Pilot Requirements & Definitions

Deliverable Number:	D.2.1
Due Date:	30.06.2023
Date Of Submission:	14.07.2023
Nature:	R
Dissemination Level:	PU
Work Package:	WP2
Lead Beneficiary	SED

Grant Agreement No 101092850

Funded by the European Union

Document Control Sheet

Deliverable Title:	Pilot Requirements & Definitions
Authors:	Krzysztof Nienartowicz (SED)
Contributors:	Laurent Eyer (UNIGE), Daniel Krefl (SED), Michael Wurzer (KTM)
Reviewers:	Foivos Zakkak (RHAT), Vincent Casillas (SIPEARL)
Approved By:	Christos Kotselidis (UNIMAN), Dionisios Pnevmatikatos (ICCS)

Document History

Version	Date	Status	Description/Comments
0.1	16.05.2023	Draft	ToC
0.2	12.06.2023	Draft	Added initial descriptions of UNIGE & SED pilots
0.3	19.06.2023	Draft	Added initial description of KTM pilot
0.4	05.07.2023	Draft	Final pilots' descriptions & KPIs
0.5	10.07.2023	Draft	Draft released for internal review
0.6	11.07.2023	Draft	Integrated changes based on RHAT review
0.7	12.07.2023	Draft	Integrated changes based on SIPEARL review
0.8	13.07.2023	Draft	Updates/clarifications
1.0	14.07.2023	Final	Final version submitted to EC

Disclaimer

AERO has received funding from the European Union's Horizon Europe research and innovation programme under Grant Agreement No 101092850. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the granting authority. Neither the European Union nor the granting authority can be held responsible for them.

Although the material and information contained in this document is considered to be precise and accurate, neither the Project Coordinator, nor any partner of the AERO Consortium nor any individual acting on behalf of any of the partners of the AERO Consortium make any warranty or representation whatsoever, express or implied, with respect to the use of the material, information, method or process disclosed in this document, including merchantability and fitness for a particular purpose or that such use does not infringe or interfere with privately owned rights.

In addition, neither the Project Coordinator, nor any partner of the AERO Consortium nor any individual acting on behalf of any of the partners of the AERO Consortium shall be liable for any direct, indirect or consequential loss, damage, claim or expense arising out of or in connection with any information, material, advice, inaccuracy or omission contained in this document.

Table Of Contents

Executive Summary

This document presents the pilots that will be used in the context of the AERO project. The purpose of the document is for use case partners to explain their use cases, their various software components, SLAS, and envisioned integrated systems for the final evaluation as well as a broader context in which the use cases are embedded: automotive, scientific high-performance computing and parallelized relational databases for Big Data time-series.

List of Abbreviations & Acronyms

Abbreviation/Acronym	Meaning
AOT	Ahead of Time
CU7	Gaia Variability Study Coordination Unit
DB	Database
DBMS	Database Management System
DCN	Document Change Notice
DPCG	Gaia Data Processing Center in Geneva
ESA	European Space Agency
GPAC	Gaia Data Processing and Analysis Consortium
HPC	High Performance Computing
JDK	Java Development Kit
JIT	Just in Time
JMS	Java Messaging Service
JVM	Java Virtual Machine
LTS	Long Term Support
MPP	Massive Parallel Processing
PG, TBase	PostgreSQL DBMS or PG MPP fork
REST	REpresentational State Transfer
SLA	Service-Level Agreement
VIN	Vehicle Identification Number

1 Introduction

The AERO project output is intrinsically linked to the possibility of using it to solve real-life problems defined in the context of WP2 "*Pilot Migration and Optimization on the EU Cloud*". The development and validation of the ecosystem is thus tightly bound to the three pilots we present here.

The pilot brought by KTM covers the dynamic and fast-paced domain of connected vehicles and utilities, defined by the also-fast-evolving legal European system defining automotive and privacy issues in the connected world. The second pilot, defined by the UNIGE Astronomy Department, touches on compute challenges related to signal processing, supervised and unsupervised classification of the Stellar Variability Studies of the ESA Gaia mission. The third pilot, interlinked with the UNIGE goals, led by SED, aspires to enable GPU acceleration of the Massively Parallel Processing (MPP) Database in order to unlock more thorough data analysis, including significant speed-ups for scientific exploration and validation via ad-hoc querying in a petabyte scale MPP scientific Database Management System (DBMS), yet generic enough to be adoptable in other domains.

2 AERO Pilots

2.1 Automotive "Digital Twins" with IoT-Cloud Interoperability (KTM)

2.1.1 Pilot Description

The application of this pilot, called Vehicle Information Service, constitutes the core of a Digital Twin concept. The Vehicle Information Service is a microservice by Pierer Mobility (parent company of Pierer Innovation) to get information about vehicles manufactured by the group - including all brands such as KTM, HUSQVARNA, and GAS. It is part of the whole Pierer Mobility backend services that comprise numerous microservices performing different tasks that involve multiple actors/stakeholders such as customers, dealers, suppliers, etc.

In order to test as many of these services as possible on the Rhea platform, a decision was made to create vertical, self-contained, and testable releases of these services that can be tested on the EU processor platforms. The number of services/releases will be grouped based on characteristics such as: the variety of software components used, external services, databases, etc. In addition, due to cybersecurity considerations, several aspects of these services will be obfuscated and/or altered without compromising neither the functionality of the use case nor their business scope. Finally, all identified microservices will be gradually released during the duration of the first year of the AERO project.

Presently, the Vehicle Identification Service is the first microservice that has been distilled from the Pierer Mobility backend infrastructure and represents a larger class of microservices that are built on top of NodeJS with database interoperability. The purpose of this service is to identify individual vehicles by their unique identification number (VIN). This number is then used to query for detailed information stored in the database. VIN is a number which globally identifies vehicles and it is used by multiple manufacturers and standardized by ISOs. The microservice obfuscates the VIN numbers of vehicles by mapping them internally to custom unique identifiers for cybersecurity reasons.

The database holds an entry for each unique identifier. The entry data include: a) information identifying the motorcycle, such as the model and article number; b) extra information, like whether the vehicle is still under warranty, and c) technical information, including data from vehicle sensors that could be a basis for a Digital-Twin construction, as well as the vehicle's service history. This information can be accessed via a REST API. Moreover, the service can validate vehicle identifiers and perform additional verification activities related to additional business logic.

The last task of the Vehicle Information Service is to manage the database. The entries in the database can be created, updated and deleted via the Vehicle Information Service. This can either be done via the REST API or via Kafka.

2.1.2 Architecture

Vehicle Information Service, being a microservice, is a small part of a bigger system of services working together. Thus, microservices need interfaces to communicate with other parts of the whole system. Figure 1 depicts the architecture around the Vehicle Information Service.

