Constructive Simulation Limitations and Cloud Scalability

Jackie Zhang, Reese Gallagher, Cris De LaPaz
Infinitas Engineering Inc.
Orlando, FL

Jackie.zhang@infinitasinc.com
Reese.gallagher@infinitasinc.com
cris.delapz@infinitasinc.com

Peter Drewes, Michael Baker, Brian McDonell
Amazon Web Services
Seattle, WA

pedrewes@amazon.com
mikejbak@amazon.com
bymcdone@amazon.com

ABSTRACT

Traditional monolithic simulation training systems have struggled to scale efficiently and meet the increasing demands of size and complexity required by the military training industry. While most legacy simulations were designed to support interoperability amongst multiple Live, Virtual, and Constructive (LVC) training systems, adding training elements to large distributed Integrated Training Environments (ITE) exercises has traditionally required purchasing and configuring new hardware and/or software, as well as a lead time of 120+ days to prepare for exercises. While this is a proven approach, it comes at the cost of large capital expenditures into hardware and software required for large-scale exercises that skew innovation-to-cost ratios.

Constructive Simulation systems have provided efficient computer-generated forces in large-scale distributed exercises, such as the Army's One Semi-Automated Forces (OneSAF). OneSAF was developed as a monolithic simulation system that does not support distributed multi-node and multi-core configuration and execution as a product of its design 20+ years ago. Scaling large LVC exercises with multiple OneSAF instances at multiple locations with increasing cyber security requirements remains challenging. With the AWS GovCloud reaching maturity and the security needed for military training exercises, it is prime time to examine the future of scalable simulations to support the PEO STRI Synthetic Training Environment (STE) architecture goals.

Our research and development efforts compare the traditional scalability approaches of monolithic simulations with that of inherent scalability available within cloud-based solutions. Based on current acquisition processes, we examine legacy simulations using OneSAF as a testbed to assess its ability to meet evolving LVC training goals. The result of this study will highlight potential configurations incorporating horizontal and vertical scaling technologies to analyze management costs and performance differentiators to the overall STE architecture while prototyping architecture to meet modern simulation and training requirements. We aim to create cloud-enabled technologies to serve the future enterprise-level hybrid LVC exercises environment.

ABOUT THE AUTHORS

Ms. Jackie Zhang has been the CEO of Infinitas Engineering since 2012, focusing on advanced software development and system integration support for US DOD training optimization. Her 20+ years of software systems engineering and research and development experiences have focused on various CGF systems development phases, especially physical and behavioral modeling, cultural modeling, and modeling Data Science techniques. Ms. Zhang received Bachelor's and Master of Science degrees in Computer Science from the University of Central Florida.

Mr. Reese Gallagher is the lead simulation engineer & researcher at Infinitas Engineering Inc., including a Bachelor of Science in chemistry from the University of Florida (UF) and a Master's in chemistry from the University of Central Florida (UCF). He has been applying chemical relationships to simulation environments.

Mr. Cristhian De La Paz is a simulation engineer & researcher working at Infinitas Engineering Inc. His background is in interactive software development, which stems from his education in Game Development at Full Sail University. He has been in the MS&T industry for over five years, leading IRAD efforts at Infinitas Engineering.

Dr. Peter Drewes – received his Ph.D. from the University of Central Florida. He has concentrated his research on performance monitoring in real-time systems. Currently, Dr. Drewes is a technical resource for Amazon Web Services, focusing on the areas of intelligent agents, sustainment, and robotic systems.

Constructive Simulation Limitations and Cloud Scalability

Jackie Zhang, Reese Gallagher, Cris DeLaPaz
Infinitas Engineering Inc.
Orlando, FL

Jackie.zhang@infinitasinc.com
Reese.gallagher@infinitasinc.com
cris.delapz@infinitasinc.com

Peter Drewes, Michael Baker, Brian McDonell
Amazon Web Services
Seattle, WA

pedrewes@amazon.com
mikejbak@amazon.com
bymcdone@amazon.com

INTRODUCTION

Legacy simulations were designed to support interoperability amongst multiple LVC training systems. However, the simulation scalability is limited by the on-hand computing, storage, and memory allocated to the simulations. Acquiring additional hardware is an event usually measured in days-weeks, not seconds-minutes. This limits large distributed ITE exercises to what can be connected and operated. With the maturity and connectivity of modern cloud environments, the scalability of resources can be achieved as needed. There are several areas of consideration that must be addressed to continue to expand constructive and virtual simulations.

To meet the required distributed agility and cost limitations, migration from the current monolithic simulation paradigm to one of modeling and simulation (M&S) services that support rapid simulation development is critical for Army modernization to keep pace. However, Army modernization cannot wait 10 to 15 years, which is the typical timeframe for building a new simulation ecosystem from scratch (Sanders & Davis, 2022).

