Evolution of Scalability with Synchronized State in Virtual Environments

Arthur Valadares, Thomas Debeauvais, Cristina V. Lopes
University of California, Irvine
{avaladar, tdebeauv,lopes} @ics.uci.edu

***Abstract*—Developing scalable software architectures to suit virtual environment applications has been a challenge even in face of extraordinary advancements in computing and bandwidth power. In virtual environments, modifications to the world must be broadcasted across all clients, creating a network and processing complexity on the server-side that grows at an $O(N^2)$ rate. Therefore, scalability will always be hard to obtain for any virtual environment architecture.**

In order to achieve scalability, modern virtual environment architectures make several assumptions about user distribution, update frequency, and read and write operation in the environment. Such constraints can greatly increase scalability, but the cost is often ignored from developers, creating undesirable restrictions on how users should behave and what functionalities they may expect.

We approach modern architecture constraints with a critical view of the implicit impositions that are made, compare the end results and performance of their deployment, and suggest new directions away from foundational and mostly unquestioned assumptions about virtual environment architectures.

***Index Terms*—software architecture, virtual world, online games, scalability**

I. Introduction

The development of a virtual environment architecture has gone through many changes since it's inception in [1]. The overwhelming demand of bandwidth and memory required left the field mostly theoretical until the 90's, when Personal Computers and network bandwidth were more readily available to the general public. Foundational papers like DIVE [2] and Diamond Park [3] presented groundwork solutions geared towards a viable shared user experience of a Massively Multiuser Virtual Environment (MMVE).

In spite of the exponential advancement in hardware in the past decades, the deployment of MMVEs remained a challenge due to its complex $O(N^2)$ nature: in a world with N clients, for every action sent from one client that modifies the world, N messages must be sent to inform other clients. The transmission and processing necessary on the server-side grows at an $O(N^2)$ rate. To provide a good quality of experience with a massive number of users, many constraints are imposed that limits the application design or the user experience in the environment.

In DIVE, the concept of spatial partitioning is presented in the form of a virtual environment architecture. Under the assumption that only the "visible world" (i.e. mimicking human viewing distance) of clients is important, dividing the world into separate enclosed spaces and delegating their responsibilities to different servers across the network greatly reduces the $O(N^2)$ complexity. This constraint has its cost: it assumes a uniformly divided world in order to scale, when in actuality an uneven object and avatar distribution is observed in virtual environments [4]. Additionally, the worst case scenario of clients interacting in region borders also creates a disruptive user experience and high demand server load. Hence spatial partitioning does not always suit the load induced by real-life groups of players [5].

To afford higher numbers of interacting users many other constraints are made, and it is easy for developers to lose perception of the cost that is being paid for their assumptions. In section II, we present the modern MMVE constraints and how they affect the users and application designers. We explore different models that challenge even the foundational spatial partitioning scheme [6]. Section III presents modern applications and their architectures, comparing their success in delivering their proposed user experience. Finally, in section IV, we present new ideas for research that lead in novel directions beyond the foundational conjectures and the most common industry practices.

II. Synchronization Models

The state manipulation performed in Virtual World consists primarily of two types: the transactional state operations (TSO) and the fast-paced self-state updates (SSU). TSO are user requests that are intended to perform a modification to shared entities, such as modifying or destroying game objects or other players. In contrast, SSU are only informational messages about an entity solely owned by one client, be it an avatar or an owned object.

TSO requires a stronger consistency approach to client requests, enforcing ACID properties, but is also less frequent in MMVEs. Common examples are trading virtual assets, and modifying health points in MMOGs. SSU are very fast paced (i.e. usually ranging from 5 to 30 updates per second), and has less constraints on consistency, as the entity's write permission belongs to only one user. Primary cases are movement and action updates that must be broadcasted to other users. While the first type of data requests are important and have its own set of research questions [7], for the purpose of scalability the second form of requests is the current bottleneck of MMVEs, deriving from the combination of fast and constant update rates, and the N^2 condition.

