An Analysis of Increased Vertical Scaling in Three-Dimensional Virtual World Simulation

Sean C. Mondesire, Ph.D.
U.S. Army Research Laboratory
3100 Technology Pkwy
Orlando, FL 32826
sean@cs.ucf.edu

Jonathan Stevens, Ph.D.
University of Central Florida
3100 Technology Pkwy
Orlando, FL 32826
jonathan.stevens@knights.ucf.edu

Douglas B. Maxwell
U.S. Army Research Laboratory
12423 Research Pkwy
Orlando, FL 32826
douglas.maxwell3.civ@mail.mil

ABSTRACT
In this paper, we describe the analysis of the effect of vertical computational scaling on the performance of a simulation based training prototype currently under development by the U.S. Army Research Laboratory. The United States military is interested in facilitating Warfighter training by investigating large-scale realistic virtual operational environments. In order to support expanded training at higher echelons, virtual world simulators need to scale to support more simultaneous client connections, more intelligent agents, and more physics interactions. This work provides an in-depth analysis of a virtual world simulator under different hardware profiles to determine the effect of increased vertical computational scaling.

General Terms
Measurement, Performance, Design, Experimentation, and Verification.

Keywords
Benchmarking; Vertical Scaling; Virtual World Simulation; Performance Analysis.

1. INTRODUCTION
The U.S. Army Research Laboratory's (*ARL*) Human Research and Engineering Directorate (*HRED*) has led an extensive investigation of the effectiveness of military personnel training within a virtual world simulation [1]. Due to the requirements set forth by the military for accurate and high-fidelity simulations [2], considerable effort is being made to increase scalability and performance in these virtual military trainers. To date, attempted horizontal and vertical scaling in these simulators have marginally increased the number of human trainers the simulators can concurrently support and improved simulator fidelity. Unfortunately, these limited successes have left the simulation and training community without proven methods and analyses that easily transform existing virtual worlds to meet the desired scale, accuracy, and performance levels required to conduct military training.

This work extended prior vertical scaling research and maintained the goal of increasing scalability in a virtual world simulator. The work extended prior vertical scaling efforts by rigorously focusing on performance differences achieved when an increased amount of processing units were given to a virtual world simulator. In the previous vertical scaling investigation, it was found that a doubled total amount of CPU cores allocated to a simulator server had a greater positive effect on scalability than when the simulator's servers were upgraded to more modern CPU technology [3].

To expand the previous research, this presented work quantifiably measured the impact of increased vertical scaling on simulator performance when 1, 2, 4, 8, 16, and 32 CPU cores were dedicated to a simulator. This new vertical scaling analysis provides a formal analysis of CPU vertical scaling in a virtual world simulator to the simulation and training community. Furthermore, this work provides a set of expectations that couple simulator performance with increased CPU capabilities due to vertical scaling. Finally, this work provides further evidence of the accuracy of using CPU utilization as a gauge for simulator performance and a detection measure for system duress.

The findings of this work are presented as follows: first a background of the military's need for accurate, high-echelon virtual training is discussed, along with the classification of simulators and a description of the studied MOSES project and OpenSimulator. Next, the methodology and experiments for analyzing increased vertical scaling with CPU processing capabilities is defined. We then present the results of the experiments and conclude this work with the main observations and this research's follow-on work.

2. BACKGROUND
The United States Army continues to expand its use of simulation for training in novel ways [4]. One of these new approaches is the use of persistent virtual environments for training, employed in a distributed manner. While virtual world technology and training is still a relatively new domain [5], the Army has been an early supporter of this technology [6]. Recent advancements in simulation technology have enabled the rendering of sufficiently realistic virtual environments that can support realistic training [7]. The focus of this research effort was to analyze various methods and server configurations to both improve and optimize the user's distributed virtual environment experience.

2.1 Simulation Classification
There are four classes of simulation used for training: live, virtual, constructive and gaming [8] [9]. *Live simulations* comprise real people operating real systems while *virtual simulations* encompass real people operating simulated systems in simulated environments. *Constructive simulations* involve simulated people operating simulated systems. *Game-based* simulation involves the employment of interactive, computer-based applications used for training purposes [10]. At times, the four classes may overlap

SIMUTOOLS 2015, August 24-26, Athens, Greece

DOI 10.4108/eai.24-8-2015.2260963

each other during the same training event; this phenomenon is referred to as "blended training".

