XSEDE Data Analytics Use Cases: Architectural Response

Version 1.0

Revision history

Version	Date	Summary of changes
1.0	2015-11-13	Initial version. Lacks responses to the quality attributes provided in the description document. (The attributes described don't follow our engineering guidelines.)

1. Introduction

This document explains how the XSEDE system architecture supports XSEDE's data analytics use cases. The use cases are described in the document, *XSEDE Data Analytics Use Cases, Version 0.3*, dated June 14, 2013 [UCDA].

This document refers frequently to *XSEDE Architecture Overview 2.0 [ARCH]* and *XSEDE Architecture Level 3 Decomposition [L3D]*. Readers should have both of these documents available for reference and should already be somewhat familiar with their contents. It also relies heavily on architectural responses to XSEDE's canonical use cases. (See the [UCCAN-*n*] references below.)

1.1 Structure of this document

This document is organized as follows. Section 2 describes and summarizes the science gateway use cases. Section 3 describes how the XSEDE architectural components should be used to implement the use cases from section 2.

1.2 References

[UCDA] XSEDE Data Analytics Use Cases, Version 0.3, June 14, 2013. (http://hdl.handle.net/2142/45702)

[ARCH] XSEDE Architecture Overview 2.0. (http://hdl.handle.net/2142/50274)

[L3D] XSEDE Architecture Level 3 Decomposition. (http://hdl.handle.net/2142/45114)

[UCCAN-1] XSEDE Canonical Use Case 1 - Level 3 Decomposition. (http://hdl.handle.net/2142/73149)

[UCCAN-2] XSEDE Canonical Use Case 2 Response: Managed File Transfer. (http://hdl.handle.net/2142/73209)

[UCCAN-3] XSEDE Canonical Use Case 3 Response: Remote File Access. (http://hdl.handle.net/2142/50327)

[UCCAN-4] XSEDE Canaonic Use Case 4 Response: Interactive Login. (http://hdl.handle.net/2142/73210)

[UCCAN-6] Canonical Use Case 6: Authenticate to one or more SP resources, SP services, and XSEDE central services: Architectural Response v1.1.2.

[UCCAN-9] Canonical Use Case 9: XSEDE User Identity and Access Management: Architectural Response v1.1. (http://hdl.handle.net/2142/73214)

[IDM] XSEDE Identity Management Use Cases: Architectural Response, Version 1.3. July 30, 2015. (https://drive.google.com/drive/u/0/folders/0B-TKED6wiDyISnlaWGpkM3NaclU)

2. Data analytics use cases

This section describes and summarizes the data analytics use cases. This summary includes both the functional descriptions and the non-functional characteristics (aka, “quality attributes”) for these use cases. The full descriptions of these use cases are documented in [UCDA].

Following familiar software engineering principles[1], the XSEDE Architecture and Design team engaged with stakeholders from XSEDE’s “big data” and advanced user services groups to document these use cases, detailing the desired user experience and associated quality attributes. The resulting use cases and quality attributes are listed in Table 1. In Section 3, we describe how these use cases and quality attribute scenarios should be implemented in the context of the XSEDE system architecture.

Table 1: Data analytics use cases (shaded) and their associated quality attributes

DA-1	**Data analytics resources and information**
QAS-DA-1.1	Accuracy
QAS-DA-1.2	Completeness
QAS-DA-1.3	Context
DA-2	**Data preparation**
QAS-DA-2.1	Accuracy
QAS-DA-2.2	Completeness
QAS-DA-2.3	Context
QAS-DA-2.4	Data owners can determine who can read and write their data
DA-3	**Instrument data analysis**
QAS-DA-3.1	Accuracy
QAS-DA-3.2	Completeness
QAS-DA-3.3	Context
DA-4	**HPC simulation data analysis**
QAS-DA-4.1	Accuracy
QAS-DA-4.2	Completeness
QAS-DA-4.3	Context
DA-5	**In-situ computational steering**
QAS-DA-5.1	Accuracy
QAS-DA-5.2	Completeness
QAS-DA-5.3	Context

[1] Barbacci, M., Ellison, R., Lattanze, A., Stafford, J., Weinstock, C. and Wood, W. Quality Attribute Workshops (QAWs), Third Edition, Software Engineering Institute, Carnegie Mellon University, CMU/SEI-2003-TR-016, 2003.

3. Architectural response

Many of the elements of the XSEDE system architecture that implement these use cases are already documented in XSEDE's canonical use cases.