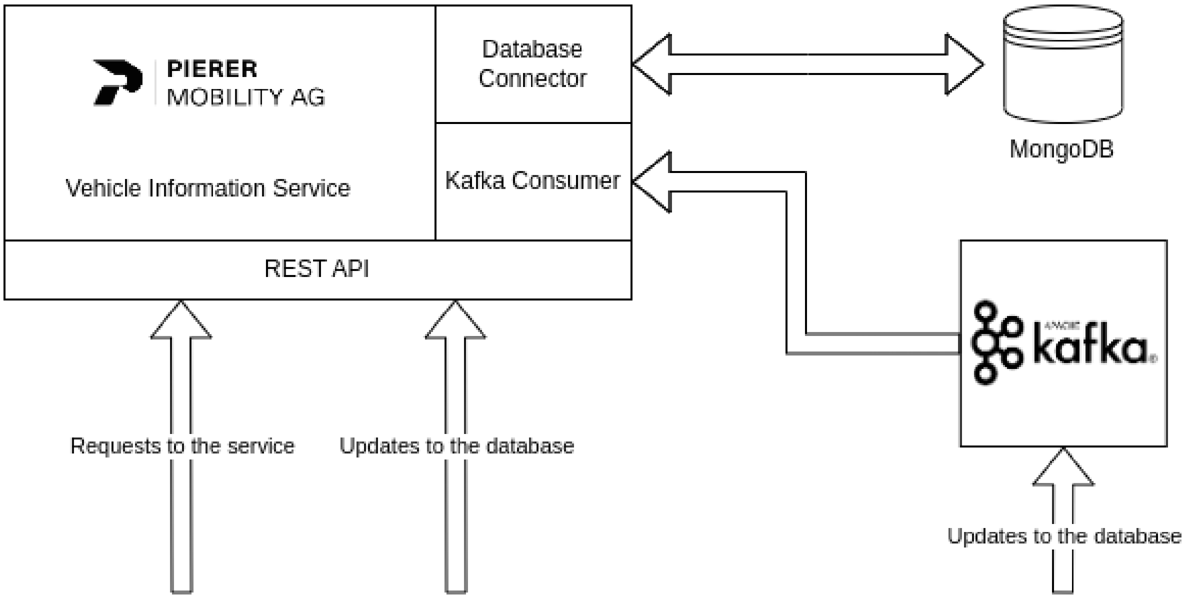

Figure 1 KTM Digital Twin backend

The service has three interfaces to the outside world:

1. The **database connector**, which is used as a pooler of the connection to the database. Data about vehicles can be fetched, updated and deleted.
2. A **Kafka consumer**, which receives updates for the database from the Kafka broker. Other services can send new entries and update or delete existing entries.
3. A REST API, which is the main interface to the service used for the tasks described in Section 2.1.1. Verifying vehicle identifiers and receiving information about a vehicle is done via this API, as it is also done for vehicle identifiers updates. This is the main way other services or users can interact with the Vehicle Information Service.

2.1.3 Software Components

Vehicle Identification Services consist of the software components/frameworks listed in the table below:

Software Component	Description
Docker	Container daemon to run application with dependencies in a container isolated from the rest of the system and other containers
NodeJS	JavaScript Engine to run JavaScript outside of the browser. For instance, it can be used to create a server application in JavaScript.
TypeScript	Type-safe superset of JavaScript. Used to ease the development of JavaScript applications and make them more robust.
MongoDB	Database used to store non-predefined data in a binary JSON format.
Kafka	Kafka is a distributed streaming platform that allows you to publish and subscribe to streams of records. The streams are identified by topic. Each client can subscribe to topics and receive a message as soon as some other client publishes on the same topic.
Zookeeper	ZooKeeper is a server application to manage an application cluster. It works with Kafka to distribute Kafka topic partitions to the brokers in the cluster so that each message is only processed once and the partition is moved to another broker if one broker has gone offline.

2.1.4 AERO Integration Components

The main integration components of this microservice are Docker and Apache Kafka. In addition, depending on the scalability requirements, a Kubernetes deployment may be considered.

2.1.5 Deployment Strategy, Evaluation & KPIs

For initial tests, KTM acquired ARM cloud instances in Microsoft Azure, as all services are presently deployed in the Microsoft cloud. As no GPU and no additional hardware extensions are currently required for this use case, the results are expected to be similar to those when running on the Rhea platform. The service and all additional components are deployed in Docker containers. Thus, it can be moved easily to other hosts if necessary.

The Vehicle Information Service repository includes a test suite mainly consisting of system integration tests, where the application is tested with the database. This suite can be used to test the application and check if the behavior is consistent across both x86 and ARM architectures. In addition, the Artillery load testing framework[1] is used for load testing and performance comparisons to the existing x86 deployments.

Fundamentally, the main KPI of this pilot (or collection of use cases) in the context of the AERO project is to achieve parity both in functionality and in performance with the currently deployed services on x86-based Microsoft Azure instances. In detail, the KPIs that will be assessed for the Vehicle Information Service, as well as all other services from Pierer Innovation are:

- To achieve 100% pass-rate across all test suites currently used (both functional unit tests, and integration tests) on the Rhea platform
- To achieve performance parity - and hence satisfy the current SLAs - in a single-node deployment scenario on the Rhea platform with an x86-based instance. The metrics of interest in this case are transactions per second, mean time between failures, latency, etc.

2.2 High-Performance Algorithms for Space Exploration (UNIGE)

2.2.1 Pilot Description

Gaia Data Processing Centre in Geneva (DPCG) and Coordination Unit 7 (CU7) based at Observatory of Geneva - part of University of Geneva (UNIGE) - are responsible for variability studies of the nearly 2.7 billion sources that the ESA Gaia mission observes. The main task is to classify and characterize variable stellar objects/sources based on their time series from five products and three domains: photometry, spectroscopy, and radial velocities. Gaia is the first Petabyte scale astronomical ESA mission and operates on one of the biggest astronomical datasets up to date. The amount and complexity of the data as well as the number of angles from which the data is analyzed, poses a plethora of performance challenges in the Big Data and HPC domains.

2.2.2 Architecture

The DPCG Integrated Variability Pipeline (IVP) is a software stack composed of dozens of modules, embedded in the VariFramework for large-scale HPC processing and for Big Data aspects. The most

[1] https://www.artillery.io

time-consuming task of IVP is the *Period Search*, which is mostly based on various Fourier-transformation techniques that operate in the frequency domain and some advanced modelling for specific types of stars. Currently, CU7 cannot look for periodic signals for all the stars from the operational dataset, as a calculation of a single period might take up to a second; a prohibitive number if run on a medium sized-cluster for over 2.5 billion stars.

