# SSC07-XII-5

# A Database Centered Approach to Satellite Engineering Data Storage, Access, and Display

Patrick Cruce University of California at Berkeley Space Sciences Laboratory 7 Gauss Way Rm 133A Berkeley, CA 94720-7450; 510-710-7745 pcruce@ssl.berkeley.edu

Bryce Roberts University of California at Berkeley Space Sciences Laboratory 7 Gauss Way Rm 238A Berkeley, CA 94720-7450; 510-643-4077 <u>broberts@ssl.berkeley.edu</u>

Manfred Bester University of California at Berkeley Space Sciences Laboratory 7 Gauss Way Rm 240A Berkeley, CA 94720-7450; 510-643-1014 <u>mbester@ssl.berkeley.edu</u>

Timothy Quinn University of California at Berkeley Space Sciences Laboratory 7 Gauss Way Rm 232A Berkeley, CA 94720-7450; 643-8379 <u>teq@ssl.berkeley.edu</u>

#### ABSTRACT

Many of the tools available today for processing and displaying engineering telemetry data of small satellite missions are limited by a real-time software design. Data access is often accomplished via sequential (linear) search, and interfaces are constrained to snap screen shots or batch processing views. To promote greater accessibility and improve data retention, we have developed the Berkeley Trending and Plotting System (BTAPS). BTAPS is a general purpose software suite that provides a record of all spacecraft engineering data over the course of the mission via a MYSQL 5.0 database and includes an interface to generate multiple simultaneous time ordered plots of mission engineering data. BTAPS can simultaneously decommutate, convert and store data from multiple satellites into the database in real time and load back-orbit data without impacting real-time operations. The system is compatible with single satellite or constellation class missions that use CCSDS compliant file formats. BTAPS was a crucial part of integration and testing for the recently launched multi-spacecraft THEMIS mission, and is central to data storage, on-orbit operations and mission analysis support. By making a database central to mission operations, we are able to provide random access to the satellite data in  $O(\log(n))$  search time, rather than being restricted to real-time and replay access to the data or O(n) search time. BTAPS also leverages the rich set of built-in interfaces that are part of MYSQL, simplifying the development of new client programs and decreasing language specific constraints on client design. BTAPS greatly increases the availability and efficiency of access to satellite engineering data, and proved to be an invaluable tool for mission support from pre-launch testing to essentially all aspects of on-orbit operations of the THEMIS mission.

# INTRODUCTION

The recent exponential progress in computer hardware makes possible significant improvements in the systems used to acquire, archive and visualize spacecraft telemetry, while the richness of freely available, open source software tools greatly reduces the software and support costs for rapid, low-cost development of such software systems. Additionally, NASA's Time History of Events and Macroscale Interaction during Substorms (THEMIS) mission more than doubled the size of the fleet of satellites operated by the Mission Operations and Ground Systems Group at Space Sciences Laboratory (SSL) at the University of California at Berkeley (UCB).<sup>1,2</sup> During the design phase of the THEMIS ground system, it became clear that sophisticated tools would be needed to be able to operate the larger number of spacecraft efficiently. Taking advantage of new technologies, the Berkeley Trending and Plotting System (BTAPS) is one of the tools that was developed to support telemetry data processing, archiving and analysis.

BTAPS ingests, decommutates, and converts raw spacecraft telemetry from multiple sources and stores the converted engineering values for random access. BTAPS was built from the ground up with the open-source MySQL database as the central component for archiving and distributing spacecraft housekeeping data.<sup>3</sup> Real-time and post-pass decommutation libraries and real-time and batch plotting software components were designed, developed, and tested at SSL. BTAPS operates in tandem with the spacecraft command and control system to play a mission critical role in evaluating the performance, health and safety of multiple THEMIS satellites simultaneously.

