Astronomical Data Analysis Software and Systems IV
ASP Conference Series, Vol. 77, 1995
R. A. Shaw, H. E. Payne, and J. J. E. Hayes, eds.

Storing and Distributing GONG Data

M. Trueblood, W. Erdwurm, and J. A. Pintar

National Solar Observatory, National Optical Astronomy Observatories[1]*, P.O. Box 26732, Tucson, Arizona 85726-6732, USA*

Abstract. The Global Oscillation Network Group (GONG) helioseismology observing network will consist of six instruments deployed worldwide to provide nearly continuous observations of the Sun beginning in 1995.

Data reduction is performed on a network of high-performance UNIX workstations to process the data and store them on Exabyte 8-mm cartridges. The single observed object (the Sun) and other constraints imposed by the nature of the project permitted developing a more robust and less expensive DSDS than is possible for open-ended general institutional support systems performing similar functions. For example, the data product file catalog was compressed by a factor of over 160 to a series of bitmaps that permit the DSDS to provide good query response to several simultaneous users on a workstation. UNIX interprocess communication and networking were used to develop a mirrored database between two DSDS workstations, providing a high level of DSDS availability to support data reduction pipeline operations.

The Global Oscillation Network Group (GONG) will record Dopplergrams of the Sun once per minute over three years. The data will be reduced at the Data Management and Analysis Center (DMAC) in Tucson, Arizona through a pipeline consisting of a network of workstations. The approximately 3 TB of data will be stored on approximately 10,000 8-mm Exabyte cartridges and will be managed and distributed by the Data Storage and Distribution System (DSDS). Details of the expected science return and data products to be produced are given in Kennedy & Pintar (1988).

1. DSDS Design

Unlike most other astronomical observatories, GONG observes only one object with a single name, and the celestial coordinates of that object at the time of observation are not important to the user community. This permits us to limit query keys to data product type and time of data acquisition. Some users want

[1]The Global Oscillation Network Group (GONG) is an international community-based project funded principally by the National Science Foundation and administered by the National Solar Observatory. NSO is a division of the National Optical Astronomy Observatories, operated by the Association of Universities for Research in Astronomy, Inc. under a cooperative agreement with the National Science Foundation.

the ability to query the catalog for data products taken at the same time as certain solar events. Consequently, we provide the means for users to add their own software to the query system to support such "correlated queries".

Wide area networks are used to distribute up to 100 MB of data per user per day. Requests for more data are distributed using removable media. Since the GONG image file size is relatively small (130 kB) and to keep data distribution straightforward, the DSDS designers decided to refrain from making spatial subsets of data, and to make the process of assembling temporal subsets easy by placing each separate data product instance (e.g., a one-minute image) in a separate disk file. This reduces the problem of distributing data to merely copying files from one cartridge to another using operating system utilities. The DSDS software developers needed no knowledge of data file internal formats, since they had no need to develop custom software to read individual data files.

Operators of other data centers suggested that we keep file names short and meaningful to minimize operator errors. GONG data file names are formed from the data product type and time of data acquisition using only lower case letters and numbers. The first file on each library (archive) cartridge contains a "table of contents" listing the names of all data files on the cartridge.

The approach of placing each data product instance in a separate file means that the file catalog will contain almost 75 million data file name entries after a three-year project. Conventional commercial DBMS products would require an unacceptably long time to execute a simple query on this many table entries, even if the file catalog were divided into several smaller tables. Furthermore, there is sufficient user interest in a catalog that can be queried on users' home institution computers to justify a query system that does not require each user to purchase a commercial database product.

To solve these problems, the DSDS designers "compressed" the file catalog by defining a file for each data product containing a bit for each possible time slot over the three-year GONG project. A bit is set in the file if the library contains that time slot's data product file. That is, in the case of images, since there are 1440 time slots (minutes) in a day, 1440 bits are used to represent a single day of a single image data product. A three-year period of a single image data product can be represented by 1440 bits $\times$ 365 days $\times$ 3 years = 1,576,800 bits or 0.2 MB. With approximately 200 data product types defined to date, many of which are produced less frequently than each minute, the entire data archive can be represented by a collection of files, one per data product type, with a total storage requirement of about 20 MB. Users without network access to the DMAC Users' Machine who have ported the query software to their own computers need to have on their home systems only those bitmap files corresponding to data products of interest, so a single investigator's collection of bitmaps might consume only 5 MB of disk space, which can fit on a few floppy diskettes. Performing a query consists of specifying a data product and a time, opening the bitmap file for that data product, and checking the single bit corresponding to the specified time.

The menu query system consists of three parts: a program that generates the menus, receives user keyboard input, and generates a "query file" as its output; a program that takes a query file as input and generates a "hits list" and an optional "misses list"; and a UNIX script that calls these other two parts.

