2021 IEEE International Conference on Microwaves, Antennas, Communications and Electronic Systems (COMCAS) | 978-1-6654-3557-4/21/$31.00 ©2021 IEEE | DOI: 10.1109/COMCAS52219.2021.9629107

Programmable Cyber-Infrastructure Architectures for Science Applications

Bijan Jabbari
ECE Department
George Mason University
Fairfax, VA 22030, USA
bjabbari@gmu.edu

Jerome Sobieski
ECE Department
George Mason University
Fairfax, VA 22030, USA
jsobiesk@gmu.edu

Ciprian Popoviciu
Tech Systems Department
East Carolina University
Greenville, NC 27858, USA
popoviciuc18@ecu.edu

***Abstract*—Cyber-Infrastructure (*CI*) is a critical element in achieving novel scientific discoveries and helping the research and development of new technologies. The next generation of research supporting CI should provide dynamic end-to-end platforms with guaranteed resource allocation and allow users to easily deploy the cyber-system elements and adapt them towards their current and future requirements. In this paper we consider a programmable cyber architecture which can implement this next generation of CI and discuss how the user requirements are met. This is based on a formal, comprehensive model for service abstraction and service construction – a framework and a generic virtualization model that offers a simplified common treatment of CI resources across all classes and services, in a scalable and secure architecture.**

***Index Terms*—Cyber Systems, Virtualization, Cloud, Fog, Edge and Multi-Access Computing, High Performance Networks. Quality Assurance, Network Programmability and Softwarization, Network Function Virtualization (*NFV*) and Orchestration.**

I. Introduction

Cyber-Infrastructure (CI) is becoming a critical enabler for the scientific and educational communities for a wide array of services and applications [1]–[4]. These communities need more than just a set of CI resources they must negotiate, assemble and manage in order to support their research. Research & Education communities need a comprehensive cyber-infrastructure service architecture allowing the network, the computational resources, the big data stores, large scale instruments and sensors, and shared spectrum to be seamlessly composed into deterministic, high-level custom solutions that fit their specific needs. They also need these resources online within short timelines and the ability to change them quickly as the experimental needs change. Research projects cannot afford being stuck between the long operational processes of production infrastructures and unreliable, ad-hoc environments.

The criticality of the network infrastructures dramatically reduces their flexibility to meet the needs of evolving research methodologies and all but eliminates their availability for exploring advanced, experimental services. It has become difficult to deliver customized services to increasingly sophisticated science teams, or to deploy advanced innovative services at scale without an expensive and time-consuming build-out of dedicated infrastructures for each new experimental idea. These constraints are exacerbated as we look beyond the network to the broad cyber-infrastructure which is still being engineered and managed in silos focused on particular services - network, or computational services, or storage facilities.

To circumvent the constraints of traditional production CIs, researchers resort to building their own environments. While these purpose-built infrastructures meet specific project needs, they are not always optimally designed, and they are not designed for sustainable use. These environments often exist in bubbles isolated from production, so they do not facilitate easy collaboration.

Different research projects traditionally emphasize certain elements of the CI individually. Data intensive research projects focus on high-capacity transport or high-capacity compute. Such projects can still benefit from shared infrastructure if the resources are available and are properly reserved. The next generation Cis, however, must support quite diverse research projects which will require flexible transport and distributed but lower capacity compute resources. Layered edge computing is just one example of potential services requiring large scale CIs with distributed compute resources.

