frontiers | Frontiers in High Performance Computing

TYPE Original Research
PUBLISHED 25 August 2023
DOI 10.3389/fhpcp.2023.1151530

OPEN ACCESS

EDITED BY
Phillip G. Bradford,
University of Connecticut, Stamford,
United States

REVIEWED BY
Arnis Lektauers,
Riga Technical University, Latvia
Muhammad Zakarya,
Abdul Wali Khan University Mardan, Pakistan

*CORRESPONDENCE
Lucas Iacono
✉ liacono@know-center.at

RECEIVED 26 January 2023
ACCEPTED 04 August 2023
PUBLISHED 25 August 2023

CITATION
Iacono L, Pacios D and Vázquez-Poletti JL (2023) SNDVI: a new scalable serverless framework to compute NDVI.
Front. High Perform. Comput. 1:1151530.
doi: 10.3389/fhpcp.2023.1151530

SNDVI: a new scalable serverless framework to compute NDVI

Lucas Iacono[1]*, David Pacios[2] and Jose Luis Vázquez-Poletti[2]

[1]Know-Center, Human AI Interaction Area, Graz, Austria, [2]Departamento de Arquitectura de Computadores y Automática, Facultad de Informática, Universidad Complutense de Madrid, Madrid, Spain

Farmers and agronomists require crop health metrics to monitor plantations and detect problems like diseases or droughts at an early stage. This enables them to implement measures to address crop problems. The use of multispectral images and cloud computing is conducive to obtaining such metrics. Drones and satellites capture extensive multispectral image datasets, while the cloud facilitates the storage of these images and provides execution services for extracting crop health metrics, such as the Normalized Difference Vegetation Index (NDVI). The use of the Cloud to compute NDVI poses new research challenges, such as determining which cloud technology offers the optimal balance of execution time and monetary cost. In this article, we present Serverless NDVI (SNDVI), a new framework based on serverless computing for NDVI computation. The objective of SNDVI is to minimize the monetary costs and computing times associated with using a Public Cloud while processing NDVI from large datasets. One of SNDVI's key contributions is to crop the dataset into subsegments to leverage Lambda's ability to run up to 1,000 NDVI computing functions in parallel on each subsegment. We deployed SNDVI using Amazon Lambda and conducted two experiments to analyze and validate its performance. Both experiments focused on two key metrics: (i) execution time and (ii) monetary costs. The first experiment involved executing SNDVI to extract NDVI from a multispectral dataset. The objective was to evaluate the overall SNDVI functionality, assess its performance, and verify the quality of SNDVI output. In the second experiment, we conducted a benchmarking analysis comparing SNDVI with an EC2-based NDVI computing architecture. Results from the first experiment demonstrated that the processing times for the entire SNDVI execution ranged from 9 to 15 seconds, with a total cost (including storage) of 4.19 USD. Results from the second experiment revealed that the monetary costs of EC2 and Lambda were similar, but the computing time for SNDVI was 411 times faster than the EC2 architecture. In conclusion, the investigation reported in this paper demonstrates that SNDVI successfully achieves its goals and that Serverless Computing presents a promising native serverless alternative to traditional cloud services for NDVI computation.

KEYWORDS

serverless computing, NDVI, cloud computing, remote sensing, precision agriculture, crops monitoring, Amazon Lambda

1. Introduction

Farmers have adopted technology as part of their daily farming activities. Thanks to technology, farmers can improve the quality of their crops, increase their profits and provide the necessary food for people. The use of remote sensing technologies to analyse and monitor crop quality at each phenological stage is increasingly widespread (Zhang et al., 2019).

Remote sensors such as multispectral cameras on drones or satellites obtain images composed of different bands of the electromagnetic spectrum that crops emit or reflect.

These electromagnetic bands permit the computation of a wide range of crop health metrics like the Normalized Difference Vegetation Index (NDVI). The NDVI is used to obtain information about the quantity, quality and development of vegetation. This metric is based on the measurement of the intensity of radiation from certain bands of the electromagnetic spectrum that crops emit or reflect (Carlson and Ripley, 1997).

Although there are different metrics that could be used to determine crop health (SAVI, EVI, GLI, among others), in this paper we focus on NDVI computation because it is the most widely used vegetation index for retrieval of vegetation canopy biophysical properties (Jiang et al., 2006). However, in future research, we will apply the current methodology to compute other metrics.

