XSEDE Canonical Use Cases

March 21, 2013

Version 1.0


XSEDE
Extreme Science and Engineering
Discovery Environment

Table of Contents

A Document History

Overall Document Authors:

Altaf Hossain
Pittsburgh Supercomputing Center (PSC)
Pittsburgh, PA
altaf@psc.edu

Ian Foster
The University of Chicago and Argonne National Laboratory
Argonne, IL 60439
foster@anl.gov

Morris Riedel Jülich Supercomputing Centre
Forschungszentrum Jülich GmbH
D-52425 Jülich
Germany

Andrew Grimshaw
University of Virginia
PO Box 400740
Charlottesville VA 22904
grimshaw@virginia.edu

David Lifka
Cornell University
512 Frank H. T. Rhodes Hall
Ithaca, NY 14853
lifka@cac.cornell.edu

	Version	Date	Changes	Author
Revised draft	1.0	04/25/2013	Ready for public facing XSEDE wiki. First 1.0 version	Hossain
First formatted draft	0.1	04/08/2013	Compiled first standard format.	Hossain

B Document Scope

This document is both a user-facing document (publically accessible) and an internal working document intended to define user needs and use cases within the overall activities of XSEDE. The definition of use cases is based on a template from Malan and Bredemeyer[1]. In general it is in keeping with the approaches and philosophy outlined in "Software architecture in practice."[2]

This document is one component of a process that generates at least the following documents, some of which are user-facing, some are as of now intended to be internal working documents:

- ***This document*** - A description of use cases [User facing]
- A set of level 3 decomposition documents, which include:
    - Quality Attributes descriptions
    - Connections diagram in UML

The use cases are presented here using the following format, derived from the Malan and Bredemeyer white paper[1] as follows:

Use Case	Use case identifier and reference number and modification history
Description	Goal to be achieved by use case and sources for requirement
References	References and citations relevant to use case
Actors	List of actors involved in use case
Prerequisites (Dependencies) & Assumptions	Conditions that must be true for use case to be possible Conditions that must be true for use case to terminate successfully
Steps	Interactions between actors and system that are necessary to achieve goal
Variations (optional)	Any variations in the steps of a use case

[1] Malan, R., and D. Bredemeyer. 2001. Functional requirements and use cases. *www.bredemeyer.com/pdf_files/functreq.pdf*
[2] Bass, L., P Paul Clements, and Rick Kazman

Quality Attributes	
Non-functional (optional)	List of non-functional requirements that the use case must meet
Issues	List of issues that remain to be resolved

C Definitions

Purge: Clear out session directories, job records, etc.
Session directory, a.k.a. sandbox, a.k.a. job working directory.
Client file system: A file system (e.g. on a desktop or campus) that may be the source of files needed for remote execution and/or the place where results from the remote execution may be placed upon completion.
Third party file system: A file system that may be the source of files needed for remote execution and/or the place where results from the remote execution may be placed upon completion.
File system: A file system accessible via a file access service, that may be the source from which files are to be moved and/or the destination to which files are to be moved.
File system authorization policy: Determines who may access files on a particular file system.
File transfer service: Receives and processes requests to initiate, monitor, and control transfers between file systems.
Remote storage server: A remote storage server can accept create, read, update, and delete operations on files and directories.
File access service: Receives and processes requests to create, destroy, read, and write files.
Directory access service: Receives and processes requests to create, destroy, read, and update directories.
Compute resource: A computer system on which the job actually runs.
Compute resource File System: A file system visible from the compute nodes where the job executes.
Compute resource authorization policy: Determines who may execute on a particular compute resource.
Execution service: Sits in front of the compute resource and is used to submit and manage jobs on the compute resource.
Accounting system: Reports resources consumed by job execution on computing resources.
A valid file transfer request: consists of a source and destination that exist, where the target has sufficient disk space, where the user has permission to read the source and write the target, and where there exists a communication path from the source to the target.

