

Dataset management

Version 0.02

David Adams

January 18, 2005

Introduction

The ATLAS and other LHC experiments will annually produce millions of data files requiring petabytes of storage. Most of the volume will be occupied by raw and first-stage reconstruction files that are not typically of interest to the average physicist, but a significant fraction of the volume and a large fraction of the files will be devoted to data that are used as input to end user analysis. Means for managing and accessing these files individually can be provided by a file management system (FMS) as discussed in an earlier note [1]. Here we address the issue of collective data management, i.e. means for users to manage, access and track the provenance of collections of files without carrying around long lists of file references and using such lists in direct interaction with the FMS. Instead, interaction is with a dataset management system (DSMS) that can catalog collective properties and avoid frequently repeating a query for each entry in a list of file references.

Our discussion is carried out in the context of the ATLAS experiment to provide concrete examples but applies equally well to other LHC experiments, other HEP experiments and other systems wishing to provide a coherent view of a large number of files. This is especially true in those cases where the data is event-oriented, i.e. made up of a large number of data blocks, here called events, where each is processed in a similar manner to construct a data collection.

For simplicity, the discussion assumes data residing in files, but the ideas apply equally well to other types of data e.g. to that residing in relational database tables or an object management system.

Datasets

We use the word *dataset* to describe a collection of data typically but not necessarily held in a collection of files. The properties of datasets and their useful operations are described in another note [2]. The relevant properties here are identity, location, content, composition, provenance and lifetime. Identity means that each dataset is labeled with a unique identifier. Location refers to the list of files (or other entities) holding the data in the dataset. Content is summarized by a label describing the type of data, e.g. ESD, AOD-electron or AOD-muon. Composition implies that datasets may be constructed from other datasets and specifically refers to the list of constituent datasets that comprise a composite dataset. We assume a simple transformation model where a new dataset is produced by transformation of an existing dataset: the provenance of the new dataset is specified by the identities of the transformation and the original dataset. There is also the possibility to form a dataset by simple merging of existing datasets; in this case the provenance is specified by the identities of the merged datasets. In either case, the datasets referenced in the provenance are called parents. Lifetime implies there are conditions where a dataset can cease to exist and resources such as the space used to hold its files or entries in database tables can be recovered.

There are many facets to the existence of a dataset. A summary of its intrinsic properties is carried in the dataset description which is normally stored in a dataset repository. These include the identity, location, content, composition and the parent dataset. These and other properties may be recorded in catalogs indexed by the dataset identifier.

A concrete dataset is one which references data, here in stored in physical or logical files. The files are specified by references such as those described in the FMS note. We also allow for the existence of virtual datasets that do not have such data references. These may be mapped to concrete datasets with a dataset replica catalog (DRC). A virtual dataset may also be used to record the prescription for creating a concrete dataset.

Dataset management

A dataset may be formed by any selection of data from any combination of files and hence the number of datasets is likely to be large, possibly much larger than the number of files. The role of the dataset management system (DSMS) is to provide managers and ordinary users with a comprehensible view of the collection of datasets. Obviously, users must be able to create datasets and record their intrinsic and other properties. The DSMS must also enable users to select datasets of interest, to access the data associated with a dataset and track the provenance of datasets. Users must also be able to control the lifetime of datasets, i.e. delete them when they are longer of interest. Finally, if (as we assume) the system is to support virtual datasets, then the DSMS is responsible for cataloging them and their associations with concrete datasets.

Dataset selection

A user wishing to examine or process a dataset must provide its ID or name to access its properties. This name or ID may be typically selected from a dataset selection catalog (DSC), sometimes called a metadata catalog. These catalogs associate dataset properties with a dataset name and ID and users can make queries on these properties to select a dataset of interest.

Data access

As previously indicated, for the datasets considered here, the associated data are found in files. The DSMS has the responsibility of determining the list of files associated with a dataset and the FMS has the ultimate responsibility of enabling users to access those files. However, in that the files for a dataset are often produced at a single site, accessed at a single site and collectively moved between sites, it is natural for the DSMS to track the sites where datasets (more precisely their data) are located and make this information available to data managers, ordinary users and workload management systems.

Provenance

We require the DSMS record the provenance of any dataset, i.e. the identities of the transformation and parent dataset or the identities of the merged datasets. The DSMS provides access to the provenance chain, i.e. to all ancestor datasets and associated transformations. The DSMS should ensure the integrity of this chain—the removal of an intermediate dataset from the DSMS should not remove all traces but leave information sufficient to understand the origin of datasets appearing later in the chain.

Lifetime

Rather than simply allowing all datasets to persist indefinitely in the DSMS, we require some mechanism by which they can be removed or at least marked as such. This is important to recover resources such as storage space for data files and to reduce clutter in catalog tables. Analysis activities naturally produce a large number of datasets that are naturally short-lived (e.g. partial results) and it is very convenient to use the same DSMS to manage these as is used for the long-lived datasets characteristic of production activities. This problem is complicated because multiple users may express interest in a dataset and because associations between datasets may propagate this interest from one dataset to many others.