In the context of AERO, we aim to overcome this limitation by taking advantage of GPU integration in our algorithms. Figure 2 highlights in red, the placement of the main AERO dependency for IVP.

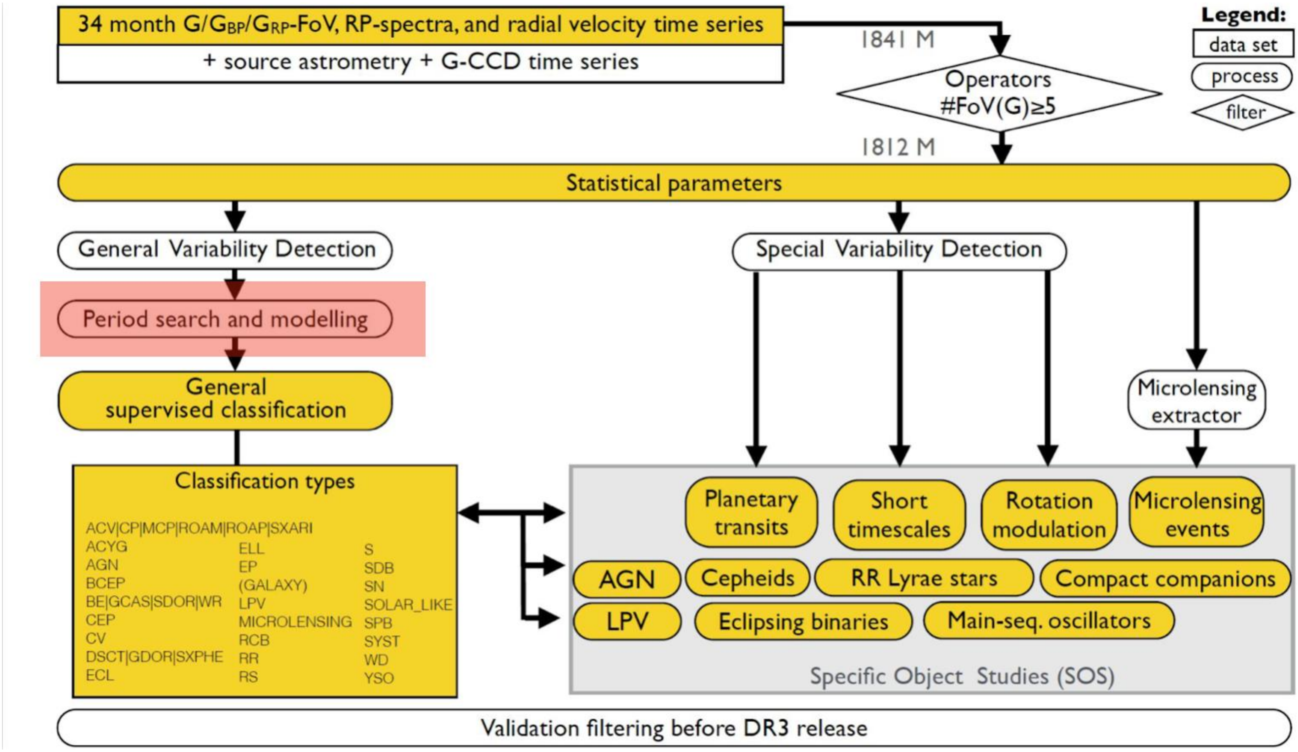

Figure 2 The Workflow of the VariCharacterization Module within IVP

Allowing the *Period Search* task to be run on all the sources is crucial to allow a deeper scientific understanding of non-variable stars, which might then allow the creation of better machine-learning models based on supervised training. Furthermore, if UNIGE manages to fit period-search of all sources (vs 20% currently) into the processing window of two-three months, it might also merge the Global Variability Detection module into the General Supervised Classification. Such a merge would simplify the pipeline and allow for a more robust scientific output, as each of the sources would go under much stricter algorithmic scrutiny - something that is currently impossible.

The integration of the GPU-enabled algorithms might also trigger the refactoring of VariFramework to switch from embarrassingly-parallel processing to a scheduled mode, where data is aggregated to be executed on the scarce-shared resource, i.e., the accelerator, when needed, and also computed on the CPUs, so that proper utilization is achieved without introducing new bottlenecks.

2.2.3 Software Components

The baseline software components of this pilot are the following:

- The VariFramework stack (around 1M lines of code) based on Java 17 LTS.

- The VariPeriodSearch module with a dozen period-search methods for sparse time-series.
- TornadoVM to enable GPU execution and acceleration.
- The dependency list of Vari software is quite big and is presented in Appendix I. It includes around 400 open-source Java modules, including
  - Apache Camel - implementation of Enterprise Integration Patterns for loose coupling integration of modules, processing and storage.
  - Apache Commons Math - for modeling and optimization implementations in Java
  - Active MQ - for JMS implementation for loosely coupled modules for processing
  - Apache OpenJPA - JPA implementation with in-house extensions to deal with Big Data
  - Spring and Spring Boot frameworks - for Inversion of Control patterns
  - H2O - supervised classification framework, for ML training and inference.

- The DPCG/SED led Database hosted at UNIGE.

2.2.4 AERO Integration Components

The main hook in the scope of AERO for UNIGE is the TornadoVM framework provided by UNIMAN. It will be leveraged to enable GPU capabilities in Java first in the VariPeriodSearch module, and then in chosen Special Object Studies modules (most likely Planets and Microlensing).

2.2.5 Deployment Strategy, Evaluation & KPIs

As UNIGE's primary goal is to enable GPU acceleration using TornadoVM, it has decided to deploy first on the Gaia dedicated x86-based cluster, which comprises 1080 CPU cores and 135 embedded Intel GPUs. It will then proceed to using Nvidia H100 and A100 on the UNIGE Yggdrasil cluster targeting possibly Gaia Data Release 4 and surely Data Release 5. Once the correctness of algorithms when using various accelerators (embedded and discrete GPUs) and backends (OpenCL, SPIR-V) is confirmed, the code will be deployed to the Rhea platform (or other AERO alternative testbeds if Rhea is not available).

Additionally, UNIGE aspires to deploy the code on the GPU-accelerated MPP database developed by SED (Section 2.3) via embedding VariFramework in the DB using TornadoVM embedded in PL/Java.