The purpose of this paper is to discuss the common issues experienced during the scaling of constructive systems in a distributed simulation and training exercise. Distributed simulation exercises often require various diverse applications to interoperate over a network that is not controlled by a single entity. Constructive simulation systems like OneSAF were developed for training and analysis exercises. However, when the number and complexity of simulation entities in the constructive environments scale, the simulation execution will also scale until the hardware reaches computational saturation. It is difficult to scale compute or memory once the simulation has been started without impacting the performance of the existing simulation. The additional CPU load for terrain analysis, added AI/ML functionality, and additional security requirements placed on large-scale exercises are factors driving additional compute requirements

The traditional solution was to upgrade or acquire additional hardware and network infrastructures to accommodate the training and exercise requirements or to sacrifice the fidelity of the training, which is not desirable for wargaming developers and end users. This has stayed the same over the last 25+ years of OneSAF development environments. This paper explores the question of the commercial cloud environments' maturity and applicability to participate in large-scale distributed training and exercises development and execution.

Current Constructive Simulation System Scalability

Currently, most LVC training exercises run in a localized configuration with a few computers connected over Local Area Networks (LAN). Constructive simulations require more computing resources than their virtual and live counterparts because these large simulation systems simulate the complete battlefields up to a brigade level. A single user can configure and monitor the environment with a limited number of entities. As requirements on geographically dispersed distributed exercises increase to utilize Wide Area Networks (WAN), the security requirements of working with sensitive data often pose issues for configuration management and initialization of the simulation exercises. Custom-developed scripts are created to set up and run the simulations but are usually plagued with security vulnerabilities and require constant maintenance to keep updated.

OneSAF has been used by the Army to provide efficient computer-generated forces in large-scale distributed exercises for training and analysis purposes. OneSAF was developed as a monolithic simulation system that does not

support distributed multi-node and multi-core configuration and execution. Scaling large LVC exercises with multiple OneSAF instances at multiple locations with increasing cyber security requirements remains challenging. OneSAF's constructive environment is also lacking in monitoring systems. A publish/subscribe alternative to the communications protocol named Bifrost has been explored as a possible microservice. This makes it possible to perform complex functions through an external interface. The Bifrost concept assists in the shifting of monolithic simulations to those that can be distributed. The idea of being able to interface different OneSAF features allows the offloading of certain tasks to additional compute resources (Bearss & Zinser, 2021).

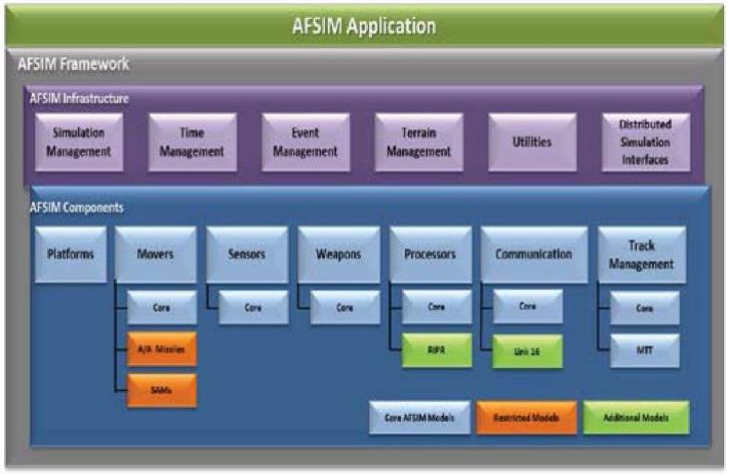


Figure 1. AFSIM Architecture (Clive, 2018)

The Air Force and Navy have also been developing federated M&S tools, such as the Advanced Framework for Simulation, Integration and Modeling (AFSIM), which has a distributed framework that enables "plug & play" federated modules. Through the smaller, loosely coupled objects, it is possible under the IDE environment to allow customizable scenario/model development and the execution of only the desired AFSIM components. This is approaching a simulation that can be effectively split and parallelized to run on cloud-based environments.

The baseline AFSIM constructive application is the Simulation of Autonomously Generated Entities (SAGE). This core constructive application batch runs a simulation based on a user-defined input file. This remote batch capability can exercise any or all of the classes or components within AFSIM. The AFSIM core is meant to run either remotely as a batch analysis or integrated into a virtual/constructive environment.

On the surface, AFSIM and OneSAF provide similar features and characteristics. However, they were designed over ten years apart, with AFSIM having a specific goal of being able to integrate development environments and organize each component into C++ classes or objects. One of the initial considerations in shifting applications such as constructive environments is the network latency that will be induced in running virtual trainers within LVC exercise environments.