Inasmuch of their essential complexity, certain premisses can be made to afford scalability. In the next subsections we show different premisses and approaches that have proven to be efficient forms of reducing complexity and increasing horizontal scalability.

A. Distributed Caching and Space Partitioning

The nature of SSU requests requires low latency and strong consistency, as they generate the observable world that users will act on to perform future actions. Yet there is no need for storing the updates, only the latest states. As such, persistence becomes flexible. Only the latest state needs to be flushed to disk at a much slower rate, and only to allow an elegant recovery from a system crash or for data analytics.

The natural approach to these requirements is the use of caching with low or immediate expiration. Yet caching SSU requires that all data must be in memory constantly, and it is likely that all world entities' states would not fit in memory. In order to scale the memory needs of SSU requests, a memory partitioning algorithm for multiple servers is required.

One of the first proposed architectures for virtual environments, DIVE [2] presents a model that is still followed by most MMVEs: a set of processes that read and modify shared data across a network. The shared data is seen as one unified memory space that is partitioned by worlds, with each process being a member of exactly one world. By dividing the geographical space of the virtual environments into sets of worlds (i.e. enclosed regions), each world is delegated to a server with the responsibility to respond to requests only for the objects located in its enclosed space. It assumes uniformity of objects per region and of requests per object.

Another less common approach is the use of a distributed memory caching (DMC) technique [8]. In a DMC, objects are located and stored by keys. DMCs provide fast access to data and partition tolerance at the cost of higher memory resource usage, and either consistency (i.e. eventually consistent state, if using a replicated DMC) or availability [9]. The main advantage of a DMC is using hashing algorithms allow localization of objects without the need for remote lookups.

While both approaches are good improvements to scalability, any static coupling of data to node will present a worse-case scenario where the load is concentrated on a single node. DMC can use hashing and become less likely to have a worst-case scenario than the space partitioned algorithm, yet it is likely to provide a worst average case result.

B. Client Connection Manager

Handling network connections demands much resource from servers, particularly due to the nature of the packets in virtual environments. MMVE packets are small sized ([12], [5], [13]), yet very frequent, causing constant interruptions and change of context in the CPU. Because user distribution in a virtual environment is not uniform, in a space partitioned world some regions will have massive number of users, where others will be left empty.

To provide balance of network connections, several MMVE servers have added a client management layer that acts as a smart packet router(e.g. EVE Online's Proxy [10], DSG's Client Manager [14], Tera's Arbiter Server [15], Pikkotekk ALG [11]). This layer is responsible to handle client connections and forward their packets to the correct server. If the world is space partitioned, the client manager is responsible to forward requests to the server the client is located in, and forward requests back to that client with world events. The addition of the client manager not only provides load balance, but also increases robustness and fault tolerance, meaning back end and client connection issues are now detached.

In applications where clients connect directly to game servers, the client is dropped from the game if the game server crashes. With a client management layer, the client can be relocated to a different region or a replicated instance of the same world can be used to transfer the client nearly seamlessly in place.

C. Distributed Services

The traditional view of gaming development has a game loop, where all the game processing occurs within one frame. This monolithic model works well under predictable circumstances found in most single player games, but in MMVEs some situations (e.g. heavy physics processing) may spike the processing load, and can possibly result in slow responsiveness or crash. The game loop can perform CPU and memory intensive operations, such as physics and script engines, that greatly varies based on the number of world entities interacting or the type of scripts that are running. Most MMVE applications separate database concerns from game loop (e.g. inventory, transactional state operations), but fail to separate CPU and memory intensive services such as physics, causing game servers to be quickly overloaded in more demanding scenarios with hundreds to thousands of users.

To further scale and separate concerns in the game loop, the heavy game processing can be separated from the core game loop. This approach is seen in [14], where Physics calculations and Script Engines are offloaded to different servers, and can constantly process requests and report results to the world's region servers, which is left to process only user requests and service results. This way, servers can scale independently, and failures do not propagate in the server responsiveness.

