

Grid Service Requirements for Interactive Analysis

D. Olson, J. Perl
16 September 2002

A document in preparation for PPDG CS11, Interfacing and Integrating Interactive Data Analysis Tools with the Grid and Identifying Common Components and Services

CS11 Home Page: <http://www.ppdg.net/pa/ppdg-pa/idadat/index.html>

Links to Related Information: <http://www.ppdg.net/pa/ppdg-pa/idadat/related-info.html>

Contents

1 Introduction	2
1.1 Purpose and Organization of this Document	2
1.2 Definitions	2
2 Four Models of Interactive Analysis on the Grid	3
2.1 Model 1: grid as black box	3
2.2 Model 2: real-time batch	4
2.3 Model 3: interactive batch	4
2.4 Model 4: pre-started analysis services	5
3 The Steps that a Physicist Takes in Performing an Analysis and the Requirements that These Steps Place Upon Grid Services	5
3.1 Select Data	6
3.2 Get Smaller Local Subset of Data	7
3.3 Inspect Smaller Dataset to Develop Cuts and Analysis Programs	7
3.4 Move Data	7
3.5 Choose Standard or Modified Versions of Code	8
3.6 Run analysis on small data subset	8
3.7 Retrieve results	9
3.8 Estimate resources for larger job	9
3.9 Negotiate availability or access to resources	9
3.10 Run analysis on large data set	10
3.11 Check status of analysis in progress	11
3.12 View results of analysis in progress	11
3.13 Suspend/resume analysis in progress	11
3.14 Abort analysis in progress	12
3.15 View results of completed analysis	12
3.16 Displays of selected individual events at varying levels of detail	12
3.17 Add refined data to data store	13
3.18 Share refined data with collaborators	13
3.19 Add tag data	13
3.20 Compare results	13
3.21 Calculate cross sections	13
3.22 Maintain audit trail (data provenance)	14
3.23 Security and access control	14
4 Conclusions and Outstanding Issues	14
4.1 Outstanding Issues	14

1 Introduction

While the initial uses of the grid have been in areas of production processing and simulation, an additional area of grid work, interactive analysis, is now under way¹. Current analysis tools (PAW, JAS, Root) have the user running on a single machine on data accessible to that machine. The dream of interactive analysis on the grid is that the user, with the push of a button, might move that job from the single node to a distributed environment, access more power and more data, and do it all seamlessly, so that his work is no harder on the grid than it was on a single machine.

Under the hood, it is not as easy to run interactive analysis on the grid as it is on a single machine. The same data may be replicated in many locations, competition for resources is much more complex, the number of higher-priority tasks than ones own is not automatically known, and therefore the best choice of how to execute a task is hard to determine. For these reasons, new forms of Grid services that are able to make reasonable choices among a range of possible job-execution strategies, autonomously or interactively, are needed. Decisions made by these services will be based on a more complete range of information about the Grid's current and future state, and may integrate user-Grid information exchanges as part of the decision process. Self-learning optimization algorithms may be required if the decisions are to be effective in some cases, such as large and complex Grids. This new class of Grid services must be used to solve these grid problems, to make these grid decisions.

1.1 Purpose and Organization of this Document

The purpose of this document is to indicate the requirements that interactive analysis will place upon grid services. We begin by discussing what we mean by interactive analysis in the grid context, showing several models of possible interactive analysis systems and highlighting their relationships to grid services. With these pictures of interactive analysis in mind, we then go on to discuss specific requirements on grid services, organizing our thoughts via a step-by-step description of the steps an end-user physicist takes in performing analysis. Finally, we extract from those requirements a list of significant issues to be addressed for interactive analysis to proceed in the grid environment. This paper will be further utilized by PPDG as input to project and activity planning both for the ongoing work in the second year, the third year project plan, and beyond. It takes a view across PPDG experiment end to end application and systems to identify cross cut common requirements and services that are needed by all.

1.2 Definitions

1.2.1 Interactive Analysis versus Batch Analysis (Production Processing)

Data analysis jobs in the grid may be broken into two groups: interactive analysis and batch analysis (production processing). The division between these two kinds of processing has to do with response time. An analysis job is interactive if it provides results or partial results in a sufficiently short time scale that the user can receive results or adjust the analysis in a single user session.

Following from this time scale issue is that interactive analysis cannot involve very time consuming data movements.

Thus while interactive analysis jobs may require many grid services, they should not require very time consuming file transfer operations (though an interactive user may discover that some bulk data needs to be moved before they can begin their interactive session).

¹ See for example <http://pcbunn.cithec.caltech.edu/GAE/GAE.htm>.

1.2.2 Catalog Service

In this document, the term "catalog service" refers to a resource that can provide information about what data is available on the grid. Underlying this resource may be a database or an updatable set of files (but to the application using this service, the underlying technology is an arbitrary choice).

