JSOC architecture overview*

The JSOC architecture consists of three principal components: a Data Record Management System (DRMS) for the storage and retrieval of information on all data, including ancillary data, in a relational database; a Storage Unit Management System (SUMS) for the physical storage and retrieval of large volumes of data; and an Applications Programming Interface (API) to facilitate interaction with DRMS and SUMS by analysis modules to be run in the pipeline (and outside as well), together with the analysis modules themselves and supporting libraries and utilities. There is also a Pipeline User Interface to control the flow of the automatic pipeline processing of AIA and HMI data from receipt of telemetry to the standard data products.

2.1. DRMS

The JSOC Data Record Management System is built around a relational (Postgres) database containing the JSOC data catalog. In the DRMS, data are organized into various data series of like data, each represented by a uniquely-named table in the database. Each row in a data series table corresponds to a data record, the atomic data object in the catalog. The columns of the table represent the set of (common) keys associated with each data record, and the values in each column of a given row represent the values associated with the corresponding key for the given record. A data record may include one or more named data segments, references to *n*-dimensional data arrays (*e.g.* images, data cubes) stored in files outside the table. It is possible to have rows in a data series table to link to rows in other tables. If desired specific columns in a table with such links can be linked to corresponding columns in the target records (rows). This allows for propagation of information to records in dependent data series. It is also possible to have data segments associated with the records in one data series linked to those of another series.

Data records may be selected explicitly by unique identification numbers, or more typically through SQL queries resulting in matching of one or more key values. A data set is defined as an arbitrary set of data records selected from one or more data series. An example would be the set of HMI Dopplergrams with Observation Times within a designated interval and having a Data Quality parameter exceeding a given threshold. Pipeline modules are designed to operate on data sets as input, and may result in the creation of new data records. Data records in the DRMS are never removed; when the data in a record are to be modified, a new record is created. For this purpose, each data series has one or more primary keys defined. All records with the same union of primary key values are assumed to refer to the same data object, and the one with the highest identification number will be selected in a match for the primary key(s). In this way, “header” information may be modified without having to rewrite the data segment.

The DRMS is designed so that it can be replicated in whole or in part at different sites. Both master and slave tables are implemented at the JSOC to provide robustness, and tables from other databases can be selectively mirrored on an individual basis, with different databases serving as the master for different tables. For this purpose the table name space is managed with uniquely-assigned prefixes. For example, all standard data products of the AIA and HMI missions (and only those products) are in data series named *aia.** and *hmi.**, respectively. We are using the

Slony-I system for mirroring and backing up of the JSOC databases. Remote mirrors are currently under development at Lockheed-Martin Solar and Astrophysics Lab, at the National Solar Observatory, and at Mullard Space Science Laboratory. We expect several others to be developed prior to the SDO launch.

2.2. SUMS

The SUMS is based on a combination of dedicated disk space and tapes in a robotic system. SUMS controls the JSOC data storage resources. It manages the disk space available for storing data and the tape systems, and it tracks the current on-line location of data. SUMS is implemented as a server program and management utilities. The SUMS server uses tables in another Postgres database to track "Storage Units". A Storage Unit is simply a directory and its subdirectories, if any. A Storage Unit can contain directories for one or more records from a single series.

SUMS gives each managed storage unit a SUMS identification in the form *Dxxxxx* where *xxxxx* is a unique serial number. SUMS allocates disk space on one of its file systems, creates a directory named with the unique identifier and provides the directory name and SUMS identifier to the requesting program (i.e. DRMS). The requesting program tells SUMS how long to keep the data online and whether or not to archive them to tape. The SUMS disks at the JSOC are configured in a RAID system and should provide secure storage online so that only the most valuable or largest collections of data need be archived to tape. SUMS also provides mechanisms for grouping data from related data series on a well-defined set of tapes.
The SUMS architecture is designed to be distributed, like the DRMS. Each DRMS instance must have its own SUMS, though it can be as small as a single dedicated disk partition, It should be possible for cooperating DRMS systems to share information about data in selected series, and to make available data that happen to be cached online at one location to another location over the network if that would be faster than staging the data from tape at the archive site. Bandwidth costing needs to be performed among the different sites.

There is a documented API for direct interface to SUMS, but it is probably not of interest to module writers; it is used internally by applications involving the DRMS API. Documentation for the DRMS API is in progress.

2.3. DRMS API and analysis modules

Data in the DRMS may be read and written with direct *psql* commands (and proper permissions of course). It is expected that application programs, also known as analysis modules, that require interaction with the DRMS will do so through a managed socket-level communication interface with the database. For this purpose, a library of C-language functions has been made available.

In order to use the standard interface to the DRMS embodied in the API, C-language analysis modules must be implemented as functions (named *DoIt()*) to be linked to a common *main()* that initiates and manages the communication with the DRMS, passing on the environment and calling parameters directly to the module.

Because the DRMS/SUMS system is designed to support temporary, non-archived data sets of arbitrary cache lifetime, it may be desirable to isolate the many different

steps of an analysis sequence (*e.g.* detrending, tracking, domain transofromation, filtering, model fitting, inversion) as distinct modules, using temporary intermediate data products. Such intermediate products may also be shared among different possible branches of one or more pipelines.

*This description was adapted from Local helioseismology in the SDO HMI/AIA data analysis pipeline, Bogart, R. S., Astronomische Nachrichten, Vol.328, Issue 3, p.352, 03/2007.