur in the current DVEs. Increasing the number of objects and users decreases the performance of DVE.

This paper proposes a novel DVE architecture, called 1P1O model, to scale up the present DVEs. In the proposed architecture, each object in DVE is treated as a separated process: one process for one object (1P1O). This concept is inspired by the real world in which there are many objects composing the world. They may be static objects and dynamic objects, and there are interactions among them. Each object has control to itself to determine what kind of properties and behaviours it should appear in the world.

This paper makes two research contributions. The first is 1P1O model, a novel DVE architecture that provides a large scale DVE. This model is unlike existing DVE architectures, where a simulator manages many objects. 1P1O breaks the DVE into three components: object simulators, universe, and Content Delivery Network (CDN). An object simulator just simulates an object. Since object simulator is an independent process, the workload-balancing problem can be solved. DVE researchers and developers can use this architecture to scale up their applications in order to accommodate many objects and users in the environment.

The second contribution is the mathematical model of 1P1O. It is able to simplify the architecture. By means of the mathematical model, we are able to determine the characteristics of the architecture including the model complexity so that we can compare the 1P1O model with others.

The rest of the paper is organized as follows. Section 2 presents the current approaches used by researches in scaling up the DVE. Section 3 explores the 1P1O model as a proposed method along with its mathematics model. The experimental results and discussion are described in Section 4. Finally, Section 5 gives some conclusive remarks and future work.

2. Related Work

In present DVEs, entities and activities are managed by a simulator. When the simulator workload increases, the researchers use two approaches to scale the DVE: splitting the region, and separating the component of DVE simulator.

2.1 Splitting the region

To decrease the simulation workload, this approach divides the region into smaller areas and each area is simulated by a simulator. Thus, the simulation workload of each area becomes smaller.

Second Life as a famous DVE application uses this approach. The world is divided into a large number of small regions. Each region has area of 256m x 256m and owned by exactly one simulation server. OpenSim [10], an open source DVE that has a system's architecture similar to and compatible to Second Life architecture, also uses this approach.

Most current Massively Multiplayer Online Games (MMOGs) use this static region-partitioning model. To prevent server crashes, game operators have resolved to use *sharding* [11]. The World of Warcraft [12], the most popular MMOG, uses this *sharding* method.

To increase flexibility of resource allocation and address the over-provision problem when the peak load occurs, managing each DVE region can be integrated with cloud computing [13][14]. Peer-to-peer (P2P) is a new solution to scale up the DVE. Generally, it distributes the server roles to peers. In [15], the authors propose hybrid approaches where the world is divided into regions and each region is assigned to a peer belonging to a structured P2P overlay. Several approaches allow only those peers that satisfy particular requirements in term of hardware capability to manage part of the simulation. These special peers are referred as Super Peers (SP) and usually manage a region so that each peer in the region is connected to them [16].

Splitting the region can also be done by considering the Area of Interest (AOI). Several works [17][18][19] use Voronoi overlays to maintain AOI, which eases the identification of neighbours. To improve the DVE performance, the hybrid of P2P and Cloud architecture can be used [20].

2.2 Separating the components

It is the second approach used by researchers to scale up the DVE. Even though the region has been split into sub-regions, the problem still occurred as the number of objects or users in a sub-region increases dramatically.

To address the problem, the researches generate alternative architectures. Some simulator components are separated from the main simulator so that the workload of main simulator can decrease. The separated component does its tasks in independent process and then the results are sent to main simulator.

Distributed Scene Graph (DSG) [8] uses this approach. It views the DVE operations in general as a collection of the "Scene" and the actors operating on the Scene using a Scene service layer. Actually, DSG is inspired by Dakstar [6] that implements DVE logic as small transactional tasks distributed across servers. DSG separates the physics and script engine as well as client managers from the main simulator. Each actor is responsible to do its task when the simulation runs, and all actors are mediated by Scene.

Another architecture is Sirikata [9][21]. Sirikata's architecture has three components: space, object host, and Content Delivery Network (CDN). It is different from the traditional approach where all objects together with their scripts and data are simulated on a single server or a cluster. Sirikata suggests using an independent Content Delivery Network (CDN), which synchronizes data that is mostly static or changes rarely, e.g. model meshes or textures. This allows reducing the network load from the space server.

3. Proposed Model

Different from the existing DVE architectures, our proposed model, called 1P1O model, has several simulators to manage the object activities and appearances. Object residing in DVE is represented by a process. It maintains the object behaviours and appearances in the environment. Two subsections below present the 1P1O model in detail and its mathematics model.

3.1 The 1P1O model

In present DVE architectures, a simulator manages many objects. As the number of objects and users increase, the workload of simulator increases. Conversely, in the proposed architecture, simulator is no longer a single process that controls many objects. Each object is treated as a process or simulator. Hence, this architecture we call 1P1O model: one process for one object. This model makes object be independent in managing itself.

