Funded by
the European Union

HORIZON EUROPE FRAMEWORK PROGRAMME

NEARDATA

(grant agreement No 101092644)

Extreme Near-Data Processing Platform

D2.3 Reference implementation of architectural building blocks

Due date of deliverable: 31-10-2025
Actual submission date: 31-10-2025

Start date of project: 01-01-2023 Duration: 36 months

Summary of the document

Document Type	Report
Dissemination level	Public
State	v2.0
Number of pages	53
WP/Task related to this document	WP2 / T2.3
WP/Task responsible	URV
Leader	Pedro Garcia Lopez (URV)
Technical Manager	Albert Cañadilla (URV)
Quality Manager	Raúl Gracia (DELL)
Author(s)	Pedro Garcia Lopez (URV), Raúl Gracia (DELL), Alan Cueva (DELL), Hossam Elghamry (DELL), Sean Ahearne (DELL), Ger Hallissey (DELL), Aaron Call (BSC), André Miguel (SCONTAIN), Maciej Malawski (SANO), Paolo Ribeca (UKHSA), Sebastian Bodenstedt (NCT), Kudamaduwage Pubudu Nuwanthika (TUD)
Partner(s) Contributing	URV, DELL, BSC, SCO, SANO, UKHSA, NCT, TUD
Document ID	NEARDATA_D2.3_Public.pdf
Abstract	Public release of stable software components of the NEARDATA architecture. Complete specification and APIs. Final description and evaluation of results obtained from validation in use cases using different workloads.
Keywords	Reference architecture, Software outcomes

History of changes

Version	Date	Author	Summary of changes
0.1	15-9-2025	Pedro Garcia lopez, Albert Cañadilla	First draft.
0.2	27-10-2025	Pedro Garcia Lopez, Albert Cañadilla	Review feedback.
1.0	27-10-2025	Pedro Garcia Lopez, Albert Cañadilla	Final version.

Table of Contents

List of Abbreviations and Acronyms

AI	Artificial Intelligence
API	Application Programming Interface
BSC	Barcelona Supercomputing Center
CAS	Configuration and Attestation Service
CC	Creative Commons
CI/CD	Continuous Integration / Continuous Delivery
CIDR	Classless Inter-Domain Routing
CLI	Command Line Interface
CNCF	Cloud Native Computing Foundation
CNN	Convolutional Neural Netowork
COG	Cloud-Optimized GeoTIFF
COPC	Cloud-Optimized Point Cloud
CPU	Central Processing Unit
CSV	Comma-separated values
DB	DataBase
DL	Deep Learning
DOI	Digital Object Identifier
EBS	Elastic Block Store
ECS	Elastic Container Service
EMBL	European Molecular Biology Laboratory
ENA	European Nucleotide Archive
FaaS	Function as a Service
FPGA	Field-Programmable Gate Array
GPU	Graphics Processing Unit
HDFS	Hadoop Distributed File System
HPC	High-Performance Computing
I/O	Input/Output
IDE	Integrated Developing Environment
INSDC	International Nucleotide Sequence Database Collaboration
K8S	Kubernetes
KPI	Key Performance Indicator

LiDAR	Light Detection And Ranging
LLD	Last-Level Defence
LTS	Long-Term Storage
ML	Machine Learning
MPI	Message-Passing Interface
MSI	Mass Spectometry Imaging
NCBI	National Center for Biotechnology Information
NCT	National Center for Tumor Diseases
NIH	National Institutes of Heatlth
NVMe	Non-Volatile Memory Express
PoC	Proof of Concept
RAG	Retrieval-Augmented Generation
SaaS	Software as a Service
SDK	Software Development Kit
SSD	Solid State Drive
TEE	Trusted Execution Environments
UCSC	university of California Santa Cruz
UKSHA	UK Health Security Agency
URL	Uniform Resource Locator
URV	Universitat Rovira i Virgili
VM	Virtual Machine
VPC	Virtual Private Cloud
WAL	Write-Ahead Logging

Executive Summary

The **NEARDATA** project tackles the growing challenge of "extreme data"—massive, heterogeneous, and geographically distributed datasets encountered in domains such as spatial metabolomics, large-scale genomic sequencing, and real-time surgical video analytics. Traditional analytics workflows suffer from network bottlenecks, high latency, and suboptimal resource utilization due to the transfer of multi-terabyte datasets to centralized compute centers. **NEARDATA** reverses this paradigm by pushing computation as close to the data as possible, thereby minimizing data movement, reducing end-to-end latency, and maximizing overall throughput.

This deliverable (D2.3) marks the public release of stable software modules, provides complete API specifications, and presents a final description of the different software outcomes produced in the project. All experimental validation is now included in D3.2 (Extreme Data Hub), D4.2 (Data Broker), and D5.2 (Use Case Experiments). At its core, **NEARDATA** comprises the *Extreme Data Hub*, which intercepts standard S3 and streaming API calls to perform predicate pushdown, partition pruning, and incremental aggregation across cloud, edge, and HPC environments.

We categorize our main software outcomes into **production-ready components** (Lithops, SCONE, Pravega, Metaspace, PyRun, Dataplug, DataCockpit) and **research-project prototypes** (Burst, NEXUS, FaaStream , Glider, Lithops-HPC). Our software outcomes have validated their technical contributions in the different use cases, including: spatial metabolomics annotation, off-sample detection, serverless genomic variant calling, and real-time surgical analytics. As we show in D5.2, they demonstrate core KPIs of this project including substantial reductions in data egress/ingress, improved latency, and robust security guarantees.

Subsequent sections detail the reference architecture, component analyses, pipeline implementations, research activities, and final conclusions to guide iterative development and consortium deployment.

1 Introduction

The NEARDATA project addresses the growing challenges of managing and processing massive, heterogeneous, and geographically distributed datasets—collectively referred to as extreme data. These challenges are particularly evident in data-intensive scientific domains such as metabolomics imaging, genomic sequencing, and real-time surgical video analytics. Traditional analytical workflows typically involve transferring multi-terabyte datasets from remote object stores to centralized compute clusters, resulting in significant network bottlenecks, latency, and inefficient resource utilization. NEARDATA aims to reverse this paradigm by bringing computation closer to the data—enabling low-latency, near-data processing that minimizes movement, reduces energy consumption, and enhances overall system throughput.

At the architectural level, NEARDATA defines two complementary platform constructs, whose functional and experimental details are provided in related deliverables:

- **Extreme Data Hub** — described in Deliverable D3.2, provides high-performance near-data processing for extreme data types through a novel intermediary service between object storage and analytic platforms. It introduces type-aware, serverless data connectors that support partitioning, filtering, aggregation, and incremental querying across cloud, edge, and HPC infrastructures, enabling an elastic process-then-compute paradigm that significantly reduces data movement and improves throughput.
- **Confidential Data Broker** — described in Deliverable D4.2, provides secure, policy-driven orchestration for distributed data access and computation. It leverages Trusted Execution Environments (TEEs) and federated learning to ensure confidentiality, integrity, and fine-grained access control across federated and multi-cloud deployments.

It is important to note that the Extreme Data Hub and the Confidential Data Broker are not implemented as single monolithic components. Instead, they represent compositional architectures that can be instantiated using different combinations of software components developed within NEARDATA. Depending on the use case or deployment, these components can be assembled into fully functional Hubs or Brokers. This modular design enables flexibility, reuse, and sustainability across heterogeneous infrastructures.

Accordingly, this deliverable (D2.3) does not re-describe the functional architectures already detailed in D3.2 and D4.2. Instead, it focuses on the **software outcomes** produced in the project, documenting their readiness, interfaces, and integration within the NEARDATA architecture. These outcomes collectively enable the practical realization of the Extreme Data Hub and the Confidential Data Broker through interoperable components, services, and execution environments.

In particular, the **PyRun platform** serves as the primary operational front-end integrating many of these outcomes—such as Lithops, DataPlug, and DataCockpit—into a cohesive environment for data exploration, orchestration, and execution of AI and data pipelines. PyRun embodies the principles of the Extreme Data Hub by enabling users to launch near-data computations, manage runtimes, and reproduce experiments across the compute continuum without managing infrastructure.

The following sections describe in detail the software components developed within NEARDATA, grouped by maturity level:

- **Production-Ready Components:** Stable and widely adopted frameworks forming the operational core of NEARDATA, including Lithops (serverless functions near data), SCONE (confidential computing primitives), Pravega (elastic stream storage), and Metaspace (metadata catalog). PyRun (SaaS front-end for NEARDATA components), DataPlug (data connector library), and DataCockpit (metadata-driven orchestration dashboard) are also part of this category.
- **Research-Prototype Components:** Experimental frameworks extending NEARDATA's capabilities toward new paradigms of computation and orchestration. These include Burst Computing (a near-data, burst-parallel executor), Nexus (metadata-driven control plane for heterogeneous backends), FaaStream (serverless stream data connector), Glider (policy-driven data mover), Lithops-HPC (serverless execution on HPC systems), Flower-Lithops Simulation Backend, and FaaSTs (AI-assisted resource predictor for FaaS environments).

Subsequent sections present how these components can be combined to form reproducible data pipelines—such as metabolomics annotation, off-sample detection, genomic sequencing, and real-time surgical analytics—demonstrating their integration within the NEARDATA architecture. Through this consolidated view, Deliverable D2.3 provides a comprehensive overview of the project's software ecosystem, illustrating how its modular outcomes collectively enable the vision of near-data, confidential, and elastic data analytics.

2 Description of Software Components

2.1 Component Classification

The components are categorized into two groups based exclusively on their level of readiness:

- **Production-Ready Components**: Fully developed and thoroughly tested solutions that are ready for deployment.
- **Research-Project Components**: Experimental prototypes produced in scientific articles .

Table 1: Summary of NEARDATA Software Components and Tools

Component / Tool	Category	Brief Description
Lithops	Production-Ready	Serverless function execution framework deployed close to the data.
Scone	Production-Ready	Confidential computing framework enabling secure, policy-driven execution of containerized workloads in Trusted Execution Environments (TEEs) across multi-cloud infrastructures.
Pravega	Production-Ready	Scalable and elastic stream storage system.
Metaspace	Production-Ready	Metadata catalog designed for large, unstructured datasets.
PyRun	Production-Ready	Serverless Python studio that enables users to write, run, and scale data science or AI workloads effortlessly on their own cloud accounts without managing infrastructure.
DataPlug	Production-Project	A framework that enables efficient, read-only, cloud-aware partitioning of unstructured scientific data in object storage through dynamic, parallel, and metadata-driven data slicing for elastic workloads.
DataCockpit	Production-Project	An interactive widget that allows scientists to browse, partition, and benchmark datasets from Amazon S3 or Metaspace directly within Jupyter notebooks for elastic, parallel data processing.
Serverless Vector DB	Research-Project	Prototype of a serverless vector database enabling similarity search over large embeddings with elastic scaling.
Lithops-HPC	Research-Project	Lithops framework to run on HPC environments.
FaaStream	Research-Project	Experimental framework combining serverless execution with stream-based storage for low-latency, elastic pipelines.
Nexus	Research-Project	Data mesh prototype introducing streamlets and swarmlets to support metadata-driven orchestration of heterogeneous backends.
Glider	Research-Project	Serverless system that enables stateful near-data computation on ephemeral storage, drastically reducing data movement and improving the performance and efficiency of serverless data analytics pipelines.
FaaSTs	Research-Project	Experimental framework to predict resource usage on FaaS environments leveraging AI and time-series.
Burst Computing	Research-Project	Experimental framework extending the serverless model with group-based invocations and efficient inter-worker communication, enabling synchronized, massively parallel workloads with reduced startup latency and data movement.

The table above summarizes the primary software components that form the NEARDATA architecture, indicating their readiness level and providing a brief functional description. The subsections below elaborate on each component's role and key features to support the detailed analyses presented later in this deliverable.

2.2 Production-Ready Components

- **Lithops:** Lithops is a serverless execution framework for Python functions that operates directly on data stored in object stores, allowing users to parallelize tasks at scale without managing servers or clusters. It supports pluggable backends such as AWS Lambda, IBM Cloud Functions, and Kubernetes, providing portability across different infrastructures. Lithops automatically handles deployment, scaling, and data movement, enabling transparent access to large datasets and simplifying the execution of distributed workloads. It also integrates seamlessly with PyRun to ensure reproducibility, making it suitable for scientific pipelines and AI applications where consistent results across environments are critical. By combining elasticity, portability, and ease of use, Lithops lowers the entry barrier to serverless computing and accelerates the adoption of cloud-native data processing.

- **Scone:** SCONE is a leading confidential computing platform that enables secure execution of containerized applications inside trusted execution environments (TEEs) such as Intel SGX enclaves. By transparently encrypting files, network traffic, and secrets, SCONE ensures that sensitive data and code remain protected even in untrusted or cloud environments, without requiring changes to application source code. It provides robust attestation mechanisms to verify code integrity, advanced secret management, and policy-driven access control, making it possible for organizations to meet strict security and compliance requirements while supporting a wide range of use cases, including secure analytics, machine learning, and multi-party collaboration in regulated industries

- **Pravega:** Pravega is a CNCF (incubating) open-source storage system designed specifically for managing continuous and unbounded data streams with strong consistency and durability guarantees. Developed to support modern stream processing workloads, Pravega introduces the concept of stream as a first-class primitive, enabling seamless integration with stream analytics frameworks. Its key features include dynamic stream scaling, which allows automatic adjustment of stream throughput based on workload, and exactly-once semantics, ensuring reliable data delivery even in failure scenarios. Pravega also supports tiered storage, combining fast in-memory or SSD-based access with long-term persistence on cheaper storage media, making it ideal for near-data processing in edge-to-cloud infrastructures. For the EU-NEARDATA project, Pravega provides a robust foundation for real-time data ingestion and processing, enabling efficient and scalable stream-based data management.

- **Metaspace:** A scalable, open-access metadata catalog and analysis platform for high-resolution imaging mass spectrometry (MSI) data. It provides a high-throughput metabolite annotation engine that automatically identifies and maps metabolites from MSI datasets, generating structured metadata for advanced search and cross-dataset comparison. Beyond annotation, Metaspace maintains an extensive spatial metabolite knowledgebase comprising thousands of public datasets contributed by the scientific community, enabling data reuse and large-scale comparative studies.

  Through its RESTful APIs and interactive web interface, users can register datasets, visualize annotations, and explore spatial metabolomics results interactively. The platform supports the open `imzML` format, ensuring interoperability with all major mass spectrometer vendors. By combining open standards, rich metadata indexing, and com

- **PyRun:** PyRun is a web-based SaaS platform that unifies access to NEARDATA components such as Lithops, Pravega, Metaspace, DataPlug, and DataCockpit. It democratizes scalable cloud computing for Python users by automating infrastructure provisioning, scaling, and cleanup on AWS and IBM Cloud, allowing the execution of data analytics and AI workloads without requiring cloud expertise.

  Seamlessly integrating distributed frameworks like Dask, Lithops, Ray, and Cubed, PyRun enables effortless parallelism and serverless execution through a browser-based development environment. As part of NEARDATA's control plane, it acts as an orchestration layer that bridges experimentation and production workflows, ensuring reproducibility and efficient resource utilization across heterogeneous infrastructures.

- **DataPlug:** DataPlug is a client-side Python framework that enables efficient, read-only partitioning of unstructured scientific data stored in object storage systems such as Amazon S3. It introduces the concept of data slicing, allowing users to create lightweight, lazily evaluated partitions without moving or rewriting data. By exploiting S3's byte-range requests, DataPlug performs massively parallel reads on large cold datasets, achieving high throughput despite storage latency. Through its plug-in interface, it supports multiple scientific domains—including genomics, geospatial, metabolomics, and astronomy—and allows data slices to be distributed seamlessly to parallel computing environments such as Dask, Ray, or PySpark for near-data processing.

- **DataCockpit:** Is an interactive IPython widget built on top of the DataPlug framework that enables scientists and engineers to efficiently upload, browse, and explore datasets on Amazon S3. It provides a user-friendly interface to access curated public and Metaspace collections, benchmark data access performance to determine optimal batch sizes, and perform dataset partitioning through DataPlug's cloud-aware slicing mechanisms. Fully integrated into Jupyter notebooks, DataCockpit streamlines data exploration and preparation for elastic, parallel workloads across distributed infrastructures.

2.3 Research-Project Components

- **Serverless Vector DB:** Serverless Vector DB is a research prototype that explores the implementation of similarity search over embeddings within serverless environments. In contrast to traditional vector databases relying on persistent clusters, it decouples query execution from storage and leverages elastic cloud-native infrastructure to automatically scale resources according to demand. The system supports large embedding datasets such as DEEP, SIFT, and GIST, making it directly relevant for AI and machine learning pipelines requiring nearest-neighbor retrieval at scale. Integrated with frameworks like Lithops, it provides portability and reproducibility across heterogeneous backends while taking advantage of autoscaling and pay-per-use models. As a prototype, it also exposes the challenges of applying serverless principles to stateful workloads, including query latency, index warm-up times, and memory efficiency. These insights position Vector DB Serverless as a foundation for on-demand, scalable AI search services in NEARDATA.

