

# Datasets for the Grid

Version 5

David Adams

September 29, 2003

## ***Introduction***

The grid provides a framework for managing and processing very large collections of data. Files are a very important unit for data handling but are not convenient for expressing a collective data view because large data collections must span a large number of files. The large data volume also makes it desirable to express some data collections as collections or subsets of existing collection rather than replicating all the data into new files.

Here we use the term dataset in a very generic manner to describe a set of data that is intended to be treated collectively. Of course, if we are to process the data in a dataset, the processing system will eventually need to determine the location of the data. Some or all of the data is likely to be replicated at multiple sites. Here we categorize datasets according to the existence and extent of this location information and identify the properties associated with datasets and their relevance to each of these categories. We also identify some common operations to be performed on datasets, list a couple useful concepts and begin discussion of one possible implementation.

We identify two main use cases: selecting a dataset and processing a dataset. On the grid, both the data and the processing of this data may be distributed over multiple nodes and sites. The support of distributed storage and processing is an important requirement for datasets.

The dataset model is appropriate for almost any type of large data sample. A case of particular interest is the management of record-oriented data such as event data in high energy physics (HEP). These are collections of a large number of objects of the same type for which the same processing algorithm is typically applied to each record (event). In the following, we often use the word event as a placeholder for any kind of record.

## ***Properties***

We identify the following properties of datasets:

0. Identity
1. Content (actually description thereof)
2. Location (of the data)
3. Mapping (content to location)
4. Provenance
5. History
6. Labels (aka metadata or physics metadata)
7. Mutability
8. Compositeness

Note that the data itself is not included in this list. All of the above are metadata, i.e. data that describe the data referenced by the dataset. Details are given in the following sections.

## **Identity**

Each dataset is assigned an identity, e.g. a name or index, so that it can be cataloged and distinguished from other datasets. Naturally this identity should be unique in any context in which the dataset will appear.

## **Content**

Content is a description of the type of data that is contained in the dataset. If the dataset is event-oriented, then the content includes the number of events, a list of event identifiers (where applicable) and the event content, i.e. the type of data in each event. For HEP data, the event ID might be the beam crossing ID assigned during initial processing and the events might, for example, contain raw data, tracks and jets.

## **Location**

Of course, there must be means to locate the data associated with the dataset. This location may be a list of physical files, logical files or a collection of persistent identifiers associated with a database or object management system such as LCG POOL. The location might include a list of processing nodes (aka CPU's) from which the data can be accessed.

## **Mapping**

The dataset must provide a mapping between its content description and the associated data. This does not necessarily imply that random access is available. It is sufficient that there be means to serially process the data as long as we know which content is being processed at any time.

## **Provenance**

Provenance is the prescription by which the dataset was (or will be) constructed. In the typical case where a dataset is created by applying a transformation on an input dataset, the provenance is the identity of the input dataset and a description (or identity) of the transformation. Another case is the merging of datasets to form a new dataset. In this case the provenance is the list of input datasets. The full provenance of a dataset can be constructed by combining its immediate provenance with that of all of its ancestors.

## **History**

The history of a dataset adds details of actual production to the provenance. These include how the processing was decomposed into jobs and which processors were used and the time of production for each job. Ideally, the data in a dataset will not depend on these details but in practice different hardware or software may result in different data.

## **Labels**

Labels refer to attributes assigned to a dataset as a whole and not covered by the other items. These might include a flag indicating whether the dataset is suitable for use in a

particular analysis or for publication. For a HEP event dataset these might also include the integrated luminosity.

### **Mutability**

Mutability tells whether a dataset may be modified without changing its identity. We identify three useful states: locked meaning no changes are allowed, unlocked meaning data may be added or removed and extensible meaning that data may be added but not removed.

### **Compositeness**

A dataset may be composite, i.e. composed of other datasets. This compositeness may arise naturally from construction (the summer dataset is made up from the June, July and August datasets) or may reflect placement of the data (a global dataset includes a datasets at the New York, Paris and Moscow computing centers). In the former case, the compositeness defines the provenance.

