SANDIA REPORT

SAND2015-0236
Unlimited Release
Printed January 2015

Toward Interactive Scenario Analysis and Exploration

A Study on Simulation Technology Optimization and Scalability with Big-Data Analysis and their Applications

Thomas Russell Gayle, Kenneth Summers, Blayde Jungels, Fred J. Oppel III

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Sandia National Laboratories

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone:	(865) 576-8401
Facsimile:	(865) 576-5728
E-Mail:	reports@adonis.osti.gov
Online ordering:	http://www.osti.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone:	(800) 553-6847
Facsimile:	(703) 605-6900
E-Mail:	orders@ntis.fedworld.gov
Online ordering:	http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online

SAND2015-0236
Unlimited Release
Printed January 2015

Toward Interactive Scenario Analysis and Exploration

A Study on Simulation Technology Optimization and Scalability with Big-Data Analysis and their Applications

Thomas Russell Gayle, Kenneth Summers, Blayde Jungels, Fred J. Oppel III
Department 6134: Interactive Systems, Simulations, and Analysis
Sandia National Laboratories (SNL)
P.O. Box 5800
Albuqeruque, New Mexico 87185-MS1004
{rgayle,klsumme,jmjunge,fjoppel}@sandia.gov

Abstract

As Modeling and Simulation (M&S) tools have matured, their applicability and importance have increased across many national security challenges. In particular, they provide a way to test how something may behave without the need to do real world testing. However, current and future changes across several factors including capabilities, policy, and funding are driving a need for rapid response or evaluation in ways that many M&S tools cannot address. Issues around large data, computational requirements, delivery mechanisms, and analyst involvement already exist and pose significant challenges. Furthermore, rising expectations, rising input complexity, and increasing depth of analysis will only increase the difficulty of these challenges.

In this study we examine whether innovations in M&S software coupled with advances in "cloud" computing and "big-data" methodologies can overcome many of these challenges. In particular, we propose a simple, horizontally-scalable distributed computing envirnoment that could provide the foundation (i.e. "cloud") for next-generation M&S-based applications based on the notion of "parallel multi-simulation". In our context, the goal of parallel multi-simulation is to consider as many simultaneous paths of execution as possible. Therefore, with sufficient resources, the complexity is dominated by the cost of single scenario runs as opposed to the number of runs required. We show the feasibility of this architecture through a stable prototype implementation coupled with the Umbra Simulation Framework [6]. Finally, we highlight the utility through multiple novel analysis tools and by showing the performance improvement compared to existing tools.

Acknowledgment

A special thanks to the many additional individuals, collaborators, and developers who helped and advised me at various stages in this work, particularly Jon Bradley, Sophia Corwell, David Danzilio, Eric Gottlieb, Richard Grochowski, Brian Hart, Brian Jeantette, Charles Little, Gloria Padilla-Gallegos, Taylor Owen, Michael Skroch, Brandon Silver, Dan Small, Raymond Trechtor, Vince Urias, Jon Whetzel, Gregg Whitford and Brian Wright and the many folks in the Sandia LDRD office.

This work was funded under LDRD Project Number 167008 and Title ”Toward Interactive Scenario Analysis and Exploration: A Study on Simulation Technology Optimization and Scalability with Big-Data Analysis and their Applications.”

Contents

Figures

Tables

Definitions

Entity Characters and objects that represent elements placed on a terrain. This includes humans, vehicles, aircraft, boats, both direct and indirect fire weapon systems, sensors, lights, cameras, and more.

Exercise A maneuver or simulated operation (e.g., force-on-force, tabletop, software simulation) involving planning, preparation, and execution that is carried out for the purpose of training and evaluation. An exercise may consist of one or more scenarios.

Scenario A hypothetical event (e.g. an emergency, attack, or theft) used to evaluate the effectiveness of a protection or response system against a defined action or threat.

Scenario Run A specific path through or occurence of a scenario, consisting of a unique sequence of events and entity decisions and actions. Each run of a scenario can have different outcomes from other runs.

Introduction

As Modeling and Simulation (M&S) tools have matured, their applicability and importance have increased across many national security challenges. From their role in "serious gaming" for training and response to a variety of detailed analyses for asset protection or neutralization studies, M&S tools show up in numerous applications areas. Most importantly, they provide a way to test how something may behave without the need to do real world testing. However, current and future changes across several factors including changes in capabilities, policy, and funding are driving a need for rapid response or evaluation in ways that many M&S tools cannot address. Issues around large data, computational requirements, delivery mechanisms, and analyst involvement already exist and pose significant challenges. Furthermore, rising expectations, rising input complexity, and increasing depth of analysis will only increase the difficulty of these challenges.

New M&S tools and solutions are needed to overcome these challenges. Since the M&S spectrum is incredibly broad and diverse, it's unlikely that a single tool will be able to coalesce the breadth of options and models available. However, specialized platforms may be able to help alleviate the issues for wide classes or M&S problem areas.

As a result, this study focuses on the space of tools that simulate seconds-to- hours of time in full 3D worlds where human factors and interactions play an important role (i.e. gaming or game-like environments). These factors can include sensing, perception, behaviors, actions, communications, and more. Example applications include emergency response and trainig, force-on-force simulations, and a variety of parameter studies on entities in a 3D dynamical world.

Many key analyses and results from these solutions often require tens to thousands of independent "scenario runs" in order to derive meaningful results. For example, many simple statistics, such as the binary win/loss ratio, require close to one thousand runs in the worst case to provide results with a high confidence value. Since modern M&S solutions generally require at least minutes to tens-of-minutes to complete a single run for a realistic scenario, complete analyses can take hours or days. Since even small changes may require a new set of runs, this compute requirement makes it impractical to experiment or rapidly iterate with the scenario itself. We believe that this lack of interactivitness fundamentally constrains the developers, analysts, and decision makers. Furthermore, it adds restrictions on the types of solutions that are even feasible. Thus, this study aims to change that notion by examining ways to minimize the time constraint on analysis solutions. Furthermore, we explore what types of tools become more feasible as a result.

The rest of this section introduces the observations and overarching background ideas associated with this work. We first discuss the role of interactivity and ways in which it currently exists in related M&S solutions. Then, we introduce cloud computing as a means to provide the computational resources and interfaces necessary to reach beyond existing solutions.