The main downside to this approach is the extra latency involved in sending tasks to the service servers, and the added complexity of synchronizing frames across separate processes.

D. Geographic Independence

All the previous solutions have greatly increased the number of concurrent users by separating independent components of MMVEs into a Service Oriented Architecture (SOA). As is common with SOA approaches, the complexity of the system increases, but individual pieces are easier to maintain and to scale.

One strong coupling still remains in all previous premisses: that world entities should be bound to the space they are

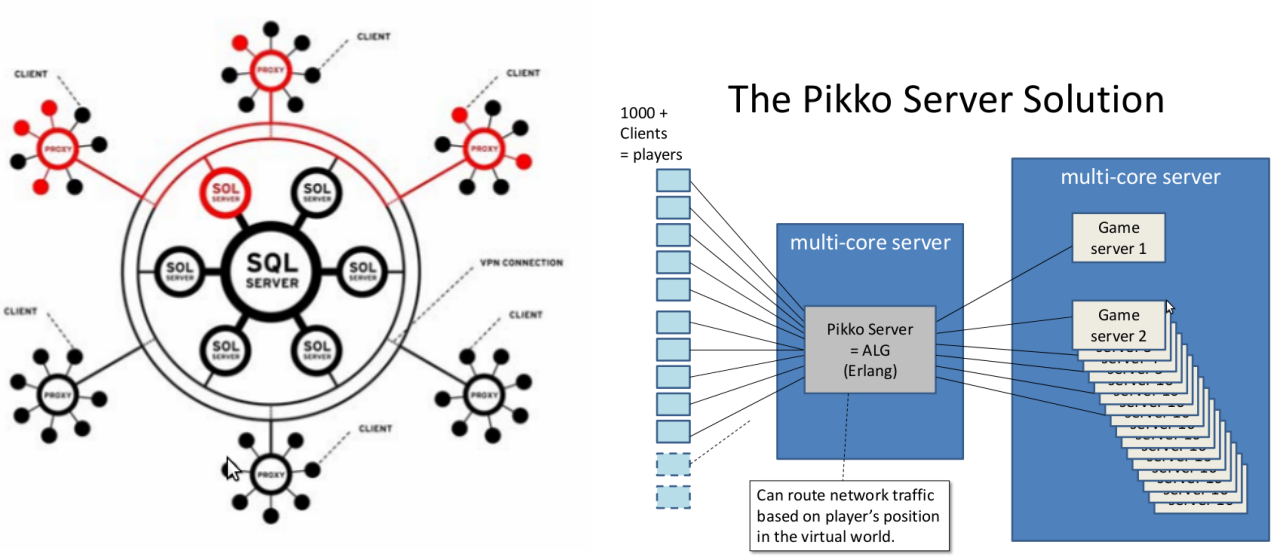

Fig. 1. Left: Architecture for EVE Online [10], Right: Architecture for Pikko Server [11]

located in. It is a traditional object-oriented approach to conceive regions owning the entities located in it, but for the purpose of scalability it can perform poorly when some regions are more densely populated with world entities (i.e. objects, NPCs, scripts) than others. Another issue is at the border of regions, where inter-server communication becomes necessary to allow interaction of world entities in between two or more regions.

Project Darkstar [6] presented an original approach to data storage based on tasks and independent from geographical location. Traditionally, data is either centralized on a data server or it is partitioned on to different servers, requiring requests being sent for read and write operations. Darkstar uses a centralized approach, but allows caching with a conflict resolution method. When a task transaction requires write access to a data item it informs the central server who checks for conflicts. If no conflicts happen, it gains exclusive access to write it, if there are conflicts, it fails and runs again after a period of time.

This approach has the benefit of not tying the data items to a single server, and allowing cached access. The possible down-side is the worst case scenario where multiple nodes request write access, creating several conflicts that either constantly needs to contact the central server for write permissions or reattempts to request write access. Either way, an increase in latency will be noticeable and unpredictable.