One of the major advantages of employing virtual, constructive and game-based simulation for training are their associated cost advantages [11] [12] especially when compared to live training. In the current fiscal environment, this is of paramount importance as budgets for training continue to decline. Due in large part to its distributed nature, virtual world training represents a major potential cost reduction candidate for military training [13] because users may train together without having to travel to the same location. This variation of virtual simulation for training is still developing, as evidenced by the absence of any embedded instructional strategy [14] and training effectiveness framework [15].

In recognition of the effectiveness of simulation-based training [16] [17] [18], the *Army's Learning Model* explicitly calls for the increased use of both virtual and game-based training. Both of these classes of simulation allow the Army to maintain Soldier proficiency in critical skills at reduced cost [19]. Doing so in a distributed manner, such as a persistent virtual environment, represents another cost efficient method to support the Army's Learning Model.

A persistent virtual environment, also referred to as a virtual world, is an "immersive simulated environment in which a participant uses an avatar (a digital representation of oneself) to interact with digital agents, artifacts, and contexts" [20]. Up to the present, virtual worlds have been primarily employed for training either at the individual level, focused on tasks such as public speaking anxiety [21], or for collective learning activities, such as a classroom, conference or virtual event. In one such example, the Linden Lab's Second Life® virtual environment was used to host approximately 800 conference attendees attending a virtual, distributed conference [22].

2.2 MOSES

ARL's HRED has created the *Military Open Simulator Enterprise Strategy* (*MOSES*) project to develop a three-dimensional training simulation platform [23]. The goal of the research project is to determine whether or not persistent virtual environments are suitable for collective, military training. The MOSES research team continues to examine the suitability of virtual worlds for collective military training through the software development of essential capabilities, such as realistic models (weapons, ammunition and their effects), terrain and artificial intelligence as well as the optimization of the architecture required to support distributed, non-deterministic training. The focus of this paper is the latter goal, architecture optimization. The MOSES virtual environment is depicted in Figure 1.

Figure 1: MOSES Virtual Environment

MOSES is a simulation architecture and management system that provides its users with a controllable and expansive virtual world [23]. *OpenSimulator* (*OpenSim*) is the MOSES project's underlying software that simulates all of the virtual world's environment, interactions, and behaviors [24]. OpenSim is an open-sourced, three-dimensional, persistent virtual world simulator that is heavily inspired by the popular social virtual world, *Second Life*®. The aim of both OpenSim and Second Life® are to provide social virtual spaces where users interact with each other through simulation avatars in building-, landscape- and object-rich environments [25].

OpenSim is a cross-platform, multi-threaded simulator written in the .Net framework. A typical OpenSim instance is a simulator server that simulates and hosts a portion of the virtual environment, known as a *region*. Users interact in the region by launching a *viewer*, an external application that connects to an OpenSim server. The viewer allows the user to control an *avatar*, which represents the user in the environment. Inside a region, an avatar can move throughout the simulated environment, socialize with other avatars, create objects and appearances, and more.

The MOSES project is focused on training military tactics through these simulated activities and environments. Previous research has shown that MOSES can be an effective collective military training simulation [26] at the small unit level. As we attempt to employ MOSES at higher echelons of training, architecture limitations must be addressed. The foremost challenge remains the number of human avatars that may operate simultaneously in the virtual environment. In our prior research [3], we discovered that server configuration, specifically the number of cores dedicated to the MOSES server, had the greatest effect on simulation performance than any other variable examined. Expanding on this discovery, this work seeks to uncover the relationship between MOSES' scalability and vertical scaling with an increased amount of processing cores. Succinctly, this follow-on work determines the effectiveness of an increased vertical scaling method with a popular virtual world simulator. In this paper we discourse on the experimentation we conducted to support the aforementioned goal.

3. METHODOLOGY

To further expand the vertical scaling discoveries made in prior research, the presented work measured OpenSim server performance on a series of increasing amounts of CPU capabilities. Specifically, the analysis compared simulator performance when the server was allocated 1, 2, 4, 8, 16, and 32 CPU cores. Analyzing performance in this manner allowed for the effect of CPU total vertical scaling to be determined on a virtual world simulator. This analysis was accomplished in two phases: 1) a minimum degradation load phase and 2) increased vertical scaling phase. All tests for both phases used the same hardware configuration but varied in number of CPU cores allocated to the server. Both phases were analyzed for two independent experiments, where each experiment differed in the tested CPU type (brand and model).

