The Sloan Digital Sky Survey Data Archive Server

Eric H. Neilsen, Jr.

Chris Stoughton

The SDSS Data Pipeline

Spectrographs collect exposures on plates

2.5m Spectrographs

2.5m Imaging Camera

Photometric Telescope

Imaging camera collects data in runs along stripes. Each CCD has a filter and a camera column. The data acquisition system:
splits data from a CCD into frames,
and stores each frame in a raw fits file.

MT Pipelines

The spectro pipeline:
* combines exposures from one night,
* extracts spectra from each fiber,
* makes science measurements on each specrum,
* stores output in files according to observation date, plate, and fiber.

Spectro Pipelines

Imaging Pipelines

The imaging pipeline:
* corrects frames for distortions,
* measures frame parameters (eg PSF),
* identifies objects
* measures their science properties,
* calibrates photometry using PT data,
* stores results in files based on run, camera column, and frame.

SDSS Directory Structure

The SQL loader loads pipeline outputs into an SQL database. Loaded data includes science and most observational parameters, but not all intermediary results; data not loaded includes
* corrected frames
* point spread functions
* QA plots

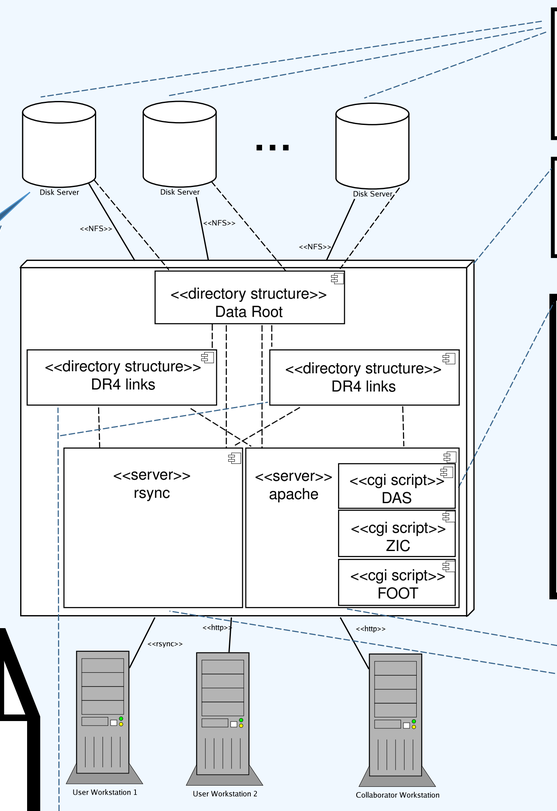
SQL Loader

The Catalog Archive Server provides high level access to the data.
The CAS does not require downloading any data to perform queries; users submit queries using a web interface, and the database server performs the query. (Of course, most users will want to download the results of queries.) Processing time and storage resources impose some limits on what queries users can perform.
Even when it is too high level to contain all the desired data, querying the CAS is often the best way to determine which files in the DAS contain the data of interest.

On-Line Database

Data Release 5 and supplemental runs contain over 12Tb of data in 17 million files, including (but are not limited to):
*** Corrected image frames**
*** Color jpeg images (unbinned and binned)**
*** Catalogs of objects found in each image**
*** Extracted and calibrated spectra**
*** Parameters measured from the spectra**
*** Astrometric calibration data**
*** Photometric calibration data**
*** Plots of the spectra**
*** QA Plots**
The directory structure organizes data by observing parameters, and does not make searching based on science criteria easy. File formats and contents are not always convenient to users. While most files follow fits standards, there are exceptions, and in many cases proper interpretation of the file contents requires detailed knowledge of the processing pipelines. Supporting their use by the public requires extensive documentation and effort by the SDSS help desk.

The Data Archive Server



A collection of linux boxes host the RAID arrays that actually hold the data, and serve the data directories to the DAS server over NFS.

