A Model of Data Processing Pipeline for Space Weather Analysis and Forecast*

Minh-Duc Nguyen[1][0000−0002−5003−3623]

Skobeltsyn Institute of Nuclear Physics, Lomonosov Moscow State University, Moscow, Russia nguyendmitri@gmail.com

Abstract. Space weather is a branch of space physics that studies various factors in the near-Earth space such as solar wind, magnetosphere disturbance, solar proton events, and others, which make a massive impact on the Earth. In practice, data measured by different satellite instruments need to be gathered and appropriately transformed before use in space weather analysis and forecast. The data processing pipeline involves a large number of various programs. It also requires in-depth technical knowledge of both satellite instruments and programming tools so that data will be processed correctly. Building such a data pipeline is time-consuming and error-prone. The correctness of the output data produced by the processing pipeline is one of the critical factors that define the success of an analysis or a forecast model. This work proposes a model that describes how the data processing pipeline might be organized and how to build a distributed data processing system based on the proposed model.

Keywords: Data processing · ETL · Space weather analysis and forecast · Space physics.

1 Introduction

Space weather is a branch of space physics that studies complex processes, so-called space weather factors, happening in the near-Earth space. The main driving force of such processes is high energy particles (protons, electrons, and alpha-particles) that are mostly ejected from solar events, heading from the Sun toward the Earth, and directly impacting Earth's heliosphere and magnetosphere, satellite and ground systems. Some of these processes, such as quasi-stationary solar wind fluxes, solar proton events, and fluences of outer radiation belt electrons, are well known and broadly studied. Real-time monitoring and forecasting the impact of these factors on satellite and ground systems are critical missions. For the last several decades, many space experiments have been launched to accomplish these missions. Hundred of satellites are rotating around

* Supported by the Russian Science Foundation, grant #16-17-00098.

the Earth at different orbits collecting data measured by various instruments in real-time. The data are later used to analyze space weather factors and to develop operational models that describe and predict the behavior of these factors and their impact.

There are three most significant challenges that scientists encounter each time a new study of space weather starts. Searching for datasets that cover a period when specific space weather events happened is one of them. In practice, datasets are not always complete. Missing data are a common issue. Another challenge is finding alternative datasets that cover a specific interval when the primary datasets have missing data. The third challenge is to transform different datasets presented in various formats into one and normalized them so that they can be used together. Solutions to these challenges are still an active topic for research. Until today, no solution can provide a smooth experience of data acquisition and match the demand of scientists of the space weather community. Several ongoing projects address these challenges, such as the Planetary Data System [1] and Euro Planet [2]. But due to the complexity and scale of these projects, it is still hard for individual researchers to benefit from their results. While the number of data products provided by these projects is vast, the available search engine and API lack the flexibility that allows researchers to search and retrieve data without any specific knowledge.

To fulfill the need of individual researchers, a Satellite Data Downloading System (SDDS) [3] has been created at the Skobeltsyn Institute of Nuclear Physics, Lomonosov Moscow State University. The system collects data from the most used satellites in space weather research and provides a user-friendly API that allows researchers to search and retrieve data with ease. The system is based on a data processing pipeline model that will be considered in detail in this paper.

2 The data processing pipeline model

Data processing pipeline for a satellite instrument is a process of reconstructing instrument and payload data at full resolution with any and all communications artifacts such as synchronization frames, communication headers, etc. During this process, data products are produced at various levels ranging from Level 0 to Level 4 [4]. Level 0 products are raw data at full instrument resolution and are not used in research due to communication artifacts. Calibrated data products of Level 1A or higher are often used. A data provider might provide data products at various levels. To be able to use data from different instruments together, it is required to process them to a correct level applying all radiometric, geometric coefficients, and georeferencing parameters.

Despite the diversity of satellites and their instruments, they all share some common procedures in the data processing pipeline. The data processing pipeline model of the SDDS system (the Model) is used to describe these common procedures and what actions should be taken in each of them.

The Model involves the following entities:

- the data processing system (the system);

- the data source;
- the gateway that connects the data source to the Internet;
- the source file provided by the data source (can be at various level);
- the satellite;
- the instrument that is set up on board of the satellite;
- the instrument file that contains scientific payload;
- the local server where the data processing system is functioning;
- the data storage where source files, instrument files, and the processing result are stored;
- the database.

The Model splits a typical data processing pipelines into seven stages:

1. connecting to a data source;
2. checking for new source files;
3. downloading the new source files;
4. extracting instrument files from the source files;
5. processing the instrument files;
6. loading the processing result to the database;
7. moving all files to the data storage.

In the connecting stage, the data processing system establishes a connection to the data source. If the data source is behind a gateway on a private subnet, the system creates a VPN connection to the gateway, adds necessary routes to the routing table, and establishes another connection to the data source. If required, the system also authenticates itself against the data source.

In the checking stage, the system searches for new source files by comparing the remote file list with the local one or using the recept (the file) that contains links to the new source files. A source file is considered new if it does not exist in the local data storage or if the last modification time or the file size differs from the existing local one.

In the downloading stage, the system downloads the new source files to the local server. After downloading, the system calculates the checksums of the files to check for correctness. Network issues are handled by the system in this stage. If the connection drops during a downloading session, the system will try to reconnect to the data source and recover the downloading operation.

In the extracting stage, the system reconstructs the instrument files from the source files. If the source file is a zip-archive, the systems will uncompress it first. If several instrument files are packed into one single source file in a custom binary data format, the system will unpack the instrument files using the format specification.