Interactivity In Scenario Exploration

Within the realm of 3D M&S tools, interativity can take on different meanings. So, this section provides a brief discussion on interactivity as it applies this study.

"Interactive" as used in the the report's title refers to the readiness or responsiveness of the application to the user. It can be quantitatively measured by time, as we do in this work. On the other hand, "interactivity" can be considered as the means of communication or the dialog between an user and a computer. We make the distinction since both concepts are important and relevant to this work.

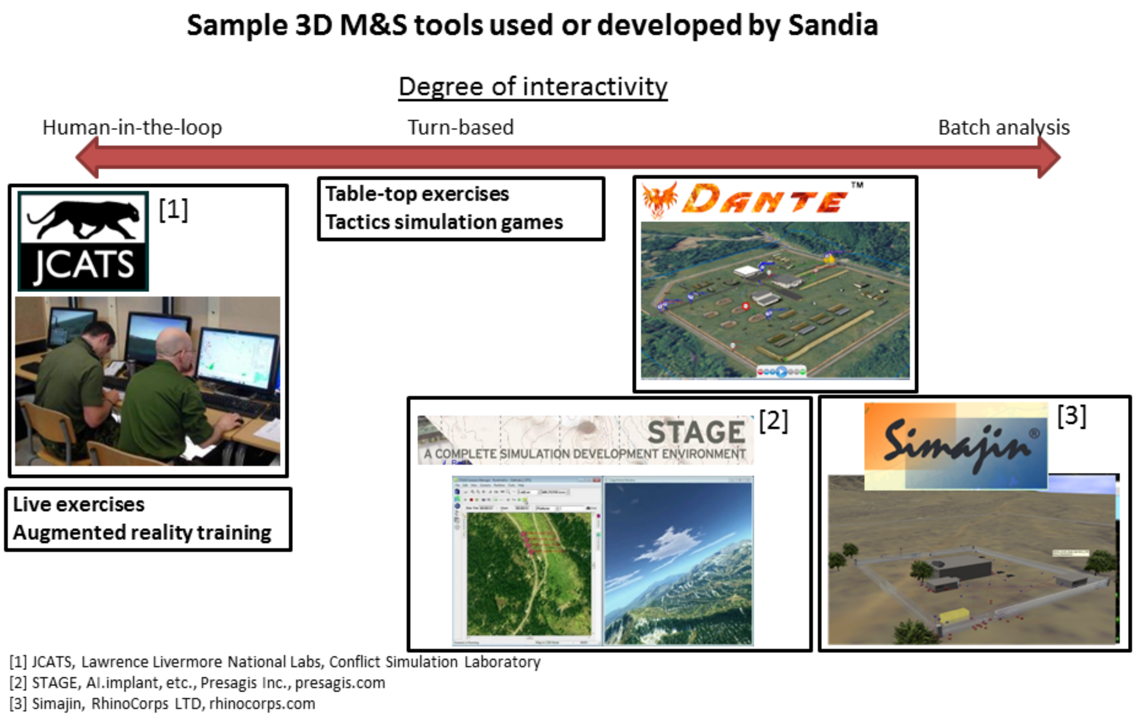

Figure 1. The Interactivity Spectrum with Example Applications.

With this in mind, the key result of this study is to improve the interactiveness of 3D M&S tools with regard to the analysis or exploration of a target scenario or exercise. As a sub-goal, we seek to enable novel modes of interactivity and analyst workflows while simultaneously enhancing existing solutions. In a little more depth, we want to consider solutions that overcome standard challenges across the spectrum of interactivity. These are described below, but are also summarized in Table 1.

Since interactivity comes in many forms, we like to consider interativity of these tools as a spectrum, with complete human control (e.g., human-in-the-loop) at one end and complete automation (e.g., batch-analysis) computer control the other. Figure 1 shows several example tools and applications along the spectrum.

Table 1. Tradeoffs by degree of interactivity.

	Human Controlled	Mixed (e.g. turn-based)	Computer controlled
Interactiveness	Quick initial results, individual runs are "real-time"	Relatively quick, pause between "turns"	Individual runs generally quick, analysis may require require 1000s of runs
Cost (man-power)	Expensive, one human for a few entities	Moderate, operators control many entities	Minimal, an operator moderates while computer makes decisions
Behaviors	Directly accounts for human behaviors (via operators)	Can be considered while run is in progress	Behaviors and coordination must be modeled

In particular, completely human controlled M&S solutions have the advantage that individual scenario runs complete in "real-time" and the accuracy of the simulation is usually very high since human players make all the decision (and, mistakes). However, it would be extremely challenging for a single human to control more than a couple of entities at a time. Therefore, performing scenario runs in this paradigm can be quite expensive especially if many entities are needed. It is impractical to do analyses that require more than even tens of runs for statistically significant results.

On the other hand, completely computer controlled solutions have the advantage that few, if any, individuals are needed to run a scenario. In general, only the analyst and an operator may be needed. However, complete analyses can take hours to days to complete and all human behavior (including errors and inefficiencies) must be modeled and implemented within the simulation. As a result, it is more difficult to model wide variation in scenario runs.

Alternatively, hybrid solutions such as "turn-based" event models try to balance both human-in-the-loop and computer controlled systems. The simulation frequently is paused to allow the operator (or, a group of operators) time to decide how to proceed. While individual runs tend to be the slowest (due to the pausing), it requires much less man-power than human-controlled solutions and allows the operators some degree of control over behaviors and coordination (more than that of computer controlled). However, like in the human-controlled situation, it is likely impractical to do this for analyses that require 100s of runs for results.

Our investigation into interactive scenario analysis and exploration is motivated greatly by Table 1. We believe that there is little or no reason why solutions need to adhere to these constraints, and that being able to "mix" or "combine" strategies will lead to novel solutions and significantly enhance existing ones. For instance, consider a turn-based solution that gives the operator or analyst a batch analysis of their current strategy (in a short amount of time). In another example, it would be much easier for an analyst explore tradeoffs in building a scenario if they were able to get results more quickly (i.e. turn the computer controlled into a human-in-the-loop tool). This would allow them to explore the scenario "space" in a way that was previously infeasible or required more subject matter expert (SME) support.

