Deliverable

D2.12 Technical development of prototype big data solutions

Deliverable information	
Work package	WP2
Lead	INGV, GFZ, KNMI on behalf of ORFEUS
Authors	Peter Danecek, Alberto Michelini, Javier Quinteros, Reinoud Sleeman, Carlo Cauzzi
Reviewers	Ian Main as WP2 Leader
Approval	[Management Board]
Status	Final
Dissemination level	[Public]
Delivery deadline	[28.02.2022]
Submission date	First submission - [22.02.2022]
Intranet path	[DOCUMENTS/DELIVERABLES/File Name]

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 821115

Table of contents

Summary

This report accompanies the technical implementation of selected strategies for big-data management anticipated in RISE Deliverable D2.11, namely:

- SeiSpark, that allows access and processing of massive datasets by creating a "computational archive" where storage resources and computational resources converge;
- dastools, that allows converting DAS data into standardised seismological data formats hence making the waveform data and metadata easily available in current seismological archives;
- the cloud strategy implemented by the ORFEUS Data Center at KNMI, that allows operating more efficiently than maintaining an in-house infrastructure; this was already established when D2.11 was compiled, and is described therein.

1. SeiSpark demonstrator

1.1 Introduction

The Istituto Nazionale di Geofisica e Vulcanologia (INGV) is a core participant of ORFEUS and operates the national seismological data archive with a regional focus on Italy and further networks or stations from the Mediterranean area, which also constitutes one of the nodes of the European Integrated Data Archive (EIDA) federation. This node provides access to the continuous seismological records through standardized protocols and formats. The data amounts to a volume of more than 115 TB of seismological waveform data from more than 1000 stations contributed by 15 permanent networks and from a steadily growing number of temporary deployments. The infrastructure for archiving and distribution of these data is well-established, functional and robust.

However, with this well-established standardized method of access it is getting increasingly more challenging to make good use of a substantial amount of the data assets. The currently established workflows require users to download the dataset of interest – usually a rather small portion of the archive – and to process them on locally procured resources. This approach gets at its limits when a more significant portion of the data holdings should be processed, both for the user and for the datacentre. The user would need to ensure adequate local resources for storage, I/O capacity and computation, while the datacentre would be challenged by the immense amount of data requests and the required network bandwidth for which the infrastructure and the provided services were not designed and implemented.

As documented in RISE Deliverable D2.11., the SeiSpark project was started at INGV with the goal to establish a completely different paradigm for the processing of the archived data, in particular by exploiting the concept of data locality and by avoiding unnecessary transfers of data on various levels as much as possible. Instead, it would become necessary to move the processing towards the place where data reside. This implies the addition of significant computational resources and an adequate framework to the seismological data archive, creating a "computational archive" where storage resources and computational resources converge. The proposed processing framework leverages on existing solutions from the Big Data ecosystem and combines them with the popular open-source scientific Python framework which is well established in seismological research.

The general guiding principles and key design criteria of the SeiSpark project can be summarized as follows:

- Applying well-established concepts from parallel computing and optimization strategies through data reusage on node level and in-memory;
- Levering on existing, well-established open-source solutions in order to limit maintenance effort and enhance long-term sustainability;

- Providing access to modern and well-known Python based seismological software for processing, analysis and visualization, to facilitate adoption by target users;
- Avoiding any need for a data transformation process to make the data accessible;
- Facilitate interactive exploration of datasets or testing of concepts for data analysis through the adoption of user-friendly interfaces;
- Ensuring immediate access to very large seismological waveform datasets, like the whole data holdings from INGV's EIDA node through pre-staging on the platform;
- Keeping the final platform sufficiently flexible and versatile, so that adoption of alternative data frameworks or accounting for new requirements remain simple;
- Planning for a scale out strategy: a realistic scenario might be to have moderate scale in-house resources and the possibility to scale out to scientific computing centres or to public cloud providers.

1.2 Implementation

The general concept of SeiSpark was initially tested and evaluated by repurposing and upgrading already available hardware and by implementing a first pilot platform. The platform was based on Apache Hadoop, HDFS, YARN and Spark, installed directly on the bare servers. The dated hardware did not provide any possibility to evaluate also more recent concepts of virtualization and modern deployment strategies. The Apache HDFS provided the distributed Object storage which stores the seismological archive in its original format and layout, thereby avoiding costly transformation processes at data ingestion. The Apache Spark framework was combined with a customized Python environment which provides the domain specific functionally and various procedures to access and process the archived data have been developed and examined. **This pilot implementation was described with further details in RISE Deliverable D2.11.**

Through funding from a national infrastructure project, it was possible to implement this conceptual setup on modern hardware, thereby unlocking the real potential of such infrastructure. The major objectives we intended to achieve with the investment were:

- increased dimension of storage capacity, thereby offering the possibility to hold the complete seismological archive of the Italian EIDA node, any other relevant datasets as well as intermediate or finale results of processing;
- current and performant, yet energy efficient, storage and processing hardware which was chosen and optimized for the specific application and makes it possible to appreciate the scaling performance and throughput in day-to-day processing tasks;
- design that puts particular emphasis and on increased amount of processing memory and high memory bandwidth in order to support best in-memory processing performance, and on top-tier networking performance to ensure good I/O performance during dataset shuffle operations;
- flexibility, manageability and continues evolution obtained by exploiting recent technologies in containerisation, automized deployment and management of containerised application, supervisor and orchestration.

