Fall 2017

Maneuverable Applications: Advancing Distributed Computing

William Clay Moody
United States Military Academy

Amy W. Apon
Clemson University, aapon@clemson.edu

Maneuverable Applications: Advancing Distributed Computing

Dr. William Clay Moody
Dr. Amy W. Apon

ABSTRACT

Extending the military principle of maneuver into the war-fighting domain of cyberspace, academic and military researchers have produced many theoretical and strategic works, though few have focused on researching the applications and systems that apply this principle. We present a survey of our research in developing new architectures for the enhancement of parallel and distributed applications. Specifically, we discuss our work in applying the military concept of maneuver in the cyberspace domain by creating a set of applications and systems called "maneuverable applications." Our research investigates resource provisioning, application optimization, and cybersecurity enhancement through the modification, relocation, addition or removal of computing resources.

We first describe our work to create a system to provision a big data computational resource within academic environments. Secondly, we present a computing testbed built to allow researchers to study network optimizations of data centers. Thirdly, we discuss our Petri Net model of an adaptable system, which increases its cyber security posture in the face of varying levels of threat from malicious actors. Finally, we present evidence that traditional ideas about extending maneuver into cyberspace focus on security only, but computing can benefit from maneuver in multiple manners beyond security.

1. INTRODUCTION

Cyberspace research has focused on applying traditional military doctrine and strategies into this new operational domain [3]. Two of the more popular concepts and topics have been situational awareness [4, 7] and key terrain [14]. An essential military principle that has received extensive theoretical investigation is maneuver [5]. This theoretical research has laid the foundation and created opportunities to build systems that possess the features of maneuver.

LTC William Clay Moody serves an assistant professor and deputy program director for Information Technology in the Department of Electrical Engineering and Computer Science at the United States Military Academy. A founding member of United States Cyber Command, he served as a Cyber Battle Captain of the Joint Operations Center, cyber defense planner for the Expeditionary Cyber Support Element—Iraq, and cyber capabilities engineer for J33 Current Operations. Clay is a 1998 graduate of Clemson University ROTC with a Bachelor's Degree in Computer Engineering. His graduate degrees are a Master's of Science in Computer Networking from North Carolina State University and a Ph.D. in Computer Science from Clemson.

Maneuver is one of the U.S. Army's nine Principles of War. Maneuver includes the application of combat power to maintain an advantage over the enemy [15]. The flexible and dynamic employment of resources ensures success by keeping adversarial conditions imbalanced and thus reduces failures, compromises, and vulnerabilities. The non-military use of the word maneuver describes an action that is not random or without purpose, but one that is clever or skillful. In both environments, the word maneuver implies deliberate movement and actions taken to achieve a specific purpose.

Distributed and parallel applications allow the execution of complex computations that previously were deemed impractical. Many recent technological advances have contributed to the widespread growth of distributed computing, namely multi-core processors, multi-processor nodes, high-speed networks, improved storage technologies, and virtualization. Regional and national researchers have combined funding and resources to build expansive, large-scale shared computing clusters to allow more efficient usage of power, space, and cooling to multiple user groups. These shared computing resources have become the standard high-performance computational platforms that annually appear on the list of the most powerful commercially available computer systems. Even with these tremendous achievements, the need for continued progress in resource availability, optimization, and security exists.

Our research is motivated by the increased interest in further defining and abstracting the concept of military maneuver in the cyberspace domain. As such, our research is focused on designing, building, and modeling maneuverable applications. We have coined the phrase "maneuverable applications" to denote the distributed and parallel systems and programs that take advantage of the modification,

Dr. Amy W. Apon is professor and chair of the Computer Science Division in the School of Computing at Clemson University. Apon joined Clemson in 2011 in this position. She was on leave as a rotator at the National Science Foundation during 2015, serving in the Computer and Network Systems Division for several programs, including the Computer Systems Research, Big Data, Smart and Connected Heath, and Extensible Parallel Systems programs. Prior to joining Clemson, Apon was the founding Director of the Arkansas High Performance Computing center. She holds an M.A in Mathematics and an M.S. in Computer Science from the University of Missouri–Columbia, and a Ph.D. in Computer Science from Vanderbilt University.

relocation, addition or removal of computing resources within the application, giving the perception of movement. These resources can be computational, network, or storage, or can be the applications themselves. These actions are deliberate, purposeful and meant to achieve an advantage over adversarial conditions. We have applied this approach to address three important topics within distributed and parallel systems, namely resource provisioning, application optimization, and cybersecurity enhancement.

2. MANEUVER FOR RESOURCE PROVISIONING

The first area of interest for studying maneuver in cyberspace platforms is in the area of resource provisioning. Our work with the Job Uninterrupted Maneuverable MapReduce Platform shows how a big data environment can be provisioned in a university setting within the current investment of high-performance computing. This is achieved by the use of maneuvering of nodes in and out of the cluster.

JUMMP, the Job Uninterrupted Maneuverable MapReduce Platform [12], is an automated scheduling platform that provides a customized Hadoop system within a batch-scheduled cluster environment. JUMMP enables an interactive pseudo-persistent MapReduce platform within the existing administrative structure of an academic high-performance-computing center by “jumping” between nodes with minimal administrative effort. Jumping is implemented by the synchronization of stopping and starting daemon processes on different nodes in the cluster. Our experimental evaluation shows that JUMMP can be as efficient as a persistent Hadoop cluster on dedicated computing resources, depending on the jump time. Additionally, we show that the cluster remains stable, with good performance, in the presence of jumps that occur as frequently as the average length of Reduce tasks of the currently executing MapReduce job. JUMMP provides an attractive solution to academic institutions that desire to integrate Hadoop into their current computing environment within their financial, technical, and administrative constraints.