Cloud computing

A requirement for our goal of interactivness is sufficient computational resources to solve the problem in a shorter timeframe. This problem has certainly be studied in many forms, and an extremely extensive variety of solutions are viable.

Standard high-performance computing (HPC) solutions tend to be poor fits for this type of 3D M&S application due largely to the event and action-driven architecture typical of these solutions. In many cases, it would be challenging to divide the work among many processing cores in an even manner. Distributed computing tends to be a better fit, as individual nodes in a cluster can handle one or more simulation runs. In other words, distributed solutions and analyses scale better horizontally as opposed to vertically. However, the cost to operate and maintain a cluster can be higher than desired for many objectives and often requires additional assistance.

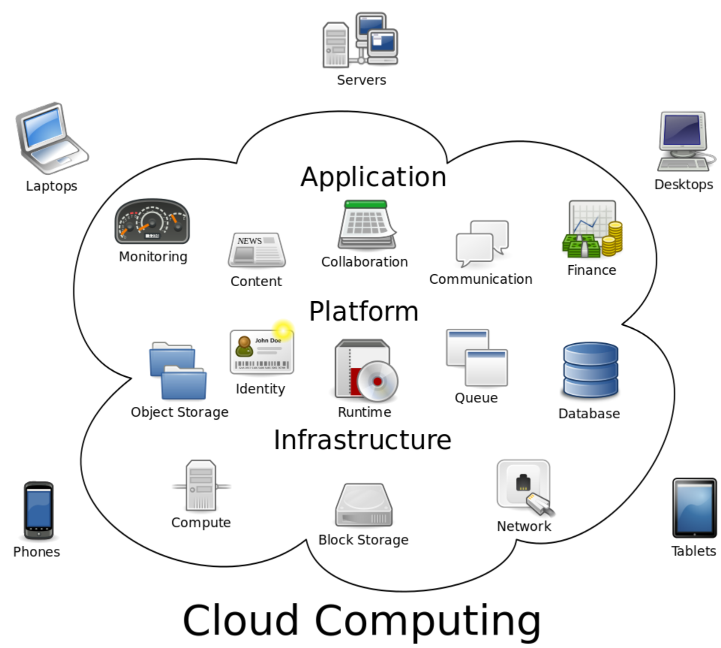

Figure 2. Cloud computing diagram [16]

Instead, we look toward rapidly maturing "cloud computing" technologies as a means to obtain the necessary computational resources with reduced maintenance and operational overhead. For a little more depth, NIST defines cloud computing as ”a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.”[8]. In a sense, cloud computing really is an evolution of distributed computing where much of the configuration and deployment options are integrated directly within the cloud “operating system”. So, within our context, we want to use cloud technologies to easily and rapidly supply the compute necessary, without the user having to worry about any of these details. Figure 2 shows a metaphor to what cloud computing looks like to the user [16]. It is a set of resources and applications that are available but mostly hidden from the user.

Our goal is use cloud features and APIs to setup a flexible parallel mutli-simulation environment for M&S applications. This environment should be flexible and be able to adjust to the needs of many users (within the constraints of the cloud it lives in). Also, since the cloud moves the bulk of the computation away from users, and added advantage is that the interface can be run on lighter and less power machines. This enables additional delivery options such as mobile devices and access from anywhere with a network route to the resources.

Our study

More specifically, in this study we examine whether innovations in M&S software coupled with advances in “cloud” computing methodologies can overcome many of these challenges. Our primary goal is to fundamentally change the architecture of related M&S solutions such that complex analyses can be completed in seconds or minutes, rather than in hours or days.

We propose a simple, horizontally-scalable distributed computing envirnoment that could provide the foundation (i.e. “cloud”) for next-generation M&S-based applications based on the notion of “parallel multi-simulation”. In our context, parallel multi-simulation simply refers to the ability to run many different paths of execution simultaneously. This maps well to our problem domain, since each scenario run tends to be indpendent from previous ones. This can include run situations where you want to allow randomness to adjust the decisions along a scenario run or if you want to vary a parameter for a study over the course of many runs.

With sufficient resources, the complexity is dominated by the cost of single executions of the desired scenario. In this study, we show the feasibility of this architecture through a stable prototype implementation coupled with the Umbra Simulation Framework [6]. Finally, we highlight the utility through multiple novel analysis tools and by showing the performance improvement compared to existing tools.

The remainder of this report is organized as follows. Details of our generic cloud framework are detailed in the section on Parallel Multi-simulation, while the Rumba section details how we connect Umbra to our cloud. The Results section provides information on our benchmark scenarios, performance results and related simulation optimizations, and additional discussion on some novel workflows and applications. The Conclusion discusses future avenues of investigation, summarizes and concludes the report.

Parallel Multi-Simulation in the Cloud

As discussed in previous sections, our goal is to build a parallel multi- simulation environment capable of handling a variety of M&S tasks. To reiterate, we define parallel multi-simulation as a way to run or follow many independent paths through a scenario simultaneously. The main goal for our target environment is implement parallel multi-simulation to provide the compute capacity necessary to run a batch analyses, parameter studies, and related operations at more interactive rates. As such, secondary goals are that it should be reliable, easy to use, easy to reconfigure, and easy to add resources. In general, the user should not need to know anything about how this environment is built in order to use it.

Development of this environment followed several steps, including selection of a cloud environment, designing the software stack used to manage and process work requests, and implementing the stack itself. This section focuses on the design and final implementation of the environment. Interested readers can refer to the the project wiki for more details about setting up a development cloud environment (and caveats along the way), setup scripts for the compute nodes, and general information about (open source) cloud providers.

Software stack design

Designing the enviornment included two primary considerations; selection of the cloud "operating system" and deciding on a work management system. Again, we focus here on the implemented design. Previous designs and may also be found on the project wiki.