- **FaaStream:** FaaStream is an experimental framework that unifies function-as-a-service execution with stream-based storage systems such as Pravega, S3, and NVMe-backed object stores. Its main contribution is enabling inter-function communication and state management through streams, introducing shuffle and state primitives that improve elasticity, reliability, and developer productivity. By leveraging stream semantics, it supports workloads such as online machine learning, feature extraction, and real-time analytics that require large-scale, low-latency dataflows. Validation through benchmarks—including ImageNet embedding extraction, Wikipedia wordcount, and a 100GB TeraSort workload—demonstrates its applicability to both compute - and I/O-intensive scenarios. Beyond performance, FaaStream incorporates mechanisms for dynamic scaling, automatic recovery, and transparent partitioning, offering a promising approach for reproducible, low-latency pipelines spanning both cloud and edge environments.

- **Nexus:** Nexus is a prototype for a data management mesh that enables programmable tiering and metadata-driven orchestration of streaming workloads. Its architecture revolves around two abstractions: *streamlets*, lightweight functions for data transformation, and *swarmlets*, scalable groups of streamlets for distributed execution. Together, they translate high-level management policies into concrete execution plans. Nexus intercepts tiering operations from Kafka, Pulsar, or Pravega into object stores such as S3, applying transformations such as compression, filtering, or routing before persistence. It integrates Redis for metadata management and exposes an S3-compatible API via S3Proxy to simplify downstream consumption. By abstracting the tiering boundary, Nexus enables declarative policies that support compliance enforcement, anonymization of sensitive data, and dynamic cost optimization. This demonstrates the feasibility of programmable, near-data management at the storage interface and aligns closely with NEARDATA's vision of extreme data infrastructures.

- **Glider:** Glider is a research prototype designed to provide transparent and policy-driven data movement across heterogeneous storage and compute environments. Its primary objective is to abstract the complexities of transferring data between object stores, stream systems, and compute backends, ensuring portability and efficiency in multi-cloud scenarios. Unlike ad-hoc solutions bound to specific providers, Glider introduces a unified model that automates replication, caching, and placement decisions based on performance and cost constraints. It is engineered to interoperate seamlessly with other NEARDATA components such as Lithops, PyRun, and Pravega, thereby facilitating reproducible and elastic workflows across distributed infrastructures. In practice, Glider addresses challenges such as heterogeneous API semantics, variable network latencies, and fluctuating workload demands by adapting transfers dynamically to maintain throughput and minimize overheads. Beyond its immediate functionality, the prototype highlights open research challenges in areas such as cross-cloud optimization, policy conflict resolution, and integration with confidential computing technologies. By abstracting low-level data logistics, Glider enables developers to focus on high-level pipeline design while ensuring efficient and reliable near-data processing, making it a key enabler for NEARDATA's broader vision of extreme data management.

- **FaaSTs:** FaaSTs is a novel approach that combines time series forecasting and the FaaS model to handle demanding workloads. First, the application is split into multiple tasks that are then executed on the available resources. A time series model continuously monitors the system and predicts the system workload, where extra workers can be temporarily deployed and assigned a pending task from the queue. This framework has been integrated, deployed, and tested into the Lithops-HPC architecture. It is currently still under continuous testing and development of enhancements.

- **Burst Computing:** Burst Computing is an experimental framework that extends the serverless paradigm to support massively parallel, synchronized workloads through group-based invocations known as *flares*. It introduces efficient inter-worker communication mechanisms and shared-memory coordination to minimize startup latency and data movement, overcoming the isolation and fragmentation limitations of traditional Function-as-a-Service (FaaS) systems. Built as an extension of Apache OpenWhisk, it incorporates a lightweight communication middleware implemented in Rust that enables collective operations such as broadcast and reduce among workers. This approach allows the execution of data-intensive analytics, graph processing, and model training tasks with improved elasticity and resource utilization. Evaluations on workloads such as PageRank and TeraSort demonstrate significant reductions in synchronization time and communication overhead, positioning Burst Computing as a step towards scalable, low-latency "serverless cluster" architectures within the NEARDATA ecosystem.

- **Lithops-HPC:** Lithops-HPC is a platform developed based on Lithops, allowing leveraging supercomputers and other HPC resources on top of the Lithops framework. The novelty of the project is to hide the complexities of resource management found in HPC environments from the end-user, allowing for easy resource configuration through a *YAML* file. The framework automatically handles the amount of resources required for the HPC environment and distributes the workloads' tasks accordingly, transparently to the user.

- **Flower-Lithops Simulation Backend:** it is a new extension added to the Flower federated learning framework, which provides a simulation backend powered by Lithops. This addition provides an alternative to Flower's default Ray-based backend, which is designed for serverful, cluster-based execution. The new backend enables more elastic, on-demand, and cost-effective federated learning simulations by leveraging the serverless computing paradigm.

- **Flower-Lithops Simulation Backend:** it is a new extension added to the Flower federated learning framework, which provides a simulation backend powered by Lithops. This addition provides an alternative to Flower's default Ray-based backend, which is designed for serverful, cluster-based execution. The new backend enables more elastic, on-demand, and cost-effective federated learning simulations by leveraging the serverless computing paradigm.

3 Architecture Summary and Component Alignment

3.1 NEARDATA Architecture at a Glance

The NEARDATA architecture has been designed to confront the challenges posed by extreme-scale, distributed, and privacy-sensitive data management. Its design philosophy rests on three fundamental pillars: scalability, to handle massive and heterogeneous data volumes efficiently; flexibility, to support diverse workloads across cloud, edge, and HPC infrastructures; and confidentiality, ensuring that data remains protected throughout its lifecycle — at rest, in transit, and in use.

The architectural model, illustrated in Figure 1, follows a layered and modular organization that promotes interoperability and composability across different components. It is structured into four horizontal layers—representing the progressive flow of data from sources to advanced analytics—complemented by a vertical control plane that provides governance, security, and orchestration capabilities across all layers. This organization reflects NEARDATA's guiding principle of separating data concerns (ingestion, transformation, and analysis) while ensuring tight integration through metadata-driven, AI-enhanced coordination.

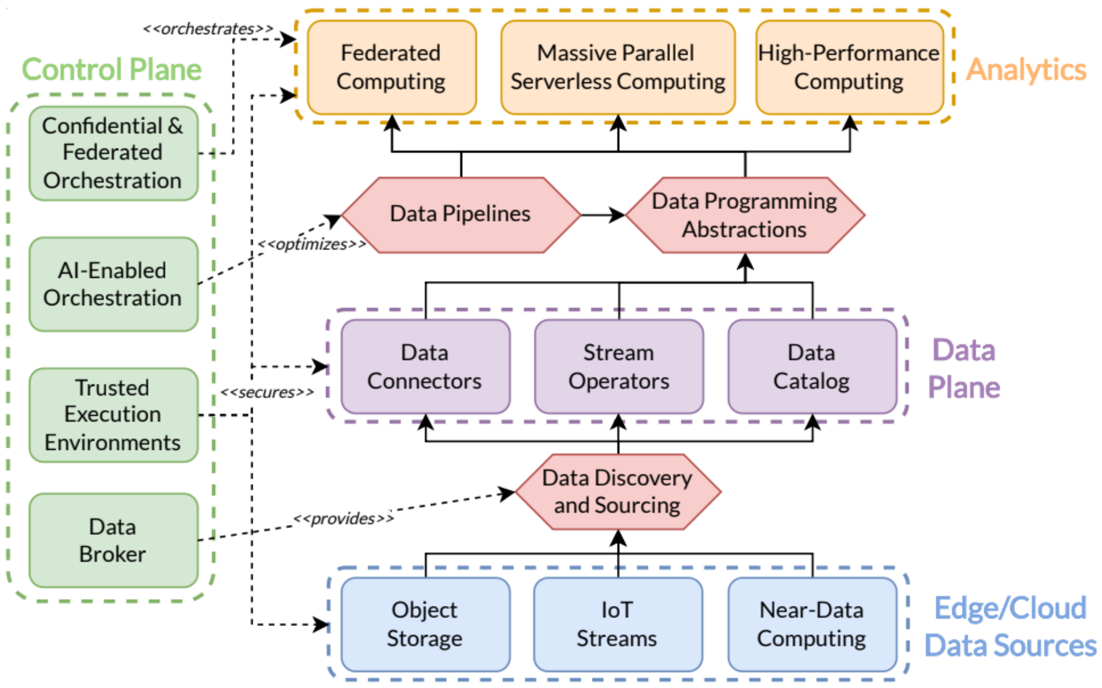


Figure 1: High-level NEARDATA architecture, structured into four horizontal layers and a vertical control plane.

Edge/Cloud Data Sources. At the foundation of the architecture lies a heterogeneous ecosystem of data origins, encompassing raw datasets stored in cloud object stores, continuous IoT streams, and edge compute nodes that generate, capture, or pre-process data prior to ingestion. This layer embodies the data gravity principle, bringing computation closer to where data is produced. By supporting hybrid ingestion patterns—batch and streaming alike—it ensures the adaptability of NEARDATA to both real-time and large-scale analytic scenarios. The layer establishes the foundation for the upper tiers, providing the raw inputs that drive the entire pipeline.

Data Plane. The Data Plane constitutes the core of data transformation and preparation. It is responsible for converting raw, heterogeneous inputs into structured, analytics-ready assets while preserving provenance, lineage, and semantic consistency. It encompasses several critical subsystems:

- Data Catalog: Offers a comprehensive metadata management service that supports indexing, annotation, and discovery of datasets, including schema information, partition structures, and provenance metadata. This facilitates transparent data access and traceability across federated environments.
- Data Connectors: Provide interoperability mechanisms for interacting with multiple storage backends and data formats. They handle dynamic partitioning, caching, and format conversion on-the-fly, minimizing data transfer overheads and enabling seamless data movement between cloud, edge, and HPC resources.
- Stream Operators: Enable low-latency, near-source processing of continuous data streams, such as sensor feeds or video analytics pipelines. They support fundamental stream operations (filtering, windowing, aggregation) to reduce the volume of data transmitted upstream while preserving the information's analytical value.
- Data Programming Abstractions: Define unified APIs and dataset interfaces that abstract the complexity of distributed data management. These abstractions allow developers and researchers to interact with remote datasets as if they were local, simplifying integration with diverse analytics and AI frameworks.

Analytics Layer. The Analytics layer serves as the execution environment for data-driven workloads. It integrates multiple paradigms—federated, serverless, and HPC computing—to provide flexible execution modes that adapt to the nature of the task and the infrastructure available. Serverless engines enable interactive analytics and on-demand scalability; federated frameworks allow for cross-domain computation without data centralization; and HPC backends support large-scale simulations and scientific workloads requiring high performance. Together, these engines empower NEARDATA to support a continuum of analytics—from edge intelligence to cloud-scale processing—while ensuring reproducibility and resource efficiency.

- Data Broker: Acts as the central policy and metadata coordinator, responsible for dataset registration, access control, and secure data exchange. It enables federated data governance across administrative domains, aligning with FAIR data principles.
- Trusted Execution Environments (TEEs): Introduce confidential computing primitives for protecting data-in-use via hardware-level isolation (e.g., Intel SGX). These mechanisms ensure that sensitive data can be processed securely even in untrusted environments.
- Confidential and Federated Orchestration: Provides policy-driven orchestration of distributed pipelines, ensuring compliance with data sovereignty, privacy regulations, and institutional governance rules. It also enables multi-party computation scenarios where execution policies are dynamically enforced.
- AI-Enabled Orchestration: Incorporates machine learning-based telemetry and adaptive optimization to improve scheduling, resource utilization, and performance predictability. This component transforms the orchestration process into a feedback-driven system capable of self-tuning according to observed workload patterns.

Together, these components form a cohesive ecosystem where data can seamlessly flow from heterogeneous sources to confidential, AI-optimized analytics pipelines, all under a unified governance and security model.

3.2 Mapping Software Components to Architectural Blocks

This section establishes the correspondence between NEARDATA's conceptual architecture and its concrete software ecosystem. Building upon the high-level model described in the previous subsection, each architectural block is now mapped to the software components that realize its functionality through a combination of mature open-source frameworks and research prototypes developed within the project.

Figure 2 illustrates this alignment, organizing the NEARDATA ecosystem into four horizontal layers—*Edge/Cloud Data Sources*, *Data Plane*, *Analytics*, and the vertical *Control Plane*. The components have been positioned according to the architectural structure previously introduced in Figure 1, ensuring a one-to-one correspondence between conceptual blocks and their concrete software realizations. Each element reflects its functional role and interaction within the system, highlighting both production-grade technologies and experimental innovations that collectively enable scalable, secure, and adaptive data analytics.

At the foundational level, the **Edge/Cloud Data Sources** layer represents the external infrastructure that provides data and computational resources to NEARDATA. This includes **Object Storage**, **IoT Streams**, and **Near-Data Computing** environments hosting large-scale scientific datasets and real-time data feeds. Access to these data origins is facilitated through components such as **DataPlug** and **DataCockpit**, which allow users to explore, benchmark, and partition public or private datasets efficiently.

The **Data Plane** forms the operational core of NEARDATA, handling data ingestion, transformation, and management. It integrates several key subsystems:

- **DataPlug**, a cloud-aware framework for dynamic and read-only partitioning of unstructured scientific data stored in object storage, enabling parallel and zero-copy data access.
- **Glider** and the **Serverless Vector DB**, which enable near-data computation and scalable vector-based similarity search.
- **Pravega** and **FaaStream**, supporting high-throughput stream ingestion and real-time data processing for continuous workloads.
- **Metaspace** and **DataCockpit**, providing metadata discovery, visualization, and data preparation capabilities within Jupyter notebooks or PyRun Cloud environments.

The **Analytics** layer executes computation across heterogeneous infrastructures. It integrates four complementary paradigms:

- **Federated Computing**, enabling collaborative analytics across multiple institutions or cloud domains without centralizing data.
- **Massive Parallel Serverless Computing**, realized through **Lithops** and extended with **Burst Computing**. While Lithops enables large-scale, embarrassingly parallel execution of unmodified Python workloads across multi-cloud environments, Burst introduces synchronous and collective execution primitives that allow functions to communicate and synchronize during execution. This design overcomes the isolation and coordination limits of traditional FaaS systems, supporting tightly coupled, parallel workloads such as reductions, broadcasts, and iterative computations.
- **High-Performance Computing (HPC)**, enabled via **Lithops-HPC**, extending serverless principles to tightly coupled workloads running on HPC clusters.

Together, these paradigms support hybrid execution models capable of combining cloud elasticity with HPC performance and federated data locality.

The vertical **Control Plane** governs orchestration, optimization, and confidentiality across all other layers. It encompasses:

- **PyRun**, which automates infrastructure provisioning, scaling, and cleanup, integrating frameworks such as Lithops, Dask, Ray, and Cubed within unified, reproducible workflows. In addition, PyRun plays a central role as the *Data Broker*, coordinating secure and policy-driven data exchange between analytics and data management components.
- **Nexus**, a data mesh prototype introducing streamlets and swarmlets for metadata-driven orchestration of heterogeneous backends.
- **SCONE** which implement trusted execution environments (TEEs) and enforce secure data handling across multi-cloud deployments.
- **FaaSTs**, an experimental framework leveraging AI and time-series analysis to predict resource usage in Function-as-a-Service (FaaS) environments, improving scheduling and resource allocation efficiency.

