Security-aware Resource Brokering for Bioinformatics Workflows across Federated Multi-cloud Infrastructures

Minh Nguyen
mnguyen@gradcenter.cuny.edu
City University of New York
New York, NY

Saptarshi Debroy
saptarshi.debroy@hunter.cuny.edu
City University of New York
New York, NY

Prasad Calyam
calyamp@missouri.edu
University of Missouri
Columbia, MO

Zhen Lyu
zl7w2@mail.missouri.edu
University of Missouri
Columbia, MO

Trupti Joshi
joshitr@health.missouri.edu
University of Missouri
Columbia, MO

ABSTRACT

Data-intensive science applications often use federated multi-cloud infrastructures to support their compute-intensive processing needs. However, lack of knowledge about: a) individual domain's security policies, b) how that translates to application security assurance, and c) nature of performance and security trade-offs - can cause performance-security conflicts for applications and inefficient resource usage. In this paper, we propose a security-aware resource brokering middleware framework to allocate application resources by satisfying their performance and security requirements. The proposed middleware implements MCPS (Multi-Cloud Performance and Security) Broker that uses a common data model to represent applications' performance and security requirements. It performs a security-aware global scheduling to choose the optimal cloud domain, and a local scheduling to choose the optimal server within the chosen cloud domain. Using real SoyKB application workflows, we implement the proposed MCPS Broker in the GENI Cloud and demonstrate its utility through a NIST-guided risk assessment.

CCS CONCEPTS

• **Security and privacy** → **Security requirements**; • **Networks** → **Cloud computing**; *Network resources allocation.*

KEYWORDS

Data-intensive application workflows, Federated multi-cloud infrastructures, Resource allocation, End-to-end security management

ACM Reference Format:
Minh Nguyen, Saptarshi Debroy, Prasad Calyam, Zhen Lyu, and Trupti Joshi. 2020. Security-aware Resource Brokering for Bioinformatics Workflows across Federated Multi-cloud Infrastructures. In *21st International Conference on Distributed Computing and Networking (ICDCN 2020), January 4–7, 2020, Kolkata, India.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3369740.3369791

ICDCN 2020, January 4–7, 2020, Kolkata, India

ACM ISBN 978-1-4503-7751-5/20/01…$15.00
https://doi.org/10.1145/3369740.3369791

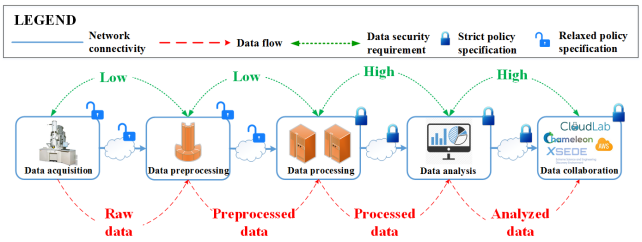


Figure 1: End-to-end lifecycle stages of a data-intensive application with dynamic security requirements using federated multi-cloud resources from domains with diverse resource policy specifications.

1 INTRODUCTION

Data-intensive science applications (e.g., in areas of high-energy physics, bioinformatics) often require specialized compute/networking/storage resources that are not always available locally on-site [1] and need to use resources in remote cloud domains for processing. Thus, researchers are increasingly adopting federated multi-cloud infrastructures (e.g., CyVerse [2], XSEDE [3]) to support compute-intensive or data-intensive science collaborations. Adoption of such infrastructures are facilitated by Software-defined Networking (SDN) [4] enabled campus Science DMZs [5] for friction-less data movement and Federated Identity and Access Management (IAM) [6] that enables campus researchers to reserve and seamlessly access local and remote cloud resources.

Allocation of such federated multi-cloud resources is typically based on applications' performance considerations (e.g., data throughput, execution time). However, such one-dimensional resource brokering fails to consider scenarios where applications' security requirements across different life-cycle stages (Low, Moderate, and High) contradict with remote domains' diverse security policies (ranging from very strict to very relaxed) as shown in Fig. 1. It is a difficult proposition for users (especially when using complex workflows) to guess how to select options available within federated multi-cloud resources in a manner that overcomes bottlenecks such as resource capacity limitations, security posture or cost factors at the various resource domains. Without a systematic framework and standardized tools, performance-security conflicts for applications and inefficient/expensive resource usage scenarios occur that are undesirable from a user perspective.

In this paper, we propose a security-aware resource brokering middleware framework for a set of SoyKB [7] bioinformatics workflows across federated multi-cloud infrastructures that features a MCPS (Multi-Cloud Performance and Security) Broker component. The proposed middleware builds upon our earlier work [8] by formalizing performance specifications or *QSpecs* and security specifications or *SSpecs* of exemplar SoyKB workflows, such as a simple RNA-Seq [9] workflow and a complex PGen [10] workflow. The middleware also facilitates a end-to-end workflow security design that formalizes and complies with diverse domain security policies or *RSpecs* used by the application relating to a local University of Missouri (MU) domain, as well as remote cloud domains, such as Texas Advanced Computing Center (TACC) [11], and Information Sciences Institute (ISI) [12].

Our novel MCPS Broker allocates multi-cloud resources to workflows using their Directed Acyclic Graphs (DAG) representations, with each stage of the workflow being represented as graph vertices. Consequently, the resource allocation becomes a multi-constrained DAG scheduling problem which is traditionally NP-complete [13]. To solve the problem, we propose a global scheduling algorithm to identify optimal cloud domain resources suitable for each stage of the workflow, and a local scheduling algorithm to identify a server/-core within the chosen domain. The result of the algorithms are a list of chosen domains and cores within the domains for each stage of a workflow that: (i) satisfies both workflow *QSpecs* and *SSpecs*, (ii) is compliant with domain *RSpecs*, and (iii) provides an optimized resource allocation for both simple (RNA-Seq) and complex (PGen) bioninformatics workflows.

In addition, we implement our MCPS Broker algorithms within a multi-cloud testbed on a GENI Cloud [14] infrastructure. The testbed replicates security policies of individual domains, such as, TACC, ISI, and MU as well as resource utilization levels at steady states. Our MCPS Broker implementation features a user portal with user and administrator dashboards that help monitor workflow and domain resource utilization status. Using simulated data mimicking RNA-Seq and PGen workflows submitted through the MCPS Broker user portal, we show how the outcome of the proposed algorithm is as good as one-dimensional performance-driven resource allocation in terms of workflow *QSpecs* satisfaction. At the same time, using National Institute of Standards and Technology (NIST) [15] based risk assessment, we show how the outcome of the proposed algorithm satisfies workflow *QSpecs* while also being compliant with the individual domains' *RSpecs*.

The rest of the paper is organized as follows. Section 2 discusses related work. Section 3 presents and overview of SoyKB workflows. Section 4 describes the system model and MCPS Broker algorithms design. Section 5 discusses the GENI testbed implementation and the results from our performance and security evaluation experiments. Section 6 concludes the paper.

