Proceedings of the 2010 Winter Simulation Conference
B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan, and E. Yücesan, eds.

SCALING VIRTUAL WORLDS: SIMULATION REQUIREMENTS AND CHALLENGES

Huaiyu Liu
Mic Bowman
Robert Adams
John Hurliman
Dan Lake

Intel Labs, Intel Corporation
2111 NE 25th Ave
Hillsboro, OR 97124, USA

ABSTRACT

Virtual worlds use simulation to create a fully-immersive 3D space in which users interact and collaborate in real time. It is still a great challenge to scale virtual worlds to provide rich user experiences, high level of realism, and innovative usages. There are three unique simulation requirements in scaling virtual worlds: (1) large-scale, real time and perpetual simulations with distributed interaction, (2) simultaneous visualization for many endpoints with unique perspectives, and (3) multiple simulation engines with different operation characteristics. In this paper, we review the challenges in meeting these requirements, present the scalability barriers we observed in current virtual worlds, and discuss potential virtual world architecture and solutions to address the challenges and overcome the barriers.

1 INTRODUCTION

Virtual worlds use simulation to create a fully-immersive 3D space for users to explore, collaborate, and interact in real time. There are two broad classes of popular virtual worlds: general purpose virtual worlds, such as Second Life®, and massively multiplayer online games, such as World of Warcraft™.

Scalability, however, is still a great challenge in virtual worlds. For instance, Second Life and World of Warcraft are limited to below 100 users interacting with each other concurrently (Gupta et al. 2009). To meet the increasing demand of rich user experiences, high level of realism, and new usages such as experiencing professional sports games in virtual auditoriums, virtual worlds must scale beyond their current capability in several dimensions.

The first dimension is to scale the number of concurrent users interacting with each other. Current virtual worlds have to split the user base and restrict interactions to achieve scalability. For instance, Second Life applies static space partitioning to decompose the space into 256m x 256m regions, each handled by one server. World of Warcraft uses sharding: a part of the virtual world is replicated into shards and different shards reside on different servers, but users on different shards are isolated from each other. Both approaches degrade the massive multi-user experience: sharding prevents large groups of users from interacting by design and static partitioned regions collapse with too many users (Gupta et al. 2009).

The second dimension is to scale the scene complexity, captured by the number of objects and the complexity of their behaviors and appearance. It is especially challenging in general purpose virtual worlds, where user generated content is dominant. Designers of such a world only provide tools, basic parts, and primitive objects (prims) to enable users to be creative and produce content. There is no limitation on what users can build and potentially infinite set of object behaviors could be expressed, hence the term "general purpose". As such, the tools and prims are not optimized for any special purposes. When a

scene becomes more complex, the demand for computation, communication, and rendering can quickly overwhelm the whole system. For example, a complex scene such as the Shengri La Chamomile region on ScienceSim (ScienceSim 2010) has 256K prims, yet it was observed that user experience degraded over 35,000 prims (FRI blog 2010). Similarly, it is argued that the fidelity and resolution in Second Life may not yet be appropriate for "serious" applications (Strassburger et al. 2008).

The third dimension is to scale the fidelity of user interaction. In current virtual worlds, user interaction is usually of simple forms and only happen within a limited range: avatars or objects can only interact with others within a small distance in the same region or shard (Horn et al. 2010). On the other hand, users demand rich experiences and the ability to express a large number of interactions with different complexity, granularity, and scope. Further, they want to interact in a broad range, for example, coordinating with others to do "the wave" virtually in the stands of a virtual soccer arena. Enabling interactions of high fidelity needs improvements and innovations in user interfaces. The challenges, however, go beyond user interfaces. For example, user actions are generally unpredictable and it is hard to apply techniques such as dead-reckoning (Singhal & Zyda 1999) to reduce the bandwidth for sending updates to users.

Meanwhile, in improving scalability through these dimensions, virtual worlds still need to deliver the features that distinguish them as virtual worlds (Singhal & Zyda 1999, Virtual World Review 2006):

- Shared space: Users have the illusion of exploring and interacting in the same space.
- Graphical user interface: The world is represented to users visually, usually in 3D form.
- Interactivity: The world allows users to interact with each other and alter, develop, or submit customized content.
- Real-time: Users should be able to see changes in the world as they occur.
- Perpetual: The world continues to evolve regardless of whether users are logged in

Comparing to traditional simulations, maintaining these features while scaling virtual worlds impose unique simulation requirements:

- Large scale, real-time and perpetual simulations with distributed interactions;
- Simultaneous visualization for many users with different perspectives;
- Multiple simulation engines with different operation characteristics and performance constraints.

In this position paper, we first review these simulation requirements and discuss where existing simulation work still falls short in meeting the requirements. We also present several scalability barriers observed from our experiments with virtual worlds. In the second half of this paper, we discuss our scalability approaches as interesting research directions to inspire innovative solutions. Our hypothesis is that virtual worlds could be scaled by a flexible and scalable architecture, dynamic load balancing, and removing redundant communication and computation in delivering views to users. The key idea in our approach is to break the simulator-centric architecture used in most virtual worlds and detach the data structure from the simulation engines (which we refer to as "actors"). The new architecture enables the data structure and the actors to be optimized and scaled independently. Our preliminary work based on OpenSim, abbreviated OpenSim (OpenSim 2010, Fishwick 2009), an open source virtual world platform that is compatible with Second Life viewers, has shown very promising results: we have demonstrated thousands of users to interact concurrently in the same scene, which is orders of magnitude increase comparing to previous best over-the-network performance in general purpose virtual worlds.