- DM-2, DM-3, DM-4, and DM-5 are mainly accomplished via either UCCAN-1 (remote job submission) or UCCAN-4 (remote login) for computation, with support from either UCCAN-3 (file access) or UCCAN-2 (managed file transfer) for data access.
- DM-1, on the other hand, is mainly accomplished via the XSEDE User Portal (XUP), which is not covered by the XSEDE engineering process at this time. The XUP may, however, be supported by UCCAN-7 and UCCAN-8. Specifically, the XUP may obtain resource details from XSEDE's information services as described in those canonical use cases and use those details to fulfill the expectations of this use case.

The following sub-section explains in more detail how the overall XSEDE architecture relates to these data analytics use cases.

The remainder of the document (§3.1-5) provides details of the implementation that are not covered explicitly in the canonical use cases. In particular, we provide additional detail on the component-to-component interactions and we provide significantly more information about how quality attributes should be satisfied.

3.0 XSEDE's architecture for data analytics

This section provides a brief overview of how the XSEDE architecture is intended to support data analytics tasks. We first refer to [ARCH §D], in which we introduce XSEDE's high-level architecture, copied below in Figure 1.

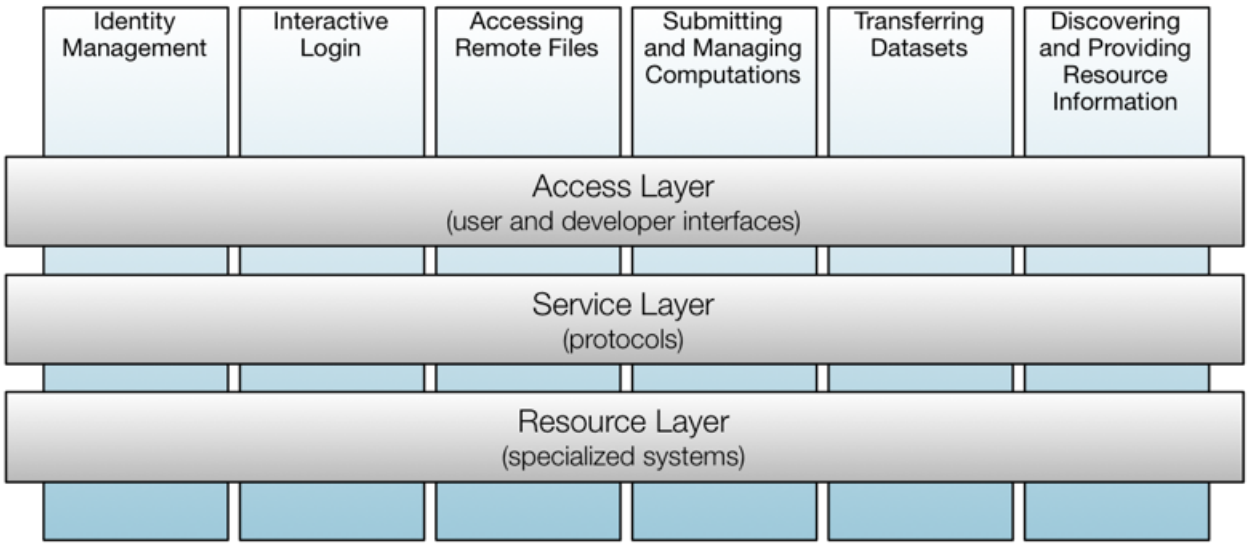

Figure 1. XSEDE architecture layers and system-wide functions

The top-most layer of the XSEDE architecture, the access layer, consists of three kinds of interfaces, described in [ARCH §D.1]: *graphical user interfaces* (GUIs), *command-line interfaces* (CLIs), and *application programmer interfaces* (APIs). Each of these interfaces may be available in one or both of the following forms: *thin-client interfaces*, accessible via software that is typically pre-installed on standard systems (Windows, Macintosh, Linux); and *thick-client interfaces*, which require software to be installed that probably isn't pre-installed on a standard system.

The service layer and resource layer are described in [ARCH §D.2] and [ARCH §D.3], respectively. The following points may be of particular interest.