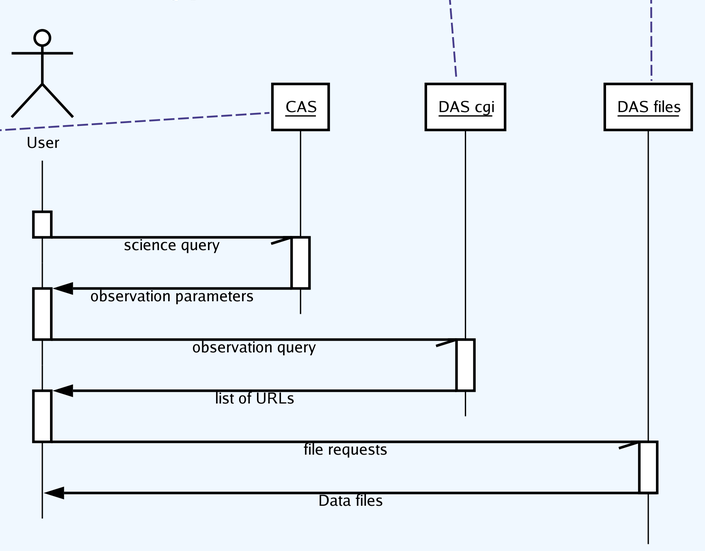
The DAS server is itself a linux box, running uncustomized versions of NFS, apache, and rsync.

A CGI script served by the apache server lets users generate lists of URLs or fully qualified file names based on the desired types of files and desired observations. Imaging observations must be designated using run number, camera column, and frame number, and specroscopic observations must be designated using plate number, fiber number, and observation date. Users can identify observations of interest using the CAS, or based on position using an additional CGI script.

Apache and rsync servers provide user access to the files in the data release directories.

Each data release has a corresponding directory tree containing symbolic links to the data that is part of the release. The directories cannot themselves be links, because the served directories often contain data that is not part of the data release.

A Typical Use



Verification and Validation of a Data Release

To ensure that we are serving the correct data, we have constructed tools that:
- verify that all files that should be served by the DAS are present,
- verify that no files that should not be served are, and
- check each file being served for corruption.

Administration tools read the definition of the data release from a collection of parameter files, and generate a list of directories and their contents based on that definition. In each directory, the tools generate and maintain a “file list file:” a list of files that should be present in that directory and their expected checksums, and store a list of the file list files that define the contents of all directories in the data release. Automated tasks then check the contents and checksums of files served by the DAS against the file lists.
This approach has several drawbacks, some of which may be addressed in future releases. First, corrupt files do not always get replaced by files identical to the originals. Some files are regenerated rather than restored from back up, in which case the data content will be the same but dates in headers may differ. The file list cannot be used to verify that the data content has not changed, and must be updated whenever this occurs.
The calculation of the checksum in a separate step from file generation introduces a second flaw: it is possible for the file to become corrupt before the checksum is calculated.
Finally, the storage of validation information in a custom format reduces their usefulness to users.
Several alternate approaches address some or all of these problems. Our fits writer could be modified to include checksum information in headers. Alternately, it could be modified to compress output files using GNU gzip, which includes a checksum. Unfortunately, neither of these approaches are practical for existing data. The custom checksum file format could be replaced by md5sum output, which would result in a standard format with minimal modification of our tools, but would only address the last of the known problems.

File Systems and Large Directory Structures

To publish a new data release on the DAS, we must generate a directory structure containing symbolic links to each of the files that constitute the data release. (Links to directories are not acceptable, because the directories hosting the data may contain data that is not part of the data release.)
This apparently simple task is surprisingly time consuming; generating the populated directory structure for the fifth data release, containing 14 million files, took more than 24 hours. The file systems we tried (ext3 and xfs) are better optimized for file creation than deletion, so removing an existing data structure was even more time consuming. Restoration of these directory structures from backup or regeneration to correct errors was therefore a problem.
We find keeping each data release on its own partition, and using a raw disk dump of the partition as a backup, to be a more practical solution.

Future Directions

While most users of the DAS retrieve only a handful of files, a handful of users retrieve large fractions of the available data. These users are typically generating local full or partial mirrors. The receiving sites are often overseas, which can slow the already long transfers. We are exploring the use of “P2P” software, particularly bitTorrent, to take advantage of existing mirrors to improve the download time by allowing clients to retrieve different parts of the data release from different mirrors.
While bitTorrent appears promising, several challenges must be met. The data release must be divided into “torrents,” which in turn must be divided into “pieces.” In the clients we have studied so far, memory availability limits the number of pieces a client can manage, the size of each piece, and the number of torrents managed. For bitTorrent to be used, we will need either to be very careful in the construction of our torrents, or construct a client that manages memory differently.