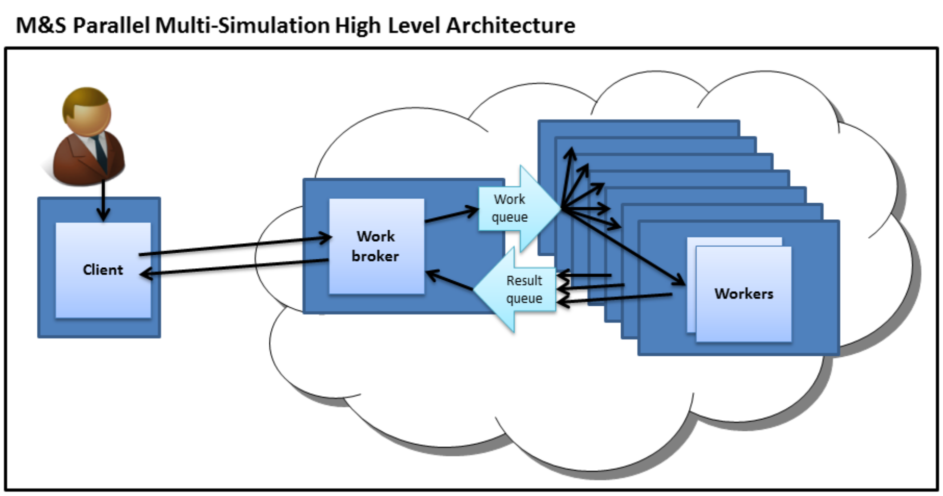

Figure 3. M&S Parallel Multi-Simulation Environment: High-Level Architecture

Our design includes the following features:

- **Work and result queues**: These are software queues that persist between runs and are used to organize the work needs to be done and the results that are ready to be returned to the user.
- **Producer-consumer work distribution model**: As there are numerous ways in which the work and results queues could be used, we have chosen to implemented a producer-consumer model on top of them to manage the work. In this model, producers tasks are responsible for adding work while consumers are responsible for looking for work available and then completing it.
- **Distributed scheduler and work broker**: The work broker is a process that handles user and client requests and publishes work to the work queues. It is also responsible for returning results to the client. In the context of our environment, it also servers as the work scheduler to determine the order in which work should be done.
- **Worker instances**: These are the instances that do the actual work. In our system, the workers are consumers of the work queue and producers for the result queues.

Figure 3 shows an example of what this design looks like. In this figure, compute instances are shown as blocks, work queues are shown as large arrows, and the smaller arrows show the flow of work. Briefly, after a client requests work, a work broker produces work for the work queue. Workers subscribe to and monitor the work queue for work to be consumed and completed. If the worker finds work it knows how to handle, it accepts the work, performs any necessary computation, and adds the results onto a results queue. As results are placed on the queue, the broker then passes the results back to the requesting user.

Since the primary requirement of parallel multi-simulation is that multiple independent paths of execution can be run in parallel, we must ensure that our design allows for this. In our design, each worker is independent. Thus, it is sufficient to ensure that each worker given a separate path of execution for a given scenario and analysis. This is largely the motivation for the work broker. It must be implemented to understand the type of work being requested (e.g. a randomized trial vs. a parameter study) and generate work accordingly for the workers. This also helps separate the logic of the work broker from other parts of the system and simplifies the overall design and implementation. With this, these components then satisfy our requirements for parallel multi-simulation.

There are a number of advantages and disadvantages to this approach. First, the design itself can add latency to the system while also removing compute resources from the workers. This adds an extra "hop" between the work to be done and the workers. Fortunately, the resource requirements for the broker are fairly minimal and the cost of the hop is very small with respect to the total compute time. In fact, in practice we have been able to run compute jobs on the work broker node with little or no observable impact in the job completion time. Second, the publisher/subscriber model also adds a small amount of latency. A work broker could simply have workers execute the work directly, instead of pushing to a queue. On the other hand, the producer-consumer model has one large benefit in that the work broker no longer needs to know about what workers are available to do work. In fact, new workers could be added or removed seamlessly without any impact on the flow of work. This benefit far outweights the overhead and cost of managing the queues. Finally, one design consideration was to keep this environment generic. While our intended use is for M&S solutions, it could be readily applied to any task where indepdent compute tasks are required.

Cloud implementation

The success of our design is highly dependent on the implementation. As such, we decided to build our entire platform based on open-source technologies that are popular, have been proven reliable, and horizontally scalable. Thus, our implmentation should be easier to maintain and update while also providing efficient scalability and flexibility.

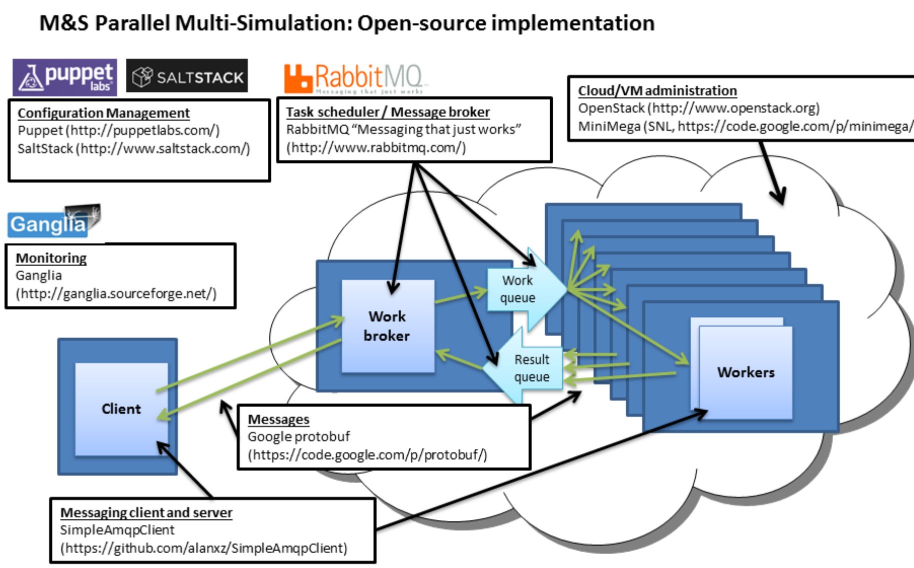

Figure 4. M&S P arallel Multi-Simulation Environment: Implementation

Figure 4 shows a detailed breakdown of what software packages are used for each component in the high level design. While other solutions are certainly possible, this combination yields an effecient and reliable solution. Many components include built-in administration and monitoring, which is invaluable in understanding what is going on in the environment.