2 RELATED WORK

Traditional resource allocation approaches are majorly limited to performance requirements scope when handling computation-intensive/data-intensive workflows. Prior works that feature different approaches based upon the broad goals of end-to-end workflow performance and dependability can be found in [16–18]. The authors in [16] deal with end-to-end QoS constraints in the resource allocation of web services. In [17], the authors tackle the problem of on-demand and concurrent application handling for accelerated performance of application workflows by using hybrid cloud computing architectures. The authors in [18] show how application traffic type such as multimedia or file transfer can determine performance requirements. *In all of these works, security requirement aspects of workflows are not explicitly considered within the user interactions while performing resource brokering across a set of federated multi-cloud domains.*

DAG Scheduling is known to be a NP-complete problem [13]. Hence, it is performed using heuristic algorithms in practical scenarios. Most heuristics are based on a list scheduling approach [19], where a weight is assigned to each vertex and edge in order to represent the costs. These weights are also used to compute the priority set for the vertices and they will be scheduled in the order of this priority list. Among the heuristics, work such as Modified Critical Path (MCP) [20], Levelized-Min Time (LMT) [21], Fastest Critical Path (FCP) [22], and Heterogeneous Earliest Finish Time (HEFT) [23] have shown impressive performance. In MCP [20], the authors assign the scheduling priority based on the latest start time. The latest start time is defined as the difference between the longest path in the graph (critical path) and the longest path from the current vertex to any exit vertex (vertex with no output edges). A path length in this case, is the summation of inherent vertex weights and edge weights. The authors in LMT [21] compute the scheduling priority in two phases. First, they categorize the vertices into different levels based the DAG topology. Within the same level, the priority list is computed based on the greatest vertex weights. The authors in FCP [22] compute the priority by calculating the *bottom level*, defined as the longest path from the current vertex to any exit vertex. Finally, in HEFT [23], the authors use the vertices' earliest start times as the priority, similar to MCP. However, HEFT is dynamic task priority algorithm, meaning the priorities are computed at each scheduling step for ready unscheduled vertices. *Our DAG Scheduling algorithms build upon the FCP approach, and mainly focus on minimizing execution costs and maximizing the performance output. Our novelty lies in our proposed approach that adds a security angle to the DAG Scheduling for a trade off balance between performance, cost and security of the resource-provisioned bioinformatics workflows.*

Existing works pertaining to *security and dependability for federated multi-cloud resources in data-intensive research communities* mostly deal with security measures and point solutions to counter confidentiality, availability, and integrity threats. They also do not consider end-to-end security design that helps in dynamic allocation and adaptation using such measures. Exemplar solutions to Big Data transfer in a federated environment [25] include Globus efforts [24] that provide the ability to use point solutions such as InCommon [26], OpenID [27] and X.509 [28] to access resources. The Large Synoptic Survey Telescope (LSST) community on the other hand provides detailed guidelines for multi-domain cybersecurity compliance with a list of threat mitigating capabilities at involved domains [29] [30]. *These communities can benefit from our formal approach of resource allocation based on multi-domain security requirements, and augment their current approach of manual co-ordination of policies to achieve end-to-end security alignment.*

3 SOYKB WORKFLOWS AND MULTI-CLOUD INFRASTRUCTURE

3.1 PGen and RNA-Seq workflows

For the purposes of this work, we consider the implementations of two high-throughput cloud-based bioinformatics data analysis workflows in the SoyKB [7] science gateway developed for soybean and other related organisms. These workflows provide biological users with an avenue to analyze their in-house generated datasets using multi-step workflows and conduct analysis in high performance computing environments that support the necessary security levels to handle Health Insurance Portability and Accountability Act (HIPPA) compliance [31].

The complex PGen workflow [10] is used to efficiently facilitate analysis of large-scale next generation sequencing (NGS) data for genomic variations. The workflow allows users to identify single nucleotide polymorphisms (SNPs) and insertion-deletions (indels), perform SNP annotation and conduct copy number variation analyses on multiple re-sequencing datasets. The PGen workflow has been developed using many widely accepted open-source NGS tools for alignment of reads, variants calling, variants filtration, vcf merging and others. As shown in Fig. 2 (a), the workflow starts by indexing the reference genome (Stage 1). Then, it aligns both pair-end or single-end fast reads against the reference genome using BWA [32] (Stages 2-7). Picard Tools [33] are also used at this step to locate duplicate molecules and assign all reads into groups with the default parameters. After alignment, SNPs and indels are called using the Haplotype caller algorithm from Genome Analysis Toolkit (GATK) [34] (Stage 9 and 10). Filtering criteria are defined in INFO filed in vcf file, where QD stands for quality by depth, FS is Fisher strand values and MQ is mapping quality of variants (Stage 12 and 13). Detected variants are then filtered using criteria $QD < 26.0||FS > 60.0||MQ < 40.0$ for SNPs and $QD < 26.0||FS > 200.0||MQ < 40.0$ for indels. Custom criteria can also be applied by the user. Outputs are generated as BAM and VCF standard formats (Stages 11, 14-16, and 15-17).

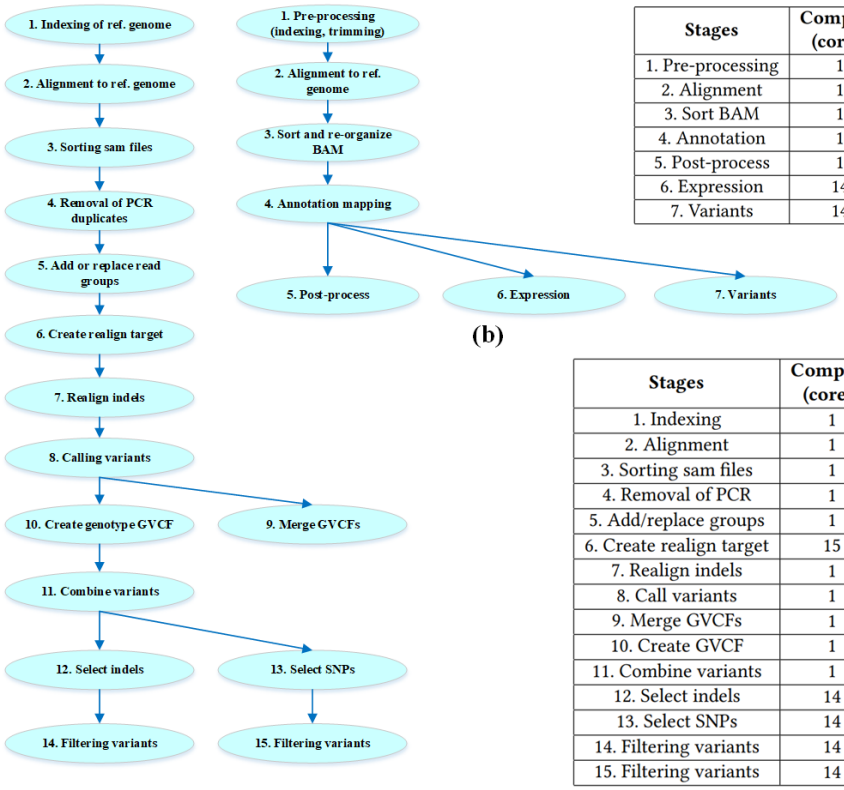
We also consider a comparatively simpler RNA-Seq [9] analysis workflow that is used to perform quantization of gene expression from transcriptomics data and statistical analysis to discover differential expressed gen/isoform between experimental groups/conditions. As shown in Fig. 2 (b), the RNA-Seq analysis consists commonly of five steps. Firstly, the reference genome is pre-processed by indexing or trimming (Stage 1). Secondly, the pair/single-end reads are aligned to the reference genome via TopHat2 [35] (Stage 2). Next, the mapped reads are summarized and aggregated over genes and isoforms to calculate the FPKMs via Cufflinks (Stage 3). Then, the transcriptome assembly generated from Cufflinks is processed via Cuffcompare to perform these comparisons and assess the quality of assembly (Stage 4). Finally, genes and isoforms expressed differentially between the different groups/conditions are identified using Cuffdiff [36] (Stage 5, 6, and 7).

