Large Scale, Multi-Domain Analysis using Systems Tool Kit (STK) and High Performance Computing (HPC)

David Meyer[1] and Sylvain Dupont[2]

I. Background

The space community faces a pool of scenarios and questions that are nearly as infinite as space itself, but when it comes to the matter of space surveillance networks (SSNs), all of these scenarios share two common problems. First, space is congested, contested, and competitive – the so-called "three Cs" [Schulte, 2011]. This environment is already a challenge to any currently operating SSN, and it will be increasingly challenging as the population of space objects continues to grow. Second, budgets (government and commercial alike) are constrained. The juxtaposition of these two problems raises such questions as:

- How can the performance of a given space surveillance network (SSN) architecture be quantified?
- Given reasonable constraints/bounds, what would be an optimal (best performance/least cost) SSN architecture?
- What, if any, common characteristics are there among optimal/nearly optimal architectures?
- If an organization needs to make an improvement, where should they place that effort for maximum effect?

Although these example questions are straightforward - pertaining to fundamental matters of architecture performance – the answers can be elusive. At the Air Force Institute of Technology's (AFIT) Center for Space Research and Assurance (CSRA), students are engaged in answering these questions and many others. While space research can (and preferably, should) be conducted in space, the cost of designing, building, and launching space vehicles is a powerful limiting factor on how much research can be accomplished in this manner. Therefore, computer-based modeling and simulation (M&S) has to be a significant part of any overall research effort.

This paper will address the approach taken by the CSRA toward improving computer-based modeling and simulation (M&S), particularly by implementing AGI's Systems Tool Kit (STK).

II. Tools

Regardless of the need that you are trying to satisfy – optimization, stochastic modeling, sensitivity analysis, vulnerability assessment, or just a quantification of how a current of future system might perform – three primary factors define the challenge for computer-based M&S. First, nearly all space scenarios are

[1] Modeling and Simulation Team Lead, Center for Space Research and Assurance, Air Force Institute of Technology, 2950 Hobson Way Bldg 640, Wright-Patterson AFB, OH 45433
[2] Director, Software Architecture, Analytical Graphics Inc., 220 Valley Creek Blvd, Exton, PA, 19341

computationally intensive and have long run-times. Second, since space is, as previously noted, congested, it is dangerous to attempt to draw conclusions from small scenarios that only evaluate one or two space objects; the problem is unlikely to scale well. Finally, a scenario may need to be run many (perhaps thousands) of times, particularly if you are performing optimization or stochastic modeling. If you combine these three factors, you can easily imagine that running a sufficiently large scenario a sufficient number of times would be a computational challenge that a desktop computer cannot handle in a timely manner (or perhaps even at all).

If the required analysis is not feasible on a desktop, then what are the alternatives? The CSRA attacked this problem on three fronts: software, hardware and "peopleware".

A. Software

AFIT space students need access to a space/multi-domain modeling software suite that enables them to develop quality results from their research. Such a software suite must incorporate physics-based modeling for orbital mechanics, orbit determination, communications, sensors, and other domains (e.g., aircraft or terrestrial vehicles). For the research described in this paper, CSRA selected AGI's Systems Tool Kit (STK) and STK Engine as its primary modeling software suite and adopted Python and Linux scripting languages for automation.

1. STK and STK Engine

STK is a multi-faceted commercial simulation software product that allows aerospace engineers to model and evaluate different designs for air, space, land, and sea systems and their interactions.

The core concept of STK is a scenario. Scenarios are collections of objects that model a system. Objects in the scenario represent real-world entities (e.g., satellites, aircraft, places, area targets) or computational tools (e.g., access between two objects, constellations, coverage). As a desktop application (available on Microsoft Windows) STK provides a straightforward method for creating scenarios, without any programming expertise, using a graphical user interface (GUI). The desktop application also provides rich and accurate 3D visualization, which facilitates "debugging" the models assembled together.

STK Engine can consume and manipulate scenarios created with the STK desktop application. STK Engine is multi-platform (Windows and Linux) and is equivalent to STK without the GUI layer.

Figure 1 describes the architecture of STK and shows how the different layers are related.