E. Dynamic Space Partitioning

To improve the numbers of users interacting in a smaller space, research has been directed to dynamic space partitioning (DSP). As opposed to traditional static region boundaries, the nodes partition the world space based on the load of users for each region. There are different known approaches to dynamically partition a region by subdividing it in smaller parts (e.g. [16],[17]), but both EVE Online and Pikko Server have developed their own optimized DSP algorithms that constantly recalculate dynamic boundaries.

In EVE Online [10], ships have spheres of influence, a calculated volume where it could interact with other world objects. When spheres collide a causality bubble is formed, aggregating the spheres under one enclosed space where actions are broadcasted. Essentially EVE Online still retains a fixed region server in the traditional space-partitioned manner, the SOL servers. The causality bubbles are for fine grained interest management, so it is still bound to the object locality based on geography.

The Pikko server [11] implements the ”Mast algorithm” for DSP. It uses a similar algorithm to cell phone towers, trading off user data from one tower to another as the users move around the environment. The user requests are load balanced across the different nodes that can move users to or from bordering masts. To allow region border interaction, proxy objects are placed in the borders that mirrors objects from the different nodes to players. The proxy objects provide an efficient but limited solution, as players will not have updates about objects far from the border. And as with the EVE Online causality bubbles, objects are still grouped by proximity, but with varying dynamic region sizes.

DSPs are great dynamic approaches to interest management, attempting to collocate frequent interacting users and objects in the same space. It is a great improvement over static spatial partitioning, but still has one limitation: it still assumes that interactivity is uniform across space and is always bound to be an area or a volume.

III. Synchronization in Practice

In Table I we have a feature comparison of popular net-working user applications from the point of view of scalability,

as presented in section II. As can be seen in the table, even though these applications are independently developed, many share the same concepts for achieving scalability. Due to the industry of MMVEs being composed mainly of proprietary technology, there is little shared information about the details of implementation, resulting in most commercial products having to reinvent and code from scratch the same solutions. Most of the information from proprietary technologies in this table is drawn from presentations in conferences from the companies' developers.

A. Counter-Strike

Counter-Strike, a mod to the Half-Life game from Valve, is a well known First Person Shooter (FPS). FPS are not intended to have a massive number of users, but is added to the list as an example of a non-scalable application. It is a multiplayer only game where players are divided into two teams, and is played on series of short rounds. FPSs are known for optimizing for performance and having poor scalability.

Information on the networking of Counter-Strike is presented at Valve's developer community webpage [18]. As we presented in table I, it has no identified scalable factors commonly found in MMVE applications, there is simply a cap of 64 maximum players in a single game to avoid overloading a server. Since the game is played in short rounds, actions performed on the game rarely have any impact, as every rounds resets all player data, with the exception of character money earned and used in the game. As such, the only demands server-side fast position updates broadcasting and resolving conflicts in players actions, commonly due to network latency. The conflicts are resolved by the server by rewinding time and verifying missed or late client packets, and any inconsistencies detected are rolled back and broadcasted to players.

B. OpenSimulator

OpenSimulator [19] is an open source server to Linden Labs' Second Life's [20] client, which was released as open source. Albeit there is no clear documentation on how the actual Second Life server is designed, OpenSimulator follows the protocol developed by Linden and hints as how the internal design of a Second Life server might be. It relies on spatial partitioning of fixed-size regions (256 meters squared), and clients communicate directly to the region servers. It has a simple service distribution model, separating some application concerns (e.g. voice, chat, asset, inventory) in peripheral servers. In many circumstances, the user directly access these peripheral servers to perform desired actions, and the region's simulation is only responsible for handling access rights (e.g. granting access and modify capabilities to a user over an asset).

