CUSTOM TEMPLATES FOR HYBRID CLOUD RESOURCES
ORCHESTRATION AND USER WORKFLOWS

A Thesis
presented to
the Faculty of the Graduate School
at the University of Missouri

In Partial Fulfillment
of the Requirements for the Degree
Master of Science

by
Ronny Bazan Antequera
Dr. Prasad Calyam, Advisor

December 2014
The undersigned, appointed by the dean of the Graduate School, have examined the thesis entitled

CUSTOM TEMPLATES FOR HYBRID CLOUD RESOURCES
ORCHESTRATION OF USER WORKFLOWS

Presented by Ronny Bazan Antequera

A candidate for the degree of

Master of Science

And hereby certify that, in their opinion, it is worthy of acceptance.

________________________________________
Dr. Prasad Calyam

________________________________________
Dr. Marjorie Skubic

________________________________________
Dr. Gordon Springer
Dedicated to my mother Sonia Antequera, my father Renato Bazan, my brother Ruddy and the rest of my family.
ACKNOWLEDGEMENTS

I would like to thank Prof. Prasad Calyam immensely for being my advisor and mentor who brought out the best in me during my thesis research. I also thank Prof. Calyam, for giving me the opportunity to be part of the Virtualization Multimedia and Networking Lab and to be part of his numerous, challenging research projects in GENI, SDN, Network Virtualization and Hybrid Cloud Infrastructures. I am extremely grateful for his constant guidance, insight and encouragement through the entire course of my research. I would like to express my gratitude to Prof. Gordon Springer and Prof. Marjorie Skubic for their interest in my research and consenting to be part of my thesis committee.

Special thanks to Alex Berryman from Ohio Supercomputer Center/OARNet who gave me invaluable suggestions throughout many test bed setups.

I would also like to thanks to Fulbright since all of this invaluable accomplished would not be possible without their support.
# TABLE OF CONTENTS

Acknowledgements........................................................................................................ ii  
List of Figures..................................................................................................................... v  
List of Tables..................................................................................................................... vii  
Abstract................................................................................................................................ viii  
INTRODUCTION................................................................................................................ 1  
  1.1 Hybrid Cloud Applications......................................................................................... 1  
  1.2. Current infrastructure deployment process............................................................ 5  
    1.2.1 Amazon Machine Images (AMI) ................................................................. 6  
    1.2.2 GENI Request Specification (RSpecs) ....................................................... 6  
    1.2.3 VMware Virtual Appliance (vApp) ............................................................ 8  
    1.2.4 ThinApp......................................................................................................... 10  
    1.2.5 Docker.......................................................................................................... 12  
  1.3. Need for an infrastructure deployment automation process..................................... 13  
  1.4. Conclusion.............................................................................................................. 14  
  1.5. Further Research.................................................................................................... 15  

2. BACKGROUND AND LITERATURE SURVEY.................................................. 14  
  2.1 Hybrid Cloud Infrastructure...................................................................................... 14  
  2.2 Network Function Virtualization............................................................................. 16  
  2.3 Literature Survey..................................................................................................... 18  

3. CUSTOM TEMPLATES............................................................................................ 20  
  3.1 Methodology........................................................................................................... 20  
  3.2 Component Abstraction Model............................................................................... 21  
  3.3 Custom Template Catalog....................................................................................... 23
3.3.1 Collection
3.3.2 Composition
3.3.3 Consumption
3.4 Uses Cases
3.4.1 Simulation as a Service (SMaaS)
3.4.2 Physiotherapist as a Service (PTaaS)
3.5 Components Abstraction
3.5.1 Hybrid Cloud Infrastructure
3.5.2 Network Function Virtualization
3.5.3 Virtual Private Network
3.5.4 Physical VLANs
3.5.5 Physical routers implementation for L2
3.5.6 Performance Monitoring Networking
4. EVALUATION
4.1 LIDAR Analytics as a Service (LAaaS) Application
4.2 Custom Template Integration with LAaaS
4.3 Custom Template Evaluation
4.4 Result Analysis
5. FUTURE WORK
6. CONCLUSION
7. BIBLIOGRAPHY
LIST OF FIGURES

Figure 1.1. Traditional infrastructure deployment approach ........................................ 2
Figure 1.2. Novelty applications infrastructure requirements ................................. 3
Figure 1.3. Example of a Hybrid Cloud Infrastructure ........................................... 5
Figure 1.4. Amazon Machine Images – Life Cycle ................................................. 6
Figure 1.5. GENI Request Specification (RSpec) .................................................. 7
Figure 1.6. Experimenter using GENI AM API to reserve GENI resources .......... 8
Figure 1.7. VMware Appliance - Life Cycle ......................................................... 9
Figure 1.8. Horizon ThinApp service ................................................................. 11
Figure 1.9. Docker Common Runtime Environment ......................................... 13
Figure 2.1. Virtualization Hybrid cloud ............................................................... 15
Figure 2.2. Classic Networking vs NFV approach .............................................. 18
Figure 3.1. A macro operator represent the user requirements. ........................... 22
Figure 3.2. Custom Templates - Unit ................................................................. 23
Figure 3.3. Custom Templates Catalog – Life Cycle ........................................... 24
Figure 3.4. Flow Diagram - Collection ............................................................... 25
Figure 3.5. Flow Diagram - Composition ........................................................... 27
Figure 3.6. Flow Diagram – Consumption ......................................................... 28
Figure 3.7 SMaaS App Workflow ................................................................. 29
Figure 3.8. Hybrid Cloud Architecture ............................................................. 31
Figure 3.9. PTaaS Architecture ....................................................................... 33
**LIST OF TABLES**

Table 1. List of domain components with resources, example list……………… 22
Table 2 Template with only two requirements…………………………. 24
Table 3. Components in the same domain that match infrastructure
requirements………………………………………………………………… 27
Table 4. A need of cloud transformation for advanced manufacturing
companies………………………………………………………………………… 30
Table 5. Template for SMaaS App. ………………………………………… 32
Table 6. Networking Approaches for P2P connectivity requirement……… 35
Table 7. Template for PTaaS App. ………………………………………… 36
Table 8. NFV performance over share hardware………………………….. 39
Table 9. Template for LAaaS App. …………………………………………. 45
ABSTRACT

Traditional model of a central supercomputer resource serving a majority of on-campus researchers’ is being challenged more distinguished in interdisciplinary and data-intensive applications were infrastructure requirements are unique. In that sense numerous novelty hybrid cloud applications require to incorporate several computing and networking resources infrastructure “components”, as part of an integral deployment solution that solve specific research problems by the combination of new technology.

