HEP Data-Intensive Distributed Cloud Computing System Design Specification Document

CANARIE NEP-101 Project
University of Victoria HEP Computing Group

March 20, 2014

Version 1.0

Revision History

Date	Reason for Changes	Version
11/03/2014	Skeleton.	0.1
20/03/2014	Initial version.	1.0

Contents

1 Introduction

The use of cloud computing technologies is becoming ubiquitous for commercial and research computing. The HEP Legacy Data project (NEP-52) contributed to this trend, establishing a distributed cloud computing environment particularly suitable for compute intensive workloads. For the last several years, researchers within both particle physics and astronomy have used this distributed cloud model to great effect. However, they have been limited to workloads with modest input requirements. The NEP-101 HEP Data-Intensive Distributed Cloud Computing project will expand the capability of the distributed cloud by enabling it to run data intensive applications. Our goal is to run ATLAS jobs that can process real or simulated data samples that require up to 20 GB per core in a 12-hour period. Although the storage per core is modest, the amount of data required could reach 40 TB per day if the system is running 1000 or more simultaneous jobs. Enabling the system for data-intensive applications requires the addition of new functionality in the existing services and the integration of new technologies or services. We plan to use a federated system for managing distributed data. ATLAS already has its data distributed over 100 centres around the world. Our goal is to have each job either stream or stage the data from the nearest centre.

1.1 Purpose

The purpose of this document is to present the system architecture, components, design details and testing environments for the **NEP-101 HEP Data-Intensive Distributed Cloud Computing** project. It is intended as a guide for current and future developers, and will be delivered to CANARIE as a work product of the NEP-101 project.

1.2 Scope

The key component of the distributed cloud computing system is the Cloud Scheduler service, which was developed in an earlier CANARIE NEP project and published as a CANARIE RPI service (Batch Services). Cloud Scheduler manages application jobs and VM images over the distributed cloud. We will need to modify the Cloud Scheduler service so that it can become data-aware while making its scheduling decision.

In addition, new technologies have been developed that would significantly enhance the operation of the distributed cloud system. For example, in 2012 OpenStack has become the most popular cloud infrastructure software and approximately half of the clouds used in our system use OpenStack. In this project we will modify Cloud Scheduler so that it can use the OpenStack application interface (API) and enhance the functionality of our VM image repository so that it can automatically distribute the images to all cloud types.

The distributed cloud uses a single Squid cache for accessing the application software. A better and more reliable method would use a central server that points the cloud to the nearest Squid cache. We have a prototype system under evaluation and plan to use it in the production system.

Further, we are very interested in exploring the use of micro-VM images. Micro-VMs have the potential to reduce the size of the VM images by three orders of magnitude to tens of megabytes. Micro-VMs have the operating system software in the same remote

software caches as the application software. This would radically impact the way we view and manage VM images.

1.3 Definitions, acronyms and abbreviations

APF	The ATLAS AutoPyFactory generates batch jobs in response to workloads within the ATLAS PanDA queue. These APF batch jobs are payload-less, that is they do not contain the application or data to perform ATLAS analysis or simulation, but they are used to secure computing resources in a distributed grid or cloud environment. Having secured the required computing resources, an APF job selects and pulls a unit of work from the PanDA queue for execution.
API	Application Programming Interface - the protocol used by an application to request services from supporting systems.
ATLAS	A HEP experiment at CERN laboratory in Geneva, Switzerland (http://atlas.ch)
Cloud	A collection of computing resources and software that provide on demand through Web Services, Infrastructure As A Service (IaaS) Virtual Machines (see below).
DCCM	Distributed Cloud Computing Model as developed by NEP-52 and enhanced by NEP-101.
HEP	High Energy Physics, sometimes referred to as Particle Physics or Nuclear Physics
IaaS	Infrastructure as a Service - a web service that provides virtual computers on demand.
Image Repository	A storage capability employed by the NEP-101 system to manage virtual machine images. Depending on its' type, an image repository may be dedicated to a single cloud or shared by multiple clouds.
JDL	Job Description Language is a text file created by a user to describe a batch job.
NEP-52 [1]	HEP Legacy Data project (Oct 2009 - Mar 2012), created the DCCM for high throughput, modest data input/output serial processing.
NEP-101	HEP Data-Intensive Distributed Cloud Computing project (Oct 2013 - Dec 2014), will extend the DCCM for data intensive ATLAS applications.
Virtual Machine (VM)	A complete computer system represented in software

