An open source long term archiving and trending solution for SSA

Ryan Melton
Ball Aerospace & Technologies Corp.

1. ABSTRACT

SSA analysis requires trending and analyzing an object's location over a long period of time. Typically, this capability has been coded from scratch when creating a ground software solution which is both time consuming and costly. The COSMOS C2 system has created DART (Data Archival Retrieval and Trending) which enables long term trending in an open source solution. Since being open sourced over four years ago, the COSMOS C2 system has eased the burden on ground system operators. Its emphasis on simple text file configuration and ease of use has enabled dramatic cost and time savings across the industry. With the addition of DART, operators have an open source solution which provides quick answers to questions like how an object moved over the past year. DART utilizes a SQL database and maintains the original COSMOS telemetry files while providing instant access to decommutated data over long time periods. DART synchronizes with the COSMOS server and stays up to date even as telemetry definitions change and data points are added and removed. This paper discusses the need and use cases for a data archiving and trending solution in SSA. It also outlines the architectural and implementation details which went into the creation of the DART tool.

2. DATA ARCHIVING AND TRENDING FOR SSA

"Life can only be understood backwards; but it must be lived forwards."

— Søren Kierkegaard

Predicting the future requires understanding the past. For Space Situational Awareness missions, storing and keeping track of satellite telemetry is a critical aspect of managing a successful mission.

Saving data is relatively easy because it only requires buying or provisioning storage space. The difficulty is making that data quick to search and easy to interpret. COSMOS DART is an open source solution to that problem.

3. USE CASES

Being able to quickly query archived data can help in answering questions about other objects in space, and about your own satellites. This section discusses these potential use cases.

Answer Questions About Any Object on Demand
For any generic object, you may only have access to its historical position. Even with this limited knowledge, there are still many questions that can be answered.

What is the largest orbital maneuver an object has ever performed?
DART automatically performs data reduction which can enable very quick responses to questions like this over large periods of time (years). By storing a telemetry point that captured the time and size of maneuvers for each object, you could quickly answer this question.

How many times has an object passed over a location in the last week?
By keeping the history of an object's position, you can always go back and see if there were opportunities for data collection or telemetry downlink at any given time.

How much fuel has an object potentially used (or have remaining) given its historical delta-v?
Using maneuver history, you can estimate remaining propellant for an object (given an initial idea of starting propellant).

What objects were near each other at a given time in history?
Quickly querying the position of a large list of objects at a given time, makes it easy to perform differencing calculations and determine which objects were near each other.

Provide Detailed Analysis of Your Own Satellites
For your own satellites, you have access to a lot more data than simply position. Storing all of your satellite's telemetry into DART offers numerous potential use cases.

Predict Lifetime Based Failures
Over time, different parts of satellites degrade due to the harsh environment of space. Whether due to thermal cycling from moving into and out of eclipse, gradual radiation damage to electronics, or other causes, trending historical data can provide predictions to when these events may occur.

Predict Differences from Normal
By querying historical telemetry values and applying machine learning techniques, computers can be taught what is normal for a set of telemetry, even when values normally vary widely, or in relation to orbital position. This definition of normal can then be used to detect anomalies.

Analyze Anomalies
When something goes wrong, you can never have enough data. DART allows quick retrieval of the events that led up to an anomaly that can help your data analysts determine what went wrong. You can also look back in history to see if the anomaly has ever occurred before (or came close to occurring and you didn't notice).

Improve analytical models
Thermal and power models are never perfect at the launch of an orbital asset. By having continual access to on-orbit data, you can refine your models over time and react to actual changes in the behavior of the satellite due to aging effects such as the reduced performance of a battery.