3.2 Workflow *QSpecs* and *SSpecs* and domain *RSpecs*

PGen and RNA-Seq workflows rely on the Pegasus [37] Workflow Management System, which splits the workflows into MPI jobs to map them into available multi-cloud domain cores. The different life-cycle stages of the workflow processing explained earlier can be either carried out within a local organization i.e., our MU private cloud or remote cloud sites of XSEDE (e.g., ISI, TACC). Following computation, the processed data with meaningful results are made available to the worldwide user community accessible via iRODS [38] at CyVerse datastore. The objective of a formalized *QSpecs* is to express the minimum compute, storage, and network resource specifications of different stages of the workflow processing life-cycle that satisfy the workflow quality of service (QoS) requirements. Figs. 2 (c) and 2 (d) express the *QSpecs* of PGen and RNA-Seq workflows in tabular forms. The *QSpecs* expresses the number of compute cores, memory storage in GBs, and network bandwidth in Mbps specifications for each stage of the workflow life-cycles shown in Figs. 2 (a) and 2 (b). In the designed *QSpecs* of PGen and RNA-Seq, the number of compute cores are calculated for a processor of standard speed across the multi-cloud domains. As for the network bandwidth specification, PGen and RNA-Seq workflows do not specify network speed requirements. Information collected from proposed *QSpecs* are used later in Section 4 for problem formulation and MCPS Broker algorithm design.

On the other hand, workflow *SSpecs* is a formal data structure to describe the minimum security requirements against confidentiality, integrity, and availability threats for every stage of the life-cycle at the granularity of both the data-level and instrument-level. This includes handling issues with multi-cloud resources for the data, instruments, software tools, and personnel involved. Our 'Data' and 'Auxiliary' security requirements driven *SSpecs* is based on NIST SP 800E guidelines [15] that is described in detail in our recent prior works [8]. As shown in Tables 1 and 2, the *SSpecs* of PGen and RNA-Seq respectively are similar to *QSpecs*, in terms of life-cycle based structure. The *SSpecs* 'Data' requirements are divided into Compute, Storage, and Network requirements, i.e., resources that deal with the data. Whereas, 'Auxiliary' requirements are those that cannot be categorized into Compute, Storage, or Network resources. These involve combinations of scientific instruments, analysis tools and related software licenses, and users/personnel requirements of the lifecycle stages. The values for these categories are given Low (L), Moderate (M), and High (H) ratings based on requirements collected through a careful and relevant discussion with the PGen and RNA-Seq workflow users. Due to the HIPPA compliance requirement of both workflows and homogeneity of stages in terms of protection against data confidentiality, integrity, and availability threats, the *SSpecs* category ratings are same across all stages and across the workflows. For both workflows, the compute and storage security requirements tend are more elaborate than network due to lack of network resource requirements. The workflows have consistently 'H' access control (AC), authentication (IA), and authorization (CA) requirements due to HIPAA sensitivity. Whereas other categories are default 'L' due to no explicit specification. The auxiliary requirements are 'L' across categories due to low program and personnel management requirements.

One of the major barriers to wider adoption of multi-cloud infrastructures for cross-domain data collaboration is the fact that researchers have little-to-no knowledge of the security capabilities of the involved domains and whether such domains can satisfy the application security requirements (often at a strict level). At the



Stages	Compute (cores)	Storage (GB)	Network (Mbps)
1. Pre-processing	1	4	NA
2. Alignment	1	20	NA
3. Sort BAM	1	20	NA
4. Annotation	1	20	NA
5. Post-process	1	10	NA
6. Expression	14	10	NA
7. Variants	14	10	NA

(d)

Stages	Compute (cores)	Storage (GB)	Network (Mbps)
1. Indexing	1	4	NA
2. Alignment	1	21	NA
3. Sorting sam files	1	21	NA
4. Removal of PCR	1	21	NA
5. Add/replace groups	1	21	NA
6. Create realign target	15	20	NA
7. Realign indels	1	10	NA
8. Call variants	1	3	NA
9. Merge GVCFs	1	20	NA
10. Create GVCF	1	10	NA
11. Combine variants	1	10	NA
12. Select indels	14	10	NA
13. Select SNPs	14	10	NA
14. Filtering variants	14	10	NA
15. Filtering variants	14	10	NA

(c)

Figure 2: (a) DAG representation of PGen workflow, (b) DAG representation of RNA-Seq workflow, (c) *QSpecs* of PGgen workflow, (d) *QSpecs* of RNA-Seq workflow

Stages	Compute							Storage							Network							Auxiliary										
	AC	AU	CA	IA	SA	SC	SI	AC	AU	CA	IA	SA	SC	SI	AC	AU	CA	IA	SA	SC	SI	AT	CM	CD	IR	MA	MP	PE	PL	PM	PS	RA
1	H	L	H	H	L	L	L	H	L	H	H	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L
2	H	L	H	H	L	L	L	H	L	H	H	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L
3	H	L	H	H	L	L	L	H	L	H	H	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L
4	H	L	H	H	L	L	L	H	L	H	H	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L
5	H	L	H	H	L	L	L	H	L	H	H	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L
6	H	L	H	H	L	L	L	H	L	H	H	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L
7	H	L	H	H	L	L	L	H	L	H	H	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L
8	H	L	H	H	L	L	L	H	L	H	H	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L
9	H	L	H	H	L	L	L	H	L	H	H	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L
10	H	L	H	H	L	L	L	H	L	H	H	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L
11	H	L	H	H	L	L	L	H	L	H	H	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L
12	H	L	H	H	L	L	L	H	L	H	H	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L
13	H	L	H	H	L	L	L	H	L	H	H	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L
14	H	L	H	H	L	L	L	H	L	H	H	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L
15	H	L	H	H	L	L	L	H	L	H	H	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L

Table 1: PGen workflow *SSpecs*

Stages	Compute							Storage							Network							Auxiliary										
	AC	AU	CA	IA	SA	SC	SI	AC	AU	CA	IA	SA	SC	SI	AC	AU	CA	IA	SA	SC	SI	AT	CM	CD	IR	MA	MP	PE	PL	PM	PS	RA
1	H	L	H	H	L	L	L	H	L	H	H	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L
2	H	L	H	H	L	L	L	H	L	H	H	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L
3	H	L	H	H	L	L	L	H	L	H	H	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L
4	H	L	H	H	L	L	L	H	L	H	H	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L
5	H	L	H	H	L	L	L	H	L	H	H	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L
6	H	L	H	H	L	L	L	H	L	H	H	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L
7	H	L	H	H	L	L	L	H	L	H	H	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L