1. XSEDE's foundational architecture provides several important data services, but it does not provide a *complete* solution on its own. XSEDE's services cover **creating** data, **analyzing and processing** data, and **storing** large collections of data. The XSEDE foundation architecture *does not* provide specific services in support of generalized metadata functions.
2. The data analytics use cases do not explicitly mention user authentication or authorization features (except as a quality attribute in DA-2), so we assume that these features are implicit. XSEDE's Identity Management canonical use cases [UCCAN-6] and [UCCAN-9], as well as the identity management use cases [IDM 1-10] provide a large set of user registration, identification, authentication, and authorization features.
3. *Data storage* and *computation* are both services offered by independent XSEDE service providers (SPs) and others. Both services can be access via access layer interfaces (as described in canonical use case responses), but each can also be accessed via the SP local file system and computation interfaces. These direct, local interfaces are represented in the XSEDE architecture as specific elements in the resource layer. Technical support for the interfaces in each layer is provided in different ways.
    a. Access-layer interfaces are supported by the XSEDE organization (support@xsede.org).
    b. Each XSEDE SP defines and supports its own resource-layer interfaces--usually directly and without XSEDE involvement.
4. The access layer is designed to mitigate faults and failures in the lower layers. Some service-layer and resource-layer interfaces provide this kind of resiliency, but not all do. For example, when transferring a large dataset of 10,000 files from one resource to another, the service layer may fail to transfer all of the files, returning an error message for the others. Access-layer interfaces will often automatically catch these failures and retry the failed transfers or even notify XSEDE staff who can intervene manually on the user's behalf.

For each of the data analytics use cases in this document, we will identify the access-layer and resource-layer interfaces that we believe are most beneficial to users and developers, and we will identify alternatives for special cases.

The XSEDE system is designed to meet the needs of a wide range of research disciplines and methodologies. We consequently support two distinct access modes for data.

- The *transfer* mode requires very little software installation or configuration and can be used with very little preparation. “Transfer” means, in essence, moving data around the system (or into/out of the system) to get it where it can be used. The user (or the user’s application) issues a series of requests for files to be moved from one location to another. These locations include XSEDE systems, campus systems, the user’s local systems, or systems at other research facilities. Thus, the user’s work consists of orchestrating a series of data transfers and data transformations at various locations. Each transfer request can include any number of files and directories, and the system ensures that all files and directories are transferred correctly and completely, notifying the user (or application) when each transfer request is complete. In addition to complete transfers, several forms of one-time synchronization (mirroring) are also supported. This access mode is described in detail in [UCCAN-2].
- The *virtual file system* mode, addresses the challenge of securely sharing data between universities and supercomputing centers, and directly between universities [cite GFFS paper]. In a nutshell the GFFS provides a secure global directory structure into which storage, compute resources, identity resources, and file system directory trees can be linked and then subsequently securely accessed using command line tools, a GUI, and by mapping the GFFS directory structure into the local file system using a GFFS-aware FUSE driver or SMB server. This is described more fully in [UCCAN-3].

3.1 DA-1 Data analytics resources and information

The following is the description of this use case from the definition document.

> *User obtains information about data analytics in the XSEDE resource portfolio, discovering existing XSEDE analytics resources and accompanying documentation. This is a pure discovery process; no data manipulation or analytics are performed. This information includes: 1) definitions and examples of data analytics in the context of XSEDE; 2) a comprehensive list and user guide for computational resources and software available for data analytics; and 3) support for gaining access to and performing data analytics on XSEDE resources.*

The use case steps further describe: (a) the user navigating the XUP and performing searches for “resources and applications” relevant to data analytics, and (b) XUP returning references to resources, applications, and additional information (user guides, allocation instructions, glossary definitions, etc.).

The fact that the XUP is described in the use case as the primary interface for this interaction is in close agreement with the architecture’s presentation of XUP as the primary access layer interface for learning about and coordinating the end users’ interactions with XSEDE. XUP’s internal implementation is not described by the XSEDE architecture, but its overall position as an access layer interface is clear.

The information about “resources and applications” to which XUP provides an interface is the domain of the sixth vertical column in Figure 1: “Discovering and Providing Resource Information.” This column corresponds to canonical use cases 7, 8, 11, and 12. [UCCAN-11] and [UCCAN-12] focus on publishing and updating this information, while [UCCAN-7] and [UCCAN-8] focus on searching for and subscribing to this information. Consequently, the implementation of XUP may make use of the mechanisms described in [UCCAN-7] and [UCCAN-8] to obtain some or all of the resource and application information needed for this use case.

3.2 DA-2 Data preparation

The description of this use case is as follows.

