Allocations Best Practices

Table of Contents

List of Figures

A. Document History

Relevant Sections	Version	Date	Changes	Author
Entire Document	1.00	07/14/12	Baseline	Jim Lupo

B. Document Scope

This document is intended to assist Campus Champions by outlining best practice activities when assisting new users with allocations.

C. [Document Body]

This is where the meat goes….

This is a reference to Figure for template purposes….

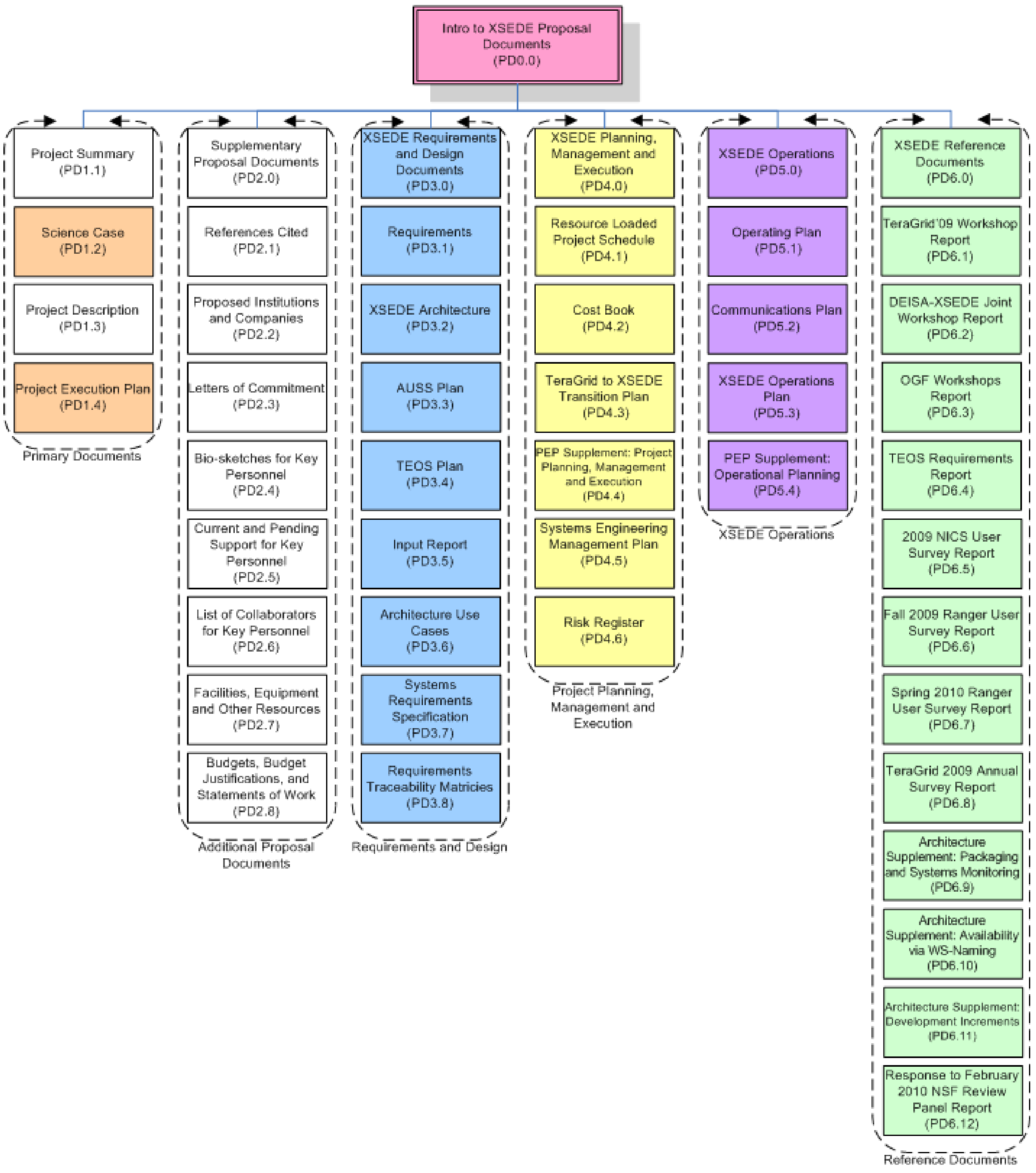
Intro to XSEDE Proposal Documents (PD0.0)
Project Summary (PD1.1)
Science Case (PD1.2)
Project Description (PD1.3)
Project Execution Plan (PD1.4)
Primary Documents
Supplementary Proposal Documents (PD2.0)
References Cited (PD2.1)
Proposed Institutions and Companies (PD2.2)
Letters of Commitment (PD2.3)
Bio-sketches for Key Personnel (PD2.4)
Current and Pending Support for Key Personnel (PD2.5)
List of Collaborators for Key Personnel (PD2.6)
Facilities, Equipment and Other Resources (PD2.7)
Budgets, Budget Justifications, and Statements of Work (PD2.8)
Additional Proposal Documents
XSEDE Requirements and Design Documents (PD3.0)
Requirements (PD3.1)
XSEDE Architecture (PD3.2)
AUSS Plan (PD3.3)
TEOS Plan (PD3.4)
Input Report (PD3.5)
Architecture Use Cases (PD3.6)
Systems Requirements Specification (PD3.7)
Requirements Traceability Matricies (PD3.8)
Requirements and Design
XSEDE Planning, Management and Execution (PD4.0)
Resource Loaded Project Schedule (PD4.1)
Cost Book (PD4.2)
TeraGrid to XSEDE Transition Plan (PD4.3)
PEP Supplement: Project Planning, Management and Execution (PD4.4)
Systems Engineering Management Plan (PD4.5)
Risk Register (PD4.6)
Project Planning, Management and Execution
XSEDE Operations (PD5.0)
Operating Plan (PD5.1)
Communications Plan (PD5.2)
XSEDE Operations Plan (PD5.3)
PEP Supplement: Operational Planning (PD5.4)
XSEDE Operations
XSEDE Reference Documents (PD6.0)
TeraGrid'09 Workshop Report (PD6.1)
DEISA-XSEDE Joint Workshop Report (PD6.2)
OGF Workshops Report (PD6.3)
TEOS Requirements Report (PD6.4)
2009 NICS User Survey Report (PD6.5)
Fall 2009 Ranger User Survey Report (PD6.6)
Spring 2010 Ranger User Survey Report (PD6.7)
TeraGrid 2009 Annual Survey Report (PD6.8)
Architecture Supplement: Packaging and Systems Monitoring (PD6.9)
Architecture Supplement: Availability via WS-Naming (PD6.10)
Architecture Supplement: Development Increments (PD6.11)
Response to February 2010 NSF Review Panel Report (PD6.12)
Reference Documents