[1] https://wiki.heprc.uvic.ca/twiki/bin/view/Main/CanarieProjectNEP52

1.4 References

References

1.5 User Perspective

The system developed will extend the services provided by DCCM to facilitate the processing of ATLAS production jobs requiring an order of magnitude more input data than is currently practical. This will more than double the class of jobs suitable for this computing environment. These services include both interactive and batch processing capabilities providing enhanced facilities for the reading and writing of large data sets, the management of network traffic through advanced caching techniques, and the optimization, management and distribution of VM images.

Interactive services are extended to manage the distribution of VM images. Previously, researchers were empowered to create and upload images to the repository, to list and instantiate (launch) images within the repository that they either own (created by them) or were shared by another user. Having launched an image the user may create a new image by logging into into the running image as the super user, modifying the image to their requirements, and saving a personal copy or snapshot of the modified image. Through the use of a web browser, the NEP-101 extensions will allow the researcher to distribute available images to any cloud for which they have credentials, either manually or automatically through the creation of distribution profiles. Additionally, the owner of an image may rename or delete a distributed image, or revoke the privileges of other users to use it.

In every case, users must authenticate with each component of the services provided.

1.6 Document Organization

The rest of this document is organized as follows. Section 2 gives an overview of the functionality of the system. It describes the general structure of the system and its informal requirements. It is intended for a general audience. The remainder of this document, sections **??**, 4, and 5, describes in technical terms the details of the functionality of the system as well as the constraints imposed on it. These sections are intended for developers.

2 System Function Overview

The DCCM established by NEP-52 was limited to processing workloads with modest amounts of data. This project, NEP-101 HEP Data-Intensive Distributed Cloud Computing project, will extend the capabilities of the DCCM with the following enhancements:

2.1 Batch Services

Batch processing is a well established methodology for managing large High Throughput Computing (HTC) workloads on traditional computing clusters and, more recently, on IaaS compute clouds. A core component of DCCM's batch services is Cloud Scheduler, which is responsible for starting and stopping VMs as required by jobs in the batch queue. NEP-101 will extend the function of Cloud Scheduler in the following areas to meet data intensive demands and the evolving cloud support software:

- Native support for the OpenStack API will allow greater control of OpenStack clouds and allow the retrieval of meta-data in support of scheduling decisions. Information gathered through this API will also underpin the next item.
- Simplified JDL. Currently, Cloud Scheduler requires the user to specify parameters specific to the cloud types to be employed. This means that a user may have to define one job requirement in multiple ways to satisfy the needs of all the clouds used. This enhancement will attempt to rationalize the parameters needed to specify job requirements so they are consistent over all cloud types.
- Currently, many of the diagnostics needed to determine why batch jobs are not running as expected are unavailable to the end user and require esoteric knowledge for understanding and correction. This enhancement will bring greater clarity to the inner workings of batch services to allow monitoring, diagnosis and correction to ensure that the system is operating as desired.