2 Four Models of Interactive Analysis on the Grid

To better understand the place of interactive analysis on the grid, it is useful to review the various models by which such analysis jobs may utilize grid services. Every user, group or experiment will make its own choice of what analysis tool to use and so one might draw an infinite number of grid/analysis tool interaction diagrams, but these diagrams can be roughly divided into four groups, four ways a user could interact with grid services, each leading to a different type of grid user experience. Though the four diagrams are quite different, their requirements on grid services have significant overlap. A single set of grid services should be able to accommodate all four models.

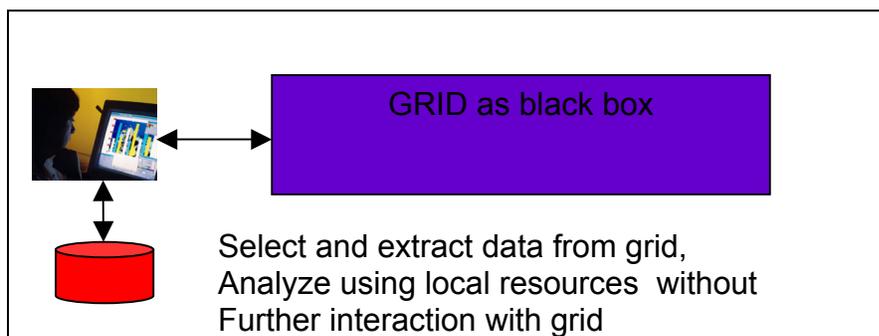
While the users will use their analysis tools of choice, grid enabled analysis environments need a number of new elements to enable effective use, including:

- System resource and utilization views to feed back to the user about the site and network states (load, current and projected performance) to support job submission strategies.
- Dialogs and services to determine and build successful task-execution strategies, given limited resources, taking the priorities and policies (of the group or experiment) into account
- Services and dialogs to assist with decisions in handling error conditions, where tasks may have to be re-scheduled or re-routed.]
- Dialogs and services to define a sequence of jobs with pre and post conditions, e.g., executing the same analysis with recalculation of the backgrounds or simulations between each one.

2.1 Model 1: grid as black box

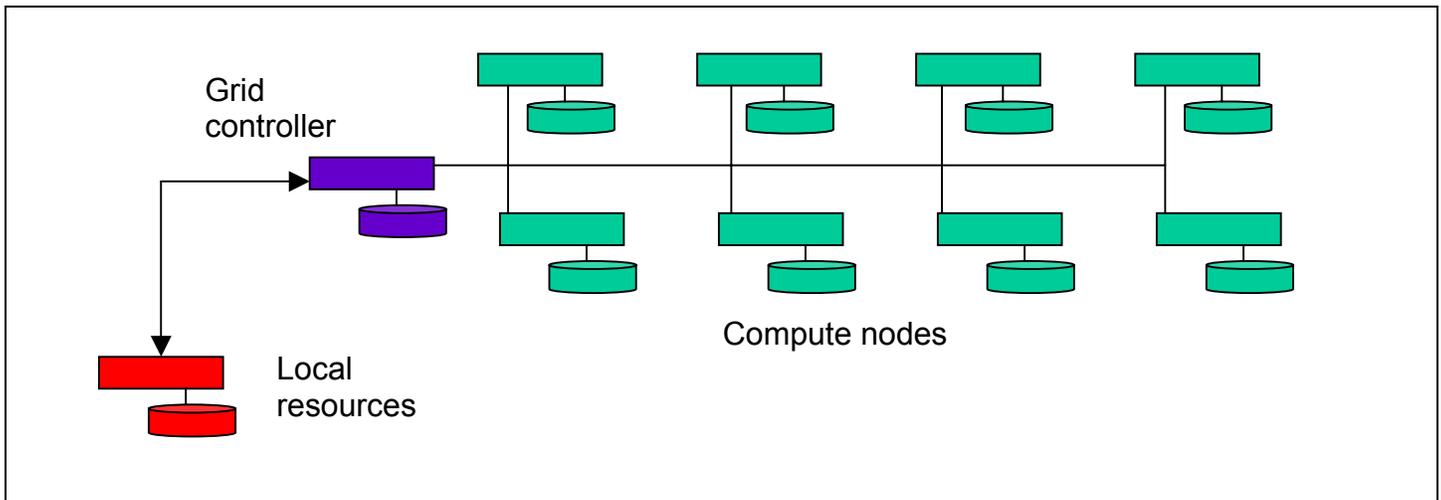
In this model the user performs queries for data from the grid, extracts data for storage on local non-grid resources and then carries out analysis of the extracted data without further interaction with grid services. The user is accessing

grid data catalog services and data extraction services, but these actions are entirely decoupled from the actual interactive analysis task. The analysis task does not use the grid.