1P1O model has three components, i.e. object simulators, universe and Content Delivery Network (CDN) as shown by Figure 1.

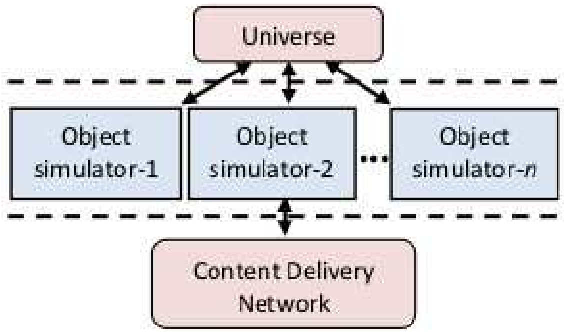

Figure 1. 1P1O architecture

3.1.1 Object simulator component

The core component of 1P1O model is represented by simulators that comprise several object simulators. Object simulator is responsible to simulate the appearances and behaviours of the object. Figure 1 shows that there are *n* object simulators simulating *n* objects in VE. The object simulator-1 is simulator for the first object, the object simulator-2 is simulator for the second object and finally, object simulator-*n* is simulator for n^{th} object.

The object simulator has two components used in managing an object: scripts and physics engine. Scripts engine is used to run the object scripts that determine the appearances and behaviours of object whereas physic engine runs the physics simulation of the object. It ensures that the object enforces the physics laws. Examples are gravity pole, collision handling, etc.

An object simulator is a process in DVE, allowing it to specify their behaviour in scripts and providing it access to the universe component. The viewer is also an object in DVE so that it is represented by a process too. Different from other objects, the viewer is able to display the environment. A user in VE may be represented by avatar.

One important thing is that each simulator can reside in different hardware since it is an independent process. Thus, it is scalable with the additional hardware.

3.1.2 Universe component

The universe component in 1P1O model is responsible to determine what objects reside in the environment, their location, and their physical properties. This component has similarity to the scene in DSG architecture [8] and the space server in Sirikata architecture [21]. It stores object properties in a VE such as object identifier, position, and physical properties.

Universe may be run by one or more computers (called *universe server*) which segment the geometric coordinates of the VE. Similar to object simulator, the universe component is scalable to the additional hardware.

Since *universe* component stores the location and properties of objects, it is able to synchronize the objects position and properties among interested simulators. All object simulators send state updates to universe, and universe just disseminates those updates to the interested simulators. *Universe* is then also able to make an object know the nearest objects.

3.1.3 Content delivery network

Content Delivery Network (CDN) is similar to CDN component in the Sirikata architecture. It stores permanent data and delivers it to the other components. Meshes that are used to display objects are example of the data stored on CDN. Viewers are able to download them to view the VE. The CDN is able to be as simple as a web server. It really needs only to serve requests for files [21].

3.2 Mathematics modelling

In this section, we describe the mathematical model for the present and 1P1O architecture. This model is used to determine the complexity of the 1P1O compared to present architecture. The complexity of 1P1O architecture depends on the complexity of object simulators and universe. Hence, this section is divided into two categories: mathematics model for simulator and universe.

3.2.1 Simulator mathematics model

As described before, the present architectures has simulator that simulates several objects. DSG uses script engine and physics engine that each simulates several objects. Although this actor is able to use multi-servers but each remains simulating several objects in a scene. Similar to DSG, Sirikata's object host also simulates many objects.

An object in DVE is able to be supposed as a set because it comprises a collection of prim (primitive object) such as a cube or a sphere. Computation component is able to make an object change in either its appearance or behaviour, so it is a function.

Let A, B be disjoint sets representing two objects in VE. Let computation component be a function f. The current simulator is able to be represented by $f(A \cup B)$. This function means a simulator that simulates many objects (object A and B) in the environment.

Suppose that $g(A \cup B)$ and $h(A \cup B)$ are script and physics engine, respectively. The DSG architecture uses those functions as their simulators that are separated from scene component, while Sirikata just separates $g(A \cup B)$ from space.

Different from two architectures above, 1P1O architecture defines each object simulator as $g(A) \cup h(A)$. It means that a simulator which contains script and physics engine for its object i.e. A. Hence, we have to show that:

$$\begin{aligned} & g(A \cup B) \cup h(A \cup B) \\ & \quad = (g(A) \cup h(A)) \cup (g(B) \cup h(B)) \end{aligned} \tag{1}$$

To prove the equation (1), we use a set theory theorem that:

$$f(A \cup B) = f(A) \cup f(B) \tag{2}$$

Proof:

$$\begin{aligned} & g(A \cup B) \cup h(A \cup B) \\ & = g(A) \cup g(B) \cup h(A) \cup h(B) \\ & = (g(A) \cup h(A)) \cup (g(B) \cup h(B)). \qquad \blacksquare \end{aligned}$$