3.1 Phase 1: Minimum Degradation Load

The first phase identified the minimum load each CPU type could support with one CPU core to trigger noticeable degradation in the simulation's user experience. To make this identification, an OpenSim instance was launched with one CPU core. Automated users (*bots*) were incrementally allowed to connect to the simulator with a 30 second delay interval until the simulator's

averaged host CPU utilization percentage first exceeded 95%. At that point, the number of bots in the region was noted as the *minimum degradation load* (*MDL*). This MDL value represents the highest amount of load the simulator can process before triggering simulation experience degradation. From previous vertical scaling investigations, it has been noted that when the simulator's host CPU utilization percentage average exceeded 95%, the user experience began to suffer due to simulation lag, avatar rubber banding, and unresponsiveness from the simulator. This phase was performed on each of the two CPU types when OpenSim was allocated a single CPU core. Table 1 displays the hardware configuration for the two phase 1 experiments.

Table 1: Phase 1 Hardware Configuration

Test Case	CPU Type	# of Cores	Memory (RAM)
1	AMD Opteron 6100	1	64 GB
2	Intel Xeon E7-4890v2	1	64 GB

Table 2: Phase 2 Legacy Hardware and Bot Configuration

Test Case	CPU Type	# of Cores	Memory (RAM)	# of Bots (MDL)
L1	AMD Opteron 6100	1	64 GB	45
L2	AMD Opteron 6100	2	64 GB	45
L3	AMD Opteron 6100	4	64 GB	45
L4	AMD Opteron 6100	8	64 GB	45
L5	AMD Opteron 6100	16	64 GB	45
L6	AMD Opteron 6100	32	64 GB	45

Table 3: Phase 2 Prototype Hardware and Bot Configuration

Test Case	CPU Type	# of Cores	Memory (RAM)	# of Bots (MDL)
P1	Intel Xeon E7-4890v2	1	64 GB	59
P2	Intel Xeon E7-4890v2	2	64 GB	59
P3	Intel Xeon E7-4890v2	4	64 GB	59
P4	Intel Xeon E7-4890v2	8	64 GB	59
P5	Intel Xeon E7-4890v2	16	64 GB	59
P6	Intel Xeon E7-4890v2	32	64 GB	59

3.2 Phase 2: Increased Vertical Scaling

The second testing phase identified OpenSim performance as a result of increased vertical scaling. To accomplish this goal, different hardware configurations of 1, 2, 4, 8, 16, and 32 CPU core allocations were each compared. The comparison was based on how well each CPU allocation performed with the minimum degradation load for each CPU type. Tables Table 2 and Table 3 outline the hardware configurations for the two phase 2 experiments.

3.3 Hardware Configuration

The two experiments independently evaluated vertical scaling with different CPU types. Experiment 1 evaluated OpenSim's scalability on a legacy 2010 Dell R815 server with 48 AMD Opteron 6100 CPU cores and a total of 512 GB of RAM. Experiment 2's evaluation took place on a prototype 2015 Intel server with 120 Intel Xeon E7-4890v2 CPU cores and 1.5 TB of RAM. To allocate the 1-32 cores for each test, the hypervisor Proxmox was used. Proxmox provided the capabilities to create Ubuntu 14.01 Server virtual machines with specific CPU and memory hardware configurations. In all tests, 64 GB of RAM was allocated to each virtual machine.

3.4 Simulator Load

In most multi-user simulation and video games, connected users represent the most scalability load on the system; OpenSim is no different. In OpenSim, each logged in user requires an avatar presence in the region, causing the simulation to keep track of the avatar's location, appearance, and activities. In addition, avatar information is also synchronized to the other avatars and their viewers. This synching increases the demand for CPU processing, memory usage, and network bandwidth to process and distribute world updates between the simulator and connected viewers.

In previous OpenSim studies, automated bots have been used to represent human-users and their associated server load. With scripted bots, the simulated avatars are OpenSim objects that perform predetermined actions that model human, animal, robot, or other behaviors [27]. Bot automated behaviors include movement, instant messaging, building construction, and more. OpenSim supports scripted bots through the Linden Labs scripting language to automate these behaviors in the virtual space. Bots have been used to test simulator load, populate virtual regions for crowd modeling, and provide the capability to allow human-operators to interact with the artificially controlled bots for social, gaming, and training scenarios.