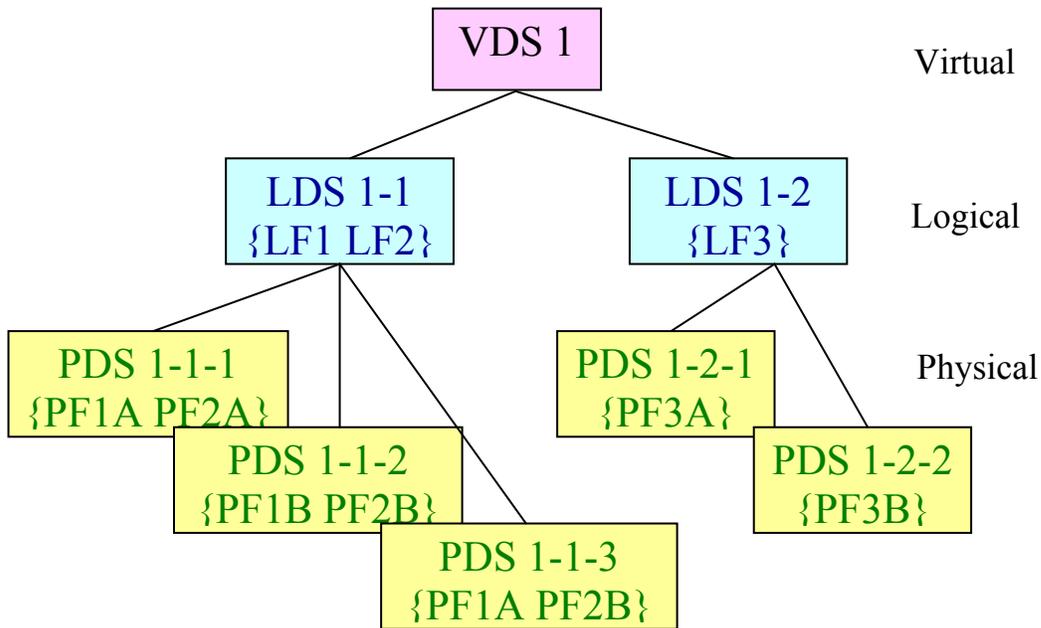
### ***Dataset categories***

Distributed file management systems such as RLS and Magda are constructed in terms of logical files and their associated physical replicas. A user may select a logical file from a file metadata catalog and then consult a replica catalog to locate a nearby physical instance of that file. In the virtual data model, a logical file may be virtual, i.e. a replica is created from the provenance when needed. This can occur because the file was never created, all replicas were deleted or no replica is conveniently accessible.

It is useful to categorize datasets in a similar way. For simplicity we restrict the discussion to datasets with location expressed in terms of files. Extension to other media such as databases should not be too difficult.

We identify four categories: virtual, logical, physical and staged. A virtual dataset is one without location information. In principle, the provenance may be used to generate or regenerate the data. A logical dataset is one whose content is expressed in terms of logical files. A file replica catalog can be used to locate the physical file instances associated with any logical file. A physical dataset is one for which physical instances have been identified for all logical files. A staged dataset adds knowledge of compute nodes.

There are one-to-many mappings between datasets in these categories as we move down the hierarchy from virtual to staged. A virtual dataset may have multiple logical representations corresponding to different collections of logical files. Each combination of physical replicas derived from this set of logical files may be used to construct a different physical representation of each logical dataset. Finally, different CPU mappings or different sub-dataset boundaries provide multiple staged representations of each physical dataset. These relationships are exemplified in figure 1.



**Figure 1. Example of a virtual dataset with two logical representations and five physical representations.**

We assume that each manifested logical, physical or staged representation of a virtual dataset is assigned a (unique) identity. For provenance tracking, these representations are equivalent and it is useful to assign a distinct identity to the virtual dataset. Thus any dataset representation, i.e. a dataset in any category beyond virtual, carries both a representation ID and a (virtual) dataset ID. It might be useful to extend this to lower categories, i.e. distinguish logical, physical and staged dataset ID's.

Next we consider the mapping of the remaining dataset properties to each of the dataset categories. Identity has already been discussed.