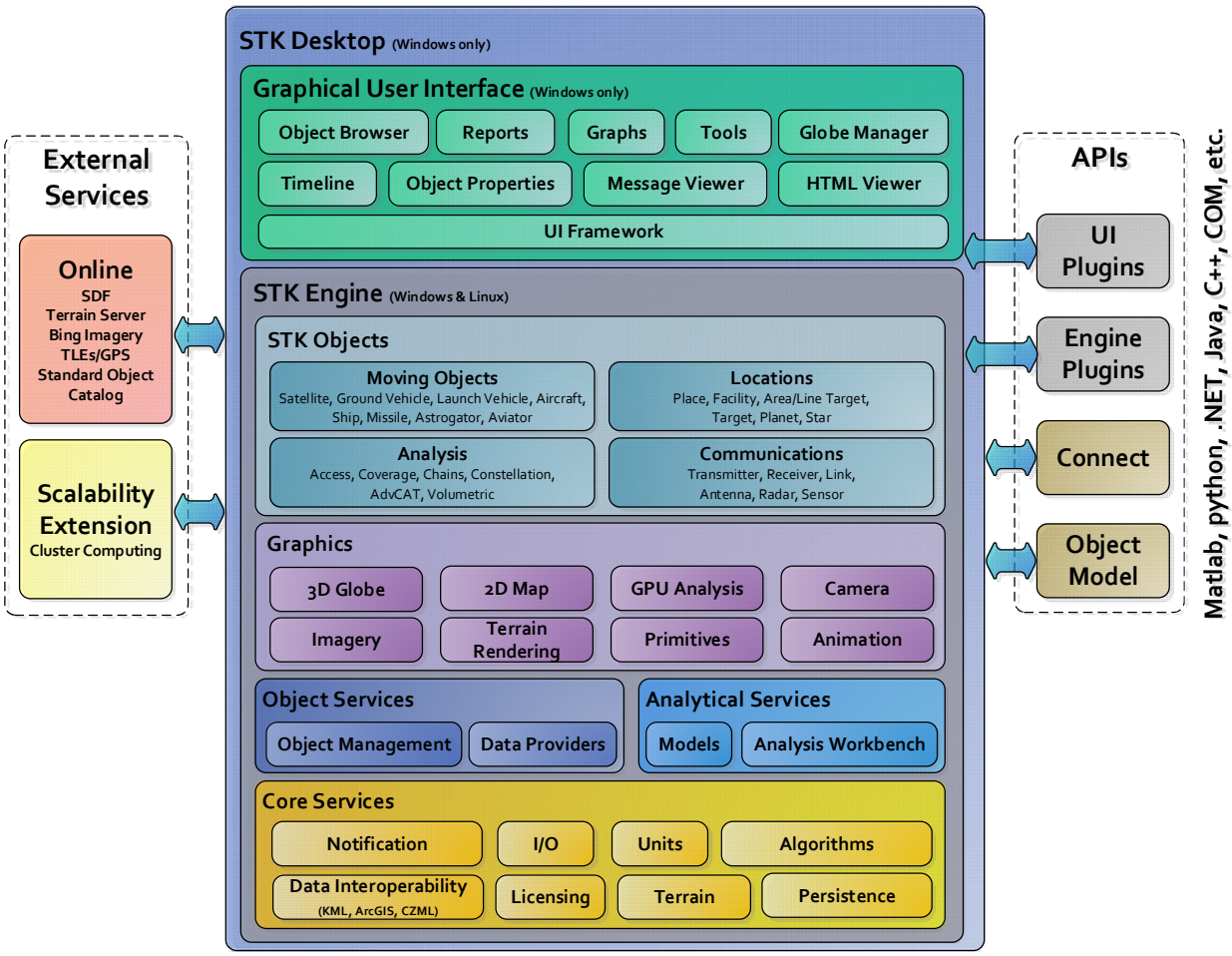

Figure 1. STK Architecture

At the high level, we can describe **STK** as consisting of two parts:

1. The **Graphical User Interface** (GUI) layer includes the various windows and user interfaces. This layer is only available on Windows.
2. **STK Engine** contains the application business logic and analytical capabilities. This layer is multi-platform and is available on both Windows and Linux. The STK Engine architecture can be further described as consisting of STK Objects, the graphics layer, and the Services (Object, Analytical, and Core).
    - STK Objects provide the simulation objects composed into scenarios. The services provided by the underlying layers dictate the implementation of these objects.
    - The graphics layer provides the 2D and 3D representations of the world and the simulation.
    - Finally, the Object, Analytical, and Core Services are the foundation and provide utilities and analytical capabilities assembled into the STK objects.

Since STK Engine does not have a GUI, it can receive interactions from a variety of application programming interfaces (APIs), including Python, MATLAB, and other programming languages such as Java and .NET.

In addition to excluding a GUI, STK Engine provides a "NoGraphics" mode, which excludes the graphics layer at run-time. This mode removes all dependencies on the graphics card and graphics libraries.

Unburdened by a GUI or graphics layer, STK Engine is well suited for running on headless compute nodes, running Linux. An STK Engine deployment on a node is comprised of STK Engine libraries and the associated data files; the process does not require root access. Flexible licensing available for STK and STK Engine also allows for the easy deployment of licenses.

This architecture provides the basis for a very efficient workflow, as described in Figure 2.

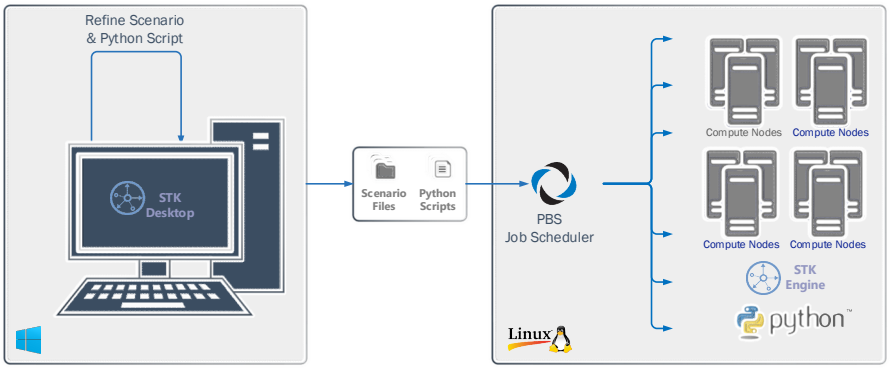

Figure 2. Workflow

This workflow facilitates the creation of multi-domain simulations that can then run on high-performance clusters:

1) An engineer uses the STK desktop application, running on Windows, to design the scenario. The GUI and rich visualization make it easy to define the simulation and iterate until you obtain the achieved system representation. No programming expertise is required at this step. While working with the GUI to perform interactive scenario creation (on a desktop computer) is useful to define a problem, it is not a practical approach for algorithms that require a large number of runs.
2) The scenario is tweaked using scripts written in Python. The scripts apply the various inputs to the model and then collect the outputs. These scripts are the "glue", varying the scenario parameters and extracting the results. You can execute and debug the scripts on Windows against the STK desktop application and then execute them against STK Engine.
3) An engineer deploys the scenario, with the accompanying scripts, to an HPC cluster running STK Engine on Linux.
4) High-level algorithms drive those scripts to implement various techniques (e.g., design of experiment, optimization, statistical, and stochastic methods). You can parallelize these runs to a massive degree, using many instances of STK Engine on a large number of compute nodes.