The NDVI is calculated as follows:

$$NDVI = \frac{(NIR - Red)}{(NIR + Red)} \quad (1)$$

Where the variables Red and NIR represent the spectral reflection measurements acquired in the red and near-infrared regions, respectively. The NDVI is calculated as the ratio of the difference between near-infrared light (NIR) and red light to the sum of these two measures. The NDVI values range between -1 and +1, representing the proportion of green leaf vegetation. Negative NDVI values, approaching –1, correspond to water bodies, as water reflects more in the visible and less in the NIR. On the other hand, values close to +1 (typically around 0.8–0.9) correspond to dense green leaves, as healthy vegetation absorbs more visible light (red) and reflects more near-infrared light. Therefore, the NDVI provides a measure of healthy green vegetation and its distribution.

In order to obtain the NDVI, there are algorithms that process large image datasets (from Gigabytes up to Terabytes). This is where Cloud and Serverless Computing play a fundamental role. Cloud Computing provides solutions to the problem of large image dataset processing due to its reliability, availability and scalability. The Cloud offers access to computing infrastructure composed of virtual machines under a pay-per-use price model. Such infrastructure can be tuned according to the user's needs because the Cloud offers a wide range of hardware and software configurations. Furthermore, the monetary costs of Cloud resources may be lower compared to the traditional in-house clusters (Zhai et al., 2011).

Moreover, it is worth mentioning that the cloud model's flexibility is a key advantage for scenarios with seasonal variations like NDVI computation. Users can easily scale resources based on demand, optimizing spending and making cloud computing a cost-effective option for handling fluctuating workloads.

Originally, the Cloud Computing model provided three types of services: Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). Given Cloud is a constantly evolving technology, nowadays there are new services that increase the original Cloud capabilities. One of these services is Function as a Service (FaaS), also called Serverless Computing.

Serverless computing is a platform that hides server usage from developers and runs on-demand codes automatically scaled and billed only for the time the code is running (Castro et al., 2019). Serverless applications are delivered to users through a set of functions which are triggered by system-generated events and user-generated events (Vazquez-Poletti and Llorente, 2018). The functions have a short lifetime and are executed through containers or virtual machines (Shahrad et al., 2019).

In this paper, we present Serverless NDVI (SNDVI), a serverless application specifically designed for NDVI calculation. SNDVI's main goal is to reduce the monetary costs and execution times of NDVI calculation. It also lay the foundations for the development of new serverless applications that will enable obtaining other crop metrics required by the agricultural ecosystem.

SNDVI was implemented by using Amazon Web Services (AWS) (Amazon, 2023c) and validated through two experiments. The first experiment consisted of the execution of SNDVI to calculate NDVI from a dataset of multispectral images collected with the Landsat 8 satellite (USGS, 2023). The results of experiment 1 showed that the entire architecture required between 9 and 15 seconds to extract the NDVI from the dataset when it runs on Amazon Lambda. The total monetary cost of the execution was about 4.19 (US Dollars), including S3 storage and Lambda functions.

In the second experiment, we conducted a benchmarking analysis comparing SNDVI with an EC2-based NDVI computing architecture in order to compare the performance of SNDVI with other architectures based on traditional cloud computing services. The results show that SNDVI fulfills its original purpose and has a good performance in terms of execution time and monetary costs. Specifically, the processing costs (the ones that don't include S3 storage because S3 monetary costs are the same in Lambda and EC2) were approximately the same in both cases: 0.06 US Dollars in Lambda and 0.061 US Dollars in Amazon EC2. In the comparison done in Experiment 2, the main difference between EC2 and Lambda was the execution time. The time required on EC2 was approximately 411 times greater than in Lambda.

Our native serverless solution has different contributions to data processing:

- The ability to minimize computing times and processing costs. In this article, we have demonstrated that serverless computing lowers the cost and execution time of NDVI calculation when compared to traditional cloud computing services such as EC2.
- Workload distribution. SNDVI can distribute the workload across a network of cloud-based servers, allowing for more efficient use of computing resources and reducing the need for costly infrastructure. This makes it an ideal solution for data processing in industries such as agriculture, where there is a vast amount of data generated from various sources, including sensors, drones, and satellites.
- On-demand big data processing. With the ability to scale up or down based on demand, SNDVI can help agriculture companies process and analyse large amounts of data quickly and cost-effectively. This can help to improve decision-making and operational efficiency, leading to better crop yields and reduced costs.