The *Distributed Scene Graph* (*DSG*) research proposed and made changes to OpenSim by decentralizing core functionality of the simulator to multiple hosts [28]. The DSG experiments uncovered that the traditional bot technology widely used in OpenSim does not accurately represent true load on the simulator. The reason is that scripted bots do not have the same presence on the simulator as a human-controlled avatar would. For instance, network traffic is not transmitted and synchronized between the simulator server and a separate machine hosting the bot. Second, physics calculations for collision detection and gravitational forces for individual bots are often not performed due to how the simulator handles scripted, non-physical objects. Third, in the DSG experiments, hundreds of bots were supported on a single region, while 63 human-controlled avatars was the maximum amount the simulator could support on the same hardware and test

configuration. These aspects and result differences conclude that a traditional scripted bot generates less CPU processing, memory, and network bandwidth resource demands on the simulator than required by a human-user. Unfortunately for scalability and load testing, the low-resource footprint of a bot is not an accurate representation of user load on the simulator.

To remedy the inaccuracies of the traditional scripted bots, this paper employed heavy-weight, realistic load bots to correctly measure simulator performance with high performance demands. In both of this work's experiments and all of their tests, new, modified automated bots were used to accurately represent human-user load onto the OpenSim server. These bots were realized through a modification of the open-sourced Phoenix Firestorm Viewer application [29]. With the modification, bots were able to automatically log into the simulator and interact with the virtual environment. Bot interaction included movement behaviors: moving forward, turning backwards, left, and right, flying, crouching, and teleporting to various points in the region. These bots were purposely constructed to generate similar load profiles of a typical human user. By modifying the Phoenix Firestorm Viewer, the new bots were able to be controlled via scripting, perform a random behavior at second intervals, transmit and receive simulation updates via network messaging, and automatically log into the simulation; these capabilities have resulted in controllable bots with comparable scalability and load testing footprints to human-users.

During the experiments, once an OpenSim instance was running and the simulator stabilized to accept incoming connections, bots were launched sequentially with a delay between each launch. Each bot required one new instance of the modified viewer to run and represented one avatar in the simulation. All of the bots were launched on two separate 2010 Dell R815 servers that were independent of the one hosting the legacy server experiment tests. Both of these bot servers were also hosted on a separate network from the OpenSim server.

3.5 Metrics

During the experiments, several metrics were collected that quantifiably measured various performance attributes of the simulator. The previous vertical scaling work compared the usefulness, accuracy, and sensitivity of the Simulator Frames per Second to Host CPU utilization percentage. *Simulator Frames per Second* (*SimFPS*) is the rate at which the simulator can process a single simulation frame in one second. Inside a simulation frame (*Sim Frame*), the simulator processes all events necessary to update the virtual world, including network messages, user logins, scripting, appearance updates, etc; physics calculations such as collision detection, gravity, and forces are calculated separately by the physics engine and comprises a *physics frame*.

Traditionally, many video games and simulations target 60 frames per second (*FPS*) as an industry standard of performance stability; originally, OpenSim was no different. In fact, the OpenSim community has accepted a standard range of 50-57 FPS to signify a stable simulator. When an OpenSim instance reported a SimFPS below approximately 47, simulation lag and a deteriorated user experience was expect. To extend the earlier definition under current OpenSim, SimFPS was calculated by taking the amount of simulation frames the simulator processes in a second, which is 11.3 during optimal, no load conditions, and multiplies it by a static correction factor of 5. This presented work has removed the correction factor (akin to using a correction factor of 1) because the use of the multiplier artificially inflates performance reporting with no added benefit. Furthermore, during the execution of phase 1 for the presented experiments, it was observed that the modified SimFPS calculation improved frame rate calculation fidelity and also reflected simulator performance. With the new SimFPS calculation, when the rate falls below 9 FPS, simulation lag and unresponsiveness begins to take place.

Host CPU Utilization Percentage (Host *CPU%*) is the average of the sum of percentage of CPU usage over the total available CPU of all available cores. In the previous research, Host CPU% proved to be more sensitive to the increased vertical scaling than SimFPS and higher Host CPU% correlated with instances of when the simulator's scalability and user experience began to degrade.

