XSEDE Architecture Overview

A Level 1 and 2 Decomposition for Software Developers and Integrators

Felix Bachmann
Ian Foster
Andrew Grimshaw
Dave Lifka
Lee Liming
Morris Riedel
Steve Tuecke

01 September 2014

Version 2.0
Table of Contents

A. Document History ................................................................................................................................... iii
B. Document Scope ...................................................................................................................................... iv
C. What is XSEDE and how does it work? ................................................................................................. 1
D. The XSEDE system architecture from 50,000 feet ............................................................................. 3
  D.1. Access-layer interfaces ....................................................................................................................... 4
  D.2. Service-layer interfaces ....................................................................................................................... 5
  D.3. Resource-layer interfaces ..................................................................................................................... 6
E. XSEDE’s system-wide functions ........................................................................................................... 7
  E.1. Identity management ............................................................................................................................ 7
  E.2. Interactive login ................................................................................................................................... 8
  E.3. Accessing remote files ......................................................................................................................... 8
  E.4. Submitting and managing computations .......................................................................................... 9
  E.5. Transferring datasets .......................................................................................................................... 10
  E.6. Discovering and providing resource information ............................................................................. 11
F. Satisfying specific research activities .................................................................................................... 12
G. References ............................................................................................................................................... 13

List of Figures

Figure 1: XSEDE architecture layers and system-wide functions .............................................................. 3
Figure 2: Architectural responses to XSEDE use cases ........................................................................... 12
# A. Document History

<table>
<thead>
<tr>
<th>Relevant Sections</th>
<th>Version</th>
<th>Date</th>
<th>Changes</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entire Document</td>
<td>2.00</td>
<td>09/01/2014</td>
<td>Rewritten based on XSEDE use cases</td>
<td>L. Liming (ed.), <em>et al</em></td>
</tr>
</tbody>
</table>


B. Document Scope

This document presents an overview of the XSEDE architecture. XSEDE has defined this architecture in concert with broad stakeholder consultation to understand the use cases and requirements of our researchers and scientists. First, we provide a brief review of XSEDE and how researchers and scientists use XSEDE. Next, we identify and describe six XSEDE system-wide functions: identity management, interactive login, remote file access, submitting and managing computations, transferring datasets, and discovering and providing resource information. Each function is supported by three principal layers of the architecture: the access layer, which facilitates the use of resources via a range of end-user-oriented and programmer-oriented interfaces; the service layer, which negotiates access to resources via well-defined network protocols; and the resource layer, which encompasses the diverse physical and virtual resources provided by XSEDE service providers. Finally, we identify and describe the specific interfaces provided by the access layer to enable each function. More detailed specifications of the interfaces provided in the access and service layers are provided in a separate document, XSEDE Architecture Level 3 Decomposition. To bring the document full circle, the final section provides references to a series of more detailed documents that spell out how the architecture implements each of the known use cases for the XSEDE system.
C. What is XSEDE and how does it work?

Scientists, engineers, social scientists, and humanists around the world—many of them at colleges and universities—use advanced digital resources and services every day. They use supercomputers, collections of data, and new tools to make our lives healthier, safer, and better. These resources and services are designed by data and computing specialists to satisfy particular scientific needs.

The XSEDE system is an integrating framework for these advanced digital resources, making them easier to use and helping more people use them. As individual resources come and go due to evolving technologies, and as the needs of researchers change over time, XSEDE provides a framework that makes it easier for researchers to gain access to the services they need and move their work to and from them.

The XSEDE system supports a wide range of research activities. These activities define the **use cases** for our system architecture. [1]

- Federating with and accessing other identity, data, and compute resource providers nationally and internationally, e.g., InCommon.
- Using XSEDE to power scientific applications used by many researchers
- Using XSEDE to analyze the data generated by large-scale scientific instruments
- Analyzing and understanding scientific data, especially at large scales
- Creating visual representations of scientific data to aid in understanding
- Performing computation that requires large amounts of storage and calculations
- Accomplishing many computation tasks in as little time as possible
- Automating complicated or repetitive scientific data production and analysis processes
- Storing, describing, distributing, and maintaining scientific data

The XSEDE architecture enables all of the above activities by offering six general-purpose **system-wide functions**. Each activity uses one or more of these functions as part of its overall strategy for accomplishing the activity. Most activities use several functions in concert.