Our implementation builds upon the following:

- **"Cloud" platform**: The "operating system" for the cloud. This defines the operations that are possible and handles allocation of compute instances as well as basic configuration of them. Our prototype system runs on an OpenStack[10] cloud provided by Center 5500 at Sandia National Laboratories. We have also experimented with Sandia developed MegaMini[9], though it was less feature-rich for configuration of our network. OpenStack provides user and resource management from a convenient web frontend as well as a programmatic interface for maintaining your resource pool.

- **Configuration management**: Configuration management enables automated control and configuration of our compute resources. For our use, its primary role was in configuring the work broker, its queues, and installing and configuring M&S tools on the workers. We have solutions using standard bash scripts, SaltStack[14], and puppet[12]. Each has pros and cons, mostly in how the configuration is delivered and the effort involved developing and updating the set assets each type of node requires.

- **Work broker and task scheduler**: One of the core pieces of infrastructure, RabbitMQ[11] provides both the messaging interface and queues. RabbitMQ runs as a server that manages a set of queues and accepts connections to those queues from clients. Its key features are simplicity (both in use and in setup), persistence, and horizontal scalability.

- **Monitoring**: Ganglia[3] is our current monitoring solution. It is a lightweight and configurable system for reporting the load on network, disk, and computing resources. Its primary value is in understanding the current compute load (to determine if we need to add more resources to a job) and in diagnosing problems with existing compute nodes.

- **Worker and client template libraries**: SimpleAMQPClient[1] provides a C++ “interface” to the cloud for both client and workers. Clients implement a SimpleAMQP process that can initate (or, request) work from the cloud. Workers implement a SimpleAMQP interface that allows it to monitor for work. By using this, it keeps the API (Application Program Interface) between C++ components simple and uniform. Similar packages are available for a wide variety of programming and scripting languages.

- **Message format**: Google protocol buffers (protobuf)[4] define the data format that actually gets placed on the queues and that gets sent between the user and the cloud. It is widely accepted due to being a lightweight binary representation that can natively allow C++, Java, and Python applications to communicate with each other. Many 3rd party packages exist to expand the range of supported languages.

- **Data transfer**: We currently use the rsync utility to transfer large data files and scenario files between the user and the cloud.

The aforementioned software packages and libraries provide all the components necessary to run and monitor the parallel multi-simulation environment. Each package was chosen either due to its relative simplicity and scalability or for its efficiency or utility.

In practice, this combination of tools has proven to be quite reliable and effective. It has served a wide variety of applications with little oversight. Further discussion the performance, reliability, and flexibilty of this solution in our results and applications section.

Rumba: Cloud-based Umbra with optimizations

This study includes several driving applications, to highlight the usefulness of the approach and to provide several benchmark applications. In particular, we target the Umbra Simulation Toolkit[6] largely due to its range of applications and accessibility. This section briefly describes Umbra, components in its Dante Application Suite, and how the parallel mutli-simulation environment fits within its workflow.

Umbra Simulation Toolkit

Briefly, the Umbra Simulation Toolkit is a suite of libraries to assist users and developers in creating 3D simulations and applications. Its libraries are also available to support existing modeling and 3D applications as needed.

Umbra has been shown to be useful in creating models of complex systems with mixed fidelity. Most importantly, it can model features of worlds at various degrees of fidelity so that users do not need to include a more accurate (and expensive) model than necessary. Extensive work has gone into building packages for Umbra, including models for sensing, communication, behaviors, actions, path planning, direct fire weapons, land, air, and sea vehicles, a wide variety of sensors, analytical abilities and much more.

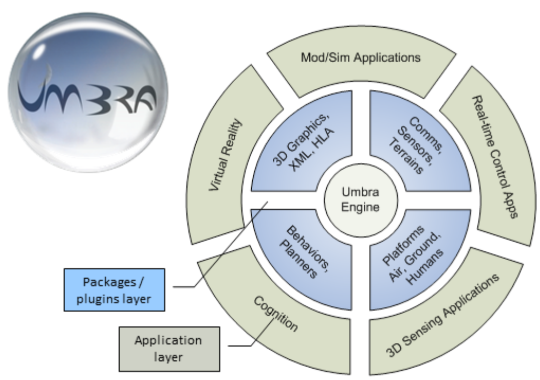

Figure 5. Umbra Simulation Toolkit

Figure 5 is a diagram that conceptually shows how Umbra applications are built. At the core of an Umbra application is the Umbra enginge. This engine sychronizes and manages each of the modules required of the application as well as the state of the simulation. One key advantage of this design is that results are guaranteed to be repeatable given the same inputs. Additional libraries or packages can then be added to the engine which are combined to build applications.

For more information about Umbra, it's packages, and applications written using Umbra, we refer interested readers to [6].

Dante Application Suite

While Umbra is our driving platform, Dante is our driving Umbra M&S application. Dante itself is used to create, simulate, and analyze scenarios. Its suite includes a set of applications for the creating terrains, testing and analyzing sensing on terrains, evaluating coverage and route planning through terrains, and most importantly for evaluating tactics and procedures among physical objects. Among other things, it excels in evaluating force-on-force and red-teaming scenarios. For more information, we refer interested readers to the Umbra Products website [5].

Figure 6. Exmaples from the Dante application suite. The left image is a snapshot from a notional force-on-force simulation using Dante. The right is snapshot from a parameter study using OpShed.

Figure 6 shows example images from both Dante and OpShed (Operations ViewShed). The left is a snapshot from a notional force-on-force exercise on a notional terrain, while the right is a snapshot during a sensativity study between a sensor and vehicle. These represent just a small range of the capabilities of Dante.

To perform analyses using Dante, users run batch analysis using randomized methods or by varying parameters of scenario. Dante includes a “Batch Manager” process to aid in the scheduling, organization, and processing results of the output. A standard “batch” runs on the local machine by default, meaning that even moderate sized scenarios may require a fairly powerful machine to process a large study in a timely manner. As previously mentioned, a goal of our multi-simulation environment is to provide a convenient way to offload the compute. Thus, Dante would be able to run on smaller and less powerful machines. Such a machine would only need to represent the scenario and process the results, making it more practical for in-the-field use or in other arrangements where access to more powerful single machines is limited.