Latency Concerns

Traditional simulation tasks running in the cloud must transmit outputs back for local rendering. This was due to latency in data transmission between the user and the cloud environment. "Low" latency in fast interactive simulations is considered around 10ms. The biggest portion of the usability of the simulation is not the pure latency but the transport delay that affects the simulation's quality. Human perception of stick maneuver to visual output is in the 50-100ms total time. This would allow a reasonable transport delay to utilize the cloud computation capabilities. Below is a table of representative time in milliseconds (ms) to get data from central Virginia into the cloud and back:

East US	West US	Canada central	EU Central	Gov
11.627	76.166	22.24	101.866	20.356

Figure 2. Cloud Ping Latency in milliseconds

Based on current metrics and cloud server locations, it is possible to integrate most simulation components below 60hz into a cloud environment. This assumes that the simulation on the cloud can handle the environment or can be configured to scale to handle the compute and memory load without exceeding the local network bandwidth for data transfers.

How Can Cloud Scale Constructive Simulations?

The core idea behind cloud computing is to enable users to leverage what resources they need when they need them. This is partly achieved with elastic resources, applications, and infrastructure that can be called on as needed to meet demand. This cloud-based autoscaling is used to make sure resources are utilized at the appropriate level. Autoscaling is related to the concept of temporary instances and services, which provide a baseline level of resources and then can scale up as memory and CPU use come under demand pressure. In general, there are several common attributes across all autoscaling approaches that enable automatic resource scaling. For compute, memory, and network resources, users will first deploy or define a virtual instance type that has a specified capacity with predefined performance attributes. That setup is often referred to as a launch configuration, also known as a baseline deployment. The launch configuration can be configured with options determined by what a user expects to need for a given workload in a distributed exercise.

- Scale Based on CPU/Memory/Network: With a reactive autoscaling approach, resources scale up and down as CPU/memory/network loads change. A small lag based on system monitoring is experienced as the new instances are brought online but can be minimized by proper configuration of autoscaling triggers.
- Scale Based on Prediction: Machine learning and artificial intelligence techniques are used to analyze loads and predict when more or fewer resources will be needed. This can reduce latency in starting new simulations.
- Scale Based on Schedule: With a scheduled approach, a user can define a time horizon for when more resources will be added.

For simulation applications, no additional hardware needs to be acquired using the cloud even after the simulation begins. Additional computing or storage may be added once the simulation is underway, reducing downtime and keeping simulation frame rates acceptable.

Scaling OneSAF Using Cloud Technologies

OneSAF is typically limited in the number of entities per computer, the complexity of the entities simulated and the overall simulation numbers the computer must handle. The idea of running millions of entities in large country-based exercises is still beyond the capability of most installation configurations. The experimentation is the question, can you minimally adjust OneSAF's configuration and increase its entity handling per computer configuration by using cloud-based features?

Equipped with the understanding of how scaling can be configured in the cloud to promote the elasticity of large-size distributed simulation exercises, as well as our intimate knowledge of how constructive simulation systems scale, especially the Army's OneSAF, we decided to create a testbed that enlists OneSAF as the initial wargaming simulation because it can scale up to brigade level scenarios that consist of thousands of simulated entities. We chose Amazon Web Service tools to experiment with cloud technologies and Docker tools to containerize OneSAF to deploy containers on the cloud.

Our current testbed consists of the tools listed here:

- AWS Elastic Container Service (ECS): To create instances and scaling
- AWS Elastic Container Registry (ECR): To host docker images
- Docker Engine: To create containers
- Docker Engine CLI: To create container images

Preliminary Experiments

OneSAF Containerization

While virtual machines have been the usual method for running OneSAF on remote servers, in recent times, container technology has allowed web applications to be deployed and run with less overhead. In fact, in order to take advantage of modern cloud ecosystems like AWS, containerization must be considered. Since the recommended operating system for

OneSAF is CentOS 7.6, we chose that as the base docker image. Using this as the base, we needed to create several container images of OneSAF's modules known as "compositions". Each composition was used to create an image by taking the OneSAF files and specifying the docker file with the dependencies, configuration files, and necessary scripts to run on startup (shown in step 1 of the below figure).