Several approaches aim to automate resources allocation processes, like Amazon Machine Images in Amazon Web Services, RSpecs in GENI and Virtual Appliances in VMware in order to describe and deploy computing and networking resources based on common models. However these solutions are isolated and work in different levels and environments, therefore there is lack of orchestration among them since the solution are not integrated.

In this thesis we study multiple real use case applications that have hybrid cloud requirements and find common components among those and abstract them to create maintain and update a Custom Template “CT” that is stored in a knowledge base for later use. So, new infrastructure deployment process time is reduced enormously and previous infrastructure would be scalable and reusable.
As a validation of our methodology implementation we were able to translate a Custom Template from the knowledge base to a new hybrid cloud application, demonstrating its functionality and importance for the new infrastructure deployment based on previous experience.
INTRODUCTION

1.1 Hybrid Cloud Applications.

Emerging interdisciplinary studies in data-intensive fields such as health, manufacturing and geo sciences are challenging virtualized computing and networking. They present unique requirements and demand use of advanced technologies/protocols to deploy resources in distributed environments traditional hybrid cloud application infrastructure deployment is manual and since requirements are not conventional, resource deployment is not straightforward especially when it comes to choose the correct infrastructure as shown in Figure 1.1. Basic steps consists in the abstraction of the hybrid cloud application requirements, analysis and design of the solution, a ‘cycle’ between partial infrastructure deployment and testing and finally infrastructure deployment.

The mentioned cycle is due to the integration of new emerging technology and techniques that need to be presented as a component for the hybrid cloud app. The goal of testing the application in different environments will help to identify bottleneck, thresholds and the most important, to compare performance among similar solutions in order to get the most adequate and efficient solution for the application.
Figure 1.1. Traditional infrastructure deployment approach

Hybrid cloud applications infrastructures are distinguished by the singularity of its resources, among them iterative/collaborative and multidisciplinary work, multi-site engineering experts and also technology integration such as network awareness Software Defined Monitoring (SDM), hybrid cloud architecture, Software Defined Networking (SDN), Network Function Virtualization (NFV), distributed resources, virtualization, Federated Identity, databases, monitoring, accounting, communication, external devices, cloud infrastructure, and so on as show in Figure 1.2
A hybrid cloud applications are based on the availability and implementation of certain technology that is crucial for its correct functionality, some of them are mentioned next:

- Monitoring, not only it is important to obtain information about memory and CPU utilization but also networking information such as bandwidth, throughput, delay, lost, and so on.
- High bandwidth, where hundreds of Mbps or even Gbps connection are required to keep alive an application.
- Computing, where dozens of CPU cores, hundreds of GB of memory and SSD storage capabilities is basic factor.
- Virtualization, such as computing virtualization where accessing to remote Virtual Desktops is required for an application
- NFV, where operational and capital expenditure benefits (OpEX and CapEX) are required.

An example of an application that requires to integrate many novel solution as part of the infrastructure we can mention SoyKB (Soybean Knowledge Base) [1] project that is a comprehensive all-inclusive web resource for soybean translational genomics and breeding. SoyKB handles the management and integration of soybean genomics and multi-omics data (including genomic sequence, microarray, RNA-seq, proteomics and metabolomics datasets). This represents rich and resourceful information, which can provide valuable insights. The application requires to add a new HPC infrastructure at University of Missouri besides: ISI (Information Science Institute), TACC (Texas Advanced Computing Center) and XSEDE that uses Pegasus workflow [2] system is to control data movements among execute compute nodes for the analysis of the millions or billions of DNA nucleotides sequenced in parallel. Data flow from MU need to be directed by Openflow Switches through Layer3 or prioritized Layer2. The iPlant resources at the U. of Arizona serve as SoyKB’s Data Store, and can be accessed using iRODS. Data can be replicated from U. of Arizona to the servers at other HPC sites over Internet2, which allows low latency data access when running the workflows on HPC resource. Compute data at MU side will require OpenStack [3] nodes and results need to be stored locally and remotely as shown in Figure 1.3.
1.2. Current infrastructure deployment process.

Provisioning and handle computing and networking infrastructure on-time involves the manual intervention and in many cases a lot of effort and time in order to abstract the application infrastructure requirements, design a solution and coordinating with several parties for its correct implementation, these process is necessary and will guarantee that the deployed infrastructure meets the application requirements.

In cloud computing world there are several approaches that aim to automate the deployment of computation and networking infrastructure, also we can distinguish that solution can be integrated to the application level and the infrastructure level, next we will describe some of the most important approaches.
1.2.1 Amazon Machine Images (AMI)

AMI is a special type of virtual appliance created by Amazon Web Services (AWS) [4] that is used to instantiate a virtual machine within the Amazon Elastic Compute Cloud (EC2). It serves as the basic unit of deployment for services delivered using EC2. The AMI Life Cycle is shown in Figure 1.4 where instances (virtual machines) can be created and stored as an images ready to be deployed in a matter of few clicks. This images can also work as a templates new images.

Figure 1.4. Amazon Machine Images – Life Cycle

1.2.2 GENI Request Specification (RSpecs)

Another approach is the RSpecs (Resource Specification) that is a common language for describing resources, resources request and reservation that GENI [5] uses as a standardized request specification document. Experimenters request resources from aggregates using a standard API called the GENI Aggregate Manager API or GENI AM API.
The AM API uses resource specification documents, commonly referred to as GENI RSpecs, to describe resources as shown in Figure 1.5. RSpecs are just XML documents in a prescribed format. Experimenters send to aggregates a request RSpec that describes the resources they want and get back from the aggregates a manifest RSpec that describes the resources they got, an explanation is published in [6]. The manifest includes information the experimenters will need to use these resources such as the names and IP addresses of compute resources (e.g. virtual machines), user accounts created on the resources and VLAN tags assigned to network links as show in Figure 1.6.

![Figure 1.5. GENI Request Specification (RSpec)](image)

The AM API allows experimenters to:

- List the resources available at an aggregate,
- Request specific resources from the aggregate be allocated to their slices,
- Find the status of resources from the aggregate that are allocated to their slices, and
- Delete resources from their slices.
1.2.3 VMware Virtual Appliance (vApp)

Another well known approach are the Virtual Appliances (vApps) from VMware. vApp templates that are based on OVF 1.0 (Open Virtualization Format). These templates can be retrieved from catalogs and transformed into virtual systems, called vApps, through a process called instantiation, which binds a template’s abstract resource requirements to resources available in a vDC (Provider).