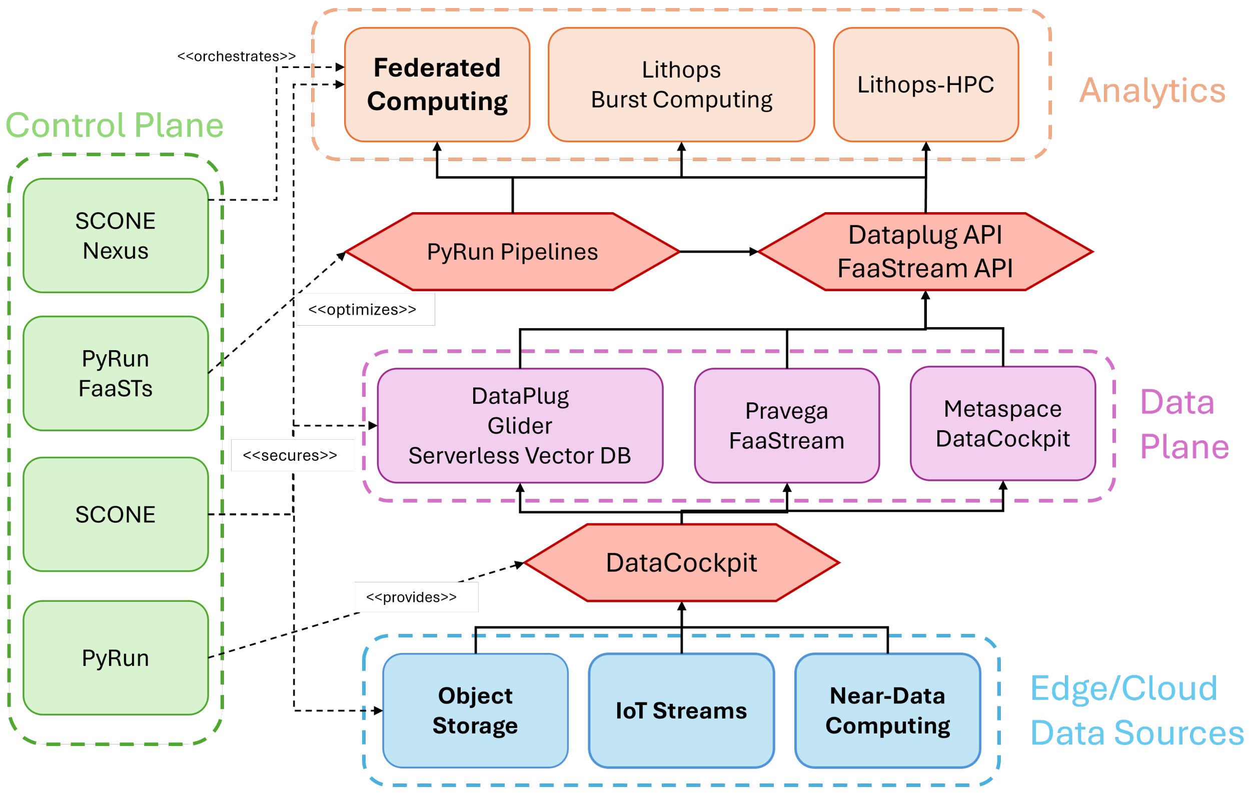


Figure 2: Overview of the NEARDATA software components organized by architectural layer. The diagram maps production-ready systems (e.g., Lithops, PyRun, Pravega, SCONE) and research prototypes (e.g., DataPlug, Glider, Nexus, FaaStream) to their corresponding functional layers, illustrating how NEARDATA achieves secure, elastic, and policy-driven operation across heterogeneous infrastructures.

Figure 2 provides a holistic view of how these components interact across the four horizontal layers and the vertical control plane. It highlights both the horizontal flow of data and computation—from ingestion to analytics—and the vertical orchestration functions that ensure confidentiality, compliance, and adaptability across the stack.

To complement the visual overview, Table 2 summarizes the relationship between each component and its corresponding architectural block, detailing its main responsibility within the NEARDATA framework.

Table 2: Component to Architecture Block Mapping

Component	**Architecture Block**	**Role / Responsibility**
DataPlug	Data Plane — Data Connectors	Provides a client-side, cloud-aware Python framework for dynamic and read-only partitioning of unstructured scientific data stored in object storage, enabling efficient, parallel, and zero-cost data slicing across multiple scientific domains.
DataCockpit	Data Plane — Data Connectors	Interactive Jupyter-based interface that leverages DataPlug to enable cloud-aware browsing, benchmarking, and partitioning of large scientific datasets in object storage, simplifying data preparation and integration within NEARDATA pipelines.
Glider	Data Plane — Data Connectors	Enables transparent, policy-driven data movement across heterogeneous storage and compute systems, managing replication, caching, and placement to optimize data locality and efficiency in multi-cloud environments.
Pravega	Data Plane — Stream Operators	Distributed storage system for data streams with tiered storage, supporting durable event writing, scaling, and retention policies.
FaaStream	Data Plane — Stream Operators	Integrates serverless execution with stream-based storage to manage low-latency, elastic data pipelines, facilitating continuous data processing close to the source.
Metaspace	Data Plane — Data Catalog	Provides a scalable metadata catalog and knowledge base for spatial metabolomics datasets, enabling automated metabolite annotation, open access to community-contributed imaging mass spectrometry data, and interactive exploration of annotated results through a web-based interface.
Lithops	Analytics — Massive Parallel Serverless Computing	Serves as the core distributed execution engine of NEARDATA, enabling large-scale, serverless, and multi-cloud execution of unmodified Python workloads across cloud, HPC, and on-premise infrastructures, while providing transparent scaling and infrastructure abstraction for data-intensive applications.

Continued on next page

Table 2 (continued)

Component	Architecture Block	Role / Responsibility
Burst Computing	Analytics — Massive Parallel Serverless Computing	Extends serverless computing with synchronous and collective execution primitives that allow functions to communicate and synchronize during runtime. This enables tightly coupled parallel workloads, such as reductions and broadcasts, overcoming traditional FaaS isolation constraints.
Lithops-HPC	Analytics — High-Performance Computing	Extends Lithops to orchestrate Python functions at massive scale in HPC environments, handling resource management transparently to the user, in contrast to traditional HPC frameworks.
Serverless Vector DB	Analytics — Data Connectors	Provides a serverless vector database that enables similarity search over large-scale embeddings with elastic scaling and on-demand compute provisioning.
Flower-Lithops Simulation Backend	Federated Orchestration	Enables more elastic, on-demand, and cost-effective federated learning simulations by leveraging serverless computing
PyRun	Control Plane — AI-Enabled Orchestration	Automates infrastructure provisioning, elastic scaling, and cleanup on AWS and IBM Cloud; integrates Dask, Lithops, Ray, and Cubed; and provides a unified web IDE with AI-ready pipelines.
SCONE	Control Plane — TEEs and Confidential/Federated Orchestration	Provides confidential computing primitives based on Trusted Execution Environments (TEEs) to secure distributed workflows and enforce privacy-preserving execution across multiple clouds.
Nexus	Control Plane — Confidential & Federated Orchestration	Coordinates metadata-driven orchestration across heterogeneous backends through lightweight streamlets and swarmlets, enabling dynamic and federated data workflows.
FaaSTs	Control Plane — AI-based Optimization	Reads resource utilization data to predict future usage and feeds these predictions to the analytics plane to improve resource allocation efficiency.

Each of the components listed above directly implements one or more functional blocks from the conceptual architecture, jointly realizing NEARDATA's design principles of near-data processing, elasticity, and confidentiality. For a detailed description of each component, including repository organization, code metrics, and integration details, please refer to Section 4.

4 Software Outcomes

This section presents comprehensive profiles for each NEARDATA software component, detailing repository information, codebase metrics, documentation, quality assurance practices, recent activity, and integration with the broader ecosystem.

4.1 Production Ready Components

4.1.1 Lithops

Over the 2023–2025 period, Lithops has undergone a steady and substantial evolution aimed at enhancing its robustness, interoperability, and developer experience across heterogeneous cloud and HPC environments. Throughout this period, the framework progressively incorporated support for newer Python runtimes (from 3.10 to 3.13) and expanded its ecosystem of execution backends, including Azure Virtual Machines, Singularity containers, Oracle Cloud Functions, and Google Cloud Run. Integration with distributed messaging systems such as Redis and RabbitMQ was also improved, enabling more reliable and efficient task coordination at scale.

From an operational standpoint, several releases focused on resilience and performance improvements, introducing automatic task retries, refined monitoring and logging capabilities, and fixes for cache handling and concurrency deadlocks. The Kubernetes backend was significantly upgraded, adding master-timeout configuration, standalone execution modes, and improved error reporting. These enhancements were complemented by a more capable command-line interface (CLI) with utilities for managing images, jobs, and workers, along with simplified configuration and deployment across AWS and IBM Cloud environments.

Infrastructure management was streamlined through features such as automatic VPC creation, refined resource limit definitions, and the introduction of EC2 EBS/CIDR configuration options. Collectively, these advancements have positioned Lithops as a mature, production-ready framework for large-scale, serverless, and reproducible data analytics workloads, consolidating its role as a cornerstone of the NEARDATA architecture for near-data computation.

The following subsections provide an overview of the Lithops repository, key software metrics, major releases between 2023 and 2025, and its supporting ecosystem. They also summarize its continuous integration practices, available documentation, and integration within NEARDATA's broader pipeline orchestration through PyRun.

Status: Production ready

Table 3: Lithops Key Metrics by Period

Metric	Value	Period
Lines of Code	35 060	All time
Forks	115	All time
GitHub Stars	351	All time
Active Contributors	47	All time
Commits	873+	2023–2025

Table 4: Lithops Releases (2023–2025)

Version	Highlights
3.6.2 (Sep 22, 2025)	Fix localhost backend error; make executor retry count configurable.
3.6.1 (Jul 22, 2025)	Fixed GCP Functions on long-running operations; fixed k8s and singularity bug with first execution.
3.6.0 (Feb 22, 2025)	Added Python 3.13 support; introduced EBS/CIDR configuration for EC2; addressed monitoring and invoker issues.
3.5.1 (Oct 22, 2024)	Improved core truth-value handling; standalone consume mode; fixed Azure VM execution.
3.5.0 (Sep 3, 2024)	Integrated Singularity backend; added Python 3.11 compatibility; implemented Kubernetes master-timeout; enabled AWS Lambda user tags.
3.4.1 (Jun 7, 2024)	Enhanced localhost error logging; resolved worker cache-loading bug.
3.4.0 (May 30, 2024)	Updated CLI runtime-list naming; reorganized extras; improved resource limits; introduced breaking changes.
3.3.0 (Apr 25, 2024)	Enabled automatic retry of failed tasks; added Cloud Run login; simplified AWS/IBM configuration; migrated to pytest.
3.2.0 (Mar 27, 2024)	Added Python 3.12 support; implemented Redis work queue; improved CLI timestamps and resiliency.
3.1.2 (Feb 22, 2024)	Added figure sizing and statistics in results; toggled IBM VPC check; fixed payload-size handling.
3.1.1 (Feb 8, 2024)	Supported Kubernetes RabbitMQ; improved Code Engine error reporting; fixed executor deadlock.
3.1.0 (Dec 5, 2023)	Introduced CLI commands for listing and deleting images, jobs, and workers; released localhost v2; enabled VM image deletion.
3.0.1 (Oct 9, 2023)	Added OCI Functions and Object Storage support; enabled Redis in Kubernetes; optimized Code Engine and Cloud Functions.
3.0.0 (Sep 4, 2023)	Added Azure VM backend; supported Python 3.10–3.11; enabled VPC auto-creation; enhanced CLI and storage features.
2.9.0 (Feb 21, 2023)	Public S3 bucket support; added OCI login; updated Kubernetes boto3 dependency; automated IBM VPC provisioning.
2.8.0 (Jan 25, 2023)	Config file path support; new CLI flags; GCP triggers; AWS session token support; wildcard partitioner.

Documentation: Written with Sphinx and hosted on ReadTheDocs (`https://lithops-cloud.github.io/docs/`); source files in the `docs/` directory. Organized into:

- **Overview & Architecture:** Introduction, design principles, and portability guidelines.
- **Installation & Configuration:** Installat, configuration schema, backend setup.
- **Compute API:** Futures, chaining, execution metrics.
- **Storage API:** Object store operations and partitioning strategies.
- **Data Processing:** Auto-partitioning methods; Dask and Joblib integrations.
- **Integrations:** Multiprocessing, StorageOS, Airflow, and scikit-learn support.
- **Advanced Features:** Monitoring hooks, custom runtimes, Prometheus integration, shared-object support.
- **Developer Guide:** Contribution workflow, code examples, changelog.
- **Additional Resources:** Blogs, conference talks, academic publications.

Testing & CI: Four GitHub Actions workflows cover linting, local unit tests, a cross-platform matrix (7 Python versions × 18 OS–Python combinations), and end-to-end integration tests.

Ecosystem & Integrations: Interfaces with Metaspace, PyRun, and multiple cloud backends; highlighted in Medium tutorials and conference presentations.

Supported Backends:

- Compute: Localhost; serverless (AWS Lambda, GCF, Azure Functions, Aliyun, Oracle Functions, OpenWhisk); container services (Code Engine, AWS Batch, Cloud Run, Container Apps, Kubernetes, Knative, Singularity); VMs (IBM VPC, AWS EC2, Azure VM).
- Storage: Object stores (IBM COS, Amazon S3, Google Cloud Storage, Azure Blob, Alibaba OSS, Oracle COS, Ceph, MinIO, Swift); in-memory stores (Redis, Infinispan).

News:

- **July 12, 2025:** Beyond 'Hello World': Powering Real-World Science and AI with PyRun [1]
- **July 11, 2025:** Effortless Serverless Python: Get Your Code Running in the Cloud in 3 Clicks with PyRun [2]
- **March 31, 2025:** Processing Cloud-Optimized Data in Python with Serverless Functions (Lithops, DataPlug) [3]
- **February 13, 2024:** How to Run Lithops over EC2 VMs Using the New Kubernetes Backend [4]

Scientific Papers:

- Serverful Functions: Leveraging Servers in Complex Serverless Workflows [5]
- Enhancing HPC with Serverless Computing: Lithops on MareNostrum5 [6]
- A Seer Knows Best: Auto-tuned Object Storage Shuffling for Serverless Analytics [7]
- Outsourcing Data Processing Jobs with Lithops [8]

4.1.2 Scone

Between 2023 and 2025, SCONE underwent a series of targeted enhancements that were closely aligned with the NEARDATA project's stringent requirements for secure, compliant, and high performance confidential computing. The release of SCONE 5.8 marked a foundational shift with the introduction of a robust governance framework and support for air-gapped operations. These features were pivotal for NEARDATA's Data Broker, enabling the definition, enforcement, and auditing of fine-grained security and compliance policies—capabilities essential for regulated domains such as healthcare (e.g., surgical AI, federated learning with sensitive patient data) and government, where isolated, policy driven deployments are mandatory. The governance extensions allowed NEARDATA to dynamically update access controls and maintain audit trails, directly supporting its multi-party data sharing and regulatory compliance objectives.

With SCONE 5.9, the addition of Write Ahead Logging (WAL) and significant improvements to the Configuration and Attestation Service (CAS) and runtime addressed NEARDATA's need for data integrity, durability, and operational resilience. WAL provided tamper-evident, crash-recoverable logging for all enclave operations, which was especially critical for NEARDATA's streaming use cases (such as Pravega-based real-time data pipelines) and federated learning workflows, where the integrity and recoverability of sensitive data exchanges are non-negotiable. The CAS/runtime enhancements enabled NEARDATA to securely orchestrate and attest multi-tenant, multi-application deployments at scale, ensuring that only authorized and attested workloads could access confidential data—an essential requirement for the Data Broker's role as a trusted intermediary.

The release of SCONE 6.0, with its support for Intel DCAP v4, was a mandatory and strategic upgrade for NEARDATA, guaranteeing continued compatibility with the latest Intel SGX attestation protocols and hardware. This ensured that NEARDATA's confidential computing environments remained secure, verifiable, and future-proof, even as the underlying hardware and cloud platforms evolved. The adoption of DCAP v4 was particularly important for maintaining robust remote attestation—a cornerstone of NEARDATA's trust model across diverse deployment scenarios, from on-premises clusters to public cloud environments.

Recent developments have further expanded SCONE's capabilities in line with NEARDATA's vision for zero-trust and multi-vendor interoperability. SCONE's zero trust architecture, as detailed in the latest SCONE documentation, enforces strict identity verification, end-to-end encryption, and policy-driven access control at every layer of the stack. This approach is now fully operational across heterogeneous, multi-vendor environments, including AWS, Azure, Google Cloud, and on-premises Kubernetes clusters. SCONE's support for multiple Trusted Execution Environments (TEEs) such as Intel SGX, AMD SEV, and Intel TDX enables NEARDATA to deploy confidential workloads seamlessly across different hardware and cloud providers, without sacrificing security or compliance. The platform's vendor-agnostic attestation and policy management services ensure that data and workloads are protected even when orchestrated across untrusted or hybrid infrastructures.

In summary, the evolution of SCONE from 2023 to 2025 was shaped by NEARDATA's real-world requirements for secure governance, air-gapped and compliant operations, data integrity and recoverability, and hardware-agnostic attestation. The most recent developments in zero trust and multi-vendor support, as documented in SCONE's official resources, have empowered NEARDATA's Data Broker and streaming applications to meet the highest standards of confidentiality, integrity, and operational trust in extreme data environments.

Status: Production ready
Repository: `https://sconedocs.github.io/`

SCONE Releases (2023–2025)

Version	Release Date	Key Features / Notes
5.8	2023-06-19	Governance, air-gapped operations, many new features
5.9	2024-08-07	Write-Ahead Logging (WAL), bug fixes, CAS/runtime improvements
5.10	2025-07-02	New features, bug fixes, CAS/runtime improvements
6.0	2025-08-08	Intel DCAP v4 support

SCONE 5.8 (June 19, 2023) Major Features:

- **Governance:** SCONE 5.8 introduced a comprehensive governance framework, enabling organizations to define, enforce, and audit security and compliance policies across confidential workloads. This includes policy management, access control, and compliance features, as well as improved audit logging and reporting capabilities.
- **Air-Gapped Operations:** This release added support for running SCONE in air-gapped environments, i.e., highly secure, isolated deployments where network connectivity is restricted or unavailable. This is essential for sectors with the highest security requirements, such as government, defense, and critical infrastructure.

Technical Impact:

- The governance and air-gapped features are critical for organizations with strict security and compliance requirements, such as those in finance, healthcare, or government sectors.
- The focus on governance and air-gapped support demonstrates SCONE's commitment to enterprise-grade confidential computing and regulatory compliance.

