520-EMD-001

Data Flow to Modelers Test and Benchmark Report

Technical Paper

June 2005

Prepared Under Contract NAS5-03098

RESPONSIBLE AUTHOR

Jon W. Robinson /s/ 6/30/2005

Jon W. Robinson, Ph.D. Date
EOSDIS Maintenance and Development Project

RESPONSIBLE OFFICE

Timothy Ortiz /s/ 6/30/2005

Timothy Ortiz, Synergy FY04 IPT Lead Date
EOSDIS Maintenance and Development Project

Raytheon Company
Upper Marlboro, Maryland

This page intentionally left blank.

Data Flow to Modelers Test and Benchmark Report

By:
Jon W. Robinson

Introduction

The purpose of this project was to improve the flow of Earth Observation System (EOS) data from the Distributed Active Archive Centers (DAAC) to Earth system modelers. The Project involved two phases. The first was to identify the impediments to the use of EOS data by earth system modelers and then propose a system to ameliorate the impediments. The second stage was to implement a prototype of the proposed system and carry out test and benchmark studies to identify the principal considerations for implementing an operational version on new hardware. This document describes the proposed system that was developed as a prototype and the results of EMD Synergy Team's test and benchmark activities.

During Phase 1, the EMD Synergy Team interviewed scientists working with the Global Modeling and Assimilation Office (GMAO) and determined that there were two main obstacles to their use of EOS data. The first obstacle was the difficulty in dealing with HDF-EOS data format, the second was the large amount of raw data that needed to be acquired and stored relative to the size of the data sets actually assimilated into global models. Presently, large amounts of high-resolution data must be acquired by global modelers and then reduced to a resolution suitable for their models. This resolution reduction results in a data set that can be one or more orders of magnitude less in volume than the raw data used to generate it. This second obstacle was resolved by moving the data reduction activity to the DAAC. Because the scientists frequently change their data reduction algorithms, moving data reduction to the DAAC, required providing the ability for scientists to easily upload their modified resolution reduction algorithms to the DAAC for execution. The first issue was resolved by allowing the scientists to provide their own I/O libraries to the DAAC for use in algorithms they upload. Placing the resolution reduction algorithm at the DAAC provides access to high speed, internal DAAC networks, which means that the massive amounts of input can be staged for processing much more quickly than if transmitted through the Internet first.

While working on the conceptual issues involved in developing such a system, we became aware of an existing system that already had some of the elements the proposed system required. That system was the Near Archive Data Mining (NADM) system that ran at the Goddard Space Flight Center (GSFC) GES-DAAC. Its main deficiencies were that it could only search the Data Pool for data, not the DAAC archive, and could only handle one input file and one output file at a time. The system that the EMD Synergy Team proposed would need to handle multiple input and output files. EMD Synergy Team invited key staff from the GES DAAC to discuss the prototype system needs and as a result developed a collaborative development strategy for implementing this prototype system.

The prototype system was designed as an enhancement to the Simple Scalable Script-based Science Processor for Mission (S4PM) system. Because the enhancement was going to be based on the NADM data mining system, this new variant was named S4PM-DM.

Structure of the S4PM-DM System

There are three main elements to the prototype system.

- There is the Simple, Scalable, Script-based Science Processor for Missions (S4PM),
- there is the Algorithm Upload and Management Module (AUMM) and
- there is the Data Search and Order Module (DSOM).

These latter two elements are integrated into S4PM, which then becomes S4PM-DM. A detailed description of the system architecture is provided in "S4PM -- Data Mining Edition, Implementation Plan"[1].

The AUMM was adapted from the NADM capability and was developed at the Goddard GES-DAAC. The EMD Synergy Team developed the DSOM. Integration of these two elements into S4PM was a cooperative venture between the GSFC GES-DAAC EMD Synergy Team IIS and GMAO.