Figure : Original document map.

User Assistance Best Practices

D. Allocations

D.1. The Campus Champion Allocation

As with any user of XSEDE resources, the first two things a new champion needs to do is request a user account and apply for their allocation. This will basically provided the *keys to the realm* so you can access the computation resources, and *currency* to spend on the services. This process is essentially identical to what every user may follow, though not every user needs their own allocation.

D.1.1. Process

Feel free to apply for an XSEDE user account at any time. This will give you access to the tools you'll eventual need. Hold off on starting the allocation request process until you receive your acceptance email. The email will introduce you to a few key XSEDE people, and prime the system to expect your allocation request.

> Both the user account and allocation requests start on the XSEDE user portal (https://portal.xsede.org/) , so visit it and bookmark it. You'll become quite familiar with it!
>
> The process details can be found under the sub-tab labled *Request Steps*. If you ever wanted the nitty-gritty, this is the page for you.

D.1.2. Appropriate Uses

Of the three allocation types, startup, education, and research, you're likely to deal most often with startup and education allocations. A statement of purpose for each is short and sweet.

D.1.2.1. Startup Allocation

The startup allocation provides SUs to new users so they can experiment on their own, or under the guidance of the champion. Example uses include:

- Gain familiarity with the HPC environment.
- Identify resources appropriate for a particular investigation (i.e. codes, processes, machine architecture, programming models, machine specific features, machine specific performance advantages).
- Run applications that may be related to research interests.
- Identify data management processes that may be required for production level runs.
- Experiment with new algorithms, be they scripts or programs.
- Initial program development.

Regression testing of existing programs.

General purpose processing in preparation for any other activity.

D.1.2.2. Education Allocation

An Education Allocation is very similar to a startup, with the exception that it specifies fixed start and stop dates. Possible uses include:

SU support for a semester long programming class.