A. Introduction

Hadoop is an open-source software tool used to implement MapReduce [1], a parallel programming paradigm for computation over large amounts of data using a cluster of commodity computer systems. Many organizations have built large-scale production data centers with dedicated computing resources for Hadoop clusters to support their analytic and scientific computation workloads. Hadoop has rapidly evolved and been adopted, thus creating a complex software ecosystem. System administrators are challenged to provide this service while maintaining a stable production environment. This is especially challenging at a typical centralized research institution where the computing infrastructures are designed to accommodate multiple research applications within existing financial, technical, and administrative considerations.

In an academic research environment, we can differentiate the usage of Hadoop into three different categories. The first category includes research applications that use Hadoop MapReduce as a tool for research purposes. These projects can either use MapReduce programs exclusively or use MapReduce programs as part of a larger workflow in a programming framework. Researchers may spend some time developing MapReduce programs and other necessary components and then focus on executing the programs to achieve the final results. As these are research applications, researchers typically alternate between running the computations and analyzing the produced outputs.

The second category is the study of the Hadoop MapReduce software suite itself. This includes studies of Hadoop MapReduce under different hardware and software configurations, development of improvements to Hadoop MapReduce, and implementations of different alternative parallel frameworks to Hadoop MapReduce. The testing of dynamic execution environments for Hadoop is difficult to do in most existing deployments.

The third category in an academic setting includes users in classroom environments who are attempting to learn to install, operate, and maintain a Hadoop cluster. While these assignments usually have short run times and use small data, the nature of the students' learning curve can lead to unintended consequences. In our experiences, teaching Hadoop MapReduce to undergraduates, we observed repeated problems, such as crashing of the Hadoop core processes, corruption of data on the cluster's storage nodes, and overloading of the cluster as students rush to complete the work before deadlines.

B. System Design

Inspired by military maneuver, our approach is to provision Hadoop as a dynamic execution environment that can be instantiated, utilized, and decommissioned when needed by a user. This is achievable from user space since initialized and starting up a Hadoop cluster does not require any administrative privileges. However, creating individual Hadoop clusters imposes overhead due to configuration, data loading and retrieval, and shutdown of the environment. Permanently dedicating a set of computational nodes to individual users places increased workload on administrators who must set policies for scheduling of Hadoop and non-Hadoop jobs while providing fairness and equal access. Our goal is to facilitate the setup of user-controlled dynamic Hadoop environments that execute within existing scheduling policies, including resource limitation, maximum usage time, and priority preemption, without administrative intervention.

The Hadoop cluster uses a single dedicated node for both the distributed file system metadata server and MapReduce master server. This dedicated node resides outside the control of the scheduler. Each worker node is scheduled as an individually scheduled job, which allows for preemption or failure of the job to only affect a single node of the Hadoop cluster. Each data storage and task execution node is established within its scheduled job. The job starts the Hadoop clients and connects them to the persistent head node. Each client waits for its trigger to jump. When the trigger to jump is received, the jumping node schedules its replacement. When notified of an upcoming jump of a slave node, the master node begins to copy the blocks stored on the outgoing node to the remaining nodes in the cluster. The master node immediately kills any task assigned to the outgoing node and reassigns them to available worker nodes. The flexible and maneuverable design of JUMMP allows us to support the user requirements presented above. By allowing

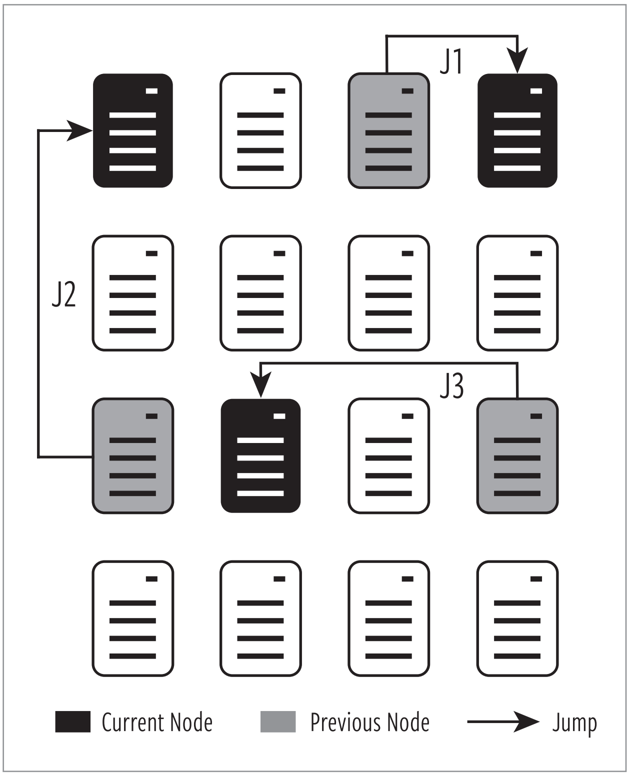

Figure 1. JUMMP maneuvers Hadoop client processing between nodes provisioning a resource within existing HPC environments

preempted or failed nodes to submit jobs for their replacements, the Hadoop cluster survives, and running jobs continue to execute until completion. Figure 1 illustrates how processes maneuver between nodes in a cluster to perform a big data analysis application.

C. Analysis

Adding maneuverability to a Hadoop cluster will obviously degrade performance. We evaluate this degradation to understand the trade-offs of this system. With each jump of a computational node, additional overhead occurs over a non-jumping cluster. This overhead comes from two separate sources: the scheduler overhead and replication of data blocks. As previously mentioned, JUMMP schedules each client as separately scheduled jobs within the supercomputing environment. Scheduling and queuing delays result in fewer worker nodes being available to perform map and reduce tasks. The JUMMP is undersized when replacement nodes in the queue are waiting to be assigned to an available node. When an existing node leaves the cluster, all blocks that are stored on its local storage must be replicated across the cluster. The master node begins this immediately upon the decommissioning of the outgoing node. This data replication consumes system resources that previously would have been used for the execution of MapReduce jobs.