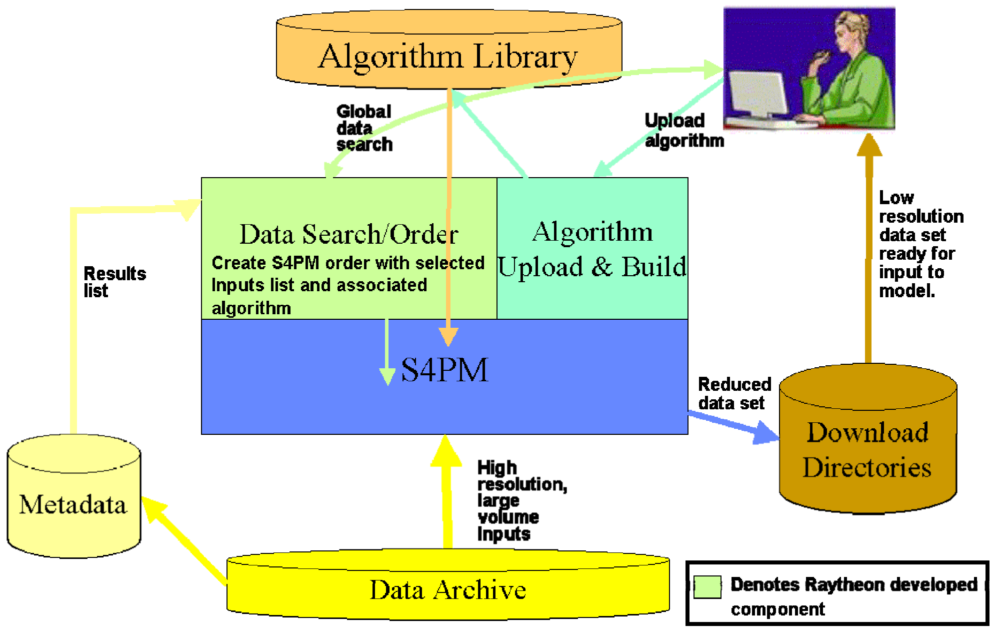

Algorithm Library
Global data search
Upload algorithm
Data Search/Order
Create S4PM order with selected Inputs list and associated algorithm
Algorithm Upload & Build
Results list
S4PM
Low resolution data set ready for input to model.
Reduced data set
Download Directories
Metadata
High resolution, large volume Inputs
Data Archive
Denotes Raytheon developed component

Figure 1. Logical Structure of Prototype S4PM-DM System

The system structure is shown in Figure 1 above. The first step is for the user to upload his algorithm to his string[2]. Before the actual upload of the algorithm, the user must

[1] Currently in Version 0.4

provide a description of the algorithm. This includes I/O requirements, library requirements, and choice of compiler and compiler flag settings. Because of security issues, libraries must currently be built and installed by the DAAC staff. Alternative methods of dealing with algorithm library needs are being examined at the present time.

Once an algorithm is installed, the user may go on line and order data for it to process. Alternatively, the user can enter a subscription, which will initiate the algorithm whenever the appropriate inputs are available, either through new data acquisition or a scheduled reprocessing.

The S4PM-DM stations can all reside on one computer, as with our prototype, or they can be distributed across a number of computers networked together as at the GES-DAAC. The only requirement for a distributed computing environment is that the data be accessible to all of the S4PM-DM stations.

User Response

During the development of the prototype, EMD Synergy Team and GES-DAAC worked closely with the primary user, GMAO, to meet their primary needs. The staff scientists at GMAO are enthusiastic about the S4PM-DM system and see it as a major step towards making the assimilation of EOS data much easier than under the current system. Two of the scientists, Dr. Michael Bosilovich and Dr. Peter Norris provided test algorithms and a third scientist, Dr. Arlindo da Silva has pledged a fourth algorithm once the prototype is done.

Dr. Bosilovich participated as the “user” during the final prototype demonstration held Friday June 10, 2005 and not only expressed his strong support for the project but also saw it as a capability that would be very useful to scientists working at academic institutions. This is because a major burden of producing a data set ready for assimilation would be carried out at the DAAC, thus relieving the academic scientist of the storage and processing burden of resolution reduction.