4. DART ARCHITECTURE

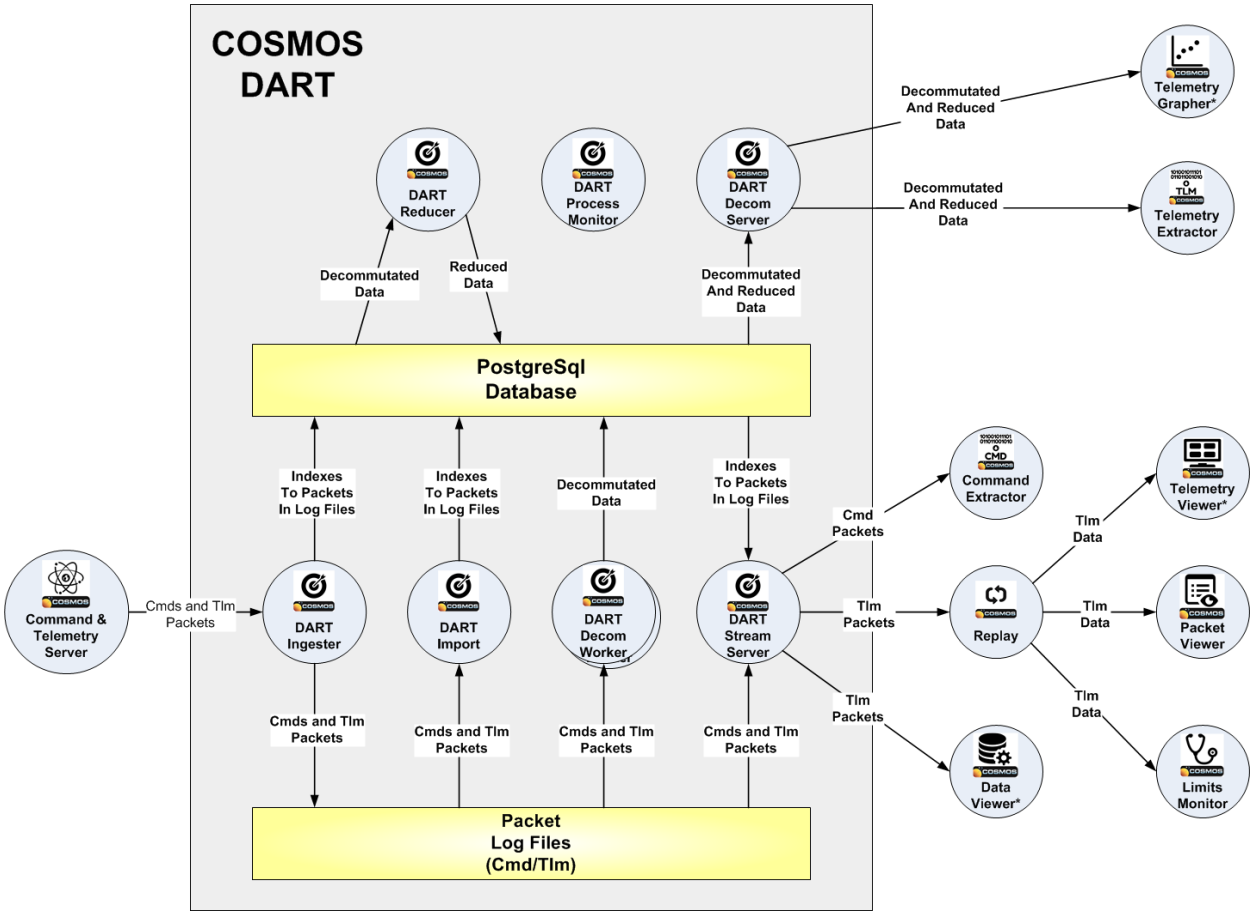

Fig. 1. DART Architecture

DART is composed of seven internal processes depicted as circles within the colored square in Fig. 1 above. Whenever DART is running, all internal applications are also running except for DART Import which is only used to manually ingest log files in off-nominal scenarios (and potentially also DART Ingester, if the user's architecture does not intend to stream realtime data into DART). Each process is described in the following paragraphs.

DART Process Monitor
This is the "main" DART process that starts all the other processes that make up the complete DART system. When the DART Process Monitor first starts up it performs an optimized algorithm to clean the database from any inconsistencies that may have occurred from a previous harsh shutdown. It then starts the next five applications below and monitors their aliveness. If any monitored process unexpectedly dies, then the DART process monitor automatically restarts it. Shutting down the process monitor also properly shuts down the other DART processes.

DART Ingester
The DART Ingester is responsible for logging raw packet data into DART in realtime as it is also being received by the COSMOS CmdTlmServer. At startup, it connects to the COSMOS CmdTlmServer and receives all realtime command and telemetry packets across a TCP/IP stream. As each packet is received, it is logged into a binary log file, and a database item is created that indexes where in the file that packet is stored.

Any data that is captured by the COSMOS CmdTlmServer when the DART Ingester is not running can be added into the database later using the DART Import tool as discussed later.

DART Decom Workers
Each DART Decom Worker process decommutates raw packets by breaking them down into their individual items and values. It then saves the decommutated data into the PostgreSQL database. An environment variable can be used to control how many Decom Workers are spawned (DART_NUM_WORKERS), which currently defaults to 2.

The DART Decom Workers are the most processor intensive part of DART and the number of workers may need to be scaled based on bit rates and the number of telemetry items that are defined.

DART Reducer
The DART Reducer process reduces decommutated data into minute/hour/day reduced data sets. Each item that can be reduced is broken down into a maximum, minimum, average, and standard deviation over the representative period. Only numerical items (integers and floats) are reduced by DART. Due to ordering constraints, the DART Reducer is implemented as a single process with a variable number of threads.

DART Stream Server
The DART Stream Server provides a stream of raw packets from DART on request. Requests typically specify a time range, and optionally can limit the stream to a specific set of packets. The DART Stream Server is used for tools that need complete raw packets of binary data .

DART Decom Server
The DART Decom server provides a JSON formatted array of decommutated or reduced data for an item on request. Again, requests typically specify a time range. Notably the DART Decom Server only returns data for one item at a time. Data for multiple items can be retrieved using multiple queries. The DART Decom Server is used by tools that need quick lookups of specific item values over time.