Table 2: RNA-Seq workflow *SSpecs*

Domains	Compute							Storage							Network							Auxiliary										
	AC	AU	CA	IA	SA	SC	SI	AC	AU	CA	IA	SA	SC	SI	AC	AU	CA	IA	SA	SC	SI	AT	CM	CD	IR	MA	MP	PE	PL	PM	PS	RA
MU	H	H	M	M	M	H	H	H	H	M	M	M	H	H	H	L	H	H	L	H	H	M	M	M	L	M	H	H	L	L	H	M
TACC	H	H	M	M	H	H	H	H	H	M	M	H	H	H	H	L	H	H	L	H	H	H	H	H	H	H	H	H	H	M	H	H
ISI	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H	L	H	H	H	H	H	M	H	H	H	M	M	H	H

Table 3: Aligned *RSpecs* of MU, TACC, and ISI domains

same time, in most cases the domain policies of security postures are diverse and cannot be easily compared with the application security specifications for compliance. In this work, we build upon our prior works [8], where we analyze the security policies of select institutions/universities related to the SoyKB workflows, such as TACC, ISI and MU. We align diverse/heterogeneous security policies of these institutions into homogeneous policy specifications or *RSpecs* that are comparable to the PGen and RNA-Seq workflows' *SSpecs* for resource brokering. We use a 3-step security alignment process where we: (a) first categorize the policies based on the type of resources (i.e., network, compute or storage), (b) then drill down security policies pertaining to each of the resource types into homogeneous formal policy statements using the "Portunes Algebra" [39], and (c) finally assign security levels to each such resources by applying the NIST SP 800E guidelines. The outcome of such a process is a homogeneous categorization of different domain *RSpecs* that is consistent with workflow *SSpecs*.

4 SYSTEM AND ALGORITHMS DESIGN

In this section, we first present the DAG scheduling system model and problem formulation. Following this, we detail global and local scheduling algorithms design.

4.1 System model and problem formulation

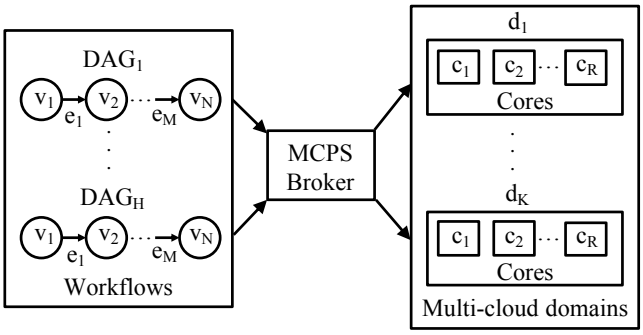


Figure 3: System model for MCPS Broker for mapping workflow DAG stages onto multi-cloud domains and to compute cores within the domains.

As stated earlier, SoyKB workflows are represented as DAGs with vertices representing individual lifecycle stages and edges representing stage transition (as shown in Fig. 2 (a) and 2 (b). For our model, we formally represent a DAG as $G = (V, E)$, where $V = \{v_1, v_2, \cdots, v_N\}$ is the set of N vertices and $E = \{e_1, e_2, \cdots, e_M\}$ is the set of M edges. At the same time, we maintain an independent set $D = \{d_1, d_2, \cdots, d_K\}$ that represents K multi-cloud domains each with R cores $\{c_1, c_2, \cdots, c_R\}$. The objective is to schedule each such DAG vertex to one or more cores within an individual domain as shown in Fig. 3. To achieve this, we divide the overall problem into two sub-problems: i) global scheduling for mapping vertices to domains, and ii) local scheduling for mapping domains to cores.

For the global scheduling, we assume that each vertex $v \in V$ has a weight $comp(v)$ which represents the data *computation time* to process the stage and each edge $(v, v') \in E$ has a weight $trans(v, v')$ which represents the *transfer time* of transferring data from vertex (stage) v to vertex v'. The values of $comp(v)$ and $trans(v, v')$ can be easily computed from workflow *QSpecs* (as shown in Fig. **??**). Each vertex $v \in V$ also has a security tuple $sec(v) = \{s_c, s_s, s_n, s_a\}$ denoting minimum security requirements for compute, storage, network, and auxiliary categories respectively. Each element of the tuple is generated from workflow *SSpecs* by taking the maximum of individual security requirements of each sub-category (i.e., AC, AU etc.) within that category (i.e., compute). For example, for PGen workflow, s_c for first stage (vertex) will be 'L' (from Table 1). At the same time, each domain $d \in D$ has a policy tuple $pol(d) = \{p_c, p_s, p_n, p_a\}$ denoting maximum security that the domain can support for compute, storage, network, and auxiliary categories respectively. In our model, $pol(d)$ is generated from domain *RSpecs* similar to $sec(v)$.

We consider B^i_j to be the binary variable that is equal to 1 if vertex v_i is scheduled to domain d_j and is equal to 0 when not. We also assume that the start and finish times of processing v_i on d_j are $ST(v_i)$ and $FT(v_i)$ respectively. Since domain d_j contains multiple cores, $ST(v_i)$ and $FT(v_i)$ are measured based on the current available fastest core. The *domain ready time* $DRT(d_j)$ of a domain d_j is defined as the finish time of the last vertex v_i of the workflow scheduled on that domain. Thus,

$$DRT(d_j) = max_{v_i \in V, B^i_{j=1}} FT(v_i) \quad \forall\ i, j \tag{1}$$

The objective is to schedule all vertices in V on domains within D in a way that the parallel completion time (schedule length) is minimized:

$$\boxed{\text{Minimize} \quad max_{d_j \in D} DRT(d_j) \quad \forall\ j} \tag{2}$$

The above objective function must be satisfied without violating the following constraints:

Precedence constraints: In a workflow DAG, the next vertex can only be scheduled if and only if all of its parents have finished; we call these vertices *ready* stages, i.e.,

$$FT(v_i) \geq max_{v_i \in V} FT(parent(v_i)) \quad \forall\ i \tag{3}$$

Security constraints: The chosen domain's security policies from *RSpecs* must satisfy the stage's (vertex) security requirements *QSpecs*, i.e.,

$$pol(d_j) \geq sec(v_i) \quad \forall\ B^i_j = 1 \tag{4}$$

Upon scheduling a workflow to a domain, it is the responsibility of the local scheduling to assign compute cores to each stage of the workflow. For each core c in a domain, we assume the computational power to be $pow(c)$. We denote $b^i_{j,t}$ as the binary variable that is equal to 1 if DAG vertex v_i is assigned to core c_t of domain d_j and is equal to 0 if not. Since the local scheduling aims to maximize the performance, we assign each vertex to the most powerful core in terms of computation speed available within that domain, thus generating the optimization problem,

$$\boxed{\text{Maximize } \; b^i_{j,t} * pow(c_t) \quad \forall\, i, j, t} \tag{5}$$

The above objective function must be satisfied without violating the following constraint:

No overlap constraint: We need to ensure that no two vertices are assigned to the same core at a given time, i.e.,

$$\sum_{i=1}^{n} b^i_{j,t} = 1 \quad \forall\, j, t \tag{6}$$

The global DAG scheduling optimization problem is NP-complete [13] and can be solved using heuristics. For MCPS broker, we utilize a modified version of Fast Critical Path (FCP) heuristic algorithm [22] due to its better performance while still maintaining a relatively low running time. For local scheduling, we apply a simple algorithm to find the core with maximum element computational power from an unsorted list while maintaining an extra Boolean list for checking busy cores. Both algorithms are described and explained next.