Suppose that there are N objects ($P_1, P_2, \ldots, P_n$) in a scene or region, so we should show that:

$$\begin{aligned} & g(\cup_{i=1}^{n} P_i) \cup h(\cup_{i=1}^{n} P_i) \\ & \quad = \cup_{i=1}^{n} (g(P_i) \cup h(P_i)) \end{aligned} \tag{3}$$

To prove (3), first we should show that:

$$f(\cup_{i=1}^{n} P_i) = \cup_{i=1}^{n} f(P_i) \tag{4}$$

Proof:

Based on associative law:

$$f(P_1 \cup P_2 \cup P_3) = f((P_1 \cup P_2) \cup P_3)$$

Then, based on equation (2):

$$f(P_1 \cup P_2 \cup P_3) = f(P_1) \cup f(P_2) \cup f(P_3)$$

Using the same way, then:

$$f(P_1 \cup P_2 \cup P_3 \cup P_4) \\ = f(P_1) \cup f(P_2) \cup f(P_3) \cup f(P_4)$$

Suppose that:

$$f(P_1 \cup P_2 \cup \cdots \cup P_k) = f(P_1) \cup f(P_2) \cup \cdots \cup f(P_k)$$

Then, for *n=k+1* we have:

$$f(P_1 \cup P_2 \cup \cdots \cup P_n) \\ = f((P_1 \cup P_2 \cup \cdots \cup P_k) \cup P_{k+1}) \\ = f(P_1) \cup f(P_2) \cup \cdots \cup f(P_k) \cup f(P_{k+1})$$

Thus,

$$f(\cup_{i=1}^{n} P_i) = \cup_{i=1}^{n} f(P_i). \qquad \blacksquare$$

Now, using (4) we can prove (3) as below:

Proof:

$$g(P_1 \cup P_2 \cup \cdots \cup P_n) \cup h(P_1 \cup P_2 \cup \cdots \cup P_n) \\ = g(P_1) \cup g(P_2) \cup \cdots \cup g(P_n) \cup \\ h(P_1) \cup h(P_2) \cup \cdots \cup h(P_n) \\ = (g(P_1) \cup h(P_1)) \cup (g(P_2) \cup h(P_2)) \cup \cdots \cup \\ (g(P_n) \cup h(P_n)) \\ = \cup_{i=1}^{n} (g(P_i) \cup h(P_i)) \qquad \blacksquare$$

The left side of equation (3) represents the present architecture simulator whereas the right side is the 1P1O object simulator. Since simulator handles *n* objects, the present architecture simulator has script complexity of O(*n*) and physics complexity of O(n^2). For 1P1O model, the simulator has script complexity of O(1) and physics complexity of O(log(*n*)). Hence, the 1P1O simulator complexity is lower than present architectures.

3.2.2 Universe mathematics model

Universe is one of the important components in 1P1O architecture. It manages the overall environment. It receives sate updates from object simulators and disseminates the updates to the interested simulators.

DSG has scene component whereas Sirikata has space to do this task. Sirikata has similarity to the 1P1O model, therefore some space components can be used in universe but physics engine. In 1P1O, physics simulation is handled by object simulators.

Let S_k be object simulators, $k \le n$. Let $t(S_1 \cup S_2 \cup \cdots \cup S_k)$ be a function to receive and disseminate state updates to object simulator S_1, S_2,..., S_k. Since physics simulation is integrated in the space, the Sirikata space component can be represented by:

$$Space = h(P_1 \cup P_2 \cup \cdots \cup P_n) \\ \cup t(S_1 \cup S_2 \cup \cdots \cup S_k) \qquad (5)$$

Equation (5) gives the result that the space component has complexity of O(n^2).

As illustrated by Figure 1, the universe component of 1P1O model has the main task to receive query from objects and disseminate the state changes to nearest objects. Thus, it is able to be represented by:

$$Universe = t(S_1 \cup S_2 \cup \cdots \cup S_k) \qquad (6)$$

Because universe handles *n* object simulator and only needs to disseminate the state updates to the nearest objects, the complexity of this component is O(*n*.log(*n*)).

4. Result and Discussion

The 1P1O model actually represents the DVE as a real world that consists of many objects, continuous space. It is different from the current DVE architectures which are based on partitioned, shared-nothing designs. This paper implements 1P1O model in peer-to-peer (P2P) network. The experimental setup, results, model verification, and discussion are presented in four subsections below.

4.1 Experimental setup

Figure 2 illustrates the 1P1O model implemented in P2P network using super peer (SP). The SP is chosen because this topology is suitable to run the universe component. Each simulator is assigned to a physical node, i.e. a peer. A node is also able to manage some simulators. All peers locating in a region are managed by the same SP. The SP is responsible to collect state changes of the objects and disseminate them to all interested peers.

