A Decision Framework for Allocation of Constellation-Scale Mission Compute Functionality to Ground and Edge Computing

Jaime Ramirez, Caleb Royer, Brian Free, Owen Brown, Robert Polutchko
Scientific Systems Company Inc.
500 W Cummins Pkwy, Woburn, MA; 978-500-9553
jaime.ramirez@ssci.com

ABSTRACT

This paper explores constellation-scale architectural trades, highlights dominant factors, and presents a decision framework for migrating or sharing mission compute functionality between ground and space segments. Over recent decades, sophisticated logic has been developed for scheduling and tasking of space assets, as well as processing and exploitation of satellite data, and this software has been traditionally hosted in ground computing. Current efforts exist to migrate this software to ground cloud-based services. The option and motivation to host some of this logic "at the edge" within the space segment has arisen as space assets are proliferated, are interlinked via transport networks, and are networked with multi-domain assets. Examples include edge-based Battle Management, Command, Control, and Communications (BMC3) being developed by the Space Development Agency and future onboard computing for commercial constellations.

Edge computing pushes workload, computation, and storage closer to data sources and onto devices at the edge of the network. Potential benefits of edge computing include increased speed of response, system reliability, robustness to disrupted networks, and data security. Yet, space-based edge nodes have disadvantages including power and mass limitations, constant physical motion, difficulty of physical access, and potential vulnerability to attacks.

This paper presents a structured decision framework with justifying rationale to provide insights and begin to address a key question of what mission compute functionality should be allocated to the space-based "edge", and under what mission or architectural conditions, versus to conventional ground-based systems. The challenge is to identify the Pareto-dominant trades and impacts to mission success. This framework will not exhaustively address all missions, architectures, and CONOPs, however it is intended to provide generalized guidelines and heuristics to support architectural decision-making. Via effects-based simulation and analysis, a set of hypotheses about ground- and edge-based architectures are evaluated and summarized along with prior research. Results for a set of key metrics and decision drivers show that edge computing for specific functionality is quantitatively valuable, especially for interoperable, multi-domain, collaborative assets.

INTRODUCTION

In this paper we discuss the problem of identifying the value of running software at the edge as opposed to on the ground, in the case of satellite systems and in particular proliferated LEO (P-LEO) architectures. We attempt to provide an ontology of the problem and a framework to analyse the value of migrating software and executing processing at the space edge.

This problem has parallels to other distributed and fog computing problems, yet it goes beyond the resource allocation process and provides unique challenges given certain specific characteristics of the space operations and its applications.

In particular, the path latency derived from the large distances involved and the effects of the type of data workflow (which usually requires reliability and tight timelines) is a driver of differentiation to analysis for ground-based solutions. The decision of where certain processing is running is not only defined by optimizing resource usage but should include other factors like security and reliability. Lastly, the inability of user input/output at the data source is a unique element to this problem. This last one characteristic is a main driver of the desire to understand this problem, since there seems to be significantly value in providing command and control autonomy to the system for making decisions about the data, but that involves that specific processing should be occurring driven itself by the data.

We examine recent literature that addresses similar problems in the contexts of asymmetric distributed computing platforms (e.g. fog computing), drawing parallels and identifying some approaches to define the best use of resources, simulating scenarios of interest and

identifying key drivers that support the decision process in architecting software for distributed proliferated space systems

Background

Overall, the problem of distributing computation among processors connected by a network is not new and recently with the advent of the cloud and fog computation and the realizations of the Internet of Things (IoT) it has received more specific attention to provide practical solutions in the case of unreliable or varying network capability, varying demand, and computationally capable edge nodes.

This concept in particular, becomes important with the advent of systems deploying flexible capability at the edge in space via "software defined satellites", where applications developed and deployed at the edge can provide new capabilities to an already deployed space asset. Systems like the USSF Space Development Agency (SDA) Transport Layer, with a flexible computational infrastructure using their BMC3 module, enables some of this flexibility. Design processes to understand when it is valuable to deploy processing capability to orbit and make best use of the overall integrated system is an ingoing question to be resolved.

Of relevance, the concept of fog processing has been recently studied as a mechanism to provide intermediate computation capability between low computationally capable nodes and highly available and capable cloud computing in order to address latency and network availability limitations. Bonomi et al. [5] introduced the concept of fog computing as a paradigm that brings computation and storage capabilities closer to the network edge. The highlighted benefits of fog computing include reduced latency, improved scalability, and enhanced privacy and security.

In terms of the applicability of edge processing for space applications, Bhattacherjee [18] explored the concept of in-orbit computing as a thought experiment delving into the possibility of deploying computing resources directly in space, specifically in orbit around the Earth, and discuss the potential benefits and challenges associated with such a paradigm. In Furano et al.[7] the authors explore the potential of utilizing artificial intelligence (AI) on the edge in space systems. The authors discuss the challenges and opportunities associated with integrating AI techniques, focusing on the benefits of processing data locally at the edge rather than relying solely on centralized systems. They highlight the advantages of edge AI in terms of reduced communication latency, enhanced autonomy, and improved resilience. The paper offers valuable insights into the challenges and opportunities but do not provide any specific framework, as of how and when porting processing to orbit makes sense.