Further observations from the previous research has encouraged this presented work to use host CPU utilization percentage as opposed to process CPU utilization. The distinction is that host CPU utilization provides a complete quantifiable measure of how the entire simulator's CPUs are performing for all parts of the system. *Process CPU utilization* is the sum of each core's total CPU usage from a single process (in this case OpenSim.exe) divided by the total available CPU usage. Similar to host CPU utilization, process CPU utilization can be greater than 100%. Traditionally, MOSES only runs OpenSim and core Ubuntu services on a simulator's host virtual machine. This standalone method of hosting a simulation means host CPU utilization provides a measure of all processing that is needed to run an OpenSim simulation, where process CPU utilization omits critical operating system factors in its reporting.

Other OpenSim metrics are total frame time and time dilation. *Total Frame Time* (*TotFT*) is the amount of time the simulator requires to process both the simulation and physics frame and an optional sleep period. The sleep period is used to throttle the simulation from processing simulation events too quickly and is relied on for synching the state between the simulation and the viewers. *Time dilation* (*Dil*) is the rate the simulation throttled. Dilation values below or equal to 1 are intended to indicate that the simulation is running at normal levels; dilation values over 1 and the simulation and physics calculations are taking longer than the expected frame time.

Although avatar count, time dilation, Sim FPS, Phys FPS, Host CPU%, and total frame time are all gathered for every test, this work focuses on simulator performance with the Host CPU% measure. During the minimum degradation load phase, these metrics are gathered at one second intervals for 30 seconds when each avatar logs in before launching the next bot. During the vertical scaling phase, the metrics are gathered at one second intervals for 50 seconds once the minimum degradation load's amount of avatars have logged in.

4. RESULTS

This experiment employed one independent variable, the number of cores allocated for server operations. The independent variable had six levels, with the number of cores allocated ranging from 1 to 32, increasing in exponential increments. The dependent variable was Host CPU Utilization Percentage (Host CPU %), defined as the average CPU usage of the total available CPU of all available cores. The experiment was executed twice - once for each hardware configuration previously described. For each hardware configuration, server performance was captured at 50 discrete intervals. The experimental procedure was executed identically for each server and core configuration.

4.1 Legacy Server Results

The first experiment executed used the 2010 Dell R815 server with 48 AMD Opteron 6100 CPU cores and a total of 512 GB of RAM. Performance metrics were collected in the same manner for all six core levels. Table 4 and Figure 2 depict the server's average Host CPU% for all six core levels.

Table 4: AMD Legacy Host CPU %

	1 Core	2 Core	4 Core	8 Core	16 Core	32 Core
Average	80.4	54.5	43.7	33.2	17.3	9.1
SD	7.0	5.9	4.0	1.4	1.4	0.8

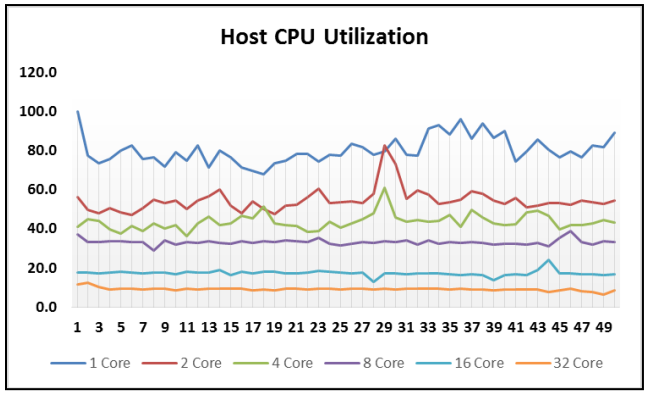

Figure 2: AMD Legacy Host CPU Utilization

ANOVA found a significant main effect of number of Cores on Host CPU Utilization $F(5, 299) = 1940.43$, $p = 0.00$. ANOVA was conducted at $\alpha = 0.05$. Post-hoc pairwise comparisons using the Tukey HSD test indicated that the mean Host CPU Utilization of all six Core configurations were significantly different from each other, $p < 0.0001$. Host CPU Utilization was found to significantly decrease with each corresponding increase in core count. Relationally, the Host CPU Utilization of 1 Core > 2 Core > 4 Core > 8 Core > 16 Core > 32 Core.