Prototype Hardware

Our prototype hardware system consists of a Dell Power Edge 6600 with four 2.7 GHz. Xenon CPUs, each with 512 Kilobytes of L2 cache and 2 megabytes of L3 cache, and a 400MHz bus speed, 1 Terabyte of SCSI RAID storage and 4 Gigabytes of memory. The operating system is Redhat Enterprise Linux Server. The disk storage system is divided into two sections, identified as Dev8-0 and Dev8-1. Dev8-0 handles basic OS support and swap space. Dev8-1 is used for data storage. This prototype machine was located in the VATC, which provided a DAAC like environment for development and testing.

[2] A user's ***string*** is a full configuration of S4PM stations with disk and CPU resources allocated to that user.

Benchmark Runs

The purpose of the benchmark runs was to identify which hardware resources would be most stressed in an operational S4PM-DM environment. The goal was to use this to provide guidance in assembling a hardware system for hosting S4PM-DM. Because S4PM-DM will be available to the world as Open Source code, we wanted to provide some variation in load levels in order to see how hardware resources became exhausted with heavier usage. Toward this end the EMD Synergy Team ran two scenarios, a light load and a heavy load. The light load was used to determine behavior of the system hardware when the demands were easily met and a heavy load was used to determine behavior when software system demands used close to a one hundred percent of the hardware's capability.

As is described below, the EMD Synergy Team found that with a light load I/O bandwidth was the limiting factor, and with a heavy load, memory, I/O bandwidth, disk speed and swap space all conspired to throttle system throughput.

Data on the performance of the benchmark runs was extracted from the System Activity Reporter (SAR) monitoring program that comes as a standard with the Redhat Linux system. The SAR was set up to sample the system every three minutes. While there are 4 Gigabytes of memory in this system, the OS uses most of it regardless of the load, setting up large caches even when executing processes do not demand large amounts of memory. Therefore, the team decided to use percent of swap space used as a stand in for memory usage. The graphs below indicate that almost no swap space was used for the light load runs, while swap space use approached 100 percent for the heavy load runs. Disk usage was measured by using the sectors/second of I/O from the two physical disk systems Dev8-0 and Dev8-1.

The benchmark runs used data that were already staged to the prototype machine's RAID disk system. This was because the EMD Synergy team already understood that the chief bottleneck for these algorithms was staging the data from the archive to the machine that executed the algorithm. Since the speed of retrieval of data from tape in the archive cannot be changed and since the machine-to-machine communication speed is typically fixed within a facility, we felt it was reasonable to exclude these steps from our benchmark runs.

Three different algorithms were used for these benchmark runs. Each algorithm was developed by staff scientists at GMAO. The first algorithm was developed for Dr. Michael Bosilovich and was tested with seven days of MOD11_L2 (land surface temperature) data. The second algorithm was developed by Dr. Peter Norris and was tested with seven days of MOD06_L2 (cloud) data. The third algorithm again developed by Dr. Michael Bosilovich was tested with five days of MOD05_L2 (aerosols) data. Two sets of benchmark runs were carried out, one on May 25, and the second May 27. The first run on May 25 was a light load run (see Table 1 for timing and number of instances of each algorithm run) and the second run on May 27 (see Table 2 for timing and number of instances of each algorithm run) was a heavy load run.

The May 25 run involved several individual runs and then a combination run. Table 1 below gives the time line for the runs on May 25. The table gives time in both hours of the day and fraction of the day because the 3-D graphics program used to create the graphs could not handle the time format. The time axis in the 3-D graphs is in decimal fractions of the day.

Run Type	Algorithm	Hour of Day		Fraction of Day	
		Start	End	Start	End
Single	MOD06_L2	14:06:44	15:39:21	0.588009	0.652326
Single	MOD11_L2	15:45:09	17:28:31	0.656354	0.728137
Single	MOD05_L2	17:49:17	18:17:49	0.742558	0.762373
Comb 1	MOD06_L2	18:20:44	20:34:37	0.764375	0.857373
Comb 1	MOD11_L2	18:20:44	20:43:54	0.764398	0.863819
Comb 1	MOD05_L2	18:20:46	19:02:39	0.764421	0.793507

Table 1. May 25, 2005 Runs.