The major difference in OpenSimulator and Second Life to most other MMVEs is allowing user generated content (UGC). Users are allowed to create, modify, and delete every object in the world given the appropriate permission. The consequence of this freedom is larger quantities of TSO, due to the possibility of objects being modified by many users. To support ACID properties in the operations on game objects, OpenSimulator relies heavily on transactional databases with a custom caching solution as the backend. For avatar location updates, cache is used to provide faster response times. The maximum number of users reported in a single region borders near 100 users.

C. Distributed Scene Graph

The Distributed Scene Graph (DSG) [14] implementation is based on OpenSimulator, and presents a more decoupled architecture than seen on its parent. It introduces the Client Manager layer and creates a stronger Distributed Service decoupling, removing heavy processing services such as Physics and Script Engine from the core game loop. In [14], 1000 connected avatars were able to interact at acceptable frame rates.

D. Open Wonderland (Darkstar)

Open Wonderland [21] is a Java open source toolkit for creating 3D environments, similar to OpenSimulator. Running on top of the middleware of DarkStar [6], it has geographical independence at it's core: no requirement of spatial partitioning from the developer's front. Darkstar also provides a Client Manager interface and caching capabilities, where it is particularly strong, using an optmized database for 50% write, 50% read. Even with geographic independence, Open Wonderland allows only 50-60 users in the same region, while providing a high interactivity environment.

E. EVE Online

EVE Online [10] presents dynamic space partitioning through its use of causality bubbles, but is still restrained to having servers dedicated to space partitions. In the event of battles, more resources can be dedicated to a particular region, but there is no load balancing between regions (i.e. solar systems). A client manager interface is also present, as can be seen in Figure 1. For data access, EVE Online reports utilizing a proprietary Object Oriented Remote Procedure Call (OORPC) solution on top of a SQL Cluster.

F. Robots Versus Tanks (Pikko Server)

Robots Vs Tanks is an FPS created for testing the Pikko Server [11]. Using dynamic partitioning through the "Mast Algorithm", the Robots Vs Tanks achieved 999 players [22]. Because it is still space partitioned, there is little interactivity between users for distant regions, and thus its space partitioning algorithm works very efficiently. It also present a Client Manager layer (Figure 1 named ALG (i.e. Application Level Gateway), and caches all data of the zone it controls in memory for fast access and response.

G. World of Warcraft

World of Warcraft [23] by Blizzard Entertainment is a well known Massively Multiplayer Online Role-Playing Game released in 2001. Due to its commercial proprietary nature, very little is known about its architecture, but to the best of our knowledge, it performs dynamic and static space partitioning and caches in memory data that needs fast access. Even though

TABLE I
SCALABILITY IN PRACTICE

Synchronization Scalability in Practice					
Application	Cache	C.M.	D.S.	G.I.	D.S.P.
Counter-Strike*					
OpenSimulator	•		•		
Distributed Scene Graph	•	•	•		
Open Wonderland	•	•		•	
EVE Online*	•	•	•		•
Robots Vs Tanks*	•	•			•
World of Warcraft*	•		•		•

C.M.: Client Manager, D.S.: Distributed Services, G.I.: Geographic Independence, D.S.P.: Dynamic Space Partitioning.
*: To the best of our knowledge.

Battle.Net could provide the service of a Client Manager layer, currently the crash of a game server also results in crash of clients. It is not clear what services are uncoupled from the game core, but the WoW Armory service [24] suggests the decoupling of game objects from the game servers into separate databases. The WoW Armory is a website that allows users to see in-game equipments, thus suggesting an API access to game data in a SOA manner.

IV. FUTURE DIRECTIONS

A. Interactivity Dynamic Partitioning

Darkstar depends upon locality of data requests being mostly in the same node to provide good performance. An optimized approach could potentially attempt to put requests of same data items in the same node, as suggested in [6]. As such, we propose an approach not based on tasks, but using the same concept of centralized data server with write access permissions as Darkstar. Yet instead of task oriented, data items are seen as objects that are owned by nodes. Given a certain ID value, any object can be looked up in the central data server and retrieved by any of the nodes (i.e. game servers). The owner node of an object can accept write and read requests and communicates back through the network.