2. Python

Most people are much more comfortable using a GUI than writing a script to drive a software application. However, at some point - especially for large problems or problems that one must run repeatedly - writing and executing a script is faster than using a GUI, even if you are not good at writing scripts. Where that

threshold lies – from using a GUI to scripting - varies from person to person and problem to problem, but the CSRA strives to enable all of its students to write scripts by teaching scripting classes, maintaining code libraries, and providing a "CSRA Primer" that contains both code samples and explanations about how to write scripts. Although learning to write scripts is a considerable investment of time, it enables you to perform some kinds of modeling and simulation that would simply be impossible using a GUI. For example, true analysis requires some amount of repetition of the scenario(s), repetition requires batch processing, and batch processing requires a script. In addition, scripts provide a "digital hardcopy" of your work.

The CSRA decided that Python would be the scripting language of choice. Why Python?

1. Support for Python. Python has its own, thorough documentation, but, in addition to that, online resources such as Codecademy or Stackoverflow ensure that a simple internet search will provide the answer to any question about Python programming.
2. The price is right. Python is free and comes installed on most computers these days.
3. Availability. Typically, any U.S. Department of Defense (DoD) computer will have Python installed. Even if certain modelers do not have access to Python, it is not difficult to translate a Python script to another language.

Among other functions, students at the CSRA use Python to drive STK. Python scripts send all of the Connect commands that a student needs to send to STK - from creating a scenario to populating constellations, to creating reports. In addition, students use Python to addressing specific simulation needs. For example, a Python script performs the scheduling of sensors. The native scheduling capability in STK is not ideal for a target-dense, multi-priority environment in which sensors must maximize the number of observations on different targets as much as possible. By calculating accesses in STK and then post-processing those accesses, the simulated scheduling can better mimic the scheduling appropriate to the scenario.

Something that was not obvious until students immersed themselves in the expanded M&S effort was the amount of post processing that would be required. Even small desktop projects can generate gigabytes of data in reports that cumulatively are millions of lines long. The result is that you can quickly generate so many results that a human would not be able to make sense of them without help. For the CSRA, this help comes in two forms: post-processing, and data visualization with Paraview.

a) Post-processing

Post-processing is the method by which data sets that are too large for an individual to comprehend are distilled down into results that can be observed and understood. At the CSRA, students use Python scripts to:

1. Open each STK generated report.
2. Parse out the information that is needed (discarding the rest).
3. Perform necessary calculations.
4. If needed, combine the data with post-processed data from other reports

5. Perform any other analysis that the modeler deems necessary (e.g., calculating means, medians, minimums, or maximums).

Post-processing scripts can rewrite data to make it more sensible to a reviewer or to facilitate plotting; Python has many great plotting tools, so for those cases where graphical data display makes sense, generating the graph should prove easy to do.

b) Paraview

Paraview is an open source data analysis and visualization application that the DSRC introduced to the CSRA. In 2017, students completed research using a genetic algorithm to find a nearly optimal terrestrial and space based GEO SSA architecture. The product of that research was a handful of 21-element "genes". However, while the work was an unprecedented leap forward in M&S capability, the presentation of results was difficult to understand and was devoid of the context needed to give the best information to modelers and decision makers. The genes are so large that it is difficult to understand what they mean and identify trends. Using Paraview, you can quickly see information that goes beyond just a final answer. For example, with a few mouse clicks, you can see things that are in common among the best performing solutions. Likewise, Paraview enables you, with very little effort, to identify strong correlations and the significance of individual predictors. Another advantage of Paraview is that it enables rapid, interactive data exploration; meaning that you can perform better, more thorough data analysis, even with large data sets. Paraview is a new tool for the CSRA, and so while its students have already recognized its power, they have not yet mastered how to use it to its greatest effect. *More information on Paraview can be found at: https://www.paraview.org/*

3. Linux Scripting

A shell script is necessary to run instances of STK in batch mode on an HPC (whose OS is Linux-based at its very core). The script performs three essential tasks:

1. It passes arguments to Portable Batch System (PBS), which is the program that places the program into the HPC queue. Where the program lands in the queue depends on its size and the loading on the HPC.
2. It starts STK and, following a pause sufficient for STK to have completed its start-up process, it starts the Python program, and then sends the Python commands to STK.
3. It monitors instances and progress, closing and restarting instances (as the correct instance) that never start, or start but fail.

B. Hardware

The CSRA has grown from using relatively basic HPCs such as the AFRL DSRC Utility Server - on which engineers learned that STK 9 could not run on an HPC, but that STK Engine 10 in NoGraphics mode could - and the HPC Spirit, which has since been decommissioned. The CSRA now uses the Thunder HPC (at the time that this paper is being published; see Figure 3.).

Figure 3. The Thunder HPC. More details can be found at: https://afrl.hpc.mil/hardware/index.html

Thunder is located at the AFRL DSRC at Wright-Patterson AFB. It has 16 login nodes and 3,216 standard compute nodes. Each compute node has 36 cores, meaning that it can run up to 18 instances of STK. Theoretically, Thunder could run almost 58,000 simultaneous instances of STK, although the most that the CSRA has actually run is 10,000 instances. At the time of installation, estimates rated Thunder as the 25th fastest HPC in the world. The first students who worked with Thunder logged over 3,000,000 core hours of run time during their tenure with the CSRA.