The problem of distributing computation functions across computational nodes has been observed from different perspectives. In some work, the focus is on identifying the architecture mechanisms to best move the processes around to most efficiently utilize the resources [2,3,8], in other work it has been focused on solving the constrained optimization problem of computation and bandwidth resources as an optimization problem [1, 11,6,13]. Most of these approaches provide formulations to solve as a function of quantifiable metrics like latency and power utilization, but become less applicable to addressing the bigger problem, where, as we claim in this paper, other metrics, some of them specific to the problem and some of them not quantifiable, drive the decision process.

iFogSim is a tool proposed by Gupta, Harshit, et al [4] to simulate and trade resource management techniques. It provides a pragmatic and useful solution to studying the Fog computation problem and a step in the right direction of providing a framework for analysis, however like other tools, it seems mostly focused on static network environment and quantifiable/modelable metrics of performance.

Most of the above work, although partially applicable, has not focused on the aspects relevant to satellite systems or military application in hostile environments. Such aspects specifically affect the analysis when metrics like reliability, security, accuracy and other metrics of military utility determine the ultimate value of the system.

App deployment in hostile environment is explicitly analyzed in the work by Satyanarayanan [12]. They propose the use of cloudlets, (small-scale cloud data centers deployed in close proximity to mobile devices) to address the challenges faced by mobile devices in hostile or resource-constrained environments, such as limited connectivity, high latency, and unreliable network conditions, providing some analysis of key considerations for resource allocation, load balancing, and security.

Other work, has more specifically considered the computation allocation with applications to the space edge computation problem.

Pfandzelter and Bermbach [14] considered the problem of quality-of-service (QoS)-aware resource placement for low Earth orbit (LEO) satellite edge computing. The authors propose a framework that aims to optimize the placement of computing resources in LEO satellite networks to ensure efficient and reliable service delivery

while considering QoS requirements, formulating it as a mixed-integer linear programming (MILP) resource placement problem while considering various factors such as communication latency, processing capabilities, and resource availability.

Dong et al. [15], introduce a computation offloading strategy for a low Earth orbit (LEO) constellation edge cloud network. The authors propose an algorithm that optimizes the computation offloading decisions to minimize energy consumption and latency while ensuring efficient resource utilization. They consider the unique characteristics of LEO satellite networks, including limited bandwidth, communication delay, and resource constraints, in formulating the offloading strategy. Other work by Lai et al., addressed the construction of cost-effective content distribution networks (CDNs) using emerging low Earth orbit (LEO) satellites and clouds. The authors propose a cooperative approach that leverages both LEO satellites and cloud resources to distribute content efficiently and minimize the operational cost of CDNs.

Qiu et al.[19] discuss the architectures, key technologies, and challenges associated with deploying Mobile Edge Computing (MEC) in Space-Air-Ground (SAG) integrated networks. They highlight the importance of MEC in enabling real-time data processing, low-latency communications, and efficient resource utilization in dynamic and distributed SAG environments. The paper presents various MEC architectures tailored for SAG networks, considering different deployment scenarios and network topologies. The authors also discuss key technologies and components such as edge servers, communication protocols, resource management, and security considerations specific to SAG environments. Additionally, the paper addresses the challenges and open research issues in integrating MEC into SAG networks, including resource allocation, mobility management, network scalability, and QoS provisioning.

All this work substantiates the need for an analysis framework to define where computation shall happen, yet they are focused on particular applications and focus on defining metrics of performance solely focused on latency and resource utilization, missing the fact that other, more complex factors, may actually drive the applicability of performing some of the edge processing on board.

To the authors' knowledge, no other work has attempted a holistic look at understanding when certain computations processes should be performed in orbit, considering the fact that there are different reasons and different applications of why to do that.

Approach

At a high-level, we propose to analyze the problem of when an application provides advantages by running at the space edge. To do that, it is necessary to perform an identification of the value of a specific application and a functional breakdown to identify how the distribution takes place, what is the functionality that can be deployed and what are the attributes that need to be resolved to provide a solution. The overall problem addressed in this paper is beyond the computation resource allocation problem.

We start by providing an ontology of the software distribution problem, then defining a process to validate the value of software migration to the edge and lastly, based on that, we simulate some scenarios to understand how specific functional allocation trades.

ONTOLOGY OF SOFTWARE IN SPACE EDGE APPLICATIONS

In this section we provide a simple ontological description of the mission execution problem, by using the concept of mission pipeline graphs. This approach is related to an MBSE-style functional allocation, and we provide specific structure and define some artifacts that are useful in the specific edge application analysis.

Our first definition statement is that a mission is performed by a composition of functions. Functions are logical units of work to transform input to output by interacting with the world and functions can be iteratively decomposed. Functions may have interdependencies, and may exchange, share, and compete for resources. The composition of functions can be defined as a graph that defines the connection of function outputs to function input. Functions, as elements of a process chain can be parallelizable or serialized.

The composition of functions to address a mission is defined as a *mission pipeline graph*.

Software implements functions that execute as computation processes by interacting with computation processors via the instruction set, other functions can interact with other hardware elements (e.g. hardware actuators, antennas).

A second statement is that the output of executing a computation process can be qualified in three competing dimensions: *time, accuracy and resource usage* (program managers might be familiar with this trifecta).

- Time, refers to as how fast the output is generated with respect to the input. It may have some qualifiers, including availability (for how long that

timeline is achieved), periodicity (how repeatable is that timeline) and reliability (will it ever happen)
- Accuracy, refers to how close the outcome is to the intended output. It also has qualifiers: how robust it is (accuracy with respect to the set of inputs), how resilient it is (accuracy despite intentionally bad inputs)
- Lastly, resources are limited and depending how the function is implemented and the operations it needs to perform, it may require more or less resources.