After a tender process has been concluded in 2021, the cluster hardware was finally installed on the premises of INGV at Rome at the beginning of 2022. Subsequently, a software project (as part of the contract) with the scope of implementing the basic software stack was defined and executed in collaboration of the supplier. The software project was concluded in September 2022 with a hand-over training and with joint acceptance testing. More recent activity is related to the further integration of the SeiSpark platform with the datacentre of the EIDA node and institution's IT infrastructure, including an initial transfer of the data archive on the SeiSpark object storages.

The installed software stack functionally replicates mostly the latest iteration of the one developed and deployed on the pilot infrastructure. However, an important difference is the addition of Kubernetes, which provides the flexibility and manageability of containerized application management and deployment. A substantial part of the software components is now deployed on the cluster on top of Kubernetes as containerized applications.

1.3 Hardware

The cluster consists of two different types of nodes, which are configured in a very different way. The three Master Nodes serve to host for various components and tasks which support the cluster's resource management and coordination. These hold various status and metadata information, and in particular these host the name nodes of the HDFS object storage. The number of three nodes is given by the need to ensure redundancy and fail-over functionalities. The eight converged Worker Nodes unify two relevant roles, storage nodes and computational worker, and provide the relevant usable resources of the cluster. These nodes are therefore equipped with significantly more computational and storage resources. In order to ensure high IO performance when accessing the distributed storage or during data shuffle operations the cluster is also provisioned with a low latency high-performance interconnect, in addition to a general-purpose service network implemented as a redundant high bandwidth Ethernet and a dedicated network for management.

The minimum requirements and characteristics requested by the tender have been exceeds by the delivered hardware in several aspects, notably with respect to storage size, processing memory and networking performance. The details of the installed hardware are reported below.

Auxiliary Master Nodes:
- Dell EMC PowerEdge R7525
- CPU: 2x 16 core; AMD Epyc 7302, 3.0 GHz base clock
- RAM: 256 GB; DDR4
- Mem bandwidth: 204.8 GB/s /CPU
- System disc: 2 x 960 GB SATA SSD, 6 Gbit/s
- Mellanox ConnectX-5 Dual Port 10/25GbE
- Mellanox ConnectX-6 Single Port HDR 200 GbE VPI Infiniband

Converged Worker Nodes:
- Dell EMC PowerEdge R7525
- CPU: 2x 24 core; AMD Epyc 7352, 2.3 GHz base clock
- RAM: 1024 GB; DDR4
- Mem bandwidth: 204.8 GB/s /CPU
- System disc: 2 x 960 GB SATA SSD, 6 Gbit/s
- SAS SSD: 2x 3,840 GB
- SAS HDD: 10x 18,000 GB
- Transfer rate: 12 Gbit/s
- Mellanox ConnectX-5 Dual Port 10/25GbE
- Mellanox ConnectX-6 Single Port HDR 200 GbE VPI Infiniband

Networking:
- High-performance interconnect: InfiniBand 200 Gb/s
  40-port Mellanox Quantum QM8700 Switch
- Service network (redundant): Ethernet 25 Gb/s
  Dell EMC Switch S4128F-ON, 28 x 10GbE, 2 units
- Management network: Ethernet 1 Gb/s
  Dell PowerSwitch S3124, L3, 24x 1GbE, 2xCombo, 2x 10GbE

