Detect & Adapt: A Resiliency Enhancement Mechanism for Space Computing Platforms

Shafkat Islam*, **Nagender Aneja***, **Ruy de Oliveira**†,
Sandhya Aneja‡, **Bharat Bhargava*** **Jason Hamlet**§, **Chris Jenkins**§
*Purdue University, West Lafayette, IN, USA
† Federal Institute of Mato Grosso - IFMT, Brazil
‡Marist College, Poughkeepsie, NY, USA
§Sandia National Laboratory, USA
E-mail: {islam59, naneja, bbshail}@purdue.edu, ruyoliv@hotmail.com,
sandhya.aneja@marist.edu, jrhamle@sandia.gov, cdjenk@sandia.gov

***Abstract*—Over the years, space systems have evolved considerably to provide high-quality services for demanding applications such as navigation, communication, and weather forecast. Modern space systems rely on extremely fast commercially available off-the-shelf (COTS) processing units, with built-in GPU, DSP, and FPGA in light-weight, energy-efficient hardware. Since such devices are not necessarily designed with security features as a priority, there must be an adaptive controller to protect this mission-critical space system from potential malicious attacks, such as memory leaks, packet drops, algorithmic trojans, and so on. These attacks can lead the system to substantial inefficiency or complete failure. Considering the hardware diversity in current space systems, we propose a framework to explore both diversity and redundancy not only of hardware but also of software to make the overall system fault-tolerant. Our approach deploys mechanisms for monitoring and orchestrating actions of redundancy, diversity, and randomization to render the system resilient unpredictably dynamic, and optimize efficiency as much as possible during abnormalities. Yet, we use rule-based and adaptive engines to keep track of the various computing units to learn the best strategies to take when the system is under attack. The robustness of our approach lies in the fact that it makes the system highly unpredictable to potential attackers and tolerates attacks to some extent, which is crucial for any mission-critical application.**

***Index Terms*—Diversity, Dynamic Orchestration, Redundancy, Detection, Space System**

Introduction

Space systems provide essential civil, commercial, and military services. These areas heavily rely on navigation, communication, and remote sensing services. Technological advancements over the last few decades have resulted in the development of smaller, lighter, and more powerful devices for use in space. This transformation has led to using efficient computational hardware known as Heterogeneous Computing Platform (HCP) [1]–[3].

HCPs can integrate various processing units, including CPUs, GPUs, DSPs, and FPGAs, into a single chip, making them highly efficient in terms of both performance and power consumption. HCPs can render space systems relatively cost-effective when used with the appropriate software. However, modern space systems typically rely on commercial-off-the-shelf (COTS) platforms, which are heterogeneous and primarily used to reduce price and development time. The downside of this strategy is that COTS platforms are not necessarily designed with robust built-in security features, which can make the systems vulnerable to attacks.

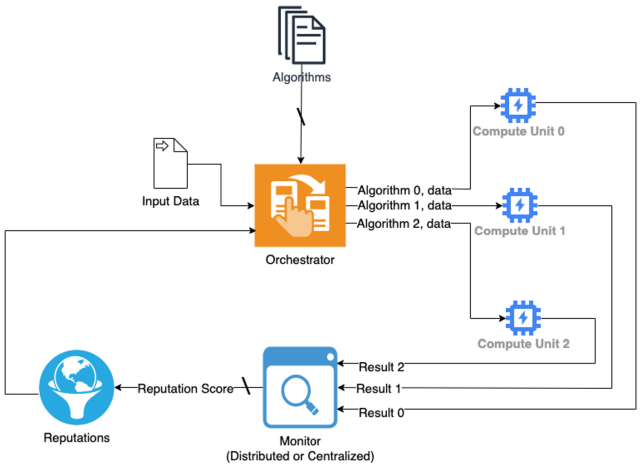

Fig. 1. Proposed HCP Orchestration & Monitoring Mechanism

Space has become increasingly crowded and competitive, which raises concerns about their security. While commercial space systems have grown exponentially in recent years, military expansion in space has complicated matters further. Moreover, a gap exists in the literature regarding investigating the security aspects of HCP platforms. As a result, there is a pressing need to explore ways to develop security solutions for the HCP-based computing environment [4]–[8].