4.2 Global scheduling algorithm

Algorithm 1 describes the global scheduling algorithm. In a workflow DAG, we call the longest path in the graph as a *critical path*. A vertex or stage with no input edges is an *entry stage*, while a stage with no output edges is an *exit stage*. A stage's *bottom level* is defined as the longest path from the current stage to any exit stage; the path length is the sum of *comp*() and $trans$() weights of the stages and edges belonging to the path. Our algorithm essentially executes the following three functions.

AddReadyStage is a void function that adds a *ready* stage to the partially sorted stage set. The stage set contains a priority list of fixed size Γ and an unsorted list. If the fixed size priority list is not full, the stage is added to the priority list, otherwise to the unsorted list. The algorithm uses *bottom level* to calculate the priority of stages.

SelectReadyStage returns the stage with the highest priority from the priority list. The priority list must be full as long as there are stages in the unsorted list. Hence, if a stage is dequeued from the priority list while there exist stages in the unsorted list, one of the tasks in the unsorted list must be moved to the priority list. That way, the priority list is always full if there are tasks that exist in the unsorted list.

SelectDomain is the security driven domain selection. A domain can only be selected if it satisfies the *Security constraints*. After eliminating disqualified domains, two candidates will be picked out: (i) the domain with the highest security policies dA and (ii) the domain that becomes available the earliest dB. The former is used to maximize security, while the latter ensures load balancing among domains. The stage with the earliest start time will be selected.

Schedule is the core function of the algorithm. It uses the *bottom level* as the static stage priority. The ready stage set is initialized with the entry stages. The scheduling loop is repeated as long as there are unscheduled stages. At each iteration, one stage is scheduled. The task to be scheduled is selected among ready tasks using *SelectReadyStage* function. The destination domain for the chosen stage is selected using *SelectDomain* function based on the *Security constraints*. If there is no satisfied domains currently, the stage will

Algorithm 1: Global scheduling algorithm.

Input: Workflow DAG $G = (V, E)$ with *comp*() and $trans$(), domain set D, ST(), and FT().
Output: Stages scheduled to domains to maximize $max_{d_j \in D} DRT(d_j)$.

```
1  Function AddReadyStage(stage):
2      if size_of(priority_list) ≤ Γ then
3          Enqueue_sorted(stage, priority_list);
4      else
5          Enqueue(stage, unsorted_list);
6  Function SelectReadyStage():
7      stage ← Dequeue(priority_list);
8      if unsorted_list is not empty then
9          v ← Dequeue(unsorted_list);
10         Enqueue(v, priority_list);
11     return stage;
12 Function SelectDomain(stage):
13     dA ← argmax_{d_j∈D}(pol(d_j)) ;  // the domain with the
        highest security policies.
14     dB ← domain becoming available the earliest ;   // the
        domain with at least one non-busy core.
15     if ST(stage, dA) < ST(stage, dB) then
16         d ← dA;
17     else
18         d ← dB;
19     return d;
20 Function Schedule(stage):
21     for v ∈ V do
22         ComputePriority(v) ;   // based on bottom level
            of v.
23         if v is an entry stage then
24             AddReadyStage(v);
25     while not all tasks scheduled do
26         stage ← SelectReadyStage();
27         if pol(d) ≥ sec(stage) then
28             d ← SelectDomain(stage);
29             Assign(stage, d);
30         for v ∈ updated ready stage set do
31             AddReadyStage(v);
```

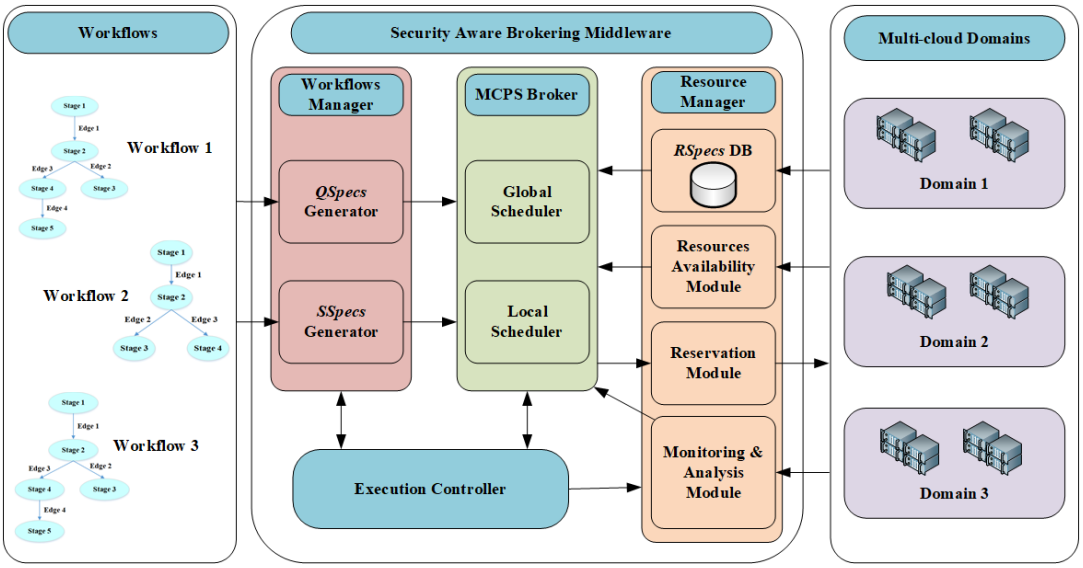


Figure 4: Security-aware resource brokering middleware services design and underlying components.

be returned back to the end of ready set. Before continuing with the following iteration, the ready stage set is updated by inserting back the unscheduled stages and adding the successors that become ready.

4.3 Local scheduling algorithm

Algorithm 2 describes the local scheduling algorithm. The approach is simpler compared to the global scheduling as it aims to map the scheduled stage to a core $c_i \in C$ inside a local organization domain that maximizes stage performance. The algorithm maintains a list of computational power *POW* for all cores within the local domain. The algorithm uses a Boolean list *busy* of same size with *C* to ensure the *No overlap constraint*; the values of this list's elements are equal to 1 if the respective cores are busy and are equal to 0 otherwise.

Algorithm 2: Local scheduling algorithm.

Input: Cores list $C = \{c_i\}$, cores' computational power $POW = \{pow_i\}$.
Output: The non-busy core with maximum *pow*.

```
busy = {0} ;                    // initialize busy list with 0.
for c_i ∈ C do
    if c_i is busy then
        busy_i ← 1;
max = pow_0;
for pow_i ∈ POW do
    if pow_i > max and busy_i == 0 then
        max ← pow_i;
return max;
```

4.4 Middleware service design

Fig. 4 illustrates the overall security-aware resource brokering middleware services design and the underlying components. The *Workflow Manager* gathers DAG information of input workflows and generate the *QSpecs* and *SSpecs* through the *QSpecs Generator* and *SSpecs Generator* modules. The *QSpecs* and *SSpecs* information are sent to *MCPS Broker* module for resource scheduling. The MCPS Broker related *Domains Resources Module* also gathers the formalized domain policies from *RSpecs database* and keeps an updated resources availability status of every domain from the *Resource Availability Module*. Using all the information, the *MCPS Broker* optimizes the application workflow to a domain and core assignment through *Global Scheduler* and *Local Scheduler* that run the global and local scheduling algorithms detailed in the previous sub-sections. Based on the results, workflows are distributed and scheduled by the *Reservation Module*. Note that the *Resource Manager* also maintains the *Monitoring & Analysis Module* for measurement, monitoring, and analysis of domain and workflow performance status. Lastly, the *Execution Controller* manages the overall middleware operation and message passing among the different middleware component modules.