In brief, the development of new data processing infrastructure in agriculture using the methodology presented in this paper has a high potential for contributing to the agricultural industry. This contribution is based on providing farmers with the tools they need

to make more informed decisions and maximize their yield while minimizing costs.

This paper is organized as follows. Section 2 summarizes some research works developed by the scientific community related to the use of serverless computing for satellite and drone image processing. In Section 3, we detail SNDVI, including its architecture and main components. Then, Section 4 presents the experiments performed to validate SNDVI and their results. Finally, Section 5 discusses the conclusions of this paper and our plan for future research.

2. Related works

In recent years satellite constellations such as Landsat, Modis (NASA, 2023) and Sentinel-2 (ESA, 2023) have generated petabytes of data. As an example, we can mention that The Sentinel satellites acquired about 24.87 petabytes of remote sensing data by the end of 2020 (Drusch et al., 2012). If we also consider drones that generate higher-resolution images, we can state without hesitation that in the upcoming years, the volume of information will be a challenge for the development of applications capable of computing such volumes of data.

Several authors have faced this challenge and developed different solutions to provide answers to the demand for computing large volumes of data in agriculture. While the use of Cloud Computing is not a novelty in remote sensing data processing, the use of Serverless Computing beyond querying sensor data is. This is why, our native serverless approach involves the application of serverless computing in the main phases of the remote sensing data process (data query, metrics computation, and results presentation). In this chapter, we analyse related works about the use of serverless computing for satellite and drone image processing.

In Wu et al. (2022), the authors validate the feasibility of a framework based on serverless computing for remote data processing. They deploy their architecture on the Alibaba Cloud and test it by performing NDVI with Landsat 8 images. In the experiment, each image is split into four parts of 256 * 256 pixels and then the NDVI is performed with each tile. After the experiments, the authors state that the NDVI computation using Serverless Computing has clear benefits depending on the hardware used on the Instance hosting the functions, but if the memory consumption is more than 4 GB, the computing time could not be reduced. Also, they found that computing times vary from 700 to 400 ms depending on the current state of the infrastructure (cold starting, already started or performance mode).

Yang et al. (2021) developed FAASRS, another framework based on Serverless Computing for remote sensor data computation. FAASRS is deployed using Amazon Lambda (Amazon, 2023a) and Python. The framework developed by the authors breaks the job by partitioning the image into small mosaics based on the user's algorithm. Then divides the task by splitting the image into mosaics according to the geospatial region. Finally, FAASRS uses each Lambda worker to perform the computation of each mosaic. While the authors use serverless computing as a fundamental component of their framework, the platform uses Amazon EC2 Virtual Machines to coordinate the execution of the Lambda workers. Therefore, this platform is not completely serverless.

In Chu et al. (2020) the IBM research team presents a serverless pipeline for ingesting, storing, and querying remote sensed satellite data using IBM Cloud. The raw data is collected periodically through ground stations and stored in object storage, which is not suitable for efficient querying. Therefore, the data is transformed into a queryable format, such as Parquet or cloud-optimized GeoTIFF, using IBM Cloud Code Engine, which accommodates elastic scaling and "spiky" activity. The transformed data is stored in IBM Cloud Object Storage, and cloud functions can be invoked to query raster imagery, given a region and time range. In general, this approach provides a simple developer experience to manage terabytes to petabytes of remote sensing data using an affordable form of cloud storage and computing.

The use of Serverless Computing together with other Cloud Computing services for remote data processing has also been addressed by scientists working in land use and cover monitoring (Ferreira et al., 2020). In their work, Ferreira et. al. evaluate the use of AWS services in these applications. The authors found that this platform contains several image datasets such as Landsat-8 and Sentinel 2, which is a great advantage compared to other options in the market. Among the different use cases discussed in the paper, there is a web platform that supports deforestation detection by allowing the visualization of remote sensing images stored in AWS S3 (Amazon, 2023b) buckets through maps services and APIs for Spatio Temporal Asset Catalogs deployed on Lambda functions.

The Terra-Byte platform developed by the Leibnitz Super Computing Center and the DLR (German Aerospace Center) is an HPC data analytic infrastructure designed to fill the requirements of earth observation application processing (Eismann et al., 2020). The core of the platform is an HPC storage space (about 30 PBytes) which stores earth observation data and serves different applications. In this infrastructure, data query is one of the main tasks to focus on. Also, Terra-Byte provides its computing infrastructure through PaaS and FaaS capabilities developed on top of Amazon Services.

