Dynamic Resource Allocation in an HPC Environment

Duane A. Gilmour and James P. Hanna
Air Force Research Laboratory, Information Directorate (AFRL/IF), Advance Computing Technology Branch, Rome, NY
{gilmourd, hannaj}@rl.af.mil

Gary Blank
Metron, Inc., Solana Beach, CA
blank@ca.metsci.com

Abstract

Analysts within the military strategic planning process need to be able to anticipate and respond in real-time to a dynamically changing battlespace with counter actions. The capability is required to accept current information into a simulation and rapidly peer into the future at any given moment to derive hypotheses about future alternatives. It is virtually impossible to identify or predict the specific details of what might transpire. Our research interest is to develop techniques to assess potential courses of action (COAs) against the adversarial environment. Utilizing HPC technology, multiple force structure simulations can be dynamically executed in parallel to concurrently evaluate the hypothesis of assessing a given COA against a range of adversarial eCOAs. The uncertainty of the adversary decision process requires simulation capabilities that spawn multiple simulations from critical decision points to evaluate alternative eCOAs. The focus of this paper is on the development of a simulation framework that provides the foundation for faster than real-time parallel COA simulations. Within this framework is a requirement to be able to dynamically allocate and reallocate resources in an HPC environment. The authors will describe techniques to clone and create variant simulations in real-time and the dynamic system requirements and use of HPC nodes.

1. Introduction

Military campaign planning is a rigorous process that includes many man-months of preparation and analysis. The process of determining the most likely actions and reactions in operational situations attempts to encompass both friendly and adversary decisions. The sequence of decisions and actions, from a friendly or an adversary perspective, that are accomplished related to a mission, can be defined as a course of action (COA)[1]. Traditionally, friendly COAs are wargamed against the "most likely" and "most dangerous" adversary COAs, herein referred to as eCOA. Wargaming is a recorded "what if" session of actions and reactions designed to visualize the flow of the battle and evaluate each friendly COA[2]. This is a highly manpower intensive process that doesn't account for a dynamically changing situation. The number of COAs wargamed is limited due to the enormous manpower resources required to evaluate each scenario. In addition, as the situation unfolds, a dynamic and volatile adversary can drive the friendly decision process, without much opportunity to evaluate potential outcomes.

There has been significant research in recent years relating to automating the wargaming process to attempt to shorten the decision cycle. Unfortunately, these conventional wargaming simulations typically execute a pre-scripted sequence of events for an adversary, independent of the opposing force. In addition, conventional wargames focus on traditional attrition-based force-on-force modeling, whereas modern campaign strategies employ and evaluate a mixture of kinetic and non-kinetic operations. One such campaign approach being pursued within the Air Force is effects-based operations (EBO). EBO is an approach to planning, executing, and assessing military operations that focuses on the effects produced from military activities, as opposed to the direct result of attacking targets[3]. A significant challenge for EBO is predicting and assessing how friendly actions result in adversary behavioral outcomes, and how those behavioral outcomes impact the adversary commander's decisions and future actions. For wargame simulations to be effective in an effects based arena, they must allow users to evaluate multiple ways to accomplish the same goal with a combination of direct, indirect, and cascading events (actions). Wargames must also be dynamic, in that the adversary force will react to friendly force actions in an unscripted manner. Current

wargame simulators are incapable of assessing an effects-based campaign approach and do not account for dynamic adversarial behavior.

2. Objective

Analysis systems are needed in the military strategic planning process to anticipate and respond in real-time to a dynamically changing battlespace with counter actions. Our research interest is to develop techniques to assess potential COAs against the adversarial environment. Utilizing HPC technology, multiple force structure simulations can be dynamically executed in parallel to concurrently evaluate the hypothesis of assessing a given COA against a range of eCOAs[4]. The uncertainty of the adversary decision process requires simulation capabilities that spawn multiple simulations from critical decision points to evaluate alternative eCOAs. The desired goal is to establish a means to evaluate the COA for critical elements related to execution and timing as well as overall effectiveness in the presence of a range of adversarial possibilities that may occur.

The focus of this paper is on the development of a simulation framework that provides the foundation for faster than real-time parallel COA simulations. The Synchronous Parallel Environment for Emulation and Discrete-Event Simulation (SPEEDES) framework was chosen to be the foundation on which to build the system because it helps exploit available high computational resources and provides much needed functionality. This framework must be able to dynamically allocate and reallocate resources in an HPC environment. Notionally, an initial emulation, or basis simulation, could be running in the real-time environment on 4–6 nodes. At critical decision points, the emulation can be cloned and the cloned simulation would rapidly execute into the future to evaluate the possible outcomes of a particular decision branch. The simulation clones require the HPC environment to dynamically allocate resources, as each decision branch will require additional HPC resources. There is uncertainty with respect to the resource requirements throughout the entire analytical process, but the framework being developed will allow the simulation to take advantage of all resources available. The authors will describe the techniques to clone and create variant simulations in real-time and the dynamic system requirements and use of HPC nodes.