The first function that the XSEDE system provides is the ability to define system-wide user identities, groups, and allocations. All XSEDE resources and services recognize these definitions.

- Each person who interacts with XSEDE maintains a **user identity** that consists of a unique name and a profile containing information of interest about him or her. To interact with the system, one authenticates to one’s user identity and obtains credentials that provide access to XSEDE resources and services.
- A **group** is a named set of user identities. Groups are used to apply actions or access control rules to many user identities at once and can also be used to obtain credentials.
- An **allocation** is permission to use a defined portion of the XSEDE resources and services over a defined period of time. Allocations are requested via application by researchers and are defined and assigned to groups of users by XSEDE personnel.
Having a common set of user identities, groups, and allocations, the XSEDE system provides services and software interfaces to accomplish the following additional system-wide functions.

- Create an interactive login session.
- Access remote files using a POSIX-style (open/read/write/close) interface.
- Submit and manage computations.
- Transfer datasets to and from resources.
- Publish, update, discover, and subscribe to resource information.

These core functions are used as the building blocks to enable the scientific user activities listed above. In each of the functional aspects above, users are equally concerned with non-functional aspects such as usability, reliability and availability. Our engineering process requires us to address these issues in the design of each system-wide function.
D. The XSEDE system architecture from 50,000 feet

Before we look at each function in detail, we need to be clearer about the kinds of interfaces that XSEDE offers. Once that is clear, we can describe each of these system-wide functions in more detail in the next section of this document.

XSEDE’s system architecture provides a framework within which the XSEDE resources and services fit together and have a sense of coherence for the researchers who use them. While each individual XSEDE resource is designed to satisfy its user needs as fully as possible, XSEDE’s system-wide functions support a more basic user need: the need to have a relationship with the collection of resources including SP resources as a whole and to maintain that relationship over time as their specific data and computing needs change and as system components change.

![Figure 1: XSEDE architecture layers and system-wide functions](image)

All of XSEDE’s system-wide functions are designed using a three-layered approach. Figure 1 shows these layers and how they relate to the system-wide functions.

It is important to understand that the components that reside in any layer may make use of interfaces provided by any other layer. This diagram doesn’t show “uses” or “provides” relationships; it is merely an organizational framework for identifying the interfaces that are available and their overall characteristics. For example, science gateways that reside in the access layer and that provide thin-client GUI interfaces to scientists can use other access-layer interfaces, service-layer interfaces, or resource-layer interfaces to engage the XSEDE system. Nothing prevents a service-layer component that provides standard protocol access to a resource from using an access-layer interface to simplify part of its work.
Each layer has a distinct model for definition, deployment, and support. Most generally: the access layer is defined, implemented, and supported by the XSEDE organization as a whole; the service layer is defined and coordinated by XSEDE but is deployed by XSEDE and other service providers (SPs) with support coordinated between XSEDE and the SPs; the resource layer is defined, implemented, and supported exclusively by SPs.

All three layers are available to XSEDE end users. The differences—and the trade-offs involved—have to do with how much specialization the user needs to be aware of across XSEDE resources and who the user relies on for technical support. The lower the user goes, the more directly he or she is working with specific resources and specific SPs. The primary “value added” of the higher-level access- and service-layer interfaces is to provide an interaction model that remains consistent despite changes in resource availability and changes in user needs.