5 EVALUATION

In this section, we describe the performance evaluation of our *MCPS Broker* implemented on the GENI Cloud infrastructure [14].

5.1 Testbed and Experiment setup

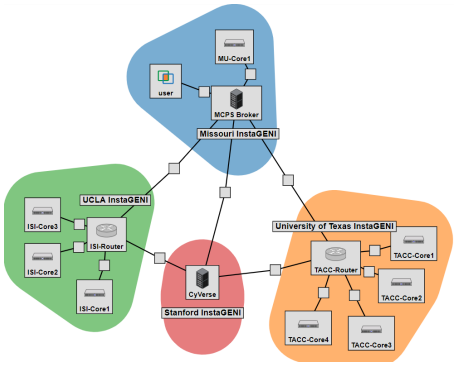


Figure 5: GENI testbed and experimental scenario setup.

Fig. 5 illustrates the testbed setup on GENI infrastructure that we use for our evaluations. We geographically distribute the multi-cloud resource domains approximately based on the real computing centers used for SoyKB workflows: MU domain is reserved from Missouri InstaGENI aggregate, TACC domain is reserved from University of Texas at Austin InstaGENI aggregate, ISI is is reserved from University of California Los Angeles InstaGENI aggregate, and CyVerse is reserved at Stanford University InstaGENI aggregate. For the testbed, MU domain has 1 core, ISI domain has 3 cores, and TACC domain has 4 cores; all cores have clock rate at 1.5 GHz. The network connectivity between MU and CyVerse domains have a bandwidth of 10 Mbps mimicking regular Internet speed, while all other connections are of bandwidth 100 Mbps mimicking actual Science DMZ based dedicated Layer 2 connectivity between domains. The compute capability and network bandwidth mismatches in terms of number of cores, their speed, and Mbps values are much more pronounced in reality; however, our values are based on GENI restrictions. The testbed also replicates security policies of TACC, ISI, and MU domains as well as dynamic resource utilization levels. For the experiments, workflows are sent from the MU domain users through the MCPS Broker (also located at Missouri InstaGENI), which decides whether the workflows are processed locally at MU or remotely at TACC or ISI based on the global and local algorithm outcomes discussed in Section 4. Regardless of wherever the workflows get processed, the results are ultimately sent to CyVerse afterwards for further post-processing, storage and bioinformatics community-wide sharing.

5.2 Performance evaluation

Fig. 6 shows the total workflow execution time comparison for RNA-Seq workflow processing of different data sizes between three possible workflow data processing and transfer scenarios: i) workflow being processed at MU and transferred to CyVerse, ii) workflow is transferred to TACC for processing from where it is transferred to CyVerse, and iii) workflow is transferred to ISI for processing from where it is transferred to CyVerse. This figure works as a baseline for evaluation as the total execution time (i.e., sum of all computation and transfer times) measurements are taken with no parallel jobs running at the candidate domains and no parallel transfers on the networks. As expected, the figure shows that the computation times at MU to be much higher than that in remote resources (i.e., TACC and ISI) due to their higher resource availability. Even with the data transfer time for remote resources added to the equation, it makes more sense to process data remotely than at MU. Furthermore, due to availability of dedicated Layer 2 network between remote resources and CyVerse unlike from MU, the total execution time difference is much more pronounced for the entire end-to-end workflow life-cycle for different data sizes (GENI allows a maximum of 1.5 GB data).

Fig. 7 compares the total transfer time outcomes between three resource brokering strategies: i) only security-driven brokering that does not consider performance optimization, ii) only performance-driven brokering that does not consider security compliance, and iii) our proposed MCPS brokering that optimizes performance and ensures security compliance. The figure shows results of 10 experiment runs with PGen workflow processing for different data sizes irrespective of the domain selections made by the schemes at

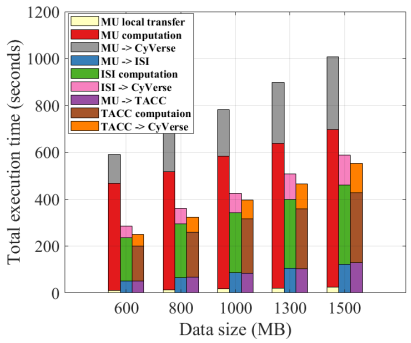


Figure 6: Total execution time baseline comparison for RNA-Seq workflow processing.

different runs. Also, during different runs, the resource availability status of the domains and network bandwidth are altered randomly. From Fig. 7 it is evident that for different data sizes, our scheme performs almost as good as only performance-driven brokering in terms of choosing domains for processing that optimize total execution time. The only security-driven brokering performs poorly as it always chooses ISI for processing irrespective of the ISI domain's resource availability for ISI being most secured (as shown in *RSpecs* comparison in Table 3). Whereas, MCPS scheme intelligently switches between ISI, TACC, and MU based on resource availability to improve performance.

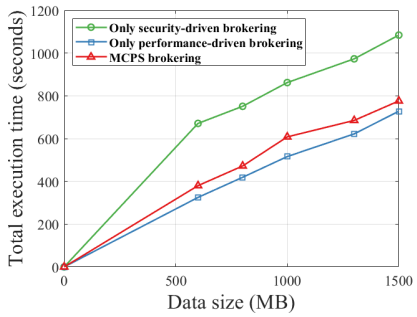


Figure 7: Total execution time comparison between different brokering schemes for PGen workflow processing.

5.3 Security evaluation

We evaluate the security compliance the three aforementioned schemes in terms of domain selection outcomes using NIST [40] based risk assessment method. The NIST method for conducting risk assessments is a widely accepted procedure to analyze the security compliance and dependability of a system. The risk assessment study allows us to compare the security compliance of only security-driven brokering, only performance-driven brokering and the proposed MCPS brokering for the SoyKB workflow processing. The risk calculation from a threat event using the NIST method is shown in Fig. 8, and involves the following steps:

- Assess the likelihood of threat occurrence on basis of probability of initiation and success.
- Assess the level of Impact in event of a successful attack.
- The Risk score is a combination of the likelihood and impact.

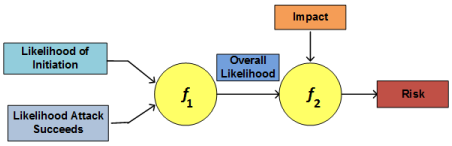


Figure 8: NIST based Risk assessment model from likelihood and threat impact factors.

We identified 5 possible threat events from the NIST guidelines of potentially 'High' to 'Moderate' security risks (based on 'Impact' values) to a candidate domain. The NIST definitions of such events along with SoyKB workflow relevance, and relative impact on the workflows are shown in Table 4. Then we assess the security compliance in terms of domain selection for processing against all 5 threats for all 3 resource brokering schemes. We use a predefined semi-quantitative scale of 0-10 as guided by NIST for the impact/likelihood event assessments, with 10 indicating *very high*, 8 indicating a *high*, 5 indicating a *moderate*, 2 indicating a *low*, and 0 indicating *very low* levels of impact. For the assessment, the three basic attack variables, e.g., Likelihood of Initiation (LoI), Likelihood of Success (LoS), and Impact of the attack are assigned values using the NIST guidelines. The final Risk value is calculated using the method illustrated in Fig. 8. In the figure, f_1 is defined as a $max()$ function with the likelihoods as arguments. Whereas, f_2 is an $\lceil avg() \rceil$ function between overall likelihood and impact. Any value upon calculation is rounded off to the nearest upper bound integer value. The rational behind such calculation is to get the most conservative estimate of the selected domains' security complaince.