For the evaluation, UNIGE has several period-search and astro-dedicated algorithms implemented in Java to be executed on CPUs. These algorithms will serve as the baseline for both accuracy and performance for the GPU implementations based on TornadoVM:

- UNIGE will assess the algorithms' performance both in isolated unit and integration tests deployed at scale running on millions of astro-sources/time-series. As explained before, tests and production runs will be conducted initially on an x86-based cluster containing GPUs; tests on the Rhea platform will be performed at a later stage.
- As the algorithms will be refactored to be rebased with Float numerical precision in order to achieve greater speedup, UNIGE will evaluate their scientific and mathematical soundness.

The primary KPI of this pilot will be the achieved speedup in the execution of *Period Search* when taking advantage of GPU acceleration leveraging TornadoVM compared to utilizing a single node with 8 CPU cores through embarrassingly parallel CPU computation. A speedup greater than 15x will be considered a success. Such speedup will enable the processing of more, and significantly longer, time-series (G-band) than in the latest Gaia Data Release within the same processing window.

A secondary KPI, strongly related to the previous one, is whether the speedup can be achieved with GPU-based processing while preserving the level of numerical accuracy achieved when processing is performed on CPUs.

Finally, another measure of success will be successfully enabling *Period Search* on spectral time-series which require 60x more data, as well as on Radial Velocity and astrometric (positional) time-series. Such a success will enable new scientific products to be published in subsequent Gaia public Data Releases. In total, around 500TB+ data might be processed using components based on the AERO stack multiple times during the cyclic operations.

2.3 HPC/Cloud Database Acceleration for Scientific Computing (SED)

2.3.1 Pilot Description

SED is the provider of a petabyte-scale MPP database based on PostgreSQL for the Gaia Variability studies of UNIGE. Our MPP database is based on TBase[2], a fork of Postgres-XL by the Chinese company Tencent[3] with improved stability and scalability. While Tencent kept the BSD license, both the source code and license have diverged significantly from the original open-source driven non-MPP PostgreSQL. The idea behind this pilot is two-fold:

- To preserve and expand the European-based know-how and open-source ecosystem of MPP databases, in particular for scientific data-intensive applications. This will be achieved through turning the stand-alone TBase fork into a dynamically loadable extension of the original PostgreSQL. We will refer to this new extension as PG-XZ.
- To exploit the accelerators' capabilities in the open source parallel and distributed PostgreSQL leveraging the AERO stack for:
    - User Defined Functions using GPU acceleration
    - DB planner/executor using GPU acceleration.

2.3.2 Architecture

Currently, SED drives the development and usage of the TBase/Postgres-XL based on the Gaia Variability use case to operate petabyte-scale for time-series in the scientific context. The enhancement of the DB will pertain to enabling GPU query-processing using Level Zero and embedding GPU functionality for time-series analysis using TornadoVM with embedded Java as User Defined Functions in the DB, based on PL/Java - an embedded Java engine within the DB to allow User Defined Functions (UDFs) to be executed close to the DB core.

---

[2] https://github.com/Tencent/TBase
[3] https://www.tencent.com/

Figure 3 depicts two levels of the envisioned GPU acceleration: created by users via User Defined Functions (UDFs) in DB-embedded Java (PL/Java) and the planner and executor automatically converting SQL queries to GPU-executed distributed plans (marked in blue in Figure 3).

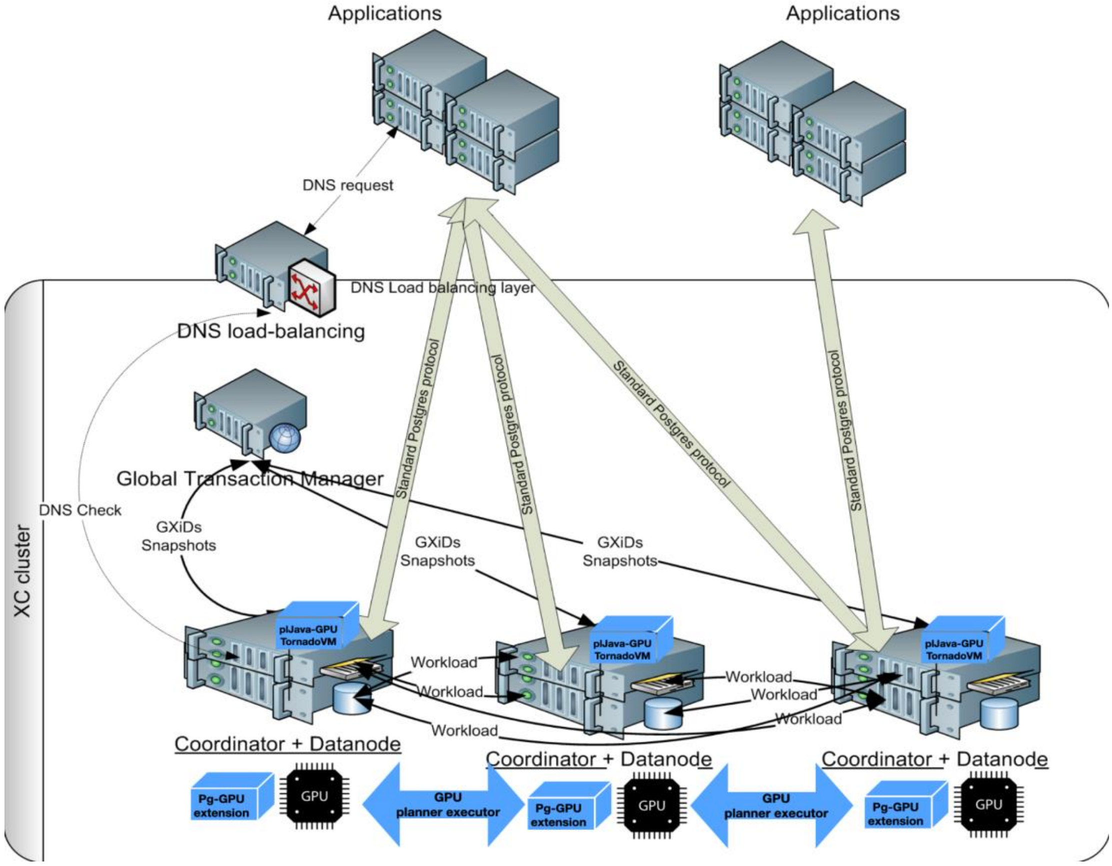

Figure 3 Two levels of GPU-enhanced MPP Postgres cluster: SQL planner and executor and GPU-enabled User Defined Functions in PL/Java+TornadoVM

2.3.3 Software Components

The main software components of interest to this pilot are the following:

- Linux (Alma Linux 9)
- TBase/Postgres-XL software stack: Around 1.3M Lines of Code, out of which ~5% is estimated to be affected if pushed to the PG extension. SED will assess in parallel the possibility of converting TBase to extension. This will allow for much wider adoption, as extensions are de-facto standards to enable new features in Postgres and would avoid stalling development as merging MPP code with vanilla PG has been proven time consuming. This would also enable new PG features for free, if not related to MPP functionality, i.e., sort speedups, new index types, new client functionality, vertical scalability improvements, new open platforms (including RISC-V) etc.
- Postgres extensions, including at least:
  - q3c[4] for spherical queries.
  - Apache Data Sketches[5] for sketches/aggregations.
- PL/Java for TornadoVM of UNIGE to deploy specific algorithms to DB as Java UDFs.

