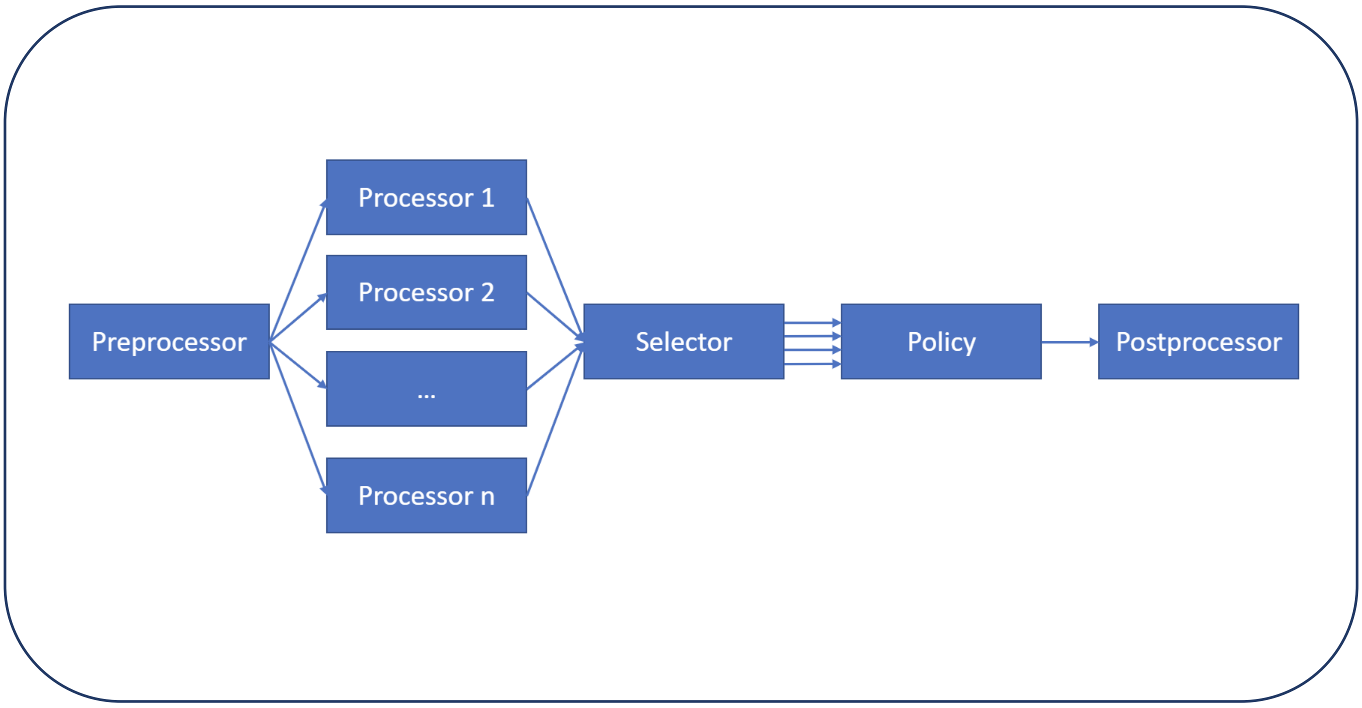
→ DISCOVERY



DCaaS: Data Compression as a Service

Executive Summary Report

Study

Campaign: OPS-SAT Experiments

Affiliation: VisionSpace Technologies GmbH (Prime)

Activity summary:

The Data Compression as a Service (DCaaS) study aims to bring the benefits of data compression to the experimenters on the ESA OPS-SAT platform. It focuses on using the POCKET+ algorithm for the compression of housekeeping telemetry data. The project contains a space component, which performs the compression of provided user data, and a ground component, in for the decompression of transmitted user data. Three use cases were selected within the scope of the project: file-to-file, packet-to-packet, and stream-to-stream lossless data transfers.

Publishing Date: 4 Nov, 2022
Contract Number: 4000137383/22/NL/GLC/ov
Implemented as ESA Express Procurement - EXPRO

1. Executive Summary

The Data Compression as a Service (DCaaS) study aims to bring the benefits of data compression to the experimenters on the ESA OPS-SAT platform. It focuses on using the POCKET+ algorithm for the compression of housekeeping telemetry data.