2.2 Software Distribution

In the earliest deployments of DCCM, users created VM images that completely encapsulated the application environment resulting in large, bloated images traversing the networks between image repositories and target clouds. Very often, though an image contained the entire application, only a small portion of the available function would be exercised by any particular job. As the model evolved, application software was migrated to a software appliance which supplies running VMs with those portions of the application that are required by the jobs that are running. This evolution greatly reduced the size of VM images, allowed the reuse of images for multiple applications, and improved network bandwidth utilization. However, other network optimization's are possible. NEP-101 will make the following enhancements in the area of software distribution:

- The deployment and utilization of distributed web caches. Ideally, there should be at least one web cache "local" to each cloud. Having a "local" web cache avoids the retransmission of the same piece application software over the long-haul network many

times, once for each VM instantiation. Instead, the software traverses the long-haul network just once and then produces network traffic locally for each new VM.

- Load based auto-deployment of web caches. When the processing requirements for an experiment are very large (100,000 batch jobs or more), the load on local web caches can become prohibitive. NEP-101 will implement methods to determine when web caches are under stress and take remedial action.

- With the proliferation of web caches in place of a single software appliance, VMs need a way to dynamically determine the best web cache to use; that is the closest, usable (not too heavily loaded) web cache. NEP-101 will implement a web cache brokerage system that will allow VMs to obtain a list of eligible web caches.

2.3 Storage Federation

To date, DCCM has been particularly suited to jobs requiring modest amounts of input data. Taking advantage of distributed and replicated data stores, commonly used by HEP and other scientific disciplines, NEP-101 will make the following enhancements:

- The deployment of a HTTP based storage brokerage system. The system will discover and catalogue distributed data objects and provide a unified index of the collection upon request. A VM requiring any particular file will be able to query the storage broker to obtain the URL of the closest available copy and will retrieve the file from that location. This will have the two fold effect of minimizing network utilization and of spreading the load across all available storage servers.

2.4 VM Image Distribution

Originally, DDCM implemented a central, HTTP based image repository to service all supported private clouds, primarily Nimbus clouds. With the ascendance of competing cloud technologies, today the majority of clouds (both private and public) require that images reside in the cloud's local repository and provide no capability of instantiating an image from a remote repository. To facilitate image distribution in this new environment, NEP-101 will provide the following new services:

- A HTTP based image distribution service. Users will be able to identify a network of clouds, upload images, and mange image distribution around their defined network. The system will provide both cloud and image centric reports and provide consistency checking to ensure that images are where they are supposed to be.

2.5 VM Optimization

In the same way that only parts of an application may be executed by a job (see Software Distribution above), parts of the Linux kernel may not be required to run an application. Efforts are underway to reduce the Linux kernel to a few megabytes and to provide additional functionality on demand. NEP-101 will explore the use of these micro-kernels to further optimize network utilization.

3 Design Considerations

This section documents assumptions and dependencies that effect the design of each component.

3.1 Assumptions and Dependencies

- Users of this system have a high technical proficiency and are both familiar and comfortable with the Linux operating system together with its command line interface, applications, and utilities.
- The system will employ high quality free software components wherever possible and will only create new components when the stated objectives cannot be achieved otherwise.
- Existing free software components employed in the system:
  - Modern distributions of the Linux operating system, notably Scientific Linux 6 (SL6) and CernVM version 3.
  - KVM or XEN, virtualization hypervisors.
  - OpenStack, Nimbus, Amazon EC2, and GCE IaaS cloud interfaces.
  - HTCondor 8.x, batch job scheduler.
  - Apache, mod_ssl, and mod_wgsi, RESTful web services platform.
  - MyProxy credential management service.
  - WebDAV, CVMFS, and backend filesystems for data management.
- Components created or enhanced by this project:
  - Cloud Scheduler, a virtual resource manager.
  - Shoal, a web cache broker.
  - Glint, a VM image distribution service.

3.2 Authentication and Authorization

- Users of the system's services will be required to authenticate themselves using either cloud specific credentials or by X509 grid certificates issued by a recognized certificate authority.
- For cloud specific credentials, users will need to contact cloud administrators and obtain credentials for each cloud they wish to use. The process of obtaining cloud specific credentials is outside the scope of this document.
- For X509 credentials, we assume that users already have a X509 grid certificate, or know how to request a new one. In addition, users will need to contact cloud administrators to ensure that their credentials are authorized to access the services they need. The process of obtaining a grid certificate or obtaining authorization for a cloud is outside the scope of this document.