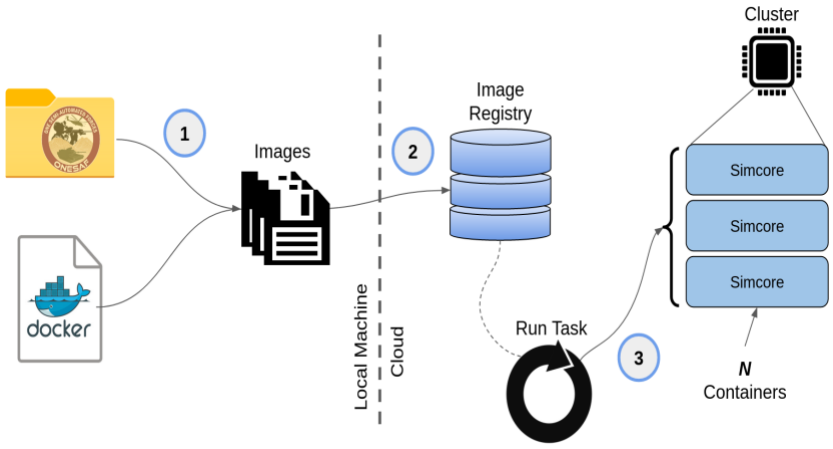


Figure 3 – OneSAF Experimentation Setup

This means that as soon as the container is launched, it runs the corresponding composition in headless mode. It should be noted that for our purposes, we were only interested in the following compositions: The User Data Gateway (UDG), the Remote Distributed Performance Monitor (RDPM), and the simcore composition. The UDG is responsible for running the process which can be accessed from the web browser. This is displayed to the user as the Web Control Tool (WCT). This tool allows a user to visualize, initialize, and run exercises. The RDPM, is a tool that runs various performance characteristics, and is what we used for all our data metrics. The simcore is a composition that simulates all the computational models and entities. While it is possible to do this with the UDG, we chose to run the UDG in a specific mode that disabled models so that only the simcores were handling the computations. This was important so that when we decided to run more than one simcore, the load could be equally parallelized.

Once the images were created, they were uploaded to the AWS Elastic Container Repository (ECR) (shown in step 2). A cluster of instances was created for each individual container. This was to ensure that each container was never vying for resources. At this point (step 3), we were able to run tasks (i.e. launch the containers) with minimal effort.

Each composition utilizes Multicast Domain Name Service (mDNS) lookup to communicate on a distributed network. This required explicit configuration of a subnet and Internet Group Management Protocol (IGMP); but this is something that most cloud providers (including AWS) allow. Additionally, each container ran a small shell command to modify the "onesaf.properties" file with the proper internal network interface address of the running instance. That said, once the tasks were launched, the containers would automatically connect with one another. This enabled us to run our exercises and collect metrics with minimal effort.

It should be noted that the images for each composition were fairly large (36 GB). While this is much less than a typical virtual machine running OneSAF (closer to 100 GB), it is still not ideal. In fact, each composition could, in theory, be reduced by creating a core OneSAF docker image, and having the compositions being very small modules. However, we chose not to attempt this space-saving optimization, as it was not the focus of the paper.

Main Experiments

Determine the Optimal Cloud Configuration

Optimality in our case, means that the application, in this case, OneSAF and its compositions, run smoothly and with varying levels of scale. Part of the appeal of using cloud is this ability to scale, but we can't say that a configuration is optimal if it doesn't take advantage of the cloud's many resources. Furthermore, if it scales, but runs prohibitively slowly due to

communication bottlenecks, then its scalability is meaningless. Therefore, a configuration is “optimal” if and only if it satisfies two criteria: scalability & performance.

For performance, we investigate the various instance types available through AWS to decide which types make the most sense for our application, and then test them in our experiments. For scalability, we will describe the methods we intend to apply for each instance and the limitations the simulation may impose on these methods.

1. Performance Focused Optimization

The following instances were the focus of our performance optimization tests: General Purpose Instances, Compute Optimized Instances, and Memory Optimized Instances. All instances utilized the highest tier networking bandwidth, as network performance was also an important factor.

General purpose instances are useful when the resources are equal proportions across the board. We decided to use this as a baseline to test against Compute Optimized and Memory Optimized.

Compute Optimized instances are made for applications that require high-performance processors. Examples include processing batch workloads, media transcoding, scientific modeling, and possibly game servers/engines. The expectation is that the OneSAF simcores would take advantage of the additional compute resources.

On the other hand, Memory Optimized Instances are tuned to workloads that process very large datasets in memory. These are your large data structures that potentially benefit from things such as cache coherence. Given the size of the docker containers (close to 36 GB) we expected this to also be a benefit.

In the first experiment, we ran an exercise with a little over 5,000 entities of opposing brigades (shown below). This number was chosen because it is not considered a large exercise but is modest enough to throttle the CPU and memory. The goal is to see which of the three instance types handles the scenario the best.