2 UNIQUE SIMULATION REQUIREMENTS AND CHALLENGES IN VIRTUAL WORLDS

There has been decades of work in parallel and distributed simulations, or "PADS" in short (Fujimoto 2003, Perumalla 2006). In these simulations, application models are partitioned into logical processes ("LP"s). The LPs interact by exchanging time-stamped events and the simulations progress by executing

the events in temporal order. The classical techniques have mainly concentrated on time management to ensure that the simulation execution is properly synchronized among the LPs. Several standards have been developed over the years, including the High Level Architecture (Kuhl et al. 1999), to promote reuse and interoperation of simulations, especially by providing support for time synchronized PADS.

Virtual worlds, on the other hand, face unique challenges that go beyond time synchronization among LPs. They usually do not require repeatable executions, hence do not require events to be processed in a "correct" order. As such, virtual worlds today usually adopt a simple synchronization model to accommodate the large scale they operate on. By statically partitioning the worlds into shards or region, each hosted by a server, there is little state shared between servers and synchronization between the servers is largely avoided. These strict partitioning techniques are more work-around than desired solutions: they actually translate a large world into many tiny worlds and greatly limit interactions and user experience.

In this section, we review the unique simulation requirements and challenges in scaling virtual worlds and discuss where the existing technologies still fall short in meeting the requirements.

2.1 Large scale, real-time and perpetual simulations with distributed interactions

Over the years, solutions have been developed for large-scale simulations (Fujimoto et al. 2003), simulation with distributed interactions (Kuhl et al. 1999), and real-time simulations (Müller 2001, Ersal 2009). What is unusual in virtual worlds is the combination of the challenges from all these hard simulation problems. The simulation in a virtual world runs in a scale that is orders of magnitude larger than traditional PADS. For example, as of year 2007, Second Life servers hosted 15400 simulator processes, 30 million concurrent scripts, and 100 TB of user generated content that run 24/7 in a continuous 3D landscape twice the size of Montréal (Purbrick & Lentczner 2007, Wilkes 2008). In addition, in the context of such large scale simulations that run continuously, virtual worlds require real-time simulation and rapid response to user inputs to provide users the illusion that they interact within the world in real-time.

To illustrate the challenges, let us consider the following scenario: fashion design in virtual worlds (Rattner 2009), where designers, sales personnel, and customers meet in a virtual space. Designers can work collaboratively to show their fashion designs on virtual runways to a widely dispersed global audience in real time. Customers could request real time changes to a design, have the changes made, and try out a model of the design in virtual showrooms. In such a scenario, many designers and customers may interact with many fashion designs and they need to see the designs change in real-time.

Showing fashion designs in virtual runways require real time cloth simulation, which itself is computation intensive and a hard simulation problem. For instance, to simulate just one piece of high resolution cloth with detailed folds and wrinkles (2 million triangles) draping over a wardrobe, it took a 16-node cluster of quard-core 2.8Ghz processors over 6 minutes to simulate one frame (Selle et al. 2009). With users interacting with the cloth in real-time, the frame rate would go even lower. On the other hand, 30 frames per second (FPS) is usually required to provide interactive frame rates in virtual worlds (Kumar 2008). To reduce the performance gap, several approaches have been proposed or applied. One is based on the observation that virtual world usages usually can tolerate not exact accurate but plausible results (Müller et al. 2001). Another approach is based on the observation that simulations in virtual worlds are usually soft real-time tasks. When the system is overloaded, time dilation is applied to slow down executions and reduce the rate of delivering updates to clients (Second Life WiKi 2010). Time dilation, unfortunately, results in degraded user experience.

To effectively address the combined challenges of the multiple hard simulation problems, it requires innovative integration of different solutions. More importantly, such a solution needs to be evaluated in the context of large scale simulations running perpetually.

2.2 Simultaneous visualization for many users with unique perspectives

In traditional large-scale simulations, data visualization usually happens after a simulation has completed and all the data have been collected (VisIt 2010). Virtual worlds, on the other hand, require simultaneous simulation and visualization. In addition, each user expects interactive visualization of the world from a viewpoint independent of others. Yet supporting multi-user and multi-viewpoint is still a grand challenge in distributed simulation (Strassburger et al. 2008).Consider the above example again. Supporting real time cloth simulation is challenging. Yet it is even more challenging is to bring the views of draping cloth to a global audience, each viewing from a unqiue viewpoint that may change constantly. This generates enormous demand for both computation and communication. As a concrete example, a small scale 3D scene with 3 billion polygons could have 79.5GB of original data to operate with (Chaudhuri 2009). Even by applying advanced techniques such as hierarchical image-based rendering, it still requires 40 to 80Mbps network bandwidth per client to support interactive visualization. A large scale 3D scene is even more challenging – a 10,000km^2 earth-like scene could have 944GB original data to operate with (Chaudhuri 2009).

To address the challenges, researchers have developed solutions such as interest management (Taylor et al. 1999) and visibility computation (Kumar 2008) to reduce network traffic. Studies to apply them in virtual worlds, however, are still at an early stage. Caching is another approach and has been used extensively by on-line games, where most of the content is static and stored on client machines. In contrast, in general purpose virtual worlds, there is massive user generated content and users continuously create and modify the content. Effective caching solutions are yet to be developed (Kumar 2008).

