ARISTOTLE
CLOUD FEDERATION

CC*DNI DIBBs: Data Analysis and Management Building Blocks for Multi-Campus Cyberinfrastructure through Cloud Federation

Program Year 1: Quarterly Report 3: Part 2 of 2

6/30/2016

6.0 Appendices

6.1 Appendix A: Science Team Advisory Committee 4/1/2016 Meeting Minutes

Submitted by science team lead Adam Brazier – brazier@cornell.edu

Invited to 4/1/2016 Science Team Advisory Committee Meeting *(attendees italicized)*
Cornell University (CU):
James Cordes cordes@astro.cornell.edu
Shami Chatterjee shami@astro.cornell.edu
Adam Brazier brazier@cornell.edu
Angela Douglas aes326@cornell.edu
Bessem Chouaia bc335@cornell.edu
Nana Ankrah na423@cornell.edu
Brandon Elam Barker brandon.barker@cornell.edu
Sara C. Pryor sp2279@cornell.edu
Patrick Michael Reed pmr82@cornell.edu
Bernardo Carvalho Trindale bct52@cornell.edu
Julianne Dorothy Quinn jdq8@cornell.edu
Resa Reynolds rda1@cac.cornell.edu
Susan Mehringer shm7@cornell.edu
Paul Redfern red@cac.cornell.edu
David Lifka lifka@cornell.edu
University at Buffalo (UB):
Tom Furlani furlani@ccrbuffo.edu
Varun Chandola chandola@buffalo.edu
Cristian Tiu ctiu@buffalo.edu
Dominik Roesch drosch@buffalo.edu
Brian A. Wolfe bawolfe@buffalo.edu

University of California, Santa Barbara (UCSB)
Rich Wolski rich@cs.ucsb.edu
Andreas Boschke andreas@cs.ucsb.edu
Kate McCurdy kate.mccurdy@lifesci.ucsb.edu

National Science Foundation (NSF)
Amy Walton awalton@nsf.edu

Meeting Purpose - *Adam Brazier, science team lead (CU)*
Get feedback from the scientists to ensure that the Aristotle project team is responsive to their needs. Key aim: reduce time to science.

#1541215

Introduction - *Amy Walton, NSF program manager*
Amy explained that Dave Lifka (PI) and Tom Furlani (Co-PI) are not at today's Science Team Advisory Committee meeting because they were at NSF this morning (4/1/2016) giving an invited talk on the Aristotle Cloud Federation. This project is of great interest to NSF. Program directors from multiple directorates and Coalition of Academic Scientific Computation (CASC) members attended the meeting and interest was very high. The talk included your 7 science use cases and why your participation is important in making sure the federation works. NSF made only 4 DIBBs awards and this one was funded because it addressed 2 important concerns of the NSF: (1) creating sustainable models for cyberinfrastructure facilities, (in this case, sharing resources), and (2) creating metrics so that researchers can quickly find and value the computing and data analysis resources they need. Adam Brazier, Aristotle Science Team lead, added that NSF is looking for progress in the science use cases in the project's monthly reports.

Infrastructure Update - *Resa Reynolds, infrastructure team lead (CU)*
- Cornell's year 1 hardware up and running. 168 cores (6GB/core) and 120TB SAN storage were added to our existing Red Cloud infrastructure. Images are being created by our science teams.
- UB's hardware is in place and they'll be installing the Eucalyptus cloud software stack in the coming weeks
- UCSB's hardware started to arrive and will continue to arrive over the next few weeks. By the end of month, it will hopefully be up and running.
- As a team, our goal is get the infrastructure up and get out of way of the science
- Each year we'll be adding more hardware. If science teams have specific requirements, e.g., more RAM to core, etc., they should let us know. Adam suggested using Slack (the recently launched science team communication platform) to communicate needs as well as lessons learned.

Portal Update - *Susan Mehringer, portal team lead (CU)*
- Purpose of the user portal is to get scientists the systems info they need, including how to get to different systems in the federation
- Features will include: allocations info, systems status (how busy each node is), science project info, user documentation/training and eventually OpenXDMoD and QBETs for advanced resource prediction
- PIs will have access to overall usage data, team member info, how much allocation is left, etc. Project team members will see a subset of this info.
- Current portal status: the federatedcloud.org site is up with a "coming soon"/latest news page. We added SSL. We will be adding InCommon for authentication. The database schema has been designed and built. UB provided us a rest API to share data between the federation sites. We're using a simple open source framework for the user portal so it's easy to reproduce and a model for others who wish to start their own federation or possibly join ours. Adam commented that the goal of the portal is to make the federation easy to use so researchers can focus on their science rather than administrative tasks.

Science Team Update - *Adam Brazier, science team lead (CU)*
- Science Use Cases are moving! Expect different rates as each project is different and also available effort is at different stages of readiness.
- One element of Use Case 2, "Global Market Efficiency Impacts," requires larger volumes than currently available on Lake Effect at Buffalo, so it will run at Cornell (first instance of cross-federation deployment)

#1541215

- Cornell is committed to having a parallel MATALB capability and will make MATLAB Distributed Computing Server (MDCS) available on the CU node of Aristotle with 64 worker processes. This and other heterogeneity across the federation, in licensed software, installed software, and in hardware (heterogeneity may evolve over time), will be advertised to researchers.

Individual Science Team Updates
Note: science team project descriptions from the proposal were distributed to meeting invitees as a separate pdf

Use Case 1: A Cloud-Based Framework for Visualization and Analysis of Big Geospatial Data - *Varun Chandula update (UB)*
- Our goal is to create a framework and interface to handle all sorts of geospatial data and eventually remote sensing data. Users will submit analysis and visualization jobs to the cloud and at the back end requests will get translated into dynamically created cluster jobs that fire up, run, and are taken down after the users get their results.
- One of our students developed the app based on Spark. We're testing it now using UB's current cloud (called Lake Effect) so we will be ready to migrate to Aristotle as soon as it's available. Our interface is working well; we're focused on the back end plumbing now. We hope to have the first cut of our working use case in a month or two.
- We're eager to share with and learn from the other use cases. The less we reinvent, the better. We can learn from each other.