To address this situation, we need a comprehensive architecture that supports service agility and allows mature, production-level services to coexist with highly experimental new concepts while efficiently sharing the underlying CI facilities. We need an architecture that allows us to deploy and qualify new, untested service models at scale while also allowing us to easily construct customized, production-quality global service environments for the research communities. All these apparently divergent service types and requirements can be delivered in slices of a shared cyber-infrastructure. While slicing at networking, compute or storage levels is not a novel concept, "slicing" at service level, involving virtualized resources across the stack has been explored only in data center environments. A more generalized architecture is needed to address the needs of science projects which span campuses, continents and the world at large, which involve multiple administrative domains. This new paradigm must provide isolation and insulation between these various slices, and it must allow/enforce explicit control of the interaction

The work is supported by the National Science Foundation grants NSF 2029221 and NSF 2029218.

between resources and services residing in potentially distinct administrative domains. Such a paradigm shift requires a model that is technology agnostic and infrastructure independent, a model that allows the network, the computational resources, the data stores, the sensors and instruments, the wired and wireless access layers, the core, the edge, to all function under a common, interoperable – orchestratable – framework that enables this broad range of CI resources to be seamlessly composed into incrementally more sophisticated solutions for science and education.

Virtualization is already being used in many CI domains to improve the utilization, elasticity and cost of infrastructure resources and services (e.g., cloud services, virtual circuits, VRFs, VLANs, etc.). Virtualization will continue to be foundational to the development of the envisioned CIs. Automation is also critical to powering these CIs. Automated services will significantly reduce the time it takes for research infrastructures to become operational; will deliver dynamic scalability and agile selection of resources while facilitating effective - and cost efficient - operations.

Lastly, it is important to architect the next generation CI with the instrumentation and management tools necessary to enable easy environment setup and environment modification while hiding the complexity of managing multiple abstraction layers. This intrinsically multi-tenant infrastructures will also require the integration of multiple support services which facilitate operations.

In this paper, we propose a formal, comprehensive model for service abstraction and service construction – a framework and a generic virtualization model that offers a simplified common treatment of CI resources across all classes and services, in a scalable and secure architecture.

II. PROPOSED SYSTEM MODEL

The proposed system model leverages the virtualization technologies within each major CI component: network, compute, and storage. However, it generalizes the concepts beyond the traditional domains. The model also builds on existing orchestration frameworks and standardized interfaces to manage both internal and external resources. The terminology related to these technologies and frameworks is used in this section.

In our work, virtualization is generalized across the cyber-infrastructure architecture. Our aim is to formalize the "meta-concepts" of virtual objects. We identify the common characteristics of all virtual objects and we define a set of operators for manipulating virtual objects through their lifecycle. How to create virtual objects, how to destroy/delete/release a virtual object when it is no longer required, how to query the state of a virtual object. The goal is to create an algebra – a set of virtualization "operators" and a well-defined set of virtual "operands", that will allow intelligent software agents to manage virtual objects generically – not just specific types of Virtual Machines (*VM*s) or Virtual Circuits (*VC*s) from a specific provider, or vendor – to be able to create virtual objects that behave repeatably and predictably regardless of which provider delivers the resource or regardless of how a provider chooses to instantiate that resource.

Foundationally, the Model views all CI as a set of virtual objects or "resources" and a set of connectors (Fig. 1). The way a virtual resource is implemented is not relevant as long as it meets the performance specifications in the resource request. Virtualization in this respect is not simply a software version of some piece of hardware or technology. Rather virtualization represents a set of formal properties that an object must possess in order to be part of a virtualization architecture.

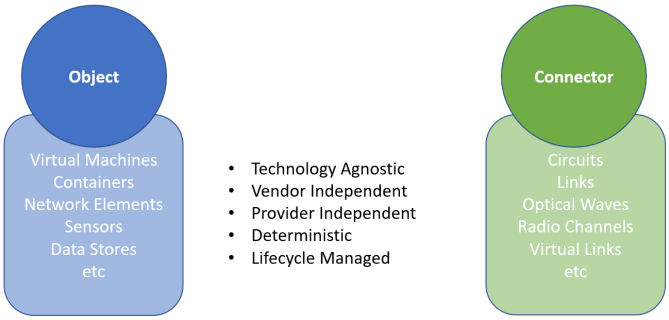

Fig. 1. The CI Virtualization Model.