### **Virtual dataset**

The content, provenance, labels and mutability (items 1, 4 and 6, 7) are clearly associated with virtual datasets. Note that part of the content may not be known before the data is generated, e.g. the events included after an event selection based on the data. Some labels may also be undetermined because they are evaluated from data. Compositeness (item 8) with decomposition into virtual datasets (reflecting provenance) is also a property of virtual datasets.

### **Logical dataset**

Assuming production was carried out using logical files, the location, mapping and history (items 2, 3 and 5) are characteristics of a logical dataset. Compositeness (item 8) with decomposition into logical datasets (typically used to reflect placement) is also a property of logical datasets.

## **Physical dataset**

The physical dataset extends the location information (item 2) to specify a physical instance of each logical file. There is a trivial extension of the mapping as well. Their might also be a more generic location, e.g. the dataset can be found at a particular site.

## **Staged dataset**

The staged dataset further extends the location to include specification of the compute nodes where the data is to be processed. The staged dataset could be composite with each sub-dataset corresponding to a single job and holding the identity of a compute element and the list of files to be processed on that element.

## **Use cases**

Our two use cases map nicely onto the above categories. A typical user selecting a dataset will base that selection on the content, provenance, labels and mutability, i.e. the virtual dataset properties. This virtual dataset is handed off to a processing system that splits the dataset into sub-datasets (e.g. corresponding to single files) and uses a compute node to process each constituent. The splitting might be accomplished by selecting a corresponding logical dataset from a logical dataset replica catalog, splitting it along file boundaries and then, for each sub-dataset, using a file replica catalog to locate a physical instance of each logical file. The physical and staged datasets implied by splitting and matchmaking might be made it explicit and recorded for use the next time the dataset is processed.

## **Operations**

Next we discuss the operations that are performed on datasets. These include the user-initiated operations of production, creation, selection and merging as well as the splitting and matchmaking that arise in a distributed processing. The important special case of an event-oriented collection is emphasized.

## **Production**

The normal means to create a dataset is to apply a transformation to an existing dataset to produce a new dataset. The provenance is the input dataset and the transformation (or their identities). In a typical HEP production, content is added to each event but the list of events (more precisely event ID's) does not change.

## **Creation**

There must be a starting point for the chain of datasets which appear in production sequence, i.e. there must be means to create a dataset without reference to any other datasets. The provenance for these created datasets is special and depends on the type of dataset. In the case of HEP, data collected from a detector over a given time period or in a single file might be used to define such a dataset. For HEP Monte Carlo, the created dataset might be a list of event identifiers with associated random number seeds.

## **Selection**

Another important transformation defines a new dataset by selecting a subset of the data from an input dataset. Typically, the result of the selection can be expressed as the

selected content, e.g. for an event dataset, the list of identifiers of selected events. An implicit logical representation of this dataset can be formed by replacing the content in any logical representation of the original dataset with the selected content. Or an explicit logical representation may be formed by copying the selected data into a new set of logical files.

An important example of selection occurs during processing. If the processing system recognizes that some of the data in a dataset is irrelevant (e.g. a tracking algorithm might not look at calorimeter data), then it can select only the relevant content in a dataset. The goal is to process a new dataset whose location (set of files) is a subset of that in the original dataset and thus reduce the amount of data that is transferred and accessed.

### **Splitting**

A typical means to distribute processing is to split a dataset into sub-datasets and simultaneously process each sub-dataset on different machines and possibly at different sites. If the dataset is composite, then it is natural to split along sub-dataset boundaries and, indeed, these boundaries were likely defined for that purpose. The splitting may be done more than once, i.e. the sub-datasets from the original split may be split further.

For event-oriented datasets, it is most natural to split along event boundaries, i.e. have each machine process the data associated with a distinct collection of events. There must be means to split datasets and to separately know the location for the data in each sub-dataset. This splitting should take advantage of the natural boundaries of the location, e.g. split along file boundaries.

The list of sub-datasets produced by splitting can be used to define a (possibly) new composite dataset. In this view, splitting is an operation that takes a dataset as input produces a new dataset or locates an existing one. The output is composite and most likely logical or physical.