Use Case 2: Global Market Efficiency Impact - *Dominik Roesch, Brian Wolfe updates (UB)*
- *Roesch:* We'd like to look at efficiency in stock prices and derivations from fundamentals across international markets, however we need access to international market data which will cost ~15K. We'll discuss with Tom Fulani to inquire about possible funding. Otherwise, our alternative plan is to bring high frequency data to UB this summer and develop a framework for it.
- *Wolfe:* Varun developed an image for us and we have access to a couple of working ssh ports. We're ready to roll. We just need some software to do image recognize on pdf images.

Use Case 3: High Fidelity Modeling and Analytics for Improved Understanding of Climate-Relevant Aerosol Properties - *Adam Brazier update (CU)*
- This project is focused on using NASA satellite data and requires WRF-Chem software. Researchers have had a variety of problems getting this software installed on various clusters around the world because it's large and complicated (really a "pig"). Cornell CAC consultant Steven Lee built a parallel version of WDF-Chem which we'll soon test running as a cluster of VM's on Aristotle. Cornell's large instance size (32 cores) may be good for this. A big advantage of the cloud is that once an image is built, we can spawn many duplicates for scalable processing as demand increases.

Use Case 4: Transient Detection in Radio Astronomy Search Data - *Shami Chatterjee (CU)*
- This is a rather hot research area; we just published a paper in *Nature*. We're looking for a single pulse in sea of noisy data that is very fast in time and frequency. We will try out a variety of approaches on Aristotle with our data which is local. Also, at the recent NSF NanoGRAV Physics Frontiers Center meeting, other institutions with radio telescope data expressed interest in using Aristotle. If we build it, they may indeed come.

Use Case 5: Water Resource Management Using OPENMORDM -*Adam Brazier update*

#1541215

- Bernardo Trindale is building an image now. Aristotle will be very helpful in developing their research apps and the potential to run full-size simulations (extremely large, thousands of cores) on AWS is particularly appealing because they may want to develop an AWS service for US municipalities.

Use Case 6: Mapping Transcriptome Data to Metabolic Models of Gut Microbiota - *Brandon Barker update (CU)*
- We plan to map transcriptome data to metabolic models using algorithms with widely varying run times. Advantages of Aristotle: (1) no need to deploy large capacity system for apps whose computing times aren't known up front, (2) we're able to spin up either Windows or Linux instances which is beneficial because in some cases the same software packages have slightly different features available in Windows vs. Linux. We have been investigating NixOS as a great way to develop reproducible scientific workflows, but will use Ubuntu and Windows for now. Bessem Chouaia said that eventually it may be helpful to have a very high memory and low CPU node.

Use Case 7: Multi-Sourced Data Analytics to Improve Food Production - *Kate McCurdy update (Sedgwick Reserve/UCSB)*
- We're busy working on our instrumentation--deploying cameras, agricultural sensors, Wi-Fi mechanisms to bait and spring traps, and testing drones. Cloud data collection will be important for projects such as irrigation management and control that uses sensing and analysis for native farm plants to better understand water use and the effect on yield. We're also doing deer population monitoring with a ground survey and we'd like to augment or offset that with image analysis.
- Our "Where's the bear" project will be trying to analyze 5 million digital photos and do some facial recognition or some kind of species recognition to monitor wildlife populations for predator protection. We're also using drones to monitor black birds (CA endangered species) with a fixed camera in a remote site which we'll try to tie into our mesh network. We're excited about the prospect of Aristotle helping us with our data and analysis needs.

6.2 Appendix B: DRAFT Federated XDMoD Requirements

Federated XDMoD Requirements

Date	Version	Person	Change
2016-04-08	1.0 draft	XMS Team	Initial version

Summary

Federated XDMoD will support the collection and aggregation of data from individually managed HPC centers into a single federated instance of XDMoD for displaying federation-wide metrics. Data particular to an individual center will be available by applying filters. Each participating center will deploy an XDMoD instance through which local data will be collected and viewed. The Client will ship data for selected resources to a central repository for

#1541215

aggregation and display by a master XDMoD instance. In essence, we will create a central repository to support the display of data collected from multiple organizations (Service Providers) including jobs, users, projects (allocations), organizational hierarchies, etc. Data from individual organizations will be mapped into a common format where appropriate. This is similar to what XSEDE has done with the XDCDB but **without a single, XSEDE-provided set of resources, users, and allocations defined in common across all of the organizations**.

Definitions

Term	Definition
Federation	A set of individually managed XDMoD instances providing data to a centrally-managed repository for collection and visualization via a single shared XDMoD instance that provides cross-federation metrics.
Service Provider (SP)	An individual organization maintaining its own local XDMoD instance and providing data to the Federation Repository.
Federation Repository	The central location where Service Providers will send their local data to be included in the federation. The Federation Repository is configured as part of the Federation Master.
Federation Master	The single XDMoD instance that provides metrics across all members of the federation. It may perform some normalization, disambiguation, and aggregation of data from multiple SPs taken from the Federation Repository.
Federation Client	An XDMoD instance running locally at an SP that is configured to collect, normalize, and disambiguate data local to the SP and send it to the Federation Repository.
Federation Administrator	An administrator responsible for maintaining the Federation Master and Federation Repository. This administrator will be responsible for elevating user privileges and approving/configuring member SPs.
Client & Master Data Federation Modules	The optional XDMoD modules to be installed allowing an instance to communicate with the Federation Repository as a client or a master.
XDMoD User	An individual with an XDMoD account (i.e., a user of XDMoD). A User does not necessarily map to a Person.
Person	An individual associated with data contained in XDMoD. For example, a person is associated with individual jobs or financial data.
Local Value Space	The local view of data provided by a particular Federation Client. Values presented in this space may be specific to the Client may not be able to be

#1541215

	compared across Clients without normalization.
Global Value Space	This is the global view of federation data provided by the Federation Master. It has been normalized across all of the clients where necessary.
Single Sign On Service Provider (SSO-SP)	An XDMoD instance when within an identity federation is considered a service provider. This will be used to differentiate between an XDMoD SP and an Identity SP

Assumptions

The design of Federated XDMoD makes the following assumptions.