Table 1 provides an overall view of the papers reviewed in our paper and SNDVI. The Table details the used approach (serverless computing, cloud computing or a combination of both), the services provider and the application domain of each previous work.

After reviewing the state of the art, we can observe that the use of serverless computing has gained significant popularity in recent years due to its ability to offer a more efficient and cost-effective way of deploying and running cloud applications. However, the development of data processing architectures fully based on Serverless Computing for Image Processing is still relatively new, and there is limited research on how to design, implement, and validate such architectures.

The above-mentioned gap in knowledge raises a significant problem for organizations (such as precision agriculture software providers) looking to improve their data processing systems through serverless computing. Therefore, our main research problem consists in defining a data processing architecture fully based on serverless computing and its performance validation.

TABLE 1 Previous works surveyed in this section.

Reference	Technologies	Provider	Application domain
Wu et al. (2022)	Serverless	Alibaba cloud	NDVI computation
Yang et al. (2021)	Serverless and cloud	Amazon	NDVI computation
Chu et al. (2020)	Serverless and cloud	IBM cloud	Earth observation data query
Ferreira et al. (2020)	Serverless and cloud	Amazon	Deforestation analysis
Eismann et al. (2020)	Serverless and cloud	Amazon and others	Earth observation applications
SNDVI	Serverless	Amazon	NDVI computation

3. SNDVI

In this Section, we discuss the architecture and the execution path of SNDVI.

3.1. Serverless NVDI model

This paper presents a serverless distributed architecture that utilizes two S3 buckets and two Lambda functions to perform calculations of the NVDI. Figure 1, illustrates the overall architecture of SNDVI. The architecture is designed to be cost-effective and scalable, allowing for easy deployment and maintenance. The two S3 buckets are used for data storage and retrieval, while the two Lambda functions are responsible for processing the NVDI calculations.

The design of this architecture allows for efficient and accurate calculation of NVDI, making it a valuable tool for various industries such as agriculture and remote sensing.

Our Serverless solution is a cost-effective alternative to running NDVI computations compared with Amazon EC2 instances-based platforms and hybrid platforms which combine EC2 and Lambda. The cost difference between serverless (AWS Lambda) and EC2 instances is a crucial aspect to consider in the choice of architecture. In our experiments, we found that the monetary cost of using Lambda and EC2 was similar, while S3 costs are the same in both cases (4.13 US Dollars). However, the significant difference lies in the execution time. For instance, the time required to compute 1000 images on AWS Lambda was approximately 411 times faster than on the EC2 architecture. This is due to Lambda's ability to execute up to 1000 functions in parallel, a capability not easily available on EC2 without the use of more instances and specialized Python parallelism techniques.

In contrast with other papers surveyed in the state of the art (Yang et al., 2021; Wu et al., 2022) SNDVI eliminates the need to manage the server infrastructure. With our serverless solution, the cloud provider takes care of all the underlying infrastructure, including scaling, security, and availability, which can significantly reduce costs. While serverless computing may not be suitable for all use cases, it can be the right option for applications that have variable and unpredictable workloads and can benefit from the flexibility and scalability of serverless architectures.

First of all, we are going to discuss the internal materials created for the optimization of Lambda functions.

3.2. Custom layers for Lambda functions

In the context of AWS Lambda, a layer is a way to package external dependencies, libraries or custom runtimes that the function code requires to execute properly. However, there are specific cases where it is not feasible to use Lambda layers due to the large size of the libraries. One of these cases is SNDVI, where a specific library that is required for the calculation of the NVDI is the Python package called "GDAL" (Geospatial Data Abstraction Library) (Warmerdam, 2008). This package provides a set of tools and libraries that allow for the efficient handling of large image datasets. GDAL can manipulate (read, write, crop, tile, re-project, etc.) raster and vector geospatial data formats such as satellite imagery.

In SNDVI, we did not use the standard Lambda layer to execute GDAL due to its size. Instead, we followed a "black box" approach by compiling GDAL together with the NDVI codes into a binary file using PyInstaller. This binary is the SNDVI component that opens the satellite multispectral image and calculates NDVI.

3.3. Custom binary compiled code

The "black box" approach used in SNDVI involves compiling the code into a standalone binary and calling it from a Lambda function using a script. This approach presents the following advantages.