Researchers with specific, short-term relationships with a particular XSEDE resource often find it easiest to work directly with the resource-layer interfaces offered by the SP. Scientists who know they'll be using XSEDE longer than the lifetime of an individual resource (three to five years), and who therefore expect to migrate between resources several times during their work, may choose to focus on access-layer interfaces because they reduce their need to adapt to new interfaces with each migration. However, it may still be necessary to use service- or resource-layer interfaces for some functions if the access-layer interfaces don’t match the user’s requirements.

The choice of layer to use can be made separately—for the most part—for each system-wide function. For example, one could choose to use access-layer interfaces for identity management, interactive login, remote file access functions, and transferring files and datasets between resources, while preferring resource-layer interfaces for submitting and managing computation jobs and obtaining resource information. For the former functions, the user can expect things to work the same across any XSEDE resources he or she uses. For the latter functions, the user expects to work closely with the SP for each resource he or she uses to understand the system details and resolve any issues.

As another example, one could choose to use service-layer interfaces—using or developing one’s own clients for standard protocols such as SSH, GridFTP, OAuth, or BES—for some functions. This might be attractive to people who are already heavy users of other systems that use these protocols, because it could make XSEDE appear similar to the systems with which they’re already familiar and for which they have greater support.

D.1. Access-layer interfaces

The access layer provides user interfaces and application programmer interfaces (APIs) chosen specifically by XSEDE to provide system-wide functions. It is geared mainly for users who know that they will be using multiple XSEDE resources during their work and who prefer consistent interactions with the system despite changes in resources and changes in their own needs. The access-layer interfaces are supported by the XSEDE organization ([support@xsede.org](mailto:support@xsede.org)) and are available via the XSEDE User Portal (XUP) [2].
XSEDE’s access layer provides the following three kinds of interfaces for each system function.

- A graphical user interface (GUI) provides a point-and-click or touch interface using graphical interface elements like checkboxes, buttons, and pop-up or pull-down menus;
- A command-line interface (CLI) allows users to access functions via a text-based interface, enabling shell scripts and high-level workflow tools; and
- An application programmer interface (API) supports integration with applications developed in Java, Python, C, and other programming languages.

Each of these interfaces may be available in one or both of the following forms.

- Thin-client interfaces are accessible via standard software that is typically available on standard systems (Windows, Macintosh, Linux), such as a Web browser or secure shell (SSH) client. The user may then, for example, simply point a built-in Web browser at a particular web address. XSEDE’s thin-client GUIs include the XUP, Globus [3][4][19], and many science gateways. [5] The Globus CLI and XSEDE’s remote login CLI are examples of thin-client CLIs.
- Thick-client interfaces require software to be installed that normally would not be present on a standard system. For example, the Genesis II GUI and the UNICORE 6 Rich Client (URC) [6] are thick-client GUIs. The Globus Toolkit CLI (e.g., globus-url-copy) [7] and the Genesis II grid shell [8] are examples of thick-client CLIs. An example of a thick-client API is the Simple API for Grid Applications (SAGA) bindings for the C, Python, or Java programming languages. [9]

D.2. Service-layer interfaces

The service layer provides standard protocols supported by XSEDE resources. The software that implements the service layer is installed on XSEDE resources by SPs for use by users and by access-layer software (thick- and thin-client interfaces). They may be installed on individual resources in the resource layer, on central (shared) XSEDE systems, or even on commercial hosting services contracted by XSEDE. The XSEDE Architecture Level 3 Decomposition [10] details the XSEDE services and their capabilities.

The XSEDE organization coordinates the selection, recommendation, and acquisition of the software described in the service layer, but it is ultimately installed and configured by SPs to fit their specialized resources. There are often variations in the way the protocols are supported by each resource. XSEDE’s access-layer interfaces are designed to mitigate these differences, but users who go directly to the service layer may need to be aware of them.

The access layer is also designed to mitigate faults and failures in the service layer. 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. Some service-layer interfaces provide this kind of resiliency, but not all do.
Because the service layer is specified by XSEDE and deployed and operated by L1-L3 SPs, the support for service-layer interfaces is collaborative. End users may seek help from the XSEDE organization or the individual SP, and in either case, the appropriate parties will resolve issues.