In the processing stage, the system executes special programs, so-called decoders, to transform the payload from lower to a higher level and store the result in the CSV format. The system might perform additional post-processing routines to produce high-level data such as Levels 2, 3, and 4. A set of instrument files can be processed in parallel. If there is a dependency between files of

different instruments, the systems will execute the processing routine in strict order.

In the loading stage, the system loads the processing result in the CSV format to the database. The schema and table structure depends on the hierarchy of instruments and data channels. In this stage, data at different resolutions are calculated inside the database using the original resolution.

In the moving stage, the system moves all files to the long-term data storage. The file and directory structure of each satellite reflects the hierarchy of instruments. Depending on the size, files can be split according to a specific time-based period: by year, by month, or by day.

3 Technical Implementation

The components of the SDDS system responsible for the data processing pipeline were implemented based on the Model described above. Most of them were designed using the microkernel [5] pattern widely used in operating system component design. The idea of the microkernel pattern is that the primary business logic is implemented in the core component. Everything else is implemented as pluggable modules that can be loaded and executed dynamically in run time. The interface between the core component and modules is determined, so different versions of a module or modules with similar features can be used interchangeably. This pattern ensures the flexibility and the scalability of the resulting system and the isolation of components.

The common logic of the data processing pipeline was implemented in a base class representing an abstract satellite controller. The changing logic is described in the configuration file, each of which belongs to a specific satellite controller. Data sources and instrument data decoders are described in the configuration file in JSON format along with other parameters, such as the order in which files from multiple data sources are processed. Each decoder has its own configuration file. The stages of the data processing pipeline are also defined in the configuration file. For example, if the data source already provided instrument files, the extracting stage can be omitted, and thus there are only six stages defined in the configuration files.

The abstract base controller has an interface consisting of a set of methods. Each method performs common actions in each stage. Each method has a set of standard-type input parameters and two function-type parameters. The first function-type parameter represents a pre-processing function that is called before any common actions in the method. The second function-type parameter represents a post-processing function that is called after all common actions in the method. A specific satellite controller is implemented as a class derived from the base class. In the derived class, the base methods might be reused as-is. The derived class might have its specific methods that are later passed to the base methods as parameters to be used as pre- and post-processing functions. If the logic of the data processing pipeline is complex, the base methods might be redefined completely in the derived class.

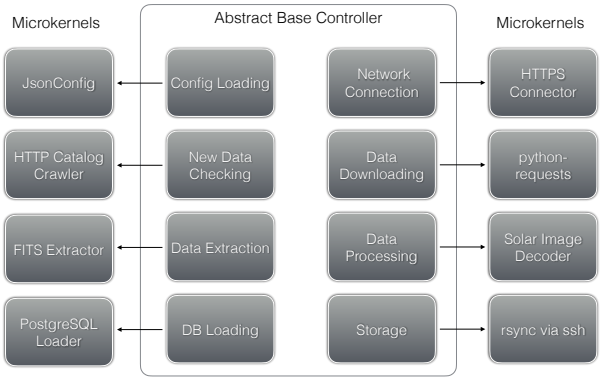

Fig. 1. The base controller and microkernels implementing the data processing pipeline of the SDO controller

The actions performed inside a base method are implemented as microkernels that can be loaded and called dynamically depending on the metadata of the satellite described in a configuration file. For example, if the data source uses HTTPS as a connection protocol when the base method responsible for establishing a connection is called, it will execute the microkernel-method for the HTTPS connection. The same approach was used in the implementation of other stages. The controller class implementing the data processing pipeline of the Solar Dynamics Observatory (SDO) [6] satellite is shown in Fig. 1.

The microkernel approach also fits when it comes to implementing actions in the instrument data processing stage. When the base method is called, it, in turn, calls the specific decoder used to process the instrument data. If several instruments use the same format to present the data, a single decoder that processes data in that format can be reused.

The satellite controller can run in automatic mode and interactive mode through the command-line interface. In the automatic mode, the controller passes each source file through the data processing pipeline. In the interactive mode, actions of a specific stage can be executed against the input file manually when the corresponding argument is passed. The interactive mode makes it possible to adapt the controller to a complex processing scenario. For example, when a number of files need to be reprocessed, instead of passing files one by one through the pipeline, one can run the controller in interactive mode to download all files to a temporary buffer first and then start processing files.

4 Conclusion

The data processing pipeline model described in this paper has been used as a baseline to design and implement the SDDS system. Applying the microkernel pattern reduces the development time required to support new satellite instruments, especially when they share the same feature or data format similar to an existing one. The flexible at the same time determined interface of the abstract base controller makes it possible to maintain the compatibility across components. Currently, data of the twenty most used satellites and geomagnetic indices are being processed by the SDDS system. Processing pipelines are executed on regular basics. The frequency varies from 5-minute intervals to 1-day.

References

1. The Planetary Data System, https://pds.nasa.gov. Last accessed 29.06.2020
2. Euro Planet, http://www.europlanet-vespa.eu. Last accessed 29.06.2020
3. Nguyen M.-D.: A Scientific Workflow System for Satellite Data Processing with Real-Time Monitoring. EPJ Web of Conferences 2018, vol. 173, 05012. https://doi.org/10.1051/epjconf/201817305012
4. National Aeronautics and Space Administration: Earth Observing System Data and Information System (EOSDIS) Handbook, https://cdn.earthdata.nasa.gov/conduit/upload/5980/EOSDISHandbookWebFinal1.pdf. Last accessed 29.06.2020
5. Richards M.: Software Architecture Patterns. O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472 (2015)
6. National Aeronautics and Space Administration: Solar Dynamics Observatory, https://sdo.gsfc.nasa.gov Last accessed 29.06.2020