ACTINIA: CLOUD BASED GEOPROCESSING

Neteler, M., Gebbert, S., Tawalika, C., Bettge, A., Benelcadi, H., Löw, F., Adams, T., Paulsen, H.

mundialis GmbH & Co. KG
Kölnstraße 99
53111 Bonn, Germany
www.mundialis.de

ABSTRACT

Whether participatory urban planning, digital agriculture or near real-time monitoring of flooded plains – the demand for processing large quantities of Earth Observation (EO) and geodata is constantly increasing. In addition to the amount of data to be processed, the lack of compatibility between different data systems has often been an obstacle.

The cloud based geoprocessing platform *actinia* is able to ingest and analyse large volumes of data already present in the cloud. Through *actinia*'s REST API, following the paradigm of computing next to the data, users can now process and analyse EO- and geodata. Due to the scalability of cloud platforms, insights and tailor made information are delivered in near real-time. Furthermore, methods and algorithms can be easily integrated into own business processes.

Actinia provides an open source REST API for scalable, distributed, and high performance processing of geographical data that mainly uses GRASS GIS for computational tasks.

Index Terms— Earth Observation applications, GIS, cloud based processing, geospatial analysis, open source

1. INTRODUCTION

While open geo- and/or earth observation data is increasingly available, the massive processing of such data still remains a challenge. This causes a mismatch between the enormous information potential of the data on the one side and its actual use on the other side. Organisations, businesses or administrations interested in such information still need special knowledge, appropriate software tools, access to the required data and also adequate processing power.

With *actinia* the company mundialis develops a cloud-based geoprocessing-platform, that aims at enabling users to access and process these kinds of geodata as easily as possible. The access to *actinia* is available through a web-based application programming interface (API, e.g. available at https://actinia.mundialis.de/). In order to use it, the API can be integrated into existing business workflows. For GIS experts and developers, usage of the well documented API and the power of *actinia* is more suitable.

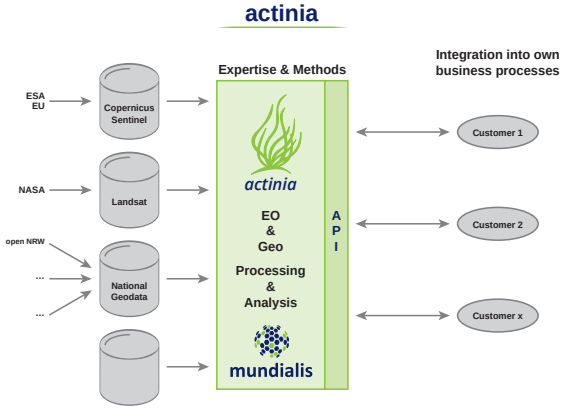

Fig. 1. Workflow of *actinia* geoprocessing engine.

It is also possible to define own processes by chaining the powerful processing-tools provided by *actinia*. Since *actinia* supports space-time cubes through its GRASS GIS backend (https://grass.osgeo.org), data aggregation over time, zonal statistics and more, the users are provided with latest technology being applied to Sentinel time series. Figure 1 shows the general workflow of *actinia* geoprocessing engine. Multiple data sources can be integrated into the different processes for EO and geodata within *actinia* which can be connected to different business processes via API.

The *actinia* engine consists of several components: i) *actinia-core* (available at https://github.com/mundialis/actinia_core), ii) *actinia-gdi* with an interface to Geonetwork Open Source for the access to a metadata catalogue (under development), and iii) *actinia-plugins* for domain-specific applications (under development).

In the remainder of the paper we will illustrate the architecture of *actinia* (user, job and data management, backend), as well as *actinia* process chains and plugins. This is followed by means of deploying *actinia* locally, on the cloud and embedded systems. After the description of integration

with other systems we briefly describe a use case, followed by conclusions.