# THEMIS BACKGROUND

The THEMIS mission, launched on February 17, 2007, is a constellation-class space science mission consisting of five satellites in highly elliptical, low inclination orbits to study magnetospheric physics related to the generation of the aurora.<sup>1</sup> THEMIS telemetry data are received at a number of ground stations located in different continents. Upon reception, transfer frames are relayed to the THEMIS Mission Operations Center (MOC) at SSL. Also located at SSL is the Berkeley Ground Station, which serves as the primary ground station to support THEMIS.<sup>2</sup>

Much like many other recent missions, THEMIS follows the Consultative Committee for Space Data Systems (CCSDS) standards for telemetry transfer frame and source packet generation.<sup>4</sup> Different Virtual Channels (VCs) are used to route and split various

types of telemetry data from the spacecraft to respective ground processing systems. Real-time spacecraft engineering data are transmitted on Virtual Channel 0 (VC0) at a relatively low data rate of ~4 kB/sec. These real-time packets are routed directly to the MOC via real-time network socket interfaces.5 Playback spacecraft engineering telemetry data are transmitted to the ground on Virtual Channel 1 (VC1) with data rates up to approximately 1 Mbit/sec, and are captured and stored in files at the ground station. These data files are delivered to the MOC post-pass via file transfer (FTP) across the Open Internet, and are subsequently processed at the MOC.

Each THEMIS telemetry transfer frame contains one or more telemetry source packets. In accordance with CCSDS specifications, each packet is labeled with a number called an Application Process Identifier (APID) that is used to logically separate telemetry data by origin on the spacecraft side. Each telemetry source packet contains a header with a time stamp and a number of physical measurements or digital status information. Each separate measurement is associated with a string identifier known as a mnemonic. For the design and implementation of the database it is important to note that each source packet generated for a given APID is guaranteed to have a unique time stamp which applies to each and every mnemonic derived from that APID.

The MOC employs the Integrated Test and Operations System (ITOS) for all real-time spacecraft command and control functions.<sup>2,6</sup> However, since ITOS does not support either real-time or playback trending functions at a level that is required to efficiently support multi-spacecraft operations, another tool needed to be either developed or procured. These circumstances inspired the creation of BTAPS as an in-house development project.

# **BTAPS DESIGN PRINCIPLES**

The design of BTAPS was guided by several general principles. First, all data stored in the database were assumed to be time stamped sequentially. This allowed very efficient access to data by indexing all time stamp columns.<sup>7</sup> It also simplified the design of a plotting interface since the development effort could be focused on providing only time ordered plots. If necessary, it is possible for two mnemonics to be plotted relative to each other within a specified time interval, but this was not the BTAPS design focus. In practice, most views of engineering data have time on the horizontal axis, so this assumption did not constrain data access after launch.

During the mission integration and testing phase, however, the spacecraft clocks are not always set to Universal Time Coordinated (UTC). Instead, the clocks may be running on Mission Elapsed Time (MET) and may repeatedly generate data with the same time stamps each time a spacecraft is powered up for testing. The resulting problem to store data tagged with nonunique time stamps was minimized by using separate logical databases for separate tests. However, the Mission Operations Team decided that the advantages in efficiency and organization made it worthwhile to maintain the general design principle of assuming sequential time tagging.

During THEMIS mission operations, simultaneous pass supports with multiple spacecraft are common. Hence, real-time and post-pass telemetry data may be arriving from as many as five satellites in parallel. To efficiently handle processing of these incoming data streams, a high degree of automation was designed into BTAPS to support database operations with minimal user interaction, thus eliminating the need for tedious and repetitive manual operations for both the process of data loading and database and decommutator configuration. This approach also allowed for the systematic removal of data processing errors through the software debugging process.

#### Database Configuration

The database schema for a given spacecraft is automatically generated by the BTAPS configuration utility from a concise, human readable description of the telemetry format. This eliminates the need for manual creation and maintenance of complex SQL code, and it leverages existing telemetry definitions used by other ground systems elements such as ITOS.