D Canonical Use Cases

Use Case UC CAN 1	Run a remote job
Description	A user executes and manages a job (sequential or parallel) on a remote compute resource.
References	
Actors	The user that wants to execute an application remotely.
Prerequisites (Dependencies) & Assumptions	The client is properly authenticated. The client has generated a job description in the appropriate format. The client already knows and has the address of the execution service that is to be used to run the job. The compute resource is able to execute the client's application, e.g., there is an appropriate binary, there is sufficient memory, etc. The compute resource does not fail during job execution. The underlying resource management system is at least as reliable as the requirements for the execution service and performs at least as well as the execution service requirements. In other words, to meet execution services quality attributes requires certain quality attributes from

	the resource management system and the compute resource. All data needed by the job are already on the compute resource file system, and all results are left on the compute resource file system.
Steps	The client sends the job description to the execution service and receives some form of job identifier in return. The execution service begins execution of the job on the compute resource. The client polls the execution service periodically to determine whether the job has completed or failed. The client purges the job from the execution service.
Variations (optional)	a) Data needs to be copied in before execution and/or out after execution, to/from a client file system and/or third party file system. b) Remote data needs to be accessed during application execution. c) The specific compute resource is not known *a priori*, and either must be found by the client or by a third party. d) The client is not authorized, using its credentials alone, to use a resource, but is a member of a community or group that is authorized to the compute resource. e) The client can register with the execution service for notifications of job state change. f) The client can interact with the session directory on the compute resource file system before, during, and after job execution by the compute resource. g) At most once semantics, meaning that if the client submits the same job request more than once (e.g., because an earlier submission was not acknowledged), the job will be executed at most once. Note, some sort of unique job identifier must be presented by the user. Further, an

	interval over which the execution management system must guarantee uniqueness must be defined. h) Submission, status checking, and job control operations work on sets of jobs. i) The client can submit a set of jobs (e.g., a parameter sweep) in a single request, and then monitor and control them as a single job.
Quality Attributes	a) Any request to the execution service is acknowledged within one second. [source: campus bridging.] b) High throughput quality attribute. There are three metrics of interest: 1) the rate that jobs can be submitted, 2) the number of active jobs, and 3) the total number of jobs the execution service can manage at a time. c) The execution service can support, without error, as many queued and active jobs as are permitted by its associated resource management system. [source: A&D members] d) Once a job completes, its status can be checked for at least 24 hours. [Note: This quality attribute needs to be verified with users. One can imagine this attribute being resource-specific, and being discoverable by the client.] e) Client request patterns that exceed the stated job submission rate, queued jobs, or active jobs are handled gracefully: e.g., by declining to accept further job submissions. [source: A&D members] f) The execution service can be restarted without loss of jobs at three Sigma (e.g., following failure or system administrator action the service can be restarted without job loss 99.9 probability). [source: A&D members] g) Valid jobs complete successfully at two Sigma. (Invalid jobs include jobs with application errors, jobs for which input files do not exist, and jobs for which the binary is

	missing.)[source: campus bridging] h) Availability of the execution service is 1.8 Sigma (minimum requirement). Note that this does not imply that the resource is executing jobs 98% of the time, rather that the execution service is available for operations such as status.[source: campus bridging.] i) If a compute resource associated with the execution service fails, then any job currently executing on that compute resource is reported as failed. [source: A&D] j) The job states must be consistent and well defined across all resources.[source: campus bridging]
Non-functional (optional)	
Issues	

Use Case UC CAN 2	**Transfer a file or files**
Description	The user transfers one or more files and/or directories from one file system to another.
References	
Actors	User: The user initiates the file transfer activity and presents a valid file transfer request. The client may be a human at a command line interface or Web browser, or an application, such as a gateway, shell script, or Java, C, or Python program.