3. Methodology

In this section, we will describe our approach for building a system to assess possible COAs and eCOAs. The first thing to note is that the system is based on the SPEEDES parallel simulation framework. SPEEDES provides the means for distributing and coordinating simulations running on multiple CPUs. By running multiple parallel simulations simultaneously, we seek to apply the maximum computing power to the problem at hand. Also, leveraging this software suite will help us build the system much faster.

3.1. The Emulation.

The key to applying simulation technology in real-time is to maintain a simulated "mirror image" of the real world situation at all times. This simulated version of the real world can be used as the starting point for evaluating COAs and for simulating into the future to help predict what might happen. We refer to the simulated mirror image as the *emulation*. The emulation is a parallel simulation running at wall-clock (i.e., real-time) speed that is continually fed intelligence, surveillance, and reconnaissance (ISR) reports so that it reflects the state of the battlespace to the extent known. The emulation will run continuously on k processors of an n-CPU HPC; the remaining n-k processors will be available to run COA evaluations or predictive simulations. For example, if our HPC has 256 CPUs and the emulation occupies 8 CPUs, the remaining 248 CPUs will be available for performing COA assessments.

The SPEEDES external module facility is a key component for building the emulation. The external module is a separate program attached to a simulation (it communicates with the simulation using sockets). It has the ability to: send/receive messages to/from the simulation, control the time advance of the simulation, kill the simulation, etc. The external module is the conduit through which ISR reports can be inserted into the emulation. Primarily, this will be done by scheduling events on simulation objects, which can alter the objects as desired. Also, there will be facilities for creating new objects or deleting existing ones. These mechanisms provide the capabilities needed to integrate ISR reports into the emulation state.

3.2. Simulation Cloning.

After researching several design approaches to creating copies of the emulation for alternative COA analysis, we selected the "cloning approach". Cloning a simulation means creating a running copy of the simulation, preferably on a separate group of "free" CPUs (i.e., processors with little or no workload). For example, if a simulation is running on CPUs 1–8 on a 128-CPU machine, we might clone a copy onto CPUs 9–16. The clone will be an exact duplicate of the original and will produce identical results as the original. By cloning the emulation and allowing it to run as fast as possible, we get

a glimpse into a possible future; this is what we call a predictive simulation. To simulate a COA, we clone the emulation and insert information into the clone (see Figure 1). This information defines the COA to be simulated.

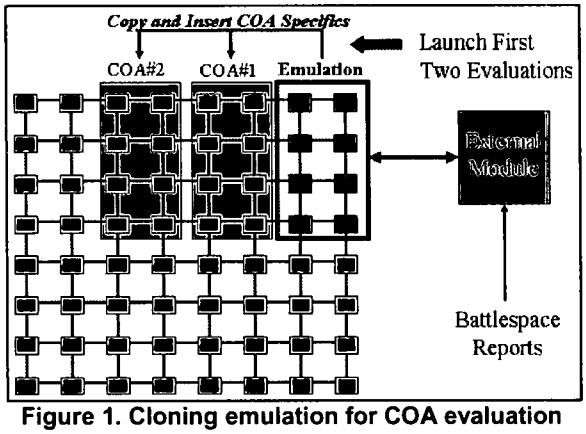

Figure 1. Cloning emulation for COA evaluation

Leaving aside the usual complications that inevitably accompany software modifications, the basic procedure for cloning a simulation is straightforward. Each SPEEDES node is copied onto an available CPU (using `fork()` or a similar utility), creating *n* child nodes; then the *n* child nodes are connected with a new set of communications links (shared memory and/or TCP/IP sockets), see Figure 2. Once the child nodes are sewn together with communications links, the clone simulation is integrated and can run forward to the designated time.

An important issue to be dealt with is controlling *where* each child node is run. One reason this is a concern is we would like to determine which nodes can communicate through shared memory. The typical HPC consists of groups of CPUs (called *compute nodes*) that can access shared RAM. A 128-CPU machine, for example, might be composed of 16 compute nodes containing 8 CPUs each. Only processes on the same compute node can communicate through shared memory (which is much faster than socket communications). SPEEDES simulations can be configured to use shared memory communications where available. To maximize performance, we need to maximize the number of shared memory links; but this requires the ability to specify the CPU on which each child node runs. What is needed is a utility like that provided by Beowulf Cluster software, – `bproc_rfork(int cpuID)` – which copies a process and migrates it to a specified CPU.