2. ARCHITECTURE OF ACTINIA

The REST API of *actinia* allows the user to process satellite images, time series of satellite images, arbitrary raster data with geographical relations and vector data. The REST interface accesses, manages and manipulates the GRASS GIS database via HTTP GET, PUT, POST and DELETE requests. Processing of raster, vector and time series is performed on data located in persistent or in ephemeral GRASS GIS databases. *Actinia* supports the processing of cloud based data, for example all available Landsat 4-8 scenes as well as all Sentinel-2 scenes. The API endpoints are realized with the Python framework called Flask-RESTful.

2.1. User management

Actinia provides a sophisticated user, user role and user group management, that allows administrators to specify fine granular access to *actinia* resources as well as time and memory consumption. The following specific concepts are implemented in *actinia*:

- user role hierarchy: superadmin – admin – user – guest, with restricted access to *actinia* endpoints for each user role,
- read-only access to the global *actinia* persistent GRASS GIS spatial database commonly used for base geodata,
- read-write access to the user group specific persistent GRASS GIS spatial database,
- maximum size of the computational region (i.e. amount of allowed pixels), and maximum number of processes for a single process for each user,
- process specific maximum computational and enqueue time for each user.

2.2. Job management

Several *actinia* REST API endpoints are provided to enqueue and manage jobs. The *actinia* REST framework supports asynchronous and synchronous processing and therefore provides a job queue to manage the execution of *actinia* process chains.

Actinia will keep track of the processing time of each computational process as well as the size of the user specific data. Processes can be terminated by their owners or administrator. Each job has a maximum runtime and will be terminated by the queuing system if the runtime was exceeded to avoid broken jobs to block the computational subsystem.

All jobs as well as their logs and resources are written temporarily into a Redis database for fast REST API access and persistently into an elastic search database for analytical and accounting tasks.

2.3. Data management

Actinia uses the GRASS GIS database to store global and user specific geo-data in i) ephemeral or ii) persistent databases.

i) Ephemeral databases exist only for the computational time and will be deleted after processing has finished. However, the computational results of ephemeral processing are available via object storage as GeoTIFF files for raster data or zipped GeoJSON files for vector data.

ii) Two kinds of persistent databases are provided in *actinia*: The read-only global persistent database and the read- and write-able user group specific persistent database. Both databases can be accessed read-only from processes that work on ephemeral databases. The *actinia* REST API allows the import of any online available geo-data source into ephemeral and user group specific persistent databases. Geo-data can be raster images, vector feature collections or raster collections. Import of online resources can be directly defined in an *actinia* process chain.

Actinia provides several endpoints to list all data of user specific accessible persistent databases. These endpoints provide functionality to query raster, vector and raster time series data and to analyze their metadata like data descriptions, valid time and transaction time stamps as well as statistics generated at runtime.

2.4. Backend

The core of *actinia* is GRASS GIS [3], an open source geoinformation system (https://grass.osgeo.org). It comes with an integrated temporal framework that provides spatio-temporal capabilities, delivering comprehensive functionality to implement a fully featured temporal geographic information system (GIS) based on a combined field and object-based approach [2]. Through that it manages spatial fields of raster, three-dimensional raster and vector type in time series referred to as space-time data sets (STDS).

2.5. Actinia process chains

GRASS GIS provides over 500 processing modules. The *actinia* REST API that is used to access the GRASS GIS backend supports the definition of process chains, in which any number of GRASS GIS modules can be defined and their data exchanged. The *actinia* process chain approach provides a great flexibility to define very complex analytic processes that involve many GRASS GIS modules as well as external tools like GDAL (Geospatial Data Abstraction Library). However, complex analytic tasks are already available as simple REST

endpoints that require no knowledge about process chain creation. Some of these endpoints require the installation of the *actinia-plugins* (see Sec. 2.6). They include following functionality:

- creation of Landsat 4-8 and Sentinel-2 raster time series,
- NDVI computation for arbitrary Landsat 4-8 and Sentinel-2 scenes,
- zonal statistics on raster time series based on vector polygons,
- spatio-temporal sampling of and algebraic operation on raster time series,
- raster and vector data export.