DART Import
DART Import is a command line tool to import previously logged data into DART. This is useful for importing older data that was never imported such as if DART was offline when the data was collected, or if stored telemetry was not sent in the same stream as realtime telemetry. Important: These log files are imported in place and become part of the DART database. The files must be placed into the DART data folder (defaults to outputs/dart/data) before they can be imported.

5. USER INTERFACES AND COSMOS INTERACTION

The primary User Interface for DART is the existing COSMOS tools that can handle data logged into files. Additional user interfaces can easily be created using the provided DART APIs.

For decommutated and reduced data, COSMOS TlmGrapher and TlmExtractor can pull data from the DART Decom Server. TlmGrapher provides a line graph style display of data, and TlmExtractor can provide that same data as a CSV file.

For streaming raw packets, Replay, DataViewer, and CmdExtractor can pull data from the DART Stream Server. DataViewer provides a scrolling text display of data which is particularly useful for viewing memory dumps and event style data. CmdExtractor pulls the details of when commands are sent and the parameters involved in each command. COSMOS Replay allows the other COSMOS tools that only support realtime telemetry to receive data from DART (as if it was in realtime).

6. DATABASE

PostgreSQL was chosen as the database engine for DART. COSMOS is a fully open source system so choosing an open source database that users could easily install and experiment with on their desktop machines was an important requirement. MySQL and PostgreSQL are the primary contenders for open source SQL databases. PostgreSQL's native support for storing array data and JSON data lead to its choice. Time series databases and object store databases were also considered, but none were found that were of sufficient ease of use, maturity, or with built in indexing capabilities.

Schema Design
The DART schema was designed with two primary requirements: fast queries, and adaptability. DART is designed to be able to provide quick answers to queries even over years of stored data. This is accomplished by a simple time organized schema and automatic data reduction.

Adaptability is equally important. Especially during development, the contents and number of command and telemetry packets in a system evolve. Fields are added and others are deleted. A fully functional data archive must be able to adapt to these changes.

Performance / Size Trades
DART has been optimized to provide quick time-based queries of command and telemetry data. Providing quick queries involves a tradeoff between storage overhead and speed of the query. Database indexes are the natural way to improve the performance of queries in modern SQL database systems such as PostgreSQL. Unfortunately indexes also take up significant disk space and can slow down writes to tables.

During development of DART it was discovered that indexes often reached up to one third of the size of the table they were indexing. Therefore, only indexes absolutely required to ensure fast performance for the normal types of DART queries were created (primarily time).

For specific cases such as reduced_state in the data tables, PostgreSQL partial indexes were utilized to only store the necessary indexes to rows that still needed to be reduced (and ignore rows that are already reduced).

7. API DESIGN

DART provides two different APIs, one to retrieve raw packet data as a stream, and another to request up to 10,000 data points at a time of decommutated or reduced data. These APIs are used directly by the COSMOS tools but may also be used by any other custom software.

Raw Packet API
The raw packet API streams raw packets over a TCP/IP connection. This API allows applications to get entire raw packets just as they were received and logged by the CmdTlmServer. Maintaining the original raw data is a critical feature to ensure that anomalies can always be fully reinvestigated down to the bit level, and to ensure that data was decommutated correctly.

Decommutated Data API
The decommutated data API follows a command / response model and as such uses the same JSON-RPC over HTTP protocol as used by the main COSMOS CmdTlmServer API. See the COSMOS JSON-API documentation for more details. http://cosmosrb.com/docs/json_api/. This API allows applications to pull up to 10,000 samples at a time from a telemetry point at either its original data rate, or in an automatically reduced form (one sample per minute, hour, or day).

8. GETTING STARTED WITH DART

Setting up DART is very simple. If COSMOS is already configured for your project, then all that remains is to create a user and database in Postgres and then run two commands in a terminal.

Detailed installation instructions for installing and setting up DART can be found at:
http://cosmosrb.com/docs/dart_install/

9. SUMMARY AND FUTURE LOOKING

DART provides a high-performance open source solution for managing large amounts of command and telemetry data for SSA programs.

Future versions of DART are planned to offer additional functionality to support mission data storage such as the raw images used to extract SSA information. This ability could allow quick retrieval and analysis (or reanalysis) of mission data from a specific point of time, or in relation to a specific target of interest.

For more information and to get started with Ball Aerospace COSMOS and DART please see http://cosmosrb.com.

10. REFERENCES

[1] Melton, Ryan, *Ball Aerospace COSMOS,* Retrieved from http://cosmosrb.com August 15th, 2019

[2] The PostgreSQL Global Development Group, *PostgreSQL,* Retrieved from https://www.postgresql.org August 15th, 2019

[3] JSON-RPC, *JSON-RPC,* Retrieved from http://www.jsonrpc.org/specification August 15th, 2019