

MODIS QUARTERLY REPORT - MARCH 1992

DR. ROBERT H. EVANS
UNIVERSITY OF MIAMI
RSMAS/MPO

NAS5-31362

=====
Due to the interlocking nature of a number of projects, this and subsequent reports will contain coding to reflect the funding source. Modis funded activities are designated with an M, SeaWIFS with an S, Pathfinder with a P, and Headquarters with an H.

=====
Given the length of this report, it deviates from the requested format in that each section contains the concepts, status, problems, and future efforts. There are three sections:

- I. Database
- II. Client/Server
- III. Data Matchup

I. DATABASE SECTION

- I.a NEAR TERM OBJECTIVES
- I.b OVERVIEW OF CURRENT PROGRESS
 - I.b.1 Database Creation
 - I.b.2 Thumbnail Sketch
 - I.b.2.1 SATELLITE Database
 - I.b.2.2 AUTOPROC Database
 - I.b.2.3 IN_SITU & ASSOCIATED Database
 - I.b.2.4 MATCHUP Database
 - I.b.3 Automatic Processing
 - I.b.3.1 Pre-Processing Actions
 - I.b.3.2 mcp
 - I.b.3.3 dbbat
 - I.b.3.4 Client/Server
- I.c FUTURE ACTIVITIES
- I.d PROBLEMS ENCOUNTERED

=====

I.a NEAR TERM OBJECTIVES

Database concepts will continue development and the integration of the database with the client/server will be expanded. Contents of the various databases will be augmented.

I.b OVERVIEW OF CURRENT PROGRESS

Overall, there are five databases planned: 1. A SATELLITE database containing information about the satellite scenes that have been spooled, ingested and are in various stages of processing; it will also contain the archive information of the products. 2. An AUTOPROC database will carry the information needed for the automatic processing of the satellite data and track the scenes as they pass through the processing stages. 3. An IN_SITU database will hold information on buoys, drifters, ship data, etc., all in situ measurements. 4. An ASSOCIATED database will deal with environmental data that cannot be termed 'measurements,' such as gridded fields and numerical model output. 5. A MATCHUP database, will link the SATELLITE data and the environmental surface [in situ] data used in calibration and validation.

I.b.1 SQL Database Creation (M)

The SATELLITE and AUTOPROC databases are created using batch jobs. Each table in a database is created using an SQL procedure file, which also inserts static data into those tables that may be used for field validation (satellite, sensor, archive, etc.).

I.b.2 Database Survey

I.b.2.1 SATELLITE Database (M)

Each satellite pass (or scene) is represented by a MAIN record in the database. The MAIN record may be generated in a number of ways: at the time of spooling, the time of ingestion, or later. This MAIN record contains all information about the scene (satellite, sensor, transmission, etc.).

Numerous tables also reside in the SATELLITE database, to be used for validation of fields on record entry and during database queries, and to track archive information. The majority of these tables will be used for field validation.

I.b.2.2 AUTOPROC Database (M)

This database contains the same relations as the SATELLITE database, and additional tables containing the automatic processing information. If a scene is to be processed as a single unit, one PROCESS CONTROL record (PCR) is created in a PROCESS-CONTROL relation. This record contains the information used in automatic processing, such as procedure(s) to be used, process step completed, and so forth. If a scene is too large to be handled easily as a whole, it may be broken up into pieces for processing, currently at the time of ingest, and a PCR is created for each piece. The processing procedures themselves are stored in the database.

I.b.2.3 IN SITU & ASSOCIATED Databases (P)

The IN SITU and ASSOCIATED databases are in early development, and may undergo significant changes. IN SITU is being designed as a general-purpose database that must accommodate many different types of data.

I.b.2.4 MATCHUP Database (P)

The MATCHUP database contains co-temporal, co-located satellite and in situ data. A prototype of a MATCHUP main record exists. Decisions and final development of the MATCHUP database can be finished when the automatic processing of the satellite data is complete. The MATCHUP process is more completely covered in Section III.

I.b.3 Automatic Processing (M)

The automatic processing capability controls the batch processing of satellite data, and is currently working in the SATELLITE database, although it should be adaptable to processing other types of data.