SCONE 5.9 (August 7, 2024) Major Features:

- Write-Ahead Logging (WAL):
- SCONE 5.9 introduced WAL, a robust mechanism for ensuring data integrity and recoverability in the event of failures. WAL is a release brought significant improvements to the Configuration and Attestation Service (CAS) and the SCONE runtime. These improvements enhance the security, scalability, and manageability of confidential workloads.

Technical Impact:

- WAL ensures that all modifications are first recorded in a tamper-evident log before being applied, supporting atomicity, consistency, and crash recovery.
- CAS/runtime improvements further enhance the platform's ability to securely deploy and manage workloads, with a focus on seamless upgrades and backward compatibility.

SCONE 5.10 (July 2, 2025) Major Features:

- SCONE 5.10 continued the trend of incremental improvements, with new features and further enhancements to CAS and runtime. The specifics of these new features are not detailed in the news summary, but the focus remains on stability, performance, and feature completeness.
- Bug Fixes: Ongoing bug fixes and further improvements to CAS and runtime.

Technical Impact:

- This release continues the trend of incremental improvements, focusing on stability, performance, and feature completeness.
- The emphasis on CAS/runtime suggests ongoing investment in the platform's core security and attestation infrastructure.

SCONE 6.0 (August 8, 2025) Major Features:

- **Intel DCAP v4 Support:** SCONE 6.0 adds support for Intel DCAP v4, a critical update for compatibility with the latest Intel SGX attestation infrastructure. This ensures SCONE can interact with Intel's attestation services using the latest protocols.

The move to DCAP v4 ensures SCONE remains compatible with the latest hardware and attestation standards, bringing security enhancements such as improved certificate management and updated cryptographic protocols. The mandatory upgrade and deprecation notice highlight the importance of staying current with security infrastructure to avoid service disruptions, especially for organizations relying on remote attestation for compliance and operational continuity.

Between 2023 and 2025, SCONE has demonstrated steady, security-focused growth, with each release introducing significant new features, reliability improvements, and expanded platform support. The most notable trends include a strong emphasis on governance, compliance, and attestation, as well as a proactive approach to hardware compatibility and lifecycle management. The mandatory upgrade to version 6.0.0 in 2025 is a critical milestone for all users, ensuring continued secure operations in line with evolving hardware and attestation standards.

Table [7] presents a list of images of sconified applications. Federated learning images, however, are not listed because they were not pushed to registry due to being very large. But the reader can find how to build them in the repository `https://github.com/neardata-eu/scone-artifacts`.

4.1.3 MinIO ported

MinIO is a cloud object storage system with S3 compatibility; it provides storage infrastructure for Lithops and has been increasingly employed in AI. It was ported to confidential computing (also called *sconification*) to cover storage protection.

Status: Experimental
Repository: `https://github.com/neardata-eu/scone-artifacts`

Technical Impact:

- Protection of data in use: every stored data will be written or read sometime; SCONE Runtime protects this mechanism;
- Protection of data at rest: if an adversary inspects a file directly on volumes protected by SCONE File Shield will not reveal anything from its original content; and it is transparent to authorized users;
- Protection of data in transit: files uploaded to or downloaded from MinIO traversing the network to/from authorized users are protected by SCONE File Shield;
- Authorization via attestation: secrets are only provisioned to an application if they are running supported by TEE security hardware; hence the communicating parties can reassure they are interacting with authentic party.

MinIO Version	SCONE Version	Key Features / Notes
RELEASE.2025-07-18T21-56-31Z	5.9.0-831-g2f59cb75a / 2025-06-20	Sconification of both server and client applications

Table 7: Confidential images made available
"experimental" due to not yet part of latest SCONE release

System	Image
Lithops Singularity + Metaspace	registry.scontain.com/sconecuratedimages/experimental/lithops:3.6.1-metaspace-5.9.0-239-g941a371d1
MinIO	registry.scontain.com/sconecuratedimages/experimental/minio:master-alpine3.21-sconectl-5.9.0-831-g2f59cb75a
Keycloak	registry.scontain.com/sconecuratedimages/experimental/keycloak:26-alpine3.20-sconectl-5.9.0-rc.5
MariaDB	registry.scontain.com/sconecuratedimages/experimental/mariadb:10.4.24-sconectl-keycloak-5.8.0
Pravega Clients	registry.scontain.com/sconecuratedimages/experimental/pravega:client_ben-alpine315-5.9.0
Pravega Clients	registry.scontain.com/sconecuratedimages/experimental/pravega:omb-benchmark-alpine315-5.9.0
Pravega Clients	registry.scontain.com/sconecuratedimages/experimental/pravega:prb-benchmark-ubuntu2004-5.9.0
Pravega Clients	registry.scontain.com/sconecuratedimages/experimental/pravega:rust-benchmark-ubuntu2004-5.9.0-rc.7

4.1.4 Pravega

Since 2023, the *Pravega core* repository has undergone significant enhancements aimed at improving scalability, resilience, and cloud-native integration. Major updates include support for Azure and

GCP bindings in the Long-Term Storage (LTS) layer, expanded data integrity checks, and performance optimizations such as segment-to-host mapping caching and improved checkpoint logging. The CLI tooling was also extended with recovery commands and diagnostics for stream metadata and transactions. These developments have strengthened Pravega's role as a foundational component for real-time, durable data streaming in distributed environments, aligning with NEARDATA's architectural needs for reliable data ingestion and processing.

Complementing the core, the Pravega ecosystem has evolved to support diverse deployment and analytics scenarios. The *Rust client* added stream cut APIs and reader tracking, broadening language support for developers. The *Zookeeper Operator* was updated for Kubernetes compatibility and multi-arch builds, while the *Pravega Operator* introduced Helm chart improvements and support for newer Kubernetes versions. Notably, Dell contributed to the *GStreamer Pravega connector*, enabling real-time video ingestion and analytics pipelines, as demonstrated in the NCT PoC. These ecosystem components collectively enhance Pravega's usability and integration with AI and edge workloads, reinforcing its role in NEARDATA's intelligent data space architecture.

Status: Production ready

Table 8: Pravega Key Metrics

Metric	**Value**	**Period**
Lines of Code	$\approx 300,000+$ (Java)	All time
GitHub Stars	2,000+	All time
Forks	409	All time
Contributors	100+	All time
Releases	89 (latest: v0.13.0)	All time
Commits	3,297+	All time

Documentation: Hosted at `https://pravega.io/docs/latest/`, the documentation includes architecture guides, deployment instructions, metrics API, and integration examples. It is maintained using Docusaurus and GitHub Pages.

Major Features:

- **Stream Abstraction:** Pravega treats streams as durable, append-only byte sequences with strong consistency and ordering guarantees.
- **Auto-Scaling:** Streams dynamically scale based on ingestion rate, avoiding static partitioning limitations.
- **Exactly-Once Semantics:** Ensures reliable delivery and processing of events even under failure conditions.
- **Tiered Storage:** Supports seamless data movement between fast local storage and long-term cloud backends (e.g., S3, HDFS, Azure).
- **Transaction Support:** Enables atomic writes across multiple events, critical for distributed stream processing.
- **Security:** TLS support and pluggable role-based access control.
- **Metrics Framework:** Built on Micrometer, supports StatsD and InfluxDB for observability.

Recent Releases:

- **v0.14.x Series (Pre-release):** Pravega introduced enhanced long-term storage support (Azure, GCP), improved CLI tooling, integrity checks in Segment Store, and expanded client language bindings (Rust, Python).

- **v0.13.0 (Sep 2023):** Added support for Azure and GCP storage backends, improved CLI tooling, and introduced integrity checks in the Segment Store.
- **v0.12.x Series:** Focused on performance improvements, native client bindings (Rust, Python), and simplified long-term storage (LTS) configuration.

Ecosystem Tools:

- **GStreamer Connector:** The `gstreamer-pravega` repository provides plugins to read and write Pravega streams using GStreamer. It includes:
  - `pravegasink` and `pravegasrc` for ingesting and retrieving video/audio streams.
  - `pravegatc` for transaction coordination.
  - Integration with NVIDIA DeepStream for writing metadata (e.g., bounding boxes) to Pravega event streams.
  - Support for HTTP Live Streaming via the Pravega Video Server.
- **Kubernetes Operators:**
  - **Pravega Operator:** Manages Pravega clusters on Kubernetes, automating deployment, scaling, and upgrades.
  - **Bookkeeper Operator:** Handles Bookkeeper clusters, which provide the underlying log storage for Pravega.
  - **Zookeeper Operator:** Manages Zookeeper clusters used for coordination and metadata services.
- **Rust Client with Python Bindings:** The `pravega-client-rust` project offers a native Rust client for Pravega. It is designed to support bindings to other languages, including Python, enabling lightweight and performant access to Pravega streams from non-Java environments.
- **Flink Connectors:** Enables integration with Apache Flink for stream processing.
- **Pravega Sensor Collector:** Ingests sensor data into Pravega streams, supports disconnected operation with local buffering.

Scientific and Technical Impact:

- Gracia-Tinedo, Raúl, Flavio Junqueira, Tom Kaitchuck, and Sachin Joshi. *"Pravega: A tiered storage system for data streams."* In ACM/IFIP Middleware'23, pp. 165-177. 2023.
- Gracia-Tinedo, Raúl, Flavio Junqueira, Brian Zhou, Yimin Xiong, and Luis Liu. *"Practical storage-compute elasticity for stream data processing."* In ACM/IFIP Middleware'23 (Industrial Track), pp. 1-7. 2023.
- Jundi, Omar, Raúl Gracia-Tinedo, Sean Ahearne, Pascal Spörri, and Bernard Metzler. *"Towards Multi-Tier Stream Data Tiering in the Cloud-Edge Continuum."* In IEEE ICNP'24, pp. 1-6. 2024.
- Finol, Gerard, Arnau Gabriel, Pedro García-López, Raúl Gracia-Tinedo, Luis Liu, Reuben Docea, Max Kirchner, and Sebastian Bodenstedt. *"StreamSense: Policy-driven Semantic Video Search in Streaming Systems."* In ACM/IFIP Middleware'24 (Industrial Track), pp. 29-35. 2024.
- Gracia-Tinedo, Raúl, Flavio Junqueira, and Tom Kaitchuck. *""Back to the Byte": Towards Byte-oriented Semantics for Streaming Storage."* In ACM/IFIP Middleware'24 (Industrial Track), pp. 43-49. 2024.

In addition to these publications related to NEARDATA, we have found 4 external publications using Pravega in the last 3 years.

License: Apache 2.0.

4.1.5 Metaspace

Over the 2023–2025 period, the METASPACE platform has continued to evolve as a central open infrastructure for spatial metabolomics data management, annotation, and sharing. Originally designed to democratize access to imaging mass spectrometry (IMS) datasets, it now serves as a mature and extensible system supporting automated, large-scale molecular annotation and metadata-driven exploration across laboratories and research institutions. Its modular design—comprising data ingestion pipelines, machine learning annotation services, and a collaborative web interface—has enabled efficient integration with other NEARDATA components such as Lithops, DataCockpit, and PyRun.

During this period, METASPACE introduced several key advancements aimed at improving scalability, usability, and analytical accuracy. These include enhanced support for multiple ML-based annotation models, the introduction of auto-refresh dataset comparison tools, and a fully production-ready machine learning annotation engine. Additionally, the platform refined its metadata handling, optimized user management, and expanded its APIs for broader programmatic interoperability.

The following sections present a comprehensive overview of METASPACE's software metrics, version history, documentation structure, testing setup, and integration within the NEARDATA ecosystem. Together, these components highlight METASPACE's role as the metadata and annotation backbone of the NEARDATA architecture, ensuring reproducible, high-quality metabolomics workflows.

Status: Production ready

Table 9: Metaspace Key Metrics by Period

Metric	**Value**	**Period**
Lines of Code	~439,329	All time
Forks	12	All time
GitHub Stars	~50	All time
Active Contributors	17	All time
Commits	171+	2023–2025

Table 10: Metaspace Releases (2023–2025)

Version	**Highlights**
2.1.6 (Aug 12, 2024)	Production-ready ML annotation support: • Multiple ML model annotations. • Dataset comparison page filter with auto-refresh. • Removed dataset updates on email change. • Fixed annotation page layout on XS devices.

Documentation: Hosted on the METASPACE Wiki (`https://github.com/metaspace2020/metaspace/wiki`), including:

- **Overview:** Objectives, architecture, core modules.
- **End-User Guides:** Data upload workflows; CSV export; Webapp and GraphQL API usage.

- **Developer Reference:** REST/GraphQL specs; data model schemas; engine internals; performance notes; integrations.
- **Installation & Deployment:** Docker Compose; Helm charts; Ansible playbooks; Lithops scripts.
- **Client SDKs:** Python and JavaScript libraries with examples.
- **How-To Guides:** Adding databases; SQL migrations; release procedures.
- **Developer Tips:** Local development and testing best practices.
- **Additional Resources:** Publications, presentations, workshops.

Testing & CI: GitHub Actions for linting, unit/integration tests (80% coverage), and schema migration checks across multiple Python versions.

Ecosystem & Integrations: Integrates with Lithops, DataCockpit, and various visualization tools; cited in several publications.

Scientific Papers:

- METASPACE-ML: Context-specific metabolite annotation for imaging mass spectrometry using machine learning [9]

4.1.6 DataPlug

Over the 2023–2025 period, DataPlug has evolved into a lightweight yet powerful client-side framework designed to facilitate efficient and scalable access to unstructured scientific data stored in object storage systems. By dynamically partitioning datasets and enabling parallel read operations, it allows researchers to process large volumes of data transparently across heterogeneous infrastructures.

The framework natively supports a diverse range of scientific formats, reflecting its multidisciplinary scope. These include generic formats such as CSV and raw text; genomics data formats like FASTA, FASTQ, and VCF; geospatial standards such as LiDAR, Cloud-Optimized Point Cloud (COPC), and Cloud-Optimized GeoTIFF (COG); metabolomics imaging formats such as ImzML; and astronomy datasets based on the MeasurementSet specification. This broad compatibility enables seamless integration of DataPlug into NEARDATA's data plane, serving as a bridge between heterogeneous storage systems and high-performance analytics pipelines.

Status: Production ready

Table 11: DataPlug Key Metrics by Period

Metric	Value	Period
Lines of Code	3,491	All time
Forks	12	All time
GitHub Stars	11	All time
Active Contributors	7	All time
Commits	156+	2023–2025

Documentation: Available at `https://github.com/CLOUDLAB-URV/dataplug/tree/master/docs`, covering:

- DataPlug API Overview
- Scaling Preprocessing with Joblib
- Developing Custom Plugins

- Supported Formats:
  - General: CSV, raw text
  - Genomics: FASTA, FASTQ, VCF
  - Geospatial: LiDAR, Cloud-Optimized Point Cloud
  - Metabolomics: ImzML
  - Astronomics: MeasurementSet

News:

- **2025:** Processing Cloud-Optimized Data in Python with Serverless Functions (Lithops, DataPlug) [3]

Scientific Papers

- **2024:** DataPlug: Unlocking Extreme Data Analytics with On-the-Fly Dynamic Partitioning of Unstructured Data [10]

4.1.7 DataCockpit

Over the 2023–2025 period, DataCockpit has evolved into an interactive IPython widget that enables scientists and engineers to explore, partition, and benchmark datasets stored in cloud environments. It allows users to browse and access both public datasets — such as those available in METASPACE or the AWS Open Data Registry — and their own private S3 buckets, providing a unified view of heterogeneous data sources within a single interface.

Built on top of the DataPlug framework, DataCockpit integrates benchmarking tools to evaluate partitioning strategies and identify the optimal data slice configuration for downstream processing. Once partitioned, the resulting data slices can be directly passed to frameworks such as Lithops for large-scale, parallel execution. This enables users to intuitively control the degree of parallelism based on the number of generated slices, without requiring manual infrastructure configuration.

As part of NEARDATA's data plane, DataCockpit serves as a bridge between data exploration and distributed computation, simplifying access to cloud-stored datasets and accelerating near-data analytics through interactive, visual, and reproducible workflows.

Status: Production ready

Table 12: DataCockpit Key Metrics by Period

Metric	Value	Period
Lines of Code	9,580	All time
Forks	0	All time
GitHub Stars	0	All time
Active Contributors	1	All time
Commits	7	2023–2025

Documentation: Available at `https://docs.pyrun.cloud/features/data-cockpit`, including:

- Overview of DataCockpit features
- Integration with PyRun Cloud
- Monitoring and visualization dashboards
- Deployment and usage instructions

4.1.8 Extreme Data Hub front-end

PyRun represents the unifying front-end of the NEARDATA ecosystem, providing a seamless environment where users can design, execute, and monitor large-scale data analytics and AI workflows. It acts as the main operational interface that integrates several core components developed within the project—most notably **Lithops**, **DataPlug**, and **DataCockpit**—into a single, coherent, and user-friendly platform. By abstracting the underlying infrastructure, PyRun allows scientists and engineers to focus entirely on logic and experimentation rather than cloud provisioning or system configuration.