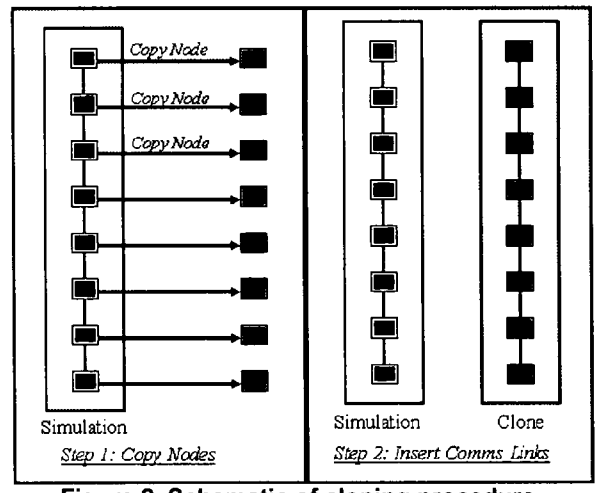

Figure 2. Schematic of cloning procedure

Another reason for wanting such a utility is because it is somewhat simpler to duplicate the configuration of the parent simulation in the clone. Ideally, the emulation will be run with a favorable configuration (in particular, all nodes linked with shared memory communications), and all clones will simply replicate this topology. However, since it may not always be possible to find a set of CPUs with the desired topology, the mechanism must be capable of cloning a simulation onto any set of *n* CPUs (where *n* is the number of SPEEDES nodes in the parent simulation; this fixed number of nodes constraint will be discussed below).

3.3. COA Evaluation.

In order to run a COA evaluation, we need to alter the clone so that it simulates the COA instead of just duplicating the parent simulation. All that is required to accomplish this is to pass a list of commands to the clone. Each command instructs a specific object to perform a given action at a specific time; in aggregate, these commands define a COA.

There is a simple mechanism for inserting such a list into the clone. First, the user provides the list (in some form) to the interface for requesting COA evaluations. This request is sent to the emulation and converted into an event that acts on a special *EvaluationManager* object. This object stores the list and then initiates the cloning process, which creates a copy of each node (e.g., using `fork()`). This copies all data in each node (which is a process), including the list of commands. In this way, the list is transferred to the clone. The *EvaluationManager* object in the clone takes the list and uses it to schedule a series of events that simulate the agenda of commands. Meanwhile, the *EvaluationManager* object in the parent simulation can just delete the list in order to save space.

One part of the cloning design which requires explanation is the constraint that the clone and parent simulations have the same number of nodes. In theory, there is no reason why the clone cannot have a different number of nodes from that of the parent. However, this would mean redistributing the simulation objects over the child nodes, which would be difficult and complicated. Also, since one can usually find a good fit of nodes to a given application, there probably won't be a great deal of utility in dynamically varying the number of nodes in the clone. Because of this, we opted for the simpler design of keeping the number of nodes fixed.

3.4. Managing HPC Resources.

In order to get the best performance out of our system, mechanisms are needed to manage available resources. The most important of these is available CPUs, but we also need to keep track of socket ports and shared memory. To manage CPUs and ports, our system will use a *ResourceAllocator* server. This program will begin by reading in a file that describes the HPC and designates a set of port numbers available for use. The HPC description will specify the number of available CPUs and list their IDs. Also, it will define the shared memory topology of the machine by listing groups of CPUs that can share RAM.

The *ResourceAllocator* will manage the allocation/deallocation of CPUs and ports. For example, when the user asks for a COA evaluation, a request for *n* CPUs and one port will be sent to the *ResourceAllocator*. If these resources are available, the request will be granted, and the COA evaluation can go forward using the granted resources; otherwise, it must wait. When the evaluation completes, it sends a message to the *ResourceAllocator*, returning the resources. In this way, the *ResourceAllocator* always knows which CPUs and ports are available for use.

The rationale for this process is to control the workload on each CPU so as to attain optimal (or at least good) performance. If we do not carefully balance the number of processes with available CPUs, some CPUs will become overloaded and slow the progress of one or more simulations. Thus, our plan is to map each process to its own CPU. These processes will be simulation nodes or *SpeedesServers* (the latter manages TCP/IP communications within simulations and between external modules and simulations).

The other resource to be managed is shared memory. Where possible, SPEEDES uses shared memory for internode communications since it is much faster than sockets. Since shared memory is limited, we must make sure to return allocated shared memory to the operating system when each simulation completes.