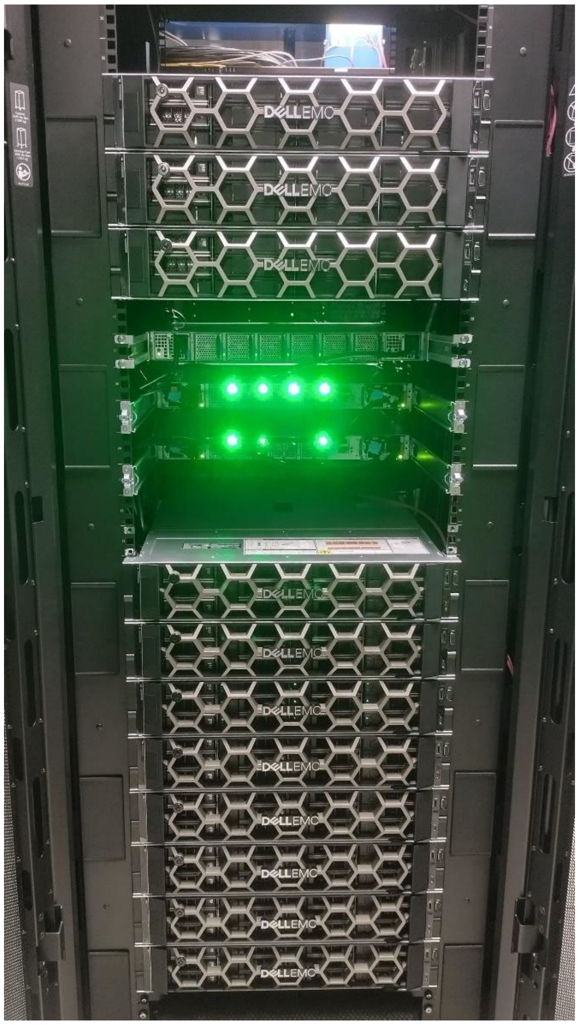


Figure 1: The cluster installed in a single full-scale rack, the master nodes are positioned in the upper part, worker nodes in the lower part.

1.4 Software

One important target for the implementation of the new SeiSpark production platform was to increase manageability, flexibility, and to facilitate the continues evolution of the platform. For this reason, Kubernetes was introduced to the updated software project. It allows to exploit virtualization and provides the management, deployment, running and scaling of software components as containerisation Kubernetes applications on a *virtual* cluster. Kubernetes assumes the function of the resource management of the cluster with respect to computational resources and replaces the YARN resource manager from the Apache ecosystem. A relevant part of the software components of SeiSpark is now deployed on the cluster as Kubernetes applications. One important exception to this, are the software components related to the Apache Hadoop cluster which serve principally the scope of providing the HDFS object storage and the management of these storage resources for the massive data repository. These components have been installed directly on the bare server hardware and not as containerised Kubernetes applications. Also, some basic components of the Kubernetes cluster and the supporting load-balancer reside directly on the bare server hardware.

The fact that the two types of nodes assume fundamentally different roles and tasks, both for the implementation of the Apache Hadoop cluster as well as for the Kubernetes cluster, has its correspondence in the set of software components which are installed on the respective set of nodes. Basically, all control, management, metadata information and state keeping is the task of the Master nodes (3 units), while the Worker nodes (8 units) provide the massive storage resources and the computational resources. The massive storage consists of large volume mechanical disks and is managed by Apache Hadoop as JBOD (just a bunch of disks) providing an HDFS distributed object storage. A smaller set of solid-state disks is managed by Longhorn software which provides a distributed block storage for the persistent volumes for the Kubernetes cluster. The worker node memory and CPU resources are managed by Kubernetes and assigned dynamically to containerized applications.

Software installed and the physical server hardware (bare metal)

Various software components are installed directly on server hardware. These basic software components are part of the main building of the basic infrastructure for management and attribution of the cluster's resources to higher-level applications and software components. In particular these can be summarised in these categories:

- Operating system
- Basic libraries or dependencies of the following
- Load balancer (master nodes only)
- Apache Hadoop cluster and the HDFS object storage
- Fundamental components of Kubernetes cluster

<u>Operating system & basic libraries:</u>
As operating system Linux Ubuntu 20.04.4 LTS has been chosen, simply because it is the distribution wildly used for the systems of the seismological datacentre. Various basic libraries were installed as dependencies of other packages e.g. Java OpenJDK, 1.8.0.

<u>Load balancer:</u>
In order to ensure the functionality of load balancing and high-availability failover the following software components were deployed and configured:

- Keepalived
- Ha Proxy

<u>Apache Hadoop cluster:</u>
Installation and management of the Apache Hadoop components is based on a packaging project, Apache Bigtop, version 3.0.1 was deployed. Apache Bigtop packaging should ensure that the various components of the Apache ecosystem are compatible among each other. The following components have been installed using the Apache Bigtop on the respective node types:

Master nodes
- Zookeeper
- NameNode (active-passive-passive)
- Journal Node (quorum journal manager, used to sync information from active name nodes to the passive ones)
- ZKFailoverController

Worker nodes
- DataNode

Apache Ranger was deployed on one of the master nodes as an independent package:
- Ranger Policy Admin (manages access policies)
- Ranger Usersync (LDAP Server integration)
- Apache Solr (in Solrcloud mode, storing and searching of Ranger audit logs)

- Postgresql versione 12 (backend SQL database for Ranger metadata)

Kubernetes cluster:
Kubernetes version 1.24.1, basic components of a Kubernetes cluster were installed on Master and Worker nodes as detailed below.