Conceptually, PyRun embodies NEARDATA's vision of the **Extreme Data Hub**: a unified environment that combines data repositories and data pipelines under a single orchestration layer. This concept was initially introduced to bridge the gap between cloud-based data storage and distributed computation, enabling near-data analytics and cross-domain interoperability. Within this framework, PyRun materializes as the control and execution plane that connects diverse data sources, streamlines workflow execution, and ensures reproducibility and transparency in large-scale data-driven experiments.

Architectural Overview

At its core, PyRun is a web-based Software-as-a-Service (SaaS) platform that allows users to run Python workloads directly on their own cloud accounts, currently supporting both *AWS* and *IBM Code Engine*. The platform automatically provisions, scales, and tears down resources as needed, eliminating manual infrastructure management. It provides a full-featured web IDE, runtime management, real-time monitoring, and integrated data access, offering a consistent user experience across all execution environments.

The architecture of PyRun is modular and extensible. Its backend orchestrates distributed execution through a microservice-based design, exposing APIs for job submission, runtime configuration, and resource tracking. The frontend integrates these capabilities into a unified interface that abstracts complex infrastructure details while preserving full transparency and control. Thanks to its tight integration with Lithops, PyRun supports function-based and data-parallel workloads at scale, leveraging serverless or HPC backends depending on the nature of the computation.

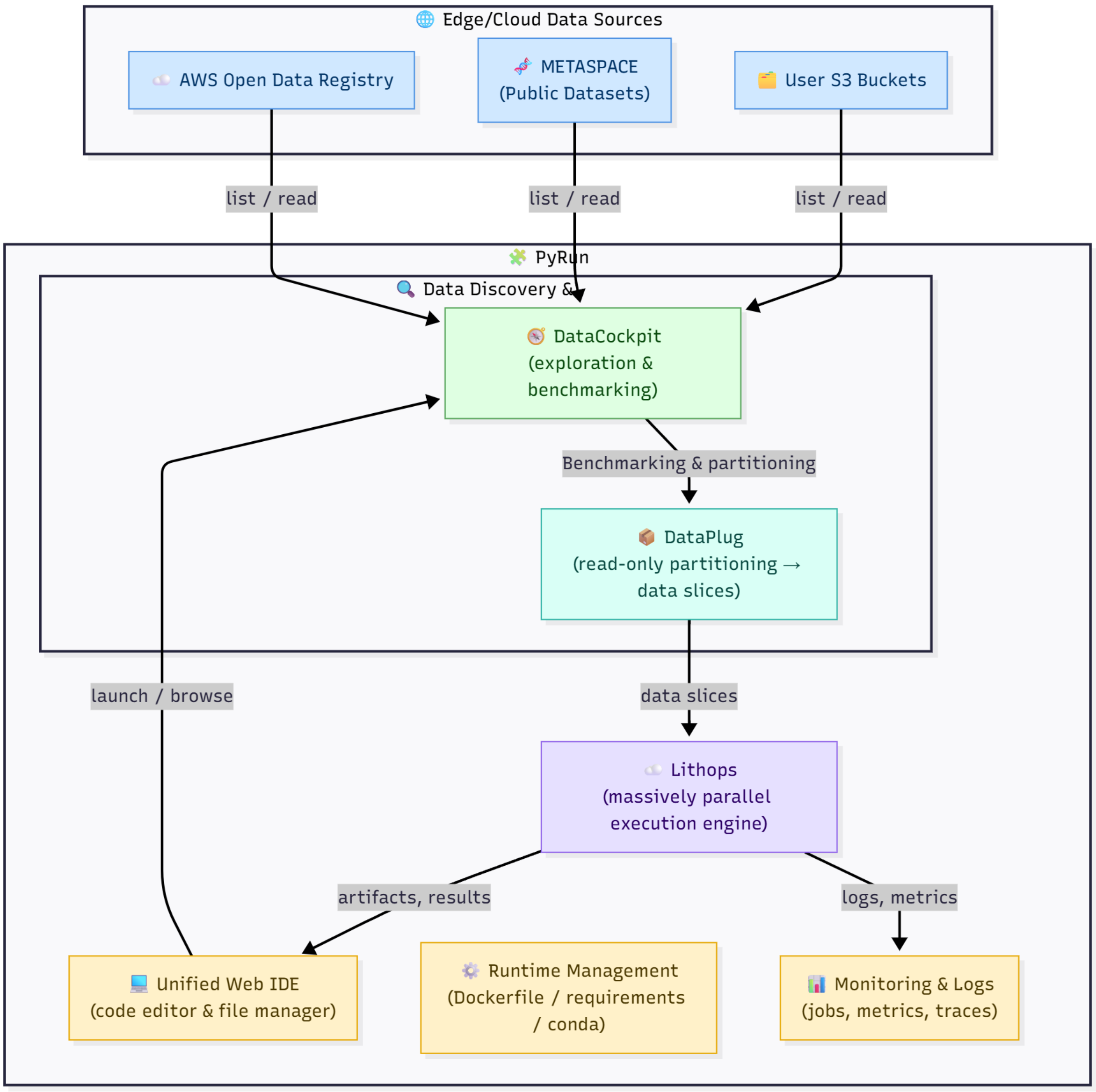

Edge/Cloud Data Sources
AWS Open Data Registry
METASPACE (Public Datasets)
User S3 Buckets
list / read
list / read
list / read
PyRun
Data Discovery &
DataCockpit (exploration & benchmarking)
Benchmarking & partitioning
DataPlug (read-only partitioning → data slices)
launch / browse
data slices
Lithops (massively parallel execution engine)
artifacts, results
logs, metrics
Unified Web IDE (code editor & file manager)
Runtime Management (Dockerfile / requirements / conda)
Monitoring & Logs (jobs, metrics, traces)

Figure 3: High-level architecture of PyRun, showing the integration of the Unified Web IDE with DataCockpit and DataPlug for data exploration and partitioning, and Lithops for distributed execution and monitoring.

As shown in Figure 3, users interact through the Unified Web IDE, which connects seamlessly with the **DataCockpit** to browse and benchmark datasets across heterogeneous storage sources (e.g., S3, METASPACE, AWS Open Data Registry). DataPlug performs on-the-fly partitioning, generating data slices that are dispatched to **Lithops** for massively parallel execution. Results and metrics are collected back into the IDE, providing a fully integrated, reproducible workflow for near-data analytics.

Integration with NEARDATA Components

PyRun acts as the central hub interconnecting multiple NEARDATA software outcomes:

- **Lithops:** Provides the distributed computing backend for function execution. PyRun seamlessly integrates with Lithops to launch tasks across heterogeneous environments, including AWS Lambda, Kubernetes, and IBM Code Engine.
- **DataPlug:** Facilitates dynamic data access and partitioning across unstructured datasets stored

in S3-compatible object storage. PyRun uses DataPlug to prepare data slices and feed them into parallel workflows.

- **DataCockpit:** Embedded directly in the PyRun interface, it enables users to visually browse public and private datasets, explore metadata, benchmark partitioning strategies, and trigger distributed processing pipelines.
- **Metaspace:** Integration with public repositories such as METASPACE enables transparent access to annotated metabolomics datasets for large-scale analysis pipelines.

By connecting these components, PyRun provides a consistent environment where NEARDATA's data plane and compute plane converge. It transforms the individual software modules of the project into a cohesive operational platform capable of executing reproducible, large-scale workflows with minimal configuration. Many of the integrated components—such as Lithops—are maintained by active open-source communities, ensuring long-term sustainability, continuous updates, and external contributions that strengthen the reliability and evolution of the NEARDATA ecosystem.

Core Functionalities

PyRun offers a comprehensive suite of functionalities designed for both research and production environments. At its center lies a **Unified Web IDE**, a browser-based environment that integrates code editing, runtime configuration, and job monitoring into a single workspace. Users can write standard Python code and execute it directly on the cloud, without leaving the interface.

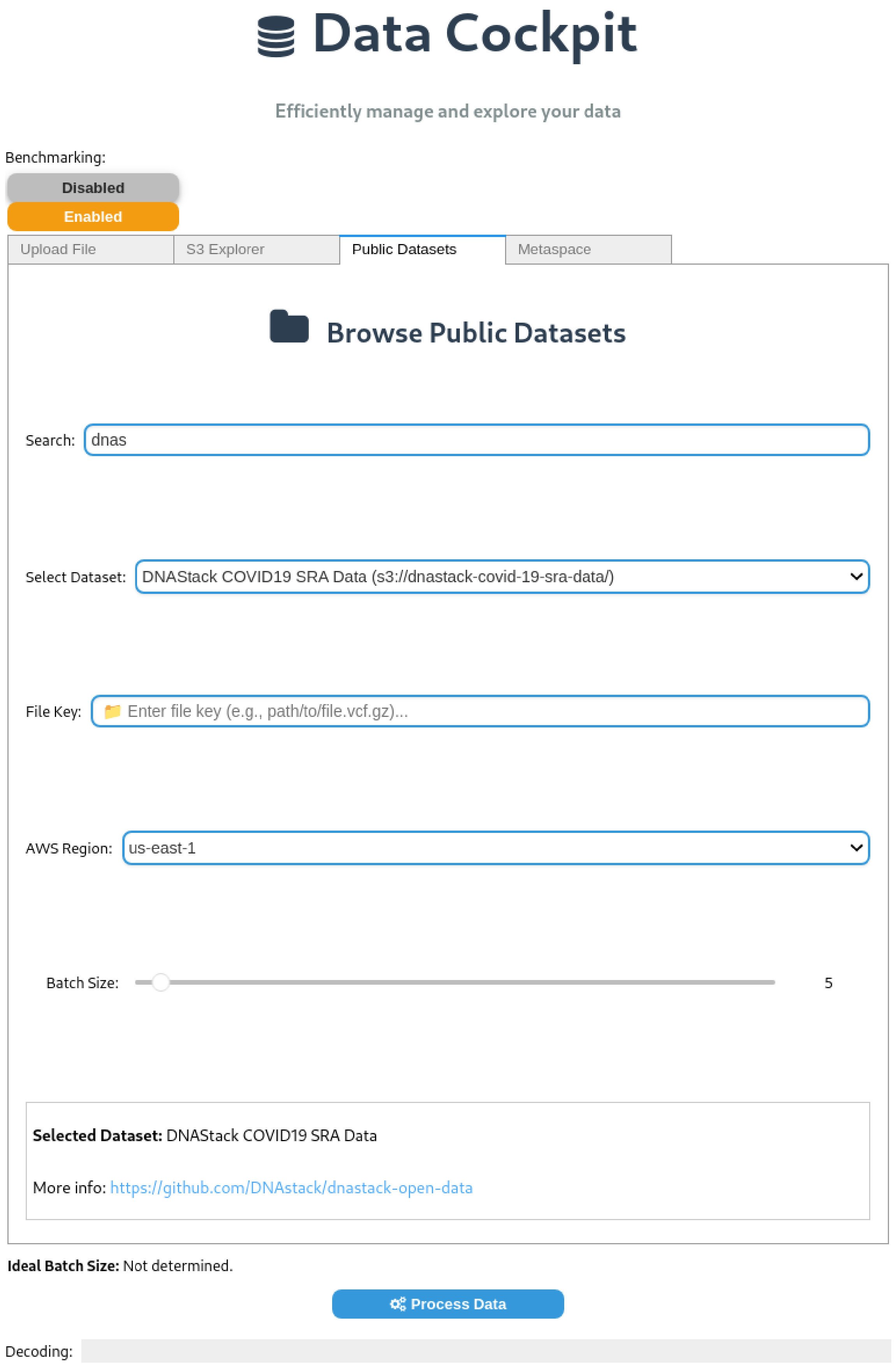
Through its **Runtime Management** system, PyRun ensures reproducibility and portability across environments by supporting definitions through `Dockerfile`, `requirements.txt`, or `environment.yml`. Each runtime can be built, versioned, and deployed automatically, guaranteeing consistent results across multiple executions.

The platform also incorporates **Real-Time Monitoring**, offering detailed visibility over job execution with live logs, performance metrics, and aggregated views from distributed workers. This facilitates debugging, profiling, and optimization of complex workloads.

A key component is the **DataCockpit Integration**, which connects PyRun with both user-managed and public data repositories. Through this interface, users can explore file structures, benchmark access speeds, and launch dynamic partitioning operations using DataPlug. The resulting data slices can then be seamlessly executed on distributed compute frameworks such as Lithops or Dask. Figure 4 illustrates the DataCockpit interface embedded in PyRun, providing a graphical and interactive layer for dataset exploration, benchmarking, and slicing.

Furthermore, PyRun includes a curated collection of **Templates and Pipelines**, which offer ready-to-run workflows for common use cases such as image classification, audio recognition, hyperparameter tuning, and text analytics. These pipelines act as both pedagogical examples and reproducible baselines for further research.

Finally, the system provides **AI-Ready Workflows**—preconfigured setups for machine learning and deep learning tasks, including retrieval-augmented generation (RAG), embedding extraction, and distributed model training—bridging serverless computing with state-of-the-art AI frameworks. Combined with its **Elastic Scaling** capability, which dynamically provisions cloud resources according to workload intensity, PyRun offers an adaptive environment ideal for both exploratory and production-scale computation.



Data Cockpit

Efficiently manage and explore your data

Benchmarking: Disabled [✓ Enabled]

Upload File | S3 Explorer | Public Datasets | Metaspace

Browse Public Datasets

Search: dnas

Select Dataset: DNAStack COVID19 SRA Data (s3://dnastack-covid-19-sra-data/)

File Key: 📁 Enter file key (e.g., path/to/file.vcf.gz)...

AWS Region: us-east-1

Batch Size: 5

Selected Dataset: DNAStack COVID19 SRA Data

More info: https://github.com/DNAstack/dnastack-open-data

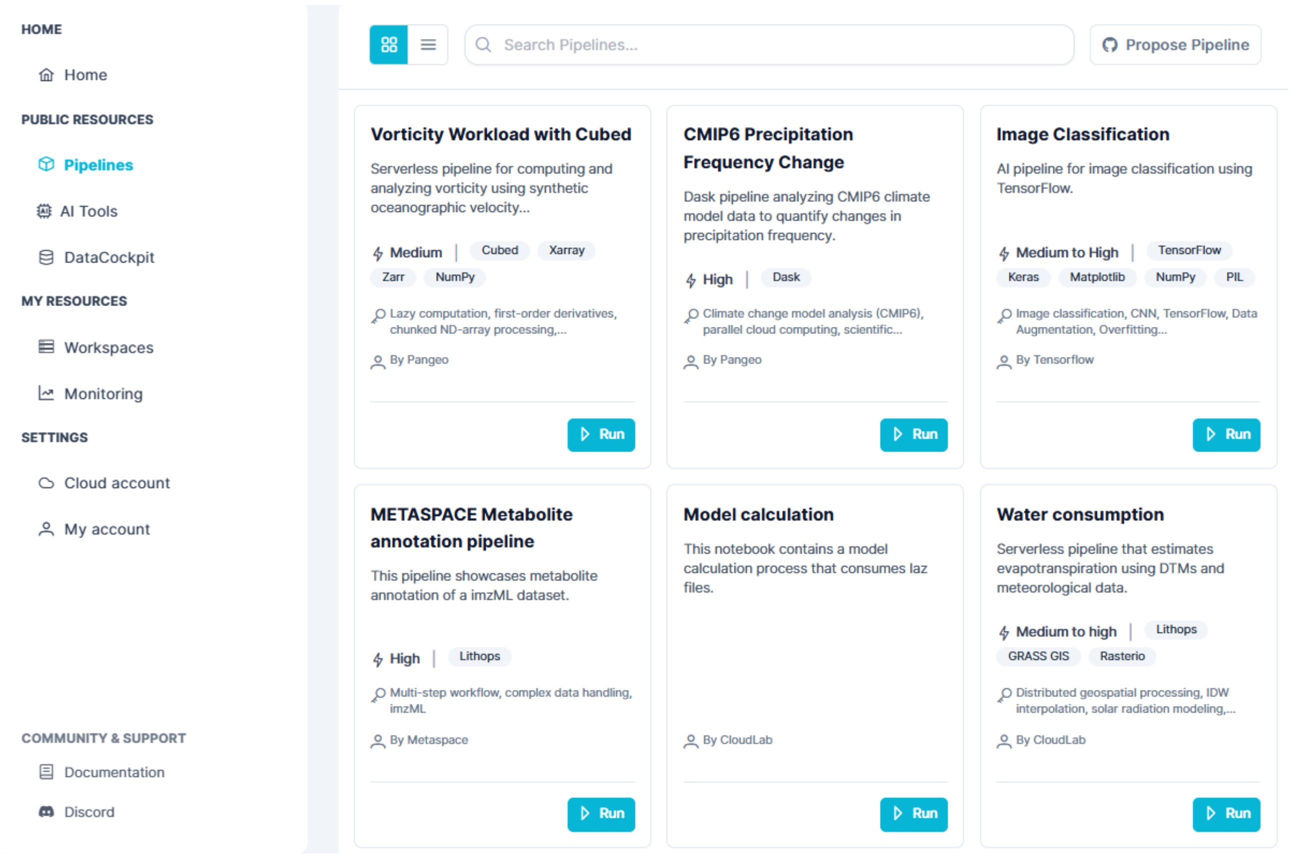
Ideal Batch Size: Not determined.