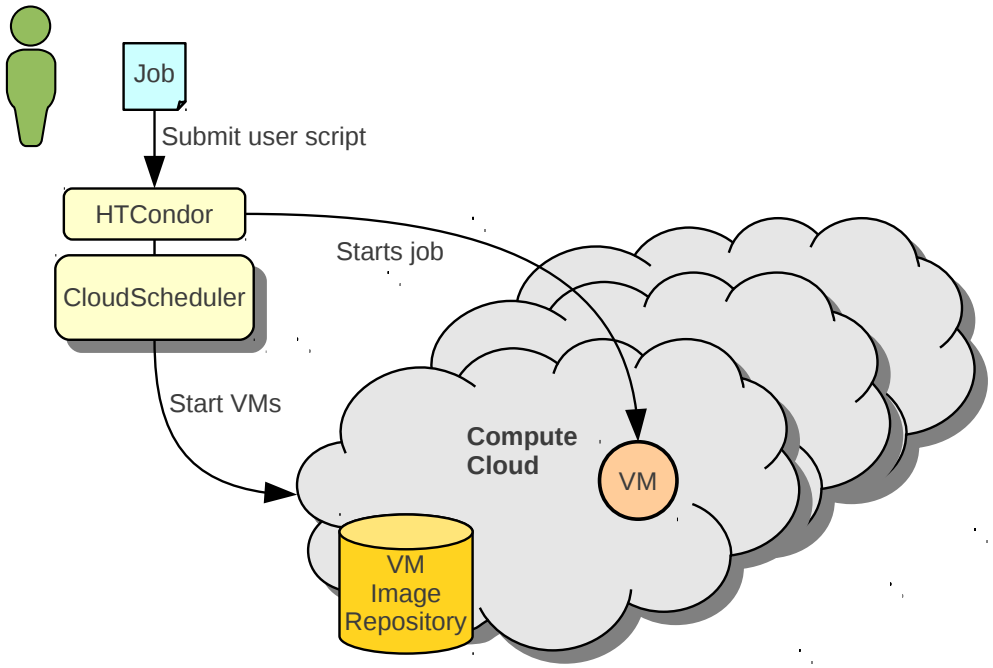

Figure 1: NEP-101 Batch Services: A user submits a Job Description file (JDL) to HTCondor, CloudScheduler sees the job in the HTCondor queue and starts the required VM, the VM registers with HTCondor, and HTcondor runs the job.

4 System Architecture

The following sections present the system architecture in terms of diagrams and text describing the data, image, and software management capabilities, followed by the interactive and batch services, a discussion of authentication and authorization, and concluding by a summary of the user interfaces by which the system is exercised.

4.1 Batch Services

This section (refer to fig.1) describes the enhancements that are being implemented as part of NEP-101. For a complete description of Batch Services architecture, please refer to [**?**].

4.1.1 OpenStack API Support

- Use the keystoneclient API library for authentication with OpenStack clouds. Documentation for this API can be found at http://docs.openstack.org/developer/python-keystoneclient/api/keystoneclient.v2_0.client.html.
- Use the nova-client API library for configuration discovery with OpenStack clouds. Documentation for this API can be found at http://docs.openstack.org/developer/python-novaclient/api.html.