4. Application

The simulation framework provides the foundation for faster than real-time parallel COA simulations. As described previously, COA analysis or wargaming is the process of performing "what if" analysis of actions and reactions designed to visualize the flow of the battle and evaluate each friendly COA. Utilizing the developed methodology, along with performing "what if" analysis in an HPC environment, affords the opportunity to evaluate friendly COAs against a range of eCOAs. This range of adversarial COAs goes well beyond the "most likely" and "most dangerous" COAs as previously described. It will encompass a dynamic adversary that responds in an intelligent manner based on friendly actions and adversary models.

There is a significant amount of uncertainty that accompanies any adversarial modeling capability. This uncertainty encompasses the process of decision making in a dynamic situation. Typically, models are abstractly created to reflect the adversary's beliefs, goals and intentions, which are based on friendly interpretation of the adversary. The uncertainty of the adversarial decision process makes it necessary to evaluate friendly COAs against a range of eCOAs. You can imagine, based on interpretations of the adversary, that numerous reactions are possible in response to a friendly action. It's these action/reaction dynamics that we are trying to capture within the COA analysis process. By simulating numerous COAs prior to and during engagement, it may be possible to estimate outcomes of adversary actions immediately after they are accomplished within an operational situation. This will allow decision makers to better respond during execution. In the following sections, we will describe examples of the application of the developed methodology and the requirements for dynamic resource requirements in an HPC environment.

4.1. Static Resource Availability for COA Evaluation.

Consider an example where we would like to evaluate two friendly COAs against a range of eCOAs. Let's assume we are limited to a 64 CPU HPC resource. If we choose to utilize 4 CPUs for emulation, we would be limited to 15 parallel COA evaluations, see Figure 3. Remember the requirement discussed in Section 3.3, that each simulation clone requires the exact same number of CPUs as the emulation. Each arrow in the Figure represents a distinct COA/eCOA analysis. In this example we are evaluating 8 distinct eCOAs against friendly COA#1 and 7 distinct eCOAs against friendly COA#2. The constraint of 64 CPUs limits the number of decision points, which can be evaluated with respect to

the adversary, to three. Each decision point represents an adversary's reaction as a result of a friendly action. As one can imagine, there is a range of decisions that must be evaluated after each action/reaction within COA analysis and limiting ourselves to three may cause a potentially dangerous adversary decision to be overlooked. However, even though the static HPC resources applied in this example limits the number of adversary reactions within COA/eCOA analysis, the mechanism and methodology being developed is a significant improvement over current wargaming technologies.

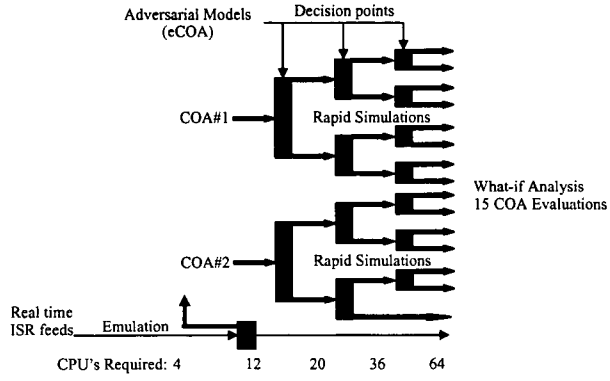

Figure 3. Simulation cloning example

In this example, we demonstrated the need for increased CPU resources during the analysis. However, there will be situations where some of the COAs converge or are extremely unlikely, in which case they need to be pruned, and CPU resources would be made available for additional COA analysis. The COA/eCOA analysis process will be a continuous give and take of HPC resources.

4.2. Dynamic Resource Availability for COA Evaluation.

Now, let's assume we have HPC resources available in a dynamic environment. Same example as before, we want to simulate two COAs but are not limited by the number of available CPU resources. We'll also assume that the COA analysis is being performed on 4 CPUs each. By having more resources available, it would be possible to evaluate more branches at each decision point. In the previous example, we only evaluated two high probability responses to each friendly action. Where having dynamic resources available will be of benefit is when the level of uncertainty is greater within the adversary model. This may occur in a situation with an extremely volatile adversary, or where little is known of the adversary decision process. Having more resources available allows the analyst to pursue many alternate decisions within the COA analysis process. Many of the decisions points could be unlikely, but the analyst is only limited by the amount of computational resources available. Not only will the increased resources allow the analyst to pursue more responses to a particular friendly COA action, it will also allow analysis of more decision points. We were limited to three decision points in the previous example. It is possible to imagine that the simulation of 2 COAs on 8 CPUs could easily expand to hundreds or even thousands of parallel analysis requiring enormous HPC resources. We don't, however, know how many resources will be required at the beginning of the analysis. There are many unknowns associated with the COA analysis and decision space and having access to dynamic HPC resources allows for greater versatility when performing the analysis. Here, we have only exemplified 2 planned friendly COAs, consider the necessity of evaluating many friendly COAs against many adversary COAs, the HPC requirements could be enormous.