Each virtual resource instance must be deterministic. Deterministic resources behave the same each time they are given the same initial conditions, a property enabling predictability and repeatability. It should be noted that virtualization itself does not impact performance. Virtualization *defines* the performance a priori and it becomes the responsibility of the implementation to deliver it. A virtual object not performing as specified reflects implementation, not specification.

Virtual resources are well bounded – that is, they are insulated and isolated from one another. A resource that is not well bounded can affect the behavior of other resources. Poorly bounded virtual objects can be a major security and stability threat to a CI. The properties of isolation and insulation enable resources to be very predictable in terms of their performance. It is important that the virtualization modules that map these resources to the infrastructure ensure that these performance guarantees are enforced across all resources.

A. Objects

All objects have a well-defined and deterministically managed lifecycle: They are first created and scheduled from available templates when the user issues a **Reserve()** request for some type of resource. The **Activate()** primitive provisions the resource and puts it in service to the associated project. Similarly, that resource instance can be taken out of service with a **Deactivate()** primitive. When the resource is no longer required, the **Release()** primitive returns the allocated object to the available resources pool. Finally, there is a **Query()** primitive that returns the state of a resource. These five primitives make up the working core of the CI object lifecyle as shown in Fig. 2.

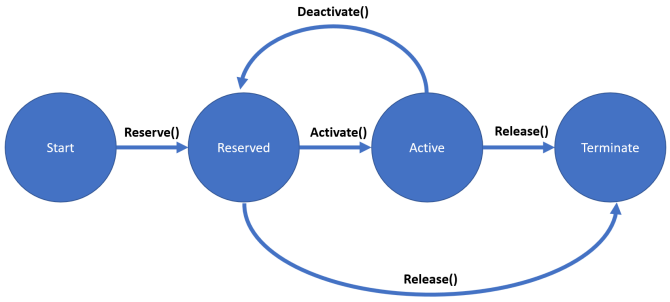

Fig. 2. The CI Object State Transition Diagram.

There are two fundamental types of virtual objects: an *atomic* object and a *composite* object. A composite object is an object that is hierarchically composed of other objects ("children" objects), and it is these children objects that define the parent's behavior. An atomic object, conversely, has no children. The behavior of an atomic class is defined and standardized by the user community, and it is up to each service provider to implement that class definition. This is done with software modules that implement the five lifecycle primitives (noted above) for that class. Composite classes derive their properties from the atomic objects they incorporate based on a template. This structure enables an easy representation of each composite object in a JSON format.

Class definitions consist of attributes (Type/Value Pairs) that act as parameters used in instantiating the object (i.e. a VM class may allow a $CPU = \{1, 2, 4\}$ attribute that specifies how many virtual CPUs (cores) a VM instance is to have).

B. Connectors/Adjacencies

Objects need a means of communicating with other objects. This is done by defining Ports for each object instance. For atomics, Ports allow data to ingress or egress the atomic object, and the lifecycle primitives configure these appropriately during the Reserve() and Activate() processes. In composite objects, however, the constituent children resources require a means of logically indicating which ports are connected – or are "adjacent" – to which other ports. Composite classes use **Adjacency** statements to describe how a port on one object is connected to a port on another object. The set of adjacencies describe the internal topology or data flow graph among a set of resources.

By grouping a set of resources (objects), and indicating their port adjacencies (connectors), one can identify new composites or entire functional services. Figure 3 describes the structure of a service interconnecting three sites.

The proposed formalization of resources might appear ambitious, however it is realistic at a time when network softwarization [5], [6] and generalized virtual functions are becoming mainstream. Any resource used in a scientific CI can be defined and managed within the proposed model regardless of underlying technology or technology implementation.

In the next section we map several science applications to the proposed conceptual model. We exemplify the benefits of generalized virtualization and the power of a fully programmable CI.