Master nodes / Control Plane:
- Api Server
- Controller-manager
- Scheduler
- Etcd

Worker nodes:
- Kubelet
- Kubeproxy
- Container runtime (containerd)

Software installed as Kubernetes applications

Software installed as Kubernetes applications includes:

Additional components of the Kubernetes cluster itself
- Persistent storage: Longhorn
- Kubernetes Metrics Server
- Kubernetes Dashboard

Jupyter related components
- JupyterHub: Point of access, login, creates on first login a persistent volume (Longhorn) for the user as personal workspace, the user then obtains a JupyterLab instance;
- JupyterLab: deployed and launched on user login, default GUI; access to personal workspace;
- Jupyter Enterprise Gateway: Manages kernels, provides info to JupyterLab, deploys JupyterLab kernels (based on public or customized images) on the cluster ;
- Private repository-k8s: used for customized images
  Note: requires two persistent volumes, for X.509 certificate and for docker images

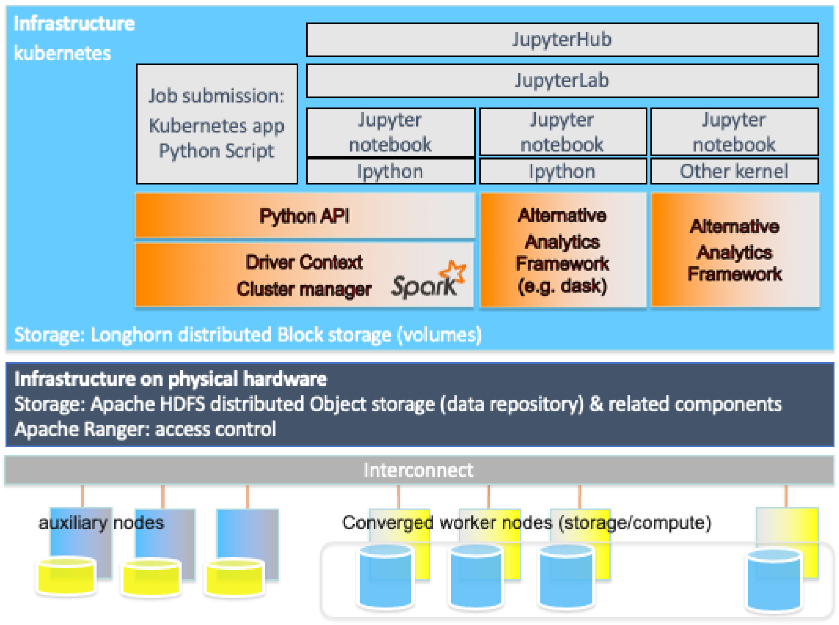

Infrastructure
kubernetes
JupyterHub
JupyterLab
Job submission:
Kubernetes app
Python Script
Jupyter notebook
Jupyter notebook
Jupyter notebook
Ipython
Ipython
Other kernel
Python API
Driver Context
Cluster manager
Spark
Alternative Analytics Framework (e.g. dask)
Alternative Analytics Framework
Storage: Longhorn distributed Block storage (volumes)
Infrastructure on physical hardware
Storage: Apache HDFS distributed Object storage (data repository) & related components
Apache Ranger: access control
Interconnect
auxiliary nodes
Converged worker nodes (storage/compute)

Figure 2: A schematic simplified setup of SeiSpark.

1.5 Status & Outlook

Current status

Current activity regards mainly the further integration of the SeiSpark cluster with other infrastructure of the seismological datacentre and with the data archive. This concerns several aspects, will lead to improved day-to-day usability of the SeiSpark platform and sets the basis for the full integration of the platform with various non-mission-critical routines and workflows. **These works include user management, integrated and unified user workspaces, data transferring and interoperability with other tools. The archived seismological waveform data have been transferred to the HDFS object storages and the procedures for the daily synchronous alignment with the consolidated data archive will be integrated into the newly developed rule-based workflow engine.**

The SeiSpark project is designed to provide two different approaches for usage of the SeiSpark platform for data access and data analysis:

1. **Interactive usage** allows for immediate and natural exploration of datasets and user-guided development of understanding of data, ideas, or processing approaches which subsequently can be transformed or integrated into more complete and complex applications. The tool of choice is the JupyterLab GUI which provides an environment already well-known and established among the typical users, completed with various preconfigured Jupyter kernels. These kernels can be further customised towards the requirements of the datacentres or of the users, without creating compatibility problems with existing applications. One such customized kernel replicates an environment, the modules and functionalities similar to the SeiSpark pilot platform.

2. **Batch processing** allows to submit jobs with consolidated, mature and potentially long-running applications. Such Spark applications are of particular interest for routine datacentre processing and are launched by submitting a job to the Kubernetes system where they run as containerized application on a dynamically created virtual cluster. As in the case of interactive Jupyter kernels the container images for the Spark environment which runs on the cluster can be highly customized and allow for continues evolution according to upcoming requirements.