> *User engages in data preparation, an activity that may include selecting, cleaning, supplementing, integrating, formatting, and modeling data. Users should be able to gather, manipulate, translate and organize their data as needed for data mining, modeling or analysis activities. For example, the computational environment should be configured such that users can: collect data from local and remote resources; store data to archival resources; stage data from archival resources to compute resources; and execute a broad array of data preparation processes, such as cleaning, integration, transforming, reduction summarizing, sampling, etc. XSEDE needs to provide tools for creating metadata, and libraries for reading standard file formats such as NetCDF, HDF5, and others. Data and metadata may be in multiple formats (flat file, relational RDBMS, Hadoop data file system, text, video etc.). Access to data may be local or remote. Some applications may need to access the internet in unconstrained ways. (Note: need to support applications that require Internet scraping, i.e., they make frequent external TCP connections. This may become a property of a particular set of resources enabling these features.)*

This use case combines two important areas of concern: (1) assembling data from multiple sources, and (2) interpreting, formatting, and annotating the data in advance of large-scale, unified analyses. The former requires access to data from many sources and a means for either physically moving the data to a common location or assembling a "virtual" collection. The latter requires the availability of software tools for interpreting a variety of formats, re-formatting data into a (presumably smaller) set of formats, and adding annotations or "metadata" that describe the data. **This architectural response deals mainly with the first concern (accessing data and assembling a unified view of the data).** The second concern (availability of software) is not architectural in nature: it focuses on the local software environment available to XSEDE users, which is determined by the service providers (SPs) who operate XSEDE computation and data resources. The XSEDE architecture doesn't play a role in making software tools available to end users on individual resources.

With regard to assembling data, the use case does not specify precisely how "collecting," "storing," or "staging" data is presented to the user. The XSEDE architecture provides two styles of access layer interfaces for this access: (1) a "transfer" style via which the user can explicitly move files between locations, or (2) a "virtual file system" style via which the user can assemble what appears to be a locally attached storage system (with notable performance differences) by attaching files and directories from one or more remote locations. Each of the following subsections covers one of these styles.

3.2.1 Transfer interfaces

The transfer interfaces are implemented as described in XSEDE's canonical use case 2, "Managed file transfer." [UCCAN-2] These interfaces require very little up-front setup, software installation, or configuration and can be initiated with little or no preparation. The main preparation is to ensure that the data collection can be physically housed on the system(s) where the data formatting software tools are available. Some effort must be spent consulting the XSEDE documentation to identify the appropriate storage systems and to obtain allocations or storage quotas on those systems.

When using the transfer interfaces, the user does not need to be logged into the XSEDE resource where the data will be kept for processing. The data can be moved to the resource prior to the user logging in. Instead, the user logs into the XUP as described in [IDM-2], and navigates within the XUP to the Globus service interface for managed file transfer, as described in [UCCAN-2].

Fetching data for processing (the second step in the use case) is accomplished using XSEDE's file transfer functions, which are made available via XUP and Globus. [UCCAN-2] Assuming both the source and destination have Globus Connect software[2], the user can either use the XUP/Globus file transfer interface via a web browser, the Globus command-line interface (via SSH), or a RESTful web application to request any number of data transfers from the data sources to the resource where data preparation will occur. Once a transfer request is made, the user can disconnect from the interface and wait for an email notification signifying the completion of the transfer(s).

Once the data to be analyzed has been collected on XSEDE resources for analysis, Step 3 (data preparation) can be accomplished using the local mechanisms on these resources. The XSEDE architecture treats these as resource-layer interfaces, and they typically consist of local job queuing interfaces, each of which provides specialized mechanisms to allow each executing job to access data on the system. These specialized mechanisms typically include: shared filesystems (shared among the discrete nodes within the system), staging mechanisms, and application programming frameworks.

Step 4 (data preservation) is accomplished in the same manner as Step 2 (data collection). In this step, the prepared data (rather than the source data) would be transferred, and the destination for these transfers would be a mass storage system provided by an XSEDE SP.

---

[2] The specific instructions depend on where the data is located initially and where the shared collection will be housed. If the data is initially on the collection manager's personal (non-shared) system, he/she should install and launch the Globus Connect Personal software (described in [UCCAN-2]) to make the data available to XSEDE. If the data is located or housed on a shared non-XSEDE system (a campus server or another research facility), that system must have Globus Connect Server installed and the user must be authorized to use the system.

3.2.2 Virtual file system interface

XSEDE's virtual file system interfaces can be used in the following manner to accomplish the work described in this use case.