- Greater control over the execution environment. The binary enables the creation of a self-contained package that is not dependent on the underlying infrastructure or runtime. This is useful for situations where the specific version of a library or dependency is important.
- Improvement of the Lambda function performance. By following the "black box" approach it is possible to reduce the overhead associated with interpreting the code at runtime, leading to faster execution times. This can be particularly beneficial for computationally-intensive tasks such as image processing or machine learning.

3.3.1. SNDVI binary functioning

First, our binary opens two input images from the multispectral dataset (NIR and Red) using the "GDALOpen()" function. Second, the binary retrieves the band of each image using the GetRasterBand method. The code execution continues accessing the number of rows, columns and the geo-transform information from the image using the RasterYSize, RasterXSize, and GetGeoTransform properties. Third, it sets the output as a

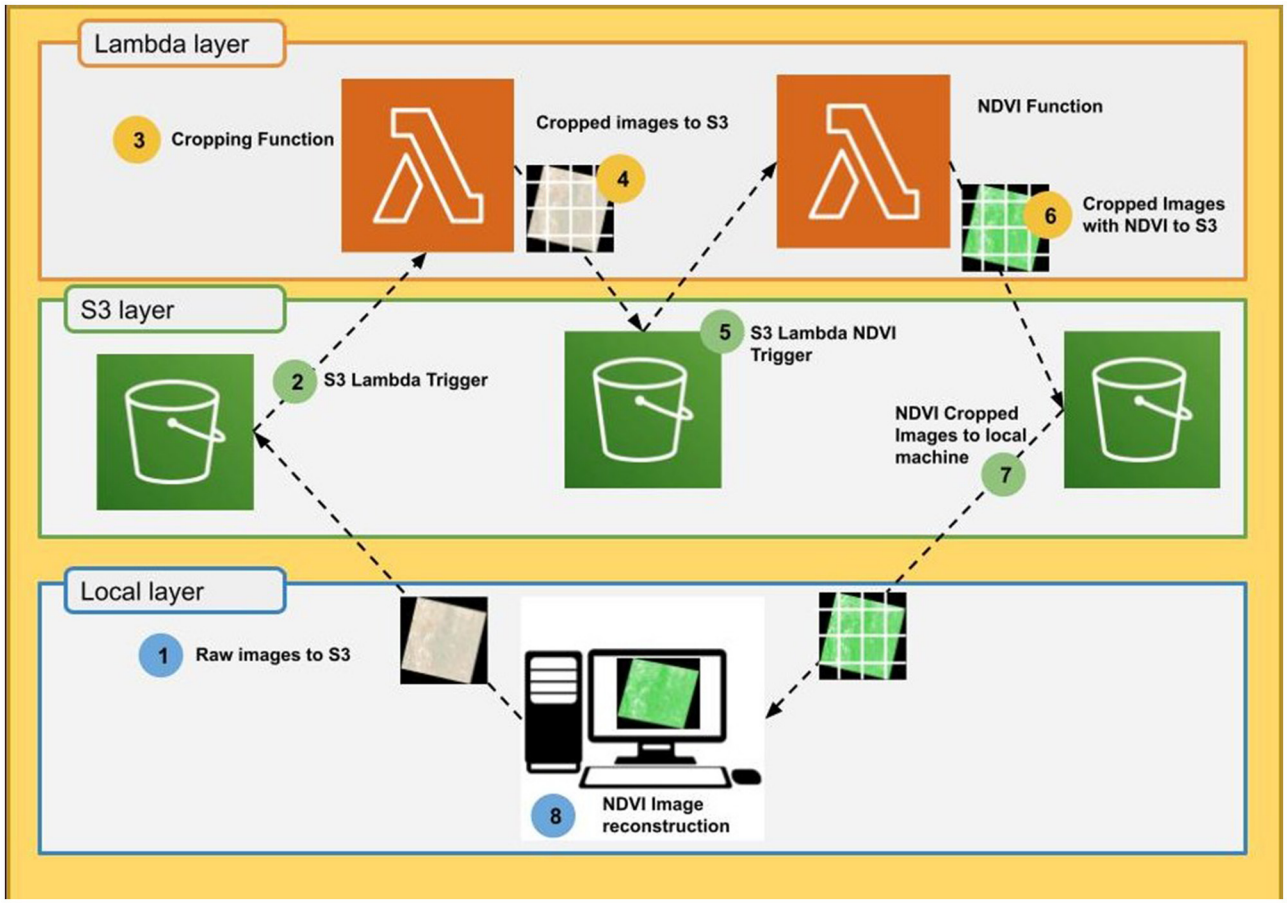

FIGURE 1
Architecture of SNDVI.