Testing

The interactive usage mode has been tested with some of the Jupyter notebooks which were developed during the pilot phase and have been used to define the basic requirements for the updated platform. The tests have been run against a customized kernel image which includes among others the Obspy package and other packages for visualization.

We are currently carrying out testing in order to develop a general understanding of various parameters or trade-offs which can affect performance, to optimize the setup or to provide users better possible guidance on starting points for parameter and application setups. In order to test scaling behaviour and the influence of possible choices we perform most of these tests as batch jobs. This also provides the possibility to scale the problem size already to substantial datasets, i.e. nearly 10% of the volume of the consolidated seismological data archive.

The major part of scaling tests is performed on a modest dataset yet representative for typical applications, i.e., one year (2021) of data for MedNet, *Mediterranean Very Broadband Seismographic Network*, network code MN, and all main streams. This amounts to approximately 0.25 TB of data. In addition, large-scale tests are performed on a dataset larger than 10 TB. It is one complete year of data for the INGV's major network*, Italian National Seismic Network*, network code IV, and in this case all available data streams, including subsampling, are considered. This type of access pattern is typical for datacentre operations when characterizing the archived waveform data.

The representative test problem is kept intentionally simple, but it ensures that the entire test dataset is required to be read completely and used for the calculation of a characteristic value. This resembles a typical embarrassingly parallel access pattern, where data needs to be accessed from storage, elaborated independently for each single data element. The result, a characteristic value for each file, is inserted into a tabular data type (DataFrame) and the table finally written in parallel to the HDFS object storage. The problem is therefore mostly I/O bound and has the primary objective to assess achievable data throughput and to check for correct functioning of the cluster setup and for the absence of major bottlenecks or configuration issues.

The scaling and parameter tests provide some first insights and understanding of reasonable compromises for the virtual cluster sizing and parallel threads of executor pods. The first observations indicate the following: Although larger cluster sizes reduce overall processing time, the marginal parallel gain decreases, so too large clusters do not represent the most cost-efficient setup. With excessively large cluster sizes, the overall processing may increase due to the costs of cluster setup and limited parallel gain. The observation regarding executor parallelism apparently levels off at around 4-6 virtual cores, which is well in line with documented behaviour or recommended values. The memory attributed to the driver pod need to be sufficiently large to support some of the operations, but otherwise memory seems not to be particularly critical for the described test. The driver pod is assigned 20 GB of RAM and executor pods consume 8 GB of RAM each.

For the moderate dataset tests, execution time can easily be reduced to the 80-60 seconds range, though it probably does not represent the most cost-efficient setup. A reasonable compromise is rather in the 360-120 seconds range and significantly lower resource costs (8 to 32 virtual cores). On the other hand, the large-scale tests could be executed with runtimes down to 80 minutes (256 or 512 cores), and again more cost-efficient setups would lay in the range of execution times around 120 minutes with only 64 virtual cores required.

Limitations and next steps

Current activity seeks further improvements and optimization the system's setup as well as the user experience. This regards mainly aspects of the management of users and resources and is guided by some limitations we observe with the current implementation and which limit a wider adoption of the platform. **None of these details affects the processing capacities of the infrastructure but regard practicalities in the day-by-day usage and management.**

User authentication

We currently lack a more robust system for user authentication and thereby have only limited possibilities to enforce policies on resource assignment and access. This regards both storage resources and the computational resources. We therefore working on the integration of JupyterHub with an institute wide authentication system. We then need to ensure that these assured identities are propagated correctly or recognized across the varies components of the system.