Figure 2 below shows the CPU and Disk I/O loads over the course of May 25. This includes time both before and after the benchmark runs to provide a "no activity" baseline. The time of day is given in decimal fractions of a day with 1.0 day equaling 24 hours. The Z axis, disk usage, measured in sectors/second, is on a logarithmic scale in Figure 2 and on a linear scale in Figure 3. Figure 3 has percent CPU usage on the X axis, % swap usage on the Y axis and sectors/second (disk usage) on the Z axis. The Dev8-0 usage is quite light as shown in these figures, less than 3000 sectors per second while

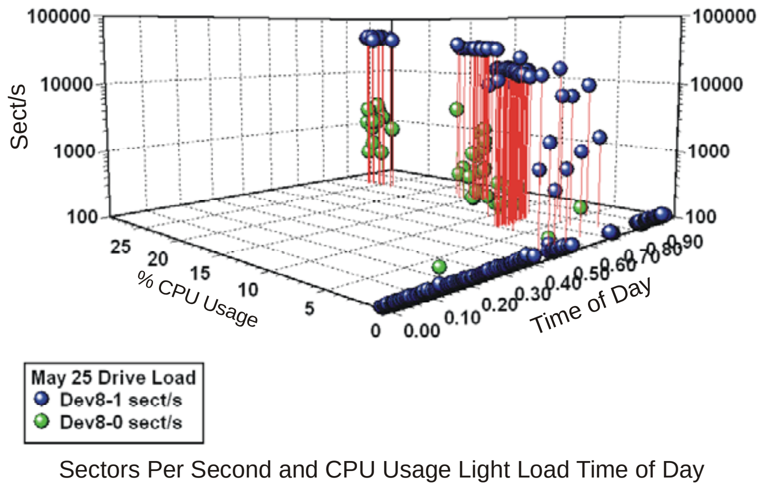
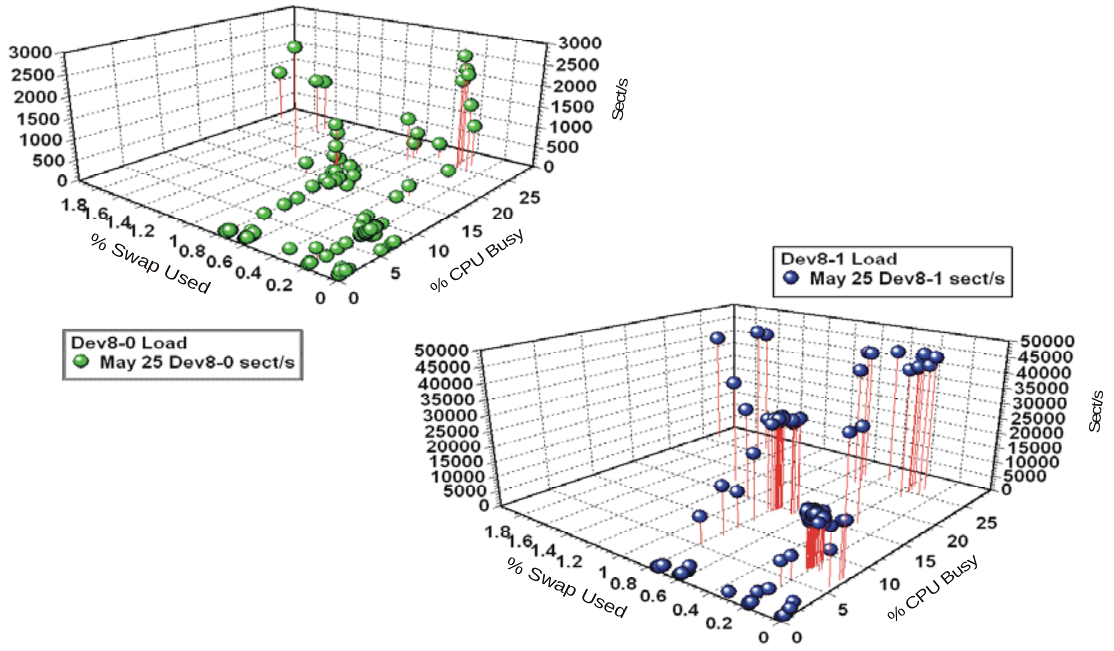