2.3 Multiple heterogeneous actors with different operation characteristics

Traditional simulations usually involve a few simulation engines whose operation characteristics are known at the time of simulation development and specific optimizations can be applied to scale the operations (Perumalla 2006). In contrast, a virtual world usually incorporates multiple heterogeneous simulation engines (the "actors") to simulate and evolve the world. Each of these actors have different scope of operations, resource requirements, performance constraints and operational characteristics. Further, some of them could be attached as plug-ins and their behaviors may not be known beforehand, which makes it even harder to apply optimizations for scalability. Examples of the actors include:

- client manager, which manipulates the world on behalf of a client and present the world to the client from a certain view point;
- per object actors, such as scripts of an object that defines the object's behaviors, etc;
- small-range actors operating on nearby objects, such as collision detection in physics simulation;
- broad-range actors operating on a large number of objects, such as N-body interactions or protein folding in scientific simulations.

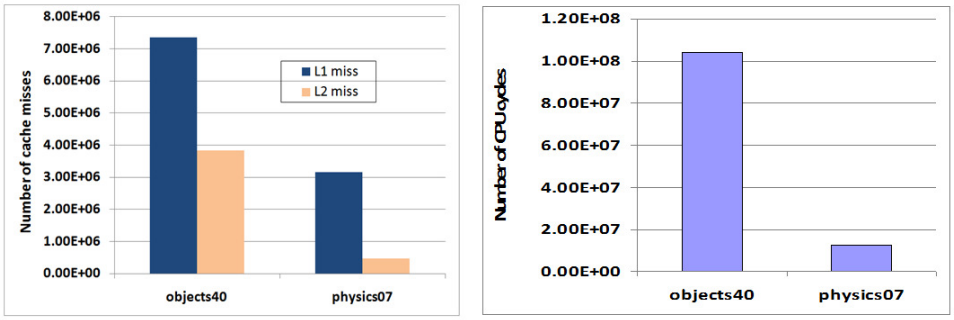

Figure 1: Script intensive vs physics intensive workloads (a) cache misses, (b) CPU stall time

To illustrate the heterogeneity of the actors,

Figure 1 shows some of our measurements of running different workloads, each for a few seconds, in OpenSim. "Object40" is script intensive and has 40,000 scripts running. "Physics07" is physics intensive and has 500 physical object involved in physics simulation (VWperf 2009). The results show that the script workload had much more cache misses than the physics workload (Figure 1(a)), and longer CPU stall time waiting for memory (Figure 1(b)). One explanation is that scripts are built for the objects individually and are stored in different sets of memory, hence has scattered memory access, while the physics actor has its own internal data structure to represent all objects and hence has high locality of memory access. The heterogeneous characteristics of the actors suggest the importance of the ability to apply appropriate hardware to different actors.

3 SCALABILITY BARRIERS

In this section, we analyze the scalability barriers of virtual worlds as constrained by resources. The analysis is based on our experiments with current virtual worlds, especially with OpenSim.

3.1 CPU utilization

When a virtual world scales up in any dimension as discussed in Section 1, the total CPU usage could increase dramatically. In particular, if every entity (object or avatar) is able to interact to with every other entity in the same space, the load would increase quadratically with the number of the entities in world. When the system is overloaded, the rate of sending updates to users (the "frame per second", or FPS) slows down and so do the responses to user inputs, both could significantly degrade user experiences.

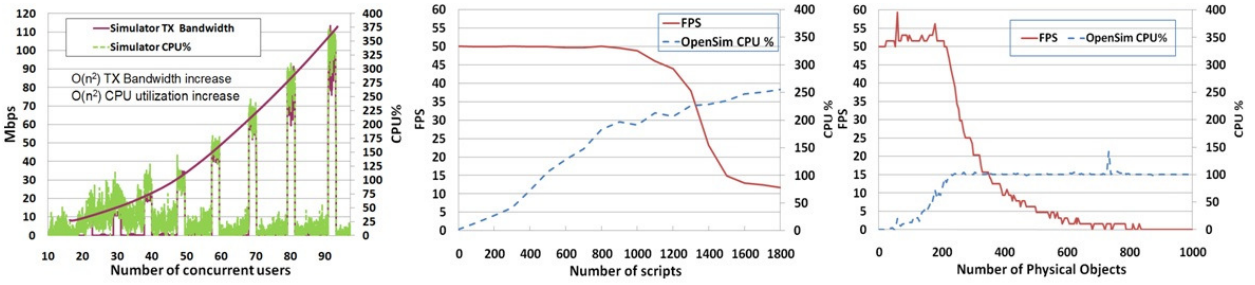

Figure 2: (a) Bandwidth and CPU utilization vs increasing number of concurrent users, (b) CPU load and Frame Per Second (FPS) vs scripting load, (c) CPU load and FPS vs physics load.

Figure 2 shows examples of CPU utilization with different load on different actors. The workloads used in the figure were based on OpenSim 0.6.3 and can be downloaded from ScienceSim (VWperf 2009). The server machine was with a Intel Core 2 Extreme QX6700 quad-processor running WinXP-32. Figure 2(a) shows the CPU load increase with the increasing number of avatars (concurrent users). In this experiment, avatars moved around continuously to generate constant updates. The figure shows that with more and more concurrent users, the CPU load increased quadratically (an $O(n^2)$ problem for the client manager to send updates from every user to all other users, and an $O(n^2)$ problem for collision detection when avatars moving around) and eventually saturated. Figure 2(b) and Figure 2(c) show the results of CPU utilization and FPS versus increasing load on script and physics engines. In both cases, the CPU load kept increasing and eventually the frame rate dropped sharply to lower than 30 FPS, at which rate the user experience degraded to a unsatisfactory level.