4.1.2 Simplified JDL

- +VMAMI and +VMInstanceType parameters to support configured cloud names in addition to and interchangeably with host names.
- Cloud specific default values to substitute for missing JDL parameters. This can allow all CloudScheduler specific parameters to be omitted from the JDL.
- Target cloud aliases will allow the DCCM administrator to create, modify and delete arbitrary groups of clouds that can be selected by the the user for the execution of their jobs. This function will effectively allow the DCCM administrator the ability to vary clouds on and offline without affecting the user's JDL. This enhancement is to be implemented in the following areas:
  - A new configuration parameter to identify the path of an alias definition file. The file will contain a JSON dictionary with each entry defining an alias name and an associated list of cloud names.
  - The "adminctl" administrator command will be enhanced with two options, one to reload the alias file and the other to display the currently defined/loaded aliases.
  - The +TargetClouds JDL parameter will be enhanced to support target cloud aliases.
- VM image selection for instantiation by image name. This will require the novaclient API and will only be possible on OpenStack clouds. However, it will allow a single specification to identify an image on all OpenStack clouds reducing the complexity of required JDL specifications.
- Best fit flavour selection. The definition of flavours on OpenStack clouds is arbitrary and therefore requires a JDL specification for each cloud to be used. By having CloudScheduler match the job requirements against the defined flavours on a cloud, the need for flavour specification can be deprecated.

4.1.3 Diagnostics & Monitoring

- Augment CloudScheduler log messages to clarify operation and simplify diagnosis.
- Expose CloudScheduler diagnostics for consumption by administrators and end users to ease problem resolution.
- Provide an operations dashboard to monitor service and facilitate problem resolution.

4.2 Software Distribution

This section (refer to fig.2) describes an enhancement to DCCM to improve the performance and efficiency of software distribution by the implementation of web caches. The enhancement will create a new broker service, a project named "Shoal", and will employ "Squid", a well known open source software for the web caches. Both the "Squid" caches and the client VMs will need to be configured to interact with the new "Shoal" service.

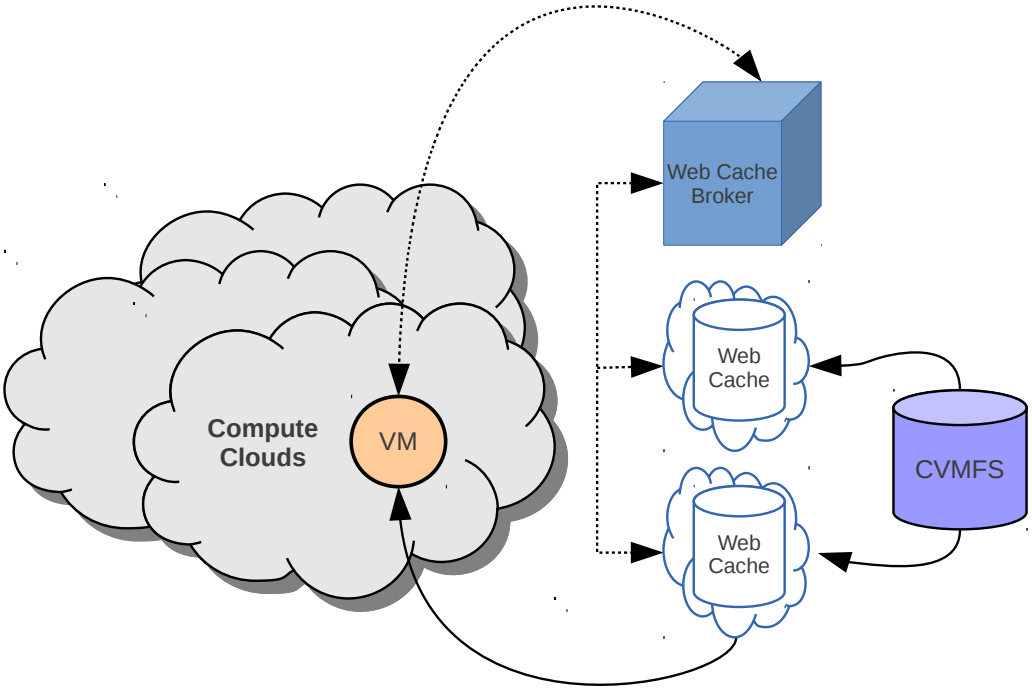

Figure 2: NEP-101 Software Distribution: When a VM starts, it obtains a list of web caches from the broker, ordered by closest location and lowest server load. The VM configures its' CVMFS client to point at the most eligible web cache. Subsequently, all software requests from the VM are satisfied by the local web cache which retrieves the required objects, just once for each item, from the CVMFS server.