All information required to extract, load, convert and store the THEMIS engineering data is stored in the database itself. In fact, there are many advantages to database use in satellite operations. A database provides a centralized repository for all information about satellite engineering telemetry. Despite a single physical location, this repository can be made accessible from anywhere. Also, a commercial quality database provides access to optimal implementations of many functions that would otherwise be very difficult Compression, logarithmic search, and an to write. abstraction from physical representation are automatically maintained, preventing problems with file management and data access that can be a major source of difficulty in software development for satellite missions.

More specifically, MySQL was selected for the BTAPS project because it is free, open source, and has excellent on-line documentation. Due to its extensive testing in the commercial sector, MySQL is considered very reliable and scales to data volumes many orders of magnitude larger than those expected for the THEMIS mission.

Finally, converted data are stored in the database along with the raw packets. Advantages accrued are increased speed of data access and decreased reliance on programmatic libraries to access converted data. This approach makes it much easier to build front ends to access data in the database. However, two distinct disadvantages are slowing the process of loading data into the database and are rendering the database less flexible if there is need to modify a conversion or the definition of a source packet during the mission. For example, if a value in the BTAPS database has a modification in the way it is converted to physical units, it will be necessary to update the conversion data in the database and either reprocess old data or leave old data with the old conversion. For the THEMIS mission, few changes are anticipated to the data format and conversions of THEMIS data and the data retrieval processes are highly automated, so it was decided that conversion prior to loading was desirable for THEMIS.

#### SOFTWARE IMPLEMENTATION

BTAPS consists of five major components: a MySQL version 5.0 database,<sup>3</sup> a decommutation and database configuration utility, a C based Application Program Interface (API) for loading data into the database, a Graphical User Interface (GUI) for accessing and plotting the data, and several other interfaces for command line and network socket access to the



database. All components with the exception of the MySQL database were developed and tested on SunOS 5.9.<sup>8</sup> The database engine used is MyISAM, which is based on the Indexed Sequential Access Method (ISAM). An overview of the software implementation is shown in Figure 1.

The data are made available by a server that runs the database software at all times. A separate database is used for each of the five THEMIS satellites. All databases are hosted by the same MySQL server. The server maintains logical separation without requiring multiple database servers. Each database holds a series of tables called "meta data tables" that describe the data and their telemetry formats. These meta data tables include data types, conversion tables, high/low limits, and byte offsets in the telemetry source packet. Since the conversion and limit meta information can vary in structure and dimension, separate tables store each type of conversion and limit meta data.

### Database Access

Client software interacts with the MySQL database through a read-only or read/write-enabled account to help protect data integrity. The read/write account is used by the data loading clients, but can only be accessed from a strictly controlled list of known host addresses. The read-only account cannot modify BTAPS data but is more widely available. This dualaccount configuration permits deployment of BTAPS data access clients at remote sites without compromising data security.

# Database Tables

The databases also contain tables to store the satellite data. The organization of these tables is driven by the structure of the CCSDS telemetry format. BTAPS maintains one table to store packets of unconverted data from each APID and one table to store packets of converted data from each APID. BTAPS data tables are optimized around the guarantee that time stamps within each APID will be unique. Organizing the data by APID allows BTAPS to minimize and negotiate any collisions of data that are collected from different sources or at different cadencies. Time stamp columns within the data tables are also indexed to optimize time series queries on the satellite telemetry data. While the physical representation of an indexed database column is opaque to the database user, indexed columns may be searched in logarithmic time. This is a major improvement over linear time search.

### Database Configuration

Due to the complexity of the telemetry format definition, a configuration utility, rather than a human author, builds the BTAPS database schema. Resource files provide all the information used to create and populate tables in the database that describe the telemetry data, as well as create tables for storing telemetry data itself. The automatic schema generation is highly mission independent, and can easily support telemetry form any satellite with CCSDS-compliant telemetry formats. Thus the database structure is determined almost entirely by the content of the configuration files that describe the satellite telemetry stream.