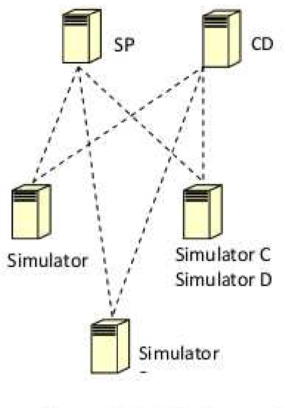

Figure 2. 1P1O implementation

The implementation of 1P1O in P2P network allows users maintain their objects. Users are able to add computational resources to run objects. They may run objects on hosts they control. It is absolutely able to deliver a large number of objects and users residing in the DVE.

A number of three applications are used in the experiment: object simulator, viewer, and universe. The objects used in the experiment are boxes with varying sizes and masses. The experiment simulates the objects falling from a certain hight to the ground and colliding with each other (see Figure 3). The collision is able to occur between object and ground or among objects. The object appearances and behaviours are simulated by an object simulator.

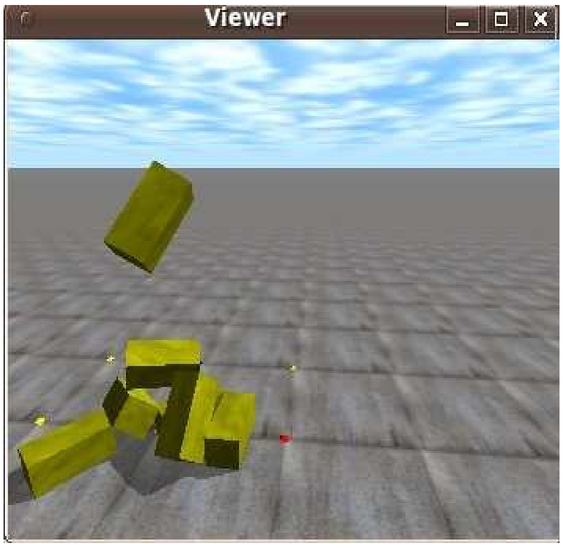

Figure 3. Falling Objects

The objective of this experiment is to evaluate the 1P1O model when it is run in P2P network. The parameters used in the evaluation process are:

- Scalability of the model; It is measured by using the frame rate of simulation with respect to varying number of objects and users involved in the DVE.
- Performance of the model; It is measured by using the CPU usage and memory allocation for simulator.

As illustrated by Figure 2, simulators are executed on each peer computer. The experiment uses two computers with dual core 1.6 GHz processor to execute the simulators and viewers. The number of simulator and viewer in each computer is the same. The universe component runs on a computer with processor 1.6 GHz.

To compare the 1P1O results, we use Sirikata model that the object host runs on a computer with dual core 1.6 GHz, and the viewers run on another dual core computer 1.6 GHz. The space server uses the similar computer specification to universe. The experiment uses varying number of objects and concurrent users.

Some variables measured in this experiment are the frame rates of universe and space with respect to the number of objects and users, CPU usage of simulator for both models, CPU usage of universe and space, memory allocation for simulator of both models, and memory allocation for universe and space. The first variable is used to determine the scalability of the model. We use this parameter to know the effect of increasing the number of objects and users with respect to the frame rate of simulation. The other variables measured in this experiment are used to determine the performance of both models.

4.2 Experiment results

The subsections below are presented based on the variables measured in the experiment.

4.2.1 Frame rates of model based on objects

Table 1 shows the frame rate of 1P1O model for varying number of objects. These frame rates are measured from the condition where universe has already received all updates and disseminated the updates to interested objects to thirty seconds after that condition. Hence, there are six measurements because the frame rate is displayed in every five second. The frame rates are stated in frame per second (fps), and the average frame rates is provided at the right side of table. From Table 1, we note that the increasing number of objects decreases the frame rate of simulation.

Table 2 illustrates the frame rates of space in Sirikata model. These frame rates are measured by using the same manner as 1P1O model. Thus, they also consist of six measurements for varying number of objects. This table also shows that the frame rates of simulation decrease with the increasing number of objects.

The comparison between average frame rates of 1P1O model and Sirikata is illustrated in Figure 4. In fact, the 1P1O model has frame rates higher than Sirikata. 1P1O model frame rates also do not strictly decrease as occurred to Sirikata.

