Using Product Access Layer to hide the complexity of data storage from Herschel users

B. Li[1] and D. Liu[2]
National Astronomical Observatories, Chinese Academy of Sciences, Beijing, 100012, PR China

GSegment Space Technologies, Inc., Beijing, 100012, PR China

S. Guest[3]
Rutherford Appleton Laboratory, Chilton, Didcot, Oxfordshire OX11 0QX, UK

M. Huang[4]
National Astronomical Observatories, Chinese Academy of Sciences, Beijing, 100012, PR China

P. Balm[5], J. Bakker[6], J. C. Segovia[7], J. Saiz[8], H. Siddiqui[9]
ESA Herschel Science Centre, ESAC, Villafranca del Castillo, Spain

and

K. Edwards[10]
SRON Netherlands Institute for Space Research, Groningen, The Netherlands

University of Waterloo, Waterloo, Ontario, N2L 3G1, Canada

Launched on May 14, 2009, ESA's Herschel Space Observatory is the largest astronomical telescope ever launched. It is in routine science operations and has already generated data for more than 25000 observations. The users of Herschel data products are astronomers, data processing software developers, instrument engineers, and calibration scientists. Different users have different scenarios and requirements on data access. Different data storage systems and mechanisms (including commercial Object Oriented database, Herschel Science Archive, local disk, remote HTTP server and so on) are selected to build up data processing applications. We developed a software package named Product Access Layer (PAL) to hide the complexity of data storage systems from users. It contains an implementation-independent interface to store, query and retrieve all kinds of data products. PAL helps users to focus on scientific data processing and analysis. It has become a fundamental component of the Herschel Interactive Processing Environment (HIPE), which is the main application interface for most Herschel users.

[1] Software Engineer, GST, B616, 20A Datun Road, Chaoyang, Beijing, China.
[2] Software Engineer, GST, B616, 20A Datun Road, Chaoyang, Beijing, China.
[3] SPIRE ICC Software Development Manager, RAL Space, RAL, Chilton, Didcot, Oxfordshire, UK
[4] Professor, NAOC, 20A Datun Road, Chaoyang, Beijing, China
[5] Software Engineer, ESA Herschel Science Centre, ESAC, Villafranca del Castillo, Spain
[6] Software Architect, ESA Herschel Science Centre, ESAC, Villafranca del Castillo, Spain
[7] Software Engineer, ESA Herschel Science Centre, ESAC, Villafranca del Castillo, Spain
[8] Software Engineer, ESA Herschel Science Centre, ESAC, Villafranca del Castillo, Spain
[9] Software Engineer, Gaia Science Operations Centre, ESAC, Villafranca del Castillo, Spain
[10] HIFI Data Processing System Architect, Physics & Astronomy, University of Waterloo, Waterloo, Ontario, N2L 3G1, Canada

I. Introduction

THE Herschel Space Observatory, launched on 14th of May 2009, is the largest astronomical telescope ever launched[1]. It is in routine science operations and has already generated data for more than 25000 observations.

Developed since 2002, seven years before the Observatory was launched, the Herschel Common Science System (HCSS) provides a common system to astronomers, instrument calibration scientists, instrument engineers and scientific software engineers to support functions to instrument and science operations from Instrument Level Tests (ILT), Integration System Tests and in-orbit operations, to post-operations [2].

The Product Access Layer (PAL) is a software package of HCSS to provide an implementation-independent interface to store, query and retrieve all kinds of data products that are distributed in various storage systems that may also evolving during different projects phases. The PAL was not in the initial design of the HCSS. It was introduced after the pipeline was starting to be developed with the help of the ILT data. Development towards in-flight scenarios began generate high-level data products which were stored in various systems. Lacking a consistent API to access these data exposed unnecessary complexities to calibration scientists, instrument engineers and scientific software developers. The PAL package overcome this challenge by defining a common set of API, providing default implementations on several storage systems used in HSC (ESA Herschel Science Centre, ESAC) and ICCs (Instrument Control Centre), and a common test suite to validate all PAL implementations.