Cloud integration

As our goal for the study is to create a parallel multi-simulation environment for modeling and simulation, the final detail is how a traditional M&S tool can be updated to use the previously described parallel multi-simulation environment.

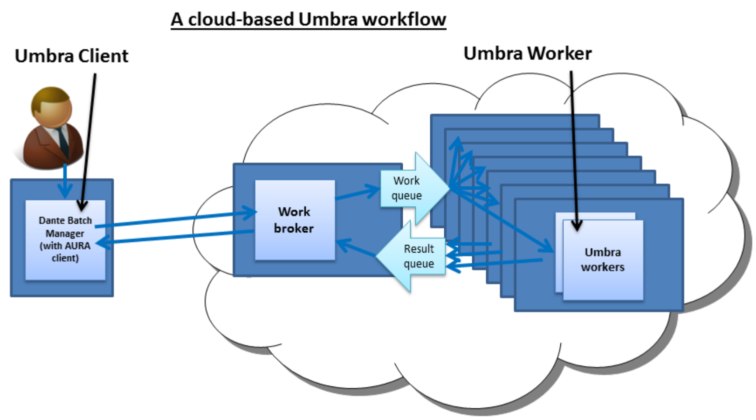

Figure 7. Rumba: Umbra's parallel multi-simuluation interface

In the end, we settled on two different solutions both of which have been integrated as shown in Figure 7. This includes:

- **Command-line interface and subprocess module**: The command-line interface wraps the SimpleAMQPClient (discussed in the Cloud Implmentation subsection) into a small application. This application mimics the batch manager's operations, although scheduling is offloaded onto the work broker. The subprocess module is a simple wrapper which allows C++ and C# libraries to call the command-line interface directly. The primary advantage of this approach the parameters passed to the module are the same as those on the command-line. Thus, debugging and testing is simplified. On the other hand, the application has to understand where the command-line utility places results for post-processing.
- **Directly integrated library**: A Rumba module was implemented that inherits from SimpleAMQPClient. Like in the command-line example, this module accepts all the parameters necessary to set up the run and then communicates with the broker. The main advantage of this approach is more control over communication with the work broker and on how results are returned and where they are stored.

In the end, both solutions worked as desired. They provided a simple way to access the multi-simulation environment. Furthermore, since it hides all management of the compute resources (which aligns with our "cloud" goals), it allows the developer to focus solely on the applications and not have to worry about managing, maintaining, and running jobs on in the distributed environment.

This allowed us to quickly and easily optimized existing applications as well as prototype several novel applications to aid in the exploration and analysis of scenarios. Details on these applications is provided in the Results section.

Results

In this section, we highlight the various outputs of our study on improving interactiveness for scenario exploration and analysis. Most notably, this includes the creation of a compelling benchmark scenario,

Benchmark Notional Scenario

While several example scenarios are included with Dante, many of them focus on single capabilities and did not adequately stress the compute environment. To overcome this hurdle, we created new notional scenarios that were more representative of what may go into an actual use case.

Figure 8. Notional Benchmark Scenario

Figure 8 shows a snapshot of our benchmark notional sceanrio from high above the terrain. A red team is staging fronts at two different locations while the protective forces (blue team) must protect both sites. The first component of the red team is an air assault team, where a helicopter navigates to a drop-off location, 6 team members fast rope down and attempt to gain entry into a nearby building. The second part of the red forces is a ground team including two vehicles each with 4 attackers and 2 snipers hiding in the nearby hills. The second force must breach barriers and perform a bounding overwatch action as they invade their target building. The blue forces include a protective tower with a single defender inside, a vehicle patrol outside the main facility, a foot patrol within the fences of the main facilities, and defenders within the target buliding. There are a total of 16 members on the red team, and 21 on the blue team.

Performance

With this model, analyzing performance is very straightforward. Let n be the number of independent compute nodes and m be the desired number of runs. Then, the total time is approximately $max(\frac{m}{n},1) \times t + c$ where $max(a,b)$ returns the maximum of a and b, t is the average time per scenario run, and c is the overhead time of copy scenario data to the cloud, passing data through the queues, and returning the results to the client. We note that this assumes that t is relatively constant between scenario runs (or, that the standard deviation on single scenario run time is small). c is very dependent on the complexity of the scenario. To help minimize this cost, the client and cloud only transmit differences if a scenario has already been used once before (by a particular user) and also cache terrains so that they are available to other scenarios using the same terrain. The differences are relatively quick and easy to compute (e.g. via diff).

We distinguish between compute nodes and processors mostly because Umbra (and Dante) are multi-threaded processes which can utilize the full number of CPUs on a single node. Although, in practice, over a large study the number of processors per node made little difference in the overall results (e.g. we saw similar performance when processing 100 runs with 10 dual-CPU nodes as we need with 20 single-CPU nodes).

In practice, our experimental results were very close to the predicted value. However, we did encounter the challenge of acquiring enough to compute nodes (n) to reduce the time to minutes for large studies. On the other hand, for small studies or quick approximations (e.g. where only 30-40 runs are used), we achieved our goal of having meaningful results returned on the order of minutes (or less). This represents a significant saving when compared to the traditional Dante Batch Manager. Furthermore, it freed the user's machine during computation, which allowed users to work on or look at other things simultaneously.

Simulation Technology Optimization

While not discussed above, the other way to reduce the total analysis time is to reduce t, the average scenario run time. We did do a little work in this area, though most of it was beyond the scope of this study. In particular, we chose to optimize one of the most demanding modules in a Dante scenario run: line-of-sight. Using Embree[2], we were able to improve line-of-sight by over 30x (from approximately $300,000$ queries per second to about $11,000,000$ queries per second). This resulted in an obserable improvement in t, but other factors still kept the run-time on the order of minutes.

In the future, we do plan to use our cloud to help further improve t. This is discussed more in the Summary section of this report.

Novel applications

One of the most promising outcomes from this study were novel applications that were enabled by this study. As alluded to in the subsection on interactivity, we aimed to merge or blend the boundaries between different modes of interactivity and also wanted to investigate what was possible with simple and plentiful computational resources. This subsection highlights the enhancements to and new applications created based on the results of this study.