The software that XSEDE recommends to SPs for implementing the service layer is available to other system providers as well, such as campus IT operators or mid-sized research collaborations. Using this software, any specialized resource can be made accessible via XSEDE’s access-layer interfaces, assuming policy agreements are arranged. For many of these products, XSEDE provides downloadable packages and XSEDE-specific configuration documents. At a minimum, XSEDE documents each recommendation in sufficient detail to replicate the recommended deployment.

**D.3. Resource-layer interfaces**

The resource layer comprises all of the advanced computing resources that can be accessed via XSEDE’s system-wide functions. These resources are designed and operated by the SPs. In addition to national-scale HPC systems, they also include a growing set of campus-based resources (e.g., institutional data repositories and specialized computation services) and services offered by large- and mid-scale research collaborations (e.g., community data repositories and specialized services for specific research fields).

Each XSEDE resource provides its own suite of user interfaces and APIs as its contribution to the resource layer. These are tailored to the purpose of the resource and may or may not overlap with XSEDE’s access and service layers. The service-layer software on each resource is configured to use the resource’s interfaces to the degree possible and is specifically designed to mitigate the differences between resource interfaces.

Scientists with short-term (less than five-year) relationships with a single resource are the most likely to prefer resource-layer interfaces. These scientists are not likely to use other XSEDE resources and hopefully won’t need to migrate to a new resource during the course of their work. They are thus unlikely to encounter the differences in resource-layer interfaces on various resources, and would not benefit greatly from the service or access layers.

Each XSEDE SP supports its own resource-layer interfaces—usually directly and without XSEDE involvement. One typically won’t need to establish an XSEDE identity to use these interfaces, as the SP will establish an identity for each user. Some SPs also allow the use of existing identities such as campus IDs or other federated ID systems.

XSEDE’s information services provide a directory of current XSEDE resources.
E. XSEDE’s system-wide functions

Now that we have defined the types of interfaces that are available on XSEDE, we can explore the interfaces offered in the access layer for each of the six XSEDE system-wide functions. This section describes the specific access-layer interfaces for each system-wide function.

E.1. Identity management

The first step in establishing a relationship with the XSEDE system is creating an XSEDE user identity.

Every XSEDE user establishes an XSEDE identity via the XSEDE User Portal (XUP), a website accessible via any standard Web browser. [2] This identity consists of a user ID (unique for each person) and a password. Once established, the XUP provides a variety of other identity features, such as maintaining a public profile. For example, the user may link his/her XSEDE identity to others, such as a campus identity. Once linked, the user may use any linked identity to login to XSEDE and gain access to XSEDE system-wide functions.

Most advanced computing functions require authorization, which is granted by the operators of each XSEDE resource. (Merely having an XSEDE identity doesn’t allow the user to do much.) The XUP also provides the user interface for requesting authorization to use XSEDE resources and for managing access to the resulting “allocations.” XSEDE users may also define their own user groups (a group of people working together on a research project, for example) that can be used to control access to remote files or to transfer datasets throughout the system. The interface for creating and managing user groups is also provided via the XUP.

The XUP should be sufficient for individual user needs.

Application and service developers who are working on campus bridging, science gateways, connecting instruments, etc. may require CLI, GUI, or API access to XSEDE’s identity services, and XSEDE provides all of these.

The Globus Nexus CLI is a thin-client CLI accessed via a standard SSHv2 client provides access to XSEDE’s group management functions. The Genesis II client includes a thick-client CLI aimed mainly at obtaining XSEDE user and group credentials. The Genesis II client also includes a GUI for accessing XSEDE identity management functions.