Figure 2

Dev8-1 approaches 50000 sectors per second at the highest CPU usage rates seen in this light load run. Since this run involved processing a relatively small amount of data, most of it could be processed in memory and so demand for swap space was limited. The swap usage in this light load run was less than 2% as shown in Figure 3. The results show that the CPU is not very busy, no more than 30%, and the access to data on Dev8-1 still reached quite high rates of above 45000 sectors per second at the highest CPU usage rates. If the S4PM-DM system is going to be handling a light load, like the May 25 run, this level of machine would handle it easily. However, even here, the rate of disk access for Dev8-1 is only about a factor of 2 below its highest rates for the heavy load runs described below. The rate of use of Dev8-0 is at least an order of magnitude less than that of Dev8-1 for the light load run, which reflects Dev8-0's low level of use for system house keeping and swap space. However, with a heavier load, this machine's limits become clear. This is shown from the results of the May 27 run.



Comparison of Dev8-0 and Dev8-1 for Light Load

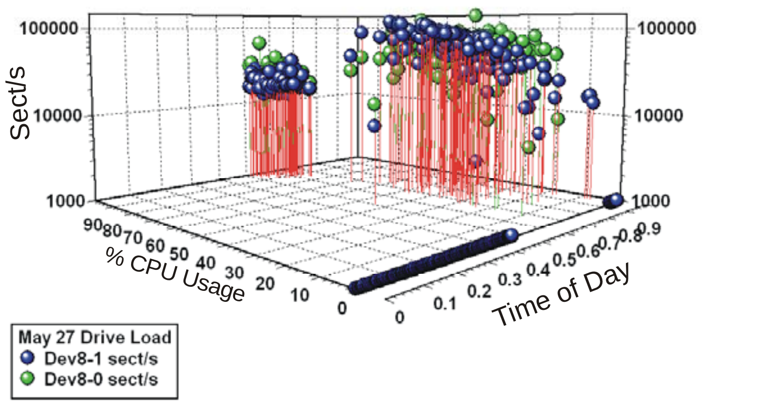
Figure 3

Table 2 gives the timeline for the May 27 run. For this run, six instances of each algorithm were initiated in quick succession. The two time stamps for the end of the MOD06_L2 run indicates the interval between when the first instance of the algorithm ended and when the last, sixth, instance of the algorithm ended. The results, Figure 4, show the CPU's approached 100

Run Type	**Algorithm**	**Hour of Day**		**Fraction of Day**	
		Start	**End**	**Start**	**End**
Comb 6 runs	MOD06_L2	12:01:34	22:01:12-22:53:33	0.501088	0.917500-0.953854
Comb 6 runs	MOD11_L2	12:09:37	15:45:21	0.506678	0.656493
Comb 6 runs	MOD05_L2	12:01:27	13:41:33	0.501007	0.570521

Table 2, May 27, 2005 Runs

percent usage and the disk performance actually dropped at the highest CPU usage rates. The cluster of points to the left in Figure 4 illustrates this. Both Dev8-0 and Dev8-1 saw very high usage rates of around 100000 sectors per second. At the high usage rates created on May 27, limitations in memory, CPU and disk speed were both observed. This indicates that for this particular scenario, the prototype machine was relatively balanced. Slightly faster CPU's might have allowed the highest disk read/write rate to be maintained at 100000 sectors per second, however, more memory would have decreased