3.2 Network bandwidth

Virtual world users are geographically spread across the globe. When the number of concurrent users or the scene complexity increases, the network can quickly be overwhelmed by the aggregated traffic of delivering updates to users. In particular, the aggregated traffic increases quadratically with the number of concurrent users. Assume there are N concurrent users in the same space and on average each generates α

updates per second. Also, assume operations issued by other entities in the world are β per second. Then the number of updates each user needs to receive is $M=\alpha N+\beta$, and the total number of updates transmitted in the network is $MN=\alpha N^2+\beta N$.

This trend of quadratic increase reflects the aggregated traffic that needs to be sent to all users through the network. It is independent of whether the system is based on client-server or peer-to-peer architectures. The quadratic increase, however, is a more serious problem in the client-server architecture if clients are connected to servers directly, because such a setup requires the servers to be able to support the aggregated traffic. Figure 2(a) shows our measurements of the total outgoing traffic with an increasing number of users. The result clearly shows the trend of quadratic increase in network traffic. Techniques such as packet aggregation or packet compression can be applied to reduce network traffic. They could lower the quadratic curve in the figure, but will not change the curve's shape. As long as every update needs to be sent to every user, this quadratic increase of aggregated network traffic still exists.

3.3 Network latency

Studies show that a truly responsive system needs to be able to update visualization at a rate of at least every 100 ms, and a walk-through system requires an update rate of every 20ms (Van Zudilova-Seinstra et al. 2009). Hence network latencies directly impact the realism of user experience. Consider two avatars playing ping-pong in a virtual world. After one avatar hits the ball, the update needs to be received in time by the machine controlling the other avatar so that its user can react as desired. The longer the network latencies, the fewer times the ball can fly back and forth and the less interactive the game is. Even if network bandwidth and CPU utilization are no longer the bottlenecks, the rate of interactive actions is still limited by network latencies.

4 CURRENT VIRTUAL WORLDS: SIMULATOR-CENTRIC ARCHITECTURE

Current virtual worlds usually distribute simulation work to a set of simulators with homogenous functionalities, where each simulator owns a shard or a region plus the complete set of simulation and communication work on the shard or region. We refer to this system architecture as the "simulator-centric" architecture. As a concrete example, the general architecture used in Second Life and OpenSim is shown in Figure 3, where the core of this architecture is a set of homogenous simulators. (There are also a set of infrastructure services for user authentication and authorization, asset and inventory management, etc. These services are usually supported separately from the core simulations.) There are several scalability limitations of this simulator-centric architecture.

- Quadratic load increase on simulators: As discussed previously, such a simulator suffers from quadratic increase of computation and communication load with the increase of the number of interacting entities.
- Limited ability in providing large view distances to clients: Each client's viewable area is limited to the regions or shards hosted on the simulators it connects to. Due to the quadratic load increase problem, each simulator usually sets a limit on the number of concurrent client connections, which in turns limits the number of simulators each client can connect to simultaneously.
- Limitations in dynamic load balancing: This architecture tends to have high overhead of workload migration, inefficient workload partitions for some simulation engines, and limited ability to apply appropriate hardware to the heterogeneous actors (Liu & Bowman 2010).

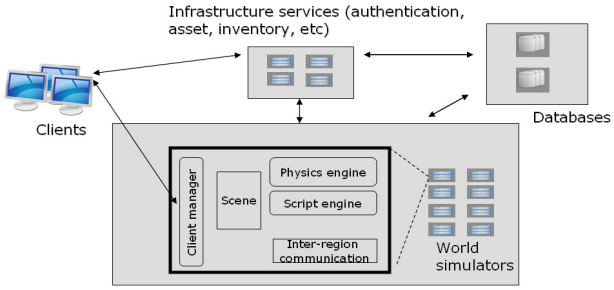

Figure 3: Simulator-centric architecture used in current virtual worlds.

Next we discuss the limitations in load balancing in more details. During dynamic load balancing, workloads need to be migrated among simulators. Our studies have shown that workload migration is an expensive process in current virtual worlds (Liu & Bowman 2010). For instance, Figure 4 shows some results of the migration overhead from our experiments using a dynamic load balancing prototype that is implemented on top of OpenSim 0.6.6. Figure 4(a) shows the migration delay versus the number of migrated primitive objects (prims) and Figure 4(b) shows the corresponding amount of data transmitted in each migration. The trend in Figure 4(a) suggests that it could take thousands of seconds to migrate 100K prims, which would result in intolerable service interruption time. (As a reference, the "Shengri La Chamomile" region on ScienceSim has about 256K prims.)