For configuration, BTAPS needs one file that describes the location of packets within the telemetry stream plus a series of database exchange (DBX) records. The DBX records describe the packet telemetry formats for the purpose of decommutation. Since DBX records were already used by ITOS, the same file format was adopted for the BTAPS utility to maintain compatibility and to allow for reuse of already available configuration files. It also allowed BTAPS to be integrated easily into the overall architecture of the Mission Operations Center.

BTAPS converts raw telemetry to physical values using seventh-order polynomials, discrete number-to-string conversions, and abstract mathematical expressions. At the time of database creation, each mathematical expression generates a raw-to-physical conversion table which is stored as meta data; conversion then simply requires a single, constant-time table look-up, eliminating the need to repeatedly reevaluate a complex mathematical function.

To manage the mathematical expression based conversions, the table generation utility interprets the expressions using a simple open source Scheme interpreter called Guile.<sup>9</sup> Using Guile eliminated the difficulty of writing an interpreter for our project. A built-in interpreter also eases the extension of the configuration tool to new file formats by allowing conversion information to be written in a relatively high-level language instead of a low-level language like C.

# Database Loading

When data are acquired from the THEMIS satellites, the programs that receive the data call a load function in the BTAPS C-API. The load function reads copies of BTAPS meta data tables from the MySQL database into memory and uses these tables to determine how to decommutate and convert the data from the telemetry stream. Consequently, it is configured entirely by the content of the database. These converted data are then inserted into the appropriate tables in the database. Collisions between any duplicate inserts are negotiated automatically by setting the database insert code to ignore all collisions. Thus originally inserted data maintain higher priority.

#### Data Retrieval

Access to the data is provided primarily via a GUI plotting client known as *btapsplot*, which can present multiple, simultaneous data views of both archival and real-time data. Alternatively, converted data values can be dumped to comma separated variable ASCII files by a command-line tool called *btaps seqprt*. Since a

database system provides an abstract and unsegmented view of all telemetry points over the course of the mission, the time span over which plots are constructed represents no obstacle to plotting. An example of the GUI is shown in Figure 2.

Plots are generated using an efficient algorithm for walking over time series points. Points are represented by the plotting program internally by a binary tree. Each node in the tree represents a single data point at a given time and points are ordered in the tree according to their time relations. The program then plots approximately one thousand children of a node representing the viewpoint. No more than one thousand points are ever needed as screen resolution prevents presentation of more than one thousand pixels of plot width. Using a binary tree allows easy zooming in and



out as the zoom function is only a walk up or down the nodes of the tree. The tree itself need only be filled in lazily. This means additional points are only added to the tree if a request to zoom is made. The plot represents a sparse sample of the data points actually stored in the database, but the sparse plot is indistinguishable from a plot using all the points insofar as the human eye is concerned.

BTAPS also includes a lightweight socket based interface. This allows users to read data from the database in ASCII format over a TCP/IP network socket connection. BTAPS reads a series of simple commands and then prints the requested data to the client. Login can be done via a telnet client or programmatically. This allows direct access to the data for the purpose of direct export and provides backwards compatibility. The socket-based client is also used to protect the database from possible corruption due to direct modification via human interaction with the MySQL client.

# HARDWARE AND OPERATING SYSTEMS

The configuration and load software was carefully tested and runs on SunOS 5.9 on a SunBlade 1500.<sup>8</sup> The software was designed with Linux compatibility in mind, but has not yet been tested under Linux. The database server itself runs under x86\_64 Linux on a rack mounted computer with a dual core 2.4 GHz AMD Opteron 280 processor and 2 GB of RAM. Data are stored on a RAID array composed of four 500 GB hard drives in a 1+0 configuration, allowing increased throughput and fault tolerance. RAID 1+0 allows the loss of one drive without any data loss. It may allow the loss of a second drive if the second drive is in a different mirror than the mirror of the first lost lost drive. The RAID 1+0 configuration allows the database to store as much as 1 TB of unique satellite data.