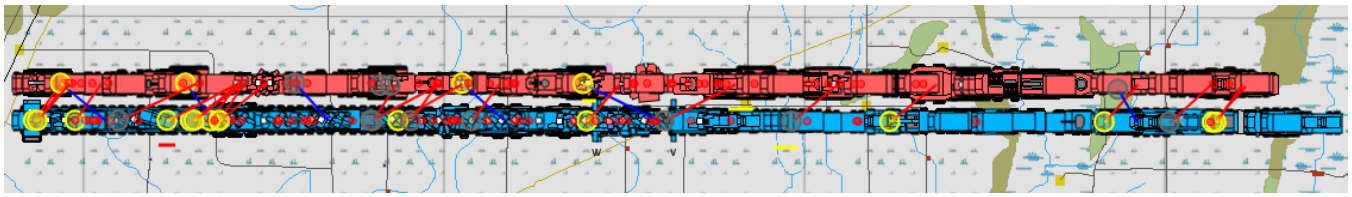


Figure 4 – OneSAF simulation of 5000 entities at close combat range

Beginning with the general-purpose instance type, we launched a container with each of the following compositions: one simcore, one RDPM, and one UDG. After the containers had established a connection, we accessed the UDG via the public IP address of the instance running the container (this is how the above image was visualized). We then initialized the scenario, waited approximately one minute, and synchronized a timer at the precise moment we began the exercise. After the startup time, the entities began their task orders, and a battle ensued. At the four-minute marker we hit stop and allowed for a one-minute cool down with the RDPM still running. At the five-minute marker, we killed the RDPM. We repeated this for the Compute Optimized and Memory Optimized instance types. The arbitrary timing was setup for this test.

Several metrics were tested from the simcore, and a handful yielded interesting results, but only one metric seemed to paint the picture of which instance type we should choose. This was the system load average. The below figure shows the average system load of a simcore across each instance type.

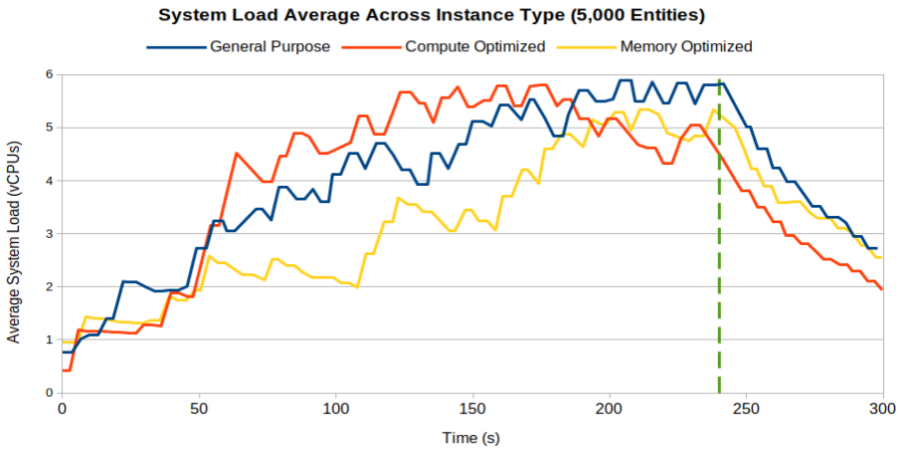


Figure 5. System Load of various configurations

Starting at time zero, the system load increases for all three instance types, but around the one-minute mark they begin to diverge, with the Compute Optimized Instance reaching its peak the fastest. At the four-minute mark (shown by the dashed line at 240 s), the simulation is stopped, and we can see that the Compute Optimized Instance trails off the fastest. It is perfectly reasonable to assume that the other instances handle this scenario just fine, however, the goal of the performance focused optimization was to find the best possible instance type to handle higher entity counts for the scalability optimization.

2. Scalability Focused Optimization

To tackle this area and devise experiments to test various configurations, we need to distinguish between horizontal and vertical scaling. Horizontal scaling implies that we will distribute the workload across multiple instances so that the resources are shared by the various compositions of OneSAF. In comparison, vertical scaling implies the ability to give any one instance more resources. To simplify the number of experiments we will focus on horizontal scaling.

Horizontal scaling means having the ability to spread the CPU and memory load across various instances in the cloud. For example, suppose we have a web server processing requests from users, the moment the server becomes taxed. In that case, the server can be duplicated on a separate instance to enable more users to engage. As we have implied thus far, OneSAF has the ability to run multiple simcores in parallel. However it was never meant to scale horizontally automatically. The issue lies in the fact that these simcores cannot join at different times without direct user intervention. While there may be a way to modify OneSAF to do this, we instead decided to run the experiment in each of the various configurations separately to simulate the auto-scaling feature horizontally.