Prerequisites (Dependencies) & Assumptions	a) The client knows the address of the file transfer service. b) The client is properly authenticated. c) The client has generated a file transfer request in the appropriate format, specifying the file(s) and/or directory(s) that are to be transferred, and the source and destination file systems. d) The source file(s) and directory(s) named in the request exist and the client is authorized to read them. e) There is sufficient space for those file(s) and directory(s) on the destination file system, and the client is authorized to write at the specified destination location. f) The file transfer service, source file system, destination file system, and intervening network do not fail during execution of the file transfer.
Steps	The client sends a file transfer request to the file transfer service and receives some form of request identifier in return. The file transfer service begins execution of the transfer. The client may poll the file transfer service periodically, during and after the transfer, to determine the status of the transfer: for example, the amount of data and number of files that have been transferred successfully, and whether the transfer has completed or failed. The transfer completes.
Variations (optional)	a) The client can register with the file transfer service to receive notifications of file transfer state change. b) At most once semantics, meaning that if the client submits the same file transfer request more than once (e.g., because an earlier submission was not acknowledged), the transfer will be executed at most once. c) The client can request that a source file or directory and a destination file or directory be

	synchronized, meaning that only files that differ between the source and destination are transferred.
Quality Attributes	a) Any request to the file transfer service is acknowledged within one second.[source: campus bridging] b) The file transfer service can support XSEDE-wide a job submission rate of up to 1 requests/second in aggregate from all users without error. [source: A&D] c) The file transfer service can support XSEDE-wide, without error, at least 1,000 active transfer requests (i.e., under management, could be pending, active, etc) requests. [Source: A&D] d) Once a file transfer completes, the client can check its status for at least one month. [Note: Source: A&D.] e) Client request patterns that exceed the stated request submission rate or total active request limits are handled gracefully: e.g., by declining to accept further requests.[Source: campus bridging] f) The file transfer service can be restarted (e.g., following failure or system administrator action) without losing track of file transfer requests that are queued or active at three Sigma.[source: campus bridging] g) Valid file transfer requests complete at two sigma. [source:campus bridging] h) The file transfer service will be available at three Sigma. [source: campus bridging] i) The file transfer service can restart a transfer that is interrupted by transient failures. [source: campus bridging] j) The combination of transfer efficiency and impact of failures and restarts provides efficiency that is at least as good as 50% of peak theoretically possible throughput of optimal network path and storage systems. [source: campus bridging] [The A&D team feels that this is a difficult to measure attribute, in

	particular, peak bandwidth of end-to-end system.] k) If a user accesses a file system for which access is not authorized, the error message returned should be consistent, meaningful, and helpful. [source: campus bridging]
Non-functional (optional)	
Issues	

Use Case UC CAN 3	**Remote File Access**
Description	The user creates, reads, updates, and deletes files and directories from one site at another using POSIX operations, e.g., create, read, write, unlink. Note that "site" can refer to an XSEDE SP, a local campus, a research lab, a desktop computer, or even a home computer.
References	References and citations relevant to use case: Global Federated File System.
Actors	User: The user initiates the file and directory access. Data owner: The user or system administrator who makes the remote data accessible.
Prerequisites (Dependencies) & Assumptions	a) The client is properly authenticated, and has required permissions. b) The client is familiar with Unix file commands, and applications use the standard POSIX file system commands. c) The file(s) and directory(s) being accessed exist. d) There is sufficient space for those file(s) and directory(s) on the destination file system. e) The file and directory services, source file system, destination file system, and intervening network do not fail during execution of the file

	transfer.
Steps	The user issues a POSIX compliant file system call on the remote file system.
Variations (optional)	a) The user accesses the remote file system or storage using command line tools. b) The user accesses the remote file system or storage using a GUI. c) The user accesses the remote file system or storage using an API other than POSIX. d) The authorized user changes the access control list on a file or directory.
Quality Attributes	a) Availability of the file and directory services is at three Sigma. b) Unauthorized access of a file or directory will result in an error message that is consistent, meaningful, and helpful. c) Changes to the remote file system are visible to users within 30 seconds. d) Consistency and update semantics must be clearly defined for the implementation. e) Disconnected operation is not a requirement.
Non-functional (optional)	
Issues	