A vApp contains one or more Vm elements, which represent individual virtual machines. It also contains information that defines operational details for the vApp and the virtual machines that it contains. The vApp lifecycle as shown in Figure 1.7 includes several distinct states:

- An OVF package, the form in which vApps are typically distributed.
- A vApp template, created when a client uploads an OVF package to a vDC.
- An undeployed vApp, created when a vApp template is instantiated without also being deployed, or a deployed vApp is undeployed.
• A deployed vApp, ready to be powered on and operated. Instantiation can include deployment, power-on, or both.

Figure 1.7. VMware Appliance – Life Cycle

Using OVF to distribute virtual machines has the following benefits:

• Ease of use. When users receive a package OVF format, they do not have to unzip files, execute binaries, or convert disk formats. Adding a vApp can be as simple as typing a URL and clicking Install.

• Virtual hardware Validation. OVF supports fast and robust hardware validation. It is not necessary to install a complete virtual machine before determining whenever it is compatible with the hypervisor host.
• Metadata inclusion. Additional metadata, such as an end-user license agreement, can be packaged with the OVF and displayed before installation.

• Optimized download from the Internet. Large virtual disk are compressed for fast download and to reduce disk space for large template libraries.

Even though the mentioned solutions present several advantages, they are isolated approaches that work at different levels and are unique for a specific environment therefore there is a lack of orchestration among them since they are not integrated.

1.2.4 ThinApp

VMware® ThinApp™ is an agentless application virtualization solution that decouples applications from their underlying operating systems to eliminate application conflict and streamline application delivery and management. ThinApp simplifies application virtualization and enables IT administrators to quickly deploy, efficiently manage, and upgrade applications without risk. With ThinApp, an entire Windows application and its settings can be packaged into a single executable and deployed to many different Windows operating systems without imposing additional cost and complexity to the server or client. Application virtualization with ThinApp eliminates conflicts at the application and operating system level and minimizes costly recoding and regression testing to speed application migration to Windows 7.

ThinApp virtualizes applications by encapsulating application files and registry settings into a single ThinApp package. IT administrators can deploy, manage, and update these ThinApp packages independently from the underlying operating system (OS). The virtualized applications do not make any changes to the underlying OS and behave the
same across different desktop configurations, which provides a stable, consistent end-user experience, and ease of management VMware® ThinApp® accelerates application deployment and simplifies application migration with agentless application virtualization as shown in Figure 1.8.

Virtualizing applications ensures faster software deployment with a more seamless end-user experience:

- **Full portability:** Virtualized applications can stream from any network share without a local client or a backend server.
- **Increased efficiency of application deployments:** Agentless virtual applications enable administrators to confidently deploy or de-commission applications on the fly with little or no regression testing, even for the most secure desktops.
- **No runtime conflicts:** Deploying virtual applications reduces lengthy QA and regression testing.
- Supportability: Single application packages can be supported by any Windows platform. Virtualized applications can run without requiring any modification of administrative security permissions, which protects the host operating system from possibly corruptive installation modifications.

1.2.5 Docker

Docker is an open platform for developers and system administrators to build, ship, and run distributed applications. Enables apps to be quickly assembled from components and eliminates the friction between development, QA, and production environments. As a result, IT can ship faster and run the same app, unchanged, on laptops, data center VMs, and any cloud. By “Dockerizing” the app platform and its dependencies, it is possible to abstract away differences in OS distributions and underlying infrastructure.

In virtual machines each virtualized application includes not only the application - which may be only 10s of MB - and the necessary binaries and libraries, but also an entire guest operating system - which may weigh 10s of GB. However the Docker Engine container comprises just the application and its dependencies. It runs as an isolated process in user space on the host operating system, sharing the kernel with other containers. Thus, it enjoys the resource isolation and allocation benefits of VMs but is much more portable and efficient. Docker replaces sandboxing with containerization as shown in Figure 1.9.
1.3. Need for an infrastructure deployment automation process.

Our work in this thesis aims to solve the provisioning computing and networking resources to different Hybrid Cloud Applications based on previous provisioning experience which could handle network and compute resource infrastructure on-demand.
2. BACKGROUND AND LITERATURE SURVEY

In this chapter, we first describe the concept of Hybrid Cloud Infrastructure and Network Function Virtualization as fundamental concepts for our thesis. We then present specific literature work conducted on abstraction methods across various domains.

2.1 Hybrid Cloud Infrastructure

Infrastructure architecture was monolithic, and each of these powerful machines could easily host 20 - 30 enterprise applications. This market was dominated by only a few hardware vendors, whose servers were expensive to purchase and maintain, took considerable time to install and upgrade, and in some cases were vulnerable to server outages that could last several hours until a vendor representative delivered proprietary replacement parts.

Virtualization was a major step towards cloud infrastructure as shown in Figure 2.1. Virtualized environments managed by internal system administrators and by default virtualization platforms do not provide the abstraction layer that enables cloud services. To cloud-enable an environment, a layer of abstraction and on-demand provisioning must be provided on top. This service layer is an important attribute of any cloud environment it hides the complexity of the infrastructure, and provides a cloud-management interface to users. Depending on the interface implementation, a cloud-management interface can be accessed through a management dashboard, REST or SOAP web services, programming APIs, or other services. For example, Amazon Web Services provides access through a management dashboard or REST/SOAP web services.
A public cloud is built over the Internet, which can be accessed by any user who has paid for the service. Public clouds are owned by service providers. They are accessed by subscription. Many companies have built public clouds, namely Amazon AWS, Microsoft Azure, IBM Blue Cloud, and Salesforce Force.com. These are commercial providers that offer a publicly accessible remote interface for creating and managing VM instances within their proprietary infrastructure. A public cloud delivers selected set of business processes. The application and infrastructure services are offered with quite flexible price per use basis.

The private cloud is built within the domain of an intranet owned by a single organization. Therefore, they are client owned and managed. Their access is limited to the owning clients and their partners. Their deployment was not meant to sell capacity over the Internet through publicly accessible interfaces. Private clouds give local users a flexible and agile private infrastructure to run service workloads within their administrative
domains. A private cloud is supposed to deliver more efficient and convenient cloud services. They may impact the cloud standardization, while retaining greater customization and organizational control.