4.2 Prototype Server Results

The second experiment executed used the 2015 Intel prototype server with 120 Intel Xeon E7-4890v2 CPU cores and 1.5 TB of RAM. Performance metrics were collected in the same manner for all six core levels. Table 5 and Figure 3 depict the server's average Host CPU% for all six core levels.

Table 5: Intel Prototype Host CPU %

	1 Core	2 Core	4 Core	8 Core	16 Core	32 Core
Average	91.2	55.1	44.5	31.6	18.9	9.3
SD	7.0	8.3	6.7	2.8	1.0	0.9

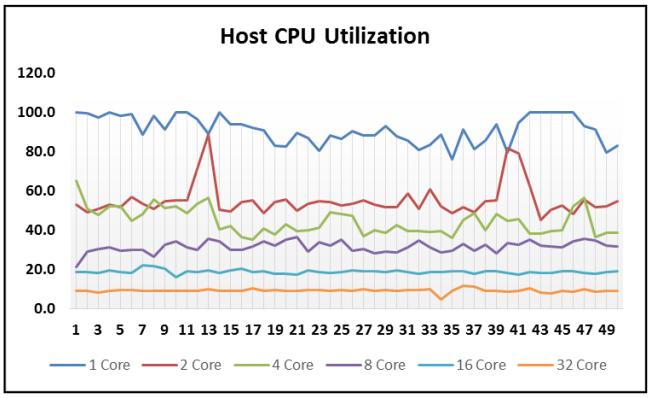

Figure 3: Intel Prototype Host CPU Utilization

ANOVA found a significant main effect of number of Cores on Host CPU Utilization $F(5, 299) = 1508.37$, $p = 0.00$. ANOVA was conducted at $\alpha = 0.05$. Post-hoc pairwise comparisons using the Tukey HSD test indicated that the mean Host CPU Utilization of all six Core configurations were significantly different from each other, $p < 0.0001$. Host CPU Utilization was found to significantly decrease with each corresponding increase in core count. Relationally, the Host CPU Utilization of 1 Core > 2 Core > 4 Core > 8 Core > 16 Core > 32 Core.

5. DISCUSSION

In this experiment we discovered similar performance results for two separate server hardware configurations. As the number of cores allocated to server operations increased, the Host CPU Utilization performance metric significantly decreased, for both servers. For both server hardware configurations, CPU % significantly decreased for every subsequent increase in the core count. A direct performance comparison was made between servers, although the minimum degradation load was different for each server. These results are depicted in Table 6 and Figure 4.

Table 6: 95% Confidence Intervals of Mean Performance Difference between Servers

Server	1 Core	2 Core	4 Core	8 Core	16 Core	32 Core
Legacy	80.4	54.5	43.7	33.2	17.3	9.1
Prototype	91.2	55.1	44.5	31.6	18.9	9.3
Difference	-10.8	-0.6	-0.8	1.7	-1.6	-0.3
95% CI	(-13.6, -8.04)	(-3.4, 2.3)	(-3.0, 1.4)	(0.8, 2.5)	(-2.0, -1.1)	(-0.6, 0.1)

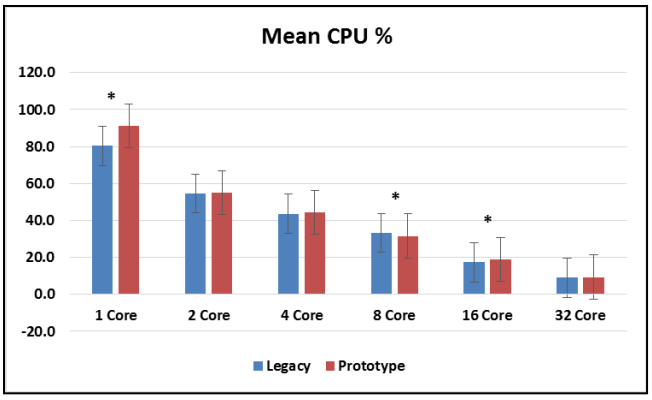

Figure 4: Mean Performance Comparison between Servers

We discovered significant performance differences between the two server configurations when allocated 1, 8 and 16 cores. No performance differences were found between the server configurations when allocated 2, 4 and 32 cores. We attribute the difference in Host CPU % to the varying minimum degradation load between each server and instead focus our analysis on the trend comparison observed with both server configurations.