Of note, it is probably not true to say that STK will run on any supercomputer. Each supercomputer manufacturer uses their own proprietary software, which is very lean for speed. Running STK on SGI computers has been relatively trouble-free with the advent of the "no Graphics" feature for STK Engine, but that success may not transfer to supercomputers from other manufacturers.

C. Peopleware

Obviously, there is no supercomputing without an HPC, nor are there any space scenarios without space modeling software. Regardless of how important those components are, however, neither of them could have achieved anything without "peopleware" – the human beings operating the computers and software. CSRA students are either astrodynamics or systems engineers, most of whom typically do not have much (or any) experience with STK, computer programming, or HPC operations when they begin working. The CSRA has taken several steps to ease the M&S mechanics burden on students so that they spend less time programming and more time researching.

1. Created the CSRA Primer for Modeling and Simulation. The Primer attempts to provide all of the information that a student will need to know regarding space M&S. Using the Primer, a student can learn how to write Python scripts, script space scenarios for STK and ODTK, data mine, get an HPC account and run scripts in batch mode, and more. The Primer contains many code snippets so that the students can apply themselves to the research without occupying too much time teaching themselves how to write scripts.

2. **Build Center-wide expertise**. In addition to the normal flow of students, the CSRA benefits from a relationship it has with NASIC. New NASIC hires, while awaiting their clearances, have the opportunity to go to AFIT and learn the same M&S fundamentals that the typical Air Force students do. The advantage of this relationship is that the NASIC employees, unlike the students, are not participants in the typical academic cycle; this creates a continuous pool of expertise at the CSRA that does not churn with every graduation. In addition, since the NASIC hires are not engaged in thesis research, the CSRA can assign them to projects where it has identified the greatest need.
3. **Build code libraries**. There is little academic or research value in struggling with the syntax required by Connect - the module in STK that takes scripted commands, sent as text, to execute various actions in STK. Therefore, the CSRA develops code libraries, so that if a student needs a specific functionality, most of the code for that functionality is already available for them. For example, there are code samples for a script that downloads all available TLEs, strips off undesired TLEs, and puts the desired TLEs into an STK scenario for visualization and analysis. The CSRA has also created code that acts as an interface with Connect for many common Connect commands. For example, if a student wants to create a scenario, the interface accepts simplified start and stop times and, after doing minor error checking in the proper format, sends the proper command to Connect. In addition, the interface keeps a running list of all Connect commands that did not receive an ACK (Acknowledgement), to facilitate troubleshooting.
4. **Offer a space modeling and simulation course**. Though it evolves over time, the CSRA developed an introductory course specifically for space students who are doing modeling and simulation for their research. The course allows them to build scenarios, as part of their coursework, which ultimately will be a significant component of their thesis research. This course also introduces students to the help available at AGI's support center, the DSRC help desk, and the DSRC Advanced User Support Teams.

Admittedly, the steps described above are specific to AFIT and may not be universally applicable. Until the DSRC stands up a Space Modeling PETTT team (Advanced User Support), performing space modeling and simulation will require significant effort. However, there are some mitigating factors. The CSRA, on a limited basis, can provide help, advice, and working code if necessary. In addition, the CSRA has been working with personnel at the DSRC to run STK for over three years now; thus, a phone call to the DSRC help line should connect those who need help with personnel experienced in overcoming technical issues with running STK on an HPC. From the perspective of the CSRA, parallelizing STK on an HPC has presented challenges, but the return on that investment is an ability to model and simulate completely new classes of problems that otherwise would be impossible to attempt.

III. The Role of High Performance Computing in Space M&S

Now the general who wins a battle makes many calculations in his temple ere the battle is fought. The general who loses a battle makes but few calculations beforehand. – Sun Tzu

Space operations are incredibly complex - governed by laws of physics that are not intuitive to many warfighters, occupying an enormous trade space, concerned with objects that are not visible to the naked eye, and confronted by new threats, issues, and concerns that arise every day. To understand, experiment with, and explore space scenarios, an organization must have computer hardware that can run space-modeling software on the appropriate scale. The focus of this paper, however, is on those problems where high performance or parallel computing is an entering argument for success.

A. Scenarios Requiring High Performance Computing

1. Optimization Problems

Given that the government perspective on space M&S is a constant demand for the "best performance" at the lowest cost, there is no shortage of space scenario subjects that could benefit greatly from optimization, and in particular, multi-objective optimization. Clearly, from a program or acquisition standpoint, quantitative analysis to maximize performance will be an optimization problem of some kind. However, the benefits of optimization are not limited to new systems. If an organization wants to improve the performance of an existing SSN architecture by adding sensors, relocating sensors, or developing better scheduling, optimization is the tool of choice; but it is also a computationally intensive process.

In general, optimization requires a multitude of runs to sweep through a sufficient portion of the (vast) trade space to find optimal, or at least nearly optimal, solutions. For example, a traveling salesman problem requires an enormous amount of computational power, since genetic algorithms require you to run many generations on optimizing sensor scheduling. Solving such a problem without an HPC would be nearly impossible, since the run-time on a desktop computer would be prohibitive.

2. Stochastic Modeling and Monte Carlo Simulation

An important distinction between STK scenarios and the real world is that an STK scenario is deterministic, whereas the real world is, obviously, not. For example, in STK, you can have constant access between space-based sensors pointing at ground-based targets as long as any constraints (e.g., slew rate) are satisfied and there is a line of site between the objects. In the real world, however, there might be clouds between a sensor and its target (obscuring the target from view), or a tasking priority that precludes the sensor from making a particular observation, or another variable factor that interferes with access. Thus, the best description of the likelihood of access at a point in time is not merely the probability - it is the probability and a measure of the variance, as well. How is the uncertainty determined? In stochastic modeling, you perform repeated problem runs in which all of the probabilistic influences (such as, in this example, the presence of clouds) are inserted into the scenario according to their probability distributions. The final performance, with variance, is the product of hundreds, to thousands, of data runs.