This degradation is evaluated with two separate experiments on the performance of JUMMP in our HPC environment. The specifications of our system nodes are shown in Table 1. In each experiment, a baseline performance metric is established for a non-jumping Hadoop cluster by repeatedly running the same MapReduce job 100 times over a static dataset. The job is executed three additional times while varying the jump time for the cluster. We record the times of the jumps, the individual task start and stop times, and the overall job run time. With these results, we quantify the overhead of jumping during the execution of a MapReduce job. With the smaller dataset, a node can jump as fast as every seven minutes.

NODE	HP SL250s
CPU	INTEL XEON E5-2665 (2)
CORES	16
MEMORY	64 GB
LOCAL STORAGE CAPACITY	900 GB
NETWORKING	INFINIBAND

Table 1: Individual node specifications inside our compute cluster [12]

The experiments are executed on a homogeneous set of nodes within the local HPC environment to ensure uniformity of test results. All tests are run on an isolated pool of 96 nodes for worker nodes. At the allocation of the initial nodes and creation of the JUMMP, the dataset for the experiment is imported, the nodes start jumping, and the MapReduce job begins to run. We use the dataset and benchmarks from PUMA, Purdue's MapReduce Benchmark Suite[11]. PUMA is developed as a benchmark suite to represent a broad range of MapReduce applications exhibiting application characteristics with high/low computation and high/low shuffle volumes. The parameters of the experiments are shown in Table 2.

APPLICATION	WORDCOUNT	TERASORT
DATASET SIZE	50 GB	300 GB
NODE COUNT	8	32
JUMP TIMES [MINS]	7/10/15	20/40/60

Table 2: Experiment Parameters for our evaluations of JUMMP [12]

The experimental results and evaluation show that JUMMP can be as efficient as a persistent Hadoop cluster on dedicated computing resources, depending on the jump time. Additionally, results show that the cluster remains stable, with good performance, in the presence of jumps that occur as frequently as the average length of Reduce tasks of the currently executing MapReduce job. Our work and results show how maneuver can be used to adequately provision a MapReduce resource. Empowered by maneuver, this resource is available within an existing HPC environment with no additional personnel investment. We note that the allocation of resources that do not produce useful work represents a monetary investment. Though we do not quantify this investment, it can be considered a cost of maneuver for resource provisioning.

3. MANEUVER FOR APPLICATION OPTIMIZATION

Our second area of interest for maneuverability in cyberspace systems is application optimization. Our research with the Flow Optimized Route Configuration Engine (FORCE) investigates this enhancement. FORCE is an instrumented, representative network testbed in which the network topology of a cluster can be maneuvered to develop novel approaches to optimize traffic flow and timing of datacenter traffic [6].

FORCE emulates a data center network using a programmable interconnection controlled by a software-defined networking (SDN) controller. SDN combined with distributed and parallel applications have the potential to deliver optimized application performance at runtime. To investigate this enhancement and design future implementation, a datacenter with a programmable topology integrated with application state is needed. The FORCE advances us down the path towards this goal. We also utilize Hadoop as a case study of distributed and parallel applications along with a simulated Hadoop shuffle traffic generator.

The testbed provides initial experimental evidence of support to our hypothesis for future SDN research. Our experiments on the testbed show a difference in application runtime a factor of over 2.5 times on shuffle traffic for Hadoop MapReduce jobs and the potential for significant speedup in warehouse scale data centers.

A. Introduction

Modifying existing production datacenters or creating entirely new experimental ones to investigate the integration of SDN into datacenter networks is costly in time, dollars, and operational output. Historically, researchers have used controlled infrastructure, called testbeds, resembling real systems and networks to experiment on computing advancements. SDN and datacenter researchers can benefit from this approach, providing meaningful discoveries if established with realistic workloads and instrumented to provide constant, measurable results. Our system is an initial step towards providing a capability that leads to developing infrastructure and processes to understand this integration.

Our overall research goal is to investigate the application of maneuver technologies and methods for optimizing the performance and energy efficiency of parallel and distributed applications in a cluster environment. Goals include the study of how the performance of distributed applications is impacted by the network topology of the datacenter[9]. SDN is a technology that can be used to easily and temporarily reconfigure physical and virtual network topologies. The FORCE testbed is a low-cost experimental platform that uses a networked set of single computers and virtualization to emulate the performance of whole racks of machines in a data center and their applications. The testbed provides an SDN infrastructure that can be used to study how changes in network topology can impact the performance of the applications.

B. Architecture

A novel design aspect of the FORCE is the use of single workstations to emulate an entire datacenter rack that is full of computing nodes. This emulation enables the study of the inter-rack networking traffic for distributed applications at a very low cost, and the study of how topologies impact performance.

The hardware of the FORCE includes one primary server, twelve client workstations, and two SDN-enabled switches. The testbed is extensible and scalable to a very large size. The set of computers used in our testbed is repurposed from upgraded student laboratories and is installed in a location that allows students to have physical access to the equipment throughout the system building and experimentation.