Finally, for application developers and system integrators, XSEDE offers a wide range of APIs for identity and group management, including:

- OAuth - A widely used interface for sharing user identity and authorization information between systems [11]
- Nexus REST - The interfaces used for XSEDE’s group management functions [12]
• WS-Trust STS - A standard Web services interface for translating one set of credentials into another set based on access rules [13]
• MyProxy - A “legacy” interface used to obtain an X.509 credential that can be used to access some XSEDE system-wide services [14]

E.2. Interactive login

Since the dawn of computing, the most basic human interface with computers has been the interactive command session. The text-based interface, consisting of prompts, commands, and output, is still common when interacting with advanced computing systems.

The XUP provides a thin-client (web browser) interface for establishing a login session on authorized XSEDE resources. The user logs into the XUP, navigates to the “My XSEDE” section of the portal, and views a list of resource where he or she has a current allocation. Clicking a link opens a new window in which a login session runs.

XSEDE also provides a thin-client remote login interface that uses a standard SSHv2 client. [15] (Most Unix-based systems—including the Apple Macintosh—have ssh clients pre-installed, and many free ssh clients are available for Windows systems.) Using a standard SSHv2 client, the user opens a connection to the XSEDE login hub (login.xsede.org) and provides his or her XSEDE userid and password. The user can then connect to any XSEDE resource on which he or she has an allocation.

E.3. Accessing remote files

Most computer applications expect data, commands, and configuration settings to be contained in objects called files. (Database-driven applications are the primary exception.) A file system provides ways to name, locate, and access these objects. The file and directory interfaces described in the POSIX family of standards [16] are widely used. (Windows, Linux, and Macintosh all provide POSIX-compliant APIs for file-system access.) A key function for making XSEDE easier to use is the ability to access and manipulate files from anywhere with an Internet connection (e.g., SP login nodes, campus, lab, laptop, and home computers) as long as the files are stored anywhere that allows incoming connections (e.g., SP file systems, campus, lab, and desktop file systems). There are two mechanisms for achieving this capability, the XSEDE Wide File System (XWFS) and the XSEDE Global Federated File System (GFFS).

The XWFS is a tightly coupled file system that requires participants to trust each other at the root level; i.e., complete and total trust. The XWFS is implemented using the IBM GPFS file system [17] and is deployed at several L1 SPs. When deployed at a site, GPFS is mounted at a well-known location. Users can then transparently access files and directories in GPFS (subject to authorization) as if they were local.

The GFFS is a federated file system. Rather than assuming a high degree of trust and requiring that sites use a particular file system, the GFFS allows the owner of a file system to export portions (rooted directory trees) of his or her local file system into a secure directory structure. There is no
pre-existing trust between client and server systems. All access is not only encrypted, but more importantly, authentication is based on strong cryptographic protocols ensuring that only trusted and authenticated clients can access the data.

The data exported into the GFFS can be stored in any native file system. The GFFS supports Windows, Linux, Luster, GPFS, and even hierarchical file systems that include tape storage. Authorization to access the exported data is based upon XSEDE user and group credentials: the client does not need to have an account on the server where the data resides.

In addition to exported file systems, the GFFS makes other resource types available for use. Compute and identity resource owners can export their compute queues, identity management services, and storage into the GFFS directory structure as well. Clients can then securely access these resources via the GFFS.

To access the GFFS, XSEDE provides the Genesis II thick-client access layer, which can be installed on Windows, Linux, and Macintosh systems. Once installed, Genesis II provides CLI, GUI, and API mechanisms that can be used to authenticate to XSEDE and obtain an identity credential and/or group credentials, which are used to access resources. Users can access GFFS resources using the CLI, API, or GUI as described above. On Linux systems, users can mount the GFFS directory structure using a GFFS-aware FUSE file-system driver. Once mounted, remote resources can be accessed as if they were local. This means that user applications and shell scripts that use the POSIX API built into the operating system (most do) will operate on remote data without modification. As one would expect, access to remote data will usually be slower than local access due to network delays.