The PAL is coded in Java/Jython [3]. It is bundled and delivered with the Herschel Interactive Processing Environment (HIPE) [4], which is an integrated data acquisition and processing tool for Herschel Data users. The source code is freely available to public under the GNU lesser general public license witch is applying to the whole HIPE software. NAOC (National Astronomical Observatories, Chinese Academy of Sciences), GST (GSegment Space Technologies, Inc., Beijing, China), HSC and SPIRE ICC are main contributor institutes of the PAL package.

II. Requirements

Most of the high level requirements of PAL are inherited from the design of the Herschel Ground Segment (HGS) [5] and the HIPE.

The HGS was designed to be geographically distributed. The main organizations include Mission Operation Centre (MOC) in Germany, HSC in Spain and three ICCs (one ICC per instrument) in different countries in Europe. It requires PAL to support accessing data from the existing storage systems that institutes are using, e.g. object database, Herschel Science Archive System and file systems.

The HGS was designed to support smooth transition across mission phases. The data structures, the configurations and underlying mechanisms of the storage systems are all evolving throughout the mission lifetime. This requires PAL to support smooth database schema evolution, no limitation on supporting new storage systems and the possibility to optimize data accessing performance according to different system characteristics.

The HIPE was designed to support multiusers with an integrated software environment combined for data retrieval, pipeline execution and analysis. As a seamless integrated component of HIPE, the PAL needs to conform to the general design principle of the HIPE, i.e. it should be object oriented, Jython syntax friendly and based on HCSS data structures.

III. Components Design

The PAL is a data access layer, implemented using the widely used Data Access Object design pattern [6]. Unlike other popular Java data access layers, such as JDBC [7], JDO [8] or JPA [9], the PAL is more astronomer friendly and is not intended to support any data structure or database features that are not used for Herschel science data processing. The only data structure that PAL supports is Product, which consists of metadata and datasets, mirroring the structure of a FITS file. As a Product can contain collections of Products, this allows complex data structures in practice. The PAL API has support for saving, loading, querying and removing as other DAL. It also has support for tagging and versioning that are needed for data processing, but there is no support for features like atomic transactions or rollbacks, though it does not prevent individual concrete PAL implementations providing such support. These limitations on data structures and features make PAL API more concise and decrease the cost of developing PAL implementations on new kinds of data sources.

As shown in Fig. 1, the components of PAL include PAL generic API, product pool implementations, PAL GUI and common test suite.

To hide the underlying storage complexity, the PAL generic API is the only application interface exposed to Herschel data users. It is a Java API that defines all functions for Herschel science products data access. It receives user's calls from the PAL GUI or users data processing scripts, and then forwards calls to underlying product pool implementations.

Product pool is a PAL abstraction to data storage systems. We have developed product pools implementations to work with Versant object database, file systems, HTTP servers and the Herschel Science Archive. Two most popular product pool implementations are Local Store and HSA Pool. Local Store wraps data sources from file systems. It is designed for astronomers to run their data processing scripts to access data on their own computers. By default, it stores products as FITS files, which is the most familiar file format for astronomers. It supports astronomers to manage their local data with customized directory structures and file names. It also supports compressing mode to save disk space. Apache Lucene[10] library was employed in Local Store to index metadata of products and speed up queries. HSA pool allows users to access and download observations data from the Herschel Science Archive, which is the principal database that contains all scientific data generated by Standard Product Generation (SPG) pipelines.

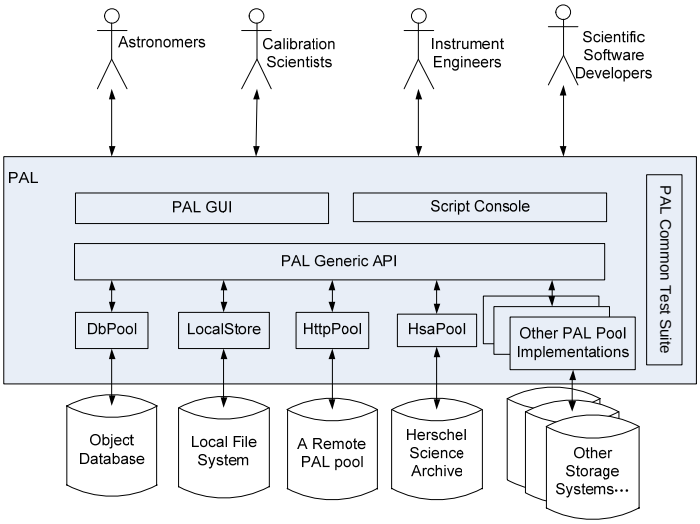

Figure 1. PAL Components overview.