Proposed HCP Orchestration & Monitoring Mechanism

The proposed HCP orchestration and monitoring mechanism consists of one computation Orchestrator ($\mathcal{O}$), one Monitoring unit ($\mathcal{M}$), and multiple Computation units ($\mathcal{C}_i$), as shown in Fig. 1. The $\mathcal{O}$ is responsible for directing the computation process of each compute unit and assigning an

algorithm to each unit from a set of algorithms $\mathcal{A} = \{a_i\}$. The $\mathcal{M}$ collects the computation response from each $\mathcal{C}_i$, aggregates it, and provides the final computation response. Additionally, the $\mathcal{M}$ assesses the computation performance of each compute unit and maintains reputation metrics of each $\mathcal{C}_i$. The reputation is also shared with the $\mathcal{O}$. Each $\mathcal{C}_i$ is responsible for performing the assigned tasks using the assigned algorithm, which is determined by the $\mathcal{O}$. It is important to note that the architecture assumes that the $\mathcal{O}$ and the $\mathcal{M}$ can communicate with each $\mathcal{C}_i$ through a communication channel. The $\mathcal{O}$ and the $\mathcal{M}$ can communicate between themselves as well. However, there does not exist any communication channel between any two compute units.

Threat Model

We assume that the $\mathcal{O}$ and the $\mathcal{M}$ are trustworthy components while an adversary can compromise one or multiple $\mathcal{C}_i$. This paper considers two classes of attacks: (i) *result manipulation attack:* attacks that target the output results, (ii)*performance degradation attack:* attacks that impact the performance of the compute units but not necessarily alter the output. Possible attack surfaces include data manipulation, output alteration, algorithmic trojan, packet drop, and memory leakage. We assume that the adversary can access the source code for conducting these sophisticated attacks. However, we assume that the adversary cannot compromise the communication channel between the $\mathcal{O}$ and $\mathcal{M}$. The adversary can introduce multiple attacks simultaneously. The description of the attacks is given below.

Result Manipulation Attacks:

Below is the description of different types of result manipulation attacks.

1) Data Manipulation: In this type of attack, the adversary arbitrarily manipulates the input data, and to make it realistic, the adversary can select the manipulated data from the original distribution. For example, if the input array is: $[5, 3, 10, 1, 6]$ after data manipulation attack it turns into $[5, 3, 10, 1, 5]$ where 6 is replaced by 5.

2) Output Alteration: In this attack, the adversary manipulates the computation output. For example, if the sorted result of a sorting algorithm is $[1, 3, 5, 6, 10]$, the adversary can modify it as $[1, 3, 5, 10, 6]$.

3) Packet Drop:: The adversary drops data packets from the communication channel between the $\mathcal{C}_i$ and the $\mathcal{M}$. An adversary can modify the communication between one or multiple compute units to the monitor.

Performance Degradation Attacks

The following are two types of performance degradation attacks:

4) Algorithmic Trojan: This type of attack involves the adversary manipulating the algorithm, which may or may not affect the output. The motivation of this attack can be one or both of the following: one is to pose an impact on the output, and the other one is to drain out the resources, i.e., energy and/or compute hour of the $\mathcal{C}_i$. For example, pivot tweaking (picking the max or min value at each iteration) in quick sort does not impact the sorting result. Still, it can successfully degrade the computation performance by ensuring the worst-case behavior of the quick sort algorithm.

5) Memory Leakage: In this attack, the attacker injects arbitrary memory-consuming instructions in the source code, which will lead to draining unnecessary memory of the compute units. In the worst case, it can lead to buffer overflow. This attack does not affect the output result but drastically reduces the compute capability of $\mathcal{C}_i$.