E.4. Submitting and managing computations

Most of XSEDE’s advanced computing systems today offer conventional interactive login nodes with a CLI for submitting computation tasks (“jobs”) to the system. Researchers use the interactive login function (§E.2) to access the login node and they use the dataset-transfer or file-access functions to transfer their applications, data, and configuration items there. Finally, they use the queuing system CLI (a component in the XSEDE architecture’s resource layer – see §D.3) to initiate jobs and to optionally transfer the resulting data to other systems for subsequent analysis. This is the most common way for individual researchers to accomplish their HPC and HTC work.

Some of XSEDE’s interaction modes—such as science gateways, campus bridging, large-scale scientific workflow, and federation & interoperability with other systems—require remote or API access to this functionality. To support these activities, XSEDE provides a system-wide interface for remote job submission and management: the XSEDE Execution Management Services (EMS).

To remotely submit or manage a job on XSEDE, client software must “understand” the standard Basic Execution Services (BES) protocol [18]. While a user or software developer can develop their own BES clients, most will use an existing BES client such as the UNICORE 6 client or the Genesis II client discussed above. Genesis II offers both GUI and CLI tools for defining, submitting, and managing jobs. UNICORE 6 provides an API.
Submitting a job consists of: i) defining the job in detail using JSDL (Job Submission Description Language), which can be provided as ASCII text or constructed graphically using the GUI; ii) obtaining a credential to authorize the work on the computing resource; and iii) instructing the Genesis II client to submit the job to one of XSEDE’s computing resources or to a global XSEDE queue that matches job requirements expressed in the JSDL to resources.

Compute resources offer the BES interface for submitting individual jobs or sets of jobs (i.e., parameter sweep jobs) and for managing their lifetimes. BES interfaces are available for most L1 SP resources. The BES resources are linked into the GFFS directory structure and accessed using client tools as described above. Other resource owners, such as campus clusters or lab clusters, can also run BES services for their compute resources and link them into the GFFS for their own use. Access control to compute resources is controlled by the resource owner. For example, a campus might choose to allow any XSEDE individual or group to access their compute cluster.

In addition to the BES interfaces and services, EMS includes a GridQueue interface and service. GridQueues accept jobs just like BES services and can be associated with several BES services. Once submitted to a GridQueue, jobs are submitted to individual BES services in a first-come/first-served fashion, subject to matchmaking. (With matchmaking, a job will only be scheduled on a BES that has the resources—such as operating system and memory—that it requires.)

**E.5. Transferring datasets**

It is all-too-common for a researcher to realize that the perfect system for analyzing his or her data is available and ready to be used, but the input data is on another system: in the laboratory on campus, on a hard drive at home, on a server at a colleague’s institution, in the campus’s mass storage system. Moving big scientific datasets from one system to another can be a hassle. Even with modern network speeds, it can take hours, days, or even weeks to transport large datasets. And even on well-maintained systems, a single transient failure can stop a daylong transfer in its tracks. XSEDE is familiar with these issues and has a solution for researchers who need to move data from one XSEDE facility to another or between XSEDE and other systems. XSEDE’s solution is called Globus [3][4][19], and is available to all XSEDE users and their colleagues.

Globus can be used with a Web browser using its thin-client interface. Once logged into the XUP, users navigate to Globus and are presented with an interface for selecting the source and destination for their transfer. After selecting the data to be moved and identifying the destination, the user clicks a button. Globus takes it from there, sorting out the details, scheduling the activity, monitoring progress, and, eventually, sending email to the user when the transfer is finished. The user can request other transfers, check the status of previous or current transfers, or cancel transfers at any time. Best of all, the researcher can go about his or her other work, knowing that the data will be where it needs to be as soon as possible.

All XSEDE storage facilities are accessible by Globus, as are many of the major data systems and storage facilities around the country. To be accessible by Globus, a resource must support the GridFTP protocol [20], a component of the XSEDE architecture’s service layer. To make a system accessible to Globus, a researcher or system administrator may download and install the Globus
Connect [21] software. Globus Connect comes in two flavors: Globus Connect Personal is for personal systems (laptops, desktops). It runs in the background on Macintosh, Linux, and Windows systems and allows Globus to move data to and from the system for the researcher. Campuses with shared servers can install Globus Connect Server to make their servers accessible via Globus for their users.