Monte Carlo simulations are similar to stochastic modeling, but have a special focus; these simulations calculate probabilistic outcomes that do not have a closed-form solution. For example, you can use a Monte Carlo simulation to analyze the effects of systems (with corresponding performance chains) on individual platforms. Suppose that an aircraft drops a guided munition that hits its target following two different Gaussian distributions, one in the X direction and one in the Y direction. If the munition's target is a runway adjacent to a baby milk factory, what is the likelihood that it hits the runway or the factory? You

cannot directly calculate the answer. In a similar manner to the stochastic modeling method discussed above, a Monte Carlo simulation inserts the probabilistic influences on the scenario according to their distributions in each of many runs. The results (perhaps the number of runway hits divided by the number of weapons launched, for example) are determined by the accumulated individual runs.

In both cases – stochastic modeling and Monte Carlo simulations - problems that require probabilistic inputs or outputs can only be solved with a multitude of runs, and therefore are difficult, even impossible, to calculate on any kind of computer other than an HPC.

3. Real Word Scale Problems

One of the challenges with space scenarios is the sheer size and complexity of some of the problems involved. For example, one area of research focus for AFIT has been GEO Space Situational Awareness (SSA). To keep the problem constrained to a computational load that a desktop computer could manage in a timely manner, you could limit the calculations to a subset of notional sensors and a subset of the GEO catalogue. However, the GEO SSA catalogue is large (and growing) and unevenly distributed around the globe, while at the same time the sensor search area is enormous and there are not enough sensors to make SSA as persistent as is desired. To take such a vast problem and reduce it to subsets that you can run on a desktop computer would present a serious risk of erroneous results, since the challenges posed by size of the problem are part of what you need to analyze and quantify in the first place.

However, the problem of "real world" scale goes beyond SSA. For example, debris field analysis and avoidance - with potentially hundreds of thousands of pieces of debris - requires high performance computing, particularly if the analysis is part of a responsive effort. Real world scale also takes on a new meaning in today's commercial CubeSat environment; where the constellations of imagers used to consist of a handful of satellites, commercial imagery enterprises now have constellations consisting of hundreds of satellites (see https://www.planet.com/products/monitoring/ for more details).

Another category of real world space scenarios that may require high performance computing is multi-domain ISR scenarios. For space-based ISR scenarios, stakeholders are usually interested in information such as revisit rates or coverage, with some measure of the quality of the resulting imagery in terms of NIIRS (National Imagery Interpretability Rating System) or ground sample distance (GSD). It is easy to define these terms with small constellations and a few targets, but the terms become much more time-dependent when you use multiple constellation planes or orbital altitudes (or regimes) [Ford et al, 2016]. If you add additional domains to the scenario - such as the incorporation of UAVs for imagery, in addition to space based assets - then you must contend with two computational complications. First, there will probably be too many imagery assets for you to perform analysis (and not just visualization of a single scenario) on a desktop computer and get the data that you need in a timely fashion. Second, revisit rate - in the traditional sense of the term - is now almost meaningless, since performance (in terms of both access and image quality) varies so much with time; descriptive statistics and time domain plots of performance derived through post-processing would be much more informative with respect to system performance.

The examples provided above illustrate just a few of the space scenarios that must take real world scaling into account.

4. Trade Space Studies / Sensitivity Analysis

Both trade space studies and sensitivity analysis, by their very nature, require many scenario runs; this is because their common purpose is variable (predictor and response) exploration.

Trade space studies involve comparisons of performance across an entire range of operation. For example, consider the question of imagery resolution versus coverage. If the imager on a satellite is held constant, but the altitude of the satellite is raised, then obviously the coverage will increase. However, the increase in coverage comes at the expense of image resolution. Are there some altitudes where both coverage and resolution performance are acceptable? One factor that complicates this question is that as a satellite's altitude increases and the sensor footprint increases in size, the speed of advance of the footprint over the earth's surface will slow – is this acceptable? To answer these questions, you could design a scenario that is realistic for the expected system tasking and then run it repeatedly, stepping through a broad swath of altitudes, exploring the trade space and quantifying the performance at each altitude until you determine the best operational altitude for your needs.

Sensitivity analysis is a method for solving questions that are similar, but distinct, from the questions answered by trade space studies. For example, conventional wisdom tells us that if one sensor is good, two sensors must be better. However, when other factors are considered, we can see that this trend does not extend linearly forever. At some point, the added benefit of yet another sensor to an already large pool of sensors does not justify the cost of the additional sensor. This so-called "knee in the curve" point is a frequent target in the acquisition community; it is the point at which you gain the most benefit with respect to the expenditure of resources. How do you find this "knee"? The answer is sensitivity analysis – that is, to make many runs of a scenario, varying the predictor variable (in this example, the number of sensors) and using the model to calculate both the expected system cost and expected system performance, such as coverage or revisit rate. You can then plot performance versus cost to identify the "knee in the curve". Because this analysis requires you to run the scenario a tremendous number of times to plot the curve, you will need to employ parallel computing for all but the most trivial of these types of problems.

IV. Case Study

Several AFIT CSRA students have been engaged in research to find notional, nearly optimal GEO space situational awareness architectures, using both space and terrestrial based sensors. The goal for these architecture designs is to optimize them with respect to minimizing the size of detectable objects, the latency between observations, and the total cost of the architecture. Further details of this research can be found in [Stern, et al, 2018], [Bateman et al, 2018], and [Felten et al, 2018].