Software, as a specific implementation of functions, has a realized profile that can help define its attributes, for example:

- how much data per unit of time it received as input it can process,
- how much output data per unit of time it produces,
- how much time it takes to run on a given processor,
- what is its expected accuracy
- How available it is, e.g. in duty cycle
- How robust it is to certain events, e.g. lack of input data, wrong input data, adversarial input data.
- Etc.

We propose a breakdown of the software (running either at the edge or not) by the functionality that it provides:

1. **System Software (Infrastructure)**: Software that implement functions that provide infrastructure to run other processes and interface with sensors and effectors. Includes in-node security and manipulation of infrastructure data. Implies any calculations and processing necessary to maintain the spacecraft operational and includes execution of payload commands (for example, executing a collection).
2. **Application software**: Software that leverages the system software functionality to execute a mission, it includes:
    a. Data processing Software: Implement functionality that executes manipulation of the mission data. This means processing and exploitation algorithms to extract or convert the data. Examples of this functionality include:
        i. Signal generation
        ii. Single and multichannel signal processing
        iii. Radar Processing:
            1. Image formation
            2. Ranging
            3. MTI
        iv. Image and Video processing:
            1. Geo-registration
            2. Channel adjustment
        v. Target/Pattern Recognition
        vi. Fusion
    b. Command and control Software: Implement functionality to perform decision making about how to generate, use and share mission data and decision making on how to achieve the mission. Examples includes:
        i. Creating and maintaining a Situational Awareness picture,
        ii. Communication and Collaboration decisions,
        iii. Resource Management,
        iv. Tasking and Mission Planning,
        v. Monitoring and Security and Access Control
    c. Testing and debugging Software: provide functionality that can support analyzing data and identifying issues either at the system software or at the application software level.

It is important to note that this includes software running on components that provide functionality, like for example payloads or integrated communication systems, since in theory for some cases, depending on specifics, some of that functionality could run on a different location.

SPACE EDGE PROCESSING

Many new space missions propose utilization of in-space processing resources to execute software components at the edge, there exists therefore a question of what processes and software shall exist and operate at the space edge.

The overall concept of utilizing a connected network of space assets is that, a networked operation can enable distributed sensors and effectors, connected via a network, to perform an Observe, Orient, Decide and Act (OODA) loop achieving better mission performance than a single node would. A space processor is thus, just an element of a larger interconnected network of processing capability, sensors and/or effectors, with varying levels of connectivity. The objective of performing any computation is to obtain an output, which can be observed as information to feed other processes along a pipeline of data to achieve the OODA loop. The attributes of each processing outcome can be qualified in 3 dimensions: time, accuracy and resource utilization.

Space-borne nodes, have particular advantages and disadvantages with respect to other network nodes operating in a different domain, being air, maritime or ground infrastructure. Their most important advantage is that their payloads (primarily sensors and communication links) have a larger field of regard over and beyond the earth surface (including restricted aerospace) per cost of deployment. The second one, is

that given the distances involved, space assets are harder to detect and physically access. This implies lower vulnerability in some adversarial scenarios. The third characteristics is that they can get nearby or physically access other objects in space.

Those characteristics can also pose disadvantages. Since these nodes are difficult to physically access, extracting data or repairing in case of failure is difficult. They are intrinsically uncrewed, and human intervention to directly manipulate or use the data at location is not available. Lastly, they are mostly predictable, and can be vulnerable, since space is a less controlled environment, it can be difficult to observe and maintain awareness, and there are less tools to defend against specific adversarial attacks or environmental threats.

In addition, space nodes have two clear (but not necessarily unique) constraints that must be considered:

1. They are resource limited (power and mass), which imposes constraints on the amount of onboard computation, the amount of transmission bandwidth, and/or a duty cycle of operation. (Driven by the available local power supply and storage and by the thermal dissipation constraints).

2. They are constantly moving with respect to the ground (except for the especial case of GEOs) and there is not a lot of flexibility on how much their trajectory can be altered.

These characteristics and constraints must be the drivers that influence the decision if an application should be run on a space node or on a terrestrial node.

Given these factors and constraints, we propose to identify a methodology to assess the effects of executing software functionality in space. First by exploring a (not exhaustive) taxonomy of applications and software functionality, then, describing a step-wise decision framework and lastly highlighting some examples.

TAXONOMY OF USE CASES FOR SPACE EDGE PROCESSSING

The Table 1 below, provides a list of example space mission use cases. Users of the methodology are encouraged to use it as a reference to identify functionality migration opportunities to be investigated and in support of defining mission pipeline graphs.

Table 1: Taxonomy of edge-software use cases

Mission Type	Mission Objective	Mission Key Metrics
Remote Sensing	Image collection	Volume of data Revisit Time Resolution Speed of response for specific resolution
1st derivative analysis	Identification of objects	Number of detected objects Sensitivity, specificity metrics
	Geolocation and tracking	Number of tracks created Track accuracy Track deconfliction metrics
2nd derivatives analysis	Activity based intelligence: -Forecasting, -Pattern of life	Prediction accuracy, sensitivity, specificity metrics
	Target processing: - Target custody - Target statistics	Total track number over time Track accuracy over time
	Geospatial Information: Structure Weather Mapping	Geographical extent Accuracy
PNT	Signal broadcast Solution delivery	Solution accuracy Solution latency
Communications	Bulk Data Small point to point Area Distribution	Bandwidth Latency Service area capacity (BW per Area size)
System Management	Planning (scheduling, tasking) Infrastructure management Network control Security	Management scope Management capacity (number of resources served)

PROPOSED ANALYSIS AND DECISION FRAMEWORK

The proposed decision framework attempts to be un-opinionated, generally useful, systems-thinking-based approach with grounding in necessary decision data. The framework includes the fact that migration of functions to the edge requires both qualitative and quantitative analysis, and requires a wholistic view and not isolated numerical calculations, simulations, or single-factor thinking. This framework is not a turn-the-crank, prescriptive recipe – it requires knowledgeable individuals to perform trades and create options within the framework. This framework is provided as a methodology of how to think about the problem, with the intent to help the reader decide what functionality may be better placed at the edge or on the ground.