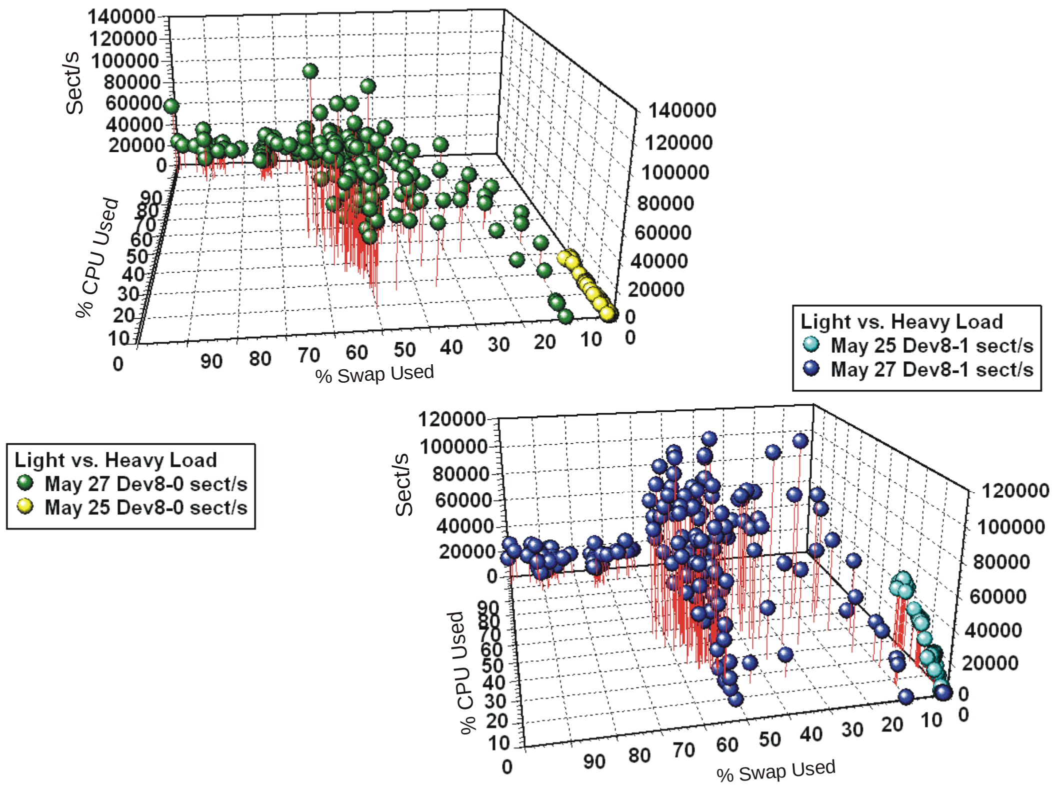
Sectors Per Second and CPU Usage Heavy Load Time of Day

Figure 4

the swap space demand and the CPU's would have spent less time moving pages in and out of disk swap. Also, providing more swap space would have improved performance. This can be seen in Figure 5, where there is a big drop off in disk performance once swap usage exceeds 50 percent.

Figure 5 provides a comparison of the low load and high load runs. The graph on the left gives the loading for Dev8-0 as a function of % CPU and % swap used while the graph on the right gives the loading for Dev8-1 on the same base. The light loads are yellow in the left graph and cyan points in the right graph, and the green and blue are the heavy loads left and right graphs, respectively. This confirms that for the light load, almost no swap was used, and little of Dev8-0's capacity was used. For the heavy load, swap usage and CPU usage increased together, while Dev8-0 usage reached a peak at about 70% swap usage and then declined as the CPU's became saturated and the swap space became exhausted.

Dev8-1, whose activity reflects data access, increases with CPU activity for both the light and heavy loads. As described earlier, Dev8-1 activity falls off rapidly with over 70% swap space used and greater than 80% CPU utilization.



Comparison of Dev8-0 and Dev8-1 for Light and Heavy Loads
Figure 5

Recommendations

Based on these benchmark runs, our prototype, an Intel-based hardware system running Linux OS hosting S4PM-DM would easily meet the requirements for a system with a load equivalent to our light load runs. But a system experiencing a load equivalent to our heavy load would require an increase in the amount of memory from 4 to 8 Gigabytes or more, with a corresponding increase in the amount of swap space on the hard disk system. In general, these three algorithms appear to be I/O intensive, so a fast disk system and fast I/O bus would be requirements for configuring a balanced system. Newer Intel based systems now have 800MHz buses compared with the 400 MHz bus in our prototype system. This increase in bus speed should provide an observable increase in performance for both a lightly loaded and a heavily loaded system. Of course, the fastest network connection available at the hosting facility would be a requirement, so that data could be quickly moved from the archive staging disks to the processing machine.

References

Lynnes, Christopher, Long Pham, Jon Robinson, Evelyn Nakamura, 2004, *S4PM – Data Mining Edition Implementation Plan* (versions 0.3 and 0.4). pp 18.

Raytheon, 2004, Phase II Technical Proposal for Synergy FY04