The plotting software is designed using the open source Qt library, so the plotting software is expected to be fully cross platform compatible.<sup>10</sup> It has been tested so far on Mac OS X, Linux, and SunOS.

The lightweight socket interface is written in Qt and has been tested on Linux and SunOS. Currently it runs on the same server as the BTAPS MySQL database.

# COSTS

The hardware costs are of the order of \$3900 in 2006. Initially a rather basic Linux PC was used to run the database server and to develop the software. Power and bandwidth costs have not been significant in proportion to those already allocated for operating the MOC. The

development of BTAPS was accomplished by a part time programmer for one year and a full time programmer for three months.

### **RESULTS AND PERFORMANCE**

BTAPS has been running since the mission integration and test period for the THEMIS spacecraft in December of 2006, and continues to run with excellent performance in support of THEMIS on-orbit operations. During environmental testing, data were streamed in real-time via TCP/IP sockets from the five satellites at NASA's Jet Propulsion Laboratory (JPL) in Pasadena into the database server at the MOC at SSL. Engineers and scientists located at both JPL and at SSL were able to retrieve and view the data in near real-time, using the plotting interface.

Currently, BTAPS is used to provide real-time and historical plots of satellite performance. Overall, the system has been extremely reliable and stable. As an example, the system has been able to support more than 20 simultaneous clients, each displaying many mnemonics in real-time, while simultaneously loading real-time data during ongoing maneuver operations. Even engineers located at remote facilities did not experience problems with data access: the speed of data acquisition is limited only by the available network bandwidth. Additionally, the efficient sampling algorithm mentioned above proved to be very effective for generation of plots over long time spans.

# Load Benchmarks

A series of benchmark tests were performed to quantitatively estimate the efficiency of database read and write operations. To determine how compactly the data are written, statistics were collected from the loading of five different telemetry files. It was found that the stored data volume was  $9.27 \pm 4.41$  (mean  $\pm$  std. dev.) times larger when stored in the database than when stored as direct copies of satellite telemetry. The high standard deviation of the compactness ratio indicates that the compactness of database storage varies highly with the content of the telemetry files.

The same file loads were timed to determine load rates. The average load rate for each file was  $14.81 \pm 0.71$  kB/sec. The timing statistics also indicated that only 35% of the time spent loading was actually spent in the loading software, indicating that the majority of the time required for loading is used by the MySQL server.

An average over five random files allowed an estimation of data download rates from the satellites for comparison to the database load rates. Real-time data

are generated by the spacecraft at a mean rate of  $4.1 \pm 1.02$  kB/sec, while stored data are produced at a mean rate of  $0.072 \pm 0.02$  kB/sec. The ratio of database load rate to satellite collection rate for these files was also calculated to assess the load tolerance of BTAPS. It was found that BTAPS can load data at 3.65 times the rate they are collected in real-time from a single satellite and 212 times the rate they are collected postpass from a single a satellite.

An analysis of data usage over the course of the mission was conducted. By averaging over the amount of data accumulated in the database from launch of the mission on February 17, 2007 until May 31, 2007 it was found that the database accumulated 0.34 GB of data per day for all five satellites combined. This means the 1 TB size limit provides the database with an extrapolated lifespan of 8.19 years until it will fill up. The amount the database can load per day was calculated from the load rate per second. The database can load 1.22 GB/day. On average it is calculated that BTAPS can load data for all five satellites are accumulating on the ground at the highest telemetry rate.

#### Retrieval Benchmarks

Data access times were also analyzed. As expected, data read times increase linearly with the number of sequential rows requested. For example, one sample found that reading 5000

rows took 2.39 seconds, reading 50000 rows took 21.59 seconds, and reading 500000 rows took 212.94 seconds, giving a mean data rate of 2252 rows/sec for sequential rows. This data rate appeared to be invariant with respect to the size of the table.