Experiment Results

Utilizing the Compute Optimized Instance, we decided to ramp up the number of entities to a little over 10,000 entities. The tests were run just as before over a period of five minutes, stopping the exercise at four minutes and allowing it to run an additional minute for data collection.

One of the metrics that the RDPM tool gives is the Simulation Queue Size. This translates to simulation elements that get queued up to be computed by the simcores. For an exercise of 10,000 entities, at the start of the exercise, the simulation queue reaches Approximately 160,000 elements (much greater than 10,000). This discrepancy is simply due to the fact that each entity requires and causes several events to occur, such as radio transmission signals, detonation, firing, behaviors, etc.

Below we can see the different results for one, two, and four cores respectively. As you might expect, the queue is roughly halved each time.

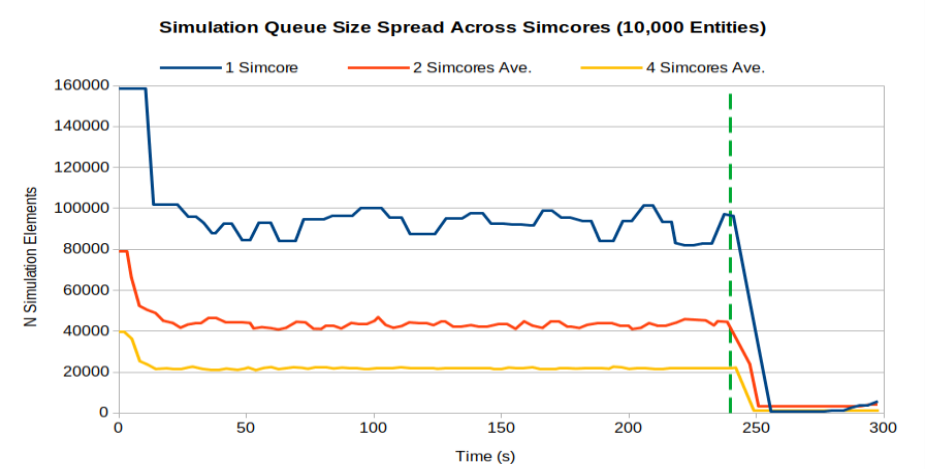


Figure 6. Simulation queue size using different number of simcores

In the cases where the simcores were more than one, the average the queue size was taken. Once again, at the four-minute marker, the exercise is ended and the queue is drained for the remainder of the run. This example shows how OneSAF can logically distribute its simulation load across multiple computers (or, in this case containers running on Compute Optimized Instances). However, logical elements need to run on physical hardware, so the memory and average load should hopefully spread as well. Below, are the System Load Average and Heap Memory Usage spread across one, two, and four simcores, respectively.

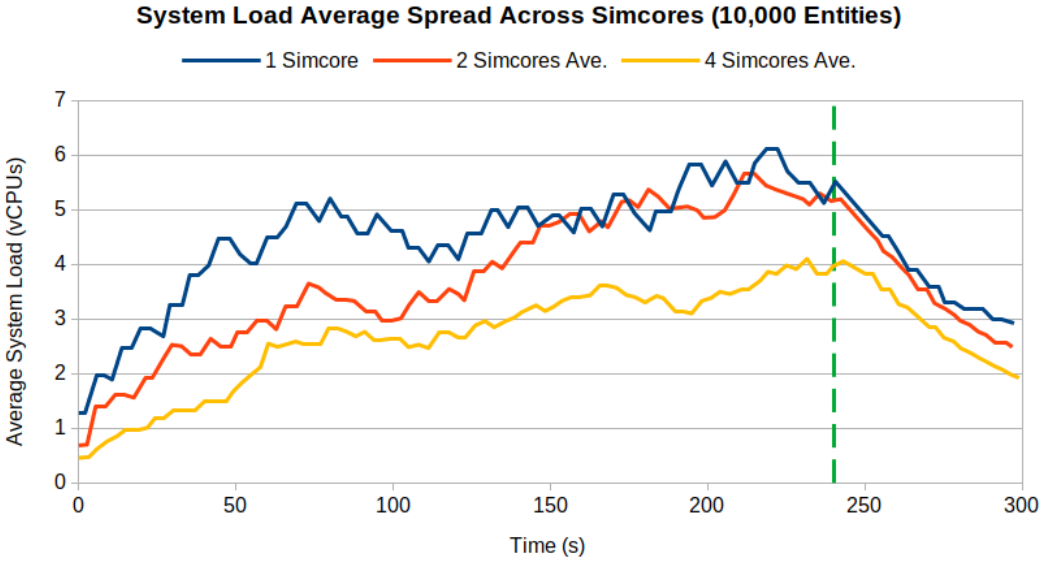


Figure 7. System Load Average with different configurations

Interestingly, the load distribution does indeed spread, but not nearly as cleanly as with the Simulation Queue Size. In fact, around the 150-250s interval the comparison of the one simcore to the two simcores averaged are very close. It's not until we reach four simcores that the load begins to drop (lower being better). This means that horizontal scaling works, but it does not work linearly. The heap memory, on the other hand, does appear to be a bit more pronounced in terms of spreading the load.