[4] https://github.com/segasai/q3c

[5] https://datasketches.apache.org/

- VariFramework components/ESA Gaia software stack to embed in the DB.

2.3.4 AERO Integration Components

The following main integration components have been identified:

- TornadoVM will enable GPU access using PL/Java extension at the application level (time-series algorithm embedded in the DB).
- The Intel oneAPI Level Zero API will be used for the development of planner/executor.
- The Intel oneAPI Level Zero API will be the main API to integrate the DB engine with GPUs. SED plans to port the PG-Strom Postgres extension (currently CUDA-based only) to the Level Zero API and integrate it into the parallel database.

2.3.5 Deployment Strategy, Evaluation & KPIs

As the primary goal of this pilot is the integration of its software stack with TornadoVM, it will follow a deployment strategy like the one presented in Section 2.2.5 (UNIGE pilot). More specifically, the deployment will be performed initially on an x86-based cluster using GPUs (a cluster comprising AMD EPYC CPUs and Nvidia A100 GPUs is expected to be available by M12 of the project), before moving to the Rhea platform or other AERO alternative testbeds.

Both SED and UNIGE work in the greater frame of the Gaia mission operations that define their main activities. Taking these into account, in the context of AERO there are two main activities:

1. Cyclic operations for Gaia Data Release 4, starting in September 2023 (M9 of the project) and finishing in early 2025. The target is to achieve GPU functionality at least via PL/Java and TornadoVM to enable GPU-accelerated functions directly in the DB (VariPeriod search, unsupervised classification).
2. Cyclic operations for Gaia Data Release 5, starting in late 2025, i.e., at the last stages of the AERO project. The target is to achieve GPU integration of the DB engine, with PG-Strom ported to the Level Zero API, enabling SQL plans to be executed on GPU.

For the evaluation of the first phase, existing crucial functions deployed in DB will be ported to TornadoVM based on embedded PL/Java. This is expected to lead to a considerable speedup for several operational activities, such as timeseries-reconstruction and deployment for ad-hoc period-search algorithms to enable quick experimentation also with hybrid supervised/unsupervised classification within the DB.

The second phase, which includes the extension of the DB engine in order to enable the DB planner and executor to leverage the GPU, will be evaluated based on the achieved speedup. Specifically, a certain number of analytical queries used by scientists in the Gaia results validation are expected to be executed considerably faster.

In general, the main KPIs are the accuracy, stability and performance of the same queries run on the current and the enhanced DB. Similar to standard OLAP/OLTP DB benchmarks, performance metrics will be obtained on specific hardware using specific real-life datasets coming from the Gaia mission. The expected performance gains for some of the queries are in the range of 2-20x while no performance degradation is expected for those that cannot profit from the acceleration.

3 Summary

This deliverable has presented the AERO pilots, how they are expected to be integrated with the AERO software stack and deployed on the AERO testbeds. Further, it has defined their evaluation strategy together with specific KPIs. The following table summarizes the key takeaways for the three AERO pilots.

Pilot partner	KTM	UNIGE	SED
Pilot	Automotive Digital Twin	High-Performance Algorithms for Space Exploration	HPC/Cloud Database Acceleration for Scientific Computing
Brief Description	Vehicle Information Service	Processing and analytics of astronomical time-series data	MPP relational databases using hardware accelerators (GPUs)
Software Components	Docker, NodeJS, TypeScript, MongoDB, Apache Kafka, Zookeeper	VariFramework stack	TBase/Postgres-XL software stack, PL/Java, VariFramework components, ESA Gaia software stack
AERO Integration Components	Docker, Apache Kafka	TornadoVM	TornadoVM, Intel oneAPI Level Zero API
Deployment Strategy	Docker containers. Initial testing on Microsoft Azure ARM instances until Rhea platform becomes available.	Initial deployment on dedicated cluster with x86 CPUs, Intel embedded GPUs and Nvidia H100 and A100 GPUs. Once integration with TornadoVM is completed, deployment on Rhea or other AERO testbeds will be attempted.	
Evaluation	Employment of a system integration testing methodology. Behavior of application will be assessed whether it is similar on x86 and ARM cores.	Compare the accuracy and performance of the Java algorithms that run currently on CPU against the GPU-accelerated implementations that will be developed in AERO via TornadoVM.	Split in two phases: 1) Successful porting of DB functions in TornadoVM, and 2) successful extension of the DB engine using the Level Zero API.
KPI(s)	100% pass-rate across all test suites on the Rhea platform. Satisfy current SLAs on Rhea similar to x86 platforms. Monitored metrics: Transactions per second, mean time between failures, latency, etc.	Performance: >=15x performance speedup with GPU-accelerated code. Achieving speedup with GPU-based processing while preserving the level of numerical accuracy achieved with CPU-based processing. Successfully enabling spectral time-series, Radial Velocity and astrometric (positional) time-series.	Performance: >=2x performance speedup leveraging GPU acceleration. Achieve similar accuracy when queries are run on the current and the enhanced DB.

Appendix I: UNIGE IVP dependency list

As a reference to the size of the dependency list of Gaia Integrated Variability Pipeline (IVP), we include a list of packages as of Q2 2023. The total size of dependencies is around 330MB. The successful pilot execution entails testing a large portion of functionality included in the modules listed below. The modules starting with Vari*, SVD*, AGIS*, Gaia* are Gaia DPAC modules. The rest are off the shelf, open-source components.