The speed of non sequential queries was also estimated. A single table was queried using an indexed column to select a random row. This process was iterated and timed. A test performed on the same table as the sequential read test above found that reading 500 random



Figure 3: Indexed Column Access Times

rows took 13 seconds, reading 5000 random rows took 109 seconds and reading 50000 random rows took 676 seconds. This indicates an average random read time of 52.7 rows/second. The time efficiency of the access improved with the number of rows accessed. 500 random rows were retrieved at a rate of 38 rows/sec, 5000 random rows were retrieved at a rate of 49 rows/sec and 50,000 random rows were retrieved at a rate of 49 rows/sec. This suggests there is significant caching of query results and table indexes.

To test the scaling of BTAPS reads with respect to the size of tables, a series of 100 random queries were performed on each table in the database. These tests were timed and repeated 50 times for each table. The query time results were then averaged. In these queries only a single indexed value was retrieved rather than a whole row to eliminate variability in timing that could arise from retrieving rows of different lengths that would occur across tables. Figure 3 demonstrates a near logarithmic relationship between the number of rows in the database table and the time to retrieve them. The estimated curve of fit was:

$$query\_time = 250 * \ln(rows\_in\_table/10^{5})$$
(1)

Where rows\_in\_table is  $\geq 10^{5}$ . This result corresponds with the theoretical expectation that column indexing will yield asymptotically logarithmic time search performance. The discontinuities in the

data curve are due to the fact that the number of rows in the BTAPS data tables are not evenly distributed, but instead cluster around several values.

### CASE STUDIES

The capacity to view real-time plots, rather than individual telemetry values, allows the perception of trends that are exceptionally difficult to perceive without real-time plots. Long-term plots expose gradual trends that could easily go unnoticed. Presenting *both* capabilities in one software tool allows an instantaneous anomaly to be rapidly correlated with past performance while still during a real-time pass support.

Figure 2 shows a plot set that is updated and monitored in real-time during spin-up maneuvers, in this case for THEMIS D. The top plot shows the spacecraft spin rate, the middle plot shows the fuel tank pressure pressure, and the bottom plot shows the temperature of two of the spacecraft thrusters. The curve in blue represents the temperature of the firing thruster and the curve in red represents the temperature of the non-firing thruster. All plots have the same time scale on the x axis.

The plots correlate the spin rate increase and the fuel tank pressure decrease as the maneuver progresses. The temperature of the non-firing thruster increases, but because cooled fuel is flowing into the firing thruster the temperature increase on the firing thruster is dampened. At the end of the maneuver the firing thruster temperature increases due to "soak back" from the hot thruster. These time ordered correlations are commonly used to assess the efficacy of the maneuver or diagnose a thruster problem before it becomes urgent.

An example of a long-term trend on THEMIS A can be seen in Figure 4. The top panel shows a curve representing the spin rate of THEMIS A, while the center and bottom plots each show two superimposed curves representing the temperatures of fuel tanks 1 and 2, respectively. The blue curves represent the liquid fuel temperatures and the red curves the temperatures of the pressurant gas. All panels are plotted on the same time axis. The long term trend shows that when the heater is activated and the tank contents heat up, the spin rate increases as the center of mass moves towards the center of the spacecraft. The month-long plots are useful for predicting tank heater activations prior to maneuvers, which are sensitive to fuel temperature and pressure. Detailed long-term analyses of telemetry correlations and the associated models of spacecraft behavior are facilitated by the plotting features provided by BTAPS. It is expected that as the THEMIS mission become more mature, BTAPS will be used to assess the longterm performance all five satellites. A typical example is the long-term performance analysis of the solar panels.