No.	NIST Threat Events	SoyKB workflow relevance	Impact
I	Craft counterfeit certificates	Unauthorized access to multi-cloud domain	High (8)
II	Deliver targeted malware for data exfiltration	Perform illegal data transfer from domain to compromised site	High (8)
III	Perform network sniffing of exposed networks	Access data in transit to get a knowhow of soft-spots	Moderate (5)
IV	Conduct simple Denial of Service (DoS) attack	Domain resources made unavailable to legitimate users	Moderate (5)
V	Exploit physical access of authorized staff	Tailgate authorized users to gain access to domain resources	Moderate (5)

Table 4: NIST based threat events and showing the relevance to SoyKB workflows and their impacts

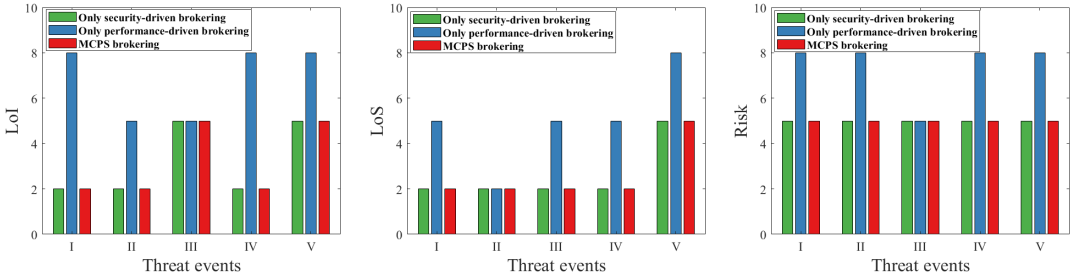
The security compliance comparison results are shown in Figs. 9(a), 9(b) and 9(c). The results are representations for both PGen and RNA-Seq workflows as both have the same *SSpecs* composition (see Tables 1 and 2). From the figures, it is evident that the likelihood of attack success and overall risk of different threats are similar for only security-driven brokering and our proposed MCPS brokering even when the risk values are the most conservative estimates as these schemes almost always choose ISI or TACC over MU regardless of the formers' resource availability. This is because the TACC and ISI have clearly laid out policies regarding protection against malware installation (Event II with High impact) with precautionary measures that makes it difficult for adversaries to initiate malware installation for data exfiltration resulting in their higher security standards (as shown in *RSpecs* comparison in Table 3). However, only performance-driven brokering sometimes chooses MU over ISI or TACC if MU has much higher resource availability in comparison to ISI or TACC, thus compromising security.

5.4 MCPS Broker Interfaces

Finally, we describe the MCPS Broker user interfaces we developed for the purposes of the testbed experiments and data collection. In the admin dashboard of MCPS broker, the administrator can monitor the resources available and the working status of each domain as well as the working status of each workflow (as shown in Fig. 10(a)). In the individual domains, Workflow ID is used as a way to differentiate the workflows sent from different users. The admin can view details of each domain resources or the details of the running workflows by selecting the relevant menu options. For example, if the admin selects the TACC option, the corresponding statistics page will be displayed. As can be seen in Fig. 10(b), each workflow stage is represented as a job in each individual domain. From this part of the user portal, the admin can also view what type of workflow of each job (PGen or RNA-Seq), the Workflow ID where it belongs to, its resources requirements, and its performance and security compliance status. The admin is further provided the option to view the detailed statistics of each workflow using the corresponding Workflow ID (as shown in Fig. 10(c)).

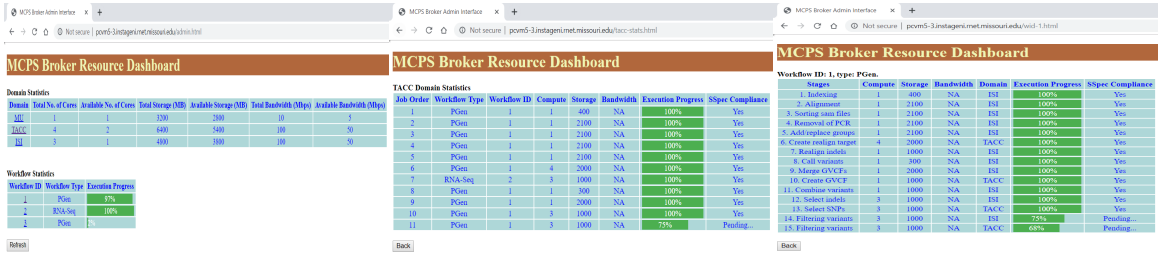
6 CONCLUSIONS AND FUTURE WORK

In this paper, we motivated the need for security aware resource brokering over traditional one-dimensional resource allocation for multi-cloud data collaboration. We demonstrated how our approach builds upon our earlier work in using formalized *QSpecs* and *SSpecs* of complex and simpler SoyKB workflows to design global and local scheduling algorithms. We showed how these algorithms used modified DAG scheduling heuristics to optimize workflow performance and ensure security compliance between workflow *SSpecs* and homogeneous domain *RSpecs* that are aligned across multi-cloud infrastructures. Our modeling and solution of the optimization problem is light-weight and achieves close to optimal allocation of federated resources across multi-cloud domains. Our implementation of MCPS Broker and case study evaluation with PGen and RNA-Seq workflows demonstrated the benefits of our proposed middleware in ensuring both performance optimization and security compliance. The results of this study inform and prove the counter-intuitive argument that it is beneficial for the data-intensive application users to scale out from local to remote for both performance and security optimization. Other data-intensive application communities (e.g., high-energy physics, astronomy sciences) can benefit from our middleware for resource provisioning, and augment their current techniques of manual co-ordination of policies. In future, we will implement the security aware brokering middleware services to MU ScienceDMZ for real SoyKB data processing case study evaluations and data collection that can be useful to fine-tune the proposed algorithms and services design.



(a) Likelihood of initiation comparison. (b) Likelihood of success comparison. (c) Overall risk comparison.

Figure 9: SoyKB workflow security compliance comparison between brokering schemes for different threat events.



(a) System administrator dashboard. (b) Domain (TACC) resource statistics. (c) Workflow (PGen) status check.

Figure 10: MCPS Broker graphical user interfaces.