2.2 Network Function Virtualization

NFV integration versus traditional networking implementation for handling unique applications requirements in diverse environments. Networking infrastructure deployment is very important since its implementation and functionality are more flexible compared with traditional networking solutions due to the fact that several networking enterprises implement network services in dedicated devices that leads to have different NOS (Network Operating System), therefore different brands requires distinct networking configuration skills and knowledge, scalability is tied to hardware traditional networking deployment, automated on-boarding, deployment and scaling of virtualized network functions cannot be implement.

In the past years system virtualization that creates many virtual systems within a single physical system; those systems are isolated operating environments that use virtual share resources, has taken the IT world and has been adopted efficiently and successfully in datacenter environments trough hypervisor technology which dramatically leverage scalability, performance and reliability of physical system transformed to Virtual Machines (VM’s). Similar to physical systems sitting in bare-metal hardware, networking hardware-based appliances inherited hardware deployment issues, falling into the same condition where specific NOS is required for specific hardware which has a negative impact in terms
of scalability, flexibility and cost, since factors as energy cost, capital investment and space to storage for those exclusive devices is becoming more and more difficult to overcome.

NFV implement software-based development by virtualizing networking functions and aims to address the hardware based problem by emphasizing virtualization technology and providing benefits such as reduced equipment power consumption for scalable solutions, detaching the NOS from the hardware, multi-version and multi-tenancy availability, targeted service introduction based on customer allocation and allows innovation to bring new services [7] as shown in Figure 2.2. NFV (Network Function Virtualization) aims to implement software-based development by virtualizing networking functions by bringing operational and capital expenditure (opex and capex) savings obtained from the implementation of general purposes hardware moreover by increasing automation through the implementation of operation and managements tools similar to those used by the information technology (IT) industry[8].

Routers have many networking functions such as IPv4/IPv6 routing, IP address management, Encapsulations, Firewall, Tunneling/VPN, Security, performance optimization, QoS policies, Administration and authentication, Diagnostics and logging and availability, our vRouters were configured with the minimal vNF required to have a stable and trustful communication hence the overhead for the application is reduced.
2.3 Literature Survey

In this cloud world, there are many approaches that try to solve the problem of automation deployment solution, that are based on previous deployed infrastructure, but those solution are not integrated among them, and there is no option for user to use different new technology and protocols based on previous solution. The need of a custom template that combine different resources and is used as a base to deploy new infrastructure for hybrid cloud application is a need. There are some work based on identification of components by using abstraction models in different contexts, in [9] the authors use component abstraction for reducing planning complexity by decomposing a problem to create operators by applying heuristics rules to planning at the component level, this work is focused in management and planning area. In [10] authors use abstraction for component based software architecture by estimating the assembly of reusable software components.
and making properties forecast to the associated architecture by using graph theory to
depict the software component-based architecture. In [11] authors use abstraction to check
large asynchronous designs by using modular model design based on the component
abstraction verification in isolation that eliminates the need of finding an accurate context
for verification.
3. CUSTOM TEMPLATES

In this chapter, we describe our solution approach by motivating the need of reusable templates for better infrastructure deployment performance. The first section presents the methodology implemented as a solution. We then describe two real hybrid cloud uses cases that were implemented without using our methodology. We also show the Life Cycle of a Custom Template and its phases, and finally we will abstract the template components for a new hybrid cloud application implementation based on the two application implemented previously.

3.1 Methodology.

The methodology for this thesis is to study multiple real use cases applications that have hybrid cloud requirements and find common components among those and abstract them to create a Custom Template “CT” create, maintain, update and store them in a knowledge base for later usage, in this sense new infrastructure deployment process time will be reduced enormously and previous infrastructure would be scalable and reusable.

Hybrid applications requirements are different for each project, however, a deployed infrastructure could meet other projects infrastructure needs by reusing certain if not all the components, hence this process could be automated by maintaining Custom Templates (CT) that are catalogs of application profiles handled by Performance Engineer [12] in where the reuse and scale of previous knowledge experience for infrastructure deployment is crucial in terms of efficiency.
3.2 Component Abstraction Model

Our model will help us to identify hybrid cloud applications infrastructure requirements based on user inputs that will be translated to a data structure that contain requirements that clearly identify domain, resources and preconditions organized in nodes. This model is composed by four steps:

A. Requirements. The model receive the user requirements based on a survey, a table is generated with different preconditions and variables in terms of Infrastructure requirement.

B. Statics facts identification. In this step key words need to be identified by comparing them with a domain base as an initial state of the problem.

C. Build abstract component. Based on static facts identification and in a domain selection, specific resources will be determined based on workflow specifications, preconditions and expectations related to application performance, quality of user experience (QoE), and data security. Each specific resource will be translated in a list node with the following data structure:

- Set ‘D’ of Domain (Static Facts)
- Set ‘R’ of resources (Infrastructure)
- Set ‘P’ of preconditions or variables (user QoE, performance, security)
- Set ‘A’ of add features (additional capabilities)
- Set ‘U’ off delete features (unnecessary capabilities)

Where domain could be one of the components described in Table 1 and components identification is done by:
\[ o = \text{part of the domain Operator Abstraction} \]

\[ o = (R(o), P(o), A(o), U(o)) \]

<table>
<thead>
<tr>
<th>Domain</th>
<th>Abbreviation</th>
<th>Resources</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network Connectivity</td>
<td>NC</td>
<td>NFV, SDN, VPN, …</td>
</tr>
<tr>
<td>Storage</td>
<td>ST</td>
<td>Local, cloud, redundant</td>
</tr>
<tr>
<td>End-to-end Virtualization</td>
<td>EE</td>
<td>Data Center Virtualization</td>
</tr>
<tr>
<td>Remote Computing</td>
<td>RC</td>
<td>Remote computing access</td>
</tr>
<tr>
<td>Monitoring</td>
<td>MN</td>
<td>N/W and System utilization</td>
</tr>
<tr>
<td>Local Computing</td>
<td>LC</td>
<td>Local computing access</td>
</tr>
<tr>
<td>Federated Identity &amp; Access Mgmt.</td>
<td>F1</td>
<td>SSO</td>
</tr>
</tbody>
</table>

Table 1. List of domain components with resources, example list.

D. Macro operator. Linked list built based on all structured nodes that represent resources requirements. An example is shown in Figure 3.1.

![Diagram](image)

Figure 3.1. A macro operator represent the user requirements.

The output of this model is a Macro Operator that it is the representation of infrastructure requirements translated to a structured linked list that will allow us to create and identify templates, as will be described in the next section.
3.3. Custom Template Catalog.