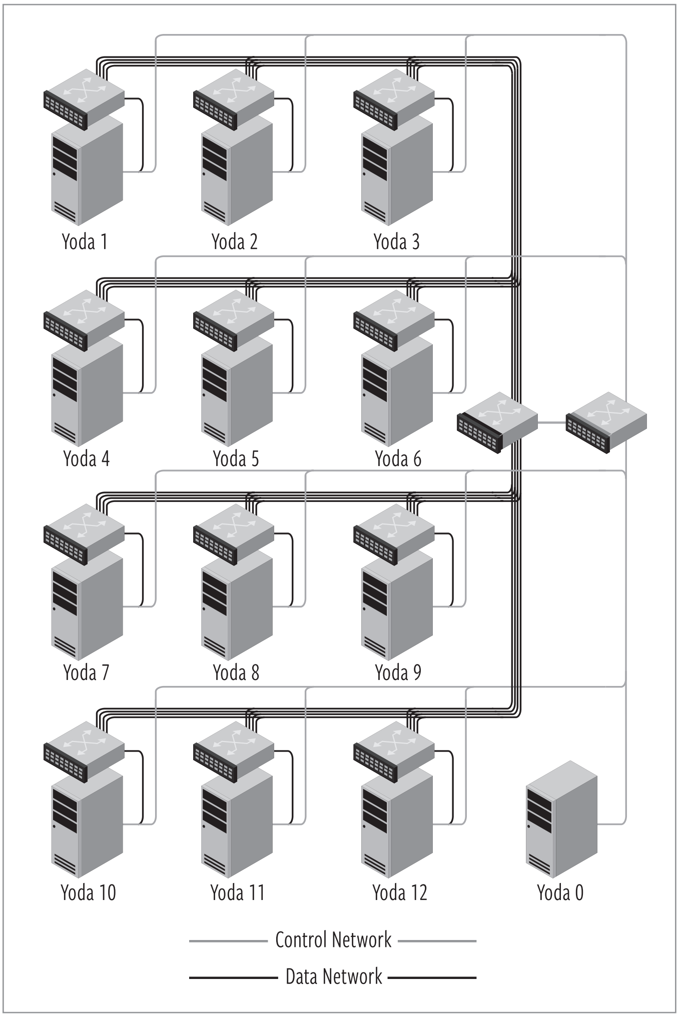

Figure 2. Wire diagram of the Flow Optimized Route Configuration Engine (FORCE) with a maneuverable network topology for application optimization

The two network switches are 48-port gigabit Ethernet switches (Pica8 Pronto 3290 48-port GBe OpenFlow-enabled) using OpenFlow [10], a popular SDN protocol. Two VLANs are established on one SDN-enabled switch for the "control" and "access" networks. The server and the workstations each have one connection to both of these VLANs. The remaining 48-port switch is connected to one of the four remaining gigabit Ethernet ports of the twelve workstations. This switch allows each workstation to be connected to a maximum of four other workstations with SDN controlled point-to-point connections. This switch is the primary target of the maneuver of the FORCE testbed. Figure 2 provides a wire diagram of the FORCE network testbed.

The core technology empowering the cyber maneuver is a custom virtual topology building package, the FORCE. The FORCE implements a virtual network topology by installing forwarding rules on the SDN switches in a cluster. The topology is described in a series of layers using the NetworkX Python package. Each layer maintains a network graph as well a procedure for discovering a path between any two vertices. The lowest layer contains information about the physical topology of hosts, switches, interfaces, and links, including characteristics such as hardware addresses and link speeds. The tool then applies subsequent graph layers that abstract each previous layer, building the virtual topology by translating edges into paths on the underlying graph. As a whole, this layered abstraction approach allows mapping of the desired connectivity among vertices on the highest level graph to the lowest level flow rules to be installed onto the SDN switches.

C. Experiment

Hadoop MapReduce is used to demonstrate the utility of the FORCE testbed. The literature [9] describes the potential speedup of MapReduce shuffle traffic that is possible by maneuvering datacenter racks that contain the Reduce tasks in close network proximity to racks that contain the Map task from the same MapReduce job. This proposed enhancement requires the dynamic reallocation of point-to-point connections between the top-of-rack switches in a two-dimensional torus topology. Our testbed is ideally suited for testing this optimization.

1. Hadoop Shuffle Simulator

The nodes comprising the testbed are not robust enough to run a real workload consisting of multiple Hadoop jobs or multiple virtual Hadoop nodes. To solve this problem, and to ensure that the testbed supports a realistic shuffle traffic network load that corresponds to the traffic between datacenter racks, we implemented a Hadoop shuffle traffic emulator within the FORCE. The emulator executes a centrally controlled software suite that synchronizes bulk network data transfers from Map tasks to Reduce tasks within the cluster. These data transfers are the same as the movement of Map tasks outputs to reducers across the cluster as seen in the shuffle phase of a real MapReduce job.

Given a set of configurations and experiments, the system can deploy the emulated MapReduce jobs across the cluster. These jobs perform the transfer of shuffle traffic. Based on the size of the data to be processed and the system block size, the simulator determines the number of Map tasks required for each job. The number of Reduce tasks is determined by a default global parameter or specific argument on a per job basis. After the system is configured with the number of Map and Reduce tasks per job and the size of the data transfer between all the Map tasks and the set of Reduce tasks in each job, the transfer of data begins. Since we are interested in studying the inter-rack traffic, any Map and Reduce tasks that reside within the same virtual datacenter rack do not transfer data.

2. Design

The testbed is configured as a 192-node cluster spread over twelve racks. Using bin scheduling as described in [9] three simultaneous MapReduce jobs are executed running with a single Reduce rack each and three Map racks. There are a total of 1GB of data to transfer from each Map task to its respective Reduce rack. A random placement of datacenter racks is placed into a 4x3 two dimensional torus topology, which simultaneously transfers the simulated shuffle traffic. Each experiment is executed 1000 times. The network topology and the shuffle times needed to complete all the transfers are recorded.