2.6. Actinia plugins

Actinia can be extended using plugins. Plugins provide endpoints for very specific computational tasks, that can be exposed as single REST endpoints. At the moment there are two plugins available: i) processing plugin for satellite data from Landsat and Sentinel-2 satellites; ii) statistical analysis plugin for raster and raster time series data as well as spatio-temporal sampling.

2.7. Support for GRASS GIS addons

The *actinia* framework supports any addon that is available for GRASS GIS. Moreover, developers and power-users can use the PyGRASS [5] and the GRASS GIS Temporal framework [2, 1] in GRASS GIS to implement high performance spatio-temporal applications that can directly be used as processes in the *actinia* process chain.

2.8. Authentication

The *actinia* REST framework uses basic HTTP authentication using user/password or authentication tokens. REST API endpoints are available to create authentication tokens for specific users.

3. DEPLOYING ACTINIA

Actinia can be deployed on a wide range of hardware and cloud environments.

3.1. Personal computer and local data center

Actinia can be deployed on a workstation computer or local data center as being a Python package together with GRASS GIS and a Redis database server. Moreover, *actinia* and all of its sub-components can also be deployed as docker image at these systems as well.

3.2. Public and private Cloud

Actinia and its sub-components: geospatial backend (GRASS GIS), user and job database (Redis), and logging (fluentd, elasticsearch, kibana) can be deployed either as docker swarm or Kubernetes cluster in scalable cloud environments. It is designed to run in thousands of containers in parallel. *Actinia* provides a granular locking mechanism, so that users that work in parallel on the same persistent database, can not interfere or corrupt each others data. *Actinia* has been successfully deployed in the Google Cloud, Amazon AWS, CloudFerro, the Open Telecom Cloud and many more.

3.3. Embedded systems

Actinia requires a Linux environment, Python3, GRASS GIS and Redis to work. These components are available on many embedded systems, which makes *actinia* fit for running in rough environments that are populated by IoT devices, remote controlled robots and drones as well as satellites to provide real-time and batch processed sensor and image analyses.

4. INTEGRATION WITH OTHER SYSTEMS

4.1. Business logic

The REST API based approach of *actinia* in combination with *ad hoc* processing of literally any online available geo-data using ephemeral databases makes *actinia* suitable to be used in business logic systems which require high performance spatial and spatio-temporal analysis and processing.

4.2. openEO wrapper and User Defined Functions

The OpenEO H2020 project (http://openeo.org/) aims at developing an open API to connect R, Python, Javascript and other clients to big Earth Observation cloud back-ends in a simple and unified way [4]. GRASS GIS is one of the back-ends; the GRASS GIS driver uses the *actinia* REST API to send processing requests based on the *actinia* process chain approach to the GRASS GIS backend (https://github.com/Open-EO/openeo-grassgis-driver). The current public endpoint is available at http://openeo.mundialis.de:5000/capabilities. The full archives of Landsat 4 - 8 and Sentinel-2 are accessible and in near future also Sentinel-1.

The support for openEO approach of User Defined Function (UDF) is currently under development and will run as part of an *actinia* process chain (https://open-eo.github.io/openeo-udf/api_docs/).

4.3. Copernicus DIAS

As part of the European Union's Copernicus program several Data and Information Access Services (DIAS) platforms are currently being implemented. They consist of cloud-based

infrastructures that will make Copernicus data available to its users along with tools and marketplaces to offer third party applications. While *actinia* is cloud vendor agnostic and compatible to Google Cloud or Amazon AWS; it would be possible to deploy *actinia* on the new European DIAS platforms.

4.4. *Actinia* shell in GRASS GIS

GRASS GIS was recently extended to support the creation of valid *actinia* process chain JSON building blocks by simply calling a GRASS GIS module with all of the required parameters. At time the development of a GRASS GIS shell extension is under way, so that module calls that operate on the local GRASS GIS database can be directly sent to an *actinia* server for remote execution. Hence, complex process chains can be tested on a local GRASS GIS installation and then send directly via HTTP request to a remote *actinia* server that resides near the cloud based geo-data (Sentinel-2, Landsat, MODIS, ...) without the requirement of deploying *actinia* locally.