Due to the singularity of research’s applications and to the integration of new technology, the infrastructure deployment may not have only one solution which means that infrastructure could be implemented successfully with more than one component. So, researchers may want to apply different components to create different environments in order to collect data and compare performance with different technology. This process will help researchers to select an optimal solution. For better visual understanding we will represent a Custom Template as a collection components (computing and networking resources) as shown in Figure 3.2.

![Custom Template - Unit](image)

**Figure 3.2. Custom Template - Unit**

A custom template is build based on Macro Operators and it is the collection of hybrid cloud components that provision all the infrastructure that a Hybrid Cloud Application requires as show in Table 2, also a custom template catalog is the collection of several custom templates.
<table>
<thead>
<tr>
<th>Domain</th>
<th>Precondition</th>
<th>Features</th>
<th>Constraints</th>
<th>App</th>
</tr>
</thead>
<tbody>
<tr>
<td>Storage</td>
<td>Cloud Storage</td>
<td>RAID5</td>
<td>1TB</td>
<td>App1</td>
</tr>
<tr>
<td>Network connectivity</td>
<td>High Bandwidth &amp; Low Latency</td>
<td>Physical Layer2 Encryption</td>
<td>200Mbps</td>
<td>App1</td>
</tr>
</tbody>
</table>

Table 2. Template with only two requirements.

Figure 3.3, show the life cycle of a custom template from its creation to its deployment. Note that the Knowledge Base is the collection of Custom Templates that initially will be empty. The details of this process is as divided is described in Collection, Composition and Consumption phases.

![Figure 3.3. Custom Templates Catalog– Life Cycle](image)

3.3.1 Collection

In this section, researchers provide the hybrid cloud application requirements as an input that is compared with the “Knowledge Base” previous infrastructure solution are
packaged and stored in the Knowledge base ready to be used in the form of Template that is the collection of several computing and networking resources that an application could use as a solution.

If a similar template is found, it is customized according to the initial requirements and presented as a solution. If not a collection of computing and networking resources related with the project are packaged to create a new template that is presented as a solution, also the new template will be stored in the Knowledge Base for later usage as shown in Figure 3.4

![Figure 3.4. Flow Diagram - Collection](image)
Decisions upon the reuse of previous templates are based on the comparison between new resources needed and the resources that a specific template could provide in order to find a template that match the requirements, if the number of matches are greater than 50% a template can be presented as a solution for its customization, if not a new template will be created including all possible infrastructure pieces that the application could need based on the application topology abstraction and, once the new template is created it will be storage in the knowledge base for future reuse.

The output of this section is a template that contains not only required resources but also resources that could be applied to the application as solution for different scenarios or with different performance. At this point a test bed setup will be setup ready to be tested in the next phase.

3.3.2 Composition

The generated template contains many pieces that need to be integrated and orchestrated, this process is done by applying, testing and comparing solutions (in the case that the template contains more than a solution is available) to get the best performance for the application. The output of each comparison will have data collected that will help the application to get the best performance solution based on the required needs as well generated data will be presented to the researchers as shown in Figure 3.5.
Figure 3.5. Flow Diagram – Composition

The main purpose of this phase is the performance components comparison that is data collected based on the different components orchestration, especially when more than one component from the same domain and same precondition are candidates as a solution of the template, an example is shown in Table 3.

<table>
<thead>
<tr>
<th>Domain</th>
<th>Precondition</th>
<th>Features</th>
<th>Constraint</th>
<th>App</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remote Computing</td>
<td>Windows Server</td>
<td>Virtual Server</td>
<td>6vCPU, 16GB</td>
<td>App1</td>
</tr>
<tr>
<td>Remote Computing</td>
<td>Linux Server</td>
<td>Virtual Server</td>
<td>6vCPU, 16GB</td>
<td>App1</td>
</tr>
<tr>
<td>Remote Computing</td>
<td>Linux Server</td>
<td>Dedicated Server</td>
<td>2x2.6, 20cores</td>
<td>App1</td>
</tr>
</tbody>
</table>

Table 3. Components in the same domain that match infrastructure requirements.
3.3.3 Consumption

Finally, only one customized template will be consolidated and presented as the most efficient solution for an application, all the computing and networking infrastructure will be deployed based on that template. Figure 3.6.

![Flow Diagram – Consumption](image)

**Figure 3.6. Flow Diagram – Consumption**

This final deployment will allow us to:

- Reuse of previous deployed infrastructure
- Reduce time-to-instantiation
- Help to select the best performance solution based on different options
- Extend and Adapt previous infrastructure deployed
- Improved quality of service (QoS)
3.4. Uses Cases.

3.4.1. Simulation as a Service (SMaaS)

**Project Description.** Small businesses in manufacturing industry need to invest in simulation & modeling tools and the infrastructure that is required to run them. Such an investment would help reduce the product design cycles but bears a very high cost as show in Figure 3.7. There is a need for adoption of cloud-based technologies for their workflows with data-intensive computation and networking which would cut down costs. An environment is required that leverages cloud and facilitates these manufacturing workflows.

![Figure 3.7. SMaaS App Workflow](image)

**Application Description.** Advances in the field of cloud computing and networking have led to rapid development and market growth in areas such as online retail, gaming and healthcare. In the field of advanced manufacturing
however, the impact has been significantly lesser than expected due to limitations in cloud platforms for fostering community engagement. To address this problem, we study a new cloud-based architecture that provides Platform-as-a-Service (PaaS) management capabilities to the manufacturing community for delivering Software-as-a-Service (SaaS) “Apps” to their customers that will change the current status of how advanced manufacturing companies are working to a new environment architecture as shown in Table 4. The architecture aims at supporting an “App Marketplace” that thrives on agile development, organic collaboration and scalable sales of next generation Manufacturing Apps requiring high-performance simulation and modeling.

<table>
<thead>
<tr>
<th>Today: Local/home-grown</th>
<th>Future: Cloud/apps</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Screen capture/Email</td>
<td>• WebEx on virtual desktop</td>
</tr>
<tr>
<td>• Asynchronous collaboration, physical Win7 PC access required</td>
<td>• Synchronous collaboration, Cross-platform/device operation</td>
</tr>
<tr>
<td>• Lots of data copies</td>
<td>• Cloud storage</td>
</tr>
<tr>
<td>• Mail DVDs, SCP</td>
<td>• Mass storage, Box.net</td>
</tr>
<tr>
<td>• Slow data transfer</td>
<td>• Data security and control</td>
</tr>
<tr>
<td>• Public Internet</td>
<td>• Fast data transfer</td>
</tr>
<tr>
<td>• Local cluster procurement</td>
<td>• Extended VLANs, GENI</td>
</tr>
<tr>
<td>• Time consuming, overprovision</td>
<td>• Elastic compute access</td>
</tr>
<tr>
<td></td>
<td>• Can handle demand bursts</td>
</tr>
</tbody>
</table>

**Traditional development lifecycle: several delays for delivery, increased cost, and overhead for productivity**

**Agile development lifecycle: quicker delivery, reduced cost and increased productivity**

**Table 4. A need of cloud transformation for advanced manufacturing companies**
**Problem description.** Advanced Manufacturing design today requires iterative/collaborative work among multi-site engineering experts in e.g., fluid/thermal analyses.