3. Analysis

With this and other experiments, a baseline for comparison with randomized topologies was established by measuring the results with 500 runs using a fixed network topology with no particular distinguishing characteristics. Statistical analysis shows significant difference in the baseline and random samplings. This is our first indication that different topology placements exhibit better and worse congestion characteristics.

Further analysis as shown in Figure 3 shows that worst case transfers times are 2.5 higher than best case transfers. A majority of the experimental runs fall in the middle in a high bell curve. This reinforces our hypothesis that intelligent maneuver of a network topology in a datacenter can maneuver away from worst-case situations, and may be able to maneuver toward optimal situations. These results support further investigation in future work of applying maneuver toward application optimization, specifically network traffic flow optimizations.

Our efforts and results show how maneuver can be used to optimize application performance. Specifically, the FORCE shows how maneuver network topology of the data center can improve the run-time execution shuffle traffic within a Hadoop cluster.

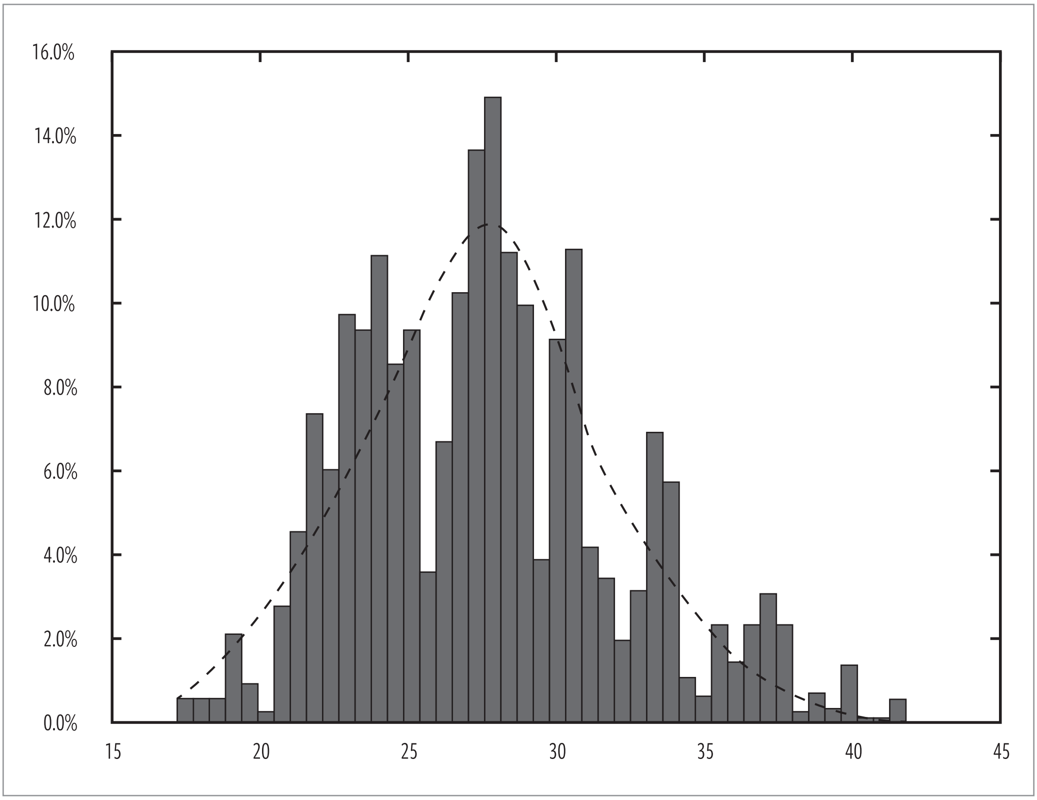

Figure 3. Histogram of simulated shuffle times evaluating the FORCE testbed under random topologies [6]

4. MANEUVER FOR CYBERSECURITY ENHANCEMENT

Our third interest area for studying maneuver in cyberspace is improving cybersecurity. Our work with the Defensive Maneuver Cyber Platform [8] introduces a Stochastic Petri Net model of improving the survivability of a distributed and parallel application with the additions of moving target defense and deceptive defense, two of the tactics of defensive cyberspace maneuver [5]. Our extended security analysis of the model provided mathematical evaluations, a prototype simulator of the event, and rules of thumb for employment [16].

Distributed and parallel applications are critical information technology systems in multiple industries, including academia, military, government, financial, medical, and transportation. These applications present target-rich environments for malicious attackers seeking to disrupt the confidentiality, integrity, and availability of these systems. Applying the military concept of defense cyber maneuver to these systems can provide protection and defense mechanisms that allow survivability and operational continuity. Understanding the tradeoffs between information systems security and operational performance when applying maneuver principles is of interest to administrators, users, and researchers. Our model enables the understanding and evaluation of the costs and benefits of maneuverability in a distributed application environment, specifically focusing on moving target defense and deceptive defense strategies.

A. Introduction

Multiple institutions in academia, industry, and government have discovered the necessity of parallel and distributed computing in data-driven business processes for the solution of complex computational problems. The significant financial investment and operational reliance of these systems create a critical infrastructure that is tightly bound to the success of the organization. The security of these platforms is vital to the survival of these establishments.

Malicious actors seeking financial or intelligence gains are targeting supercomputers and distributed computing centers at an increasing rate. Their methods and efforts to disrupt the confidentiality, integrity, and availability of the systems require network security professionals and researchers to invest remarkable amounts of time and money into protecting these assets.

We are motivated to attempt to improve the security of distributed systems by introducing the military concept of maneuver. Our approach to integrating maneuver into parallel and distributed computing is to add the elements of moving target defense and deceptive defense. The system is designed and modeled using a Stochastic Petri Net in which individual nodes maneuver between three different operating modes. The model is evaluated to understand how the state space and probability distributions are impacted under different configurations.