Component	Size	Component	Size
activemq-broker-5.16.4.jar	1.6M	activemq-client-5.16.4.jar	1.8M
activemq-jms-pool-5.16.4.jar	256K	activemq-openwire-legacy-5.16.4.jar	1.1M
activemq-pool-5.16.4.jar	128K	activemq-spring-5.16.4.jar	584K
AGISDm-17.1.0.jar	672K	AGISLab-17.1.0.jar	6.3M
AGISTools-17.1.0.jar	960K	all-1.1.2.pom	32K
ant-1.9.6.jar	2.6M	ant-launcher-1.9.6.jar	128K
antlr-2.7.7.jar	856K	antlr-runtime-3.5.2.jar	584K
apiguardian-api-1.1.2.jar	32K	arpack_combined_all-0.1.jar	7.1M
arpack_combined_all-0.1-javadoc.jar	7.1M	aspectjweaver-1.9.7.jar	2.8M
assertj-core-3.22.0.jar	6.2M	bounce-0.18.jar	624K
byte-buddy-1.12.9.jar	4.5M	byte-buddy-agent-1.11.13.jar	664K
camel-activemq-3.16.0.jar	288K	camel-api-3.16.0.jar	928K
camel-base-3.16.0.jar	584K	camel-base-engine-3.16.0.jar	904K
camel-bean-3.16.0.jar	568K	camel-bean-starter-3.16.0.jar	128K
camel-browse-3.16.0.jar	128K	camel-browse-starter-3.16.0.jar	96K
camel-cloud-3.16.0.jar	384K	camel-cluster-3.16.0.jar	160K
camel-controlbus-3.16.0.jar	160K	camel-controlbus-starter-3.16.0.jar	96K
camel-core-3.16.0.jar	32K	camel-core-engine-3.16.0.jar	480K
camel-core-languages-3.16.0.jar	616K	camel-core-model-3.16.0.jar	1.5M
camel-core-processor-3.16.0.jar	872K	camel-core-reifier-3.16.0.jar	672K
camel-core-starter-3.16.0.jar	288K	camel-core-xml-3.16.0.jar	416K
camel-dataformat-3.16.0.jar	128K	camel-dataformat-starter-3.16.0.jar	96K
camel-dataset-3.16.0.jar	256K	camel-dataset-starter-3.16.0.jar	128K
camel-direct-3.16.0.jar	160K	camel-direct-starter-3.16.0.jar	96K
camel-directvm-3.16.0.jar	192K	camel-directvm-starter-3.16.0.jar	96K
camel-dsl-support-3.16.0.jar	96K	camel-file-3.16.0.jar	632K
camel-file-starter-3.16.0.jar	128K	camel-health-3.16.0.jar	192K
camel-jms-3.16.0.jar	664K	camel-jmx-3.16.0.jar	320K
camel-language-3.16.0.jar	160K	camel-language-starter-3.16.0.jar	96K
camel-log-3.16.0.jar	192K	camel-log-starter-3.16.0.jar	96K
camel-main-3.16.0.jar	656K	camel-management-3.16.0.jar	680K
camel-management-api-3.16.0.jar	544K	camel-metrics-3.16.0.jar	288K

camel-mock-3.16.0.jar	448K	camel-mock-starter-3.16.0.jar	96K
camel-ref-3.16.0.jar	128K	camel-ref-starter-3.16.0.jar	96K
camel-rest-3.16.0.jar	320K	camel-rest-starter-3.16.0.jar	128K
camel-saga-3.16.0.jar	128K	camel-saga-starter-3.16.0.jar	96K
camel-scheduler-3.16.0.jar	160K	camel-scheduler-starter-3.16.0.jar	96K
camel-seda-3.16.0.jar	256K	camel-seda-starter-3.16.0.jar	96K
camel-spring-3.16.0.jar	416K	camel-spring-boot-3.16.0.jar	568K
camel-spring-boot-starter-3.16.0.jar	32K	camel-spring-main-3.16.0.jar	128K
camel-spring-xml-3.16.0.jar	680K	camel-stub-3.16.0.jar	128K
camel-stub-starter-3.16.0.jar	96K	camel-support-3.16.0.jar	1.2M
camel-test-3.16.0.jar	224K	camel-test-junit5-3.16.0.jar	288K
camel-test-spring-3.16.0.jar	256K	camel-test-spring-junit5-3.16.0.jar	288K
camel-timer-3.16.0.jar	160K	camel-timer-starter-3.16.0.jar	96K
camel-tooling-model-3.16.0.jar	224K	camel-util-3.16.0.jar	600K
camel-util-json-3.16.0.jar	192K	camel-validator-3.16.0.jar	160K
camel-validator-starter-3.16.0.jar	96K	camel-vm-3.16.0.jar	128K
camel-vm-starter-3.16.0.jar	96K	camel-xml-io-util-3.16.0.jar	128K
camel-xml-jaxb-3.16.0.jar	160K	camel-xml-jaxb-dsl-3.16.0.jar	96K
camel-xml-jaxp-3.16.0.jar	512K	camel-xml-jaxp-starter-3.16.0.jar	96K
camel-xpath-3.16.0.jar	224K	camel-xpath-starter-3.16.0.jar	96K
camel-xslt-3.16.0.jar	288K	camel-xslt-starter-3.16.0.jar	96K
checker-qual-3.31.0.jar	640K	classmate-1.5.1.jar	320K
cloning-1.9.12.jar	160K	common-3.6.jar	448K
commons-beanutils-1.9.3.jar	664K	commons-codec-1.11.jar	744K
commons-collections-3.2.1.jar	984K	commons-collections4-4.4.jar	1.2M
commons-compress-1.19.jar	1.0M	commons-configuration-1.7.jar	760K
commons-dbcp-1.4.jar	576K	commons-digester-1.8.1.jar	560K
commons-io-2.6.jar	632K	commons-lang-2.6.jar	696K
commons-lang3-3.12.0.jar	992K	commons-logging-1.2.jar	288K
commons-math3-3.6.1.jar	2.9M	commons-pool-1.6.jar	480K
commons-pool2-2.11.1.jar	560K	commons-rng-client-api-1.3.jar	96K
commons-rng-core-1.3.jar	384K	commons-rng-sampling-1.3.jar	416K
commons-rng-simple-1.3.jar	224K	converter-gson-2.4.0.jar	32K
core-1.1.2.jar	584K	disruptor-3.4.2.jar	384K
dozer-5.4.0.jar	664K	duke-1.2.jar	688K
ejb-api-3.0-alpha-1.jar	224K	ejml-all-0.34.jar	32K
ejml-cdense-0.34.jar	352K	ejml-core-0.34.jar	584K
ejml-ddense-0.34.jar	736K	ejml-dsparse-0.34.jar	288K
ejml-fdense-0.34.jar	728K	ejml-simple-0.34.jar	584K
ejml-zdense-0.34.jar	352K	error_prone_annotations-2.11.0.jar	96K
failureaccess-1.0.1.jar	32K	flanagan-1.0.jar	1.6M
fluent-hc-4.5.9.jar	160K	freehep-io-2.0.5.jar	320K
fst-2.57.jar	808K	GaiaMdbDm-20.0.18.jar	8.6M