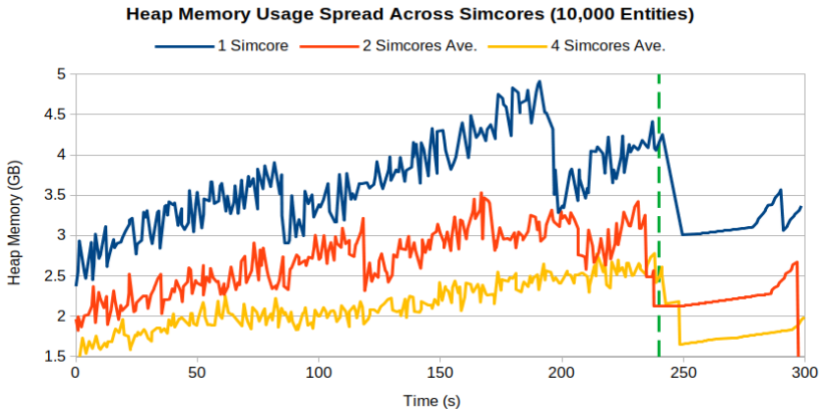
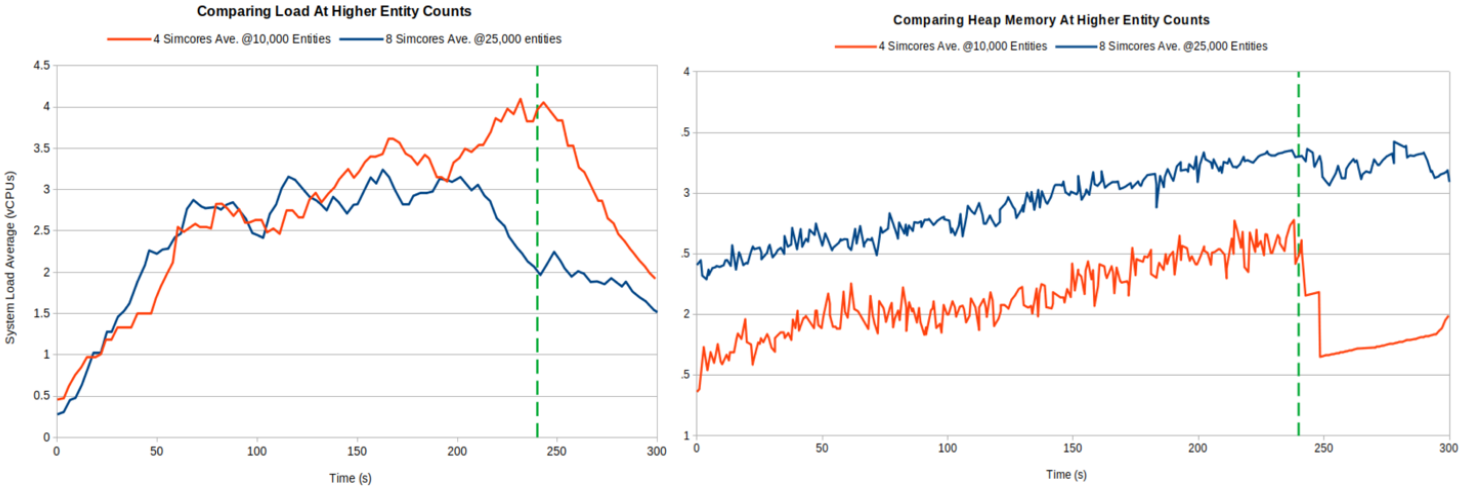


Figure 8. Heap Memory Usage Average with different configurations

As one would expect, the heap memory increases as the exercise progresses and drops once the exercise ends. In either case, the load was indeed spread across the two and four simcore configurations; with four seemingly being the best.

What Happens When Adding More Simcores?

This is the comparison between 8 Simcores @25,000 and 4 Simcores @10,000.



The results indicate that for the system load average, even @25,000 entities, so long as you have 8 simcores, the average load is still less than 4 Simcores @10,000 (i.e. it spreads the load quite nicely).