For automation and lightweight integration with applications, Globus provides a thin-client CLI. The CLI is accessible via a standard SSHv2 client. (Most Unix-based systems and Macintosh systems have ssh clients pre-installed, and many free ssh clients are available for Windows systems.) Using a standard SSHv2 client, the user issues individual commands to the Globus CLI (cli.globus.org). Authentication is handled via SSHv2’s industry-standard mechanisms. Globus also offers a REST API for integration with applications.

**E.6. Discovering and providing resource information**

The resources to which XSEDE provides access are constantly changing. New resources are commissioned, old resources are decommissioned, resource-sharing agreements change, and individual systems are frequently upgraded. XSEDE’s information services provide interfaces for browsing and searching the current state of the system and all of the resources to which it provides access.

Researchers with Web browsers who wish to obtain status information can do so using the XUP. After logging in to the XUP, there are many system information and monitoring pages available, including the Resource Monitor in the My XSEDE section. More technical detail can be found at info.xsede.org, the thin-client GUI interface for XSEDE’s information services.

The Genesis II access software provides a suite of thick-client GUIs and CLIs that access and display XSEDE system information. Genesis II can be downloaded and installed on Mac, Windows, and Linux systems.

An interactive CLI called “xdinfo” offers CLI access to XSEDE information, including both browsing and searching functions. The tool can be downloaded and run on Linux systems and it is available on most XSEDE interactive login services.

Applications that need service-layer (network protocol) access to XSEDE information services can use REST, the Web Services Resource Framework (WSRF) [22][23], or any tools that support the AMQP 0.9.1 or 1.0 protocol [24]. REST interfaces typically cover search and update functions, while WSRF and AMQP typically cover publish and subscribe functions, though there is overlap.
F. Satisfying specific research activities

At the beginning of this document (§C), we listed ten research activities or "use cases" that the XSEDE system supports. These activities outline the things that researchers do on a day-to-day and year-to-year basis with XSEDE and its resources. The system-wide functions described in the previous section enable each of these activities in ways that can be carried on over time, despite the never-ending introduction and discontinuation of individual XSEDE resources and services.

Having identified these user activities, the next step is to explain in detail how the system-wide functions enable each activity. These explanations comprise a set of documents that we collectively call the "architectural responses." Figure 2 identifies the specific architectural response to each XSEDE user activity. Each response is a separate document.

<table>
<thead>
<tr>
<th>XSEDE use case</th>
<th>Architectural response(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Federating with and accessing other identity, data, and compute resource providers nationally and internationally, e.g., InCommon</td>
<td>Campus bridging, Federation and interoperability</td>
</tr>
<tr>
<td>Using XSEDE to power scientific applications used by many researchers</td>
<td>Science gateways</td>
</tr>
<tr>
<td>Using XSEDE to analyze the data generated by large-scale scientific instruments</td>
<td>Connecting instruments</td>
</tr>
<tr>
<td>Analyzing and understanding scientific data, especially at large scales</td>
<td>Data analysis</td>
</tr>
<tr>
<td>Creating visual representations of scientific data to aid in understanding</td>
<td>Data visualization</td>
</tr>
<tr>
<td>Performing computation that requires huge amounts of storage and calculations</td>
<td>High performance computing (HPC)</td>
</tr>
<tr>
<td>Accomplishing many computation tasks in as little time as possible</td>
<td>High throughput computing (HTC)</td>
</tr>
<tr>
<td>Automating complicated or repetitive scientific data production and analysis processes</td>
<td>Science workflow</td>
</tr>
<tr>
<td>Storing, describing, distributing, and maintaining scientific data</td>
<td>Data management</td>
</tr>
</tbody>
</table>

*Figure 2: Architectural responses to XSEDE use cases*
G. References