5. POTENTIAL OF ACTINIA IN THE FIELDS: AGRICULTURE, LULC AND FORESTRY

Automated Sentinel-1 and Sentinel-2 time series preprocessing using *actinia* is a prerequisite for numerous Earth Observation applications. For agricultural monitoring, we developed a cutting frequency index besides crop type classification. Furthermore, Land Use/Land Cover (LULC) time series status and change maps are derived in *actinia* using machine learning algorithms. For example, LULC maps are the basis for modeling and decision making in the field of water management and projecting the water consumption in the future in the region of Drâa in Morocco.

Being weather independent and generated by active sensors, SAR images are most adequate to monitoring tropical rain forest disturbances. Sentinel-1 image processing requires large RAM which becomes an issue when working with desktop computers, e.g. using SNAP (Sentinel Tool Box). For massive SAR data processing, SNAPPY (SNAP Python API) as the Python interface to the SNAP processing technology is most promising and allows the integration of SNAPPY based processing chains into the *actinia* cloud environment. As *actinia* itself is mainly based on GRASS GIS, the SNAPPY processing chain can be wrapped as a standard GRASS GIS Python addon. A study area in Guyana of a managed forest concession is selected to evaluate the potential of *actinia* to monitoring the forest logging damages between the years 2015 and 2017 using Sentinel-1 images. The *actinia* based processing shows a considerable processing time reduction due to the automation of the processing steps (example: i) EO analyst – 50 min for data selection, download, manual preprocessing, index creation; *actinia* – ¡ 5 min for data selection and automated processing), allowing for fast decision making and support through improved reporting in the context of sustainable development and climate change commitments.

6. CONCLUSIONS

The cloud based geoprocessing engine *actinia* follows the central paradigm of Big Data to "bring the software to the data". It is cloud vendor agnostic open source software. Its functionality be extended through user plugins written in Python, C, or originating from other software packages. The user management supports group, roles and access rights. Process chains can be written in JSON (developer friendly), an *actinia* shell is available (power user friendly); it supports integration into own business processes through API (business friendly). To comply with the general data protection regulations it can be deployed in European cloud solutions.

REFERENCES

[1] S. Gebbert and E. Pebesma. The GRASS GIS Temporal Framework: Object oriented code design with examples, January 2017. URL https://trac.osgeo.org/grass/browser/grass/trunk/lib/python/docs/src/Temporal-Framework-API-Description.pdf.

[2] S. Gebbert and E. Pebesma. The GRASS GIS temporal framework. *International Journal of Geographical Information Science*, 31(7):1273–1292, 2017. doi: 10.1080/13658816.2017.1306862.

[3] M. Neteler, M.H. Bowman, M. Landa, and M. Metz. GRASS GIS: A multi-purpose open source GIS. *Environmental Modelling & Software*, 31:124–130, 2012. doi: 10.1016/j.envsoft.2011.11.014.

[4] E. Pebesma, W. Wagner, P. Soille, M. Kadunc, N. Gorelick, M. Schramm, J. Verbesselt, J. Reiche, M. Appel, J. Dries, A. Jacob, M. Neteler, S. Gebbert, C. Briese, and P. Kempeneers. openEO: an open API for cloud-based big Earth Observation processing platforms. In *EGU General Assembly Conference Abstracts*, volume 20, page 4957, April 2018. URL https://meetingorganizer.copernicus.org/EGU2018/EGU2018-4957-1.pdf.

[5] P. Zambelli, S. Gebbert, and M. Ciolli. Pygrass: An Object Oriented Python Application Programming Interface (API) for Geographic Resources Analysis Support System (GRASS) Geographic Information System (GIS). *ISPRS Int. J. Geo-Information*, 2(1):201–219, 2013. doi: 10.3390/ijgi2010201.