I.b.3.1 Pre-Processing Actions (M)

There are three steps that must be done before the automatic processing can be used: the database must be created, and a MAIN record and a PROCESS CONTROL record (PCR) must be entered into the database.

I.b.3.2 Master Control Program (mcp) (M)

A command file is used to start the Master Control Program (mcp) in batch. This program directs the spawning of other batch jobs, one for each PCR that is ready for processing. That is, mcp queries the database for the next PCR that is ready to be processed, and submits a batch file (dbbat) assigned to process the file associated with that PCR.

mcp keeps track of the number of jobs that are submitted, executing and completed. In the final form , mcp will retrieve not only the next record to be processed, but also the processing steps to be used.

I.b.3.3 dbbat (M)

When mcp spawns off a dbbat job, that job is associated with one PCR in the database. It will be modified to accept the processing steps from mcp, and then start DSP and execute the required steps.

I.b.3.4 CLIENT/SERVER Processing (M)

The autoprocessing software is currently working on a VAXStation 3200 running VMS 5.4 and using RDB/VMS as the database engine. The conversion to distributed processing on UNIX computers is in progress. The client/server database is more completely covered in Section II.

I.c FUTURE ACTIVITIES

The database will continue to expand in contents and capability.

I.d DATABASE PROBLEMS

I.d.1 Inefficient Schema

After consultation with a database expert, it was determined that the extensive use of keyword codes in the database schema, while saving disk space, would slow down queries and modifications of the database. Therefore, the schema were revised to eliminate the use of table codes. This involved not only the revision of the database creation files, but also the in-house interface (an extensive set of programs and subroutines) that is currently used with the satellite and autoprocessing relations.

I.d.2 RDB/VMS Contact only.

Previously, the database could only be used under a VAX RDB/VMS database. The creation files were in RDB's RDO Interactive Interface files, and static data was loaded by special purpose files using GENERIC, one of the in-house programs. Additions to, modifications of, and retrievals from the database were made using the in-house interface, which used the RDB/VMS-specific RDML (Relational Data Manipulation Language) statements embedded in FORTRAN programs. Currently, the creation and loading files have been converted into SQL procedures. It is intended to replace the RDML statements with their

equivalents in embedded SQL. This can be done and tested on the VAX machines, as our VMS/V5 machine has an RDB database that includes an SQL interface. Once this conversion is made, the system should operate on any SQL-based database.

I.d.3 VAX/VMS use only.

The in-house interface makes extensive use of VAX/VMS Operating System library calls and error-handling. These VMS-specific calls can be replaced by more general use subroutines that are compatible with any operating system.

=====

II. CLIENT/SERVER SECTION (M)

II.a NEAR TERM OBJECTIVES

Development of the design and implementation will be extended; further testing of client server will be continued.

II.b OVERVIEW OF CURRENT PROGRESS

II.b.1. Client/Server Concept

II.b.2. Our Proposed Client/Server Model II.b.3. Current State

II.c FUTURE ACTIVITIES

II.d PROBLEMS ENCOUNTERED

II.a NEAR TERM OBJECTIVES

Continued development of the client/server is the primary objective. Assuring adequate resources for design and implementation of the monitoring functions is also an important near term goal.

II.b OVERVIEW OF CURRENT PROGRESS

II.b.1 Client/Server Concept

The most commonly used paradigm in constructing distributed applications is the client/server model. In this scheme, client applications request services from a server process. Nominally, a server provides network services; a network service is a collection of one or more remote programs. These remote programs implement one or more remote procedures. The client and server operation is based on a known set of conventions that must be implemented at both ends of a connection before service may be rendered (and/or accepted). This set of conventions comprises a protocol that may be symmetric or asymmetric.

RPC, Remote Procedure Call, is used to implement the client/server model. RPC is a high-level communications program that allows network applications to be developed using specialized procedure calls, providing a degree of independence from the underlying networking mechanisms. The RPC model is similar to a local procedure call model in that one thread of control logically winds through two processes -- the client's process (the caller) and the server's process (the procedure called). The reliability of an RPC model depends on the reliability of the transport protocol underneath it. For this reason, this implementation RPC is running on top of TCP/IP.