While not surprising, these results confirm and support the results of our previous study. Namely, the number of cores dedicated to server operations has a practical, and statistical, significant effect on performance. This confirmation of our previous work will provide valuable insight for the future direction of the MOSES architecture as we seek to optimize the virtual world experience for our military users.

6. CONCLUSION

The presented work analyzed the scalability and system performance of increased vertical scaling on a three-dimensional virtual world simulator. To perform the analysis, this research extended a previous vertical scaling investigation by evaluating the performance effects of when CPU capabilities are increased exponentially to the open-sourced simulator OpenSim.

The prior investigation uncovered that an increased amount of CPUs dedicated to a simulator has a greater positive performance effect than vertically scaling up to more modern CPU technology. Additionally, the metric of host CPU utilization was discovered to be sensitive enough to be an accurate measure of system load in the CPU-intensive simulator. Extending on those observations, this work evaluated OpenSim performance for its host CPU utilization percentage when the simulator experienced performance-affecting scalability load. Experiments under two different CPU types have shown that each exponential increase in the amount of CPU cores yielded a significant decrease in host CPU utilization. Although expected, these findings are noteworthy because they indicate that more CPU capabilities could result in faster simulation processing, the increased vertical scaling could yield more evenly distributed simulator load among the additional cores, and the simulator could support more load without compromising on the user experience and system responsiveness.

Additionally, this work has made a series of contributions. Collectively, this work has provided the simulation and training community with an empirical study of increased CPU capabilities by vertical scaling, a simple method of measuring system duress, and has moved the military training project MOSES and its users closer to realizing their higher-echelon training objectives. First, this work has provided a set of expectations associated with CPU vertical scaling. Now, virtual world users and developers can set expectations of their own simulator performances based on the presented OpenSim analysis. These expectations can be beneficial in determining the amount of hardware necessary to support expected simulator loads. Second, with host CPU utilization exclusively analyzed in this work, the same community has further evidence of the effectiveness of using this metric to gauge system performance and indicate periods of simulator duress. Again, host CPU utilization proved to be an accurate, sensitive metric because OpenSim is a CPU-intensive simulator. This empirical study further provides support of using the metric as a simple and useful system measure. Finally, this work and its presented experiment results move the MOSES project and OpenSim closer to accomplishing its scalability objectives set forth by the military. The progress is attributed to the project's community now having a set of expectations with this method of vertical scaling, a new understanding of the hardware costs for moving the simulator towards its higher-echelon levels, and a deeper notion of the performance improvements CPU capability increases can yield.

To continue the progress of this work, several future research investigations have been identified. Two such contributions are the creation of a prediction model and a set of benchmark tools. The prediction model will produce a mathematical function that can determine how much load a simulator can support when given a specific set of hardware capabilities. This model will be useful for MOSES, OpenSim, and virtual world simulation communities to serve a demonstration that highly accurate simulation configuration pre-planning is possible. The set of benchmark tools will allow virtual world simulation users to easily measure simulator performance and diagnose periods of system duress. Both works extend this presented work's use of vertical scaling, data collection, experiment design, and system measurement.

7. REFERENCES

[1] U.S. Army Research Laboratory, "Home," 2015. [Online]. Available: http://militarymetaverse.org/. [Accessed 13 3 2015].

[2] L. Morton, *U.S. Army Training Concept*, Department of the Army, HQ, Colonel, GS, Deputy Chief of Staff, G, 2011.

[3] S. Mondesire, J. Stevens and D. Maxwell, "Vertical Scalability Benchmarking in Three-Dimensional Virtual World Simulation (Pending Publication)," in *SummerSim Conference*, Chicago, IL, 2015.

[4] M. C. Mishkind, A. Boyd, G. M. Kramer, T. Ayers and P. A. Miller, "Evaluating the Benefits of a Live, Simulation-Based Telebehavioral Health Training for a Deploying Army Reserve Unit," *Military Medicine*, pp. (12), 1322-1327, 2013.

[5] A. Lele, "Virtual reality and its military utility," *Journal of Ambient Intelligence and Humanized Computing*, pp. 4(1), 17-26, 2013.

[6] M. Schulzke, "Rethinking Military Gaming America's Army and Its Critics," *Games and Culture*, pp. 8(2), 59-76, 2013.

[7] J. H. Wong, A. B. Nguyen and L. Ogren, "Serious Game and Virtual World Training: Instrumentation and Assessment (No. NUWC-NPT-TD-12-118)," NAVAL UNDERSEA

WARFARE CENTER DIV , NEWPORT, RI, 2012.