### **Matchmaking**

Processing is carried out by a collection of jobs each of which processes a subset of the data on a particular compute node. The procedure of associating these data subsets with compute elements is known as matchmaking. Staged datasets are not required for processing but provide a means of recording this assignment. If they are used, then matchmaking is the operation which produces a staged dataset. It is not always desirable to make a clean separation between splitting and matchmaking.

### **Merging**

It is common to merge one or more datasets to form a new dataset. This occurs both in the definition of a new dataset (e.g., combining data from two run periods) and when combining the data output from the jobs in distributed production of a dataset.

For virtual datasets, these cases are likely treated differently. In the former case, it is natural to define the new dataset as a composite. In the latter case, the compositeness is an artifact of production and is not included. The merged content is evaluated and recorded.

For logical datasets, we have the option of recording the compositeness, merging the relevant properties (location, mapping and history), or merging properties and the data itself, i.e. copying it to new logical files. Or we can carry out more than one of these operations and record multiple logical representations of the dataset.

### Event-oriented data

For event-oriented data, it is natural to split the content into the event ID's and the content of each event. We say that an event dataset is consistent if it has the same content for each event. It is normally of interest to split along event boundaries and to make selections and merges that result in consistent datasets. Figure 2 illustrates the fundamental operations performed on an event dataset.

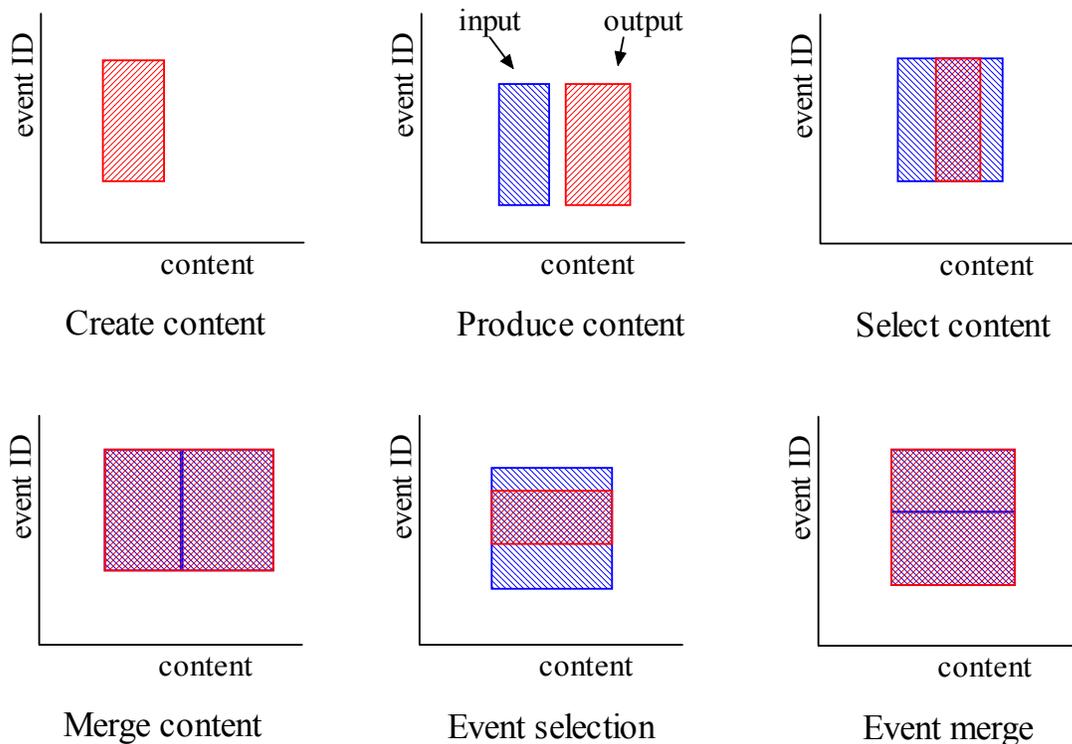


Figure 2. Fundamental event dataset operations. In each figure, the input dataset or datasets are shown in blue and the output in red. Content refers to the content of each event.

