How Automation Has Improved the Performance of the MMS SDC

Julie Barnum, Kim Kokkonen, Kristopher William Larsen, James Craft and Christopher K Pankratz

Laboratory for Atmospheric and Space Physics, University of Colorado, Boulder, CO

IN11E-0700

Abstract

The Magnetospheric Multiscale (MMS) Science Data Center (SDC) at the Laboratory for Atmospheric and Space Physics (LASP), at the University of Colorado, has managed MMS science and ancillary data processing and distribution since MMS launched in March 2015.

The MMS SDC employs automation in nearly every part of its operations. Automation is used to start up processing "runners" that listen on queues for new processing jobs, which are triggered by configurable timing rules, including cron and operational events, or certain data/data tables being available. A separate set of SDC code then automatically creates processing jobs and tracks its progress. The MMS SDC runs processing jobs for each instrument (47 different job types in total), ranging from levels l1a to l3, for "survey" and "burst" modes, plotting, and CDF creation. The SDC runs anywhere from a few hundred to over 2,000 jobs per day (on average, 1,000 jobs per day). The MMS scientist-in-the-loop (SITL) process requires the SDC to be available and ready to handle issues 24/7/365, which can be challenging due to the limited staffing on MMS a few years into the mission.

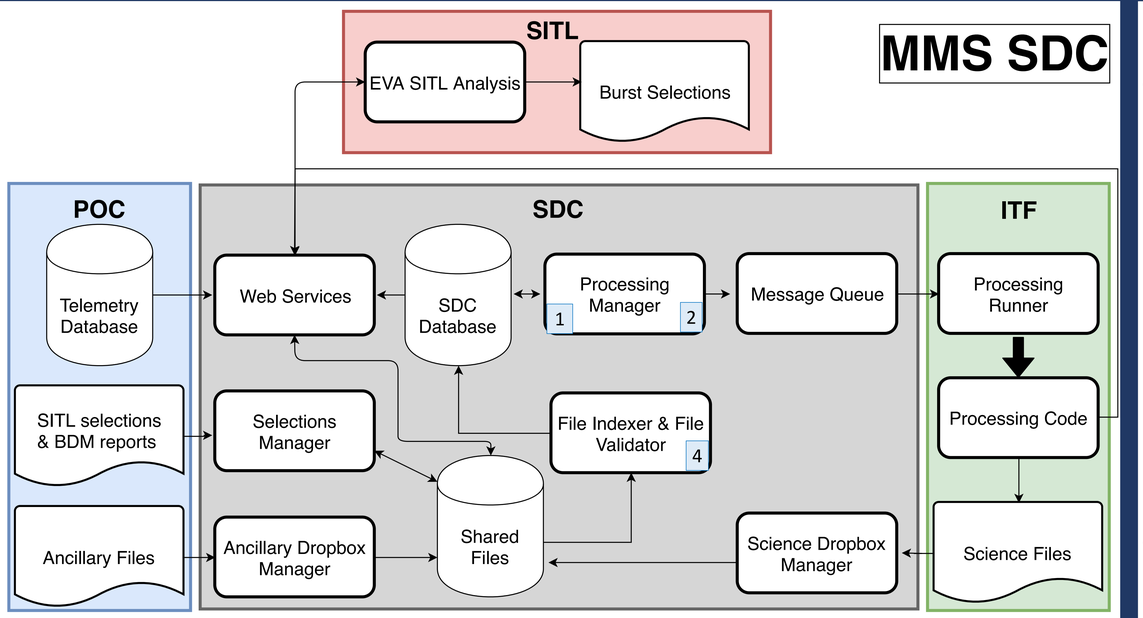
Automated processing relieves some of the load from the software engineers working on the SDC, as well as ensuring continued smooth operation of the MMS SDC. Various areas for improvement, and extra automation, have been implemented. This poster will focus on automation improvements to keep the system running smoothly with almost no human involvement.

1) Processing Runner Monitor

- Ensures that the MMS SDC can constantly have runners listening for processing jobs
- Occasionally, processing jobs would fail without any notification to the SDC (especially problematic during network outages that brought down processing runners)
    - Added code to the Processing Manager that keeps track of each processing runner's PID
    - Also added a cron job that runs at the top of each hour to check that each PID is still active on the specified machine, if it isn't, the SDC is notified via email