2.2 Model 2: real-time batch

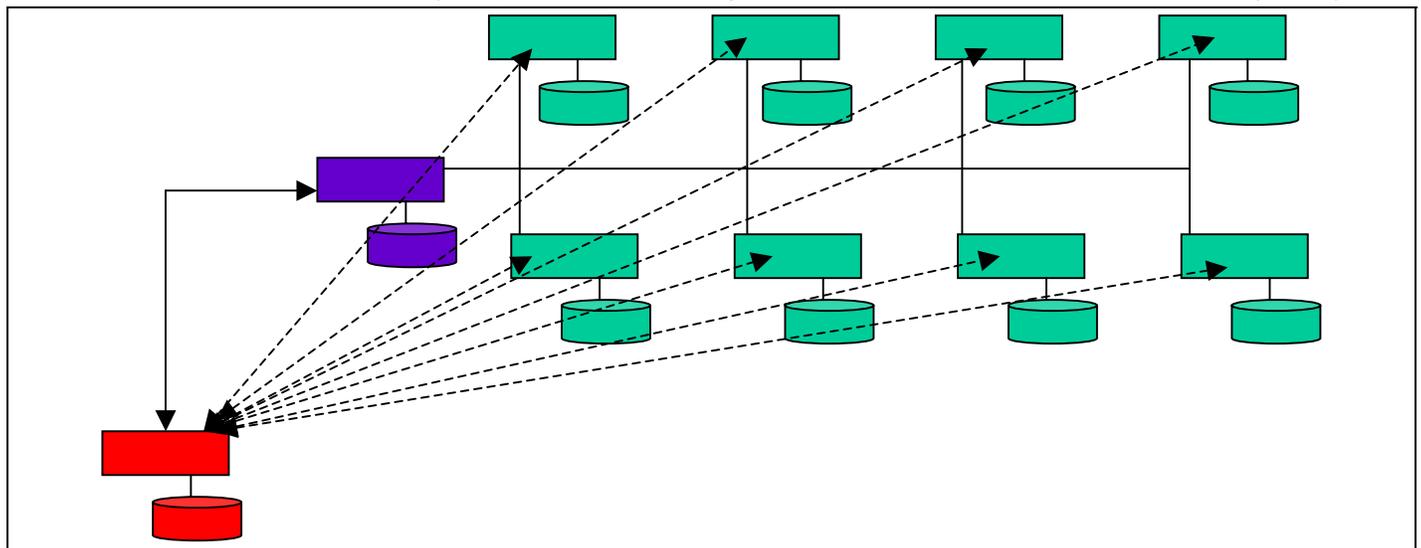
In this model work is submitted to a grid scheduler which distributes it across a number of nodes as



batch jobs. Intermediate results from the individual batch jobs are returned to the user for monitoring, but the only possible interaction with these batch jobs is to kill them.

2.3 Model 3: interactive batch

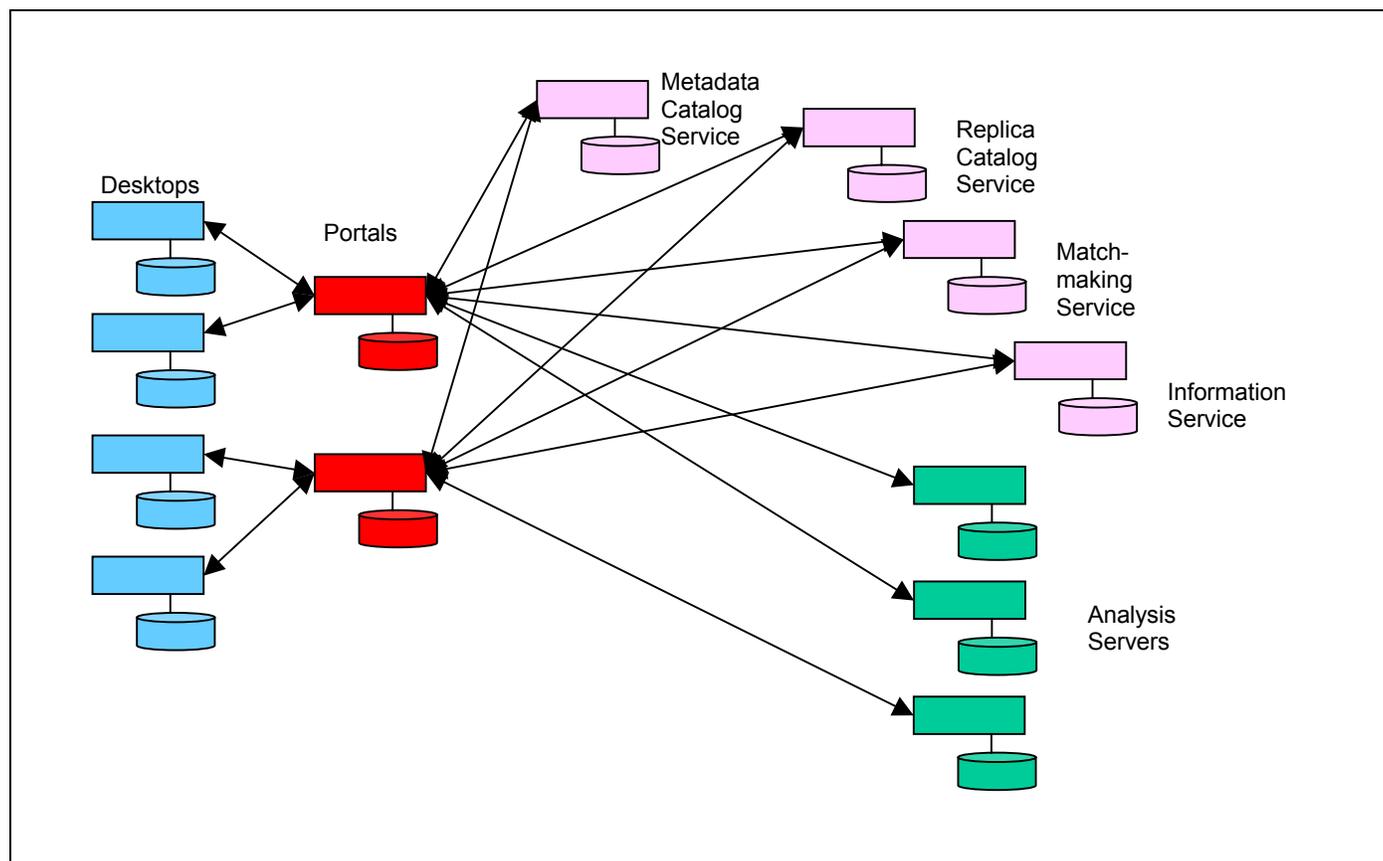
This model is similar to the real-time batch model, with the addition of logical control channels between the user and the individual analysis processes on the grid compute nodes. The user can modify analysis