The overall proposed framework flow is visually shown in Figure 2. For the math-minded reader, "Independent Variables" can be considered to be the mission objectives (as defined by selected Metrics of Value), the function and interrelation (described by the mission pipeline graph), and the deployment of those functions across processing capability in the network nodes. "Dependent Variables" will be the Metrics of Value, which are driven by models and trades, and themselves guide the final decision outcome which is a selected architecture.

In particular, the framework considers 4 steps:

1. Generation of a mission pipeline graph

The first step in assessing (an) application(s) suitability for edge or ground-based hosting is to define the application (mission) functional workflow as a graph, that is, identifying how inputs relate to outputs from other functional elements, dependencies and resources, and how the function output feeds into other functional elements. The goal is to break down the functionality into computational units that could be placed at specific nodes in the network (see [11]), and enable trade analysis.

The key process in defining if functionality shall be allocated to a specific node (space or ground) is to evaluate (qualitatively and quantitatively) how the execution of an application or process in a particular node affects the performance drivers, while considering the constraints, the advantages and the disadvantages and to quantify how each of these three drivers affect the end metric of performance of the mission.

To perform this analysis process there are few steps that need to be performed:

1. Identify the application definition
2. Identify how it can be "connected"

That is, how a particular application fits within a data workflow, how its required input can be provided by another process in the system and how its output are useful to another process in the system.

This requires, first identifying the mission or missions it can support as use cases and then identifying where in those missions.

To do so, several dimensions are to be considered.

1. Generic use case addressed: identifying how the utilization of a functional software element fits within a given mission profile. In section 4 we provide a taxonomy of use cases that can be useful in identifying the functional workflow.
2. Application functional type: Following the descriptions in section 2, the user can support the description of the interconnected graph
3. Prerequisites: Clear identification of what are the specifics of the input.
4. Identification of System Assumptions:
   Qualitative (soft) assessment of the ability of the application to be ported within an operational framework, i.e., does it provide the functionality required for the operational use cases?, does its input/output profile fits within an operational need?

The outcome of this first step should be the definition of a functional graph that for the functional modules that are being traded, able to capture the interfaces between processes.

Examples

Figure 1 - Figure 4 graphically describe some examples for some missions of interest. 1: ISR mission, 2: SSA mission, 3: ISAM mission, 4: PNT mission and 5: Communications mission. They provide the description for notional missions describing the overall concept of mission pipeline graphs. The level of fidelity in the steps and the scope of the overall pipeline has to be defined driven by the software function that is being analyzed.

For example, in the ISR mission the objective is to identify targets on the earth surface by using an EO/IR sensor and we are identifying if elements of the In order to do that we identify the functional composition of the mission going into some detail to some of the predictive modeling software is useful to be run onboard. Possible functional decomposition, the level of detail granularity shall be determined by the specific analysis, in order to make sure the modeling described in section 2, supports the analysis of trading the functionality to the edge.

The overall sequence for this mission is assumed to include:

1. Area selections: namely, some process with or without human interaction must define where the system is looking for targets.
2. Collection planning: based on a definition of locations, the system shall define how it is collecting data at those locations. In order to make that decision it uses to functions, one is modeling the performance metrics that drive the selection, the other one is the combinatorial exploration and selection form the action space. The performance function include sun artifact prediction, cloud coverage and GSD prediction.
3. Execution of the collection (which required input from the process maintaining state of the system)

4. Image processing: includes georectification
5. Target recognition

Figure 1. Models the described scenario in example 1.

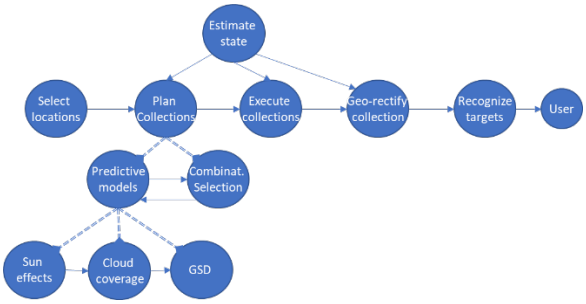

Figure 1. Mission Pipeline Graph Example for an target recognition ISR mission

In the SSA mission, there are cycles, driven by the fact that the plan of collections for tracking, affects the overall plan for collections.

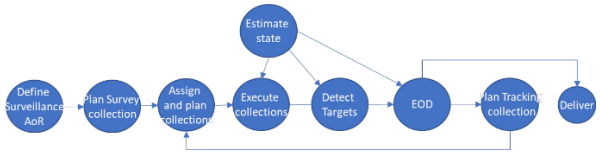

Figure 2. Example of a mission pipeline graph for an SSA mission

In the ISAM mission example, the calculation of a sequence of activities is driven both, by the function that define the next objective and the analysis of the relative pose of the system.