Table 1. Universe frame rates based on objects

Number of object	Universe frame rates (fps)						
	1	2	3	4	5	6	Avrg
10	59.157	59.325	59.204	59.105	59.686	59.421	59.316
20	58.898	59.324	59.647	59.635	58.720	59.439	59.277
30	59.260	58.957	59.438	59.121	58.904	59.313	59.166
40	59.132	58.889	59.507	59.011	59.283	58.745	59.095
50	58.799	59.206	59.147	58.958	59.075	59.241	59.071
60	58.645	58.850	58.903	59.472	59.212	59.366	59.075
70	59.206	59.112	58.758	59.037	58.845	59.221	59.030
80	59.050	58.824	59.318	59.156	58.912	58.860	59.020
90	58.932	59.155	58.876	59.033	58.812	59.149	58.993
100	58.878	58.975	59.261	58.980	59.186	58.774	59.009

Table 2. Space frame rates based on objects

Number of object	Space frame rates (fps)						
	1	2	3	4	5	6	Avrg
10	59.378	58.613	58.472	58.528	59.363	59.187	58.924
20	59.152	59.330	58.965	58.749	58.207	59.047	58.908
30	59.251	57.769	58.340	58.384	58.186	58.518	58.408
40	56.988	58.077	57.598	58.952	57.993	57.787	57.899
50	56.551	57.767	56.330	57.154	56.823	57.549	57.029
60	57.092	56.890	56.956	56.881	57.192	56.166	56.863
70	56.767	57.362	57.327	56.712	56.322	56.767	56.876
80	57.533	56.122	55.833	55.512	56.598	56.165	56.294
90	56.516	56.255	56.578	55.580	55.783	56.966	56.280
100	55.321	55.913	55.780	55.986	56.184	55.477	55.777

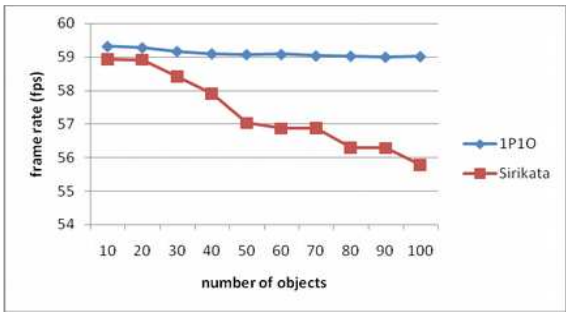

Figure 4. Average frame rates of models based on varying number of objects

4.2.2 Frame rates of model based on users

The frame rate of 1P1O model based on the number of users can be seen in Table 3. The increasing number of users also decreases the frame rates of simulation. Further, the frame rates of Sirikata are drawn by Table 4. They are measured from frame rates of space that determines other components in Sirikata architecture. In fact, similar to 1P1O, the Sirikata frame rates also decrease with the increasing number of users in VE.

Table 3. Universe frame rates based on users

Number of object	Universe frame rates (fps)						
	1	2	3	4	5	6	Avrg
10	59.544	59.540	59.440	59.532	59.586	59.622	59.544
20	59.482	59.530	59.577	59.545	59.550	59.539	59.537
30	59.360	59.372	59.338	59.410	59.304	59.313	59.350
40	59.530	59.529	59.530	59.527	59.533	59.520	59.528
50	59.366	59.206	59.347	59.336	59.342	59.414	59.335
60	59.381	59.238	59.290	59.272	59.252	59.266	59.283
70	59.417	59.381	59.408	59.387	59.403	59.415	59.402
80	59.249	59.253	59.285	59.256	59.232	59.246	59.254
90	59.198	59.212	59.196	59.245	59.234	59.156	59.207
100	59.178	59.205	59.216	59.195	59.186	59.198	59.196

Table 4. Space frame rates based on users

Number of object	Space frame rates (fps)						
	1	2	3	4	5	6	Avrg
10	59.189	59.240	59.248	59.230	59.566	58.519	59.165
20	58.709	59.095	58.994	59.664	58.395	59.041	58.983
30	58.161	58.766	57.554	57.530	58.715	58.074	58.133
40	57.902	57.912	57.113	58.169	57.551	57.647	57.716
50	58.183	57.829	57.530	58.534	58.351	58.185	58.102
60	57.993	57.704	57.167	58.126	57.107	56.561	57.443
70	56.198	57.690	56.768	57.185	58.111	57.570	57.254
80	56.117	56.918	57.199	57.200	57.827	58.461	57.287
90	55.981	56.364	56.338	57.521	57.793	57.323	56.887
100	56.971	56.372	55.913	56.182	56.567	57.033	56.506

It means that increasing the number of users increases the workload of simulation.

The comparison of average frame rates between 1P1O and Sirikata model is able to be seen in Figure 5. Similar to the last section results, this figure shows that the frame rate of 1P1O model is higher than Sirikata. The frame rates of 1P1O model do not strictly decrease as occurred to Sirikata.