jobs as they run and may have a rich desktop environment providing fine-grained control and feedback in near real time of the processing being carried out on each node. The lines in the diagram represent the control channels but are not meant to imply any particular implementation method. These control channels may be used for such functions as return of partial results, progress tracking (knowing when the task will finish), problem mitigation, knowing when task re-direction may be beneficial or receiving warnings of upcoming situations (a large increase in projected time to completion, upcoming resource pre-emption etc.).

2.4 Model 4: pre-started analysis services

In this model there are persistent (or pre-started) analysis services running on servers in the grid which are capable of executing some analysis tasks for users. The ROOT-Apache module (Carrot) or Java



Analysis Studio (JAS) servers are examples of these types of services. In the diagram, a users analysis session is started from a portal which interacts with assorted grid services in order to determine which analysis servers to contact and how to divide the work into separate tasks for each analysis server. The user may actually be sitting in front of another desktop machine for a user interface but it is imagined that the portal maintains the state of the users analysis session. Also indicated in the diagram is that a number of users each with separate analysis sessions may be working with the same or other portals that are interacting with the same set of analysis servers, at the same time. This model does not necessarily have the view of single processors being used for single jobs for a period of time as in a batch model.

3 The Steps that a Physicist Takes in Performing an Analysis and the Requirements that These Steps Place Upon Grid Services

To understand the specific requirements that one or more of the above interactive analysis models places on grid services, we consider the set of steps that end-user physicists take in performing analysis. For each of these steps, we consider what grid services must be involved and what requirements are placed upon these services.

This section is largely based on discussions that took place during a PPDG CS-11 workshop held June 18, 19, 2002 in Berkeley.

In most cases, where the following document discusses the need for some information to be exchanged with the user, we actually mean both directly with the user (though a user interface such as a web service) and with the user's analysis application (through an API).

3.1 Select Data

Users start by specifying what data is to be analyzed.

1. Users may specify data by particular characteristics that they desire in events (a set of cuts).
2. Users may specify specific events or event collections.
3. Users may specify specific versions of specific events or event collections (such as the latest reconstruction).
4. Users may specify specific physical files (at least until they trust the lfn to pfn matching).
5. Users will prefer to use data that has good quality of service (e.g., available on disk rather than tape).
6. Users must specify not only which events but also which components of those events and which versions of those events. (It is possible to imagine analysis systems in which the determination of which components are needed may only be determined at run time, i.e., algorithmically, but such a situation is best handled by a two pass solution, breaking the work of determining needed event components into a separate job from the data analysis job). These components may be TAG, AOD, ESD, SIM, etc. or even user-defined subsets such as "MyAOD", "MyESD", "MySIM".
7. Users will need a way to make sure that the required calibration/geometry databases are available for the selected data.
8. Users need a way to be sure that the data they select at one moment (and associated calibration/geometry databases) will still be present when their analysis job is run.
9. Users may wish to make a persistent handle to this defined set of data (and associated calibration/ geometry databases) so that the same set can be used again later by them or some other user.