REFERENCES

[1] R. Mount, D. Skinner, "Scientific collaborations for extreme-scale science workshop report," *US Department of Energy*, Tech. Rep, 2011.
[2] CyVerse - https://www.cyverse.org.
[3] XSEDE - https://www.xsede.org.
[4] I. Monga, E. Pouyoul, and C. Guok, "Software-defined networking for Big-Data science - Architectural models from campus to the WAN," *Proc. of IEEE SCC*, 2012.
[5] E. Dart, L. Rotman, B. Tierney, M. Hester, J. Zurawski, "The Science DMZ: A Network Design Pattern for Data-Intensive Science", *Proc. of IEEE/ACM Supercomputing*, 2013.
[6] M. Gaedke, J. Meinecke, and M. Nussbaumer, "A modeling approach to federated identity and access management," *Proc. of 14th International Conference on World Wide Web (WWW)*, 2005.
[7] T. Joshi, K. Patil, M. R. Fitzpatrick, L. D. Franklin, Q. Yao, J. R. Cook, Z. Wang, M. Libault, L. Brechenmacher, B. Valliyodan, X. Wu, J. Cheng, G. Stacey, H. T. Nguyen, D. Xu, "Soybean Knowledge Base (SoyKB): A Web Resource For Soybean Translational Genomics", *BMC Genomics*, Vol. 13, No. 1, S15, 2012.
[8] M. Dickinson, S. Debroy, P. Calyam, S. Valluripally, Y. Zhang, R. Bazan-Antequera, T. Joshi, T. White, and D. Xu, "Multi-cloud Performance and Security Driven Federated Workflow Management," *Proc. of IEEE Transactions on Cloud Computing (TCC)*, 2018.
[9] K. Kalari, A. Nair, J. Bhavsar, D. O'Brien, J. Davila, M. Bockol, J. Nie, X. Tang, S. Baheti, J. Doughty, S. Middha, H. Sicotte, A. Thompson, Y. Asmann, and J. Kocher, "MAP-RSeq: Mayo Analysis Pipeline for RNA sequencing," *Proc. of BMC Bioinformatics*, 2014.
[10] Y. Liu, S. Khan, J. Wang, M. Rynge, Y. Zhang, S. Zeng, S. Chen, J. Maldonado dos Santos, B. Valliyodan, P. Calyam, N. Merchant, H. Nguyen, D. Xu, and T. Joshi, "PGen: large-scale genomic variations analysis workflow and browser in SoyKB," *Proc. of the 13th Annual MCBIOS conference*, 2016.
[11] Texas Advanced Computing Center - https://www.tacc.utexas.edu.
[12] Information Science Institute - http://www.isi.edu.
[13] M. R. Garey and D. S. Johnson, "Computers and Intractability: A Guide to the Theory of NP-Completeness", *W. H. Freeman and Co.*, 1979.
[14] "NSF GENI Infrastructure" - *https://www.geni.net.*
[15] "Security and Privacy Controls for Federal Information Systems and Organizations", *NIST SP800-30 Technical Report*, 2013.
[16] T. Yu, Y. Zhang, K. Lin, "Efficient Algorithms for Web Services Selection with End-to-End QoS Constraints", *ACM Transactions on the Web*, 2017.
[17] R. B. Antequera, P. Calyam, S. Debroy, L. Cui, S. Seetharam, M. Dickinson, T. Joshi, D. Xu, and T. Beyene, "ADON: Application-Driven Overlay Network-as-a-Serviceor Data-Intensive Science," *IEEE Trans. on Cloud Computing*, 2016.
[18] W. Kim, P. Sharma, J. Lee, S. Banerjee, J. Tourrilhes, S-J. Lee, and P. Yalagandula, "Automated and Scalable QoS Control for Network Convergence," *Proc. of ACMINM/WREN*, 2010.
[19] G. C. Sih and E. A. Lee, "A compile-time scheduling heuristic for interconnection-constrained heterogeneous processor architecture", *Proc. of IEEE Transactions on Parallel and Distributed Systems*, 1993.
[20] M.-Y. Wu and D. D. Gajski, "A programming aid for message-passing systems", *Proc. IEEE Transactions on Parallel and Distributed Systems*, 1990.
[21] M. Iverson, F. Ozguner, and G. Follen, "Parallelizing existing applications in a distributed heterogeneous environment.", *Proc. of IEEE International Conference on Heterogeneous Computing Workshop*, 1995.
[22] A. Radulescu and A. J. C. van Gemun, "On the complexity of list scheduling algorithms for distributedmemory systems", *Proc. of ACM Int'l Conference on Supercomputing*, 1999.
[23] H. Topcuoglu, S. Hariri, and M.-Y. Wu, "Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing", *Proc. of IEEE Transactions on Parallel and Distributed Systems*, 2002.
[24] R. Ananthakrishnan, J. Bryan, K. Chard, I. Foster, T. Howe, M. Lidman, S. Tuecke, "Globus Nexus: An Identity, Profile, and Group Management Platform for Science Gateways and other Collaborative Science Applications", *Proc. of IEEE CLUSTER*, 2013.
[25] P. Calyam, A. Mishra, R. Bazan Antequera, D. Chemodanov, A. Berryman, K. Zhu, C. Abbott, M. Skubic, "Synchronous Big Data Analytics for Personalized and Remote Physical Therapy", *Elsevier Pervasive and Mobile Computing (PMC)*, 2015.
[26] Internet2 Incommon, https://www.incommon.org.
[27] OpenID, http://openid.net.
[28] X.509 Specification, http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0.pdf.
[29] B. Baker, K. Borne, T. Handley, J. Kantor, J. Hughes, R. Lambert, C. Lee, H. Larrieu, R. Plante, "LSST Data Management Cybersecurity Draft Plan", 2015.
[30] S. Debroy, P. Calyam, M. Nguyen, A. Stage, V. Georgiev, "Frequency-Minimal Moving Target Defense using Software-Defined Networking", *IEEE International Conf. on Computing, Networking and Communications (ICNC)*, 2016.
[31] Health Insurance Portability and Accountability Act (HIPAA) - https://www.hhs.gov/hippaa.
[32] H. Li and R. Durbin, "Fast and accurate short read alignment with Burrows-Wheeler transform", *Proc. Bioinformatics*, 2009.
[33] Picard - http://broadinstitute.github.io/picard/index.html.
[34] A. McKenna, M. Hanna, E. Banks, A. Sivachenko, K. Cibulskis, A. Kernytsky, K. Garimella, D. Altshuler, S. Gabriel, M. Daly, M. A. DePristo, "The Genome Analysis Toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data.", *Proc. Genome Research*, 2010.
[35] D. Kim, G. Pertea, C. Trapnell, H. Pimentel, R. Kelley, S. L. Salzberg, "TopHat2: accurate alignment of transcriptomes in the presence of insertions, deletions and gene fusions", *Proc. Genome Biology*, 2013.
[36] C. Trapnell, D. G. Hendrickson, M. Sauvageau, L. Goff, J. L. Rinn, and L. Pachter, "Differential analysis of gene regulation at transcript resolution with RNA-seq", *Proc. Nature Biotechnology*, 2013.
[37] E. Deelman, G. Singh, M. Su, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, G. Berriman, J. Good, A. Laity, J. Jacob, D. Katz, "Pegasus: A Framework for Mapping Complex Scientific Workflows onto Distributed Systems", *Scientific Programming*, 2005.
[38] Integrated Rule-Oriented Data System (iRODS) - http://irods.org.
[39] W. Pieters, T. Dimkov, D. Pavlovic, "Security Policy Alignment: A Formal Approach", *IEEE Systems Journal*, Vol. 7, No. 2, pp. 275-287, 2013.
[40] R. S. Ross, "Guide for Conducting Risk Assessments", *NIST SP800-30- Rev1 Technical Report*, 2012.