B. Background

Applegate [5] introduces four tactics of defensive cyber maneuver. Two of these elements are the subject of this application. Moving target defense is an attempt to change the attack surface of systems to cause an adversary to invest additional resources. This additional expenditure of effort increases the probability of detection, tactic compromise, and failure. Deceptive defense includes presenting decoy and seemingly susceptible systems as attractive targets. These targets are closely monitored and give an early indication of enemy activity while diverting attention away from legitimate and valuable systems. Our research focuses on modeling how moving target and deceptive defense can be integrated into a parallel and distributed computing system.

Petri Nets are a graphical, modeling system for concurrent systems. Visually a Petri Net is bi-partite, weighted, directional graph. There are two types of nodes, a place, indicated by a circle, and a transition, indicated by a straight line or bar. Weighted arcs connect places to transitions or transition to places. Tokens indicated by dots in the model can be found in places. Places represent conditions of the system while transitions represent actions taken when certain conditions are met. The presence of a token in a place indicates that a certain condition is true. A transition is considered enabled when input places have a number of tokens equal to or greater than the weight of the incoming edges. Enabled transitions can fire in a non-deterministic manner. When a transition fires, it removes tokens from input places equal to incoming edge weights and deposits tokens in out places equal to the weight of outbound edges.

Many extensions to basic Petri Nets exist. One such extension is the Stochastic Petri Net (SPN). Stochastic Petri Nets add a new node called a timed transition, indicated as an unfilled bar. Timed transitions introduce a delay between enabling firings. This delay is a random variable drawn from a Poisson distribution. Each timed transition has its own firing rate for its delay distribution. Once enabled, each timed transition computes its random delay. The timed transition with the shortest delay fires first. SPNs are an extension of Continuous Time Markov Chains and allow their associated mathematical tools to be applied to the analysis of SPNs.

C. System Design

Our system is composed of multiple nodes. Each node can run in one of three modes and nodes are constantly maneuvering between these modes. These modes are operational, idle, or deceptive. An operational node is an active contributor to the computation of the distributed system and is a valuable target that we want to protect. A deceptive node appears to an outside observer to be identical to an operational node by listening to the same network ports and sending and receiving traffic in statistically similar manners. The deceptive node, though, is not doing any constructive work for the system but running a monitoring tool that can detect an intruder, similar to a honeypot [13]. An idle node is a node that is in neither an operational nor a deceptive state.

In our SPN model, each single node is represented with three places and four transitions. The three places represent the three modes of operation. The four timed transitions are the maneuvers between operational and idle modes and deceptive and operational modes. A node cannot maneuver directly between operational and deceptive modes. Arcs connect places and transitions all with a weight of one. A single token can be found in one of the three places, indicating the current running mode of this node. Timed transitions from operational to idle and idle to operational have the same firing rate. This rate is referred to as the deceptive maneuver rate since that is the rate in which a node maneuvers towards deception. Likewise, the transitions from deceptive to idle, and idle to operation have the same operational maneuver rate.

The entire system is made up of multiple individual nodes with a few configurable parameters. N represents the total number of nodes, O is the minimum number of operational nodes the system must maintain, while D is the minimum deceptive nodes. It is important that the sum of O and D is less than N so that we can ensure we have a least one idle node so that the system can actually maneuver. The system wide operational maneuver rate is ro, while rd is the deceptive maneuver rate. The initial configuration of a system is set to be the minimum values of operational and deceptive nodes. Figure 4 depicts a system consisting of a total of eight nodes with a minimum of three operational nodes and two deceptive nodes.

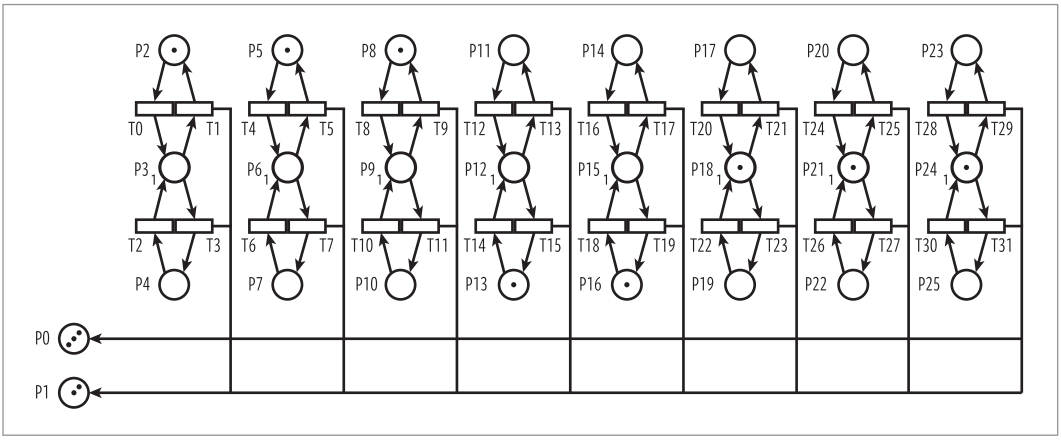

Figure 4. The model of defensive maneuver cyber platform with eight nodes [8]

D. Analysis

To understand all the potential configurations of the system with given set of parameters, we need to quantify the state space of the Defensive Maneuver Cyber Platform (DMCP). As previously mentioned, the system has a defined minimum value for the number of operational and deceptive nodes. Since N is fixed, the maximum value for the count of operational and deceptive nodes can be calculated to be N-D and N-O respectively. The total number of markings or states that the Petri Net can have is quantified by summing the count of valid markings for each valid combination of (o,d) which is the current number of operational and deceptive nodes. This formula is similar to the calculation of computation of multistate combinations since each of the N nodes can be in state operational, deceptive, or idle.