Dante Scenario Editor with Analysis Exploration

Traditionally, the Dante Scenario Editor (DSE) is used to build a scenario and the Dante Batch Manager (DBM) is used to evaluate or run analyses against the scenario. The extra hop is not only time-consuming (since DBM can take a while to complete for even a handful of runs), but it distracts the user from the task at hand. Furthermore, it inherently causes users and analysts to make many changes to a scenario without completely understanding how those changes impact the overall results.

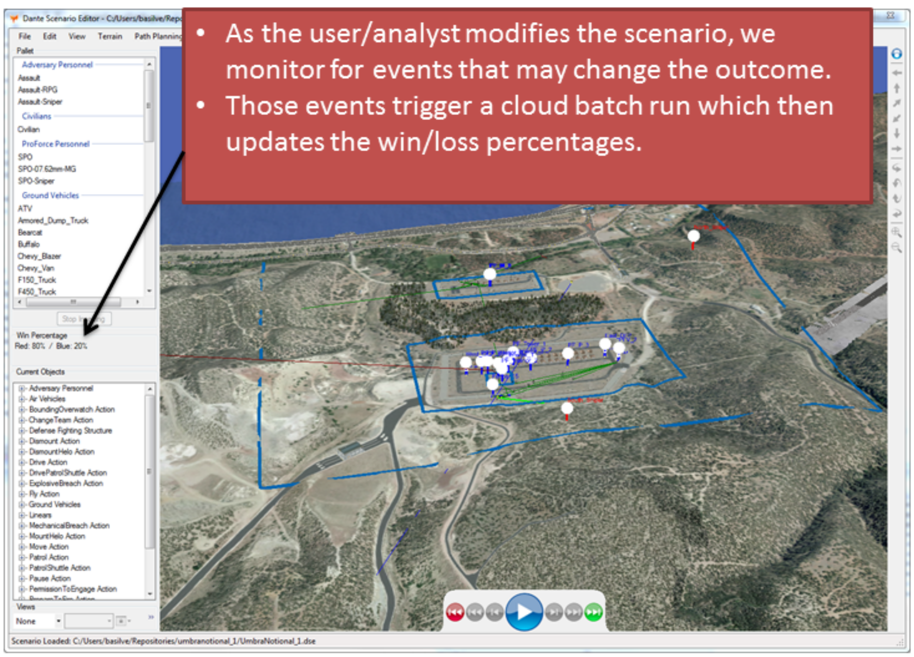

Figure 9. Analysis Exploration in Dante

Thus, our first experiment to blend the boundaries along the 3D M&S interactivity spectrum was to modify DSE to behave more like a turn-based game. The idea is that while the user or analyst is making changes to a scenario, changes that may impact the results will automatically trigger a small batch analysis in our parallel multi-simulation environment. This allows the user to see the results of small changes, and also let's them stay focused on task of building the scenario. Though difficult to see, a sample win/loss ratio example is shown in Figure 9.

With the help of the cloud, it was very quick to build and deploy this application. Preliminary feedback from a few users was extremely positive, with some seeing this as a substantial improvement over their current workflow.

K-best paths pathways analysis

Our next experiment was to imagine an application that would have been less impractical given traditional workflows. We chose to examine a different approach to pathways analysis. Briefly, pathways analysis is a type of analysis done to determine the effectiveness of an entire system over the span of many possible action paths. For example, it is useful in determining routes or regions that have highest success rate for a red team.

Pathways analysis is generally performed by building a graph that represents the ways through each layer of facility and the costs associated with going between nodes in the graph. Different metrics are then used to determine the cost along numerous paths through the graph and then graph algorithms are used to generate the related analyses. While the approach is technically sound, it tends to be difficult to evaluate and include human behaviors and the vast number of ways a human may physically move between layers.

To overcome these limitations, we experimented with doing a pathways analysis by first determining the k most likely paths an entity may take through a scenario and then doing a forward simulation batch analysis on each path. To do this, we make use of the path planning module in Umbra. For each path produced by the planner, we increase the heuristic value (cost) of traveling over areas that have been traversed before. Thus, subsequent routes tend to avoid areas that have already been explored. For each path, we perform an approximate batch analysis to determine the win/loss ratio for each path. Finally, the results are combined and displayed overlaid on the original terrain.

To test this approach, we examined engagement avoidance (see Figure 10) and sensor avoidance (see Figure 11) as example applications.

We color paths based on success, where blue paths mean that the blue team won and red paths mean that the red entity reached his goal. Furthermore, the "saturation" or intensity of the color determines how successful that path was. For example, a bright blue path means that the blue team always won, whereas a less intense blue path means that they usually won, but not every time.

In the engagement avoidance experiment, the goal is for the red entity to reach a yellow vehicle behind two blue assets. From the example, it is clear that certain paths always lead to success and others to failure.

In the sensor avoidance experiment, we also show white paths which are those in which the red entity did not reach the goal within the allotted time. The results show that while there is