32-bit floating point (−1 to 1), and then it calculates the NDVI. In SNDVI the GDAL default format (32-bit floating point) is used to do the NDVI computation. Finally, the binary writes the output image using the GDT_Float32 data type. This image is the NDVI result.

3.3.2. SNDVI binary generation

The NDVI binary was generated using PyInstaller (Cortesi, 2022). This tool packs a Python application and all its dependencies (e.g., GDAL and Numpy) into a single package. The user can execute the packaged app without installing a Python interpreter or any packages. Since the Lambda functions run on the Amazon Linux operating system, the binary file must be compatible with this operating system. Table 2, details the main steps of the procedure implemented to compile the NDVI binary file.

TABLE 2 Binary NDVI file generation procedure.

Step	Details
1	Deploy of Amazon EC2 instance running Amazon Linux OS.
2	PyInstaller installation.
3	Installation of dependencies required by NDVI code (GDAL, Numpy, etc.).
4	Creation of .py file with source code for NDVI calculation software.
5	Test and debugging of NDVI python code.
6	Compilation of binary file using PyInstaller.
7	Downloading of binary file from EC2 instance to local storage system.
8	Uploading of generated binary file to S3 bucket to be called by the Lambda function.

3.4. Execution path

The first step in the SNDVI execution starts at the local layer and is the streaming of the satellite raw images to Amazon S3 using the Amazon Command Line (see Process 1 in Figure 1). When a new image arrives in the S3 bucket, there is a trigger (3) that activates a Lambda function (3) called Cropping Function. This function is responsible for cropping the image into smaller fragments due to limitations in storage space. There are at least two ways to accomplish this: (i) by using a version of OpenCV, a

popular computer vision library or (ii) through Pillow layers (Clark et al., 2019), which is a powerful image manipulation library in Python.

Once the image has been cropped, the fragments are then deposited in the second S3 bucket (4), triggering (5) the next Lambda function. This function called NDVI can then be used to perform additional processing on the fragments, such as the NVDI calculations or other image analysis tasks.

This function loads the compiled binary from S3 in Lambda and also provides the binary with the image to be processed from the S3 input bucket. The binary is able to perform the NVDI calculations thanks to the use of the previously mentioned layer of GDAL. Once the calculations have been completed, the resulting image is deposited (6) in a final S3 bucket ready for download (7) via Amazon CLI.

Finally, when the NVDI calculations have been completed, edge computing is used (8) to combine all the images together in a single output file, which can be easily accessible and analyzed.

4. Methods and materials

4.1. Experiments

In this section, we detail the experiments performed to evaluate SNDVI. In the first experiment, we analyzed the SNDVI performance and validated the quality of the NDVI obtained after SNDVI execution. In the second experiment, we conducted a benchmark of SNDVI with another NDVI architecture based on Amazon EC2. In both experiments, we used two performance metrics: execution time and monetary cost. Such metrics have been validated in previous works (Iacono et al., 2018).

4.1.1. Experiment one. Serverless performance

The first experiment was conducted within the AWS environment, using Lambda functions running Amazon Linux 2018.03 and S3 for data storage and retrieval.

The objective of this experiment was to evaluate the performance and cost of the architecture, as well as to identify any limitations or bottlenecks in the system. To accomplish this, we stored numerous images in the system and monitored the processing times and costs.

The dataset used for the experiment corresponds to the Operational Land Imager sensor (OLI) of the Landsat 8 satellite. This satellite (USGS, 2023) is part of the Landsat Data Continuity Mission (LDCM) which is a collaboration between NASA and the United States Geological Survey (USGS). Landsat 8 images have 15-meter panchromatic and 30-meter multi-spectral spatial resolutions along a 185 km swath.

The dataset contains all the bands from the OLI sensor and the corresponding metadata. The dataset size is about 1 Gbyte. In this paper, we used the images from Bands 4 (Red) and 5 (Near-infrared) from the OLI sensor. The data size of such bands is in total 190.1 MBytes.The images correspond to the Province of Mendoza, Argentina. Mendoza is the main wine producer in Argentina and one of the biggest players in the global market. Figure 2 shows a polygon indicating the area of study. The area is determined with the coordinates shown in Table 3.