For a given N, the range of possible markings is inversely proportional to the minimum values O and D. The lower the minimum values, the larger the marking state space. The largest state space is when O and D are 0 and is approximately e^N. The minimum value for the state space is when one of the minimum levels of operational or deceptive node is maximized, and the other value is minimized. For instance, for N = 8, O=7 and D=0, there is a minimum of only 17 valid marking. As Figure 4 shows, this minimum plot is approximately on the order of log n.

One of the goals of the DMCP is to allow the system to change the rate of maneuver between deceptive and operational based on the threat conditions. Specifically, alarms and alerts from the embedded honeypots software in a deceptive node along with other external network security information will provide indications when the system should increase the deceptive maneuver rate. During a low threat environment, the operational maneuver rate can be increased so that the system has more operational capacity. We strive to study the impact that the maneuver rate has on the operational and deceptive composition of the system. An open-source SPN analysis tool[2] is used to calculate the steady state probabilities for the 8-3-2 DMCP shown in Figure 5. The stacked bar chart in Figure 5 presents the probability of states in which there are 3, 4, 5, and 6 operational nodes with varying ratios of the maneuver rates. As Figure 5 shows, as the probability of deceptive states increases, the probability of a deceptively maneuverable cluster rises to more than 95%. As the probability of operational mode increases then the probability of having 5 or more of our nodes operational (thus increasing operational output) rises to more than 95%. These results show that by adjusting the maneuver rates, we can influence the system towards operational or deceptive.

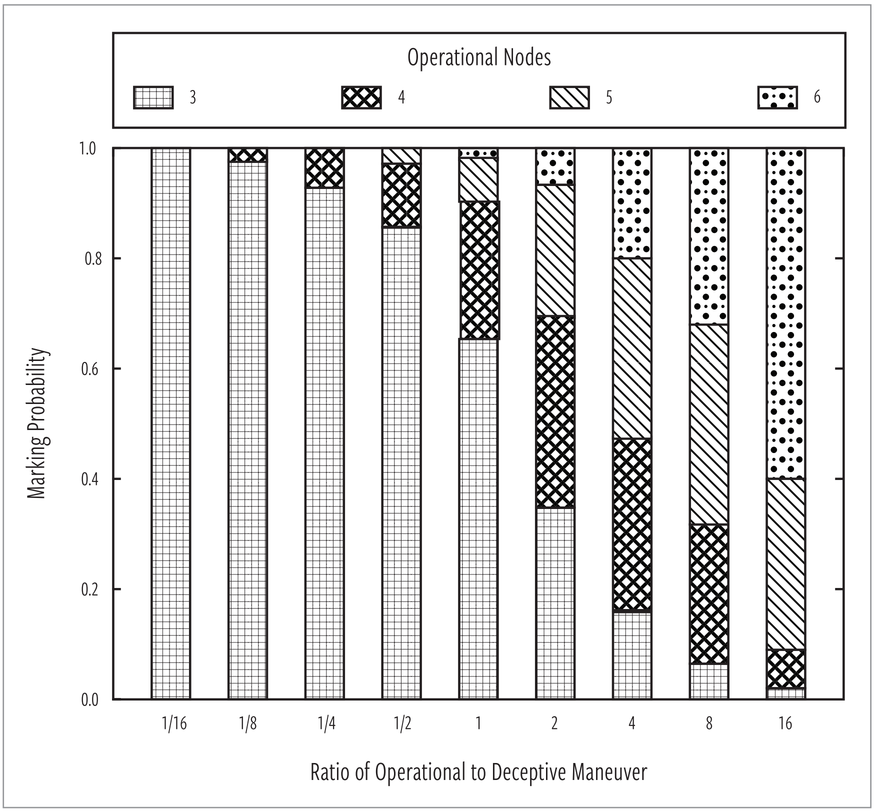

Figure 5. Marking probability of an 8-3-2 defensive maneuver cyber platform when varying the rates of operational and deceptive maneuver [8]

Our attacker model used to evaluate the DMCP is a single adversary who will target only operational or deceptive nodes. The attacker is unable to distinguish between operational and deceptive nodes and selects with equal probability a random active node to target. Once selected, the attacker targets and attempts to gain user or elevated access to the node using its various exploitation tools. After a random amount of time being targeted, the node is considered to be compromised.[16]

In order to increase the probability of survival, we desire to maximize the probability that a target node is deceptive and the probability that targeted operational nodes maneuver before the node can be fully compromised. We created a series of equations showing how each variable in the system can be adjusted to increase these probabilities. We present a series of rules of thumb for system administrators to employ to find the correct balance between security and operations of a cluster.

Finally, we built a prototype maneuver resource maneuver manager to validate the mathematical model. This system incorporates a discrete event simulator allowing each system variable to be adjusted to view the trade-off between operations and security. The system also provides visualization of the current system state, allowing an experimental run to be followed and tracked by researchers. The maneuver manager allows us to refine our rules of thumb for deployment of the system. These models provide the foundation for a system to be built that extends experimentation of the value of cyber maneuver to the security of computing systems.

5. CONCLUSION

We have presented our work in designing, building, and modeling maneuver applications to advance the state of the art in distributed and parallel computing. This work demonstrates how the military concept of maneuver can be applied to distributed computing in multiple facets.