This approach is similar to DMC. But in DMC, "objects" are stored based on static algorithms (e.g. hashing in Memcached or location in spatial partitioning). Instead, objects should be relocatable to other nodes to distribute the load of active and demanding objects, based solely on the load (i.e. interactivity of clients and the world entities), which is a dynamic runtime factor. The challenge, and future research direction, becomes how to organize client requests and objects to be as collocated as possible so the load on each node is balanced and the region border effect is reduced. The punishment for performing requests to different nodes is the added network latency involved, which is also a problem in Darkstar, but in this approach the nodes have the power to reorganize data access requests to be as collocated as possible. In essence, there is a need for a DMC with configurable algorithm for locating and accessing data in the distributed nodes.

B. Virtual Environments and the Web

Presently web servers are optimized for high availability for tens or hundreds of thousands of users. Usual Web applications have little demand to read and modify data at fast speeds, so no consistency is needed. In case of conflicts, which are rare, users are instructed to retry operations.

The Web has become a platform for interactive social applications, with Facebook and Twitter leading phenomena of social networks. The new interactive Web requirements seen in Facebook and Twitter are becoming similar to MMVEs: fast rate synchronized state updates, interest management (e.g. friends list), and backend distributed service oriented architectures for data analytics [25] to name the most important. The update rate is many times slower than a virtual environment, but the number of users sharing the same resources is much higher. Web protocols are already proving to be too slow for users, who demand more consistent and immediate interaction. Additionally, the social networks are adding multiuser gaming (e.g. Facebook apps), which is essentially the traditional network gaming on top of HTTP.

In light of these events, new protocols of bidirectional communication in the manner of TCP are being developed and approved, the most known being Websockets [26]. The bidirectional communication protocols will bring faster interaction to the Web, but without the constraints from REST, fault tolerance and scalability will no longer be an automatic feature. There is a need for research directed at the highly interactive social Web that is growing, melding traditional REST and HTTP with full bidirectional TCP and sockets.

V. CONCLUSION

Creating immersive virtual environment that enables massive user interactions remains a great challenge to computer science community, possessing both accidental and essential complexity characteristics. State synchronization and the N^2 problems remain essential, yet scalability can still be achieved under a set of constraints, albeit at the expense of side effects as higher latency and application design restrictions.

The scalable solutions practiced by the MMVE applications discussed in this paper have similar inspiration, but entirely separate development processes. It demonstrates the need of proper research of software architectures for massive multiuser networked applications not only as a guide for new applications, but also as a collaborative mean between the industry seniors and the academy research.

The growth of social media in the Web has inadvertently brought similar challenges that were familiar to MMVEs, with low latency multi user interaction, state synchronization, and interest management being the most representative. Technologies in the form of fast bidirectional connections (e.g. Websockets) are in the process of becoming standards, and yet there is research to be done to improve such multi-user applications. Understanding the challenges of MMVE research has become fundamental to welcome the new age of fast, collaborative, and socially interactive environment in the Web.

REFERENCES

[1] I. Sutherland, "The ultimate display," *Multimedia: From Wagner to virtual reality*, 1965.

[2] C. Carlsson, "DIVE A multi-user virtual reality system," *Virtual Reality Annual International*, pp. 394–400, 1993. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=380753 http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=380753

[3] R. Waters, D. Anderson, J. Barrus, and D. Brogan, "Diamond park and spline: A social virtual reality system with 3D animation, spoken interaction, and runtime modifiability," 1996. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.2.9500

[4] L. Itzel, F. Heger, G. Schiele, and C. Becker, "The quest for meaningful mobility in massively multi-user virtual environments," *2011 10th Annual Workshop on Network and Systems Support for Games*, pp. 1–2, Oct. 2011. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6080991

[5] K. Chen, P. Huang, and C. Lei, "Game traffic analysis: An MMORPG perspective," *Computer Networks*, vol. 50, no. 16, pp. 3002–3023, 2006. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S1389128605003956

[6] T. Blackman and J. Waldo, "Scalable Data Storage in Project Darkstar," Tech. Rep., 2009.