1. The federation will be tightly coupled. Members of the federation will have a vested interest in a working collaboration and will cooperate to manage items such as a common organization hierarchy, fields of science, and method for disambiguating people common to multiple sites in the federation. Federated XDMoD is not designed to map or disambiguate arbitrary information from members that do not conform to the federation-established guidelines.
2. Each SP will maintain their own set of local people, local usernames, local job identifiers, fields of science, organizational hierarchy, and other data. The process of federating this information will be handled by the Federation Master and any mapping will need to be agreed upon a-priori by the Federation.
3. Each SP who is a member of the federation will install and maintain their own local instance of Open XDMoD. This instance will collect data local to the SP and deposit it into the Federation Repository.
4. Federation Clients must request membership and be added to the federation by a Federation Administrator before they can begin sending data to the Federated Repository.
5. Each SP will maintain the appropriate version of XDMoD needed to properly report their data to the Federation Repository. If their version is incompatible, membership will be rejected.
6. It is possible for the same person to hold accounts at more than one SP, each with potentially a different local username. A mechanism will be provided for these people to be disambiguated at the federated level.
7. It is possible, especially in a federated cloud environment, for a job to both span multiple resources and to migrate from one resource to another (e.g., Aristotle cloud instances).

#1541215

8. Not all SPs will be required to provide the same set of data. For example, a subset of SPs may provide low level job information while another subset may provide application kernel data. The Federation Master will display data for those SPs that provide the data and charts will be annotated appropriately so that it is clear to the user what they are viewing.
9. The XDMoD Federation Client running at each SP may be a Single Sign On Service Provider (SSO-SP) using a supported institutional Identityrovider (IdP) as well as local (non-SSO) XDMoD accounts. The Federation Master can provide authentication via each supported Federation Client's configured IdP and will also support XDMoD accounts local to the Federation Master. Authenticating to the Federation Master using accounts local to a Federation Client is not supported.
10. Data federation will be enabled by two Data Federation modules that will be provided for XDMoD. These will allow a local XDMoD instance to ship locally collected data to the Federation Repository as well as allow a Federation Master to pull data from the repository.

Data Collection

Data will be collected locally at each Federation Client (SP) and transmitted to the Federation Repository, as shown in Figure 1. From there it will be ingested and aggregated into the local data warehouse of the XDMoD Federation Master where cross-federation data will be presented for visualization. Note that an XDMoD federation can have a **single** Federation Master for an unlimited number of Federation Clients while a Client can be a member of **more than one federation** (e.g., CCR DIBBS may be a member of the Aristotle federation and the CCR federation). A client can configure one or more resources as part of a federation. *A single XDMoD instance cannot serve as both a Master and Client.*

#1541215

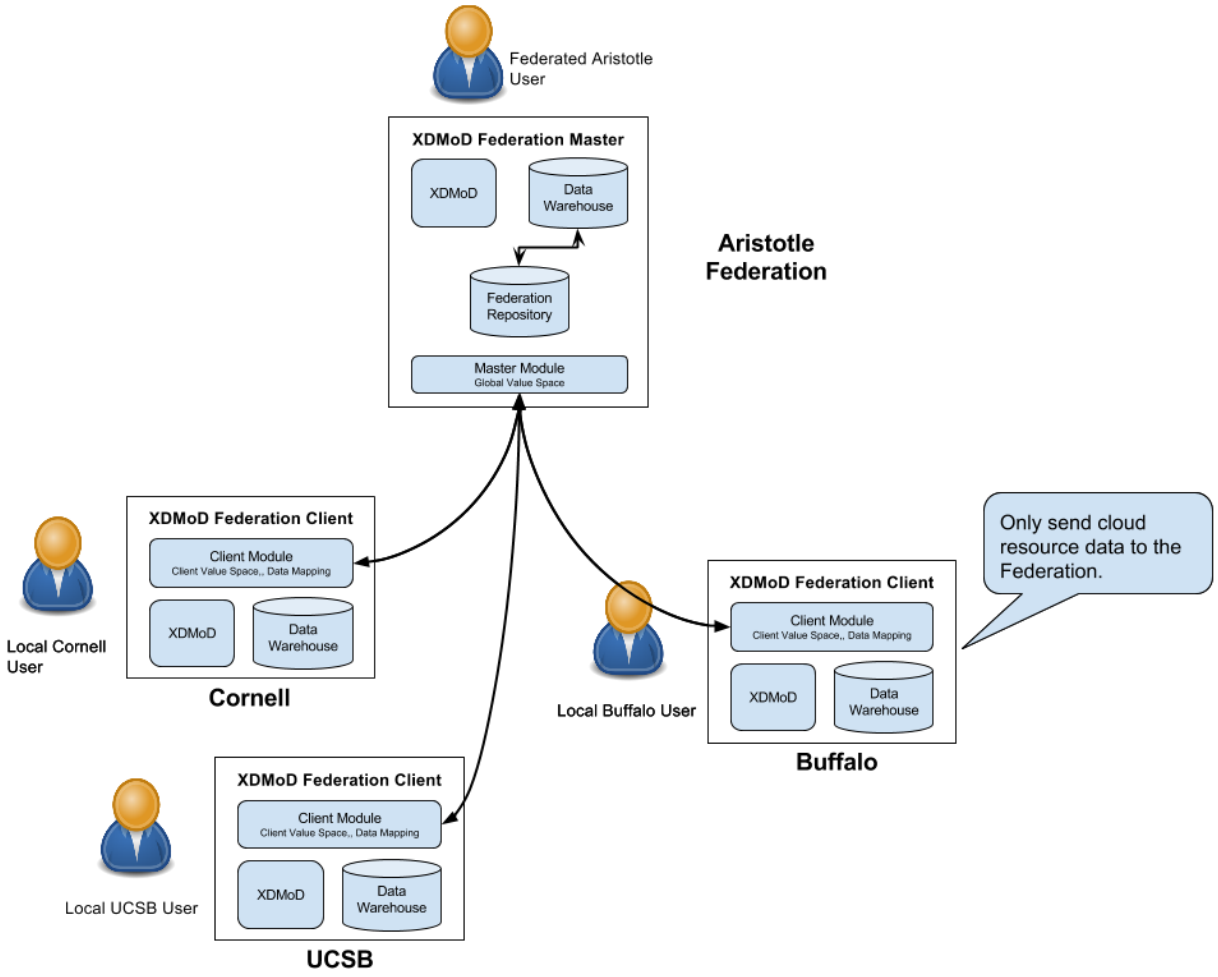

Figure 1: Example data flow for Aristotle Cloud Federation

Local XDMoD Installation

Each participating SP will install and maintain a fully-functional local instance of XDMoD and install and configure the Data Federation Client module. Data (jobs, people, local usernames, accounts/projects, FoS, organizational hierarchy, etc.) will be collected locally using whatever means appropriate for their specific configuration, and stored in the local XDMoD data warehouse. This local installation will be capable of displaying metrics for the SP. If an SP wishes to configure only the portion that collects data and not the web portal, that configuration will be supported as well. The Federation Master will also be a fully-functional XDMoD instance, however, rather than obtaining data from local sources it will obtain data from the Federation Repository. If the local installation (Client) has installed any optional XDMoD modules such as

#1541215

Application Kernels or XDMoD Value Analytics, these modules must also be installed on the Master in order for it to display information specific to these modules.