To help end users to browse products in their product pools easily, PAL provides a GUI component that is integrated within HIPE. It allows users to issue queries, check results and look into products.

The PAL common Test suite was introduced to ensure all PAL product pools implementations work in the same behavior, which is fundamental for data processing scripts to be reused among different environments. More than 300 test cases were defined and developed based on JUnit[11] library.

IV. Current Status and future work plan

The PAL performs within requirements in its most demanding environment, which are Herschel's in-orbit operations, where the amount of raw telemetry data received by HSC is about 1.5 GB per day. Up to 50 GB science data are processed and ingested into HSA per day during normal operations. The amount of data can be up to 500 GB if HSC is performing bulk reprocessing. Up to 300 observations are produced by the HSC and ingested into the HSA per day. The PAL implementation is currently dealing with 57TB worth of products in the archive, translating to 17 million products.

The development of the PAL is still in progress. Most of the future work is focused on continued performance optimization on various product pool implementations, including faster data saving and loading, less memory consumption, less storage space consumption and faster querying.

Appendix A
Acronym List

API	Application Programming Interface
DAL	Data Access Layer
ESA	European Space Agency
ESAC	European Space Astronomy Centre
FITS	Flexible Image Transport System
GST	GSegment Space Technologies, Inc.
GUI	Graphic User Interface
HCSS	Herschel Common Science System
HGS	Herschel Ground Segment
HIPE	Herschel Interactive Data Processing Environment
HSA	Herschel Science Archive
HSC	Herschel Science Centre
HTTP	Attitude Control and Determination
ICC	Instrument Control Centre
ILT	Instrument Level Test
JDBC	Java Database Connectivity API
JDO	Java Data Objects API
JPA	Java Persistence API
MOC	Mission Operation Centre
NAOC	National Astronomical Observatories, Chinese Academy of Sciences
PAL	Herschel Ground Segment Product Access Layer
RAL	Rutherford Appleton Laboratory, UK
SPG	Standard Product Generation

Acknowledgments

Herschel is an ESA space observatory with science instruments provided by European-led Principal Investigator consortia and with important participation from NASA. B. Li and M. Huang wish to thank National Astronomical Observatories, CAS and Chinese Academy of Sciences for their funding support KJCX2-YW-T20.

References

[1] Pilbratt, G.L., Riedinger, J.R., and Passvogel, T. "A&A." 2010: 518.

[2] Brumfitt, J., and Zäschke, T., Herschel Common Science System Overall Architecture and Design. December 4, 2007.

[3] Jython, Jython: Python for the Java Platform, Software Package, Ver. 2.5, 2011.

[4] Ott, S. “The Herschel Data Processing System – HIPE and pipelines – up and running since the start of the mission”, ADASS XIX, ASP Conference Series, Vol. XXX, 2009.

[5] HGSSE. Herschel Ground Segment Design Description. 2006.

[6] DAO, “Core J2EE patterns – Data Access Object”, Sun, 2004.

[7] JDBC, Java Database Connectivity, Software Package, Java SE 6, Oracle, 2011.

[8] JDO, Java Data Objects, Software Package, Ver. 3.0, The Apache Software Foundation, 2011.

[9] Biswas, R., “The Java Persistence API - A Simpler Programming Model for Entity Persistence”, Sun, May 2006.

[10] Apache Lucene, Apache Lucene Index Engine, Software Package, Ver. 3.0, The Apache Software Foundation, 2011.

[11] JUnit, JUnit testing framework, Software Package, Ver. 4.0, Kent Beck, 2011.