HCP Resiliency

HCP resiliency is defined as the platform's capability to execute its computing task accurately, even in the presence of adversarial events or abnormalities. Each HCP platform has a different level of tolerance to adversarial events based on its attributes. An HCP platform's resiliency depends on three key features: the ability to identify abnormal events, respond to them, and recover from them. A platform is considered more resilient than others if it can ensure these three features efficiently and promptly without compromising the ongoing computing accuracy.

Computation Strategy

This section outlines different computation strategies that can be adopted by the $\mathcal{O}$ to enhance the resiliency of HCP, depending on the computation application.

Computation Partition (CP): This strategy involves partitioning a computation task into multiple sub-tasks, with each $\mathcal{C}_\rangle$ responsible for performing one or more of the sub-tasks. The $\mathcal{O}$ assigns the sub-tasks to each unit $\mathcal{C}_i$ and determines the algorithm or sub-task execution process. Finally, the $\mathcal{M}$ accumulates the results of each sub-task from each $\mathcal{C}_i$, combines them, and outputs the final result.

Triple Modular Redundancy (TMR): In this strategy, three compute units perform an identical task/sub-task, using the same algorithm or execution process.

Triple Modular Diversity (TMD): This approach involves at least three compute units performing identical tasks/sub-tasks but with different task execution processes/algorithms. The $\mathcal{O}$ determines the process/algorithm for each unit.

Hybrid Strategy: In this strategy, both the algorithm and compute unit can change over time. It can be performed consistently with CP and/or TMR/TMD.

Hierarchical Detection Mechanisms

This section provides an overview of various attack detection strategies along with their merits and demerits. The monitoring system can use one or more of these strategies to detect attacks in C_i, depending on the application's requirements.

Brute Force Approach: In this approach, the $\mathcal{M}$ checks each smallest segment of the output result. The definition of the *smallest segment* depends on the characteristics of the application. For example, in sorting, the *smallest segment* is each pair of two consecutive elements. However, this method may not

work for applications where there is no single absolute correct result, such as a compression algorithm. Moreover, checking every output segment may consume lots of computational power.

Probabilistic Approach: In the probabilistic approach, the $\mathcal{M}$ obtains some predefined specific portions of the correct computation results beforehand. It matches them with the computed ones from the $\mathcal{C}_i$. The advantage of this approach is that it reduces the burden on the monitor, while the disadvantage is that it trades off the detection guarantee.

Fingerprinting: In the Fingerprinting approach, the $\mathcal{M}$ accumulates execution statistics, such as memory usage, computation time consumed, etc., from each $\mathcal{C}_i$ and maps those statistics with standard or acceptable ones to detect any abnormality. The monitor can use a machine learning model or other statistical method for detection. However, accumulating those statistics accurately is challenging.

Hashing Based Approach: In this approach, the $\mathcal{M}$ obtains the hash value of a portion of correct output from the $\mathcal{O}$ and matches it with the result from $\mathcal{C}_i$. This approach consumes fewer resources during detection and is also accurate. However, getting the hash value of outputs for all applications may not be possible.

Attribute-based Checksum Approach: In this approach, the $\mathcal{M}$ collects some attributes of the input data/output results (not actual result) from the $\mathcal{O}$ and matches those attributes with the output from $\mathcal{C}_i$. The *attribute* differs in different applications. This method also trades between attack detection possibility and detection resource consumption. For example, in a sorting application, the $\mathcal{M}$ can acquire the sum of all elements from the $\mathcal{O}$ and match that sum to the output of the $\mathcal{C}_i$.

Adaptive Orchestration Mechanisms

The use of adaptive orchestration mechanisms has the potential to provide resiliency, and defend against cyber-attacks. There are two primary types of adaptive orchestration mechanisms: rule-based adaptation and machine learning-based adaptation.