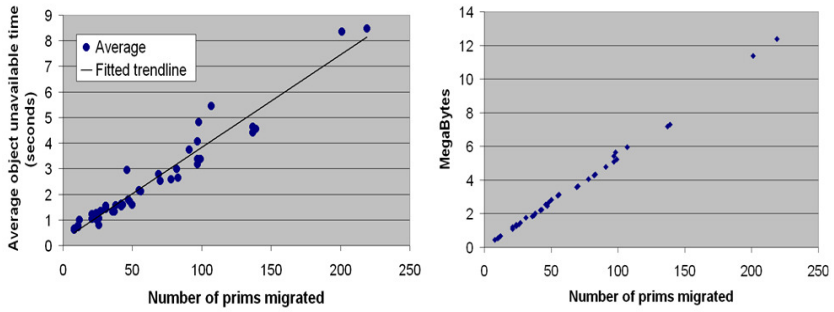

Figure 4: Migration overhead (a) object unavailable time (b) amount of data transmitted in each migration, both ordered by the number of migrated prims in each migration.

We observed that the fundamental reason for the high overhead of workload migration is due to the simulator-centric architecture. Since the objects and the actors operating on them are tightly coupled and physically co-located, every time when the workload in a certain space is migrated, all objects as well as all the simulation work in that space need to be transferred. Hence although objects migration could be optimized to some extent (Liu & Bowman 2010), it cannot be avoided.

A second observation is that aggregating operations of the heterogeneous actors in homogeneous simulators results in inefficient workload partitioning for some actors: it is hard to apply a general workload partitioning policy to satisfy the requirements and constraints of the heterogeneous actors. For instance, physics engine may prefer partitions that preserve object locality while client managers may prefer partitions based on area-of-interest of its clients. Yet most existing studies on load balancing in virtual worlds have mainly focused on satisfying the requirements and constraints of client managers and few have taken into consideration of the constraints of other actors (Liu & Bowman 2010).

A third observation is that, as discussed in Section 2.3, different actors have different compute or communicate characteristics. They may improve performance if each runs on "appropriately configured" hardware. Treating virtual world operations as a set of homogeneous simulators, however, limits the ability to map the heterogeneous actors each to appropriate, heterogeneous hardware.

5 SCALING VIRTUAL WORLDS: NEW DIRECTIONS

In this section, we briefly discuss our approaches in overcoming the scalability barriers as interesting directions for future research. Besides our work, there are several other studies to scale virtual worlds from different aspects: scalable architecture (Horn et al. 2009), scalable data storage (Waldo 2008), scalable message exchanging (Horn et al. 2010), and scalable synchronization protocol (Gupta 2009). These are all in initial steps and there are still many open research problems that call for innovative solutions.

The key idea in our approach is to break apart the simulator-centric architecture and view the virtual world operations as a collection of the "Scene" and the actors operating on the scene. This new architecture is based on a Distributed Scene Graph (DSG), illustrated in Figure 5.

- Scene is the data structure that stores the properties of the space in a virtual world (such as terrain, the sun's position, etc) and the properties of the objects contained in the space. It maintains the authoritative copies of these properties. For load balancing, the Scene may be partitioned and each scene partition could reside on different hardware (hence the "distributed scene graph").
- Each actor observes a portion of the virtual world and applies operations to the objects in this portion, which we refer to as *an actor subscribing to a portion of the Scene*. For each type of actors, there may be multiple instances running and operating on different portions of the Scene. For simplicity, we use "actor" as short for "actor instance".

The actors operate on the objects through the scene interface and thus become portable, which in turn enables them to scale independently. Actors could be either computation or communication intensive. As illustrated in Figure 5, for computation intensive actors, we apply dynamic load balancing to scale their operations. For communication intensive actors, especially the client managers, DSG makes it possible for them to be detached from other actors and run on hardware that is provisioned for high speed networking. Further, separating actors from the Scene provides a logical fit for a detail reduction service as another type of actor to enable large viewable areas by applying multiple levels of detail. Next, we discuss each of these solutions in more detail.

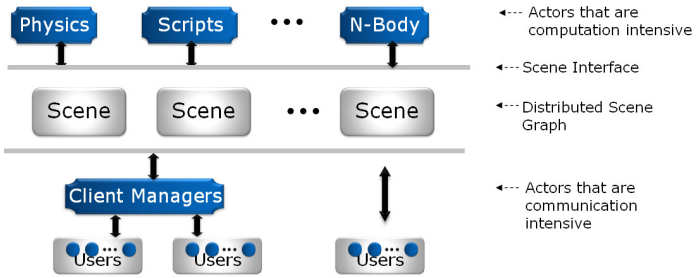

Figure 5: Virtual World Architecture based on Distributed Scene Graph (DSG)

5.1 Distributed Scene Graph

In the DSG architecture, Scene becomes an information hub to connect different actors that interact with and through the Scene. It exposes the operations on objects through the scene interface, acts in response to actor requests (add, remove, update) and distributes object updates and world events (such as collision events, sensor events) to interested actors. By separating the Scene from the actors, Scene could mainly focus on data management, state synchronization, and event distribution.

Detaching actors from the Scene brings back challenges that were by-passed in the simulator-centric architecture. One challenge is time management, or clock synchronization, among the actors that advance their simulation time differently. For instance, physics engines usually run in duty cycles, or a type of "synchronous simulation". Script execution, in contrast, is event-based and a type of "asynchronous simulation". (In the simulator-centric architecture, the problem is simplified since all actors operating on the

same virtual space are located on the same server.) Another challenge is developing efficient and scalable message exchanging protocols to deliver updates between the Scene and the actors.

Technologies developed in the past have provided a basis for addressing these challenges. Examples are "mixed synchronization" for heterogeneous actors (Perumalla 2006) and pub/sub models for message distribution (Ostrowski et al. 2007). A grand challenge in virtual worlds is to enhance these technologies to work in a scale that is orders of magnitude larger than what they have been applied to in the past.