Module Support

In order to support federation of data collected and presented by a particular XDMoD module, the module must be installed on both the Master and at least one Client and *the module must be federation-aware.* This means that the module must specify which dimensions that it will make available to the global space and be able to map any keys in those dimensions from the Client's Local Value Space to the federation Global Value Space based on information provided by the Master prior to sending that data to the Federation Repository. In addition, the module ETL must be structured properly to allow the Federation Master to loop over the ingestion actions for all Clients that have sent data and ingest their data from the Federated Repository.

For example, a Client will have its own set of resource ids that are unique to the Client but not the federation. Prior to sending data to the Master Repository, the Client must be able to map from a local resource id to a global resource id. Another example is an organizational hierarchy. Clients will likely have their own local hierarchies while the Master will provide a normalized view of this data so the clients must map between their own Local Value Space to the federation's Global Value Space.
It should be noted that initially, only data stored in the data warehouse will be available for federation. This includes data from the Jobs, Allocations, and Accounts realms as well as *summarized* data from the SUPReMM realm, **but not job-level information from SUPReMM that is stored in MongoDb** since this data is not stored in the data warehouse. A future task will enable pass-through of job-level requests to individual Clients.

Data Federation Modules

Two optional modules will be provided to allow a local XDMoD instance to participate in a Federation: a Federation Client and a Federation Master module. These modules will facilitate data transfer and communication between Clients and Master. *These modules are mutually exclusive in that they cannot both be installed into the same XDMoD instance.*

Federation Client Module

The Federation Client module is installed into an existing XDMoD instance to allow data to be extracted from the local XDMoD data warehouse and shipped to the Federation Repository. It provides the following functionality

- Generate a request to the Federation Master for inclusion in the Federation. This request should include information such as the name of the Federation Client site, administrator email address, IP address of the Client for configuring access rules, the local resources to be included in the federation, and any other information deemed necessary for proper configuration.

#1541215

- If a Client supports multiple resources, select which of the resources will participate in the federation.
- Ship data from the local Client data warehouse to the Federated Repository, performing mapping to a common format where required.
- Provide a mechanism for mapping or normalizing, where necessary, local data values to global values supported by the federation prior to sending it to the Repository.
- Based on configuration options, send only new or changed data to the Federated Repository.
- Send data to the Federated Repository that was added between a provided start and end date.
- Provide a status report of data that has been sent to the Federation Master

Federation Master Module

The Federation Master module is installed into an XDMoD instance to allow it to read SP data from the Federation Repository and import it into the Federation Master's data warehouse to provide metrics across the federation. Data will be presented to the user from the Federation Master's data warehouse. It provides the following functionality:

- Process a Client request and generate configuration information to enable proper configuration of the Client. Configuration information will include any normalization information such as resource or organization ids unique to the federation to use when mapping data prior to sending to the Repository.
- Create database schemas, accounts, and access control in the Federation Repository for each approved Client.
- Provide information to the Clients that will allow them to properly map data between local values and values supported by the federation. For example, each Client may have their own organizational hierarchy and will need to be able to map these values to the global hierarchy supported by the federation as a whole.
- Retrieve data from the Federated Repository and add it to the Master's local data warehouse, performing normalization as required.
- Provide a status report of data that has been received from each Federation Client.

Data Mapping and Normalization

Individual XDMoD Clients will configure their instances to collect and organize data relevant to their local needs. For example, resources will be locally named and an organizational hierarchy would be created based on the hierarchy used at the local institution. When displaying data for the federation as a whole, the Master will provide a global, unified view of the data. This will likely require some mapping of local values provided by the Client into the unified view

#1541215

presented by the Master. For example, the federation will decide on a global organizational hierarchy to be presented by the Master but Clients may have established their own hierarchies. A mechanism must be provided for the client to map data from their local values to the federation-supported values prior to uploading to the Master Repository.

XDMoD will provide a mechanism for the federation, through the Federation Master, to define a global view to be used when displaying values across the federation. This view will be made available to the Clients by the Federation Master module and the Federation Client module will contain a mechanism to perform the mapping prior to sending the information to the Repository. Multiple mappings will be supported since a Client can report to multiple federations.

Federated Repository

The Federated Repository is the central data repository that will be used to receive and collect data from the Federation Clients (SPs) and serve as the primary data source for the Federation Master. The repository will be a single database instance, with each client depositing data into a separate schema to compartmentalize information. Schemas will be named appropriately to ensure there are no namespace collisions (i.e., federated_nics) with names controlled by the Federation Master. Rather than provide a REST API (and need to maintain that software stack), the repository will be accessed directly by both the Client and the Master XDMoD modules. While a separate database for the Repository is recommended, it is not required as long as there are no namespace conflicts.

Each Client will utilize a separate database schema to deposit their data and will authenticate using a user created by the Federation Administrator through tools provided by the Federation Master Module specifically for this purpose. The schema and username will be generated as part of the process of adding a new Client (see Configuration) and will be named based on the short name assigned to the Client. For example, a Client named “NICS” would deposit their data into the schema “federated_nics” using the user “xdmod_nics”. Data deposited into this schema must be in a known structure as defined by the Client Module.

Each Client will transfer data collected locally into their Federated Repository schema using a process defined by the Client Module to ensure that the data is in the correct format. Mapping may need to be performed by the Client in cases where local data does not match the organization or format supported by the Master (e.g., a different organizational hierarchy). The client will specify this mapping.

- People, local usernames (system accounts)
- Organizational hierarchy, Field of Science hierarchy
- Allocations/projects, PIs/project managers
- Resource definitions (types, specs, code, etc.)
- Job records (Master Records and Tasks)

#1541215

- SUPReMM and Application Kernel data

The Repository, through the Federation Master, will define the following items. Federation Clients will be able to query this information to synchronize their own data structures, if desired, or create/update mappings from their own local data to the format supported by the Repository. The Client is responsible for providing a mapping between their own local data and the definition provided by the Master in the ETL action that transfers data from the Client data warehouse to the Repository.