However, from the CSRA's perspective, what is most important about the research is not the specific answers that it produces, but rather the nurturing of the cumulative modeling, simulation, and analysis process and capability. Development of this capability is not merely incidental; it is vital. Space has become

the newest warfighting domain, but the cost of the domains that it U.S. is already engaged in challenge its ability to compete effectively in this new domain. The U.S. must use resources efficiently and deploy systems optimally; but how can it ensure that it succeeds in these regards? The CSRA's research program has determined that the best answer is by employing parallel processing of space scenarios using the hardware, software and peopleware that are appropriate for the task. To illustrate the point, this paper will now examine the M&S methods developed from the CSRA's research on GEO SSA optimization.

To begin the GEO SSA optimization research, a student writes a Python script for STK on a desktop computer, which creates a toy version of the SSA scenario that they want to evaluate. The CSRA students always begin their research on a desktop computer for visualization purposes because STK's excellent visualization, reports, and graphs are vital for the debugging and verification of the initial scenarios that they create.

Developing a scenario that correctly models all of the components of a problem is challenging. As this paper has already discussed, space scenarios, with their myriad assumptions and choices, are remarkably complex. How does the modeler know that all sensors are working as envisioned, or that a sensor can only observe a target that is big enough, illuminated enough, and reflective enough in the right direction, or that the scheduling of observations is functioning correctly, or that terrestrial telescopes are not taking observations at noon local through clouds? The modeler must intensively evaluate the results, question them, and verify them against what the results should be.

Once a student verifies the toy scenario, they modify the Python script that created it so that the scenario can run on an HPC and then transfer the script to the AFRL DSRC's HPC, Thunder. There are two primary differences between the toy scenario and the version of it created for the HPC. First, the toy scenarios evaluate only a subset of the GEO catalog while the HPC versions evaluate the entire (unclassified) GEO catalog set of TLEs. Second, a physically plausible (in terms of size and locations), notional, terrestrial and space-based GEO SSA architecture is inserted into the HPC scenario, with several factor levels available for sensor size, number of sensors, and ground location or orbit. The combined effect of these differences is that the HPC version of the scenario is much larger than the toy version, making it better suited for drawing inferences about a nearly optimal architecture.

In addition to increasing the scenario size, The CSRA adds several other components to the HPC evaluation – custom sensor scheduling, genetic optimization, instance monitoring, and Paraview data conversion.

The sensor scheduler that is native to STK looks at a target for as long as possible, whereas in an SSA scenario, an operator normally employ sensors to observe as many targets as possible. The CSRA developed code that uses accesses generated by STK to feed a greedy heuristic. The heuristic, after accounting for sensor transit, settling, and integration time, maximizes the number of targets observed and prioritizes targets that do not have a recent observation recorded.

The CSRA adds genetic optimization to the HPC evaluation by implementing a Python package called, "Inspyred" [Garrett, 2017]. Genetic optimization provides an efficient method to sweep an enormous trade

space - some scenarios have had at least 10^{19} different combinations of predictor variable values (e.g., 0.5m, 1.0m, or 1.5m diameter terrestrial telescope apertures for each telescope at each site).

A Linux script written by the CSRA handles instance tracking for the HPC evaluation. The script monitors each instance of a scenario run and then restarts or re-runs an instance should it initially fail. There are two reasons why this script is important. First, while the probability of any individual instance failing is low, the 24+ hour run-time combined with the thousands of instances that are run ensures that there will be a non-zero number of failures. The second reason is that, as the number of instances of STK running simultaneously increases, the number of non-starts increases at a greater than linear rate. AGI determined that the solution to this second problem is to increase the amount of time that STK will await license retrieval before closing. However, the CSRA completed and tested the Linux script that solved both problems before AGI determined the solution, and this script remains their solution. Since it was not clear how it would affect the GA if many instances of STK randomly dropped before completion, the CSRA concluded that it was important to ensure that all instances run to completion.

The final component that the CSRA adds to the HPC evaluation is a small Python program that translates each final generation GA produced architecture to CSV text format, so that the visualization tool, Paraview, can open it. The DSRC offered Paraview (https://www.paraview.org/) to the CSRA M&S Team after the team approached them for help with visualization. The CSRA team recognized that visualization plays a key role in enabling both researcher and sponsor to understand what the results mean. The results of the initial exploration of GEO SSA Architecture optimization [Stern et al, 2018] were difficult to interpret, since they were displayed as a 1x21 matrix in which each column was another variable and the value in the column was the optimal value for that variable, as identified by the GA. Because the matrix was completely context-free, it provided the researcher with an enormous amount of data, but with no logic behind it.

Paraview, on the other hand, performs data analysis on the fly without seeming as if that is what it is doing. The user never sees an F-statistic, a P value, or a T-statistic. Rather, the user sees graphical output that shows context and relationships, and which allows for quick and easy data exploration, all at the click of a mouse. Figure 4 is a Parallel Coordinate Plot created by Paraview, showing the respective values for the number of sensors for each sensor site/orbit of the ~1,900 optimal fron of non-dominated solutions. The darker, thicker lines in this figure indicate that there are more near-optimal solutions for those values than there are for values connected by lighter, thinner lines; thus, typically, a near optimal solution for this architecture would include four sensors at the site "La Palma". Perhaps even more important than Paraview's images is the speed at which it produces them, which facilitates data exploration.

