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 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 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	Track state transitions as individual Tasks under a Master Record. Record a type with each Task.

(provision, active, suspend, terminate).	
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.
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 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 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 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.

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

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 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.