- Master Field of Science definition
- Master Organizational Hierarchy definition
- Location information (state, country)
- Bucket/histogram information?
- Master list of organizations
- Master (disambiguated) person list
- Master list of types (resource type, account type, user type, etc.)
- List of service providers (clients)
- Resource metadata (types, specs, code, etc.)

Optional Modules

Optional XDMoD modules may introduce new realms or dimensions into the data warehouse. In order to support federation of optional data, the Master and at least one Client must have the module installed and the module must support data federation. Supporting federation of data generated by optional modules will be supported at a later date.

Federation ETL

Federation Clients

Clients will collect data based on the existing XDMoD ETL process. This typically includes the shredding of resource manager log files, querying cloud resource tools, and importing of CSV files. This data will be sent to the Federation Repository where it will be processed by the Master. Sending data to the Federation Master should be performed by a distinct ETL action that can be run independently of local ETL processes. Normalization and/or mapping of the data will be performed by this Client ETL action as needed. Data to be mapped includes:

- Organizational Hierarchy: The hierarchy is based on the organizational unit that a user running a job has been assigned to. **We do not currently support a user belonging to multiple organizational units, although this is a requested feature and may be handled via tagging to avoid double-counting.** The Master will support a single organizational hierarchy for the federation. A method will be developed that will allow Clients to map

#1541215

local hierarchies to the master. Since we assume a tightly coupled federation, we assume that SPs will work together to do this.

- Field of Science (FoS): The field of science for a job is not associated with a person, but is assigned through a project or allocation that the job was run under. A person may run jobs under multiple FoS. Mapping Client FoS data prior to sending to the Federation Repository will be handled similarly to the organizational hierarchy.

Federation Master

The Master will perform ETL on Client data in the Federation Repository to ingest and aggregate data across all Clients, adding this data to its own data warehouse and performing normalization of the data as needed. The Master will generate all metrics from the data in its own local data warehouse. Data to be normalized includes:

- Person disambiguation: The same person may have accounts at multiple sites in the federation (possibly with different usernames). A method must exist to disambiguate them.

The Master will perform ETL using the Federation Repository as its primary data source. When performing ETL from the Repository, the master will execute a loop over the ingestion ETL actions for each client and store the data in its own data warehouse. Following ingestion, a standard aggregation ETL can occur to populate the aggregate tables. This ensures that the Master is a fully functional instance of XDMoD containing all necessary data in the event that access to a fact table is needed. The Master will need to maintain enough information to properly execute ETL appropriately based on the installed modules and the data that each Client has sent to the Repository.

Federated vs. Local Authentication

The XDMoD Federation Client running at each SP may be a Single Sign On Service Provider (SSO-SP) using a supported institutional Identity Provider (IdP) as well as local (non-SSO) XDMoD accounts. The Federation Master can provide authentication via each supported Federation Client's configured and will also support XDMoD accounts local to the Federation Master mapping accounts to their disambiguated identity. Authenticating to the Federation Master using accounts local to a Federation Client is not supported.

Configuration

Ideally, adding a new Client to the federation should be as simple as creating a new Client request (a request and approval is necessary to avoid namespace conflicts) via a command-line tool and sending that request to the Master for approval. Upon approval the Master will generate a configuration template to be installed on the Client that will enable the federation. The configuration might include an approved name and metadata for the Client as well as

#1541215

information for connecting to the a destination host, username, and credentials. Note that the abbreviation assigned to a Client may not be the same as the one requested in order to keep abbreviations and other identifying information unique.

Use Cases

Use Case: Aristotle Cloud Federation

The Aristotle Cloud Federation (https://www.cac.cornell.edu/about/news/aristotle.aspx) is a project to build a federation of Eucalyptus cloud installations at Cornell, the University at Buffalo, and UC Santa Barbara. One goal of the project is to build an allocations and accounting model that will allow institutional administrators to track utilization across federated sites as well as allowing users to utilize resources at any of the three sites and migrate instances between sites. Each site will install a local copy of XDMoD along with the Client Module and the Federation Master will be hosted in the cloud. The Master will provide utilization metrics for the federated cloud as a whole as well as each individual cloud installations. It is expected that Aristotle will use inCommon to authenticate users at each site. Note that, as shown in Figure 1, Buffalo will have multiple resources included in its local XDMoD installation and only the DIBBS cloud data will be configured to be a part of this federation.

Use Case: NSF-funded Projects (Non-XSEDE)

NSF is interested in viewing utilization metrics for multiple NSF-funded HPC resources in a single instance of XDMoD. Currently, metrics for all XSEDE resources are available via a single XDMoD instance but there are other NSF-funded resources such as Blue Waters, Open Science Grid, and LIGO that are not part of XSEDE. Rather than requiring program officers to access separate XDMoD instances local to each resource, a single federated instance where data can be viewed across all of the resources will be made available.

Use Case: CloudyCluster

Clemson is working to develop CloudyCluster (http://www.cloudycluster.com/) an infrastructure allowing users to easily deploy an HPC compute cluster on AWS. It deploys complete with an included scheduler, compute infrastructure, and storage based on OrangeFS, EFS and S3, as well as a variety of popular Open HPC software. Initial discussion between XMS and Clemson generated some interest for building XDMoD into a CloudyCluster deployment, as well as providing a federated XDMoD where each CloudyCluster deployment would provide data to a central repository for Clemson to view and use to provide user support.

Future Features

#1541215

1. Allow the Master to request that the client re-send data for a particular date range. This grants the Federation Administer the ability to possibly rectify issues with data transfer without needing to involve staff at Client sites.
2. Pass-through of SUPReMM job viewer requests from the Master to the Clients. Initially, we will not plan to support federation of non-summarized job data.
3. Allow the SP to act as an IdP for the Federation Master, allowing SP local accounts to be used to login to the Federation Master.

Implementation Details

Implementation details recorded from discussions among XMS team members.

Client Identities

When sending data to the Repository, can we use a multi-column key using the federation organization id for the client along with client-specific keys. For example, can a client resource_id be combined into a key (federation_client_id, resource_id) for unique identification?