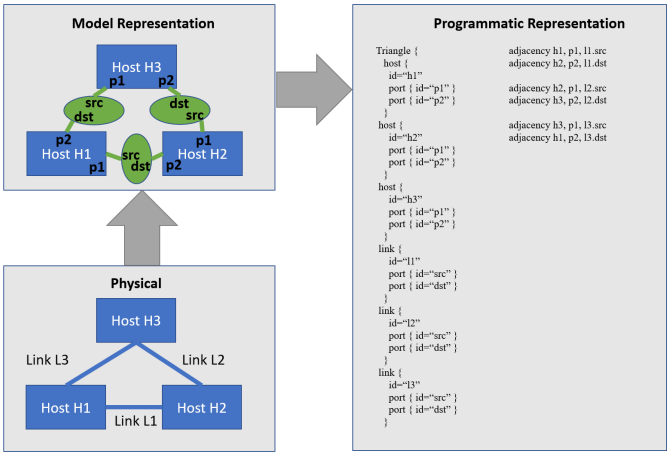

Fig. 3. Logical and Formal Definition of a Service Connecting Three Sites.

III. Illustrative Science Applications

Scientific research today is increasingly a distributed, collaborative undertaking. Science teams are often multi-national, their instruments are shared globally, and it is increasingly dependent upon large scale cyber-infrastructure that is itself globally distributed. For illustration, here we present three major science applications that make the case for the programmable CI architecture proposed in the paper. These science applications expressed direct interest in working with the authors of this paper towards the development and testing of the proposed architecture.

A. The Radio Astronomy Application

Radio Astronomy (*RA*) employs a process called very long base-line interferometry (*VLBI*) [2] that uses multiple radio telescopes from around the world to listen in unison to a single faint radio source often millions of lightyears away. These telescopes digitize and record the radio signals they receive and forward them to a computational facility for correlation and analysis to generate a highly accurate picture of the source object. The data has traditionally been recorded on tape and shipped to the correlator. More recently, the data was recorded to disk and transported over high-speed networks, a process that is still not executed in real time.

The international Square Kilometer Array (*SKA*) project is constructing the next generation of VLBI platform consisting of some 3000 radio telescopes distributed across southern Africa and Australia. These sensors generate terabits-per-second of data in many different wavelengths, and the instruments themselves can be part of many different observations simultaneously. To accomplish this there must be efficient and integrated scheduling and reservation processes, intermediate data pre-processing provisioning, network transport, correlator scheduling at possibly many different computational facilities,

and both intermediate and final data storage facilities – all at state-of-the-art performance.

The programmable CI can provide, on-demand, performance guaranteed resources needed by this science application. It can adjust allocations depending on dynamic needs or project re-configuration. It facilitates the operation of multiple experiments at the same time on a shared infrastructure without concerns about interference or security. Moreover, in the context of the generalized virtualization model, the sensors themselves can be mapped into atomic objects that can be orchestrated within the service. This will allow intelligent software to optimize sensor/instrument assignment and scheduling, real time analysis to verify observational parameters while the observation is taking place, acquisition of computational correlation resources for both real time and full-scale analysis, high performance storage, and bulk data transport capabilities across global high speed network links. The SKA project itself does not own all these resources – it utilizes shared infrastructure such as networks, computational facilities, and storage and archiving resources, and so the entire SKA facility must be integrated with and interwork with many other science programs that may have very little in common with radio astronomy. A rigorously virtualized service model for SKA and VLBI services would allow the science software to integrate these resources into a single virtual services environment programmable to individual observations, adaptable to changing conditions or experiment requirements, optimal in terms of operational readiness.