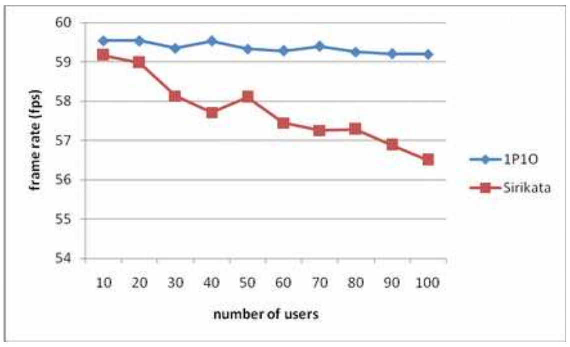

Figure 5. Average frame rates of models based on varying number of users

4.2.3 CPU usage of simulator

The CPU usages of both 1P1O object simulator and Sirikata object host for varying number of objects are able to be seen in Figure 6.

From this figure, we note that CPU usage of 1P1O simulator is lower than Sirikata. Furthermore, the difference of the CPU usage between both models is striking. For example, the CPU usage of 1P1O simulator for 100 objects is 38.17% and this value is almost equal to CPU usage of Sirikata simulator for 30 objects (39.87%). Thus, there is significant decreasing of CPU usage of 1P1O simulator compared to Sirikata simulator.

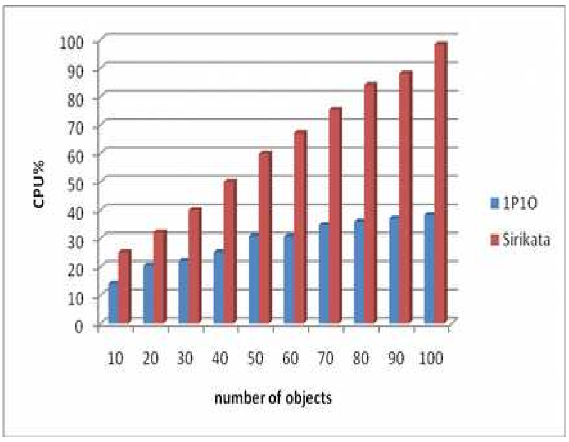

Figure 6. CPU usage of 1P1O & Sirikata Simulator

4.2.4 CPU usage of model

The CPU usage of universe and space for both models for varying number of objects are depicted by Figure 7. The Sirikata model has CPU time longer than 1P1O model. This behaviour is the same as CPU usage of simulator. It means that CPU usage of simulator influences the CPU usage of model.

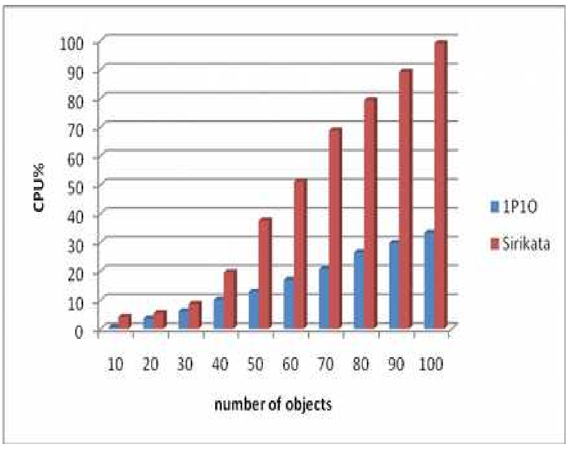

Figure 7. CPU usage of models

4.2.5 Memory allocation for simulator

The memory allocation for 1P1O simulator compared to Sirikata simulator is drawn by Figure 8. From the figure, we note that the 1P1O object simulator uses less memory than Sirikata.

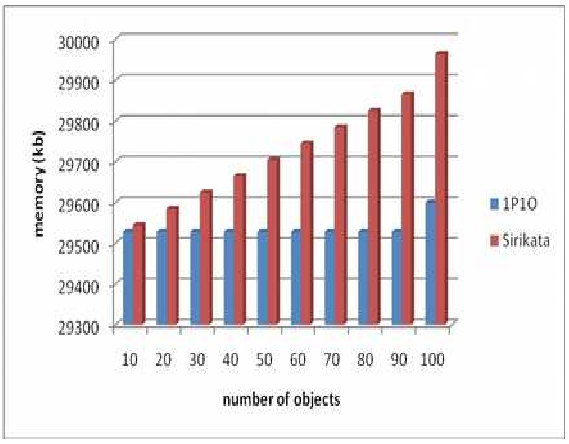

Figure 8. Memory allocation for simulator

The memory allocation for Sirikata simulator is linear whereas 1P1O simulator has constant trend-line for a certain interval.

4.2.6 Memory allocation for model

The comparison of memory allocation between 1P1O and Sirikata model can be seen in Figure 9. From the figure, 1P1O model uses less memory than Sirikata. For 1P1O model, we note that increasing number of object in VE has “delaying time” for increasing memory allocation. The memory allocation for 1P1O model is unchanged from 10 to 90 objects. The value just increases as the number of objects is 100 and this value remains less than Sirikata.