The dataset [Earth Resources Observation and Science (EROS) Center, 2013] was collected by the Satellite on the 10th of March 2021 at 14:27:23 Argentina local time (GMT-3).

The initial function in our architecture was responsible for receiving and cropping the images. This function can crop each image into 1000 segments, so it can be executed in parallel 1,000 times. Each fragment followed its own parallel execution path.

In order to evaluate performance, we measured the processing times for the entire architecture, including the cropping and NVDI calculation steps. Additionally, we monitored the monetary cost associated with running the architecture in AWS to determine the total monetary cost.

In this paper, we didn't perform extra experiments regarding SNDVI scalability. We consider that this is not necessary because the underlying serverless technology used to develop SNDVI is scalable by design. Instead of relying on dedicated servers, SNDVI code runs in small units called "functions", which are automatically deployed as needed in response to demand. This means that the ability of the application to scale horizontally is directly related to the number of function instances that are activated to handle the load.

The alignment of the SNDVI functions with AWS Lambda operations is a key aspect of our architecture's efficiency and effectiveness. AWS Lambda is designed to run code in response to events, making it an ideal platform for executing SNDVI functions, which are event-driven and stateless by nature. Each SNDVI function is triggered by an event, such as the arrival of a new image, and runs independently of others, allowing for high levels of parallelism and scalability. This alignment with Lambda's operational model not only simplifies the architecture but also optimizes resource usage, as Lambda automatically scales to match the rate of incoming events. Furthermore, the pay-per-use pricing model of Lambda aligns with the sporadic and unpredictable nature of SNDVI function invocations, leading to cost-effectiveness.

4.1.2. Experiment two. Lambda and EC2 comparison

To validate the performance of Amazon Lambda over other AWS services for NDVI computation in terms of execution time and costs, we conducted a second experiment where we ran SNDVI on an Amazon EC2 t2.medium instance running the Amazon Linux OS version 18.3 (AMI 2018.03.0.20230404.0). In this case, we executed SNDVI with the same dataset used in the experiment carried out with pure Lambda services. In the EC2 version of SNDVI, we added the cropping function of Lambda to the Local layer. Also, we did some minor adaptations on the S3 layer, to work with EC2 instead of Lambda. In other words, we set up a file synchronization script on the EC2 instance that checks if a new image arrives in S3. This script replaces the Lambda trigger used on the SNDVI pure Lambda version. Also, we deployed the NDVI computing function of the SNDVI Lambda version on the EC2 layer. For statistical reasons, the SNDVI EC2 version was executed ten times.

10.3389/fhpcp.2023.1151530

FIGURE 2
Study area.

TABLE 3 Study area coordinates.

Corner	Latitude	Longitude
Upper Left	−32.11601	−70.35679
Upper Right	−32.11846	−67.89544
Lower Left	−34.22611	−70.38971
Lower Right	−34.22876	−67.86864

In this investigation, we have conducted tests using sequential execution on Amazon EC2 instances, specifically focusing on evaluating the capabilities of Amazon Lambda and EC2 at equivalent price points. It is important to note that in order to fully exploit the parallelization potential of EC2, larger and more expensive instances would be required, which would ultimately result in a higher overall cost for the system architecture. As Lambda allows for the execution of up to 1,000 functions concurrently, it provides a more cost-effective solution for parallel processing when compared to EC2, which would require the use of more instances and specialized Python parallelism techniques.

When considering the utilization of Python parallelism techniques on EC2, it is essential to acknowledge the inherent challenges and drawbacks associated with implementing multithreading in Python. One challenge is the Global Interpreter Lock (GIL), which limits the concurrent execution of threads in a Python application, thus impeding the full utilization of multiple CPU cores. To overcome this limitation, it is often necessary to run compiled scripts embedded within the Python code, employing additional tools such as Cython or Numba to compile performance-critical sections of the code. This, however, increases the complexity of the codebase and may introduce further challenges in terms of code maintenance and debugging. As a conclusion, the use of Amazon Lambda, with its inherent parallelism capabilities, offers a more efficient and cost-effective approach for processing large-scale NDVI computations compared to Amazon EC2 instances.

4.2. Results

4.2.1. Experiment one. Results

The results have shown that the entire architecture has generated processing times of between 9 and 15 seconds, with the use of a compiled binary being the limiting factor.