Rule-based Adaptation: This strategy is based on predefined rules that dictate how the orchestration process and components should be changed based on performance data provided by the monitoring system. While this approach can respond to attacks, it is static, which means that an adversary can predict the response if they observe the platform's behavior for a sufficient time, allowing them to execute advanced cyber-attacks.

Machine Learning-based Adaptation: This strategy uses data-driven models to create an adaptive orchestration strategy. This approach collects training data from the platform and uses supervised, such as deep neural networks, or adaptive machine-learning models, such as reinforcement learning, to create data-driven models. With the requirements and criticality of the application, the orchestration system can adopt a continual learning framework. This type of orchestration is adaptive to the changing behavior of the adversary and can change the orchestration strategy dynamically with appropriate modifications. However, to make the orchestration unpredictably dynamic the objective function of machine learning model should contain a randomization term.

Baseline Orchestration Mechanisms

The baseline orchestration mechanisms, such as random orchestration and round-robin orchestration, are unresponsive to any attack or abnormality in the platform.

Random Orchestration: In this mechanism, the orchestration system randomly assigns the execution algorithm and component to $\mathcal{C}_i$ during each execution cycle. Thus, the $\mathcal{O}$ is unresponsive to any attack or abnormality in the platform.

Round-robin Orchestration: In this mechanism, the orchestration system assigns the execution algorithm and component to $\mathcal{C}_i$ in a round-robin manner during each execution cycle. Therefore, this method is also unresponsive to any adversary.

Conclusion

The use of heterogeneous computing platforms (HCP) has expanded beyond space computing and is now widely used in various applications such as critical infrastructure, edge computing, and AI. This paper proposes a framework for monitoring and orchestrating HCP using detection and adaptation-based strategies. These strategies can be modified to suit different applications such as sorting, compression, gradient descent calculation, and machine learning algorithms. Overall, this paper presents a versatile approach to manage HCP for a range of applications.

Acknowledgement

This research is supported, in part, by the Sandia National Laboratory (SNL). The sand number is: SAND2023-11822A. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of SNL or the U.S. Government.

References

[1] R. Jassemi-Zargani, S. Bourdon, and V. Fong, "A hierarchical evaluation of space-based systems performance," in *2011 2nd International Conference on Space Technology*, pp. 1–4, 2011.

[2] S. Prongnuch and T. Wiangtong, "Heterogeneous computing platform for data processing," in *2016 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS)*, pp. 1–4, 2016.

[3] A. Shaked, L. Tabansky, and Y. Reich, "Incorporating systems thinking into a cyber resilience maturity model," *IEEE Engineering Management Review*, vol. 49, no. 2, pp. 110–115, 2021.

[4] C. G. Starling, M. J. Massa, C. P. Mulder, J. T. Siegel, J. E. Cartwright, D. L. James, R. Piliero, B. M. Williamson, D. W. Brown, R. Lott, C. J. MacArthur, A. P. Hays, C. Trotti, and O. Popp, "The future of security in space: A thirty-year us strategy," tech. rep., Atlantic Council, 2021.

[5] C. J. Castelli, "Closer commercial ties urged: Dod sees space as increasingly congested, contested, competitive," *Inside the Air Force*, vol. 21, no. 11, pp. 10–12, 2010.

[6] L. Vessels, K. Heffner, and D. Johnson, "Cybersecurity risk assessment for space systems," in *2019 IEEE Space Computing Conference (SCC)*, pp. 11–19, 2019.

[7] M. Manulis, C. P. Bridges, R. Harrison, V. Sekar, and A. Davis, "Cyber security in new space," *International Journal of Information Security*, vol. 20, pp. 287–311, Jun 2021.

[8] M. Zhuo, L. Liu, S. Zhou, and Z. Tian, "Survey on security issues of routing and anomaly detection for space information networks," *Scientific Reports*, vol. 11, Nov. 2021.