1. The user must be registered with XSEDE's XUP and have an active allocation to use one or more of XSEDE's compute resources. The allocated resources must already have the data preparation tools required by the user installed and accessible by the user. The resources must also already have the Genesis II software installed and must be configured to allow the user to mount the GFFS virtual filesystem in his or her "user space."
2. All of the data to be prepared for analysis must be either: (a) already located on a system that has the Genesis II software installed and is configured to allow the user to export data to the GFFS virtual filesystem; or (b) transferred to a system that meets the criteria in (a). (See §3.2.1 for a means of transferring data to such a system.)
3. The user will login to each system where the data is located (see 2a above) and use the Genesis II GFFS client command line tools or GUI to export the data into GFFS as described in [UCCAN-3].
4. Once Step 3 has been performed on all systems where data is located, the user will login to the XSEDE system(s) where data preparation tools are available, mount the GFFS filesystem (see Step 1 above), and then execute the data preparation tools using the GFFS pathname(s) to the data.
5. Data preservation for the prepared data (Step 4 in the use case description) is best accomplished using the transfer interfaces described in §3.2.1.

3.3 DA-3 Instrument data analysis

This use case is almost exactly the same as DA-2 from an architectural point of view. The only significant difference is that in addition to preparing the data, this use case also includes a step for "using" the data. The only discernable difference between *preparing* the data and *using* the data seems to be the software tools that are used in each step. Note that although "instrument data" is specified in the name of the use case, the steps described do not say anything about how data originating from instruments is different from any other sort of data.

Given this similarity, the architectural response to DA-3 is identical to the response to DA-2.

3.4 DA-4 HPC simulation data analysis

This use case is also very similar to DA-2 from an architectural point of view. Two variations are described: (a) a "post-processing" scenario in which the HPC simulation generating the data to be analyzed is completed prior to this use case beginning, and (b) an *"in situ"* scenario in which the data analysis happens on the same system as the simulation that generates the data.

The post-processing case is identical to DA-3 so we again refer to the response to DA-2 for this architectural response.

The *in situ* case is slightly different, but the difference is one of subtraction, not addition. Specifically, the data collection step (Step 2 in the use case description) is eliminated because the data is already available on the system where preparation and analysis are to occur. Consequently, the response to DA-2 can also be used for the *in situ* case, but the data collection step may be ignored.

3.5 DA-5 In-situ computational steering

The description of this use case was slightly garbled, so we have had to rely on personal experience with computational steering to interpret what was meant. We believe that Step 5 of the use case description was intended to be something similar to the following.

5. The XSEDE user monitors the activity of the running job and inspects the simulation diagnostics or other intermediate results, possibly using interactive computational resources to assist in interpreting the intermediate results.

We furthermore assume that Step 6 is meant to imply that when the user "modif[ies] parameter file(s) *in situ* as response to diagnostics evaluation," the running simulation job becomes aware of the parameter changes and adjusts its behavior for the remainder of its execution.

Given this understanding of the use case, we propose that Steps 3 (creation of the parameter file) and 6 (modification of the parameter file) are actually instances of the data collection/gathering step in the previous use cases (see Step 2 of DA-2, for example). The parameter file could be transferred to the XSEDE resource via the transfer interfaces (§3.2.1) or

created on a remote system (perhaps the user's desktop system) and made visible on the simulation system via the virtual file system interface (§3.2.2). If jobs executing on the simulation system can see files on the system's interactive partitions, an even simpler method would be to simply create the file locally on the simulation system using interactive access ([UCCAN-4]).

In Step 6 of this use case description, the end user needs a mechanism by which he or she can change the simulation's parameter file, at which point it is implied that the simulation will change its behavior for the remainder of its execution.

If the simulation running on the system can access files on the interactive parts of the system (where the user logs in and prepares jobs for execution), then the simulation code may be written to periodically check for changes to the parameter file. This, with no involvement from the XSEDE system architecture, would be the simplest way to implement Step 6.

If the simulation running on the system can not access files on the interactive parts of the system, then the end user needs a mechanism for notifying the running simulation of a change to the parameter file. The XSEDE architecture provides one access-layer mechanism for interfacing with executing jobs: the EMS (execution management system) offers the ability to signal a running job. This feature is accessed via the UNICORE and/or Genesis II command-line interfaces. (See [UCCAN-1].) This feature is highly dependent on the EMS implementation provided by the resource on which the simulation executes, and its behavior and quality attributes may vary from resource to resource. This feature does, however, allow the end user to alert the running simulation job of the fact that the parameter file has been changed, at which point the simulation code can reload its parameters from the updated file and proceed with modified behavior.