Process Data

Decoding:

Figure 4: DataCockpit interface embedded within PyRun, enabling interactive exploration and partitioning of datasets.

In addition to its data exploration capabilities, PyRun provides an integrated view of available computational workflows. Figure 5 illustrates the **PyRun frontend**, where a subset of ready-to-run pipelines can be launched directly from the interface. These include AI-oriented workflows such as image classification, hyperparameter optimization, and data preprocessing, as well as NEARDATA-related pipelines such as spatial metabolomics analysis and serverless geospatial processing. This visual layer enables researchers to deploy complex pipelines with minimal configuration effort.



Vorticity Workload with Cubed
Serverless pipeline for computing and analyzing vorticity using synthetic oceanographic velocity...
Medium | Cubed Xarray Zarr NumPy
Lazy computation, first-order derivatives, chunked ND-array processing,...
By Pangeo

CMIP6 Precipitation Frequency Change
Dask pipeline analyzing CMIP6 climate model data to quantify changes in precipitation frequency.
High | Dask
Climate change model analysis (CMIP6), parallel cloud computing, scientific...
By Pangeo

Image Classification
AI pipeline for image classification using TensorFlow.
Medium to High | TensorFlow Keras Matplotlib NumPy PIL
Image classification, CNN, TensorFlow, Data Augmentation, Overfitting...
By Tensorflow

METASPACE Metabolite annotation pipeline
This pipeline showcases metabolite annotation of a imzML dataset.
High | Lithops
Multi-step workflow, complex data handling, imzML
By Metaspace

Model calculation
This notebook contains a model calculation process that consumes laz files.
By CloudLab

Water consumption
Serverless pipeline that estimates evapotranspiration using DTMs and meteorological data.
Medium to high | Lithops GRASS GIS Rasterio
Distributed geospatial processing, IDW interpolation, solar radiation modeling,...
By CloudLab

Figure 5: PyRun frontend showing a subset of available pipelines ready for execution.

Relation to CLOUDLESS

PyRun has been developed through the joint efforts of the **NEARDATA** and **CLOUDLESS** projects. While conceived within NEARDATA as a unified frontend integrating its software components, its continuous development within CLOUDLESS has further expanded its capabilities as a general-purpose orchestration and execution platform for distributed and serverless computing.

Deployment and Extensibility PyRun's architecture supports seamless extensibility through modular APIs. New pipelines or datasets can be registered through simple metadata definitions, while existing components can be connected through the internal orchestration layer. The platform natively supports Python-based workflows but also exposes REST and SDK interfaces for integration with external systems. This flexibility allows PyRun to function both as a standalone experimentation environment and as a front-end to hybrid cloud infrastructures.

Moreover, PyRun hosts several NEARDATA pipelines directly in production-ready form. Workflows involving distributed image analysis, spatial metabolomics annotation, and serverless geospatial analytics have been deployed and validated within PyRun. The platform thus not only serves as an interface but also as the execution venue where the project's research outcomes are demonstrated and benchmarked.

Impact and Outlook

PyRun consolidates NEARDATA's mission of simplifying access to extreme-scale data analytics by merging data access, orchestration, and execution under a unified control plane. Its combination of runtime management, real-time monitoring, and integrated data exploration represents a significant step toward transparent, reproducible, and scalable scientific computing. Beyond its role in NEARDATA, PyRun's co-development within CLOUDLESS positions it as a general-purpose hub for serverless and multi-cloud experimentation, offering a blueprint for future European research infrastructures focused on open, cloud-native science.

In summary, PyRun is both a technological integrator and a functional demonstrator: it connects the diverse results of NEARDATA into a single operational environment, while also standing as a mature software product that embodies the project's vision of a truly interoperable and data-centric Extreme Data Hub.

4.2 Research Projects

4.2.1 FaaSTs

FaaSTs is a novel approach that combines time series forecasting and the FaaS model to handle demanding workloads. First, the application is split into multiple tasks that are then executed on the available resources. A time series model continuously monitors the system and predicts the system workload, where extra workers can be temporarily deployed and assigned a pending task from the queue. This framework has been integrated, deployed, and tested into the Lithops-HPC architecture. It is currently still under continuous testing and development of enhancements.

Status: Research prototype

Documentation: the documentation is available at `https://gitlab.bsc.es/datacentric-computing/lithops_telemetry_forecasting`. It contains several subrepositories with the different elements (telemetry predictor, scrapper and agent as well as the scripts to deploy the infrastructure with examples). Each of the repositories contains its own *README.md* with documentation on how it is distributed.

Ecosystem & Integrations: this prototype it is directly integrated into Lithops-HPC. Although the telemetry predictions and data gathering could be run standalone, the integration within an architecture using it is only done with Lithops-HPC. Nonetheles,s anybody could integrate it with other architectures.

Repository & Activity:

Table 13: FaaSTs Key Metrics by Period

Metric	Value	Period
Lines of Code	+8700	All time
Forks	0	All time
GitHub Stars	1	All time
Active Contributors	3	All time
Commits	253	All time

Key Features:

- Effortless Execution: integrated within Lithops-HPC it runs transparently to the end-user, no configuration needed.
- Automatic scalability: automatically adjusts resources to the Lithops-HPC backend according to availability and predicted resource usage.

4.2.2 Lithops-HPC

Status: Research-Prototype

Table 14: Lithops Key Metrics by Period

Metric	Value	Period
Lines of Code	2,916	All time
Forks	1	All time
GitHub Stars	0	All time
Active Contributors	2	All time
Commits	35	All time

Documentation: The documentation is available at `github.com/neardata-eu/lithops-hpc`. It contains an easy-to-follow guideline for writing the configuration file and how to run. Documentation on how to develop workflows is the same as for the Lithops framework described above.

Ecosystem & Integrations: It allows running in a wide range of environments: from accelerated HPC resources (i.e, GPU, FPGAs) to cloud environments such as Amazon AWS. It allows mixing different resources in the same environment, so the user can either choose or run their workloads in all of them at once, at their convenience.

With the integration of FaaSTs, it is also possible to automate the decision on which backend to run each of the workloads' tasks.

News:

- **March 1, 2024:** First release of Lithops-HPC [6]

Scientific Papers:

- Enhancing HPC with Serverless Computing: Lithops on MareNostrum5 [6]

4.2.3 Serverless Vector DB

Description: Retrieval-Augmented Generation and similar workloads use vector databases for unstructured data analysis, but traditional cluster-based systems struggle with elastic scaling under variable workloads. Serverless vector databases built on cloud functions offer a promising alternative but face challenges due to statelessness and dynamic data. This research project evaluates these challenges, showing that traditional clustering-based partitioning performs poorly in serverless setups. A block-based partitioning method is proposed, which handles dynamic data better and matches popular cluster-based solutions in performance—at lower cost—for sparse workloads.

Core Components:

- **Lithops:** Orchestrates workers on ingestion and vector search, balancing data load and enabling support for different backends.
- **Compute Backends:** Enable data ingestion, data indexing, search and data aggregation in a stateless way.
- **Storage Backends:** Are responible for data and index storage in a managed way.

Inputs: The vector database has been benchmarked with 3 popular datasets listed below. Those inputs must be preprocessed into a suitable format for the prototype. Stored in CSV files, entries must consist of two elements (the id and the vector space-separated) and each entry be written in a separate line.

- **DEEP:** from which are extracted subsets of 100k, 1M, 10M, and 100M embeddings of the Deep1B with 96 dimensions normalized with L2 distance extracted from the last fully-connected layer of a GoogLeNet model trained with the ImageNet dataset. `https://www.tensorflow.org/datasets/catalog/deep1b`

- **SIFT:** consists of 10M embeddings with 128 dimensions, the Scale Invariant Feature Transform (SIFT) transforms the image data into a large number of features (scale invariant coordinates) that densely cover the image features. `https://archive.ics.uci.edu/dataset/353/sift10m`
- **GIST:** consists of 1M embeddings with 960 dimensions, a gist represents an image scene as a low-dimensional vector. `https://huggingface.co/datasets/fzliu/gist1m/tree/main`

Repository & Activity:

Table 15: Serverless Vector DB Key Metrics by Period

Metric	Value	Period
Lines of Code	1,580	All time
Forks	1	All time
GitHub Stars	2	All time
Active Contributors	4	All time
Commits	18	All time

- Last update: October 27, 2025

It should be noted that most of the work was conducted in a private repository with other contributors prior to the creation of the aforementioned repository (65 commits, four total contributors).

Documentation:

- Comprehensive `README` and detailed paper published in .
- No formal unit tests; validated via example and additional experiments found in `https://github.com/neardata-eu/serverless-vdb-experiments`

Reproducibility: Guidelines are provided in the README with small demo-prepared files to run a basic example of the vector db. Lithops supports multiple backends, out of which AWS Lambda with AWS S3 and Kubernetes with MinIO have been tested and used for experimenting. Any data source adapted to the specified format can be used with the prototype perfectly.

Status: Research-Prototype

4.2.4 FaaStream

Description: The project introduces FaaStream, a serverless data processing framework designed to simplify Function-as-a-Service (FaaS) workflows by using elastic, tiered data streams as a unified storage solution. Traditional FaaS systems rely on multiple external storage options, which complicates development due to trade-offs in performance and scalability. FaaStream addresses this by providing dataflow-style features—such as inter-function communication, data shuffling, and state management—built on top of streaming storage. The framework demonstrates improved performance, efficiency, and fault tolerance, highlighting the value of stream-based storage in unifying serverless computing environments.

Core Components:

- **Pravega:** Used as an elastic and tiered storage system between worker stages on the pipeline. Constitues a part of the storage layer.
- **S3 and NVMe Volumes:** Used as sinks, object storage and write-ahead logs, they constitue a part of the storage layer.
- **Lambda:** Used as workers for processing data on each step of the pipeline through user defined functions. This constitues the computing layer.

- **FaaStream Manager**: Used as a coordinator for workers on the different stage of the pipeline, this component constitutes the coordination layer.

Inputs: The FaaStream project has been tested and benchmarked with subsets from different publicly available datasets listed below.

- **ImageNet:** composed by images organized following the WordNet hierarchy, it has been used to benchmark Embeddings Generation process with a subset of images having an approximate size of 35KB each. Raw bytes from those image are extracted to additionally benchmark Hashing process. `https://www.image-net.org/download.php`
- **Plain Text Wikipedia 2020-11:** composed by Wikipedia entries from which 120 files' text are extracted for the conducted benchmarks over WordCount process. `https://www.kaggle.com/datasets/ltcmdrdata/plain-text-wikipedia-202011`
- **TeraGen/TeraSort/TeraValidate:** The dataset is generated randomly and subsequently sorted by the key. Following prior works, a 100GB dataset is used for evaluating the TeraSort benchmark.

Repository & Activity:

Table 16: FaaStream Key Metrics by Period

Metric	**Value**	**Period**
Lines of Code	20,270	All time
Forks	1	All time
GitHub Stars	0	All time
Active Contributors	2	All time
Commits	132	All time

- Last update: May 19, 2025

Documentation:

- Comprehensive `README`.
- No formal unit tests; benchmarks are provided in the repository with additional instructions.

Reproducibility: Guidelines are provided in the README to deploy the pipeline. Multiple benchmarks are provided as well to evaluate the performance.

Status: Research-Prototype

4.2.5 Nexus

Description: Nexus is a data management mesh designed to transparently mediate storage operations between streaming systems and external tiered storage. It introduces a programmable substrate for advanced data management during the tiering process, enabling location-aware, extensible, and policy-driven stream data handling. Nexus supports stream-aware processing, routing, and caching through modular components that operate on tiered data chunks.

Core Components:

- **Streamlets:** Executable data management functions (e.g., compression, routing, filtering) applied to tiered data chunks.
- **Swarmlets:** Sets of Nexus worker instances deployed across Edge/Cloud infrastructures to execute streamlets.

- **Policies:** Declarative rules that orchestrate streamlet execution pipelines, including location constraints and hardware requirements.

Inputs: Nexus operates on tiered data offloaded by streaming systems such as Apache Kafka and Apache Pulsar. It intercepts storage operations via standard APIs (e.g., AWS S3) and processes only the relevant data files:

- **Kafka:** Operates on .log files offloaded to S3-compatible storage; preserves indexing logic for compatibility.
- **Pulsar:** Operates on -ledger-<id> files; indexing files are (by default) left untouched.
- **Pravega:** Operates on segment data chunks; metadata files are (by default) left untouched.

Repository & Activity:

Table 17: Nexus Key Metrics by Period

Metric	Value	Period
Lines of Code	13,043	All time
Forks	1	All time
GitHub Stars	3	All time
Active Contributors	4	All time
Commits	65	All time

- Last update: September 2025

Documentation:

- README includes architecture overview, deployment guides for Kafka and Pulsar, and local simulation instructions.
- Metadata management and streamlet descriptors are documented in the docs/ folder.

Reproducibility: Nexus can be run locally using a filesystem backend to simulate tiered storage interception. Full deployment guides are provided for Kafka and Pulsar, including configuration examples and supported file formats. Nexus supports standard S3 APIs via S3Proxy and uses Redis for metadata management.
Status: Research-Prototype

4.2.6 Confidential Data Exchange Platform: Advanced Security Modules

Research prototype enabling secure, policy-driven brokering and exchange of sensitive data between providers and consumers. Integrates advanced confidential computing and security research:

- CRISP:Provides robust confidentiality, rollback protection, and integrity for persistent storage by encrypting data at rest, binding versions with hardware monotonic counters, and detecting tampering or unauthorized modifications.
- LLD (Last-Level Defense):Ensures application-level integrity and confidentiality by running sensitive workloads inside hardware-enforced TEEs, with runtime attestation, memory encryption, and defense-in-depth against privileged attacks.
- TICAL:Secures the software supply chain by enforcing trusted, integrity-protected compilation, using cryptographic attestation and reproducible builds to guarantee that only verified code is deployed and executed.

- SinClave:Enforces hardware-assisted singleton properties for critical TEE-based services, preventing state forking and duplication, and ensuring unique, trusted execution of sensitive operations.
- Latency-Security Tradeoff: Informs the platform's design for streaming and real-time analytics, balancing TEE-based security with practical performance through batching, multi-threading, and enclave-aware buffering. Supports federated analytics, real-time AI, and regulatory compliance in distributed, multi-tenant environments, with end-to-end auditability and policy-driven access control.

Table 18: Research Outcome Integration into NEARDATA's Confidential Data Broker

Scientific Paper & Repository	**Security Integration Contribution**	**Integration Point to Data Broker**	**Integration with NEARDATA Ecosystem**
CRISP: Confidentiality, Rollback, and Integrity Storage Protection for Confidential Cloud-Native Computing [Paper (IEEE S&P 2024)]	Ensures data-at-rest confidentiality, integrity, and rollback protection. Integrates with file system shield to transparently encrypt and authenticate persistent data. Prevents rollback attacks on stored data.	Storage subsystem (via SCONE FS shield or similar TEE-backed storage layer).	Integrated as the storage security layer, ensuring all persistent data handled by the broker is protected and auditable.
Last-Level Defense for Application Integrity and Confidentiality (LLD) [Paper (ACM 2024)]	Provides runtime integrity, memory safety, and side-channel mitigation. Implements control-flow integrity and dynamic attestation. Ensures sensitive logic/-data is only accessible within the enclave.	In-enclave runtime (application logic and analytics modules).	Embedded in the TEE runtime, providing a last line of defense for all sensitive computations within the broker.
TICAL: Trusted and Integrity-protected Compilation of ApPLications [Paper (IEEE S&P 2024)]	Secures the software supply chain by ensuring only trusted, integrity-checked code is executed. Enables remote attestation of code provenance. Supports regulatory compliance and operational trust.	Build/deployment pipeline (attested binaries, provenance metadata).	Used in the CI/CD pipeline to produce attested, integrity-checked broker components; supports remote attestation for data owners and auditors.

Continued on next page

Table 18 continued from previous page

Scientific Paper & Repository	**Security Integration Contribution**	**Integration Point to Data Broker**	**Integration with NEARDATA Ecosystem**
SinClave: Secure and Efficient In-Network Key Management for Multi-Tenant Datacenters [Paper (arXiv 2023)]	Enforces singleton properties for critical TEE-based services (e.g., key managers, policy enforcers). Prevents state forking and service duplication. Maintains state continuity and trust in distributed deployments.	Critical service coordination (key management, policy enforcement modules).	Integrated as a singleton enforcement mechanism for critical broker services, ensuring state continuity and preventing split-brain scenarios.
Understanding the Latency-Security Tradeoff [GitHub]	Provides empirical analysis of performance overheads in TEE-based streaming. Guides system design for batching, multi-threading, and enclave-aware buffering. Ensures predictable performance under high data rates.	Streaming/data processing layer (Pravega, federated learning orchestration).	Used to tune the broker's streaming and analytics modules for optimal security-performance balance, ensuring practical deployability in real-world workloads.