Need to enable small businesses to easily adopt cloud-based technologies for their workflows with data-intensive computation and networking. National Center for Manufacturing Science (NCMS) report suggests that access to technologies can reduce product design cycles by 66%.

Advanced Manufacturing ‘Apps’ marketplaces as shown in Figure 3.8 are emerging that allow small businesses to provide expertise-driven modeling & simulation web services (SMaaS) tailored to their customer needs based on:

- PaaS with elastic HPC back-ends
- Cloud networking

![Figure 3.8. Hybrid Cloud Architecture](image-url)
The SoyKB template built based on our component abstraction model is shown in Table 5.

<table>
<thead>
<tr>
<th>Domain</th>
<th>Precondition</th>
<th>Features</th>
<th>Constrains</th>
<th>App</th>
</tr>
</thead>
<tbody>
<tr>
<td>End-to-end Virtualization</td>
<td>DB Virtual Servers, Storage</td>
<td>Cloud Infrastructure</td>
<td>2TBHDD storage</td>
<td>PTaaS</td>
</tr>
<tr>
<td>Network connectivity</td>
<td>High Bandwidth &amp; Low Latency</td>
<td>Physical Layer2</td>
<td>200Mbps</td>
<td>PTaaS</td>
</tr>
<tr>
<td>Network connectivity</td>
<td>High Bandwidth &amp; Low Latency</td>
<td>vRouter in the same system</td>
<td>45Mbps</td>
<td>PTaaS</td>
</tr>
<tr>
<td>Network connectivity</td>
<td>High Bandwidth &amp; Low Latency</td>
<td>vRouter in the dedicated HW</td>
<td>150Mbps</td>
<td>PTaaS</td>
</tr>
<tr>
<td>Remote Computing</td>
<td>Windows 2008R2</td>
<td>Virtual Servers</td>
<td>4vCPU, 4GB RAM</td>
<td>PTaaS</td>
</tr>
<tr>
<td>Remote Computing</td>
<td>Windows 2008R2</td>
<td>Virtual Servers</td>
<td>4vCPU, 6GB RAM</td>
<td>PTaaS</td>
</tr>
<tr>
<td>Remote Computing</td>
<td>Windows 2008R2</td>
<td>Virtual Servers</td>
<td>6vCPU, 12GB RAM</td>
<td>PTaaS</td>
</tr>
<tr>
<td>Monitoring</td>
<td>Networking Monitoring</td>
<td>Bandwidth consumption</td>
<td>2 scripts need to be executed</td>
<td>PTaaS</td>
</tr>
</tbody>
</table>

Table 5. Template for SMaaS App.

3.4.2. Physiotherapist as a Service (PTaaS)

**Project Description.** This application is a two way high-definition video conferencing displaying implemented in standard desktop systems, that streams synchronously data generated by Kinect sensors through Google Fiber for connecting Patients at Kansas City and Medical Expert at Missouri University (MU). Patient information such as sway and joint alignment parameters are calculated to understand how good/bad the patients are in their postural balance while doing different physical therapy exercises suggested by the Physical therapist as shown in Figure 3.9.
Figure 3.9. PTaaS Architecture

Application Description. The application is developed in C#, it’s a WPF (Client) using .Net 4.5 and Visual Studio 2012. Also, Microsoft Kinect SDK 1.8 is used to get RGB, Depth and Skeletal and Audio data from Kinect through the ports 4530, 4531, 4532, 4533 respectively that are allowed from Kinect by a C# console service application. Algorithms calculate the bending/sway parameters from the 3D joint positions of Kinect Skeletal tracking system. Features such as voice commands, sway detection, and analysis of walking pattern (gait) and other exercise related metrics were added to the application.

Problem description. Patients and therapist systems generated 300+Mbps as shown in Figure 3.10 and require to have a direct channel of communication as they
were physically connected in the same network over a high bandwidth networking infrastructure as GoogleFiber [13]. Per-to-per (P2P) communication is suitable for this configuration since a therapist can work with only one patient at a time.

**Figure 3.10. Data generated in ideal conditions.**

The therapist system is provisioning with a public IP, in contrast patients systems get private only private IP’s that are not reachable from outside the local network where the system is located.

Huge amount of data is generated in both sides and need to be storage for post-analysis, among the data we can mention the actual live video-conferencing data, patient video that is recorded by the therapist, and networking communication information.
The GENI rack accomplish 3 functions: keep tracking of the network health among users (elders) and therapist through a software-defined measurement and performance monitoring framework (Narada metrics), to keep data of users into a DB (such as sway, gait and other physical activity data of the elders) for later analysis by a therapist, and to storage kinect videos from which you can extract more data.

This application infrastructure presented many challenges such as having a P2P connection between Patient and Therapist system, were several networking connection approaches where tested including NFV over share and dedicated hardware as shown in Table 6.

<table>
<thead>
<tr>
<th>Approach</th>
<th>Details</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>VPN</td>
<td>Our campus security policies restricted the bandwidth to 10Mbps and 20Mbps were allowed after some configuration.</td>
<td>Due to the limited bandwidth this option was discarded.</td>
</tr>
<tr>
<td>VLAN</td>
<td>GENI allows to use VLAN infrastructure. We placed two system located in two different places. We got a bandwidth of 650Mbps.</td>
<td>Since integration of this solution with Google fiber was not feasible due to physical limitations, this option was discarded.</td>
</tr>
<tr>
<td>Physical Routers</td>
<td>A layer 2 connection was established through 2 physical routers in a lab environment.</td>
<td>Besides the default constrains that hardware-based routers represent over vRouters, these solution was not completely feasible since the devices need to have static public IPs in both sides.</td>
</tr>
<tr>
<td>NFV</td>
<td>Case1: vRouter was installed and configured over Virtual Box, in the same systems were the PTaaS application was installed</td>
<td>Due to the overhead that represent having both solution in the same physical system, we got about 75% packet lost, only 45Mbps transmitted. Hence this option was discarded.</td>
</tr>
<tr>
<td></td>
<td>Case2: vRouter was installed in a dedicated hardware over an hypervisor</td>
<td>Overhead was reduced and 150Mbps was obtained. Also, operational and capital expenditure (CapEx and OpEx) were reduced similar to the NFV Case1.</td>
</tr>
</tbody>
</table>

Table 6. Networking Approaches for P2P connectivity requirement.
The PTaaS template built based on our component abstraction model is shown in Table 7.