Virtual data

We assume a virtual data model based on virtual datasets rather than virtual files as in Chimera [3]. If a transformation is run twice with the same input, the output files and datasets are assigned different identities in both cases and the produced concrete datasets may be mapped to the same virtual dataset. Or the data contained in the files for a dataset may be copied into a new set of files used as the basis for an equivalent dataset and these two datasets similarly mapped to a virtual dataset. These virtual datasets have all the properties of concrete datasets except, by definition, they have no location information. The DSMS is required to record these properties and the concrete-to-virtual mappings.

Lifetime management

We assume a claim-based lifetime management strategy for datasets similar to that used for files in the FMS [2]. A dataset remains active as long as there is at least one active claim on that dataset. When no claims remain, a dataset is inactive and may be deleted. As in the file case, these claims may be external (also called explicit) and held by a user or data manager, or, unlike the file case, these claims may be internal (aka implicit), implied by association with another claimed dataset. These associations may arise from composition, provenance or concrete-virtual mapping with their effect depending on this nature.

External claims may be released at any time and have an expiration time at which they are automatically released. The claimer may reset this value at any time as long as the corresponding dataset remains active. Internal claims have the same lifetime as the external claim that invokes them.

Claim attributes

A dataset claim carries attributes specifying the implied claims made on associated datasets. We identify three important attributes: constituent claiming, ancestor claiming and replica claiming. Others are possible.

A dataset claim with constituent claiming implicitly claims the constituents of that dataset. The attribute has a value that indicates whether it extends to implicit claims, i.e. whether claimed datasets also claim their constituents. This might be restricted by a number of levels. We say a claim has complete constituent claiming if it extends to all implied claims.

A dataset with ancestor claiming implicitly claims the datasets in its provenance. Again this may be limited to the immediate parent (or parents) or may extend back multiple generations. A complete ancestor claim extends along the full provenance chain leading up to the claimed dataset.

A claim on a virtual dataset may include replica claiming that extends the claim to one or more concrete replicas. The motivation for this attribute is to ensure that one or more replicas exist for the lifetime of the virtual dataset. The attribute could indicate how many replicas should exist and perhaps a policy for which ones are preferred. It is possible that the implied claims are conditional; a replica dataset is claimed only if there are no other replicas for the claimed virtual dataset.

ATLAS claims

Here we discuss how the above claim attributes might apply to ATLAS. We assume the lowest level datasets will hold file references and claims to those files and so full constituent claiming is required to ensure that the data for a dataset remains accessible. Ancestor claiming might be useful in some cases, e.g. to ensure that the parent ESD remains available for an AOD dataset, but there may be cases where we would allow the corresponding ESD to be deleted. It is not clear whether replica claiming is worth the trouble of implementing a conditional claims mechanism. This depends on whether and how virtual datasets are used in ATLAS.

Use cases

We identify some use cases for the DSMS:

1. A processing system acting on behalf of a user performs a transformation, records the new dataset and its provenance in the DSMS and claims the dataset for that user.
2. The user extends the lifetime of that claim
3. The user inserts the dataset into a DSC and assigns attributes.
4. Another user selects the dataset using implicit and assigned attributes.
5. A processing system given this dataset as input finds sites where the dataset is present, i.e. its files are available.
6. A processing system or data manager asks to move a dataset to a site.
7. The input in the previous two cases is a virtual dataset.

Interface

The detailed interface is left to a later version of this document. At the VO level, we expect something like the dataset repository, selection catalog (DSC) and replica catalog (DRC). The interface must allow users to determine the sites where a dataset is available (i.e. its files are present) and must provide means to make a dataset available at a site. It is not clear whether this should be accomplished by interacting with a VO- or site-based service.

Implementation

The interface should be specified before going to much further with implementation. DIAL [3] provides classes describing the dataset repository, DSC and DRC and rudimentary implementations based on MySQL tables. Lifetime management and data placement have not yet been addressed.

The DSMS is distinct from the FMS to break the problem into more manageable pieces. Current grid middleware development projects address file management and so are most likely to be receptive to the FMS ideas and to deliver mature services with a similar level of functionality. ATLAS and other VO's will likely have to provide much of the implementation of the DSMS at least in the short term.

Conclusions

Concepts and use cases for collective data management have been provided along with rudimentary discussion of interface and implementation. The unit of management is the dataset introduced in an earlier note. The dataset management system (DSMS) described here makes use of the file management system (FMS) described elsewhere. The DSMS makes use of a claims mechanism for lifetime management similar to that used in the FMS.

References

1. "File management on the grid", D. Adams, <http://www.usatlas.bnl.gov/ADA/docs/fms.pdf>.
2. "Datasets for the Grid", D. Adams, <http://www.usatlas.bnl.gov/~dladams/dataset/docs/griddataset.pdf>.
3. The DIAL home page is <http://www.usatlas.bnl.gov/~dladams/dial>.