3.1.1 Requirements on Data Catalog Service

1. Data catalog services must be very reliable, easy to access, queryable (accessible to end user and to applications)
2. Data catalog service should have a hierarchical view (may have different overlapping views), like a file system (able to browse, set access controls, highly reliable, allow locks to protect from deletion).
3. Data catalog services must include quality of service information (how accessible in location and/or time is the data, what is its proximity to processing power)
4. Data catalog services must include information at the event component level, not just the event level.
5. Data catalog services must include attributes to identify various possible characteristics of "good" events.
6. Data catalog services must include a way to tell what calibration/geometry databases are associated with a given set of data.
7. Data catalog services must allow the user to create persistent handles to denote the data they have selected (and associated calibration/geometry databases). This selection may have been by event characteristics, specific events, specific versions of those events or specific files. Persistent handles should be available for all of those different data selection methods (sometimes the user wants to rerun a job but with all the latest data that meets the old job's selection criteria, other times the user wants to rerun with exactly the same files, and so forth).

3.1.2 Requirements on Job Control System or Interactive Analysis Interface

1. Job control system or interactive analysis interface must be able to accept data selection via persistent handle created during data selection process.
2. Job control system or interactive analysis interface must be able to scale to a global set of resources with priority, policy and quota limits. The complexity of job control, that arises from the constraints, are much more evident in the case of interactive analysis.

3.2 Get Smaller Local Subset of Data

Users often want to obtain a smaller local subset of data which they can inspect to design their analysis job. Inspection is done using a fast, "paw-like", tool.

1. Users may want this subset of data to be made by a merge of data from several sources rather than just a subset of data at one source.
2. Users may want to update or regenerate this subset periodically or on a trigger.

3.2.1 Requirements on Replica Management Service

1. Replica management services should allow user to move selected smaller subset of data to their local machine.
2. Replica management services may need to merge small amounts of data from several different locations to create the desired smaller subset of data.
3. Replica management services may need to be able to formulate data delivery strategy: timing and/or scheduling data delivery, optimization of choice of data sources.

3.3 Inspect Smaller Dataset to Develop Cuts and Analysis Programs

Users design cuts and analysis programs by inspecting smaller datasets using fast, "paw-like", tools. Such tools may be designed to look at local data or may use distributed designs to inspect data without first moving it to the local machine. These tools need not be capable of performing complete analysis jobs. Their purpose is to inspect data, not transform it.

3.3.1 Requirements on Matchmaking Service

1. If this inspection is to be performed on the grid rather than with an entirely local tool (because, for example, the dataset though small is not all in one place), the matchmaking service must be able to tell us where we can find the required combination of data availability and cpu availability to make this inspection task run quickly. Depending on the problem, the right answer might involve a large amount of cpu on one system or small amounts of cpu simultaneously on many systems.
2. Matchmaking may be so smart that it tells us where to run, or it may simply function as a catalog service of nodes cross-referenced by cpu and data access ability which can be used by some desktop component or portal that determines how to distribute the work.

3.3.2 Requirements not directly associated with grid services

1. If the inspection job does not work from entirely local data, may need a distributed paw-like tool (with ability to dynamically load scripts).
2. If the inspection job does not work from entirely local data, may need an API for creating, filling, manipulating analysis objects (e.g., histograms, tuples). AIDA is an example of such an API.
3. May need converters for multiple analysis tool data formats.

3.4 Move Data

Users may find that some data or calibration/geometry databases need to be moved before they can begin interactive analysis. Movement may be to a grid node or to the local system.

(For the simplest model of data analysis, "grid as black box", this is in fact the only remaining step before they become independent of the grid.)

3.4.1 Requirements on Replica Management Service

1. Replica management services should allow user to move selected data and associated calibration/geometry databases to a system where they intend to work (may be a grid node or their local system).
2. Replica management services should allow user to specify data to be moved by giving the persistent handle created during the previous step.
3. Replica management services may need to merge small amounts of data from several different locations to create a data set that the user wants on the local machine.

3.5 Choose Standard or Modified Versions of Code

Users may want to specify standard or modified versions of code (e.g., my better track finding)

3.5.1 Requirements on Job Control System or Interactive Analysis Interface

1. Job control system or interactive analysis interface must allow user to specify what code is to be used (in turn requiring ability to specify what the code requires to run, i.e., cpu, platforms, libs, OS versions, etc.)

3.5.2 Requirements on Data Provenance System

1. Data provenance system should tag output files with information about what resources were used (including code configuration). This is especially important if analysis jobs are being done to qualify alternate code for later production use or if results are to be considered valid for publication.

3.6 Run analysis on small data subset

Users generally run their analysis on a small subset of their selected data before they proceed to a larger job.

1. Users would prefer to use the same tools for analysis of this small data subset as they use for the larger complete data set (or perhaps the interface for small datasets is a subset of the one for large datasets).
2. This small job may be either local or distributed. Small here just means smaller than the entire selected dataset. The job may still be large enough to benefit from distributed processing.
3. The subset doesn't need to be a random sample. Users run a larger job on entire data set before publishing results.
4. For this first analysis, users generally prefer the fastest/most convenient data subset.
5. Once this subset is selected, users may want a persistent handle attached to it for later consistent reuse.
6. The small job will give a useful estimate of the resources and time-to-completion required for the larger job.