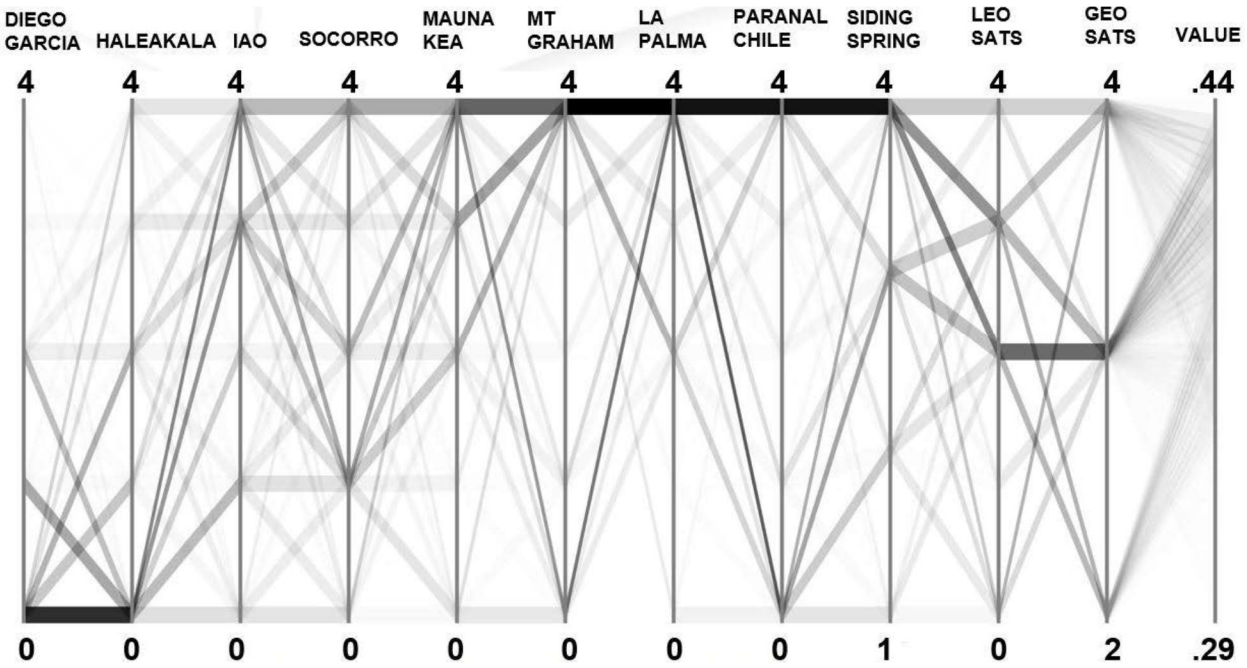

Figure 4. Parallel Coordinate Diagram of notional architecture elements.

Remember, the goal of the GA optimization is to produce a set of nearly optimal architectures (we say, "set", because the optimization is on multiple objective functions and thus different architectures may be optimal based on the weighting given by the decision maker for each individual objective). You would expect that across the set of nearly optimal architectures, there would be many characteristics in common and several actionable trends. You would also expect, in general, that if you looked at a collection of architectures that produced the optimal values, you could obtain great insight about what makes an architecture perform well and what does not. Examples of such insight from Figure 4 would include:

1. If the SSA community were to invest in another telescope(s), it should not be located in Diego Garcia (cloud cover severely limits SSA gain for the cost).
2. Sites such as Mt Graham, La Palma, Paranal, and Siding Spring are strongly maxed out in the number of sensors assigned (to control the size of the problem, the maximum number of sensors allowed per site was set to 4). It would be worth raising the maximum allowed number to see if these sites would use more sensors, and how much of a performance increase there would be.
3. Space-based sensors *have to be* part of the solution set. Note that most of the near optimal solutions have at least two LEO satellites and that there is an absolute minimum of two GEO satellites.

The speed and ease-of-use of Paraview enables data exploration in ways that are impossible with tables of data and summary statistics. In its own way, the wealth of information provided by visualization tools is every bit as vital to analysis as the ability to do parallel processing. Without the visualization of results, the challenge of wading through all of the data produced and turning that into actionable information is nearly insurmountable.

V. Results

As we have discussed above, the true results of this case study are not specific answers such as where to invest in new telescopes or satellites; the results are the processes developed by AFIT CSRA students to solve the previously unsolvable. The combination of hardware, software, and peopleware that the CSRA assembled enables a quantification of architecture performance that can provide decision-makers with the data that they need to make informed decisions. Given the increasing importance (and difficulty) of SSA amid tight budget constraints, particularly within the DoD, the need for modeling and simulation is profound. This study has demonstrated that the tools required to perform such analysis exist. This paper has highlighted three pivotal components to a successful space M&S program:

1. HPCs. The DSRC's HPCs have proven to be exceptional at solving large-scale, computationally intensive problems, and doing so quickly. The computers are reliable and, most importantly, DSRC staffing for these computers is both talented and motivated. A problem that can be parallelized, even just pleasingly so (the CSRA's problems can be described as pleasingly parallelized because individual instances of the problems are run simultaneously, in parallel, each on its own core), is a particularly good fit for evaluation on an HPC. With an HPC, you can explore a vast trade-space in the same amount of time that it takes to run just one instance of a problem on a desktop computer.
2. STK. While there are many reasons why the CSRA chose STK to be their space modeling and simulation tool, the predominant ones were exceptional product support, the extensive collection of physics models, and the benefits of using software that is already in broad use in the community. Of particular note, the broad repertoire of physics models within STK ensures that the CSRA will continue to rely on it even as the center's capabilities expand over time. However, for all of STK's strengths, there are a couple of additional functionalities that would be highly beneficial to the DoD. For many reasons, space problems are generally broken into three main pieces: the satellite itself (orbit, coverage, access), the payload (e.g., imagery quality or link budget), and orbit determination/SSA. For the research conducted at the CSRA, the satellite models included in STK exceed the requirements; the sensor scheduler, however, does not. AGI offers a separate scheduling module, but that package does not have a Linux variant and as such, custom post-processing is required for any scheduling beyond the default version. In addition, models that are necessary for orbit determination/SSA are unavailable for STK; AGI does produce Orbit Determination Tool Kit (which offers a collection of models useful for orbit determination and SSA), but it, too, lacks a Linux variant. Without this modeling, you cannot run SSA scenarios - arguably, the least defined aspect of space modeling and simulation - on an HPC unless you perform the orbit determination with many simplifying assumptions and workarounds.
3. Visualization. The key to understanding the terabytes of data produced on an HPC while solving a problem is visualization of that data. The CSRA students excel at scripting complex scenarios and working through hardware and software shortcomings, but a human cannot sift through all of the generated data and see trends or identify "the answer" without visualization tools. Although HPCs are necessary for running the problems, the visualization tools are even more important; without them, all that you have is data, not information.