On the other hand, heap memory is still better in the case of 4 Simcores @10,000, indicating that even though we added more simcores, more memory is used due to using the extra 15,000 entities. However, in both cases, the volatility of the heap wasn't as bad as in the case for the 1 simcore @10,000, and 2 simcores @10,000. This possibly means that more simcores may lead to less unnecessary garbage collection.

Lessons Learned on Cloud-based Scalability Enhancements

What makes the cloud approaches attractive is not the almost unlimited amount of CPU/GPU power that can be brought to solve a problem; it is the underlying architecture that allows the desired performance of applications and CPUs to be monitored while scaling up or down as needed. Built into the cloud configuration files, runtime APIs have the ability to take advantage of auto triggers to expand or contract the simulation environment across virtual computers. Once a customer is ready to deploy their simulation system, the cloud APIs can automatically set up the exercise environment, connects low-latency compute instances into a networked cluster, and distributes the simulation across instances. Based on simulation-based training needs, the built-in cloud APIs replicate and synchronize the data across the instances to create a single, unified simulation. Therefore multiple concurrent users can interact and manipulate the simulation in real time. This shifts the balance of the accuracy of the simulation with the capacity of their hardware to the automated functions of the cloud.

By using the same philosophy, if the cloud-based API is utilized, various micro simulation-related services can make up a next-generation constructive environment to spread across the training exercise environment (cloud and on-premise) to support large-scale simulation and training exercises.

Future Work/Experiments

The current explorations demonstrate some of the scalability in moving a simulation environment such as OneSAF into the cloud for scalability without changing the source code of the environment or the simulation. Immediate scalability can be achieved via inter-process communication that interferes with the linear scaling of entities across distributed simulations. Future explorations will examine if a cloud-based simulation environment can be broken down into more logical micro-simulation environments with reduced overhead in inter-process communication.

The purpose of this research was to explore the environment without major changes to the simulation environment as well as the deployed simulation systems. The next phase of research and development will be to utilize a microsimulation that can be better utilized to examine the limitations of cloud-based simulation environment. We have identified two main approaches:

- Engage more cloud resource management to optimize exercise performance based on simulation exercise requirements, such as the number of entities, and the complexities of training scenarios.
- Explore the possibility of using OneSAF, and/or any large legacy constructive simulations as a service. This requires analysis of OneSAF baseline code to determine how to compose the containers into micro-sims for specific training requirements.
- Enhance auto scaling with specific exercise requirements with an on-demand CPU/memory or additional spawning. The auto scaling can take place without affecting the simulation environment through the use of horizontal and vertical cloud scaling techniques. We will test the limitations with different approaches using DIS/HLA OneSAF-compliant simulations. Based on similar gaming approaches, this should be done in the millions of entities.

We want to conduct a more thorough exploration of the limits of horizontal scaling with current and legacy simulations. With more instances/simcores, we want to be able to answer questions such as, what is the largest scenario/exercise that could be run using OneSAF in the cloud? What bottlenecks and other inefficiencies could exist and be solved to improve performance and also better inform the design of future cloud-enabled simulation platforms? This will involve adjusting the baseline.

SUMMARY

Despite the course architecture of OneSAF being designed long before the adaption of cloud technologies, it has shown that it can take advantage of cloud scalability through containerizing and distributing certain modules. For this experiment, it was the simcore portions. In the short term, this can reduce the need for additional purchased hardware for simulation spikes. Since the parallelization of the simcores resulted in a less than linear growth of capabilities, the simulation communication overhead is already causing large impacts on scalability, even at the 10,000-entity range. It is this area of the simulation architecture that will be explored next. The goal is to increase the simulation count without sacrificing entity complexity which is normally done in war-gaming environments to break through to larger counts (Wardaszko, 2018).

Going one step further, we would also like to explore the implications of applying such scaling capabilities to hybrid LVC exercise environments where a variety of training and simulation systems joining large-scale joint-level distributed exercises.

We envision the future enterprise-level LVC training exercises consisting of training systems as services, platforms as services, local labs, systems joining over WAN, VPN, as well as different types of clouds, etc. Constructive simulation systems, as a core component of such hybrid exercises, will need to step up on scalability capabilities.

REFERENCES

Bearss, E. & Zinser, R., (2021). Control and Management of Clustered Simulation Systems. *Proceedings SIW 2021*.

Clive, P., (2018). Advanced Framework for Simulation, Integration and Modeling (AFSIM), *Int'l Conf. Scientific Computing*

Sanders, C. & Davis, G., (2022). Army Simulations Just Don't Cut the Mustard. Retrieved from https://asc.army.mil/armyalt/Summer2022/html/htmlArticles/index.html?origin=reader&page=30&article=/28_article.html

Wardaszko, M., (2018). Complexity in Simulation Gaming. *Developments in Business Simulation and Experiential Learning, Volume 45*