4.3. Dynamic Resource Availability for Predictive Simulation.

The previous examples discussed the utilization of HPC resources from a static perspective, i.e., evaluating planned friendly COAs against a range of adversarial possibilities. This would be accomplished prior to plan execution. The first example was constrained by limited HPC resources, whereas the second exemplified the need for greater dynamic resource availability. There is a requirement, however, for analysts to be able to continuously assess changing conditions during execution and anticipate future events. The ever changing dynamics of operations, as well as the difficulty of modeling the adversarial decision process, requires continuous plan evaluation and prediction. This is a much more dynamic application of the methodology along with HPC resource allocation. The concept of mirroring the state of the battlespace within emulation is consistent as before. The major difference is utilizing the emulation as a starting point for parallel predictive simulation, vice COA evaluation. Here, the concept would be to constantly mirror the ongoing operation and race ahead in time to evaluate possible futures, given the current state of the operation, along with plans and tactics, as best known. This allows for dynamic situational assessment, comparison of the actual verses planned COA, as well as providing alerts to the decision makers on new threats or opportunities. This will allow analysts to be able to anticipate and respond in real-time to a dynamically changing battlespace with counter actions and determine whether engagement results are consistent with predictions.

The previous examples utilized 4 CPU's for emulation as well as for each COA simulation. In a more

dynamic execution situation, there may be a desire to utilize even more resources. Of course, this will be driven by the potential latency verses speedup associated with utilizing many more CPUs. The dynamics of resource allocation may be even greater than during the planning process. As each new event occurs in the battlespace, or as more intelligence information surfaces, the analyst may want to estimate potential futures based on this new information. There will be a continuous demand for more resources, based on the constantly changing emulation state. In addition, because of the constantly changing emulation state, there will be constant pruning of simulations, which release HPC resources. The simulations become obsolete due to the adversary's decisions and actions. The simulation process is based on a range of adversary decisions, and not specifically one decision. In this instance, resources will have to be released as the obsolete simulations are pruned, and repopulated with the emulation state to again race ahead in time to estimate possible futures.

5. Summary

The innovative application of HPC technology provides the foundation to utilize analytical simulations in static as well as dynamic fashion. This is a major paradigm shift in the use of simulation technology, i.e., moving from static to dynamic simulations. The methodology allows for the evaluation of alternatives when considering the adversarial environment, prior to and during operational engagement. The concept also provides the capability to accept current real-time information and be able to rapidly peer into the future at any given moment.

In a military application, this capability can be applied such that a simulation tool could emulate the current operating picture, and be able to rapidly peer into the future at any given moment and assess possible COAs. The application of dynamic resources available in an HPC environment affords the opportunity for faster than real-time parallel COA simulations. This technology can be applied during the Effects Based Operations (EBO) planning process, as well as in a predictive dynamic situation. In either application, decision makers will have a better vision of the future battlespace as well as an increased awareness of the impact of decisions when evaluating an adversarial situation. Both are necessary in the ever changing dynamics of adversarial situations and the ability to stay within the adversary decision loop.

5.1. Conclusion.

A framework and concept for dynamic resource allocation in an HPC environment has been described. This is a major shift from current HPC utilization procedures. It brings to the forefront the necessity to have resources available, at any given moment. The examples described here were related to COA/eCOA analysis. The requirements for dynamic HPC resource allocation will continue to grow in the future. The necessity for real-time decision making in the presence of enormous quantities of data, and a dynamic volatile adversary, is one such example. To realize a future of information and decision superiority, HPC resources will continue to be pushed beyond current operating procedures.

References

1. Joint Publication 1-02, *Department of Defense Dictionary of Military and Associated Terms*, March 2004.

2. Joint Publication 3-30, *Command and Control for Joint Air Operations*, June 2003.

3. McCrabb, Maris J., "Concept of Operations for Effects-Based Operations." Aerospace Command, Control and Intelligence, Surveillance, and Reconnaissance Center, Langley AFB, VA, February 2002.

4. Surman, Joshua, Robert Hillman, and Eugene Santos, Jr., "Adversarial Inferencing for Generating Dynamic Adversary Behavior." *Proceedings of the SPIE 17th Annual International Symposium on Aerospace/Defense Sensing and Controls: AeroSense 2003*, Orlando, FL, April 2003.