<table>
<thead>
<tr>
<th>Domain</th>
<th>Precondition</th>
<th>Features</th>
<th>Constrains</th>
<th>App</th>
</tr>
</thead>
<tbody>
<tr>
<td>End-to-end</td>
<td>DB Virtual Servers, Storage</td>
<td>Cloud Infrastructure</td>
<td>2TBHDD storage</td>
<td>PTaaS</td>
</tr>
<tr>
<td>Virtualization</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Network connectivity</td>
<td>High Bandwidth &amp; Low Latency</td>
<td>Physical Layer2</td>
<td>200Mbps</td>
<td>PTaaS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>vRouter in the same system</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Network connectivity</td>
<td>High Bandwidth &amp; Low Latency</td>
<td>vRouter in the dedicated HW</td>
<td>150Mbps</td>
<td>PTaaS</td>
</tr>
<tr>
<td>Network connectivity</td>
<td>High Bandwidth &amp; Low Latency</td>
<td>vRouter in the dedicated HW</td>
<td>150Mbps</td>
<td>PTaaS</td>
</tr>
<tr>
<td>Remote Computing</td>
<td>Windows 2008R2</td>
<td>Virtual Servers</td>
<td>4vCPU, 4GB RAM</td>
<td>PTaaS</td>
</tr>
<tr>
<td>Remote Computing</td>
<td>Windows 2008R2</td>
<td>Virtual Servers</td>
<td>6vCPU, 6GB RAM</td>
<td>PTaaS</td>
</tr>
<tr>
<td>Remote Computing</td>
<td>Windows 2008R2</td>
<td>Virtual Servers</td>
<td>6vCPU, 12GB RAM</td>
<td>PTaaS</td>
</tr>
<tr>
<td>Monitoring</td>
<td>Networking Monitoring</td>
<td>Bandwidth consumption</td>
<td>2 scripts need to be executed</td>
<td>PTaaS</td>
</tr>
</tbody>
</table>

Table 7. Template for PTaaS App.

3.5 Components Abstraction

A lot of work have been done in order to generate necessary components to construct Templates based on different hybrid cloud application implementation, note that some research application needs to be deployed in a testbed environment first before to be implemented in the actual environment due to many factors such as collaborations projects in where one of the ends is not ready for the application deployment, applications that need to be tested before the specific infrastructure for the application is deployed, and to test capabilities in order to understand functionality. For that process a collection of testbeds are also needed to be deployed and be ready for the Template implementation.
In the next section we will describe the different components that were abstracted from several hybrid cloud applications.

### 3.5.1 Hybrid Cloud Infrastructure

The best infrastructure for a hybrid cloud app requires both public cloud and dedicated environments. Cloud infrastructures were deployed in different physical locations and with different capabilities. Two of them are deployed in GENI rack at Missouri and Ohio and another in Missouri University. The clouds were established with VMware Horizon View that delivers virtualized and remote desktops as well applications through a single platform and supports end users with access to all their resources through a single workspace as shown in Figure 3.11.

![VMware Horizon Cloud Infrastructure](image)

*Figure 3.11. VMware Horizon Cloud Infrastructure*
3.5.2 Network Function Virtualization

Routers have many networking functions such as IPv4/IPv6 routing, IP address management, Encapsulations, Firewall, Tunneling/VPN, Security, performance optimization, QoS policies, Administration and authentication, Diagnostics and logging and availability, our vRouters were configured with the minimal vNF required to have a stable and trustful communication hence the overhead for the application is reduced as is shown in Figure 3.12.

![Figure 3.12. NFV implementation](image)

Figure 3.12. NFV implementation
A Vyatta 5400 vRouter is deployed over Virtual Box over non-dedicated hardware that allows L2 encrypted communication between the systems using NFV. The same output is achieved by implementing the vRouter over dedicated hardware to get better performance, since the first approach has some bandwidth threshold due to the share environment as show in Table 8.

<table>
<thead>
<tr>
<th>Activity</th>
<th>Bandwidth</th>
<th>Jitter ms</th>
<th>Loss %</th>
<th>RTT ms</th>
<th>Download rate Kbps</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Throughput bits/s</td>
<td>Transfer bits/s</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Average</td>
<td>391395.50</td>
<td>34325.00</td>
<td>14.53</td>
<td>73.84</td>
<td>30.67</td>
</tr>
</tbody>
</table>

Table 8. NFV performance over share hardware

3.5.3 Virtual Private Network

Two components were abstracted from VPN infrastructure. The first one by using Mizzou campus VPN connection that gave us up to 10 Mbps and later, after provide networking information to campus networking area, we got 20 Mbps, which is its maximum capacity due to internal firewall policies.

However some hybrid cloud application requires up to 20x times of bandwidth allowed. So, a software VPN solution was established in in a high bandwidth connection in our Research Network path (RNET) that has a threshold of 1Gbps.

3.5.4 Physical VLANs

The GENI mesoscale has a set of VLANs that are advertised at GENI rack aggregates as exclusive OpenFlow VLANs as shown in Figure 3.13.
These VLANs can be reserved in the GENI mesoscale in a way that the VLAN will be exclusive to a given experiment as shown in Figure 3.14 where two campuses are connected over two different VLANs.
3.5.5 Physical routers implementation for L2

This infrastructure was based on the needs of P2P connectivity requirement were two systems need to be accessible by using private IPs. Two CISCO 3925 series devices were implemented in order to create an Overlay Network over a wide area network that provides communication between the endpoints as they were being connect by logical link as is shown in Figure 3.15.
Since the whole functionality of the application lays down on the networking health, a detailed, timely and accurate information about the network path between patients and therapist is significant important in order to get performance measurement in IP networks, moreover having this metrics information without having to deal with many uncorrelated tools that are focused on certain metrics and deploying this functionalities in multi-domain environments is essential. End-to-end performance monitoring services needs to be
deployed in the network infrastructure that enables users to performing various custom metrics integration at system, network and application-levels into relevant measurement archives, and obtain timely and accurate performance intelligence (e.g., log analysis and anomaly event notifications).