### DISCUSSION

While BTAPS easily handles the demands of THEMIS mission operations, several issues in the performance data deserve greater analysis. The first is the 9.27 expansion ratio between raw telemetry and database storage. Although certain parts of the raw telemetry, such as error correction blocks, are thrown away during decommutation and conversion, both raw and converted data are stored. This gives a minimum compactness ratio of 2. Fields that are only a few bits wide in raw telemetry may need to be converted to engineering values and stored as 32- or 64-bit real numbers. Finally, indexes and log files also contribute to the data expansion. In light of this, a nine-fold increase in data size is quite modest.

Extrapolation of the data storage rates predict that the amount of time required to fill the existing disk storage is significantly longer than the anticipated mission duration, so data storage is not a major constraint. If storage space limits are reached, MySQL offers database level compression, which could easily be activated, albeit at the cost of some lost performance.

The existing BTAPS design should scale well to much larger data volumes, as long as the server file system supports it. Though not presently used by BTAPS, the newer Solaris Zettabyte File System (ZFS)<sup>11</sup> allows compression on the level of the file system, which eliminates any slack space at the end of disk blocks that might decrease compression efficiency. ZFS also permits vertical scaling of the file system, thus making it trivial for the file system to grow along with increasing database size.

The load rate of 14.8 kB/sec is well in excess of the requirements for the THEMIS mission. The only way the system could conceivably fall behind is if telemetry data were being received and processed from four of the THEMIS satellites at the same time. The original implementation of the load API focused on simplicity and reliability, and little priority was placed on speed optimization, so further refinements of the load API are likely to increase the effective load rate significantly.



Figure 4: Long-term Plot of Spacecraft Spin Rate and Fuel Tank Temperatures

Nevertheless, measurements show that approximately 35% of the processor time during the load operation is used by the load client, while the rest is used by MySQL as it parses, stores and indexes the data it receives. Thus, the cheapest way to improve speed performance would be to purchase hardware that can improve MySQL server and operating system performance.

The load API currently loads each time stamped row of telemetry data into MySQL as an individual SQL statement. Loading multiple rows in a single statement and making use of table read/write locking could potentially increase data write performance, at the cost of increased read latency.

The decreasing query times for increasing numbers of index queries illustrates database and operating system optimization as a result of index and query caching in memory. This result is interesting because it is similar to the types of serial indexed queries that would be performed using the sparse sampling plotting algorithm. While it is expected that this performance increase will be bounded, these data do not identify an upper bound for this increase in performance.

The data also suggest that users of BTAPS can expect a logarithmic increase in search times as the data grow linearly. This indicates that BTAPS will have only a minimal degradation of query performance even as the data grow very large. More analyses as the mission continues can better remove some of the discontinuities in the search time graph to better understand the relationship between the number of rows in the BTAPS database and performance.

The difference in read times between bulk queries and indexed queries suggests that it may not be useful always to apply the sparse sampling techniques for plotting discussed in the implementation section. It will be more time efficient to use sparse sampling only when:

$$n \ge p * (rs/rr)$$
 (2)

In this equation, n is the number of points in the entire time frame requested, p is the number of points required to create a human readable plot, rs is the rate of sequential reads and rr is the rate of random reads. In the case of the BTAPS efficiency measurements, it would only be time efficient to sparsely sample when the time span requested contains greater than ~46,000 data points. In cases where  $n \le 46,000$  the database will spend time doing excess tree lookups that will degrade performance from the optimum.

Currently, a less precise rule is applied where sequential reads are used for the short term plotting done by the real-time interface and a sparse lookup is used for the non-real-time historical trends.

The real-time trend approach to satellite operations is an item of interest. It contrasts strongly with the value and limits approach often used by command and control software. Value and limits systems show controllers the current values for different telemetry mnemonics and generate messages that indicate when the values are greater or less than a preset limit. This system is inherently reactive. If a value is approaching a limit, the flight controller on console may not notice the trend prior to the actual limit violation.