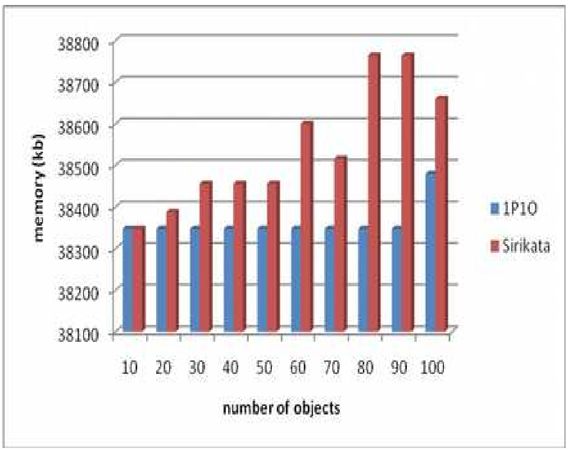

Figure 9. Memory allocation for models

4.3 Model verification

In this section, we verify two mathematics models: simulator and universe model. To verify these models, CPU usages resulted in the last section are required.

4.3.1 Simulator model verification

Consider Figure 6 that illustrates the CPU usage of simulator for 1P1O and Sirikata model. Sirikata is one of the current architectures that has object host for simulating the object scripts. With this figure, Figure 10 shows the trend-line of both simulators.

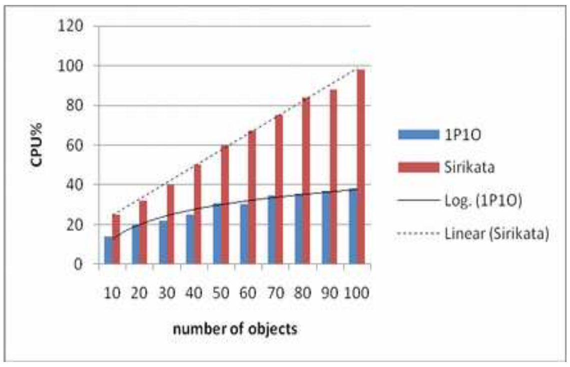

Figure 10. Trend-line of Simulator CPU Usage

From Figure 10, we note that the trend-line of Sirikata object host is linear so that its complexity is O(n). Further, the trend-line of the 1P1O model is logarithmic so that the complexity of 1P1O object simulator is O(log(n)). Thus, they comply with mathematics model in the equation (3).

4.3.2 Universe model verification

To verify the mathematical model in equation (5) and (6), consider the CPU usage of space and universe in Figure 7. By drawing the trend-line of the graph, the complexity of current model (Sirikata) approaches O(n^2) and the complexity of the 1P1O model approaches O(n.log(n)) as illustrated by Figure 11. Thus, they comply with equation (5) and (6), respectively.

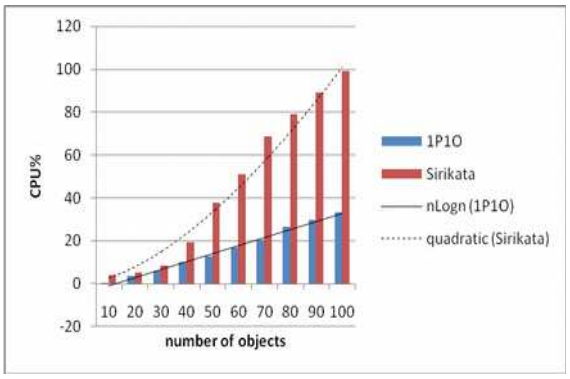

Figure 11. Trend-line of models CPU usage

4.4 Discussion

Based on the experiment results, the frame rate of 1P1O model implemented in P2P is higher than Sirikata for both simulator and model. It occurs for varying number of objects and users. The results show that 1P1O model is more scalable than Sirikata. The increasing number of objects and users still provide rich user experiences.

The good results are also shown by CPU usage and memory allocation. The CPU usage of 1P1O model is less than the current model. It has relationship to CPU usage of simulator that is also less than current model. Further, the memory allocation for simulator and model of 1P1O are also less than the current model. With these results, they show that the 1P1O model has better performance than current model.

5. Conclusion

We have proposed a new DVE architecture, called 1P1O, which is able to scale the distributed virtual environment. Treating an object as a process and this process can be distributed over the network. This architecture may allow additional hardware in the network.

The mathematical model of 1P1O has also been provided to ease determining the complexity of model. We have demonstrated that 1P1O model has frame rate improvement compared with current DVE model. The results also show that our model performs better than existing models for both CPU usage and memory allocation. The results comply with the mathematical model.

Finally, we can conclude that the 1P1O model is more scalable with better performance than the current DVE architecture. The 1P1O model can be used by researchers and developers to accommodate many objects and users in their DVE without degrade the DVE performance and user experiences. Our future work is to implement the 1P1O model in client-server network and compare the results with P2P implementation.