Narada Metrics is a software-defined measurement and performance monitoring framework. The framework consists of a Central Intelligence System (CIS) and a number of Measurement Point Appliances (MPAs). Users can choose metrics from a comprehensive catalog and view results on the Narada Metrics dashboard. Results are displayed in form of a grid and dynamically populated graphs as shown in Figure 3.16.
4. EVALUATION

4.1. LIDAR Analytics as a Service (LAaaS)

The LAaaS application can be implemented in several different circumstances in where many videos may be collected by civilians and surveillance cameras that can be extremely useful for first responders trying to get valuable information from the input. However, watching and analyzing numerous videos on separate screens can be a cumbersome task. Registering a set of 2D videos with a 3D model can provide an intuitive venue for viewing multiple videos simultaneously and determining the physical relationships between cameras. In such a setup, it is likely that the user will want to work with the dynamic 3D environment from a remote location, requiring that videos be transferred over a network to be registered with a 3D model. The approach for this application is to combine the fields of computer vision and high-speed networking to create a system that takes in HD videos, streams the data to a server where a dynamic 3D model is constructed, and provides a virtual scene navigation program for viewing the videos in a 3D scene from a mobile device as shown in Figure 4.1.

![Figure 4.1. Overview of the LAaaS](image-url)
4.2 Custom Template Integration with LAaaS

Based on our component abstraction model we were able to obtain our macro operators as shown in Table 9 and identify several hybrid cloud components in the Knowledge base,

<table>
<thead>
<tr>
<th>Domain</th>
<th>Precondition</th>
<th>Features</th>
<th>Constraint</th>
<th>App</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network connectivity</td>
<td>High Bandwidth &amp; Low Latency</td>
<td>Layer3</td>
<td>600Mbps</td>
<td>PTaaS</td>
</tr>
<tr>
<td>Network connectivity</td>
<td>Low Latency</td>
<td>Wireless</td>
<td>10Mbps</td>
<td>SMaaS</td>
</tr>
<tr>
<td>Monitoring</td>
<td>Networking Monitoring</td>
<td>Bandwidth</td>
<td>2 scripts</td>
<td>PTaaS</td>
</tr>
<tr>
<td>End-to-end Virtualization</td>
<td>Virtual Desktops accessible through</td>
<td>Cloud Infrastructure</td>
<td>500 users</td>
<td>SMaaS</td>
</tr>
<tr>
<td>Remote Computing</td>
<td>Windows Server with different</td>
<td>Virtual Servers</td>
<td>Linux Server</td>
<td>SMaaS</td>
</tr>
</tbody>
</table>

Table 9. Template for LAaaS App.

Collection. Identify all the commonality and variability related with provisioning resources. Hybrid Cloud, Overlay Network, Computation resources, and High Bandwidth requirements were identified as the infrastructure for this application. The SMaaS Template was identified as a closest solution and selected since it has similar requirements such as a pool of Virtual Desktops, network measurement performance, Virtual Servers with image processing specification and Data Store Servers.

Composition. Once having a template the next step is to customize it by adding or removing components. In this case we added some features to the Template such as Benchmarking tool, different specification for the server and a combination of Mizzou Wireless connection and Research Networking infrastructure as a Networking component.
Consumption. Our test bed setup consisted of clients connected to wireless network that represents a standardly available campus enterprise network and a compute manager VMware Horizon View connected over a higher-speed campus research network as shown in Figure 4.2. We emulate a network made available in a disaster scenario in which, these two networks can be used in parallel.

![Infrastructure diagram for the LAaaS Application](image)

**Figure 4.2. Infrastructure diagram for the LAaaS Application**

### 4.3 Custom Template Evaluation

We tested transferring the data of interest over different types of networks and processing the videos on various server configurations to determine the capabilities of such a system and the necessary requirements for it to provide a high quality user experience.
To obtain a 3D model for the location of interest, previous data collected by Leica C10 HDS LIDAR scanner was used. This scanner provides a high resolution point cloud of a scene and 2D images of the scanned subject using a built-in camera. The output data from the scanner also consists of files containing the internal and external camera parameters for each image. This video data was transferred to the server through thin clients that were able to stream data by using curl functionality that is authenticated by the FTP server in the virtual server. Finally the HPC server conducted data processing as shown in Figure 4.3.

Figure 4.3. 3D planes for dynamic objects

4.4. Result Analysis

Creating 3D planes for dynamic objects. Figure 3.4. Left: Transfer time over Mizzou Wireless Network. Right: Transfer time over Research Network.
Figure 4.4. Collection stage transfer times for varying video sizes.

Creating 3D planes for dynamic objects. Figure 4.5 shows 3D planes constructed for moving objects identified in video using our modeling method. Left: Time required to compute 3D pose for increasing number of dynamic objects in videos on different server configurations. Right: CPU utilization on server during computation stage for different configurations.

Figure 4.5. Performance during the computation stage.
5. FUTURE WORK

Research projects are limited to many factors that in the past could not be accomplished due to the lack of technology and resources. Currently, data-set sizes are growing exponentially and collaboration projects among widely dispersed teams of scientist are increasing, the data movement among teams requires to put aside the current obsolete methods in where data movement demand is a key factor for success. The science DMZ aims to overcome these challenges that the existing networking technologies are facing by providing resources in a non-common networking environment, ensuring data movement with reduced latency by dynamic path selection, reliable data delivered, dynamic queue management and resource provisioning hence allows high-performance applications to be successful. In this sense, our Custom Templates can be customized by adding a new Science DMZ components. So, more work will need to be done in order to convert this emergent infrastructures in more suitable, repeatable and reconfigurable for a variety of application hence this work is the beginning of the exploration around Data Intensive Applications.

Figure 5.1 illustrate the cyber infrastructure deployed where components such as SDN and OpenFlow 1.3 [14] are integrated as part of the solution.
Figure 5.1. Double ended Science DMZ implementation
6. CONCLUSION

The work done around Hybrid infrastructure in terms reusable and scalable infrastructure is still limited. Through our Custom Templates approach and our methodology we were able to abstract computing and networking resources in order to create Templates for its later usage moreover we validated its implementation and translate a Custom Application from the knowledge base to a new hybrid cloud app, demonstrating its functionality and importance for the new infrastructure deployment.

We believe this has a huge impact since our campus is going forward to the 100Gig network and clusters for hybrid cloud are close to be placed hence more a more hybrid cloud application will require to deploy adequate infrastructure in a time manner.
7. BIBLIOGRAPHY


[8] NFV white paper