Workspace visibility

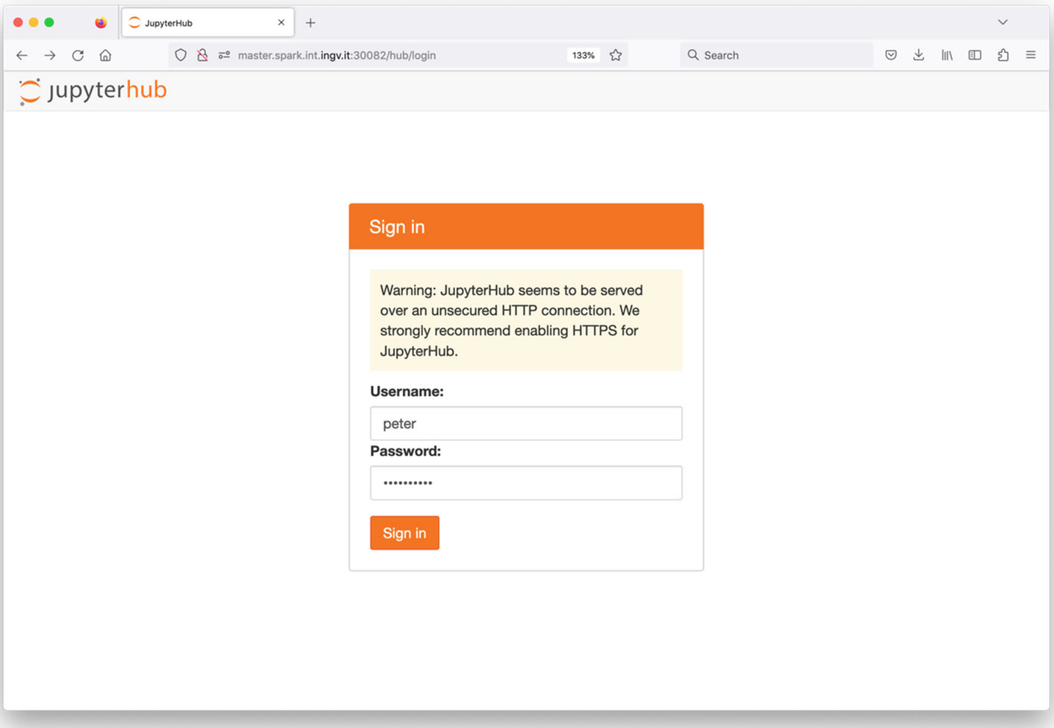
JupyterHub already provides some method to create a persistent volume for each user's workspace. This workspace is visible to the JupyterLab GUI environment and is the place where the user's notebooks, scripts or other files can be stored. However, we are currently examining how to make this user's workspace available in the Jupyter kernels themselves, both Spark driver and executors. This is currently conditioning the flexibility of the overall system and make the user experience somewhat cumbersome. We therefore work on adapting the kernel's image, configuration or scripts in order to enable the mount the user's workspace volume also inside the driver and executer pods of the Spark kernels. We also would prefer to provide the users with a unified environment both for interactive work and for batch submission where they have access to the same workspace, scripts, user modules, small configuration or data files.

Access to physical host

Currently batch job submission requires access to one of the physical hosts of the cluster, from which the jobs are submitted to Kubernetes via spark-submit. However, this setup adds overall complexity in managing user accounts also on the physical hosts, does not scale for a larger number of users and therefore should be avoided. The fact that the job submission already takes place on Kubernetes that dynamically creates a virtual cluster, i.e., a scalable set of pods, for the execution of the job suggests a different possibility, while the need for access to the physical host is not obvious, and therefore should be replaced by accessing a dedicated pod on the Kubernetes cluster for the submission.

Usage extensions

In the current state SeiSpark offers immediate access to archive seismological data only. This was considered a first logical step to allow for the exploration of very large datasets and limiting the scope of the project was necessary in order to reduce the overall complexity. Therefore, there is currently no direct access to real-time data and the platform cannot be used immediately for operational tasks on real-time data. Still the current SeiSpark platform offers opportunities for tasks like back-testing of candidate models or for massive search and tuning of model parameters and system optimisation based on available archived data. Nevertheless, the Apache ecosystem offers components and frameworks which provide real-time or streaming like functionalities and processing models, like for example Apache Kafka, Apache Spark (Structured) Stream or Apache Flink, and these could provide a path towards operational applications. In fact, there is a generic interest to explore various additional components or completely new frameworks and to assess their potential in the field of seismology. Such activity might become the subject of future work or future projects, and have already been anticipated in the updated design of SeiSpark when we opted for Kubernetes for increased flexibility.