5.2 Dynamic load balancing

Load balancing is essential for scaling operations by dynamic allocation of resource to match load in virtual worlds (Liu & Bowman 2010). We observe that the load balancing problem in virtual worlds can be formulated as a distributed constraint satisfaction problem (Yokoo 2001). In the problem formulation, we assume that the space in a virtual world is continuous and has a finite size. Further, we assume that the space consists of a collection of "quarks", where a quark is the basic unit for balancing workload. All quarks are of a regular shape and have the same size.

In a constraint satisfaction problem, there are a set of variables, the domains of the variables, and a set of constraints about the variables. Given a cluster of servers, we define the variables in the load balancing problem to as follows, where the domain for these variables is the set of all possible subsets of the quarks:

- *Scene Partition* on each server *s*, represented by a set of quarks, as illustrated in Figure 6(a). The server hosts the objects that reside in any quark in the scene partition and provides a scene interface to access these objects.
- *Scene Subscription* of each actor running on each server. It includes the set of quarks that an actor subscribes to for accessing object properties.

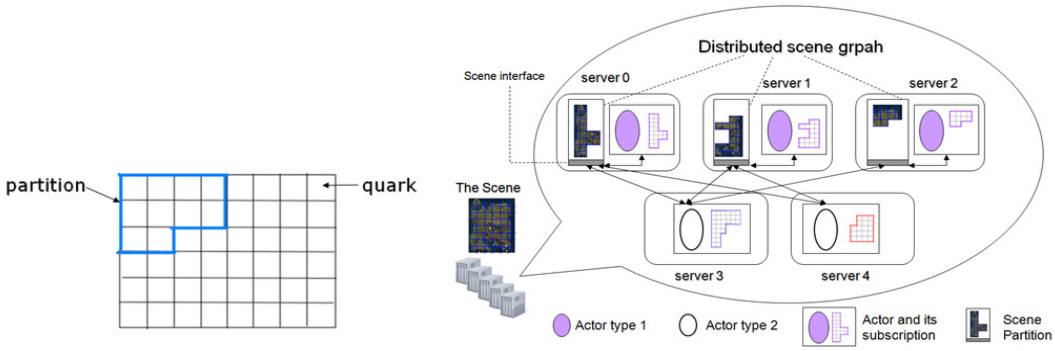

Figure 6: (a) Example of quarks and partitions in 2D space, (b) Load balancing example: mapping DSG and actors to hardware

Based on the DSG architecture, load balancing is mainly about adjusting the scene partition on each server, the assignment of actors to appropriate servers, and the scene subscription of each actor such that the performance and server capability constraints are satisfied. Examples of the constraints include:

- Server capability constraints: the accumulated computation load and the bandwidth usage on each server should not exceed its computation and bandwidth capacity.
- Physics engine performance constraints: Physics engines usually work in duty cycles. At least two constraints could be specified. First, the delay of synchronizing state between a physic engine and DSG is less than a threshold. Second, the execution time in each duty cycle is less than the wall-clock time of the duty cycle (no time dilation).

Load balancing in virtual worlds is a complex constraint satisfaction problem as it has several variables and multiple constraints. It is our future work to develop appropriate solutions. Intuitively, the actors of the same type would negotiate and trade their scene subscriptions to balance load, where the ba-

lancing is subject to server capacity constraints but independent of other type of actors. For illustration purpose, Figure 6(b) shows an example. The Scene is decomposed into three scene partitions, each hosted by a different server. The type 1 actors co-locate with the scene partition that each operates on, while the type 2 actors reside on different servers and each subscribes to an appropriate set of quarks that does not overload its server. All actors update DSG through the scene interface. Note that load balancing between type 2 actors only trigger adjustment of their scene subscriptions and there could be no object migrations.

5.3 Client management

As demonstrated in Figure 2, a linear increase in the number of concurrent users results in quadratic increase of CPU and network usages. The load of managing these concurrent client connections and processing messages can overwhelm a single server with as few as 100 users, depending on their level of interactivity, the specific message protocol, and the types of changes being made to the scene.

In the DSG architecture, client management can be detached from the Scene and the core simulations and assigned to independent actors ("client managers") that run on separate hardware. It brings several benefits. First, the networking bottleneck on a single server is removed – distributing updates to clients is relegated to the client managers. It allows scaling the number of client connections through adding hardware to host more client managers and enables the Scene to be more complex in quantity, variety, and behaviors of the in-world objects. In addition, client managers can be optimized and load balanced separately even through third party services (such as content distribution networks). Second, if the client managers are placed closer to the clients and high speed networking is provided between the Scene and the client managers, it could potentially reduce the network latencies from the Scene to the clients.

As a proof of concept, we have built a prototype that implemented DSG and a set of client managers that communicate with the Scene through the scene interface. An example setup is shown in Figure 7. The client managers are responsible for accepting and authenticating client connections and proxying bidirectional updates between clients and the Scene. We have demonstrated thousands of users to interact concurrently in the same scene, which is orders of magnitude increase comparing to previous best over-the-network performance in general purpose virtual worlds.