Mapping Local to Global Values

Local values may need to be mapped to globally determined values. For example, the federation will decide on a global organizational hierarchy to be presented by the Master but Clients may have established their own hierarchies. A mechanism must be provided for the client to map data from their local values to the federation-supported values prior to uploading to the Master Repository.

ETL

The current plan is for Clients to send data (appropriately mapped) to the Federated Repository and for the Master to use the Repository as its primary data source. The Master will perform ETL ingestion actions to bring Client data into its own data warehouse where it can then perform standard aggregation actions. This will ensure that the Master is still a fully functional XDMoD instance in its own right, while incurring the storage overhead of the redundant data. If we were to attempt to aggregate directly on data in the Repository (spread across multiple Clients) we would need to join across the multiple schemas and possibly impose restrictions on the ETL actions.

#1541215

6.3 Appendix C: XDMoD Requirements Document-Job Reporting for Cloud and Other Non-Traditional HPC Resources

XDMoD Requirements Document

Job Reporting for Cloud and Other Non-Traditional HPC Resources

Date	Version	Person	Change
2016-05-23	1.0 draft	XMS Team	Initial version

Summary

The existing XDMoD data warehouse was developed primarily to report on data generated by individual HPC jobs run on traditional HPC resources. With the advent of open source cloud solutions such as Eucalyptus and Open Stack, as well as non-traditional HPC resources such as Hadoop running alongside traditional HPC clusters at many centers, we must re-examine the infrastructure used to store and report on center utilization as well as the definition of an HPC job within XDMoD. The capabilities of the XDMoD data warehouse will be updated to support these new resource types and become more flexible to better manage new types developed in the future.

Resources considered as part of this design are

1. Eucalyptus clouds
2. IU Jetstream (OpenStack) (http://jetstream-cloud.org/files/Jetstream_User_Guide-3-3-16-11am-Reduced.pdf)
3. TACC Wrangler (https://portal.tacc.utexas.edu/user-guides/wrangler)
4. SDSC Comet (http://www.sdsc.edu/support/user_guides/comet.html)
5. PSC Bridges (http://www.psc.edu/index.php/bridges/)

XSEDE Reporting Notes

Initial discussions with XSEDE resources such as Wrangler and Bridges (2016-04-22) indicate that they plan to report reservations to the XDCDB as jobs (with SUs charged for the reservation). This allows them to handle accounting details such as charging, canceling reservations, and refunds of SUs but does nothing for tracking finer-grained usage on XSEDE resources. **During these discussions it was reiterated that the XDCDB was designed as an accounting database and not a historical database of job records.** SPs also plan on reporting individual job records to the XDCDB with a local charge of zero SUs where available (such as

#1541215

jobs run on the HPC side of innovative resources) however these detailed jobs will not be available, for example, for Hadoop jobs. It is suggested that SPs use job attributes to indicate job types (such as reservations, hadoop, and hpc) as well as the reservation that a job has run under. It is likely that we will need to augment data from the XDCDB with additional data sources to properly report on innovative resources.

During follow-on discussions with Wrangler they will attempt to obtain activity records for Hadoop reservations and have reached out to Amy Schuele for help with job attributes.

Definitions

Term	Definition
XDCDB	The XSEDE Central Database. This is primarily an accounting database and not designed to accurately house historical information.
Traditional HPC Resource	A computing infrastructure, typically focused on high performance, managed by a resource scheduler such as Slurm or PBS. The infrastructure is managed by professional staff and a user is generally not able to modify the configuration.
Cloud Resource	An HPC resource where the underlying complexities are typically abstracted from the user and the focus is on scalability and flexibility using virtual machines rather than performance. The user has more control over the selection and configuration of individual machines.
Cloud Instance or VM	A virtual machine instance running on a cloud computing infrastructure.
Reservation	An allocation of a set of compute resources for a finite amount of time, for access by a particular project or group. Any authorized project or group user may run a job under the reservation.
Job	A job is defined as a request for and the consumption of resources by a user. A job may have zero tasks (e.g., a reservation that had no jobs), a single task (e.g., a traditional HPC job), or multiple tasks (e.g., a job array or cloud instance).
Master Record	A **request for resources** that includes metadata about the user, account/project, (initial) resource, a charge, and possibly other information related to the request. This is a resource request container that encapsulates individual tasks that consume resources. The Master Record can represent a single HPC job, a reservation, a job array, or a cloud instance.
Task	A child of a Master Record that is an actual **consumer of resources** such as

#1541215

	an HPC job, job array task, or intermediate reporting for a cloud instance. A task includes resource_id, core count, memory used, start time, end time, a type, and the resource where the task executed, and possibly other metadata. Note that the resource where a task executed may not be the same as that where the task was requested.

Job Reporting Requirements

Requirements for defining and tracking both traditional and non-traditional HPC jobs.

Requirement	**Recommendation**
Support traditional (individual) HPC jobs, job reservations, job arrays, and cloud jobs.	Modify the job definition be more general and support the concept of sub-jobs or tasks.
Support reporting on jobs that have not yet completed.	The Master Record will support an undefined end date if the job is not complete that will be updated on job completion.
Support intermediate reporting for long-running jobs.	Track intermediate reporting records as Tasks under a Master Record.
Support multiple suspends/resumes for cloud jobs and track individual job state transitions (provision, active, suspend, terminate).	Track state transitions as individual Tasks under a Master Record. Record a type with each Task.
Support heterogeneous jobs (multiple image types).	Since each instance is instantiated separately (Eucalyptus) there is currently no consistent way to group multiple instances together into a single, related job. Therefore, each job will have a single instance type. However, infrastructure should be put into place to allow arbitrary grouping of Master Records to support data analysis and future possibilities.
Support SUPReMM for cloud (VM, bare metal).	Install data collectors on the bare-metal node controllers, bake collectors into the supported cloud images, and pull information directly from the cloud control stack (e.g., Eucalyptus). Data will need to be verified across all collectors.
Track storage usage of idle/suspended jobs.	Implement storage reporting for persistent and volatile storage.
Support on-the-fly changes to VM configurations.	Cloud instances may change their core count and memory without terminating (this requires a start/stop). The start/stop event will trigger the start of a new Task and the updated metadata will be included in the new task.
Support varying SU definition across resources	Define SUs and provide a conversion factor.