B. The Data Transfer Node

Science applications are increasingly data intensive. That is, the raw data collected at sensors and from instruments is growing at an exponential rate. This intermediate data is exploding as analysis and simulations increase their resolution, and the output products are increasingly interactive data sets intended to be explored through real time interactive visualization tools. The input data to these analysis workflows is often coming from diverse repositories that themselves are rarely found in one place. In a global science environment, the data and the computational and visualization facilities are not always collocated. Thus, large data sets must be transported from place to place for processing. With the existing internet architecture, and even with high-speed network links, such bulk data transfers must be optimized for each specific data transfer in order for such intermediate processing to be done as quickly as possible. Unpredictable network activity, physical distances between data repositories, storage interface performance, dependency scheduling, suboptimal data store data processing facility pairing, etc will affect this stage of the science workflow.

To address the needs of data intensive science applications the concept of Data Transport Nodes (*DTN*s) emerged in infrastructure design. DTNs are dedicated platforms that have one interface attached to a local high performance data storage array, and one interface attached to a high-performance network. DTNs combine hardware and software to evaluate the network between a data source and its destination and to configure themselves accordingly to optimize the transfer. DTNs typically interwork with other DTNs to have both the source and the sink coordinated in their expectations of the network and storage performance.

However, DTNs are still very specialized, very complex systems. They are still evolving, and still require expert support to design and operate. And so they are still very scarce. A virtualized, programmable CI would integrate a DTN in the architecture as a virtual object. Service providers, or collaborating user communities, could simply instantiate a DTN virtual class from a common library of virtual resources, where and when needed. These could be dedicated to particular science communities – say a high energy physics DTN at a campus data center, or they could be made available to a broader community of science users. A dedicated, virtual DTN would be able to still leverage all of the available hardware performance at the source or sink, and could be tuned to the users' data management requirements. DTNs instantiated within the programmable CI would also benefit from more accurate end-to-end view of allocated and available resources leading to improve data management and increased performance. In addition to shear performance optimization, such virtualized services in an application specific role could also enable enhanced or customized data protection and security profiles.

C. Virtualization for Remote Physical Infrastructure

Increasingly we see sensors and instruments – and people and autonomous agents – being deployed to highly remote locations such as space based platforms: Hubble, Webb, StarLink, ISS, etc. We already see interplanetary agents in terms of science platforms, and will likely see the establishment of Lunar and Martian expeditions and bases over the next two decades. Emerging services such as large scale LEOS constellations will enable high-speed network access to any location on Earth – enabling open ocean research, polar research, and both deep ocean and remote terrestrial data reachability. These facilities are not physically reachable and unlike some present, specialized and dedicated science platforms, they will require reconfiguration options.

In these environments, the programmable CI allows the platforms to be designed for generality and longevity, and a virtualization abstraction layer (software) will define more specialized service constructs. The integration of various types of resources within the programmable CI enables the configuration of more complex, more reliable experiments that would benefit from edge computing resources and alternate paths. This will allow the service layer to be hardware agnostic, and the underlying infrastructure to be designed for reliability and generality.

IV. Beyond simple virtualization

Virtualization for specific infrastructure elements or aspects of the CI has been considered for many years and has been applied towards resource optimization and agility for specific

domains. In the data center environments virtualization of compute, storage and network is successfully orchestrated to deliver cloud services. It would thus be easy to infer that the proposed model does not to go beyond the current state of the art. However, we believe that the organic development of virtualization concepts led to significant inconsistencies which constrain the development of large-scale CIs. There is not even a practical definition of what it means to be a "virtual" object. There is no common model for how one virtual object interoperates with other virtual objects. There is not even a common concept of how we create virtual objects or release them, or query them to determine their state. Again, while technology specific, vendor specific, application specific definitions do exit, consistent definition spanning domains are lacking.

We believe that the systematic approach proposed in this paper to generalizing the concept of virtualization and to the integration of a wide range of resources within the model using orchestration frameworks broader than a data center and a single admin domain is unique and it provides the path towards a programmable CI architecture in line with the technological and operational needs of modern science applications. The authors build upon work done by GEANT network and testbed services on the Generalized Virtualization [7], work led by one of the paper authors. This work requires significant further development in multiple areas such as: modeling and management of composable services, modeling and management of policies, standardizing multi-domain operation, modeling and integration of new object types and the development of a complete architecture for managing individual services and the system in its entirety.