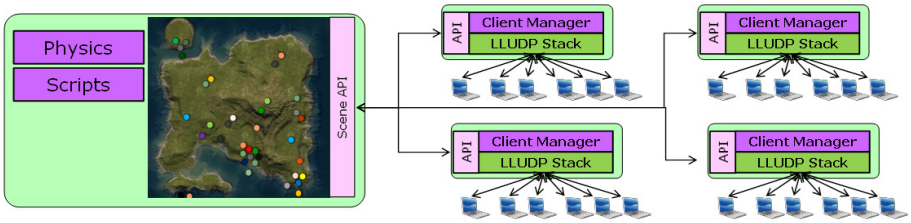

Figure 7: Prototype of client managers: detaching client management from Scene

5.4 Level-of-detail reduction

Another important aspect of improving user experience and fidelity of interaction is to significantly increase the viewable area for each user. Due to the bandwidth constraints in providing broad views, technologies such as multi-level detail representations and visibility computation need to be applied to reduce the communication demand. On the other hand, generating multi-level details and applying visibility computation for every user is computationally expensive, hence it is also necessary to explore the shared perspectives among nearby viewpoints to reduce the computation demand (Chaudhuri 2009).

The DSG architecture provides a logical fit for introducing a detail reduction service between the Scene and the clients. It is our future research to develop a "reduction pipeline" that combines partial rendering, level of detail reduction, and aggregation and compression techniques to expand the viewable distance in general purpose virtual worlds by at least two orders of magnitude. We are experimenting with solutions such as surface elements (Pfister et al. 2000) and image based rendering (Shum and Kang 2000) to apply highly compressed scene representations for remote regions that may only occupy a small number of pixels on a client's screen. The primary challenges include handling frustum culling for the differ-

ent perspectives of all users, managing the high volume of data transferred through the reduction pipeline, and handling smooth visual transitions between level of detail changes.

6 CONCLUSION

There has been decades of research on parallel and distributed simulations. Yet the growing popularity of virtual worlds brings unique challenges to the simulation of a fully-immersive 3D space with high realism and in which massive numbers of users interact in real time. In this paper, we discussed these unique challenges and explained the gap between the demand and the existing solutions. Many existing distributed simulation technologies need to be enhanced to operate in a scale that is orders of magnitude larger than what they have been applied to before. On the other hand, current virtual worlds adopt a simulator-centric architecture and static partitioning to simplify distributed simulation problems such as time management and load balancing. As we observed, this architecture also has scalability limitations.

Virtual worlds are both computation and communication intensive and require solutions that address resource constraints on both fronts. We are working on a new architecture that detach actors from the Scene and enable application of scalability solutions independently on different actors and. There are still many open research problems to be addressed in scaling virtual worlds beyond their current capability.

REFERENCE

Chaudhuri, S., D. Horn, P. Hanrahan, and V. Koltun. 2009. Image-Based Exploration of Massive Online Environments. Stanford Computer Science Technical Report, CSTR 2009-02.

Ersal, T., M. Brudnak, A. Salvi, J.L. Stein, Z. Filipi, H.K. and Fathy. 2009.Development of an Internet-Distributed Hardware-in-the-Loop Simulation Platform for an Automotive Application, In *Proceedings of ASME Dynamic Systems and Control Conference*, Hollywood, California.

Fashion Research Institute (FRI) Blog. 2010. Available via <http://shenlei.com/2010/01/18/sciencesimland-grant-program-overview-faq> [accessed May 28, 2010].

Fishwick, P. 2009. An Introduction to Opensimulator and Virtual Environment Agent-based M&S Applications. In *Proceedings of the 2009 Winter Simulation Conference*, eds. M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin and R. G. Ingalls. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Fujimoto, R. M. 2003. Distributed Simulation Systems. In *Proceedings of 2003 Winter Simulation Conference*, eds. S. E. Chick, P. J. Sanchez, D. M. Ferrin, D. J. Morrice. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Fujimoto, R. M., K. Perumalla, A. Park, H. Wu, M.H. Ammar, and G.F. Riley. 2003. Large-scale network simulation – how big? how fast? In *Symposium on Modeling, Analysis and Simulation of Computer Telecommunication Systems* (MASCOTS).

Gupta, N., A. Demers, J. Gehrke, P. Unterbrunner, and W. White. 2009. Scalability for Virtual Worlds. In *Proceedings of the 2009 IEEE international Conference on Data Engineering* (ICDE).

Horn, D., E. Cheslack-Postava, T. Azim, M.J. Freedman, and P. Levis. 2009. Scaling Virtual Worlds with a Physical Metaphor. *IEEE Pervasive Computing* 8 (3), 50-54.

Horn, D., E. Cheslack-Postava, B.F.T. Mistree, T. Azim, J. Terrace, M.J. Freedman, and P. Levis. 2010. To Infinity and Not Beyond: Scaling Communication in Virtual Worlds with Meru. Stanford Computer Science Technical Report, CSTR 2010-01.

Kuhl, F., R. Weatherly, and J. Dahmann. 1999. *Creating Computer Simulation Systems: An Introduction to the High. Level Architecture*. Prentice Hall PTR, 1999

Kumar, S., J. Chhugani, C. Kim, D. Kim, A. Nguyen, P. Dubey, C. Bienia, and Y. Kim. 2008. Second Life and the New Generation of Virtual Worlds. *Computer* 41 (9), 46-53.

Liu, H. and Mic. Bowman. 2010. Scale Virtual Worlds Through Dynamic Load Balancing. In *Proc. of 14th IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications*.