#1541215

Job viewer support for reservations, job arrays, and cloud instances.	Update the job viewer to support a Master Record containing Tasks.
Support migration of VMs across resources (availability zones?).	VMs may be started on one resource and migrated to another resource. Ensure these are properly tracked as part of the same Master Record by including a resource identifier in each Task. **Note: Eucalyptus does not currently support this without shutting an instance down, migrating it, and restarting it on the new resource.**

Job Reporting

Traditionally, XDMoD has defined a job as a single job executed on a traditional HPC resource with reporting performed at the completion of the job. Information stored for a job included a submission time, start time, end time, number of cores and nodes used, the user that ran the job along with the allocation or project under which it was run, and a charge for the job. Missing from this record are treatment of job arrays, reservations, and add-ons such as GPU accelerators or Xeon Phi cards. In addition to supporting individual HPC jobs, we will also support cloud jobs and jobs run on "innovative" XSEDE resources such as Wrangler, Bridges, and Comet. As part of this effort, we will also add handling for reservations and job arrays and lay the groundwork for general support of other types of resources in the future.

Traditional HPC vs. Cloud Jobs

The table below shows some basic differences between traditional HPC jobs and cloud jobs that will be addressed as part of this document.

Traditional HPC Jobs	**Cloud Jobs**
• Run time upper bound known at submit time • Reporting on job completion • Static/fixed hardware resources • Typically 1-to-1 mapping to physical cores/memory • Single resource	• Unpredictable run time • Intermediate (as-you-go) reporting • Can start/suspend/resume multiple times • Varying image configurations within a job • Dynamically change image configuration (cores, memory) • Resource migrations • Oversubscription

Recording Usage

Currently, each (traditional HPC) job consists of a single record created upon completion of the Job and recording the job identifier, resource where the job ran, user information, account information, start/end time of the job, and node and core count information. To support tracking and reporting of traditional HPC jobs, reservations, job arrays, cloud jobs, and intermediate reporting, we will restructure the way that jobs are recorded.

For each Job we will record a Master Record which will describe a request for resources by a user. It is assumed that all Tasks executing under the same Master Record will be associated

#1541215

with or charged to the same account and that the Master Record will be updated to reflect the current charge. We cannot simply sum the charges associated with each Task because some resources will record a charge for a reservation and a zero charge for each Task under that reservation (e.g., Wrangler). The Master Record will record the following information

- Master Record identifier
- Resource where the job was initiated
- User that requested the resources
- Account to charge for the resources
- Charge (the total charge for the Job)
- Type (HPC job, cloud job, job array, reservation, etc.) dimension
- Submission time
- Start time
- End time. May be unknown if the Job has not completed, or may indicate the end data for a reservation
- Flag indicating that the job was completed. This may be derived from the end time.

Each Master Record will have zero or more Tasks associated with it where each Task represents the consumption of some amount of resources. A reservation with no associated jobs is an example of a record with zero tasks. Each Task will record the following information

- Task identifier
- Associated Master Record identifier
- Local task identifier (local job id, cloud instance id, etc.)
- Resource where the task executed
- User that executed the task
- Task type (task or event type: HPC task; cloud task such as provisioning, boot, suspend, resume, migration, or long-running update)
- Accounting charge for the task, if any
- Task submission time (may be unknown)
- Task start time
- Task end time
- Number of nodes consumed (1 for cloud instances)
- Number of cores consumed
- Memory allocated, if available

Auxiliary information will be stored separately and linked to a Master Record or Task as needed. For example, the list of cloud image identifiers and types can be related separately as can EBS volume and other storage related information. Future analysis or feature enhancements to cloud infrastructure may allow groups of related cloud jobs to be more easily

#1541215

identified. In this case we can create additional database structure to group a set of related Master Records as a cloud job. *Note that storing auxiliary information does not require that it be aggregated and displayed. We may collect information for potential use at a later date.*

Potential auxiliary information for a cloud may include

- Cloud availability zones
- Node controllers

Supported Dimensions

We will support the following dimensions for drill down capability:

- Master Record type
- User (and derived fields such as Organization)
- Account (and derived fields such as FoS)
- Completed/Running
- Task Type, Cloud instance type
- Resource
- Task type
- Cloud availability zone (future)
- Node controller (future)

Non-Traditional Accounting Data Format and Collection

Detailed accounting data for non-traditional HPC resources is not likely to fit into the existing schema of the XDCDB. Initial discussions (2016-03-22 Wrangler discussion) with the XSEDE accounting team have indicated that the XDCDB may not be the ideal location to store detailed accounting information specific to a particular type of resource (e.g., cloud resources) and the SPs should be responsible for the collection and storage of this information. **The method for the collection of local accounting information for individual resources is beyond the scope of this document and we assume that each SP is responsible for the collection, compatible formatting, and delivery of this information to a location where it can be brought into XDMoD.**

We will utilize a "File Format as API" methodology for the ingestion of non-traditional HPC data. XDMoD ingestors will be provided to read data from a pre-defined and infrastructure agnostic file format. A schema will be provided as well as tools to extract data from supported sources such as Eucalyptus log files collected via an ELK stack and place data into this format for ingestion into XDMoD. Individual installations are welcome to develop their own methods for extracting log data and placing it into the specified format. This is the same methodology used in the SUPReMM and XDMoD-VA data pipelines.

Support for Reservations and Job Arrays

XDMoD will support reporting of reservations and job arrays in addition to individual jobs. Both reservations and job arrays will generate a Master Record for grouping the individual Tasks that they contain. We must have a way to provide an XSEDE-wide comparison across resources

#1541215

where some resources may not be consistently providing a reservation and individual jobs that ran under that reservation. For example, reporting a reservation and individual tasks under the reservation for HPC jobs but only a reservation for Hadoop jobs.

Cloud Jobs (Instances)

Clouds currently support the concept of instances rather than a job with a collection of related instances. Similar to reservations and job arrays, instantiating a cloud instance will generate a Master Record and individual events related to the instance will be treated as Tasks. Events for cloud instances are stored with a timestamp, optional start and end times, the event type, and possibly other metadata. All events will be tracked to support future analysis, even if we do not initially report on them or choose to aggregate several together. It is likely that we will want to report on the system (e.g., provisioning, suspend, resume, termination) vs. user time (e.g, actual time the instance was running for the user) used by an instance. Events may include

- A user request for instance startup (instance provisioning)
- Booting of the instance (the instance has been provisioned and is starting)
- Instance Suspend/Resume notices
- A user request for instance termination
- A system “instance terminated” notice
- An intermediate report of resources consumed for a long-running instance

A user may “spin up”, or instantiate a number of related cloud instances but there is currently no consistent and enforceable way to group instances together if they are a part of a larger cloud job. Users may provide metadata to this effect, but it is not guaranteed. While we cannot enforce this, we may develop heuristics to infer the information.