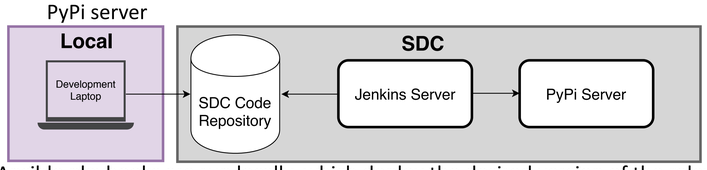
2) Processing Job Watchdog

- Ensures that team members can adequately monitor the MMS SDC's health and functionality
- Occasionally, processing jobs would run for a much longer than normal time period, forcing the SDC to manually kill them
    - Added a watchdog timer
    - If processing jobs run > 12 hours, they're automatically killed, and the SDC and relevant ITF are each notified via email
- This functionality was extended to monitor Logstash JDBC ingest jobs which also occasionally hung

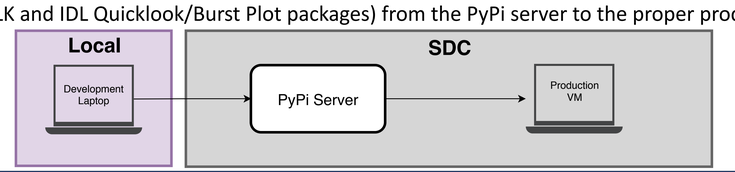


3) ELK and IDL Quicklook/Burst Plot Ansible Deploy

- The MMS SDC uses ansible, an open-source ITF automation platform, which simplifies deploys and allows flexibility in the deploy location
    - The MMS SDC's IDL Quicklook and Burst Plot code, and its ELK code, were modified to use this approach
- The packages can be be deployed following the same pattern as other MMS ansible deploys
    - First, local changes are merged with the MMS SDC git master branch
        - The master branch is then pushed to the MMS SDC Bitbucket repo
        - A Jenkins job then pulls the master branch's code from Bitbucket, runs tests in the repo, and when successfully completed, kicks off a 2nd Jenkins job that builds the various SDC Python packages, tags them with a version number, and pushes the packages to the local MMS SDC PyPi server



- Then, Ansible playbooks are run locally, which deploy the desired version of the relevant packages (here, the ELK and IDL Quicklook/Burst Plot packages) from the PyPi server to the proper production VM

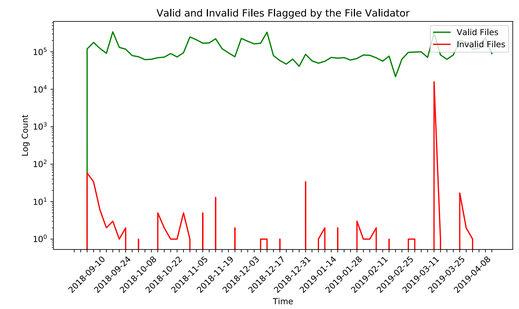


Important Acronyms

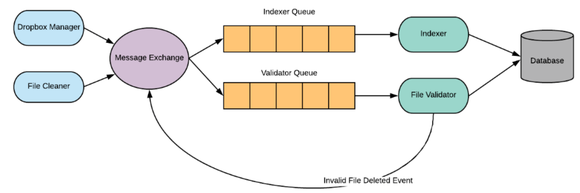
- MMS: Magnetospheric Multiscale
- SDC: Science Data Center
- SITL: Scientist-in-the-loop
- ITF: Instrument Team Facility
- ELK: Elasticsearch, Logstash, Kibana
- JDBC: Java Database Connectivity
- PyPi: Python Package Index
- CDF: Common Data Format
- RabbitMQ: open-source message-broker software
- POC: Payload Operations Center
- BDM: Burst Data Management

4) File Validation

- Ensures the integrity and validity of CDF files that end up in Shared Files
    - Automating this helps speed up the rate at which the SDC learns about invalid files; the SDC can also monitor invalid files via the Kibana GUI



- Works via a RabbitMQ queue system; the Science Dropbox Manager publishes file add, update, and delete notifications to a message exchange, which broadcasts the message to the File Indexer and Validator queues (File Validator only considers adds/updates)



- File Validator runs in the background, listening for messages on the queue – when it receives a message, it begins the validation process where it performs two checks on data to ensure that data files are of good integrity and are not corrupted at any point in the process of uploading the files and copying the files from an ITF's dropbox to the Shared Files
    - 1st: md5checksum (corruption check between manifest and file in ITF dropbox, and manifest and file in Shared Files)
    - 2nd: If md5checksums pass, the validator performs cdfdump (CDF validation performed on file in Shared Files)
- Validation occurs in a separate, asynchronous stream so there is no delay to time-sensitive SITL files
- Files remain in an ITF's dropbox until validation is complete, then every valid file is deleted from the ITF's dropbox
- Invalid files are **not** deleted automatically to allow for manual inspection

MMS SCIENCE DATA CENTER

HTTPS://LASP.COLORADO.EDU/MMS/SDC/PUBLIC/

Laboratory for Atmospheric and Space Physics
University of Colorado **Boulder**