VI. Conclusions and Next Steps

A. The Need for Computational Horsepower

Space scenarios, regardless of their size, are computationally intensive. Adding real-world size, scope, or complexity increases the need for fast, parallelized processing. Performing optimization, stochastic modeling, or sensitivity analysis contributes an additional layer of processing need to the problem. Fortunately, the research described in this paper has demonstrated that you can use high quality M&S software such as STK – in combination with scripted automation and custom functionality - on an HPC to solve such computationally intensive problems. For example, one CSRA student, working on optimizing GEO SSA architectures, evaluated approximately 225,000 architectures in 2 days; work that would have taken over 110 years to run to completion on a standard desktop computer.

B. The Necessity of Scripting

Scripting enables the repetition needed for optimization and stochastic modeling, and allows you to implement custom functionalities that may be unavailable in the primary M&S software that you are using. In addition, well-designed scripts are requisite for the batch processing that runs the parallelized scenarios, monitors node health, and checks for report completion. Perhaps most importantly, scripts are necessary for post-processing to include the usage of optimization routines.

C. Summary of Capability

The CSRA team has demonstrated the power of state-of-the-art hardware, software, and peopleware when they are intelligently combined to solve space problems. The synergy of these three forces has enabled the CSRA to take a potent leap forward with space modeling and simulation, analyzing scenarios that previously would not have been feasible to attempt. CSRA students have focused on optimization, but their research has demonstrated that other computationally intensive analyses, such as stochastic modeling and Monte Carlo simulation, are also possible using the approach that they have developed. Space modeling and simulation, enabled by high performance computing, has the potential to provide answers that are sorely needed by leaders and decision-makers in the space community, the DoD, and other relevant organizations. However, with respect to what this technology can do, the surface has only been scratched.

D. Next Steps

AFIT has three main goals:

1. To further the education of students by, in part, enabling great research.
2. To conduct research that solves problems for the USAF and the DoD.
3. To further the state of the science.

To these ends, the CSRA continues to push aggressively towards solutions for space M&S problems, not just with the goal of answering specific questions, but also to uncover tools and methods that will improve the quality of the research. The visualization effort described in this paper is but one example of the ways

in which the CSRA deliberately looks for the means to get better at what they do. Improvement of M&S must be an ongoing process, and the AFIT CSRA team is continuing to broaden its experience base and take advantage of the many high-powered tools of the trade, such as STK and HPCs.

In addition to the CSRA's efforts, several features on the AGI product roadmap will facilitate HPC activities. Improved support for Python on Linux is planned for a future release; specifically, enabling tighter integration between Python scripts configuring a scenario and STK Engine. This advancement will expose more fine-grained APIs, providing better performance and more flexibility with how Python can modify a scenario. AGI also plans to support the STK Parallel Computing Extension on Linux. This support will accelerate computations within the same scenario (currently Coverage, DeckAccess, Chains, Volumetric) when STK Engine runs on an HPC. In addition to these currently planned improvements, AGI is evaluating whether other products might benefit from running in a similar architecture. For example, the ODTK product roadmap includes porting the ODTK engine to Linux so that the same approach will be feasible with ODTK. Using STK Components, a set of low-level Java and .NET libraries, is another possibility.

Finally, the CSRA has learned that (at the time that this paper is being published) the DSRC is building a new, classified HPC that will greatly improve the utility of the research performed by CSRA students; instead of being limited to notional problems, students will be able to research real answers for real problems. More importantly, this development is a significant step forward for the USAF and DoD – organizations with a pressing need for analysis of actual systems, architectures, and scenarios.

VII. References

- Bateman, Jr., Mark G., Colombi, John M., Cobb, Richard G. and David W. Meyer (2018). Exploring Alternatives for Geosynchronous Orbit Space Situational Awareness, Journal, Military Operations Research (MOR). (Submitted March 2018)
- Felten, Michael S., Colombi, John M., Cobb, Richard G. and David W. Meyer (2018). Multi-objective optimization using parallel simulation for space situational awareness, Journal of Defense Modeling and Simulation (JDMS). (Submitted March 2018)
- Ford, T., Meyer, D., Colombi, J., Scheller, B., and Palmer, C., Journal of Defense Modeling and Simulation https://doi.org/10.1177/1548512916656291, 2016
- Garrett, A., "inspired (Version 1.0.1) [Software]" Available: https://github.com/aarongarrett/inspyred
- Schulte, Gregory, L., 2011, https://www.spacefoundation.org/news/schulte-space-congested-contested-competitive
- Stern, Jordan L., Wachtel, Steven T., Colombi, John M., Meyer, David W. and Richard C. Cobb. (2018). Multi-objective Parallel Optimization of Geosynchronous Space Situational Awareness Architectures. AIAA Journal of Spacecraft and Rockets (Accepted, March 2018)