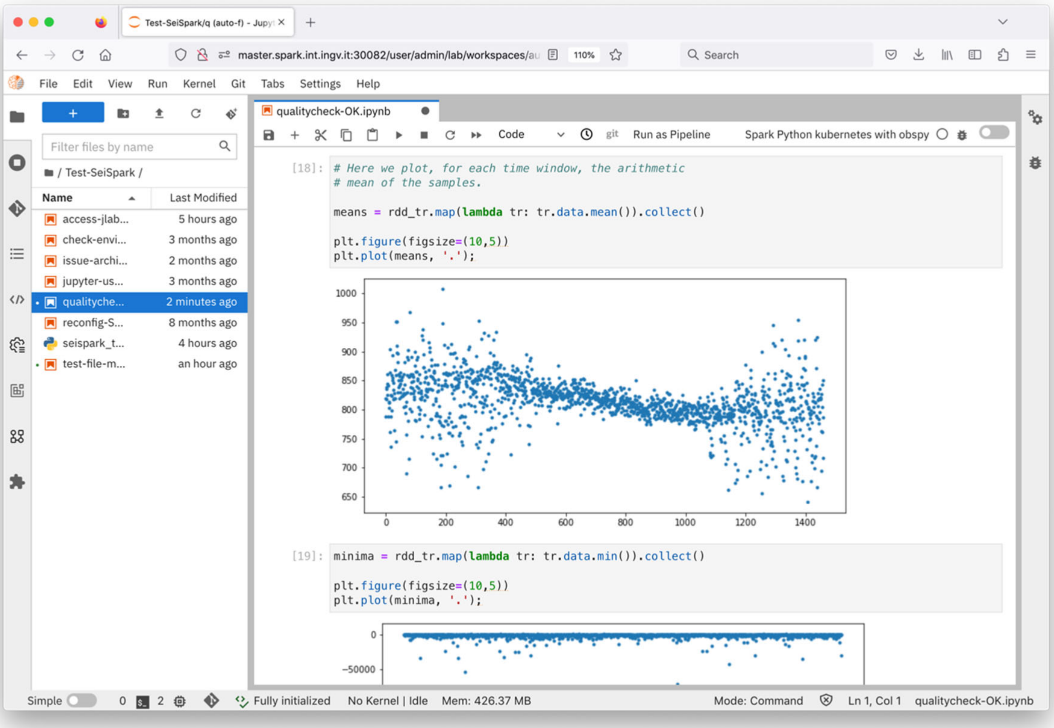


Warning: JupyterHub seems to be served over an unsecured HTTP connection. We strongly recommend enabling HTTPS for JupyterHub.

Username: peter

Password: ••••••••••

Figure 3: User access via JupyterHub login dialog.


Test-SeiSpark/q (auto-f) - Jupy
master.spark.int.ingv.it:30082/user/admin/lab/workspaces/au
110%
Search
File Edit View Run Kernel Git Tabs Settings Help
Filter files by name
/ Test-SeiSpark /
Name
Last Modified
access-jlab... 5 hours ago
check-envi... 3 months ago
issue-archi... 2 months ago
jupyter-us... 3 months ago
qualityche... 2 minutes ago
reconfig-S... 8 months ago
seispark_t... 4 hours ago
test-file-m... an hour ago
qualitycheck-OK.ipynb
Code
git
Run as Pipeline
Spark Python kubernetes with obspy
Simple 0 2
Fully initialized
No Kernel | Idle
Mem: 426.37 MB
Mode: Command
Ln 1, Col 1
qualitycheck-OK.ipynb

```
[18]: # Here we plot, for each time window, the arithmetic
      # mean of the samples.

      means = rdd_tr.map(lambda tr: tr.data.mean()).collect()

      plt.figure(figsize=(10,5))
      plt.plot(means, '.');
```

```
[19]: minima = rdd_tr.map(lambda tr: tr.data.min()).collect()

      plt.figure(figsize=(10,5))
      plt.plot(minima, '.');
```

Figure 4: JupyterLab and SeiSpark customized Jupyter notebook

2. Dastools demonstrator

The software package “dastools” was developed within the context of RISE T2.6 with the aim of being able to standardize datasets generated using the new, disrupting DAS (Distributed Acoustic Seisnsing) technology.

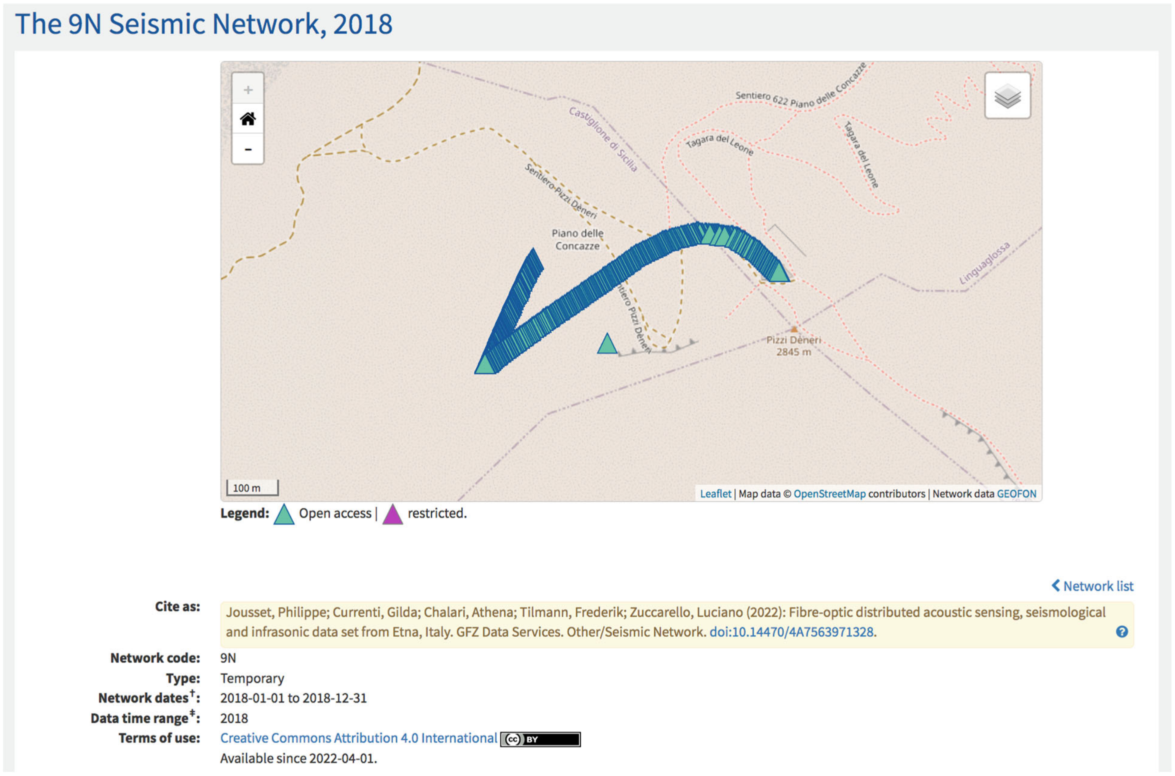
By means of this software, users can convert data in proprietary format (e.g. TDMS, OptoDAS HDF5) to the standard miniSEED widely used within the community. Moreover, a suite of FDSN web services (i.e. dataselect, station) was implemented in order to be able to provide the data without an implicit conversion. Namely, the services can be run on top of the raw data and users will access it in a seamlessly (and standard) way. Notable among the consolidation tasks and improvements carried out since the compilation of D2.11 is the support to the OptoDAS interrogators. The software is available and documented on GitLab at https://git.gfz-potsdam.de/javier/dastools.