3.6.1 Requirements on data catalog service

1. Data catalog service should make it easy for a user (or an application) to find the fastest/most convenient data subset (while still optimizing other criteria such as resource usages, quotas, etc.).

2. Data catalog service should allow user (or an application) to attach a handle to this subset for consistent later reuse.

3.6.2 Requirements on Matchmaking Service

1. Matchmaking service must be able to tell where we can find the required combination of data availability and cpu availability to make this analysis task run quickly (while still optimizing other criteria such as resource usages, quotas, reliability, traffic, etc.). Depending on the problem, the right answer might involve a large amount of cpu on one system or small amounts of cpu simultaneously on many systems.
2. Matchmaking may be so smart that it tells us where to run, or it may simply return a list of nodes cross-referenced by cpu and data access ability which can be used by some desktop component or portal that determines how to distribute the work.

3.7 Retrieve results

Users retrieve and study results of the small analysis job.

1. Results may be kept remotely or locally.
2. Users would like ability to receive partial results while job is running.
3. There may be a need to collate results from distributed processing.
4. Results may include print, hists, tuples, files, etc.
5. Resource usage totals are also important results.

3.7.1 Requirements not directly associated with grid services

1. Retrieval of results requires the results collection part of an interactive data analysis API (such as AIDA).

3.8 Estimate resources for larger job

Before proceeding with analysis of their full selected data samples, users often need to estimate resource requirements.

1. Initial estimates can be made by extrapolation from resource usage in the small analysis job.
2. Estimates should then be updated based on feedback from large jobs in progress.
3. Estimates should include clock time as well as resource usage (with the understanding the clock time estimates are highly state-dependent).
4. Make initial estimate from extrapolation of resource usage by smaller job.
5. Update estimates based on progress.

3.8.1 Requirements not directly associated with grid services

1. Retrieval of results from small analysis jobs must include resource usage information.
2. Resource use information should be available from large analysis jobs in progress.

3.9 Negotiate availability or access to resources

Users need to know whether their projected resource requirements (storage, cpu, network, software, OS) can be met. They may negotiate tradeoffs between the amount charged against quota, and certain job characteristics: priority, location of execution, reliability, etc.

Users should be able to renegotiate resources for an already running job (so, for example, a user can prevent a job from being killed when they know it is five percent short of completion).

3.9.1 Requirements on authorization service

1. Authorization service should have an API to allow users or their applications to negotiate resource requirements (storage, cpu, network, software, OS).
2. Authorization service should allow processes to run based on permissions rather than always requiring reservations.
3. Authorization service should allow renegotiation for jobs already in progress.
4. Authorization service should provide cost determinations to aid user in negotiations.

3.10 Run analysis on large data set

Having tested their analysis on a small subset of data and having determined that sufficient resources might be available, users then begin the same analysis on the full selected data set. They may obtain information about the running job and may affect the run in various ways discussed in following sections (check status, view partial results, connect/disconnect, suspend/resume, abort)

1. Users would appreciate a seamless experience as they switch from running their analysis on a small data subset (perhaps on their local machine) to running on a large data set on the grid.
2. Users would also like a seamless switching between running connected (monitoring and controlling their large analysis job) or unconnected. The job should not disappear just because the desktop tool has been disconnected.
3. Users should be able to redirect, change priority, change data set size or suspend jobs in progress (all with good feedback as to how this affects "cost").
4. Users require a unique job-id for each job. This serves as identifier for log-files and possibly also to derive a unique seed for random numbers. The job-id must be generated and returned to the user at the beginning of the execution of the job.
5. Allow user to save state of their session so that they or someone else can run the same job (or take this job as a starting point and modify the parameters from there)
6. Users do not want jobs to abort simply due to token expiring or arbitrary session timeouts.
7. Users would like the analysis system to rapidly recover (preferably without user noticing) from system failures or network failures.
8. Users would like the analysis system to automatically shift subparts of the job from failed or slow systems to alternate systems.
9. Users would like the analysis system to survive crash on certain individual events. Do not let entire large job fail just because user code crashes on a few particular events.
10. Users would like jobs to have checkpoints so that work need not be entirely repeated following system/network failures or resource limits.

3.10.1 Requirements on Matchmaking Service

1. Matchmaking service must be able to tell where we can find the required combination of data availability and cpu availability to make this analysis task run quickly. Depending on the problem, the right answer might involve a large amount of cpu on one system or small amounts of cpu simultaneously on many systems.
2. Matchmaking may be so smart that it tells us where to run, or it may simply return a list of nodes cross-referenced by cpu and data access ability which can be used by some desktop component or portal that determines how to distribute the work.

3.10.2 Requirements on Job Control System or Interactive Analysis Interface

1. Job control system or interactive analysis interface must allow user to specify what code is to be used (in turn requiring ability to specify what the code requires to run, i.e., cpu, platforms, libs, OS versions, etc.).
2. Job control system or interactive analysis interface must provide a unique job-id for each job.
3. Job control system or interactive analysis interface should recover from system failures or network failures.