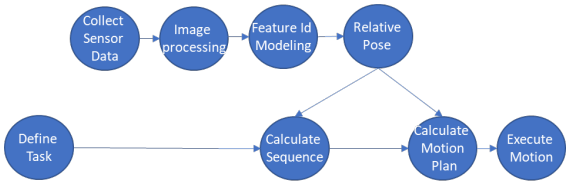

Figure 3. Example of a mission pipeline graph for an ISAM mission

In the case of the PNT mission graph is very simple, we assume there is a constant function of maintaining ephemeris and clock drift, which affects the generation of the PNT signal to be transmitted.

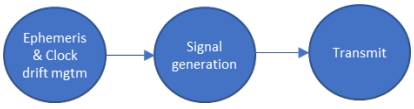

Figure 4. Example of a mission pipeline graph for a PNT mission

Lastly, a mission pipeline graph example of a satellite communication mission that receives data from a user to be sent to another location, requires collecting the data from the user, scheduling the transmission to the destination, modulating the signal as needed, and finally transmitting.

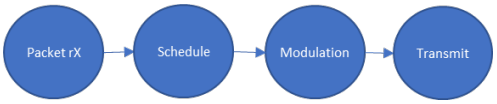

Figure 5. Example of a Rx-Tx Satellite communications mission

2. Identify and Model Key Metrics of Value

The second step in the process is the definition of Metrics-of-Value and an approach to measuring these figures.

A mission pipeline graph, as a composition of functions where each function has time, accuracy and resource usage attributes (as defined in section 2). A way to think about the composition of the Metrics of Value is by looking at thse categories of time, accuracy and resource usage.

The overall composed mission timeline can be derived from the composition of mission pipeline graph functions timelines. Similarly, the overall composed use of resources, can be derived from the composition of mission pipeline graph functions resource usage and can be ultimately compiled as a mission cost metric, given that to provide the resources there will be a monetary cost.

Lastly, the overall composed mission accuracy is the Mission Performance Metric (i.e. its outcome is as close to as intended). Mission performance has different dimensions, since the ultimate mission utility can have quantifiable tangible and intangible attributes of success.

The Metrics-of-Value need constraints or expectations, without it the system design can be non-sensical and can include metrics such as Intangibles and Goodwill, or Quality metrics.

Both qualitative and quantitative metrics might necessary for a complete evaluation, and for the purposes of the framework, quality attributes and binary/categorical factors are also termed “metrics”, as it is assumed they can be verified in some way (“measured”). If it is not possible to verify a requirement or desire, it cannot be included in the assessment.

There is no magic formula to compose mission performance from individual function composition, since the effects of the inputs to a function and the way resources are used can have different very different effects to the output, based on the solution

implementation. However, we believe the above framework provides a way to think about how to define Metrics of Value.

Methods to measure the metrics may include usage of checklists (for categorical or scorable factors), publicly available "comparables" data, parametric models, physics-based models, and high or low-fidelity simulations that allow assessing the resulting metrics in a repeatable (though not necessarily deterministic) manner. Moreover, the Metrics of Value can be analyzed for subsets of the mission without having to define the ultimate end-to-end Metrics.

Overall, most metrics fall within a few categories and the specific mission must tailor these to the desired [sub]set. The Metric of Value categories and notional examples are provided in Table 1.

Table 2: Metrics of Value and Examples

Metric Category	Example
TIME • Latency • Availability • Periodicity	("1000ms maximum delay from photons on sensor to processed output") ("50% Orbit-Average duty cycle for payload operations") ("1 pps +/- 0.0001%")
COST • Components • Resource Usage $ **Or as derived from resource needs:** • Quantity • Type • Usage Model	("Total cost $20M per satellite") ("Maximum cost of $100,000 per week for provisioned downlink") ("Min. 3 simulataneuos observation points [satellites]") ("Min. 3 sats with onboard GPUs") ("Requires consuming 10m/s of delta-V per week of the host")
PERFORMANCE **Measurable quantities** • Probability of Success • Numeric Performance **Intangibles and goodwill success:** • User Preference • Feedback/Visibility • Confidence/Trust • Ease of Use • Environmental, Social, Governance (ESG) **Categorical Quality** • Security and Classification Constraints • Compliance Open Standards	("Mean-Time-Between-Failures of 1000 hours") ("1 meter 3-sigma error of the position solution") ("Users like the intuitiveness of the User Interface") ("Users ignored reported warnings after being overloaded with warning signals") ("The use case is more environmentally responsible") ("System is certifiable by DISA to DoD Impact Level 6") ("System uses CCSDS standard for all uplink/downlink transmissions")

In some cases, it may be possible to reduce the Metrics of Value to a subset, and reducing to purely quantitative metrics may be useful to build "lumped parameter" models, such as adding up all of the delays in a system to trade competing architectures (at least along a 'Time' metric, though potentially changing Cost or other metrics).

In the metric selection process, it is important to identify metrics that can be calculated from a modeling and simulation activity, as these will potentially require development of such models and/or simulations. If required, the modeling & simulation (M&S), must be carefully selected to be able to implement the traded scenarios and provide the required data. The fidelity of the models should be tailored in accordance to the desired fidelity of the answer.

It is beyond the scope of this paper to identify the main influential trades and metrics for all possible missions and Mission Function Graphs, and future work may address the Pareto-dominant trades and metrics.

3. Step 3 – Modeling and iteration

The modeling and iteration step focuses on generating a representation of the Metrics-of-Value identified in step-2 to the necessary level of fidelity such that they can be traded against variations of the mission pipeline graph and its allocation to nodes. The objective is to iteratively obtain numeric and qualitative values of the traded configuration and improve the solution. Figure 7 and Figure 8 show the solution trade space in terms in the example 1 in section 6 shows a description of the type of trades that should be performed based on the mission pipeline graph.