This software is already used at GEOFON to include some of these datasets in our seismological archive. We do it by means of the dockerized[1] version of the software package, that is running at our institution and is made available to all users needing it. Once the instruments and the drives with the raw data are attached to our internal server, the PI can make a backup of the data before starting to operate with them. After that, the software can convert the data and they will be archived with a proper FDSN network code, as well as a DOI.

An example is the network 9N(2018) with DOI 10.14470/4A7563971328. Here, the authors use a dedicated fibre-optic cable to obtain strain data and identify volcanic events and image hidden near-surface volcanic structural features at Etna volcano, Italy. This dataset comprises strain-rate

[1] Dockerizing is the process of packing, deploying, and running applications using “Docker”. Docker is a software tool that makes applications available and ready to use as one software package in a container. Dockerizing is a very popular strategy in information technology nowadays.

data from 1 iDAS interrogator (~750 traces), velocity data from 15 geophones and 4 broadband seismometers, and infrasonic pressure data from infrasound sensors. Further explanations of the data and related processing steps can be found in Jousset et al. (2022).



The 9N Seismic Network, 2018

Legend: ▲ Open access | ▲ restricted.

Network list

Cite as:	Jousset, Philippe; Currenti, Gilda; Chalari, Athena; Tilmann, Frederik; Zuccarello, Luciano (2022): Fibre-optic distributed acoustic sensing, seismological and infrasonic data set from Etna, Italy. GFZ Data Services. Other/Seismic Network. doi:10.14470/4A7563971328.
Network code:	9N
Type:	Temporary
Network dates†:	2018-01-01 to 2018-12-31
Data time range‡:	2018
Terms of use:	Creative Commons Attribution 4.0 International (cc) BY
	Available since 2022-04-01.

Figure 5: 9N seismic network, integrated in the GEOFON archive using dastools

3. Cloud strategy demonstrator at ORFEUS Data Centre ODC

The cloud strategy implemented at the ODC c/o KNMI was already described in D2.11. All seismological waveform data at ODC c/o KNMI are stored in the Amazon Simple Storage Service (S3) as objects in so-called buckets. A bucket is a container for objects, while an object is a file and any metadata that describes that file. Amazon's S3 is able to efficiently handle very large volumes of data, and its API is a widely adopted standard for object storage APIs. Applications run by ODC c/o KNMI are progressively being migrated to AWS, while new products are developed primarily within AWS from their start. **Among the new developments occurred since the compilation of D2.11 is the calculation and storage of Power Spectral Densities (http://www.orfeus-eu.org/data/odc/quality/ppsd/; Koymans et al., 2021), for supporting e.g. data quality clients, that runs completely on AWS.**

4. References

- Jousset, P.; Currenti, G.; Schwarz, B.; Chalari, A.; Tilmann, F.; Reinsch, T.; Zuccarello, L.; Privitera, E.; Krawkzyk, C. (2022). Fibre optic distributed acoustic sensing of volcanic events. Nature Communications. doi:10.1038/s41467-022-29184-w
- Koymans, M. R., Domingo Ballesta, J., Ruigrok, E., Sleeman, R., Trani, L., & Evers, L. G. (2021). Performance assessment of geophysical instrumentation through the automated

analysis of power spectral density estimates. Earth and Space Science, 8, e2021EA001675. https://doi. org/10.1029/2021EA001675