The real-time trend approach allows the user to quickly make a qualitative estimation of a mnemonic's rate of change and points of inflection. This allows the controller to make a predictive assessment of the situation. It is hoped that this approach will better allow flight controllers to address problems before they become urgent.

A final issue to consider is the possibility of a BTAPS extension. BTAPS was designed and tested with the needs of SSL and THEMIS in mind, but care was taken to make the fundamental design as mission-generic as possible. It has been successfully tested with telemetry files from another CCSDS compatible NASA mission, the Reuven Ramaty High Energy Solar Spectroscopic Imager (RHESSI).<sup>2</sup> The abstraction is in place so that adding compatibility with other mission formats and operating systems to the BTAPS load software would require minimum modification to the current stable software release. The Qt framework that the plotting and access utilities use should already provide Windows compatibility, although these have not been tested on this platform. Quick and simple access to the data significantly decreases the development cost of satellite software tools that would have been too costly to develop without a database infrastructure. Random access web interfaces, satellite simulators, and predictive limit checkers are also being considered.

# CONCLUSION

BTAPS provides an effective automated system for engineering data archival and storage. It stores data in a network accessible, secure central archive with optimized search capabilities. Software interfaces provide functions for efficient plotting and data export. The system meets its theoretical expectations and exceeds all predicted mission constraints. It features the required flexibility to provide access to all spacecraft data in the THEMIS constellation to users in a variety of different software environments. BTAPS allows users to observe time series plots in real-time in a way that significantly increases user awareness of the satellite state and expected performance. BTAPS uses a combination of efficient data search and efficient plotting to create historical plots covering time scales as long as the duration of the entire mission. Overall, the Berkeley Trending and Plotting System has proven to be an extremely valuable tool to support all aspects of THEMIS mission operations.

# Acknowledgments

The authors wish to thank Dr. Vassilis Angelopoulos, THEMIS Principal Investigator, for supporting the BTAPS development, and Jonathan Loran, the IT Manager supporting the Mission Operations Group, for acquiring, configuring and maintaining the BTAPS server. The THEMIS mission is operated by the University of California at Berkeley under NASA contract NAS5-02099.

# References

- Angelopoulos, V. et al., "Time History of Events and Macroscale Interactions during Substorms (THEMIS) – A NASA Medium-class Explorer Mission Phase A Concept Study Report," University of California, Berkeley, CA, October 2002.
- Bester, M., Lewis, M., Quinn, T. and Rauch-Leiba, J., "Automations of Operations and Ground Systems at U.C. Berkeley," Proceedings of the 5<sup>th</sup> International Symposium on Reducing the Cost of Spacecraft Ground Systems and Operations (RCSGSO), Pasadena, CA, July 2003.
- 3. MySQL Open Source Database, MySQL AB, Cupertino, CA, 2007, <u>www.mysql.com</u>.
- Recommended Standards (Blue Books), Consultative Committee for Space Data Systems, Reston, VA, 2007, <u>www.ccsds.org</u>.

- Bester, M. and Stroozas, B., "Telemetry and Command Frame Routing in a Multi-mission Environment", Proceeding of the International Telemetry Conference, Las Vegas, NV, 2007.
- 6. Integrated Test and Operations System, the Hammers Company, Greenbelt, MD, 2006, <u>www.hammers.com</u>.
- Ramakrishnan, R. and Gehrke, J., Database Management Systems, McGraw-Hill, New York, 2003.
- 8. SunOS 5.9 and 5.10 Operating Systems, Sun Microsystems, Inc., 2006, <u>www.sun.com</u>.
- 9. Guile Scheme Programming Language Interpreter, 2007, <u>www.gnu.org</u>.
- 10. Qt GUI Application Framework, Trolltech, 2007, <u>www.trolltech.com</u>.
- 11. Solaris Zettabyte File System (ZFS), Sun Microsystems, Inc., 2004, <u>www.sun.com</u>.