The process in this step shall focus on iteratively identifying options, generating metrics and performing trade analysis of the alternatives. As the process evolves, the designer shall focus on asking 2nd-level questions, looking at the definition of mission pipelines that may improve or may provide insight into the solution space. Are there options? How do those options trade?

For example, in a time sensitive mission, the system designer may consider how latency can be reduced by modifying the input: Is adding more links to the design and placing the software process in other satellites an option in the design? Can the process be located on the ground and add more receiving ground stations? Is that a feasible option?

Or for example, in a mission where security is a key metric, how can mission security be ensured where the raw data is unclassified but processed data is classified? Can black data be downlinked to ground for fusion, then

perform high side fusion? Can data be processed in orbit in a classified "enclave" and use classified connections to the ground? Which of these solutions provide better outcomes with respect to the identified metrics? Does it need higher level of fidelity?

4. Step 4 – Decision Analysis

In order to feedback into the iteration of options in step-3, setting up some mechanism to analyze the trades is necessary. In particular, it is necessary to address the fact that the problem is a multi-objective optimization problem, with multiple dimensions and quantifiable and non-quantifiable metrics. We recommend 3 major elements to be used in the analysis:

1. A pareto analysis with linear combination of scores, by defining a simple numerical scale to the intangible metrics
2. A scorecard with pass/fail analysis of intangible utility requirements
3. A decision template that defines hard/medium/soft requirements.

A resourceful exploration of these analysis elements will provide guidance in the design and will enable, process-based decision making about when certain software functionality should de deployed in orbit.

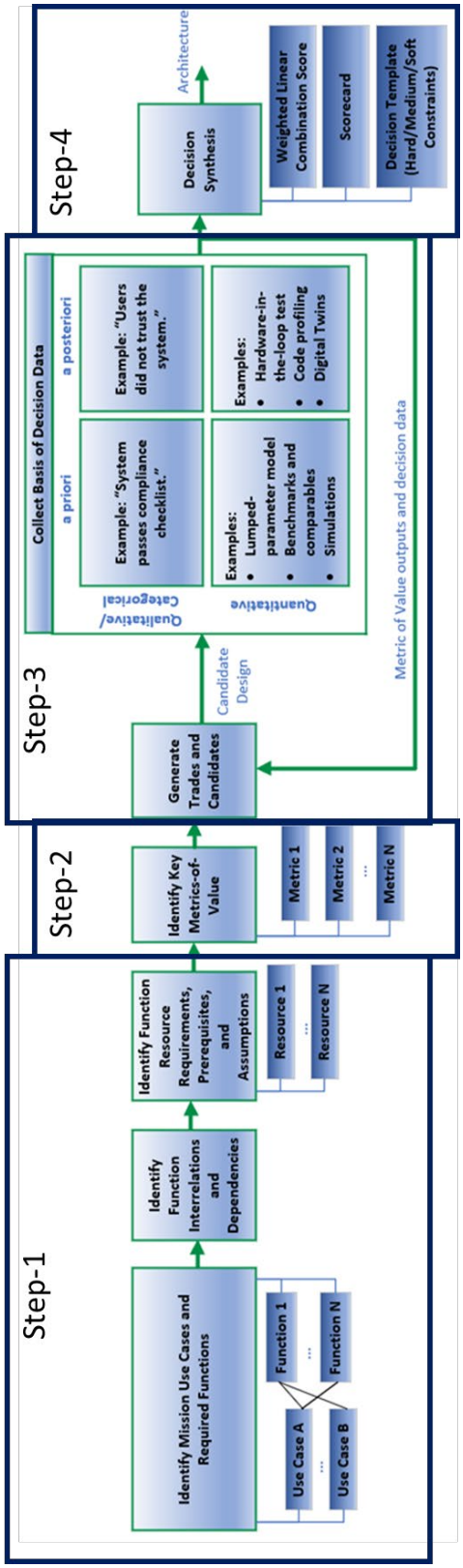

Figure 6: Overall Decision Framework Workflow

RESULTS

In order to test the proposed process, we implemented a couple of examples with different level of complexity. The results are described in this section.

Example 1. Trade-off analysis functionality edge processing target recognition.

For this example, we focused on analyzing the trade-off of performing target recognition for an ISR mission at the space edge vs. on the ground.

In the example we use a low-fidelity model to quantify and analyze the performance against time and accuracy leading to two possible functional implementations, either geo-rectifying and performing ATR in orbit or sending images to the ground and performing ATR on ground infrastructure. Figure 7 and Figure 8 show the solution trade space in terms of the distribution of the pipeline elements. The analysis shows that depending on the capacity of the network connectivity between the edge nodes and the ground, performing edge processing in orbit can be a better option.