We are using the cross Atlantic testbed BRIDGES [8] as a proof of concept for the proposed architecture. We envision collaborations with testbeds of various types in order to bring within the model a growing set of object and connector types. The science applications running on the BRIDGES testbed will benefit from the programmable CI architecture and provide opportunities to identify new objects, new composites and new services for the architecture.

In ultimate analysis, the proposed programmable CI architecture is normalizing many of the current major trends in infrastructure development such as NFV, SDN, Edge Computing, IOT, Cloud orchestration. This is not an ambitious undertaking but rather a natural outcome of a consistent approach to managing CI resources. This consistent approach combine with orchestration and policy management enables consistent, deterministic, rapid deployment of research environments that leverage existing resources. Two keys to the successful adoption of this architecture will be a good standardization of inter-domain management of services to facilitate integration and easy deployment of the control plane and easy integration of existing and new resources. Both of these areas are part of the future work pursued by the authors of the paper.

V. Conclusions

This paper presents a proposed approach to managing CI resources and an approach to bringing them together into flexible, agile, deterministic services that support modern science applications. The CI resources are mapped into generalized virtualization concepts and they are managed within a clearly defined lifecycle to create a virtualized, programmable cyber-infrastructure. This systematic approach to resource management, independent of technologies, providers or vendors goes beyond the traditional, siloed views on virtualization to create the foundation for more diverse services in support of science applications. The value of the proposed architecture was demonstrated in the context of three specific science applications, applications which explicitly expressed interest in the envisioned programmable CI. The proposed model builds on previous work which supports basic programmable testbeds and intends to leverage the new, NSF funded trans-Atlantic testbed BRIDGES to continue the development of the model. Future work will focus on the development of composite objects and services, on multi-domain operation, policies management and environment management.

References

[1] P. Freeman, D. Crawford, S. Kim, J. Munoz, "Cyberinfrastructure for Science and Engineering: Promises and Challenges," *Proceedings of the IEEE*, vol. 93, no. 3, pp 682-691, Mar. 2005.

[2] T. Lehman, J. Sobieski, B. Jabbari, "DRAGON: A Framework for Service Provisioning in Heterogeneous Grid Networks," *IEEE Communications Magazine*, vol. 44, no. 3, pp. 84-90, Mar. 2006.

[3] NSF Cyberinfrastructure Council, Cyber-Infrastructure Vision for 21st Century Discovery, National Science Foundation, Mar. 2007.

[4] I. Rodero, M. Parashar, "Data Cyberinfrastructure for End-to-End Science" *IEEE Computing in Science & Engineering*, vol. 22, no. 5, Sept.-Oct. 2019, pp. 60-71.

[5] Galis *et al.*, "Softwarization of Future Networks and Services - Programmable Enabled Networks as Next Generation Software Defined Networks," *2013 IEEE SDN for Future Networks and Services (SDN4FNS)*, Nov. 2013, pp. 1-7.

[6] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, and H. Flinck, "Network Slicing and Softwarization: A Survey on Principles, Enabling Technologies, and Solutions," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, Third Quarter 2018.

[7] S. Naegele-Jackson, J. Sobieski, J. Gutkowski, M. Hazlinsky, "Creating Automated Wide-Area Virtual Networks with GTS – Overview and Future Developments,," *IEEE International Conference on Cloud Computing Technology and Science (CloudCom 2016)*, Luxembourg, Dec. 2016, pp. 602-607.

[8] B. Jabbari, J. Sobieski, C. Popoviciu, "BRIDGES - Binding Research Infrastructures for the Deployment of Global Experimental Science," April 2020. [Online]: https://cnl.gmu.edu/bridges