Müller, M., L. McMillan, J. Dorsey,, and R. Jagnow. 2001. Real-time simulation of deformation and fracture of stiff materials. In *Proceedings of the Eurographic Workshop on Computer Animation and Simulation* (Manchester, UK, September 02 - 03, 2001).

OpenSim. 2010. Available via <http://opensimulator.org/wiki/Main_Page> [accessed May 28, 2010]

Ostrowski, K., K. Birman, and D. Dolev. 2007. Extensible Architecture for High-Performance, Scalable, Reliable Publish-Subscribe Eventing and Notification. *International Journal of Web Services Research* 4 (4), 18-58.

VWperf. 2009. OpenSim Performance Tests . Available via <http://www.sciencesim.com/wiki/doku.php/opensim/performance_tests> [accessed April 28, 2010].

Purbrick, J. and M. Lentczner. 2007. Second Life: The World's Biggest Programming Environment. Keynote Address. OOPSLA 2007.

Perumalla, K. 2006. Parallel and Distributed Simulation: Traditional Techniques and Recent Advances. In *Proceedings of the 2006 Winter Simulation Conference*.

Pfister, H., M. Zwicker, J. van Baar, and M. Gross. 2000. Surfels: surface elements as rendering primitives. In *Proceedings of the 27th Annual Conference on Computer Graphics and interactive Techniques*, SIGGRAPH 2000.

Rattner, J. 2009. Opening address: The rise of the 3D internet - advancements in collaborative and immersive sciences. In *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis* (SC '09).

Second Life WiKi. 2010. Time dilation. Available via <http://wiki.secondlife.com/wiki/LlGetRegionTimeDilation> [accessed May 17, 2010].

Selle, A., J. Su, G. Irving, and R. Fedkiw. 2009. Robust High-Resolution Cloth Using Parallelism, History-Based Collisions, and Accurate Friction. *IEEE Transactions on Visualization and Computer Graphics* 15 (2), 339-350.

ScienceSim. 2010, Available via <http://www.sciencesim.com/wiki/doku.php> [accessed May 28, 2010]

Shum, H.-Y. and S.B. Kang. 2000. Review of image-based rendering techniques. In *Proceedings of SPIE Int'l Conf. on Visual Communications and Image Processing,* Perth, Australia, June 2000, pp. 2-13.

Singhal, S. and M. Zyda. 1999. *Networked Virtual Environments: Design and Implementation*. ACM Press/Addison-Wesley Publishing Co.

Strassburger, S., T. Schulze, and R. Fujimoto. 2008. Future Trends in Distributed Simulation and Distributed Virtual Environments. Peer Study Final Report. Version 1.0. Ilmenau, Magdeburg, Atlanta.

Taylor, S. J., J. Saville, and R. Sudra. 1999. Developing interest management techniques in distributed interactive simulation using Java. In *Proceedings of the 1999 Winter Simulation Conference*, ed. P.A. Farrington, H.B. Nembhard, D.T. Sturrock , G.W. Evans. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Van Zudilova-Seinstra, E., T. Adriaansen, and R. Van Liere. 2009. *Trends in Interactive Visualization: State-of-the-Art Survey*. Springer London.

VisIt. 2010. Available via <https://wci.llnl.gov/codes/visit/about.html> [accessed May 28, 2010]

Virtual World Review 2006, "What is a Virtual World", Available via <http://www.virtualworldsreview.com/info/whatis.shtml> [accessed May 28, 2010].

Waldo, J. 2008. Scaling in games and virtual worlds. *Communications of ACM* 51 (8), 38-44.

Wilkes, I. 2008. Second Life's Architecture. Qcon San Francisco 2008 Conference. Available at <http://www.infoq.com/presentations/Second-Life-Ian-Wilkes> [accessed May 28, 2010]

Yokoo, M. 2001. *Distributed Constraint Satisfaction: Foundations of Cooperation in Multi-agent Systems*. Springer Series on Agent technology.

AUTHOR BIOGRAPHIES

HUAIYU LIU is a research scientist in the Virtual Worlds Infrastructure team, Intel Labs. She holds a Ph.D. in Computer Sciences from the University of Texas at Austin. While at Intel, she had worked on

multi-radio networks and energy efficient communications. Her current research is developing scalable infrastructure for virtual worlds. Her email is <huaiyu.liu@intel.com>.

MIC BOWMAN is a principal engineer in Intel Labs and leads the Virtual World Infrastructure research project investigating systems architectures for scalable virtual environments. He received his BS from the University of Montana, and his MS and PhD in Computer Science from the University of Arizona. He joined Intel Architecture Lab in 1999. While at Intel, he developed personal information retrieval applications, context-based communication systems, and middleware services for mobile applications. In addition, he led the team that built and deployed the first version of PlanetLab, a global testbed for networking research. His email is <mic.bowman@intel.com>.

ROBERT ADAMS is a research scientist in the Virtual worlds Infrastructure team, Intel labs, where his main research interest is performance and metrics of virtual worlds. His email address is <robert.adams@intel.com>.

JOHN HURLIMAN is a software engineer in the Virtual Worlds Infrastructure team, Intel Labs, where he researches and develops future computing architecture for Immersive Connected Experiences. His email is <john.hurliman@intel.com>.

DAN LAKE is a software engineer in Visual Application Research team, Intel Labs. His current research is in the area of scalable architectures for massively interactive and immersive virtual worlds. He holds a Bachelor's degree in Electrical Engineering and Master's degree in Computer Science from Portland State University. His email is <dan.lake@intel.com>.