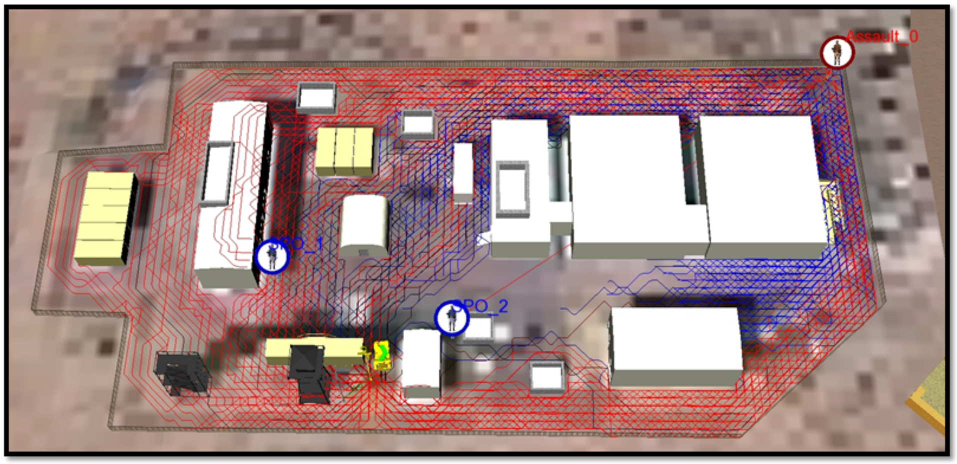

Figure 10. Pathways Analysis for Engagement Avoidance

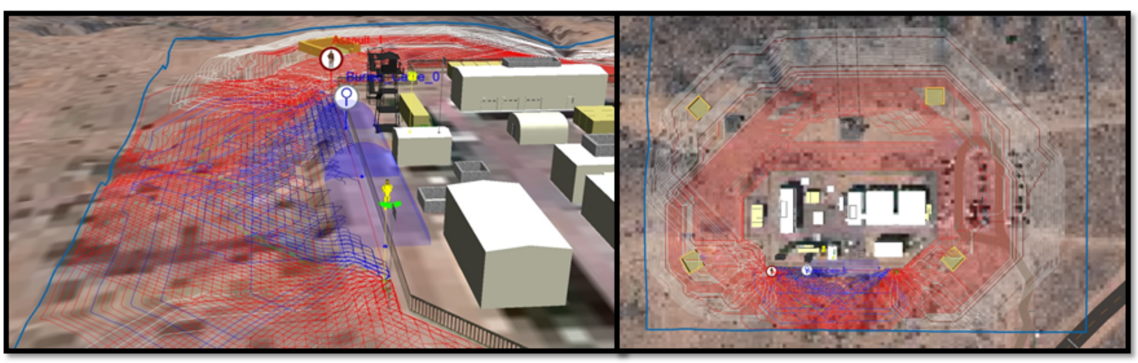

Figure 11. Pathways Analysis for Sensor Avoidance

definitely a region where the red entity would be sensed, some paths further away from the sensor still resulted in him being sensed.

An added benefit of variant of pathways analysis is that you can see a success “frontier,” or regions where the trend changes from success to failure. This could be useful in trying to determine how to adjust or optimize a blue team layout.

Future Work and Summary

Future Work

While our current solution works well in practice, there are several avenues for future work some of which are included here.

In our current implementation, the lower bound for the total time for analysis and exploration is the time required by the slowest scenario run. While we could certainly do more profiling and optimization for our current M&S solution architecture, there are many more ways to distribute portions of the compute to other servers. In particular, some functionality such as path planning is shared and duplicated between each scenario. A more sophisticated architecture might be able to utilize a shared route-finding service that can additionally cahce results that are guaranteed to be identical between runs. That would greatly reduce the cost associated with those elements.

Another area that could be improved is the post-processing of results. Many tools have been developed to allow cloud and related distributed environments to readily handle and process large amounts of data (e.g. BigTable, Hadoop). As the work required for analysis results becomes increasingly complex and the number of runs grows, big-data solutions can help to minimize the data and compute demands on the user's machine.

Summary

In this study, we proposed, implemented, and tested ways in which we can greatly improve the interactivness of scenario analysis and exploration. Our solution used an environment built with "cloud" computing ideals for horizontally scalable parallel multi-simulation.

We showed that this environment is generic and capable of running with compelling and relevant M&S tools and applications. In fact, the generalism of the approach makes it suitable for many other compute-heavy software tools, where independent results can be run in parallel (e.g. as a kernel in an *n*-particle simulator). Furthermore, the environment was built entirely with scalable open-source technologies. This has two distinct advantages: it can readily incorporate more compute resources without having to restart the environment and is maintainable.

We highlighted the impact of this environment by running it against modern and meaning applications and workloads. As expected, we show that it can reduce the total time for analysis and exploration (inversely) dependent on the number of available compute nodes. Furthermore, he highlight the utility of the environment by showing two new and innovative uses of such a solution.

Overall, similar paradigms of off-loading compute to clouds such as this is very benificial and is worth additional exploration.

References

[1] Alan Antonuk. Simpleamqpclient: Simple c++ interface to rabbitmqc-c, 2015. [Online; accessed 12-Jan-2015].

[2] Intel Corporation. Embree: High-performance ray tracing kernels, 2015. [Online; accessed 13-Jan-2015].

[3] Ganglia. Ganglia monitoring system, 2015. [Online; accessed 12-Jan-2015].

[4] Inc. Google. Protocol buffers: Google's data interchange format, 2015. [Online; accessed 12-Jan-2015].

[5] Sandia National Laboratories. Umbra, dante, operations viewshed (opshed) analysis, opinsight visualization, scream behavior engine, cstk. `http://umbra.sandia.gov/products.html`, 1997–2014. [Online; accessed 13-Jan-2015].

[6] Sandia National Laboratories. Umbra simulation framework. `http://umbra.sandia.gov/`, 1997–2014. [Online; accessed 18-Sept-2014].

[7] Tamara K. Locke. Guide to preparing SAND reports. Technical report SAND98-0730, Sandia National Laboratories, Albuquerque, New Mexico 87185 and Livermore, California 94550, May 1998.

[8] Peter M. Mell and Timothy Grance. Sp 800-145. the nist definition of cloud computing. Technical report, Gaithersburg, MD, United States, 2011.

[9] MiniMega. Minimega – vm management utility, 2015. [Online; accessed 12-Jan-2015].

[10] OpenStack. Openstack cloud software, 2015. [Online; accessed 12-Jan-2015].

[11] Pivotal. Rabbitmq - messaging that just works, 2015. [Online; accessed 12-Jan-2015].

[12] PuppetLabs. Puppet configuration management system, 2015. [Online; accessed 12-Jan-2015].

[13] Rolf Riesen. How to be conformant. *Psychology Today and Tomorrow*, 784(3):121–130, 2002.

[14] SaltStack. Saltstack configuration management, 2015. [Online; accessed 12-Jan-2015].

[15] John E. Smith. On the use of dry erase markers in science. *Journal of Pen and Pencil*, 784(3):121–130, 2002.

[16] Wikipedia. Cloud computing — Wikipedia, the free encyclopedia, 2015. [Online; accessed 12-Jan-2015].