4.2.7 Glider

Glider is a research prototype designed to provide transparent and policy-driven data movement across heterogeneous storage and compute environments. Its main objective is to abstract and automate data transfers between object stores, stream systems, and compute backends, ensuring portability and performance in multi-cloud and hybrid infrastructures. By leveraging metadata-aware decision mechanisms, Glider dynamically optimizes replication, caching, and placement based on workload profiles, cost models, and performance constraints.

Unlike ad-hoc scripts or platform-specific data movers, Glider introduces a unified model that integrates with NEARDATA components such as Lithops, PyRun, and Pravega, facilitating reproducible and elastic workflows. It supports on-demand data movement, dynamic adaptation to network fluctuations, and transparent cross-cloud coordination, positioning it as a key enabler for near-data execution in distributed environments.

Status: Research-Prototype

Table 19: Glider Key Metrics

Metric	**Value**	**Period / Notes**
Lines of Code	26,740	All time
Forks	1	All time
GitHub Stars	2	All time
Active Contributors	11	All time
Commits	4	(2023—2025)

- Last update: December 11, 2024

Documentation: Available in the `README.md` file of the official repository (`https://github.com/CLOUDLAB-URV/glider-store`), which provides installation instructions, architectural overview, and usage examples.

Integration with NEARDATA Ecosystem: Glider serves as a bridge between the Data Plane (object and stream storage) and the Analytics Plane (serverless and HPC computing), providing the substrate for efficient near-data execution. Through its policy-based transfer orchestration, it supports hybrid workflows that span multiple infrastructures, seamlessly linking storage locations to execution backends. Its design aligns with the NEARDATA vision of enabling extreme data analytics through composable, elastic, and transparent middleware.

Scientific Papers

- Glider: Serverless Ephemeral Stateful Near-Data Computation [11]

4.2.8 Burst Computing

Burst Computing is an experimental executor that enhances the serverless paradigm by introducing group-based invocations (flares) and efficient inter-worker communication, aiming to support tightly coupled, high-performance workloads in near-data environments. Developed to address the limitations of traditional FaaS isolation and fragmentation, Burst enables synchronous collective operations (e.g. broadcast, reduce) among stateless functions, improving coordination and reducing data movement.

Status: Research-Prototype

Repository: Available at `https://github.com/Burst-Computing/lithops-burst`. The organization also maintains multiple supporting repositories, including:

- `burst-communication-middleware` — implements the core communication layer in Rust.
- `openwhisk-burst` — integration with Apache OpenWhisk to enable flare invocations.
- `lithops-burst` — bridges Burst primitives with Lithops for seamless invocation.
- Additional repositories such as `burst-validation` and `burst-doc` provide validation tools and documentation.

Table 20: Burst Computing Key Metrics

Metric	Value	Period / Notes
Lines of Code	51,571	All time
Forks	1	All time
GitHub Stars	0	All time
Active Contributors	4	All time
Commits	214	(2023–2025)

- Last update: Novembre 16, 2024

Documentation: Basic documentation is available via the repository's `README` and sub-modules.

Integration with NEARDATA Ecosystem: Burst is intended to integrate with Lithops (via `lithops-burst`) to enable advanced synchronous parallelism in workloads managed through PyRun. It complements DataPlug by enabling complex analytics to run closer to data, especially when tasks require inter-node synchronization or collective communication.

4.2.9 Flower-Lithops Simulation Backend

The Flower federated learning framework was extended with a simulation backend powered by Lithops. This addition provides an alternative to Flower's default Ray-based backend, which is designed for serverful, cluster-based execution. The new backend enables more elastic, on-demand, and cost-effective federated learning simulations by leveraging the serverless computing paradigm.

The integration allows Flower to run federated learning clients as ephemeral, stateless functions across a wide range of cloud providers (e.g., AWS, Google Cloud, IBM Cloud) and HPC environments managed by Lithops. This approach removes the need to maintain persistent infrastructure for simulations, which aligns with NEARDATA's objective of promoting scalable, near-data AI workloads with minimal operational overhead. This component connects Flower's federated learning capabilities with the multi-cloud execution power of Lithops, making large-scale simulation environments more accessible.

Status: Research-Prototype
Repository & Contribution: The component is a direct contribution to the main Flower framework (`https://github.com/adap/flower`), with the development branch for this feature available at `https://github.com/janprz/flower/tree/add-lithops-backend-support`. The integration was implemented as a new `LithopsBackend` module, submitted to the upstream project to expand its core capabilities. As an extension of a major open-source project, its metrics are encompassed within the broader Flower ecosystem.
Key Features:

- Dynamic Scalability: Federated learning clients are executed as serverless functions, enabling simulations to scale automatically to thousands of clients without manual provisioning.
- Cost-Efficiency: The serverless model operates on a pay-per-use basis, drastically reducing costs compared to maintaining dedicated virtual machines or clusters for the duration of a simulation.
- Multi-Cloud Flexibility: By building on Lithops, a single Flower simulation can orchestrate clients across different cloud backends simultaneously, enhancing resilience and avoiding vendor lock-in.
- Seamless Integration: The backend implements Flower's native `Backend` interface, making it a drop-in replacement for the existing Ray backend with minimal changes to user code.

Technical Implementation: The core of the contribution is the `LithopsBackend` class, which abstracts the execution logic. It maps the lifecycle of a Flower simulation client to a serverless function invocation managed by Lithops. This design allows Flower's simulation engine to offload client execution requests to the Lithops framework, which in turn handles the function deployment, data serialization, execution, and result retrieval from the underlying cloud or HPC infrastructure.

4.3 Consolidated Software Metrics Across Components

This subsection consolidates repository-level metrics reported in Section 4 for all NEARDATA software outcomes. Periods are indicated when the commit counts refer to a specific timeframe (e.g., 2023–2025); otherwise, values are all-time.

Table 21: Consolidated repository metrics for NEARDATA components (as reported).

Component	**Category**	**LOC**	**Stars**	**Forks**	**Contrib.**	**Commits (Period)**
Lithops	Production-Ready	33,217	351	115	47	873+ (2023–2025)
SCONE	Production-Ready	—	—	—	—	—
Pravega	Production-Ready	∼300,000+	2,000+	409	100+	3,297+ (all time)
Metaspace	Production-Ready	∼439,329	∼50	12	17	171+ (2023–2025)
DataPlug	Production-Project	3,491	11	12	7	156+ (2023–2025)
DataCockpit	Production-Project	9,580	0	0	1	7 (2023–2025)
Lithops-HPC	Research-Prototype	2,916	0	1	2	35 (all time)
Serverless Vector DB	Research-Prototype	1,580	2	1	4	18 (all time)
FaaStream	Research-Prototype	20,270	0	1	2	132 (all time)
Nexus	Research-Prototype	13,043	3	1	4	65 (all time)
Glider	Research-Prototype	26,740	2	1	11	4 (2023–2025)
Burst Computing	Research-Prototype	51,571	0	1	4	214 (2023–2025)
FaaSTs	Research-Prototype	8700+	0	0	3	250+ (all-time)

4.4 Repository Index of NEARDATA Components

4.4.1 Production Components

Table 22: Repository Links for NEARDATA Production Components

Component / Tool	**Repository URL**
Lithops	`https://github.com/lithops-cloud/lithops`
SCONE	`https://sconedocs.github.io/`
Pravega	`https://github.com/pravega/pravega`
Metaspace	`https://github.com/metaspace2020/metaspace`
DataPlug	`https://github.com/CLOUDLAB-URV/dataplug`
DataCockpit	`https://github.com/CLOUDLAB-URV/data-cockpit`

4.4.2 Research Prototypes

Table 23: Repository Links for NEARDATA Research Prototypes

Research Prototype	Repository URL
Lithops-HPC	`https://github.com/neardata-eu/lithops-hpc`
Flower-Lithops Simulation Backend	`https://github.com/janprz/flower/tree/add-lithops-backend-support`
Serverless Vector DB	`https://github.com/neardata-eu/serverless-vdb`
FaaStream	`https://github.com/neardata-eu/FaaStream`
Nexus	`https://github.com/neardata-eu/nexus`
Glider	`https://github.com/CLOUDLAB-URV/glider-store`
Burst Computing	`https://github.com/Burst-Computing`
FaaSTs	`https://gitlab.bsc.es/datacentric-computing/lithops_telemetry_forecasting`
SCONE-Flower simulation	`https://github.com/neardata-eu/nct_tud_fl_demo`

5 Pipelines

This section describes representative data-processing pipelines that combine NEARDATA's production frameworks (Lithops, Scone, Pravega, Metaspace) with research prototypes (DataPlug, DataCockpit) to deliver end-to-end analytic workflows. For each pipeline, we detail:

- **Overview**: Scientific or functional objective.
- **Core Components**: NEARDATA modules employed.
- **Repository & Activity**: GitHub URL, commit count (2023–2025), contributors, stars and forks.
- **Documentation**: Available docs, unit tests, CI setup.
- **Reproducibility**: Invocation via PyRun.
- **Status**: Research prototype or production-ready.

5.1 Metabolomics (EMBL)

5.1.1 Metabolomics Annotation Pipeline

Overview: Executes the METASPACE spatial metabolomics annotation engine on cloud resources using Lithops for both serverless and hybrid compute.

Description: Lithops enables scalable, cost-efficient execution of METASPACE's annotation workflow on large imaging mass spectrometry datasets. METASPACE—developed by the Alexandrov group at EMBL Heidelberg—identifies metabolites and lipids in imzML data with confidence scores and false-discovery rates. By offloading compute to serverless functions (e.g. AWS Lambda, GCF, Azure Functions, IBM Cloud Functions, OpenWhisk) or VMs, the pipeline avoids multi–CPU-day runtimes and large local storage demands. Lithops auto-scales to hundreds or thousands of parallel invocations during highly parallel stages, then scales down for sequential steps, optimizing both performance and cost.

Core Components:

- **Lithops**: Distributes annotation tasks across cloud functions and VMs.
- **Metaspace**: Provides RESTful APIs for data ingestion and annotated-molecule retrieval.

- **PyRun**: Packages the workflow in a reproducible runtime, capturing dependencies, configuration, and outputs.

Implementations:

- Serverless Only: All stages run as Lithops functions on a FaaS backend.
- Hybrid: Memory-intensive stages on pre-provisioned VMs; parallel stages on serverless functions (IBM Cloud or on-premise).

Repository & Activity:

- URL: `https://github.com/metaspace2020/Lithops-METASPACE`
- Last update: March 19, 2021

Table 24: Metabolomics Annotation Pipeline Key Metrics by Period

Metric	Value	Period
Lines of Code	5,070	All time
Forks	4	All time
GitHub Stars	12	All time
Active Contributors	7	All time
Commits	288	All time

Documentation:

- Detailed `README` and Jupyter notebooks for both serverless and hybrid modes.
- Example configurations and step-by-step tutorials.
- No formal unit tests; validated via example runs and integrated with PyRun.

Status: Production-ready.

5.1.2 Off-Sample Detection Pipeline

Overview: Filters "off-sample" ion images—those not originating from the biological specimen—using ML and DL classifiers on METASPACE annotation outputs.

Description: This workflow ingests METASPACE's annotated ion images and applies a gold-standard dataset—curated by expert labeling—to train and evaluate models that distinguish on-sample versus off-sample spectra, improving downstream analysis accuracy.

Gold-Standard Creation:

- Retrieved ion images from public METASPACE datasets.
- Expert curators labeled images as on-sample or off-sample.
- Gold standard supports both classical ML and CNN approaches.

Model Development:

- Biclustering & Co-localization: Method from Ovchinnikova et al. (2020) to detect spatial off-sample artifacts.
- Deep Learning: CNN trained on the curated dataset for high-accuracy classification.

TagOff Web App:

- Front end built on the METASPACE webapp for manual curation.
- URL parameters: `db` (molecular database), `ds` (dataset timestamp), `max` (max images), `user` (annotator ID).
- Exports curated labels back into METASPACE's "Annotations" page.

Repository & Activity:

- URL: `https://github.com/metaspace2020/offsample`
- Last update: October 8, 2020

Table 25: Off-Sample Detection Pipeline Key Metrics by Period

Metric	Value	Period
Lines of Code	38,173	All time
Forks	0	All time
GitHub Stars	6	All time
Active Contributors	4	All time
Commits	52	All time

Documentation:

- Comprehensive `README`, plus notebooks for gold-standard creation, training, and evaluation.
- Automated tests for data loading and inference; CI ensures reproducibility.

Scientific Paper: [12] introduce an AI-driven workflow that recognizes and filters off-sample ion images in imaging MS data with F_1-scores up to 0.97.

Status: Research prototype.

5.2 Genomics

5.2.1 Genomic Analysis Pipeline (UKHSA)

Overview: Implements a serverless, scalable variant-calling workflow for whole-genome and targeted sequencing using Lithops on AWS Lambda. By partitioning FASTA/FASTQ files into fine-grained chunks, alignment and variant-calling tasks run in parallel without managing HPC clusters.

Description: Traditional HPC pipelines suffer queue delays and fixed capacity. Serverless computing provides instant elasticity and pay-per-use billing but requires careful data partitioning, I/O orchestration, and fault tolerance. This pipeline demonstrates migrating a complex bioinformatics workflow to FaaS with minimal developer effort while ensuring accuracy and reproducibility.

Core Components:

- **Lithops**: Orchestrates AWS Lambda invocations for preprocessing, alignment (BWA), sorting (SAMtools), and variant calling (FreeBayes).
- **S3**: Stores reference genomes, read chunks, and intermediate files.
- **PyRun**: Bundles the workflow in a versioned runtime for bit-for-bit reproducibility.

Datasets:

- **Trypanosoma brucei**: 35 MB FASTA (TritrypDB); reads SRR6052133 (668 MB).

- **Human (hg19)**: 905 MB FASTA GZip (UCSC); reads SRR15068323 (1.2 GB), ERR9856489 (12.1 GB).
- **Bos taurus**: 781 MB FASTA GZip (Ensembl); reads SRR934415 (16.5 GB, single mate).

Engineering Challenges:

- Partitioning: Balancing chunk size with Lambda limits (15 min, 10 GB RAM).
- I/O Orchestration: Reducing cold starts and S3 latency via pre-warmed Lambdas and tuned concurrency.
- Fault Tolerance: Idempotent tasks and retry logic for transient failures.

Repository & Activity:

- URL: `https://github.com/CLOUDLAB-URV/serverless-genomics-variant-calling`
- CI: GitHub Actions for linting, unit, and integration tests.
- Last update: May 23, 2024

Table 26: Genomics Analysis Pipeline (UKSHA) Key Metrics by Period

Metric	Value	Period
Lines of Code	4,325	All time
Forks	2	All time
GitHub Stars	0	All time
Active Contributors	5	All time
Commits	174	All time

Documentation:

- Detailed `README.md` with end-to-end examples and tuning tips.
- Docstrings in `serverlessgenomics/pipeline.py` for API reference.
- Published in WoSC '23: "Scaling a Variant Calling Genomics Pipeline with FaaS" (ACM DL: 10.1145/3631295.3631403).

Scientific Paper: The design and implementation of this serverless genomics pipeline are described in detail by Arjona et al. [13], highlighting data partitioning, I/O orchestration, and fault-tolerance challenges when migrating a variant-calling workflow to FaaS.
Status: Research prototype.

5.2.2 Genomic Analysis Pipeline (BSC)

Overview: Development of a high-performance processing pipeline for the discovery of variants interaction and its association with complex disorders

Description: Identifying associations between multiple genomic variants and specific disorders is essential for advancing our understanding of disease mechanisms. However, this task remains challenging due to the vast volume of data that must be simultaneously stored, processed, and analysed. To address this, the integration of scalable, dynamic, and distributed computational tools is critical for managing complex data analytics workloads efficiently. The proposed workflow ingests and partitions genomic datasets, enabling a machine learning-based approach to analyse the data in a combinatorial fashion. This orchestration facilitates parallel processing of data segments, thereby enhancing analytical throughput and precision.

Core Components:

- **MPI**: Parallelized HPC processing.
- **Lithops-HPC**: Serverless scalable platform adapted to run on HPC environments.
- **Dataplug**: Dynamic partitioning tool.
- **RabbitMQ**: Queue management system.
- **GekkoFS**: Lightweight, temporary distributed file system for high-throughput HPC.
- **Dask**: Parallel framework for distributed tasks, with a library to distribute XGBoost (Random forest algorithm) into multiple GPUs.
- **NVIDIA**: High-performance computing, deep learning, and AI workloads.

Datasets:

- **70KforT2D**: 1.285 TB; $1.5 * 10^6$ million variants; 22,802 paired case-control individuals
- **UK Biobank**: 3.286 TB; $1.5 * 10^6$ million variants; 55,438 paired case-control

Engineering Challenges:

- Partitioning: Shuffling the dataset into disordered partitions.
- I/O data ingestion: Effectively read extreme amounts of data and distribute it across readers in multiple nodes/cores.