[8] D. D. Hodson and R. R. Hill, "The Art and Science of Live, Virtual and Constructive Simulation for Test and Analysis," *The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology*, 2013.

[9] P. Roman and D. Brown, "Games – Just How Serious Are They?," in *Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC)* , 2008.

[10] B. Bergeron, Developing Serious Games (Game Development Series), Charles River Media, 2006.

[11] J. Orlansky, C. Dahlman, C. Hammon, J. Metzko, H. Taylor and C. Youngblut, "The Value of Simulation for Training," Institute for Defense Analyses, Alexandria, 1994.

[12] M. Riecken, Powers, J. C. J., S. K. Numrich, P. M. Picucci and M. & Kierzewski, "The Value of Simulation in Army Training," in *The Interservice/Industry Training, Simulation & Education Conference (I/ITSEC)*, 2013.

[13] D. Dutta, "Simulation in Military Training: Recent Developments," *Defence Science Journal*, pp. 49(3), 275-285, 2013.

[14] J. J. Vogel-Walcutt, L. Fiorella and N. Malone, "Instructional strategies framework for military training systems," *Computers in Human Behavior*, pp. 29(4), 1490-1498, 2013.

[15] R. N. Landers and R. C. Callan, "Training evaluation in virtual worlds: Development of a model," *Journal For Virtual Worlds Research*, p. 5(3), 2012.

[16] T. M. Sotomayor and M. D. Proctor, "Assessing Combat Medic Knowledge and Transfer Effects Resulting From Alternative Training Treatments," *The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology*, 2009.

[17] T. C. Lisk, U. T. Kaplancali and R. E. Riggio, "in multiplayer online gaming environments," *Simulation & Gaming*, p. 1046878110391975, 2011.

[18] C. A. Blow, "Flight School in the Virtual Environment: Capabilities and Risks of Executing a Simulations-Based Flight Training Program," *ARMY COMMAND AND GENERAL STAFF COLLEGE*, no. SCHOOL OF ADVANCED MILITARY STUDIES, 2012.

[19] R. J. Stafford and M. W. Thornhill II, "The Army Learning Model: Changing the Way Sustainers Train," *Army Sustainment*, pp. 44(2), 28, 2012.

[20] L. Dawley and C. Dede, "Situated learning in virtual worlds and immersive simulations," in *Handbook of research on educational communications and technology*, New York, Springer, pp. 723-734.

[21] O. D. Kothgassner, A. Felnhofer, L. Beutl, H. Hlavacs, M. Lehenbauer and B. Stetina, "A virtual training tool for giving talks," *Entertainment Computing-ICEC 2012*, pp. 53-66, 2012.

[22] N. S. Shami, T. Erickson and W. A. Kellogg, "Common Ground and small group interaction in large virtual world gatherings," in *ECSCW 2011: Proceedings of the 12th European Conference on Computer Supported Cooperative Work*, Aarhus Denmark , 2011.

[23] U.S. Army Research Laboratory, "Military Metaverse," [Online]. Available: http://militarymetaverse.org/. [Accessed 14 April 2015].

[24] Overte Foundation, "Main Page," 10 11 2014. [Online]. Available: http://opensimulator.org/. [Accessed 13 3 2015].

[25] Linden Research, Inc., "Second Life Official Site," 2015. [Online]. Available: http://secondlife.com/. [Accessed 13 3 2015].

[26] S. Lackey, J. Salcedo, G. Matthews and D. Maxwell, "Virtual World Room Clearing: A Study in Training Effectiveness," in *Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC) 2014*, Orlando, FL, 2014.

[27] Linden Research, Inc, "Bot - Second Life Wiki," 16 11 2011. [Online]. Available: http://wiki.secondlife.com/wiki/Bot. [Accessed 13 3 2015].

[28] J. G. W. A. R. H. L. Douglas B. Maxwell, "A Distributed Scene Graph Approach to Scaled Simulation-Based Training Applications," in *Interservice/Industry Training, Simulation & Education Conference (I/ITSEC)*, Orlando, Fl, USA, 2014.

[29] The Phoenix Firestorm Project Inc., "Firestorm Viewer," 2015. [Online]. Available: http://www.firestormviewer.org/. [Accessed 13 3 2015].