POCKET+ is a universal and lossless compression algorithm for fixed-length time series data. It is the successor of POCKET, which is under a patent held by ESA. In POCKET+, only the compression procedure is defined, and it is an asymmetrical compression algorithm with a more complex compression operation. It can be an adaptive compression method, but this is not mandatory. POCKET+ compresses satellite housekeeping telemetry data in near real-time. This data is of constant length and often has repeating patterns and fixed positions where the data changes. This extracts the most recent changes, and only the changes are transmitted, assuming that the remaining data is unchanged. With data compression, more offline spacecraft operations are enabled, lowering operational costs. Before this project, there was no ESA implementation of the algorithm available for the industry. We have demonstrated the use of the POCKET+ algorithm with this study and published an open-source implementation for further investigations of its performance.

Three use cases were selected within the scope of the project: file-to-file, packet-to-packet, and stream-to-stream lossless data transfers. The first case compresses files onboard the spacecraft, saves the result as a file and reverses the process on the ground. We can avoid overhead generated by the compression algorithm's robustness level and repetitive synchronization frames by selecting a lossless data channel. This method is expected to provide optimal compression performance, and we can compare traditional compression methods against the POCKET+ algorithm. The second case takes telemetry packets onboard the spacecraft and transfers the compressed data over a lossless ZMQ connection to the ground. The last case takes an input stream onboard the spacecraft and transfers the compressed data over a lossless TCP connection to the ground. This is possible thanks to the features of the OPS-SAT platform. In the ground systems, we extract the data and provide the decompressed stream to experimenters. The use cases are extended by their respective runtime functions.

The current POCKET+ implementation for OPS-SAT uses manually configured contexts and is enabled only for selected data aggregations. To overcome this limitation, we developed the CompressionCache. The central component in the CompressionCache holds multiple objects that can apply a compression algorithm to a given data packet.

All compression objects process every data packet, and it is possible to use stateful compression algorithms. A newly introduced mechanism distinguishes which compression object compresses which data packet. With this approach, it is no longer necessary to manually handle single compression contexts. The entire packet stream can be fed into the CompressionCache, and it decides autonomously which packets to compress. Different algorithms can compete within the CompressionCache against each other.

The structure of the data processing pipeline is as follows: The preprocessor gets the input data passed before the actual processor uses it. This processor performs the actual data compression. It contains a compress function, as well as reset, stash, and pop functions, allowing for the use of stateful processors. The selector returns the index of the selected processor. The policy gets all available information about the cache's status to enable a wide range of cache management functions. Finally, the postprocessor completes the final steps before handing the data to the runtime. On the DecompressionCache side, the structure is nearly identical, except that there is no selection of a processor since we get this information from the pre-processor, and no policy.

The file runtime for the CompressionCache is implemented as a mix-in, extending the CompressionCache with a run function. This runtime processes one input file at a time and stores the compressed output to a new file. It must be called with parameters for every file to be processed. The executable terminates after the compression of the input file is finished or if an error occurs during runtime.

In the ZMQ runtime, the networking interfaces are a listening ZMQ socket for incoming data and an outgoing ZMQ connection that sends compressed data to the ground based DecompressionCache stream runtime. The executable terminates after all packets have been sent, or if an error occurs during runtime.

The TCP runtime, unlike the file and ZMQ runtimes, does not stop after processing a certain amount of data, it runs continuously until an error occurs that cannot be handled or when the process is stopped by the operating system. The interfaces are a listening TCP socket for incoming data, and an outgoing TCP connection that sends compressed data to the ground based DecompressionCache stream runtime.

We plan to test and deploy our use cases on the OPS-SAT platform, and after the application is validated, we will reach out to other experimenters and offer our product as a service to increase their data return.

On the side of technical developments for the CompressionCache, we plan to further improve its performance in the future. We want to reduce the hardware utilization and increase the data throughput. This can be accomplished by tuning the internal algorithms developed in the scope of this project and by refactoring our POCKET+ implementation.

Besides the applications on the OPS-SAT platform, we want to explore use cases outside of the field of spacecraft operation.

Drones for delivery services often use cell phone networks and satellite internet as a fallback option when operating in remote areas. These areas can be underdeveloped or have poor mobile network coverage due to geographic constraints. Such a scenario could be a drone delivering emergency medication to a remote island. We see a potential benefit for the operators of the drones by decreasing the traffic sent over satellite connections, which is typically very expensive and billed by the amount of data transferred. The same billing policy is true for industrial IoT applications which use cellular networks or specialized networks, like LoRaWAN.