The query file defines the scope of a query. Each record or row of the file is a file name that specifies the data product type and a single time slot. Although the menu system generates the query file, a user could generate the query file using custom software or even a simple text editor. The hits list is a file of similar format containing all the file names that satisfy a query, and the misses file lists the files in the space defined by the query not in the library. When the hits list reflects the data subset the user wants, the user runs a program that reads the hits list and generates a data request. The DSDS operator fills the data request by extracting the appropriate cartridges from the library, and copying the requested files to the distribution medium that was requested.

In addition to the menu system, the DSDS provides a query tool written in C that can be used interactively to see if a particular data product for a single time is in the database. It also generates hits and misses list files. The same program can be called from a UNIX script (command file) written by the user to form the query. The query tool uses a typical UNIX command form in which the data product name and the date and time are specified using input parameters in the UNIX command line that invokes the program.

The file catalog for tracking the location of each data file was designed around the method of storing data files on Exabyte cartridges. Pipeline operators store groups of disk files on tape using the UNIX *tar* program. Each *tar* file contains files of only one data product type for a 24-hour period, up to 1,440 image files. The file catalog, instead of listing each individual data file, lists only each *tar* file on a tape and the beginning and ending dates/times of the data. When a new data tape is checked into the DSDS, a row representing each *tar* file on the tape is inserted into the file catalog database table and the bits in the bitmap files corresponding to each new data file are set. The bitmap files are then copied over to the Users' Machine for immediate use in queries. The bitmap "fills in" the individual time slots between the beginning and ending dates/times in the file catalog database table, listing the availability of a data file for each possible time slot. This design compressed the file catalog by a factor of 160 over using RDBMS tables to store all file names. On-line data are handled in a similar way, with the pipeline collecting on-line files into pseudo-*tar* groups.

Another feature of the file catalog design is the ability to store all versions of a data file. The bitmaps reflect only whether a file exists in the archive, not its version, so queries on version numbers are not permitted and only the latest version of data are distributed in routine operations. But if a situation arises in which the science requires access to a previous version, this can be handled as a special request that must be approved by project management.

2. Database Mirroring for High DSDS Availability Levels

The DSDS is the central hub of the DMAC through which all Exabyte data cartridges must pass from one pipeline stage to another. If one stage of the pipeline goes down, other stages can continue processing data until no more input tapes are available. If the DSDS goes down, the entire DMAC stalls. To provide a DSDS with a high level of availability to the DMAC, the database is mirrored on two workstations, and DSDS applications are designed to run on either workstation. During normal DMAC operations, an operator on either DSDS workstation can perform any DSDS function and the results appear on both workstations in near real-time (within a few seconds). This is achieved using UNIX message queues and sockets (communications links). When an application reads from the database, the read is performed only on the local workstation. But if an application writes to the database, it first writes to the local database. If the local database is updated without error, then the application places a Structured Query Language (SQL) command or a bitmap update command in a buffer and gives the buffer to a routine that places it on a local message queue. A database mirror daemon removes the buffer from the queue and sends it over a socket to the other (remote) DSDS Operations Machine (OM), where a daemon receives the buffer from the socket and places it on another message queue on the remote OM.

On the remote OM, one of two daemons removes the buffer from the message queue. If the buffer is a bitmap update command, the bitmap daemon dequeues the buffer and updates the bitmap. This daemon processes bitmap update commands from both the local and remote OM's, and sends a new bitmap to the Users' Machine when it receives a buffer coded to indicate that updates to the current bitmap are complete. If the command is an SQL command to the database, another daemon removes the command buffer from the queue and performs an SQL EXECUTE IMMEDIATE on the buffer. This process is repeated in the opposite direction, enabling either OM to perform any DSDS function and keep the other OM's database current.

When a socket goes down (such as when an OM fails), or if an OM daemon tries to send a buffer to the remote OM, it receives an error from the socket. The daemon then places the buffer in one of two files on the local OM, depending on whether the message is an SQL message or a bitmap update message. When the remote OM comes back up, the remote database is brought up, and all sockets and daemons are reestablished. The DSDS operator on the remote OM then copies over these files and runs a recovery program that reads the files and places the buffers back on the message queue for processing. This permits one OM to keep processing by itself while the other OM is down, and permits the DSDS to weather most single points of failure (excluding power failures, which bring down the entire DMAC, but which are rare enough and of short enough duration to be no more than a temporary nuisance).

References

Kennedy, J. R., & Pintar, J. A. 1988, in Astronomy From Large Databases, eds. F. Murtagh & A. Heck (Garching, ESO), p. 367