SU support for application specific workshops or tutorials.

Note how each of these examples encompass a fixed time period during which an allocation would be valid.

D.1.3. Management

D.1.3.1. Assigning Users

Once again, the XSEDE User Portal is your friend. Users are added to an allocation from the *Add User* sub-tab under *My XSEDE.*

The champion should be the only one who adds users to a champion allocation.

Users who win startup and education allocations need to control the addition of appropriate users.

D.1.3.2. Dealing with Misuse

Any allocation holder is responsible for monitoring the use of their allocation. Keep in mind that once a user is assigned to an allocation, they are free to charge against it. That means if they are careless, they can quickly consume the entire allocation. The key implication is that once you start assigning users to your champion allocation, you need to pay close attention to how it is being expended. You have several tools to draw on:

Again, the XSEDE Portal is the go-to resource. In addition to assigning users, you can view reports of current allocation balance.

Review the weekly consumption report that is emailed out each week.

If a user should get out of control, the quickest remedy is to remove them from the allocation.

If for some reason your allocation is exhausted, you may repeat the allocation request process, but ask for a ***supplement*** to your existing allocation account. Be sure to explain what happened and what you'll try to do to prevent future inappropriate use (no way to guarantee how new users will behave, but you can suggest what you'll do differently).

D.2. User Allocations

D.2.1. Guiding Type Selection

Guiding a user in the preparation of an allocation request is rather straight-forward 3-step process. The actual steps taken may vary depending on what application is being considered and how much existing performance information is appropriate to the computations being considered:

Determined the resource needs based on the type of program to be run.

Determine the type of allocation needed to support the scale of the model being considered.

Determine the computational resource that best satisfies the processing needs.

Total run time estimates – will dictate SUs required and duration of needed runs.

Memory estimates – will constrain the achitecture appropriate for the program.

Data storage – will impact the processing steps in terms of data management. Any long term storage needs may constrain the systems that can be used or require separate requests for storage.

D.2.1.1. Assessing Computational Needs.

What programming model will be used? This question is applicable to both custom developed code, and for well established codes. The model used is one of the primary determination factors used in selecting a resource to run on.

Distributed memory (MPI or message passing programming).

Shared memory (OpenMP or threads)

Accelerated (GPU)

For custom programs, encourage the user to benchmark it's performance. This is especially true since demonstrating an understanding of how the proposed application performs for a specific problem is a major decision factor in the proposal review process.

How does processing time vary with problem size (i.e. weak scaling)?

How does processing time vary with increasing numbers of processors (i.e. strong scaling)?

For community codes, related the problem proposed to existing performance figures reported in the literature, if possible. One could perform problem specific benchmarks as well. (See Appendix A)

D.2.1.2. Proposal preparation.

What science is being done?

What methods are being used? This should concentrate on the numerical methods being implemented if a new code is being written, or make reference to the fact that well known community codes will be used, and that the methods are appropriate for the problem being considered.

Demonstrate an understanding of the performance behavior of the chosen application for the problem being posed. The user may need help understanding how many processors can be efficiently utilized by the program for the size of the problem, and the length of time required to reach a solution. A good estimate requires understanding:

Amdahl scaling behavior (strong scaling) – efficiency as nodes are increased at constant problem size.

Gustafson scaling behavior (weak scaling) – constant efficiency as work and nodes used are scaled up in lockstep.

Time required for code to make forward progess towards solution. That is, use a short run be used to form a reliable estimate of how long it will take to reach the full solution. This is a characteristic of the solution method used by a program.

It should be stressed that the proposals are not judged primarily for the science content. However the science value is weighed against the amount of time requested, rising as the requested allocation increases.

D.2.2. Guiding Resource Selection

Once the user's application is properly characterized, it becomes an easy task to match it against the available resources.

A modest problem using a well-known application is likely suited to any resource on which it is installed. If the user is comfortable with a particular resource, stay with it.

A large scale problem will require a close match with the application characteristics. Working through the proposal estimation of required resources is a major requirement for matching application to resource.

A. Program Characteristics

Application	**Model**	**Scaling**	**GPU**
VASP	SMP		
NAMD	MPI	Excellent Weak	Yes
Gromacs	MPI	Excellent Weak	Yes