4.2.1 Web Cache Discovery & Utilization

- During boot and periodically thereafter, VMs will execute a web cache discovery client which will be configured to retrieve a list of eligible web caches via HTTP. The list will be tailored to the requestor based on GeoIP location information of both the VM and all known web caches. The list will also be ordered by web cache processing load, so that selection of the "best" cache can be made by assessing the proximity and responsiveness.
- In response to an update of eligible web caches, the VM will update it's CVMFS configuration to optimize the retrieval of software.

4.2.2 Web Cache Broker

- The broker will service an AMQP message queue, processing transactions originating from appropriately configured web caches. The transactions will contain details of the origin including it's IP (for GeoIP location) and load.
- The broker will maintain an in memory catalogue of web caches and associated information for dissemination to VM clients upon request.
- The broker will service HTTP transactions requesting web cache information. The information return will be tailored to the requestor based on their GeoIP location.
- The broker will periodically verify the operation of known web caches by retrieving a well known file from the cache.
- Triggered by an AMQP transaction high load report and controlled by configured thresholds, the broker will initiate the auto deployment of additional web caches within a region.

4.2.3 Web Cache Identification, Location, & Load

- Web caches will be configured to periodically send transactions to an AMQP message queue. The transactions will identify the cache, it's location and load.

4.3 Storage Federation

This section (refer to fig.3) describes an enhancement to DCCM to facilitate the retrieval of large input data sets. By utilizing storage elements/servers that are currently distributed around the world, eliminating latencies that are associated with retrieving data over long-haul networks, processing large datasets within the distributed cloud becomes feasible. In order for batch jobs to dynamically configure the origin of their input datasets, a brokerage system with the following capabilities is required:

- The service must be capable of aggregating many storage endpoints distributed around the globe.

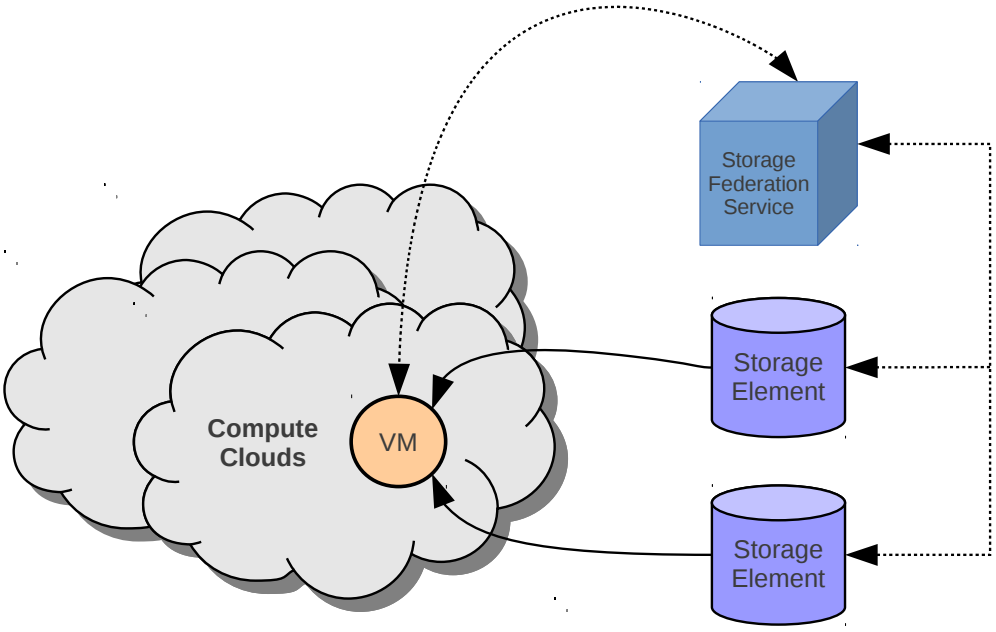

Figure 3: NEP-101 Storage Federation: The storage federation server is configured with the URLs of available storage servers, which it polls to obtain their served file trees. The retrieved file trees are collated into a single tree, each uniquely named file appearing just once, which is presented to VMs through HTTP. When the VM attempts to retrieve a file from the storage federation server, it is redirected to the closest storage server and the data flows directly from that server to the VM.