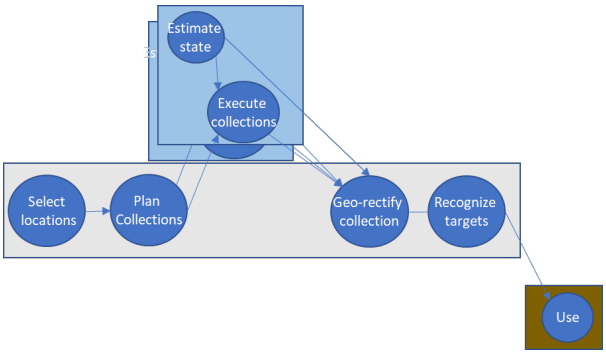

Figure 7. Option A: ISR target recognition mission with processing on the ground

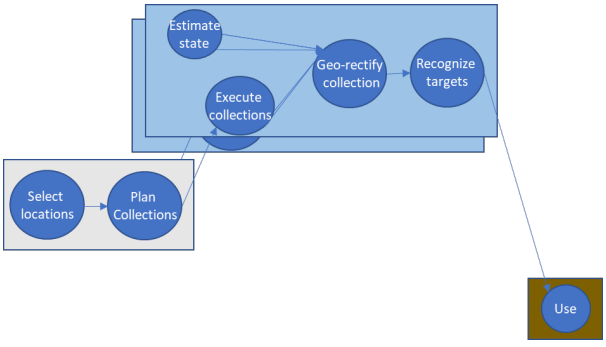

Figure 8. Option B: ISR target recognition mission with in-orbit processing

The analysis compared a traditional execution, described in Figure 7, where images are collected and sent to the ground for processing, vs. a scenario where images are processed on board and the resulting data is delivered described in Figure 8.

Notice that the graph description helps identifying the reasonable allocation of functionality at the node. It is not useful for example to 'just move' the analyzed function to the edge, some of the required input functionality also should to be moved with it.

Figure 3 shows an example of the trades that the analysis framework enables, providing some visibility of the value of pursuing some of the porting of functionality to operate at the edge when only considering the time of execution as the key Metric-of-Value. Option A does only improve with increased bandwidth, whereas option B is limited by the onboard processing. At some high bandwidth point, the trade lines cross and Option A has a better performance than Option B.

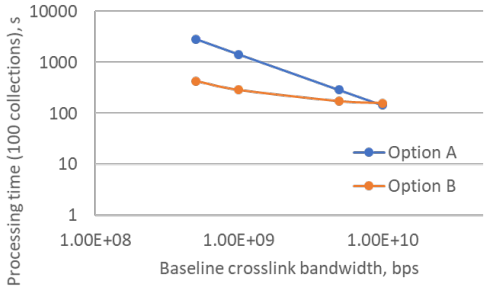

Figure 9. Average time to data delivery as the metric-of-value for option A and option B

Example 2: Edge processing trade-off for a complex combined space and air missions

In this example, we developed an example based on a more complex system, examining the trade between edge and ground-based processing for a fire monitoring mission which involves coordination between space-based assets searching for active wildfires and air-based UAV assets performing higher resolution fire monitoring surveillance. In this case, we implement a simulation of the entire system. We model the dynamics of the system using a physics-based simulator and the steps in the logic when processing the sensor data and the decision making in the involved nodes, as well as the network latency and routing of data.

In the mission simulated, the space layer is providing advanced notice of wildfire areas based on infrared sensing of hot spots, for the air-layer to route around,

avoiding these risk areas to enable image capture in less smoky air. Figure 3 below shows the difference in this mission when satellite image processing is done on the ground versus at the edge, onboard the satellites.

In this case, the modeling provides an analysis of a combined performance metric, which is the average and standard deviation of the time that the aircraft spent on high risk, high smoke zones. This metric was selected as the key metric of success of the entire system. The results are then captured via MonteCarlo simulation which enable to capture statistics and analyse performance and robustness to initial conditions. The example shows a 2.7% improvement in the median time spent in risk zones by using edge processing as opposed to ground processing with the associated delays. This improvement mostly arises from the improved response time due to edge computing, which provides relevant mission information to the air-layer with more time to allow for rerouting around risk zones.

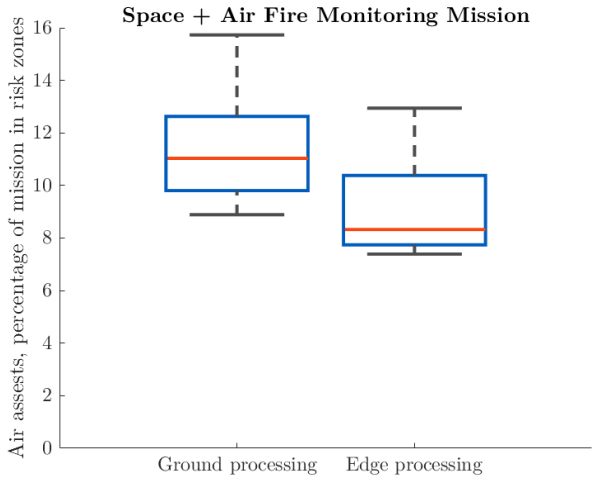

Figure 10. In a fire monitoring mission, the satellites identify wildfire locations with either ground or edge processing and relay the information to the air-based assets, which collect higher resolution data. The air-based assets route around the identified high-risk areas, which have smoke that will occlude data collection. Box plots are shown with the median indicated in orange, the 25th and 75th percentiles indicated by the blue box, and the gray lines indicating the full extent of the data.

CONCLUSIONS

In this paper we provided a view of the problem of identifying when a software application should be deployed to the space edge. We presented a structured decision framework with justifying rationale to provide insights and begin to address a key question of what mission compute functionality should be allocated to the space-based "edge", and under what mission or architectural conditions, versus to conventional ground-based systems.

This framework does not exhaustively address all missions, architectures, and CONOPs, however it is intended to provide generalized guidelines and heuristics to support architectural decision-making.

Results for a set of missions highlighting different key metrics and decision drivers show that in several cases, edge computing for specific functionality is quantitatively valuable, especially for interoperable, multi-domain, collaborative assets.