### Concepts

There are a couple other concepts that are useful in the discussion of datasets. These are equivalence and completeness.

#### Equivalence

Two datasets are said to be equivalent if they point to the same data with the same content and mapping. Different representations of the same virtual dataset are trivially equivalent. More interesting is the case of two virtual datasets with different provenance.

## Completeness

The provenance of a dataset can be used to discover all the (dataset) data from which the data in the dataset was derived. A dataset is said to be complete if all such ancestor data is included in the dataset. The number of datasets that appear in provenance tracking may be greatly reduced by only tracking complete datasets.

## Implementation

The dataset properties identified here vary widely in their size, nature and intended use. The implementation of a system to provide access to all these properties is likely to include multiple components, each intended for accessing particular properties. Here we describe one possible implementation. There are others that are equally plausible.

We assume a typical user wishes to locate a virtual dataset (VDS) and let the processing system to choose the representation to be processed. The user may consult a dataset selection catalog (DSC) to choose the VDS of interest. The processing system consults a dataset replica catalog (DRC) to find an appropriate logical dataset (LDS) for processing.

The VDS and LDS are described by objects in the object-oriented sense of the word, i.e. there are corresponding classes in the relevant language or languages (C++, Python, Java, etc.). These objects have persistent representations so they can be exchanged between processes, across networks and over periods of time when no process is running. The persistent and transient representations should be portable, meaning that much or all of the data associated with a VDS or LDS is carried directly by the object and does not require access to a global dataset repository.

We do anticipate the existence of repositories holding descriptions of VDS or LDS objects. Given a dataset ID, these repositories will return the persistent description of the corresponding dataset. These repositories are not intended to facilitate the selection of datasets based on dataset properties. This capability is provided by the DSC and DRC.

Table 1 shows the assignment of event dataset properties to the components described here. In some cases the properties are decomposed into sub-properties. We expect history to be recorded in a distinct production catalog which is not discussed here.

Property		VDS	LDS	DSC	DRC
identity	virtual	X		X	X
	logical		X		X
content	# events	X	X	X	
	event ID's	X	X		
	event content	X	X	X	
location	logical files		X		
	site (with data)				X
mapping			X		
provenance	parent			X	
	xform			X	
history					
labels				X	
mutability		X	X	X	
compositeness	virtual	X		X	
	logical		X		X

**Table 1. Assignment of properties to components.**

## ***Other dataset definitions***

The word dataset is widely used. We suspect that almost everyone agrees that the “dataset” is appropriate unit for data selection as long as we are not too precise in the definition of the term. We have made a very general definition and have identified categories and properties of datasets. Here we try to connect with definitions of datasets and data collections in other work. Corrections to our understanding and suggestions for other definitions are welcome.

### **HEPCAL**

The HEPCAL group has identified common use cases for LHC applications on the grid. Their definition of dataset is close to that of the logical dataset introduced here. They also state that all datasets are immutable (locked).

### **POOL**

The LCG POOL group is developing a hybrid event store for use at the LHC. They define collections which are persistent pointers to objects of the same type. The case of most interest is a collection of events or event headers. The persistent pointers include a specification of the logical file where the referenced object can be found. In a typical application, the event headers describe the data in the event and provide pointers for each referenced object. Thus the collection of event headers provides the content, logical file locations and mapping characteristic of a logical dataset.

If a referenced object is copied into a new file, the copy is assigned a new pointer. However POOL has made provision for users to provide a mapping from the old pointer to the new one. If this is implemented, then the same collection could be used with different sets of logical files. In this case, the collection is closer to a virtual dataset.

## ***Changes from earlier versions of this paper***

Version 2 of this note adds a third category, virtual, to the two dataset categories, logical and physical. It also changes the meaning of these latter categories.

Version 3 extends location to include processing and adds the dataset category staged.

Version 4 adds the properties mutability and hierarchy and provides comparison with other dataset definitions, specifically HEPCAL and POOL. The list of operations is also extended.

Version 5 identifies implementation components and adds the corresponding table. It also adds the VDS/LDS/PDS figure. It also replaces the dataset property name hierarchy with the name compositeness.