The caller sends a message containing the required parameters for the requested procedure to the server process and awaits the results. On the server side, a process is dormant waiting the arrival of a call message; upon arrival, the server process activates, extracts the procedures' parameters, computes the results, sends a reply message, and is deactivated. Thus the server process activates, services the client request, and performs whatever appropriate actions the client requested. Once the reply message is received, the caller's execution resumes and the results are processed.

II.b.2 Proposed Client/Server Model

Our mcp [master control program] system is a distributed application over a VAX/VMS and UNIX network. The complete mcp system consists of the database, resource monitor, dbbat [a command procedure generator], performance monitor and mcp itself. The monitor programs are machine/system dependent.

The database and its attendant programs, acting as the server, initially will be VAX resident. The rest of the mcp system resides on UNIX platforms where multiple copies will be able to access the database. The connection between these two parts is accomplished using the client/server mechanism. This allows more efficient use of resources by moving computations previously done in the VAX/VMS to UNIX systems and creating a distributed client environment.

The mcp is responsible for the control of processes accessing the database. When mcp starts, it will enter an endless loop executing periodic calls to the resource monitor to determine whether there are sufficient resources to start a new dbbat process. If insufficient resources are available, then the mcp process will return to an inactive state. Otherwise a dbbat process will be started. The dbbat process will query, through the client/server mechanism, the database about the status of tasks, if any, to be performed. Parameters encoded in the message include service type and server's name [in case of multiple servers]. We have defined three service types at this time. Type 1 is the initial client contact [dbbat] with the database, and requires the database to be invoked. Type 2 reflects continuing dbbat communication with the database, but the database has already been

invoked. Type 3 indicates a job has completed in UNIX environment and the database table, `process_control`, needs to be updated to reflect the completion.

When a server activates, it provides service to the client by running the procedure that will contact the database. If the client requests a type 1 service then a procedure to invoke the database will be executed first.

With type 1 or type 2 service requests, a procedure, `dbconnect`, will execute which in turn runs `db control` to check the process control table first to determine status. If there are records marked `SUBMITTED`, an array is created containing steps to be carried out to process the specified data file; the array will be converted into the reply message. If the client requests a type 3 service, i.e., update the database that a job has completed, the job status is changed to `FINISH` in the `process_control` table.

To avoid creating excess traffic over the network, we plan for the client/server to communicate only twice for each job. First communication is the client's request for a job and the server's response in returning the steps of a job from the database over the network. Second communication occurs when the job completes and `dbbat` sends a message to update the job's status in the process control table.

II.b.3 Current State

Although the entire system has not been implemented, the mechanism of this client/server model has been established. We have a working `mcp`, `dbbat`, client and server. The database has been redefined with SQL. The client/server has been tested on the old `RDML`, the `RDO` defined relations, as well as the new SQL defined the database. The server can provide type 1 service. The server will invoke the database, fetch all the required steps and pass them to the client in the reply message. The client then will decode the message and write the message into a `.dsp` file. This `.dsp` file will serve as a command procedure for `dbbat` process. `Dbbat` requests a job through the client and gets it back as a command procedure to be executed.

Our first test used a `CZCS` data file and three command procedures for `L2` conversion, `L3` conversion, and a mapped image file. Some appropriate steps reflecting these three command procedures were put into the database. `Dbbat` obtained the job steps from the database through the client/server, and when the command procedure was executed, a mosaic image was produced.

Our second test used `AVHRR`. The database is defined in SQL. In this test, our server was able to fetch the steps and pass them back to client. Although the client wrote a command procedure, it was not executed.

The type 2 and type 3 services are being coded.

II.c FUTURE EFFORTS

Work will continue on types 2 and 3 services. We will need to develop a resource manager and performance monitor in concert with GSFC. We also need an error handler, which will be easier to write when our system is more fully defined.

II.d PROBLEMS ENCOUNTERED

II.d.1 The client/server was simplified to minimize message exchange.