[7] W. White, C. Koch, N. Gupta, J. Gehrke, and A. Demers, "Database research opportunities in computer games," *ACM SIGMOD Record*, vol. 36, no. 3, pp. 7–13, Sep. 2007. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1324185.1324186

[8] S. Paul and Z. Fei, "Distributed caching with centralized control," *Computer Communications*, vol. 24, no. 2, pp. 256–268, Feb. 2001. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0140366400003224

[9] E. A. Brewer, "Towards Robust Distributed Systems," pp. 7–10, 2000. [Online]. Available: http://www.cs.berkeley.edu/ brewer/cs262b-2004/PODC-keynote.pdf

[10] D. Brandt, "Scaling EVE Online, under the hood of the network layer," 2005. [Online]. Available: www.research.ibm.com/netgames2005/papers/brandt.pdf

[11] D. Almroth, "Pikko Server," in *Erlang User Conference*, 2010. [Online]. Available: http://www.erlang-factory.com/conference/ErlangUserConference2010/speakers/DavidAlmroth

[12] S. Harcsik, A. Petlund, C. Griwodz, and P. l. Halvorsen, "Latency Evaluation of Networking Mechanisms for Game Traffic," in *Netgames*, 2007.

[13] P. Svoboda, W. Karner, and M. Rupp, "Traffic Analysis and Modeling for World of Warcraft," *2007 IEEE International Conference on Communications*, pp. 1612–1617, 2007. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4288941

[14] D. Lake, M. Bowman, and H. Liu, "Distributed Scene Graph to Enable Thousands of Interacting Users in a Virtual Environment," in *Netgames*, 2010, pp. 2–7.

[15] S. Koo, "How to Support an Action-Heavy MMORPG from the Angle of Server Architecture," in *CGDC*, 2010.

[16] B. Van Den Bossche, B. De Vleeschauwer, T. Verdickt, F. De Turck, B. Dhoedt, and P. Demeester, "Autonomic microcell assignment in massively distributed online virtual environments," *Journal of Network and Computer Applications*, vol. 32, no. 6, pp. 1242–1256, Nov. 2009. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S1084804509000575

[17] A. Restrepo and A. Montoya, "Dynamic server allocation in virtual environments, using Quadtrees for dynamic space partition." *Proceedings of IASTED Computer*, 2003. [Online]. Available: http://www1.eafit.edu.co/rvirtual/Publications/iastedCST2003-159.pdf

[18] Valve, "Source Multiplayer Networking." [Online]. Available: https://developer.valvesoftware.com/wiki/Source_Multiplayer_Networking

[19] OpenSimulator, "OpenSimulator." [Online]. Available: http://opensimulator.org

[20] Linden Labs, "Second Life." [Online]. Available: http://secondlife.com

[21] J. Kaplan and N. Yankelovich, "Open Wonderland : An Extensible Virtual World Architecture," no. January 2010, 2011.

[22] J. Reahard, "Networking firm attempts 1000-player FPS battle," 2011. [Online]. Available: http://massively.joystiq.com/2011/03/09/networking-firm-attempts-1000-player-fps-battle/

[23] Blizzard Entertainment, "World of Warcaft." [Online]. Available: http://us.battle.net/wow/en/

[24] ——, "World of Warcraft Armory." [Online]. Available: http://wowarmory-us.com/

[25] A. Thusoo, Z. Shao, S. Anthony, D. Borthakur, N. Jain, J. Sen Sarma, R. Murthy, and H. Liu, "Data warehousing and analytics infrastructure at facebook," *SIGMOD '10*, p. 1013, 2010. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1807167.1807278

[26] I. Fette and A. Melnikov, "The WebSocket Protocol," 2011. [Online]. Available: http://tools.ietf.org/html/rfc6455