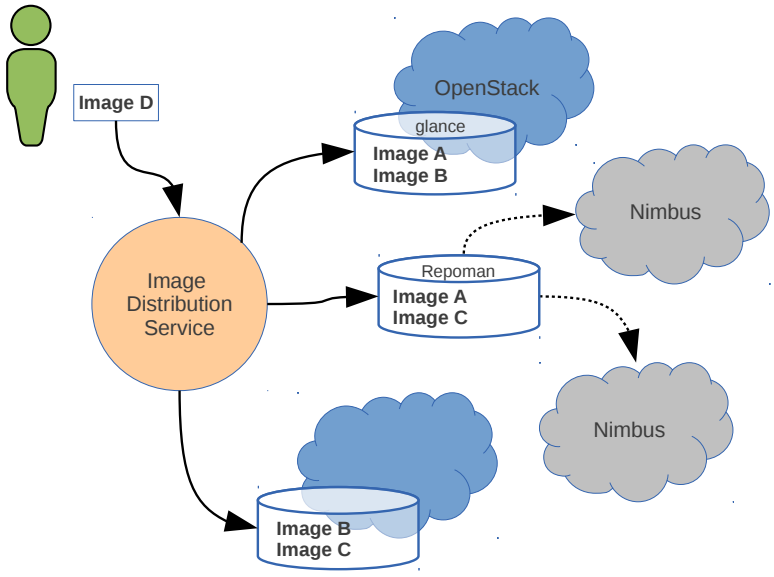

Figure 4: NEP-101 Image Distribution: A user logs in through a web browser to the Image Distribution server, uploads an image they wish to distribute, and configures the target clouds, providing location and credentials. Subsequently, through the browser interface, the user can distribute or remove images to and from any of the clouds they have configured.

- Filesystem services of both the federator and the storage endpoints must be available through HTTP.
- The federator will collate the POSIX file trees of all configured storage servers into a single file tree. Identically named files from different servers will be considered as multiple copies of the same file at different locations. The resulting catalogue will also maintain the location of each file copy.
- When a directory request is received through HTTP, the federator will present the aggregated POSIX file tree to the client.
- When a file request is received through HTTP, the federator will redirect the client to the closest copy of the file based on locations of both the client and the file. The locations of both the requestor and the target file are determined through the IP of each and GeoIP databases/functions.

4.4 VM Image Distribution

This section (refer to fig.4) describes a new image distribution service for DCCM. The service will have the following capabilities:

- The service will be available to authenticated users only; keystone authentication is desirable.
- The service will provide a web dashboard offering the following functions to the user:
  - User definition of cloud targets and cloud credentials.
  - VM image upload.
  - Copy image to one or more targets.
  - Delete image from one or more targets.
  - Both cloud centric and image centric reports.
- The service will provide system integrity checking to ensure accurate reporting of image locations. Since images can still moved and deleted manually on individual clouds, a reconciliation process will need to be instituted.

4.5 VM Optimization

NEP-101 will investigate the utilization of micro-kernels which employ distribution methods very similar to the software distribution methods described above. However, in this case, the bootable kernel is pruned to a few megabytes (normally three or four gigabytes). As other kernel functions are required, the modules are fetched and loaded from the network. This project will concentrate its efforts on CernVM version 3, which utilizes these techniques.

5 System Testing

System testing procedures are under development.