Repository & Activity:

- URL: `https://gitlab.bsc.es/datacentric-computing/lithops-hpc-genomics`
  Last update: September 1st, 2025
- URL: `https://gitlab.bsc.es/datacentric-computing/lithops-hpc-examples`
  Last update: July 1st, 2025

Table 27: Genomics Analysis Pipeline (BSC) Key Metrics by Period

Metric	Value	Period
Lines of Code	8,916/1,998	All time
Forks	0/1	All time
GitHub Stars	0/1	All time
Active Contributors	6/5	All time
Commits	29/15	All time

Documentation:

- Detailed documentation on how to deploy Lithops-HPC is found in the repository `https://gitlab.bsc.es/datacentric-computing/lithops-hpc-examples`.
- The source code for the GWD use case is available in `https://gitlab.bsc.es/datacentric-computing/lithops-hpc-genomics`, explaining how to launch.
- The source code for MDR use case with Ltihops-HPC is available in `https://gitlab.bsc.es/datacentric-computing/mpi-genomics-mdr` along with its documentation.

- The scientific method pre-print is published in MedRxv: "Genetic Profiling and Early Detection of Type 2 Diabetes Subtypes through Sex-Stratified GWAS and Explainable AI" (DOI: https://doi.org/10.1101/2025.07.24.25332120)

Scientific Paper: The design and initial implementation of Lithops-HPC is available by [6], explaining the adaption process of the Lithops framework into supercomputers that resulted into the Lithops-HPC architecture. On the other hand, [14] explains the biological side of the use-case.

Status: Research prototype.

5.2.3 Genomic Analysis Pipeline (Sano)

Overview: Implements a resource-intensive pipeline in the cloud in order to create the Transcriptomics Atlas by processing publicly available RNA-sequencing data from NCBI-NIH/ENA.
Description: The goal of this use case is to create the **Transcriptomics Atlas**, in which data from a representative set of human tissues were uniformly processed. This resource can be of use in a wide range of scientific applications. The pipeline which is designed for this task consists of accessing FASTQ data, alignment of RNA sequences, and a normalization step. Alignment is expensive in terms of compute resources and time. Processing hundreds of terabytes of such data is challenging and requires a cost-efficient approach and thoughtful optimizations.
Core Components:

- AWS services: EC2, S3, Lambda, ECS (Fargate mode), DynamoDB, SQS
- Terraform - Management of cloud infrastructure
- Boto3 - Python library for AWS services
- Lithops - For serverless processing of sequences with Salmon-based pipeline

Datasets:

- NCBI NIH Sequence Read Archive - largest public biological data repositories, containing petabytes of sequencing data from millions of experiments (part of International Nucleotide Sequence Database Collaboration (INSDC)).

Engineering Challenges:

- Resource-intensive application - STAR aligner (requires over 30GB of memory within node and is CPU-intensive)
- Human genome index distribution to nodes
- Cost-efficient processing of large dataset

Repository & Activity:

- URL: `https://github.com/neardata-eu/transcriptomics-atlas-sano`
  Last update: September 30th, 2025
- URL: `https://github.com/neardata-eu/salmonless`
  Last update: July 7th, 2025

Table 28: Genomics Analysis Pipeline (Sano) Key Metrics by Period

Metric	Value	Period
Lines of Code	10,427/8,729	All time
Forks	0/0	All time
GitHub Stars	0/0	All time
Active Contributors	1/1	All time
Commits	273/13	All time

Documentation:

- 4 publications about Transcriptomics Atlas
  - *Optimizing Star Aligner for High Throughput Computing in the Cloud* - [15]
  - *Serverless Approach to Running Resource-Intensive STAR Aligner* - [16]
  - *Accelerating Cloud-Based Transcriptomics: Performance Analysis and Optimization of the STAR Aligner Workflow* - [17]
  - *Novel Approaches Toward Scalable Composable Workflows in Hyper-Heterogeneous Computing Environments* - [18]
- Comprehensive `README`
- Unit tests in Docker

Status: In Development.

5.3 Surgery (NCT)

5.3.1 Semantic Video Search

Overview: A policy-driven semantic video search solution that exploits tiered storage in streaming systems.

Description: Streaming systems are an increasingly appealing substrate for managing video data via stream abstraction. However, if we consider a large stream collection, it can be hard for data scientists to discover and locate relevant videos, let alone specific video fragments. Knowing this problem, StreamSense is proposed, which allows users to deploy AI models that generate embeddings from video frames via policies.

Core Components:

- **Pravega**: Handles video stream ingestion and provides real-time and historical access to frames.
- **GStream**: Provides a mechanism to decode and process video streams in realtime.
- **Milvus**: Stores embeddings of key frames in two separate collections: a global one (containing a fraction of the frames) and a specific one (containing all key frames). This structure enables both intravideo and intervideo similarity search.

Repository & Activity:

- URL: `https://github.com/neardata-eu/video-stream-indexing`
- Last update: July 14, 2025

Table 29: Surgery (NCT): Semantic Video Search Key Metrics by Period

Metric	Value	Period
Lines of Code	6,887	All time
Forks	0	All time
GitHub Stars	0	All time
Active Contributors	2	All time
Commits	97	All time

Documentation:

- Comprehensive `README` and detailed paper published in `https://dl.acm.org/doi/10.1145/3700824.3701097`.
- Video demos in /media.
- No formal unit tests; validated via example.

Status: Production-ready.

5.3.2 Federated Learning within SGX Enclaves Using SCONE (TUD)

Overview: A secure and confidential federated learning solution leveraging SCONE and Trusted Execution Environments (TEEs) to protect computation and data in adversarial settings.

Description: Federated Learning (FL) enables collaborative model training while keeping raw data confidential. This project integrates SCONE to run FL inside Intel SGX enclaves, using dynamic attestation, network/file shielding, and confidential orchestration. Memory inspection experiments confirm that sensitive data remains protected even against privileged attackers. Performance overhead primarily arises from enclave startup, which is mitigated for longer-running workloads.

Core Components:

- **Federated Learning Application:** Python-based client/server FL logic.
- **SCONE Runtime:** Provides TEE support and enclaves for confidential execution.
- **SCONE CAS (Configuration & Attestation Service):** Manages network shielding and attestation.
- **Kubernetes Orchestration:** Deploys and scales FL workloads securely with confidential orchestration.
- **Security Features:**
  - Network Shield: Encrypts and authenticates network traffic between FL clients and server enclaves.
  - File System Shield: Encrypts training data at rest with dynamic key rotation.
  - Dynamic Attestation: Continuous verification of enclave integrity and policy compliance.
  - Confidential Orchestration: Ensures deployment metadata and secrets remain inside enclaves.

Repository & Activity:

- URL: `https://github.com/neardata-eu/scone-artifacts/tree/main/images/flowerml/FederatedChal`
- Last update: July 17, 2025

Documentation:

- SCONE official documentation: `https://sconedocs.github.io/`

Table 30: Federated Learning within SGX Enclaves Using SCONE - Key Metrics

Metric	Value / Note	Period
Lines of Code	808 *Includes Python scripts, configuration files, for setting up and validating the federated learning workflow within SCONE.*	All time
Forks	0	All time
GitHub Stars	0	All time
Active Contributors	2	All time
Commits	54	All time

Status: Operational and validated. Security experiments confirm confidentiality and integrity.

6 Conclusions

The NEARDATA project has resulted in the creation of an extensive software ecosystem that collectively embodies the principles of high-performance near-data computing, secure orchestration, and reproducible research. Spanning more than a dozen software components, the project's outcomes cover the full data lifecycle—from acquisition and transformation to analysis, visualization, and governance—thus enabling a unified approach to managing and processing extreme, distributed, and heterogeneous datasets.

A central contribution is the establishment of the **Extreme Data Hub** as a practical and extensible construct realized through modular components such as **DataPlug**, **DataCockpit**, and **PyRun**. PyRun in particular plays a pivotal role by acting as a user-facing frontend that integrates and exposes many of the project's core results, including pipelines developed with **Lithops**, **Lithops-HPC**, **Burst Computing**, and other backend systems. Through this unified interface, researchers and practitioners can execute, monitor, and reproduce data-driven workflows across heterogeneous infrastructures, bridging experimental research and operational deployment.

The software stack demonstrates a strong emphasis on **reproducibility and transparency**. Each component has been released in public repositories with clear documentation, containerized runtimes, and reproducible deployment scripts. The integration of continuous integration (CI) pipelines, Docker-based environments, and version-controlled configurations ensures that computational experiments can be re-executed deterministically, thus reinforcing the scientific rigor of the project's results. In addition, PyRun provides a reproducibility layer by standardizing how runtimes, data connectors, and pipelines are executed across multiple infrastructures.

Another distinguishing achievement of NEARDATA is the coexistence of **production-grade frameworks** and **research prototypes** within a single architectural vision. Mature components such as **Lithops**, **Pravega**, **SCONE**, and **Metaspace** are already maintained by active open-source communities and deployed in real production settings. These communities contribute continuously to code maintenance, bug resolution, and new feature development, ensuring long-term sustainability beyond the project's funding period.

Complementing these mature systems, several **research prototypes**—including **Nexus**, **FaaStream**, **Glider**, **Burst Computing**, and **Lithops-HPC**—have been developed as exploratory platforms to validate new paradigms such as serverless HPC, streaming-based orchestration, and cross-cloud near-data computing. These prototypes have been validated experimentally in publications and integrated where possible into PyRun, allowing them to be executed alongside production-ready components.

The project's approach has also fostered significant synergies with other research projects in the DataNexus cluster and with URV's CLOUDLESS project, where shared development efforts have strengthened PyRun's orchestration layer and runtime management capabilities. This collaboration has yielded a unified, scalable, and multi-cloud execution framework that now serves as a reference implementation, demonstrating the benefits of cross-project integration in European research initiatives.

In summary, NEARDATA delivers a comprehensive portfolio of interoperable software artifacts that collectively advance the state of the art in extreme data management and confidential computing. These software outcomes are:

- **Open and Reproducible:** Each component is publicly accessible, version-controlled, and documented to ensure transparent reproducibility of results.
- **Integrated and Interoperable:** Through PyRun and shared data connectors, components can be composed into reproducible end-to-end pipelines across cloud, edge, and HPC infrastructures.
- **Sustainable:** Production-ready frameworks are supported by active open-source communities, ensuring ongoing maintenance and growth.
- **Innovative:** Research prototypes explore novel directions in near-data execution, serverless elasticity, and cross-cloud orchestration.

The result is a robust, modular, and extensible software foundation that not only fulfills NEARDATA's objectives but also lays the groundwork for future research and industrial exploitation in large-scale distributed analytics, secure data sharing, and AI-driven cloud orchestration.

References

[1] Daniel Alejandro Coll Tejeda. Beyond 'hello world': Powering real-world science and ai with pyrun. Medium blog post, July 2025. URL `https://danielalecoll.medium.com/beyond-hello-world-powering-real-world-science-and-ai-with-pyrun-4a5878278ddf`.

[2] Daniel Alejandro Coll Tejeda. Effortless serverless python: Get your code running in the cloud in 3 clicks with pyrun. Medium blog post, July 2025. URL `https://danielalecoll.medium.com/effortless-serverless-python-get-your-code-running-in-the-cloud-in-3-clicks-with-pyrun-4c3`

[3] Daniel Alejandro Coll Tejeda. Processing cloud-optimized data in python with serverless functions (lithops, dataplug). Medium blog post, March 2025. URL `https://medium.com/@danielalecoll/processing-cloud-optimized-data-in-python-with-serverless-functions-lithops-dataplug-c6e6f`

[4] Daniel Alejandro Coll Tejeda. How to run lithops over ec2 vms using the new kubernetes backend. Medium blog post, February 2024. URL `https://medium.com/@danielalecoll/how-to-run-lithops-over-ec2-vms-using-the-new-k8s-backend-4b0a4377c4e9`.

[5] Germán T. Eizaguirre, Daniel Barcelona-Pons, Aitor Arjona, Gil Vernik, Pedro López García, and Theodore Alexandrov. Serverful functions: Leveraging servers in complex serverless workflows. Zenodo, 2025. URL `https://zenodo.org/records/15019182`. Industry Track.

[6] Andres Benavides Arevalo, Daniel Coll Tejeda, Aaron Call, Pedro García López, and Ramon Nou Castell. Enhancing hpc with serverless computing: Lithops on marenostrum5. In 2024 IEEE 32nd International Conference on Network Protocols (ICNP), pages 1–6, 2024. doi: 10.1109/ICNP61940.2024.10858564.

[7] Germán T. Eizaguirre and Marc Sánchez-Artigas. A seer knows best: Auto-tuned object storage shuffling for serverless analytics. Journal of Parallel and Distributed Computing, 183:104763, 2024. URL `https://doi.org/10.1016/j.future.2023.05.006`.

[8] Josep Sampé, Marc Sánchez-Artigas, Gil Vernik, Ido Yehekzel, and Pedro García-López. Outsourcing data processing jobs with lithops. IEEE Transactions on Cloud Computing, 11(1):1026–1037, January 2023. ISSN 2168-7161. doi: 10.1109/TCC.2021.3129000.

[9] Bishoy Wadie, Lachlan Stuart, Christopher M. Rath, Bernhard Drotleff, Sergii Mamedov, and Theodore Alexandrov. Metaspace-ml: Context-specific metabolite annotation for imaging mass spectrometry using machine learning. Nature Communications, 15(9110), 2024. doi: 10.1038/s41467-024-52213-9. URL https://www.nature.com/articles/s41467-024-52213-9.

[10] Aitor Arjona, Pedro García-López, and Daniel Barcelona-Pons. Dataplug: Unlocking extreme data analytics with on-the-fly dynamic partitioning of unstructured data. In Proceedings of the 24th IEEE International Symposium on Cluster, Cloud and Internet Computing (CCGrid). IEEE/ACM, 2024. doi: 10.5281/zenodo.14161823. URL https://zenodo.org/records/14161823.

[11] Daniel Barcelona-Pons, Pedro García-López, and Bernard Metzler. Glider: Serverless ephemeral stateful near-data computation. In Proceedings of the 24th ACM/IFIP International Middleware Conference, Middleware '23, page 15–27, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400704351. doi: 10.1145/3590140.3629119. URL https://doi.org/10.1145/3590140.3629119.

[12] Katja Ovchinnikova, Vitaly Kovalev, Lachlan Stuart, and Theodore Alexandrov. Offsampleai: Artificial intelligence approach to recognize off-sample mass spectrometry images. BMC Bioinformatics, 21(1):129, 2020. doi: 10.1186/s12859-020-3425-x. URL https://bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-020-3425-x.

[13] Aitor Arjona, Arnau Gabriel-Atienza, Sara Lanuza-Orna, Xavier Roca-Canals, Ayman Bourramouss, Tyler K. Chafin, Lucio Marcello, Paolo Ribeca, and Pedro García López. Scaling a variant calling genomics pipeline with faas. In Proceedings of the 9th International Workshop on Serverless Computing (WoSC '23), pages 59–64. ACM, 2023. doi: 10.1145/3631295.3631403. URL https://dl.acm.org/doi/10.1145/3631295.3631403.

[14] Lorena Alonso-Parrilla, Miguel Ángel Pérez-Elena, Mohammed Yousef Salem Ali, Maedeh Mashhadikhan, Nicolás Gaitán, Leila Satari, Rodrigo Martín, Anthony Piron, Xavier Farré, Natalia Blay, Lydia Ruiz, Aikaterini Lymperidou, Cecilia Salvoro, Rafael de Cid, Josep Lluís Berral, Juan R González, Ignasi Morán, Miriam Cnop, and David Torrents. Genetic profiling and early detection of type 2 diabetes subtypes through sex-stratified gwas and explainable ai. medRxiv, 2025. doi: 10.1101/2025.07.24.25332120. URL https://www.medrxiv.org/content/early/2025/07/24/2025.07.24.25332120.

[15] Piotr Kica, Sabina Lichołai, Michał Orzechowski, and Maciej Malawski. Optimizing Star Aligner for High Throughput Computing in the Cloud. In 2024 IEEE International Conference on Cluster Computing Workshops (CLUSTER Workshops), pages 162–163. IEEE, 2024. doi: 10.1109/CLUSTERWorkshops61563.2024.00039.

[16] Piotr Kica, Michał Orzechowski, and Maciej Malawski. Serverless approach to running resource-intensive star aligner. In 2025 IEEE 25th International Symposium on Cluster, Cloud and Internet Computing Workshops (CCGridW), pages 1–4. IEEE, 2025.

[17] Piotr Kica, Sabina Lichołai, Michał Orzechowski, and Maciej Malawski. Accelerating cloud-based transcriptomics: Performance analysis and optimization of the star aligner workflow. In International Conference on Computational Science, pages 257–265. Springer, 2025.

[18] Jonathan Bader, Jim Belak, Matthew Bement, Matthew Berry, Robert Carson, Daniela Cassol, Stephen Chan, John Coleman, Kastan Day, Alejandro Duque, et al. Novel approaches toward scalable composable workflows in hyper-heterogeneous computing environments. In

Proceedings of the SC'23 Workshops of the International Conference on High Performance Computing, Network, Storage, and Analysis, pages 2097–2108, 2023.