GaiaParameters-21.1.0.jar	1.2M	GaiaTools-20.6.1.jar	4.3M
GaiaToolsDm-21.3.0.jar	1.1M	geojson-jackson-1.5.1.jar	128K
geronimo-j2ee-management_1.1_spec-1.0.1.jar	128K	geronimo-jms_1.1_spec-1.1.1.jar	160K
geronimo-jms_2.0_spec-1.0-alpha-2.jar	224K	geronimo-jta_1.1_spec-1.1.1.jar	96K
groovy-4.0.5.jar	8.6M	groovy-jsr223-4.0.5.jar	128K
gson-2.9.0.jar	664K	guava-31.1-jre.jar	3.7M
h2o-algos-3.30.0.1.jar	1.7M	h2o-automl-3.30.0.1.pom	32K
h2o-bindings-3.30.0.1.pom	32K	h2o-core-3.30.0.1.jar	5.2M
h2o-ext-mojo-pipeline-3.30.0.1.pom	32K	h2o-genmodel-3.30.0.1.jar	736K
h2o-genmodel-ext-xgboost-3.30.0.1.jar	128K	h2o-jaas-pam-3.30.0.1.jar	32K
h2o-logger-3.30.0.1.jar	32K	h2o-tree-api-0.3.15.jar	32K
h2o-webserver-iface-3.30.0.1.jar	64K	hamcrest-2.2.jar	544K
hamcrest-all-1.3.jar	720K	hamcrest-core-1.3.jar	224K
hamcrest-library-1.3.jar	256K	hawtbuf-1.11.jar	256K
hibernate-commons-annotations-5.1.2.Final.jar	352K	hibernate-core-5.6.8.Final.jar	8.7M
HikariCP-4.0.3.jar	576K	htmlIndex-3.0.2.jar	728K
httpclient-4.5.9.jar	1.2M	httpclient-cache-4.5.9.jar	584K
httpcore-4.4.11.jar	736K	httpmime-4.5.9.jar	224K
interval-tree-1.0.0.jar	128K	istack-commons-runtime-4.0.0.jar	160K
itext-2.0.1.jar	2.2M	j2objc-annotations-1.3.jar	96K
j3d-core-1.3.1.jar	3.2M	j3d-core-utils-1.3.1.jar	1.8M
jackson-annotations-2.15.2.jar	352K	jackson-core-2.15.2.jar	960K
jackson-databind-2.15.2.jar	2.0M	jakarta.activation-2.0.0.jar	288K
jakarta.annotation-api-1.3.5.jar	160K	jakarta.persistence-api-2.2.3.jar	584K
jakarta.transaction-api-1.3.3.jar	96K	jakarta.xml.bind-api-3.0.0.jar	544K
jama-1.0.3.jar	192K	jandex-2.4.2.Final.jar	648K
java-cup-11b-2015.03.26.jar	416K	java-cup-11b-runtime-2015.03.26.jar	128K
javaparser-1.0.11.jar	704K	javassist-3.25.0-GA.jar	1.2M
javax.activation-1.2.0.jar	352K	javax.activation-api-1.2.0.jar	256K
javax.annotation-api-1.3.2.jar	160K	javax.persistence-2.1.0.jar	576K
javax.servlet-api-4.0.1.jar	416K	jaxb2-basics-1.11.1.jar	560K
jaxb2-basics-ant-1.11.1.jar	32K	jaxb2-basics-runtime-1.11.1.jar	584K
jaxb2-basics-tools-1.11.1.jar	560K	jaxb2-default-value-1.1.jar	64K
jaxb2-value-constructor-3.0.jar	32K	jaxb-api-2.1.9.jar	448K
jaxb-api-2.3.1.jar	544K	jaxb-core-2.3.0.1.jar	672K
jaxb-core-3.0.0.jar	560K	jaxb-impl-2.4.0-b180830.0438.jar	1.5M
jaxb-runtime-3.0.0.jar	1.3M	jaxen-1.0-FCS.jar	608K
jaxrpc-api-1.1.jar	160K	jboss-serialization-4.2.2.GA.jar	544K
jcl-over-slf4j-1.7.7.jar	128K	jcommon-1.0.23latex.jar	736K
jdom-1.0.jar	568K	jfreechart-1.0.10.jar	1.7M
jfreechart-1.5.0.jar	1.9M	jfreechartbinding-0.0.6.jar	544K
jhealpix-3.2.0.jar	968K	JLargeArrays-1.5.jar	648K

jmathtex-0.7pre.jar	648K	jmh-core-1.21.jar	920K
jmotif-0.97.jar	736K	jmotif-isax-0.97.jar	584K
jmx_prometheus_javaagent-0.17.0.jar	944K	jna-4.0.0.jar	1.3M
jniloader-1.1.jar	96K	joda-time-2.9.9.jar	1.1M
jolokia-jvm-agent-1.6.2.jar	872K	jopt-simple-4.6.jar	288K
jsap-2.1.jar	320K	jsr305-3.0.2.jar	128K
JTransforms-3.1.jar	1.6M	junit-4.13.2.jar	792K
junit-addons-1.4.jar	256K	junit-jupiter-api-5.8.2.jar	608K
junit-jupiter-engine-5.8.2.jar	648K	junit-jupiter-params-5.8.2.jar	984K
junit-platform-commons-1.8.2.jar	448K	junit-platform-engine-1.8.2.jar	600K
krasa-jaxb-tools-1.4.jar	128K	kryo-5.1.1.jar	768K
libpam4j-1.8.jar	128K	listenablefuture-9999.0-empty-to-avoid-conflict-with-guava.jar	32K
log4j2-logstash-layout-1.0.5.jar	480K	log4j-api-2.17.2.jar	712K
log4j-core-2.17.2.jar	2.2M	log4j-jul-2.17.2.jar	160K
log4j-slf4j-impl-2.17.2.jar	128K	lombok-1.18.24.jar	2.4M
lz4-1.3.0.jar	648K	lz4-java-1.4.0.jar	784K
management-api-1.1-rev-1.jar	160K	metrics-core-4.2.7.jar	544K
metrics-jmx-4.2.7.jar	128K	metrics-json-4.2.7.jar	128K
minlog-1.3.1.jar	32K	mockito-core-3.12.4.jar	1.1M
mojo2-runtime-api-0.13.7.jar	256K	mtj-1.0.4.jar	680K
mvel2-2.4.12.Final.jar	1.2M	native_ref-java-1.1.jar	256K
native_system-java-1.1.jar	256K	net-ivoa-fits-0.1.jar	600K
netlib-java-1.1.jar	416K	netlib-native_ref-linux-armhf-1.1.jar	1.6M
netlib-native_ref-linux-armhf-1.1-natives.jar	1.6M	netlib-native_ref-linux-i686-1.1.jar	1.9M
netlib-native_ref-linux-i686-1.1-natives.jar	1.9M	netlib-native_ref-linux-x86_64-1.1.jar	2.1M
netlib-native_ref-linux-x86_64-1.1-natives.jar	2.1M	netlib-native_ref-osx-x86_64-1.1.jar	2.2M
netlib-native_ref-osx-x86_64-1.1-natives.jar	2.2M	netlib-native_ref-win-i686-1.1.jar	2.2M
netlib-native_ref-win-i686-1.1-natives.jar	2.2M	netlib-native_ref-win-x86_64-1.1.jar	3.0M
netlib-native_ref-win-x86_64-1.1-natives.jar	3.0M	netlib-native_system-linux-armhf-1.1.jar	720K
netlib-native_system-linux-armhf-1.1-natives.jar	720K	netlib-native_system-linux-i686-1.1.jar	848K
netlib-native_system-linux-i686-1.1-natives.jar	848K	netlib-native_system-linux-x86_64-1.1.jar	864K
netlib-native_system-linux-x86_64-1.1-natives.jar	864K	netlib-native_system-osx-x86_64-1.1.jar	960K
netlib-native_system-osx-x86_64-1.1-natives.jar	960K	netlib-native_system-win-i686-1.1.jar	968K
netlib-native_system-win-i686-1.1-natives.jar	968K	netlib-native_system-win-x86_64-1.1.jar	1.1M