4. Job control system or interactive analysis interface should be able to decompose the work of an interactive analysis session into subparts to take advantage of available processing resources or to match distribution of the dataset.
5. Job control system or interactive analysis interface should automatically shift subparts of the job from failed or slow systems to alternate systems.
6. Job control system or interactive analysis interface should not let entire large job fail just because user code crashes on a few particular events.
7. Job control system or interactive analysis interface should have checkpoints so that work need not be entirely repeated following unsurvivable system/network failures or resource limits.

3.10.3 Requirements on Portal

1. Portal must maintain state of user analysis jobs such that user can disconnect and reconnect without interrupting running jobs.

3.10.4 Requirements not directly associated with grid services

1. Need an API for creating, filling, manipulating analysis objects (e.g., histograms, tuples). AIDA is an example of such an API.
2. Must be scalable to large number of nodes (1000's?). If interactive access to 1000's, then need asynchronous network protocols to allow parallelization of network traffic.
3. No part of the system should cause the job to fail simply because a token has expired or a session has timed out.

3.11 Check status of analysis in progress

Users must be able to check the status of analysis jobs in progress.

3.11.1 Requirements on Job Control System or Interactive Analysis Interface

1. Job control system or interactive analysis interface must allow user to check status of an analysis job in progress. This could be as simple as a "list all jobs" command but would be better handled as a progress API or web service.

3.12 View results of analysis in progress

Users must be able to view results of analysis jobs in progress.

3.12.1 Requirements on Job Control System or Interactive Analysis Interface

1. Job control system or interactive analysis interface must allow user to view results of an analysis job in progress. This requires an API or web service that allows you to attach to a job, get a list of analysis objects and fetch them (may require distributed fetch and merge). AIDA could be part of the solution.
2. Job control system or interactive analysis interface must be designed to be highly responsive to requests for partial results.

3.13 Suspend/resume analysis in progress

Users would like to be able to suspend/resume analysis jobs in progress either to free up resources for other jobs or to avoid running when they are not present to monitor progress. It is understood that it may be too difficult to do this reliably (implies a checkpointable system).

3.13.1 Requirements not directly associated with grid services

1. It is acknowledged that it may be very difficult to make all parts of the analysis chain able to suspend/resume. Such a mechanism places many requirements on the analysis chain for persistency.

3.14 Abort analysis in progress

Users must be able to abort analysis jobs in progress. Response from abort command should be prompt to avoid wasting resources.

1. Users would like the option to get partial results back after an abort.
2. Users would like the option to have output of aborted jobs cleaned up or not cleaned up.

3.14.1 Requirements on Job Control System or Interactive Analysis Interface

1. Job control system or interactive analysis interface must respond promptly to user request to abort a job (with options to return or not return partial results and options to clean up or not clean up other output). The interface must ensure that post-abort cleanup is performed (if requested) and that the system is left in a consistent state.

3.15 View results of completed analysis

Users must be able to view results such as histograms, fits, print out, tuples, files, etc. (which may require collating results from distributed processing).

1. Users would prefer to have the same API to access this information whether it is from a job that is in progress or a job that is complete.
2. Users would appreciate partial results even if the analysis job fails (due to resource limits, system failures, code crashes, etc.). But the user must be informed that such results are incomplete and user should be told what part of the data set was or was not processed.

3.15.1 Requirements on Job Control System or Interactive Analysis Interface

1. Job control system or interactive analysis interface must allow user to view results of a completed analysis job. This requires an API or web service that allows you to get a list of analysis objects and fetch them (may require distributed fetch and merge). Ideally this would be the same API as is used for viewing results from analysis jobs in progress. AIDA could be part of the solution.
2. Job control system or interactive analysis interface must return partial results even if the analysis job fails (due to resource limits, system failures, code crashes, etc.). But the user must be informed that such results are incomplete and user should be told what part of the data set was or was not processed.

3.16 Displays of selected individual events at varying levels of detail

Users may want to produce event displays of selected individual events at varying levels of detail.

1. Users may want to ask for event displays while the job is running, or they may want to come back after the job has run, asking for specific event displays based on what they see in histograms or other output.
2. Users may want to be able to stop an analysis job at a particular event and then turn on event display.

3.16.1 Requirements on Job Control System or Interactive Analysis Interface

1. Job control system or interactive analysis interface should provide a way to ask for event displays of selected individual events at varying levels of detail either for jobs that are running or for specific events after the job has run. The HepRep event display interface may be useful.
2. Job control system or interactive analysis interface should provide a way to stop an analysis job at a particular event and then turn on event display.

3.17 Add refined data to data store

User analysis jobs should be able to add refined data (new AOD) and results to the group/collaboration data store and update the data catalog service to reflect presence of this new data.