Acknowledgments

The paper was funded partly by University of Riau. The helpful comments and suggestions of the reviewers are acknowledged.

REFERENCES

1. SECOND LIFE, **Second Life**, from http://www.secondlife.com, 2014, accessed on 27-09-2014.

2. SHEPHERD, T., **Second Life Grid Survey – Region Database**, from http://www.gridsurvey.com, 2012, accessed on 05-10-2012.

3. FAROOQ, U., GLAUERT, J., **Scalable and Consistent Virtual Worlds: An Extension to the Architecture of OpenSimulator**, Proceedings of 2011 International Conference on Computer Networks and Information Technology, 2011, pp. 29-34.

4. CARLINI, E., RICCI, L., COPPOLA, M., **Flexible Load Distribution for Hybrid Distributed Virtual Environments**, Future Generation Computer Systems, vol. 29, no. 6, 2013, pp. 1561-1572.

5. LIU, H., BOWMAN, M., **Scale Virtual Worlds Through Dynamic Load Balancing**, Proceedings of IEEE/ACM Symposium on Distributed Simulation and Real Time Applications, 2010, pp. 43-52.

6. WALDO, J., **Scaling in Games and Virtual Worlds**, Communications of ACM, vol. 51, no. 8, 2008, pp. 38-44.

7. BYELOZYOROV, S., JOCHEM, R., PEGORARO, V., SLUSALLEK, P., **From Real Cities to Virtual Worlds using an Open Modular Architecture**, The Visual Computer, vol. 28, no. 1, 2012, pp. 1-13.

8. LAKE, D., BOWMAN, M., LIU, H., **Distributed Scene Graph to Enable Thousands of Interacting Users in a Virtual Environment**, Proceedings of Annual Workshop on Network and System Support for Games, 2010, pp. 140-148.

9. HORN, D., CHESLACK-POSTAVA, E., MISTREE, B.F.T., AZIM, T., TERRACE, J., FREEDMAN, M.J., LEVIS, P., **To Infinity and Not Beyond: Scaling Communication in Virtual Worlds with Meru**, Stanford Computer Science Technical Report, CSTR 2010-01, 2010.

10. OPENSIMULATOR, **Open Simulator**, from http://www.opensimulator.org, 2014, accessed on 15-07-2014.

11. DEBEAUVAIS, T., VALADARES, A., LOPES, C.V., **RCAT: A Scalable Architecture for Massively Multiuser Online Environments**, from http://www.ics.uci.edu/~tdebeauv/files/2013-RCAT.pdf, 2013, accessed on 12-05-2013.

12. WORLD OF WARCRAFT., **World of Warcraft**, from http://www.worldofwarcraft.com, 2014, accessed on 12-05-2014.

13. NAJARAN, M.T., HU, S.Y., HUTCHINSON, N.C., **SPEX: Scalable Spatial Publish/Subscribe for Distributed Virtual Worlds Without Borders**, Proceedings of 5th ACM Multimedia Systems Conference, 2014, pp. 127-138.

14. RICCI, L., CARLINI, E., **Distributed Virtual Environments: From Client Server to Cloud and P2P Architectures**, Proceedings of International Conference on High Performance Computing and Simulation (HPCS), 2012, pp. 8-7.

15. OLANDA, R., PEREZ, M., ORDUNA, J.M., **Hybrid P2P Schemes for Remote Terrain Interactive Visualization System**, Future Generation Computer Systems, vol. 29, no.6, 2013, pp. 1522-1532.

16. KIM, K.C., YEOM, I., LEE, J., **HYMS: A Hybrid MMOG Server Architecture**, IEICE Transactions on Information and Systems, vol. E87, 2004, pp. 2706-2713.

17. ALMASHOR, M., KHALIL, I., TARI, Z., ZOMAYA, A.Y., **Automatic and Autonomous Load Management in Peer-To-Peer Virtual Environments**, IEEE Journal on Selected Areas in Communications, vol. 31, no. 9, 2013, pp. 310-324.

18. CARLINI, E., COPPOLA, M., RICCI, L., **Evaluating Compass Routing Based AOI-Cast by MOGs Mobility Models**, Proceedings of SIMUTools'11, 2011, pp. 328-335.

19. VAN DEN BOSSCHE, B., DE VLEESCHAUWER, B., VERDICKT, T., DE TURCK, F., DHOEDT, B., DEMEESTER, P., **Autonomic Microcell Assignment in Massively Distributed Online Virtual Environments**, Journal of Network and Computer Applications, vol. 32, no. 6, 2009, pp. 1242-1256.

20. RANJAN, R., ZHAO, L., **Peer-To-Peer Service Provisioning in Cloud Computing Environments**, The Journal of Supercomputing, vol. 65, no. 1, 2013, pp. 154-184.

21. SIRIKATA, **Sirikata**, from http://sirikata.com, 2014, accessed on 20-05-2014.