II.d.2. Determination of the number and timing of interprocess messages.

In an attempt to reduce network traffic, interprocess communication has been deliberately held to a minimum. The initial plan was to have dbbat communicate through client/server twice with the database, first time to acquire a job and second time to indicate job completion. While this plan worked, there were problems when there were no pending jobs. In this case, mcp would fork dbbat whenever resources are adequate; this wasted resources. The design was modified to let mcp contact the database. When resources are adequate and there is a job for dbbat, then mcp will activate a dbbat. When the task is complete, mcp will send a message to update the database.

II.d.3. We need to have a mechanism to control the batch jobs, the dbbat forking must be monitored and queued. A method for dealing with abnormal termination of dbbat jobs is required.

The release of dbbat batch jobs must be monitored and queued in a controlled manner. The queuing mechanism development must be completed. Further development is required to provide a progress monitoring facility and allow the query and manipulation of the batch queue. This reinforces the requirement to develop a complete monitoring package for the system.

=====

III. DATA MATCHUP (P)

III.a NEAR TERM OBJECTIVES

Adding additional matchups to the database is clearly a near term objective. Examining ancillary datasets to determine their applicability within the matchup concept will continue.

III.b OVERVIEW OF CURRENT PROGRESS

A MATCHUP database includes various in situ and satellite quantities, coincident (or nearly coincident) in space and time. The MATCHUP database can be used to monitor sensor calibration and to test and improve geophysical algorithms. We have been developing a general methodology for doing matchups. There are four main steps:

- a. Compilation of in situ sea surface temperature (SST) and other environmental data.
- b. Development of a generic methodology for the identification of NOAA spacecraft orbits that coincide (within a specified time-space window) with in situ observations.
- c. Extraction of AVHRR data corresponding to times and locations where in situ data exist.
- d. Matchup of in situ and AVHRR data.

Step One involves compiling and reformatting the in situ data, primarily fixed and drifting buoy data. Reformatting is mainly a redating; it is easier to do matchups if the time coordinates are continuous. The date is converted into seconds relative to a reference data.

Step Two is the generation of a reduced list of in situ data. We have chosen to use orbit routines in DSP to generate a list of "times of closest approach" (TCAP). The TCAP methodology is generic in nature and can be used for any spacecraft, provided an orbital model is available. Its advantages are: (a) it reduces the volume of in situ data to consider further, (b) it provides a limited list of satellite orbits from which data are to be extracted.

Two checks are made; the first ensures that the observations fall within a specified temporal window and the second check ensures that the in situ location falls within the area scanned by the sensor. These steps produce a series of times, lats and lons of in situ observations, which will be fed to the extraction routines (the following step). This step excludes a significant amount of the in situ data.

Step Three extracts satellite data for the times and locations specified above. This step is sensor-specific and a decision must be made what data to extract. For example, using AVHRR, we extract data for a 3x3 pixel box centered at the in situ location for all 5 channels, plus geometric and sensor info (sat zenith angle, solar zenith angle, baseplate temperature).

Step Four merges the in situ and satellite extractions, building the MATCHUP data set. This step involves a further verification that the data fall within the specified time/space windows. The MATCHUP database can then be used for calibration and algorithm development and validation.

The experimental MATCHUP database is now being used to test new SST algorithms and criteria for cloud identification

III.c FUTURE ACTIVITIES

The MATCHUP database will be expanded and testing using the database will continue.

III.d PROBLEMS ENCOUNTERED

III.d.1 The GSFC group will no longer participate in the compilation of moored buoy data. We will need support to continue this activity (and we have already received some 1981 buoy data from NODC).

III.d.2 Similar to Problem 1, it involves lack of programming support for the compilation and processing of drifter data sets.

III.d.3 There was an error in our TCAP routines that results in lack of prediction for some locations.

Jim Brown has examined the prediction problem and, as a result, acquired a new set of TCAP routines from D. Baldwin (U. Colorado). After evaluating the implementation of the new TCAP program, it was determined that the improvement overcame many of the known problems. At this point, lists of global extractions can be generated.