3.17.1 Requirements on Data Catalog service

1. Data catalog services must provide an API/UI so that user analysis jobs can declare what refined data (new AOD) and results they have added to the group/collaboration data store.
2. Data catalog services must be very reliable, easy to access, queryable (accessible to end user or to applications)

3.18 Share refined data with collaborators

Users often want to share refined data with collaborators (individuals, physics groups, etc.).

1. Users require tools to set access permissions on refined data that they have added to the data store.
2. Users require tools to announce changes to data catalog services.

3.18.1 Requirements on Data Catalog service

1. Data catalog services should allow users to set access permissions.
2. Data catalog services may provide tools to announce changes made by users.

3.19 Add tag data

Users may want to add tag data based on their results.

3.19.1 Requirements on Tag Data Collections

1. Tag data collections should be writeable by user analysis jobs.

3.20 Compare results

Users often want to compare results of various jobs (real to simulated, etc.). Useful comparison tools provide the ability to overlay, add, subtract and divide histograms, the ability to access histograms from multiple jobs and the ability to mark and save certain histograms for use as standard references.

1. Users may want to make such comparisons as part of an automatic comparison/quality control system.
2. Users require a solid audit trail (data provenance information reflecting choices of data, code, parameters, etc.) to make such comparisons useful.

3.20.1 Requirements not directly associated with grid services

1. Desktop analysis tools need the ability to overlay, add, subtract and divide histograms, the ability to access histograms from multiple jobs and the ability to mark and save certain histograms for use as standard references.
2. Desktop analysis tools provide additional value if their comparison abilities can work as part of an automatic comparison/quality control system.

3.21 Calculate cross sections

The last step of obtaining physics results is the calculation of cross sections. Such calculations rely on a thorough understanding of whether the analysis job ran on the complete selected data set.

1. Users may be able to proceed with incomplete results, but they must know exactly what parts of the specified data set failed to run.
2. Users should have the ability to rerun those parts of the data that failed to run and have these new results merged into the previous results.

3.21.1 Requirements on Job Control System or Interactive Analysis Interface

1. Job control system or interactive analysis interface must return partial results even if the analysis job fails (due to resource limits, system failures, code crashes, etc.). But the user must be informed that such results are incomplete and user should be told what part of the data set was or was not processed.
2. Job control system or interactive analysis interface should allow the user to resubmit the job to rerun those parts of the data that failed to run and have these new results merged into the previous results.

3.22 Maintain audit trail (data provenance)

At all steps of the analysis, it is essential to maintain an audit trail (data provenance).

3.22.1 Requirements on Job Control System or Interactive Analysis Interface

1. Job control system or interactive analysis interface must provide a way to keep histories of past jobs, their inputs and outputs and what transformations were applied.

3.23 Security and access control

At all steps of the analysis, it is essential to maintain security and access control.

3.23.1 Requirements on Authentication Services

1. Authentication services must provide user and group level authorization for read/write/update access to everything. All other services must be designed in such a way that authentication is respected.

4 Conclusions and Outstanding Issues

While the working physicist will find few surprises in the above requirements, it is clear that interactive analysis places significant requirements on grid services.

Grid services most strongly impacted by interactive analysis requirements are the Data Catalog Service and the Job Control System (or Interactive Analysis Interface), but many other services are also involved. The heavy requirements placed on the Data Catalog Service strongly suggest the use of a Database Management System behind this service.

4.1 Outstanding Issues

Some of the outstanding issues are listed below.

4.1.1 Need for standard data definitions

At least among the group assembled for the CS11 workshop at Berkeley in February, there was significant confusion over data definition terminology (such as "dataset"). Agreement on clearer definitions must be made for documents such as this to be effective. It may be best to take these definitions from the HEPICAL document (<http://fca.home.cern.ch/fca/HEPCAL.doc>).

4.1.2 Need for “grid object” definition

Interactive analysis users are generally concerned with data at a finer grain than just whole events. They need to know which components of a given event are available for their analysis (TAG, AOD, ESD, SIM, etc.). To make progress in this area, it would be useful to develop a shared “grid object” definition.

4.1.3 Need for debugging tools appropriate for the grid environment

New code is often used during interactive analysis. If this analysis is to proceed in a grid environment, distributed debugging tools must be available to help the user. It is not sufficient for the user to debug their code just on their local machine. They must have tools to help with more difficult questions such as why their code that ran fine on their local machine then fails on some other grid node.

4.1.4 Need for common metadata catalog schema

If different experiments could agree on a common metadata catalog schema, it would make it much easier to design analysis tools that can be shared across experiments.

4.1.5 Matchmaking services

Will matchmaking services be so smart that they can tell the analysis tool where to run, or will they simply return a list of nodes (cross-referenced by cpu and data access ability) that serves as input to decision making by the desktop tool or portal?

4.1.6 User interface portal

Understand requirements for user view of grid services and interactions relevant to interactive data analysis. Explore applicable technology and prototype grid-enabled interactive analysis environments.