In the area of resource provisioning, the Job Uninterrupted Maneuverable MapReduce Platform deploys a Hadoop cluster within an existing academic high performance-computing (HPC) environment. JUMMP supports high availability and continuous computing for research and education while incurring no additional financial or administrative overhead. It is shown to be as efficient as a persistent Hadoop cluster on dedicated computing resources, depending on the jump time. The cluster remains stable, with good performance, in the presence of jumps that occur as frequently as the average lengths of Reduce tasks.

In the area of application optimization, the Flow Optimized Route Configuration Engine provides the design and prototype development of a datacenter testbed with a reprogrammable network topology. The FORCE testbed includes a Virtual Topology Engine that builds virtual network topologies over physical links with SDN flows and a Flow Network Evaluation system to generate a network congestion estimation score. We highlight the design and portray the development of a Hadoop shuffle traffic simulator placing realistic loads on datacenter networks. Experimental results indicate placement of computation racks within a datacenter topology potentially has significant impact on the Hadoop shuffle traffic completion time.

Our research into modeling a Defensive Maneuver Cyber Platform with Stochastic Petri Nets demonstrates cybersecurity improvement through maneuver. This model introduces a distributed and parallel application utilizing moving target defense and deceptive defense tactics to increase survivability in the presence of a cyberattack. An SPN model is used to analyze the trade-offs between security and operations in the Defensive Maneuver Cyber Platform.

ACKNOWLEDGMENT

We would like to recognize the significant contributions of our co-authors on our original works. Jason Anderson, Edward Duffy, Hongxin Hu, Linh Ngo, and K.C. Wang have been extremely supportive and influential in our efforts. Additionally, we would like to acknowledge the hard work of the entire staff of Clemson Computing and Information Technology (CCIT) team in maintaining and administering the Palmetto Cluster and our extensive campus infrastructure. Our work would not be possible without their expertise and effort. This research is supported by part by US NSF MRI Grant #1228312 and US NSF Grant #1405767.

NOTES

1. Jeffrey Dean and Sanjay Ghemawat, “MapReduce: Simplified Data Processing on Large Clusters,” Commun. ACM 51, no. 1 (January 2008): 107–113. doi:10.1145/1327452.1327492.

2. Nicholas J. Dingle, William J. Knottenbelt, and Tamas Suto, “PIPE2: A Tool for the Performance Evaluation of Generalised Stochastic Petri Nets,” SIGMETRICS Perform. Eval. Rev. 36, no. 4 (March 2009): 34–39. doi:10.1145/1530873.1530881.

3. S. Liles, M. Rogers, J.E. Dietz, and D. Larson, “Applying Traditional Military Principles to Cyber Warfare,” In 2012 4th International Conference on Cyber Conflict (CYCON), 1–12, 2012.

4. G. Conti, J. Nelson, and D. Raymond, “Towards a Cyber Common Operating Picture,” In 2013 5th International Conference on Cyber Conflict (CyCon), 1–17, 2013.

5. Scott D. Applegate, “The Principle of Maneuver in Cyber Operations,” In Cyber Conflict (CYCON), 2012 4th International Conference on, 1–13, IEEE, 2012, http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6243974.

6. William Clay Moody, Jason Anderson, Kuang-Ching Wang, and Amy Apon “Reconfigurable Network Testbed for Evaluation of Datacenter Topologies,” In Proceedings of the Sixth International Workshop on Data Intensive Distributed Computing, 11–20. DIDC ’14. New York, NY, USA: ACM, 2014. doi:10.1145/2608020.2608023.

7. J. Dressler, C.L. Bowen, W. Moody, and J. Koepke, “Operational Data Classes for Establishing Situational Awareness in Cyberspace,” In Cyber Conflict (CyCon 2014), 2014 6th International Conference On, 175–186, 2014, doi:10.1109/CYCON.2014.6916402.

8. W.C. Moody, Hongxin Hu, and A. Apon, “Defensive Maneuver Cyber Platform Modeling with Stochastic Petri Nets,” In 2014 International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 531–538, 2014.

9. Guohui Wang, T.S. Eugene Ng, and Anees Shaikh, “Programming Your Network at Run-Time for Big Data Applications,” In Proceedings of the First Workshop on Hot Topics in Software Defined Networks, 103–108. HotSDN ’12, New York, NY, USA: ACM, 2012. doi:10.1145/2342441.2342462.

10. Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner, “OpenFlow: Enabling Innovation in Campus Networks,” SIGCOMM Comput. Commun. Rev. 38, no. 2 (March 2008): 69–74. doi:10.1145/1355734.1355746.

11. Faraz Ahmad, Seyong Lee, Mithuna Thottethodi, and T. N. Vijaykumar, “PUMA: Purdue MapReduce Benchmarks Suite,” ECE Technical Reports, October 2012, http://docs.lib.purdue.edu/ecetr/437.

12. William Clay Moody, Linh Bao Ngo, Edward Duffy, and Amy Apon, “JUMMP: Job Uninterrupted Maneuverable MapReduce Platform,” In 2013 IEEE International Conference on Cluster Computing (CLUSTER), 1–8, 2013. doi:10.1109/CLUSTER.2013.6702650.

13. Lance Spitzner, Honeypots: Tracking Hackers, Vol. 1. Addison-Wesley Reading, 2003.

14. D. Raymond, G. Conti, T. Cross, and M. Nowatkowski, “Key Terrain in Cyberspace: Seeking the High Ground,” In Cyber Conflict (CyCon), 2014 6th International Conference, 2014.

15. Department of Defense Dictionary of Military and Associated Terms, November 8, 2010 (last amended July 1, 2017), http://www.dtic.mil/doctrine/new_pubs/dictionary.pdf.

16. William Clay Moody, “Designing, Building, and Modeling Maneuverable Applications within Shared Computing Resources” (PhD diss., Clemson University, 2015).