Acknowledgements

The authors would like to thank Gavin Strunk and Joseph Jackson for also contributing on modeling activities that provided the initial ideas for this paper and Dr. Michael Pagels at the Space Development Agency who originally posed some of the questions that inspired the authors to think about this problem.

References

1. Taneja, Mohit, and Alan Davy. "Resource aware placement of IoT application modules in Fog-Cloud Computing Paradigm." 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM). IEEE, 2017.
2. Dutta, Debojyoti, et al. "Embedding paths into trees: VM placement to minimize congestion." Algorithms–ESA 2012: 20th Annual European Symposium, Ljubljana, Slovenia, September 10-12, 2012. Proceedings 20. Springer Berlin Heidelberg, 2012.
3. Lukovszki, Tamás, and Stefan Schmid. "Online admission control and embedding of service chains." Structural Information and Communication Complexity: 22nd International Colloquium, SIROCCO 2015, Montserrat, Spain, July 14-16, 2015. Post-Proceedings 22. Springer International Publishing, 2015.
4. Gupta, Harshit, et al. "iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments." Software: Practice and Experience 47.9 (2017): 1275-1296.
5. Bonomi, Flavio, et al. "Fog computing and its role in the internet of things." Proceedings of the first edition of the MCC workshop on Mobile cloud computing. 2012.
6. Keshavarznejad, Maryam, Mohammad Hossein Rezvani, and Sepideh Adabi. "Delay-aware

optimization of energy consumption for task offloading in fog environments using metaheuristic algorithms." Cluster Computing (2021): 1-29.

7. Furano, Gianluca, et al. "Towards the use of artificial intelligence on the edge in space systems: Challenges and opportunities." IEEE Aerospace and Electronic Systems Magazine 35.12 (2020): 44-56.
8. Akbar, Aamir, et al. "SDN-enabled adaptive and reliable communication in IoT-fog environment using machine learning and multiobjective optimization." IEEE Internet of Things Journal 8.5 (2020): 3057-3065.
9. Liu, Bingwei, et al. "Information fusion in a cloud computing era: a systems-level perspective." IEEE Aerospace and Electronic Systems Magazine 29.10 (2014): 16-24.
10. Munir, Arslan, et al. "Artificial intelligence and data fusion at the edge." IEEE Aerospace and Electronic Systems Magazine 36.7 (2021): 62-78.
11. Wang, Shiqiang, Murtaza Zafer, and Kin K. Leung. "Online placement of multi-component applications in edge computing environments." IEEE Access 5 (2017): 2514-2533.
12. Satyanarayanan, Mahadev, et al. "The role of cloudlets in hostile environments." IEEE Pervasive Computing 12.4 (2013): 40-49.
13. Mahmoodi, S. Eman, R. N. Uma, and K. P. Subbalakshmi. "Optimal joint scheduling and cloud offloading for mobile applications." IEEE Transactions on Cloud Computing 7.2 (2016): 301-313.
14. Pfandzelter, Tobias, and David Bermbach. "QoS-Aware Resource Placement for LEO Satellite Edge Computing." 2022 IEEE 6th International Conference on Fog and Edge Computing (ICFEC). IEEE, 2022.
15. Dong, Feihu, et al. "A Computation Offloading Strategy in LEO Constellation Edge Cloud Network." Electronics 11.13 (2022): 2024.
16. Lai, Zeqi, et al. "Cooperatively constructing cost-effective content distribution networks upon emerging low earth orbit satellites and clouds." 2021 IEEE 29th International Conference on Network Protocols (ICNP). IEEE, 2021.
17. Pfandzelter, Tobias, and David Bermbach. "Edge (of the earth) replication: Optimizing content delivery in large leo satellite communication networks." 2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid). IEEE, 2021.
18. Bhattacherjee, Debopam, et al. "In-orbit computing: An outlandish thought experiment?." Proceedings of the 19th ACM Workshop on Hot Topics in Networks. 2020.
19. Qiu, Yuan, et al. "Mobile Edge Computing in Space-Air-Ground Integrated Networks: Architectures, Key Technologies and Challenges." Journal of Sensor and Actuator Networks 11.4 (2022): 57.
20. Guo, Yaning, et al. "Towards Optimization for Large-scale Earth Observation Missions from a Global Perspective." 5th Asia-Pacific Workshop on Networking (APNet 2021). 2021.
21. Picard, Gauthier. "Auction-based and distributed optimization approaches for scheduling observations in satellite constellations with exclusive orbit portions." arXiv preprint arXiv:2106.03548 (2021).
22. Zeleke, Desalegn Abebaw, and Hae-Dong Kim. "A New Strategy of Satellite Autonomy with Machine Learning for Efficient Resource Utilization of a Standard Performance CubeSat." Aerospace 10.1 (2023): 78.
23. Cruz, Helena, et al. "A review of synthetic-aperture radar image formation algorithms and implementations: a computational perspective." Remote Sensing 14.5 (2022): 1258.
24. Zhang, Liang, et al. "Fast superpixel-based non-window CFAR ship detector for SAR imagery." Remote Sensing 14.9 (2022): 2092.
25. Zhang, Fan, et al. "Accelerating spaceborne SAR imaging using multiple CPU/GPU deep collaborative computing." Sensors 16.4 (2016): 494.
26. Agrawal, A. K., et al. "Accelaerated SAR image generation on GPGPU platform." 2011 3rd International Asia-Pacific Conference on Synthetic Aperture Radar (APSAR). IEEE, 2011.