Intermediate Job Reporting

We must provide the ability to receive accounting records during job execution and provide reporting for long-running jobs or cloud instances. This is different from the current environment where we receive accounting information only upon job completion. Tasks will be used to collect intermediate reporting data allowing as fine a granularity as desired (e.g., daily, hourly, etc.).

Migrating Jobs Across Resources

New resources such as cloud federations (e.g., Aristotle) may support the idea that an individual job or cloud instance can be migrated from one resource provider (the source) to another (the destination) . If the underlying infrastructure supports movement of unique identifiers between resources and logs the events when these transitions occur, we will support one-to-one tracking of the migration of an individual instance from a source to a destination resource. Any instances outside of the one-to-one mapping will be considered a new Job. Migrations across resources will be recorded as an outgoing migration Task on the source and a corresponding incoming migration Task on the destination, possibly with a new local instance id. The tool that performs the migration will need to report any new instance identifier on the destination resource.

#1541215

Job Metrics Display

Non-Traditional HPC Realm

Cloud metrics will initially be placed into their own Realm in order to ensure that existing traditional-HPC metrics are not skewed or adversely affected by their addition. Metrics will also be clearly labeled as traditional vs non-traditional. Over time, we will be able to identify metrics that are not adversely affected by the addition of non-traditional resources or can be normalized and include them in overall summary metrics.

This is also true of low level performance information. Raw vs. virtualized data will be presented in separate SUPReMM realms to ensure that the summation values will provide sensible data. The ability to plot data on same chart will still be supported. For example, plotting of raw node controller performance values against the sum of the virtual performance of all VMs running on that node controller.

Examples of existing metrics that may be adversely affected by the inclusion of non-traditional resources are:

Metric	**Impact**
Job Size (core count)	Cloud jobs are likely to be smaller than traditional HPC jobs. This may skew values towards smaller jobs, but that may be OK. Provide drill-down and filters by resource or job type so users can compare.
CPU Hours	If cloud resources are oversubscribed, the value should be normalized to actual CPU hours rather than virtual.
Node Hours	What does this mean when cloud jobs can have heterogeneous node (VM) configurations? Do we care?
Number of Jobs Ended/Started/Running	How will VM suspends/resumes affect this? We may see an artificial increase in values.

Cross-Resource Comparisons

We will strive to retain the ability to present an aggregate XSEDE-wide (also center wide and Federation-wide) view of resources where possible but realize that this may not always be the case. Metrics for traditional and non-traditional HPC resources will be displayed concurrently only in those cases where we are sure that the existing metrics will not be adversely affected. For example, CPU Hours consumed.

Non-traditional HPC Metrics

#1541215

In addition to the standard set of job metrics such as CPU hours, number of jobs started/ended, etc. (by user, FoS, SP) we will also support cloud-specific metrics. *Since cloud usage will likely be different than HPC usage, how will including cloud jobs affect existing metrics and how should this be communicated to the user? Should they be separated out on their own?*

Support the following cloud-specific metrics:

Metric	**Description**
VM image usage	• Number of each VM image started or active active • Resources consumed by VM image (CPU hours, network, etc.) • VM types (by user, PI, SP, etc.)
Utilization	How do we characterize cloud utilization. Support over-subscription via a conversion factor?
SUPReMM	• In-VM statistics • Per-resource statistics (bare metal) • Storage transfer rates
Storage	• Storage utilization for active and suspended jobs (by user, PI, SP, etc.) • EBS vs instance store
State transitions	Number of suspends/resumes (by job, user, SP, etc.)
User activity vs setup/teardown	Time spent in system vs user time (provisioning/setup/teardown vs. active user time).

Use Cases

Use Case: Traditional HPC Job

A traditional HPC job running on a single resource currently generates a single entry in the jobs table. Under the new model, a single job will generate one Master Record and a single Task under that record. The Master Record will contain general information about the job while the Task will contain information more detailed information about the resources consumed by the job. Both will be populated at the completion of the job.

Use Case: Reservation

A reservation will generate a single Master Record containing general information about the reservation including the total number of SUs charged and will be created as soon as the reservation is reported. As users run jobs and consume resources under the reservation, individual Task records will be created (upon job completion) to track the detailed activity under the reservation. It is possible that a user could create a reservation but not run any jobs under that reservation - this will result in a Master Record with zero Tasks. Since XSEDE

#1541215

supports reservation cancellation and the refund of SUs, it is possible that the charge associated with the reservation may change over time.

Use Case: Job Array

A job array is treated similarly to a reservation, although both the Master Record and associated Tasks may be created at the completion of the job array. The Master Record represents the job array while individual tasks, steps, or sub-jobs executed as part of the job array are recorded as Tasks under the Master Record.

Use Case: Cloud Instance

Cloud instances are more complicated than traditional HPC jobs in that the cloud infrastructure typically generates a larger number of trackable events and can also be migrated between resources. Since we are currently unable to reliably group multiple cloud instances into a single logical job in the sense that we might group multiple HPC nodes allocated to a single HPC job, each cloud instance is treated as a single job and generates its own Master Record.

The lifespan of a cloud instance can be broken into distinct periods with each period identified by a known event. These include provisioning of the instance, booting the instance, suspension, resumption, termination, attaching a storage volume, migration to another node controller, migration to another resource, and a periodic reporting of usage. Events can be grouped into system events, such as provisioning or migration between hardware controllers, or user events such as suspend and resume. Multiple events of the same type may be recorded over the lifetime of a cloud instance, such as suspend/resume cycles. Each of these events will cause a new Task to be recorded under the Master Record, allowing us to record the history of the instance. In addition to Tasks triggered by an event, a Task may also be recorded to track periodic usage of a running instance. Initially, a Task will record a start time but may not record an end time if the Task has not yet completed. The end time may be inferred from a subsequent task. For example, a Task may be created to record a provisioning event with a start time and no end time until a boot event is recorded, at which time the end time of the provisioning Task will be the start time of the boot Task. If an instance is suspended, this will be recorded as a Task and will also serve to mark the end of the previous Task.

#1541215