netlib-native_system-win-x86_64-1.1-natives.jar	1.1M	numericalmethods-1.0.jar	656K
objenesis-3.2.jar	256K	ognl-2.6.9.jar	584K
okhttp-3.10.0.jar	824K	okio-1.14.0.jar	384K
opencsv-2.3.jar	128K	openjpa-all-3.2.2-CU7.jar	7.7M
opentest4j-1.2.0.jar	32K	operator-22.1.3.jar	648K
org.apache.bval.bundle-1.1.2.jar	904K	postgresql-42.6.0.jar	1.5M
preferences-3.6.jar	480K	reflectasm-1.11.9.jar	320K
retrofit-2.4.0.jar	416K	rxjava-1.2.0.jar	1.5M
saxpath-1.0-FCS.jar	128K	simple-5.1.6.jar	664K
slf4j-api-1.7.36.jar	224K	snakeyaml-1.30.jar	744K
SOS_AGN-SB-22.1.0-r763042M-20230322163342.jar	640K	SOS_BlackHoles-SB-22.1.0-r766975M-20230608151118.jar	512K
SOS_CepheidAndRRLyrae-SB-22.1.0-r767271-20230608152241.jar	1.1M	SOS_EclipsingBinaries-SB-22.1.0-r768413M-20230608153341.jar	1.7M
SOS_FlaringAndRotationalModulation-SB-22.1.0-r769585M-20230608164703.jar	872K	SOS_LPV-SB-22.1.0-r769578M-20230608160620.jar	680K
SOS_Microlensing-SB-GJ_22.1.0-r752441M-20220922134308.jar	784K	SOS_PlanetaryTransits-22.1.0.jar	592K
spring-aop-5.3.19.jar	792K	spring-aspects-5.3.19.jar	224K
spring-beans-5.3.19.jar	1.1M	spring-boot-2.6.7.jar	1.8M
spring-boot-autoconfigure-2.6.7.jar	2.0M	spring-boot-starter-2.6.7.jar	32K
spring-boot-starter-aop-2.6.7.jar	32K	spring-boot-starter-data-jpa-2.6.7.jar	32K
spring-boot-starter-jdbc-2.6.7.jar	32K	spring-boot-test-2.6.7.jar	640K
spring-context-5.3.19.jar	1.7M	spring-core-5.3.19.jar	1.9M
spring-data-commons-2.6.4.jar	1.7M	spring-data-jpa-2.6.4.jar	784K
spring-expression-5.3.19.jar	704K	spring-jcl-5.3.19.jar	128K
spring-jdbc-5.3.19.jar	840K	spring-jms-5.3.16.jar	680K
spring-messaging-5.3.16.jar	976K	spring-orm-5.3.19.jar	616K
spring-test-5.3.19.jar	1.2M	spring-tx-5.3.19.jar	744K
stringtemplate-3.2.1.jar	568K	super-csv-2.3.1.jar	416K
super-csv-dozer-2.3.1.jar	96K	SVD_CombineDetection-SB-22.1.0-r752443-20230608132815.jar	288K
SVD_PlanetaryTransits-SB-22.1.0-r769183M-20230608133535.jar	848K	SVD_ShortTimeScale-SB-22.1.0-r769233M-20230608133947.jar	704K
SVD_SolarLike-SB-22.1.0-r752446M-20230503175741.jar	992K	Taglets-1.0.jar	160K
text-3.5.jar	664K	tornado-api-0.15.1-dev-dev.jar	664K
tornado-api-0.15.1-dev.jar	664K	tornado-matrices-0.15.1-dev-dev.jar	96K
tornado-matrices-0.15.1-dev.jar	96K	transaction-api-1.1.jar	96K
txw2-3.0.0.jar	320K	unitils-core-3.4.6.jar	608K

validation-api-1.1.0.Final.jar	288K	VariCharacterisation-SB-22.2.0-r766896M-20230608132529.jar	1008K
VariClassification-SB-22.1.0-r765167M-20230418135919.jar	1.6M	VariConfiguration-SB-22.3.0-r768498-20230526083739.jar	3.4M
VariFramework-SB-22.3.0-r769095M-20230608093113.jar	1.1M	VariGeneralDetection-SB-22.1.0-r766967M-20230530222247.jar	968K
VariLcModels-13.0.0.jar	824K	VariModelling-SB-22.2.0-r766899M-20230608105318.jar	1.2M
VariObjectModelInterface-22.1.2.jar	160K	VariObjectModel-SB-22.3.0-r769141M-20230608114920.jar	6.9M
VariPeriodSearch-SB-22.2.0-r769520M-20230608114654.jar	2.7M	VariPostTakerValidation-SB-21.3.0-r760698-20230127132659.jar	448K
VariStatistics-SB-22.2.0-r769391M-20230608095826.jar	1.2M	vecmath-1.3.1.jar	704K
weka-stable-3.8.1.jar	11M	xbean-spring-4.17.jar	552K
xgboost-predictor-0.3.15.jar	192K	xmlpull-1.1.3.1.jar	32K
xmlunit-1.4.jar	448K	xpp3_min-1.1.4c.jar	160K
xstream-1.4.11.1.jar	1.0M	xz-1.6.jar	448K
zstd-jni-1.5.2-3.jar	6.4M		