During the experiments, we processed 1,000 images and found that the worst-case scenario was a processing time of 15 seconds per image. The limiting factor, in this case, was the use of the compiled

10.3389/fhpcp.2023.1151530

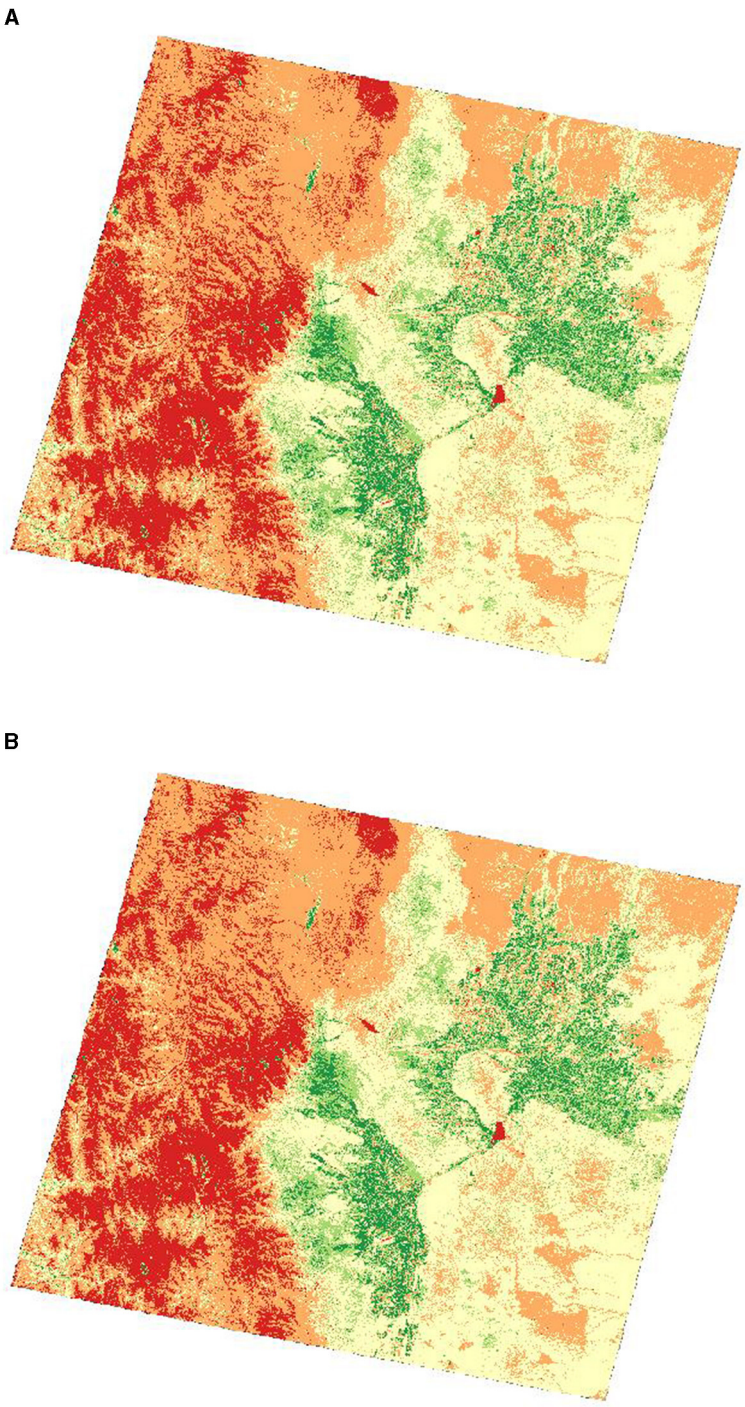

FIGURE 3
NDVI results comparison. **(A)** Output from SNDVI. NDVI calculation with SNDVI. **(B)** Output from QGIS. Baseline NDVI obtained with QGIS.

binary. Although the compiled binary improved performance, it couldn't be optimized any further.

The results of the cost analysis showed that for each parallel execution of 1000 images (if we store all the data in the S3 bucket for one month without deleting it) the total cost of services in AWS would be about $4.19 (US Dollars).

This cost analysis demonstrated the cost-effectiveness of our architecture. The use of the serverless architecture, along with a compiled binary containing GDAL, is a cost-efficient solution for performing NVDI calculations. It is important to note that this cost was calculated with the assumption of running all the processes for a month, and the cost may change depending on how many processes are executed and how often they are launched. However, this cost is low compared to the cost of maintaining and scaling a traditional cloud architecture.

TABLE 4 Lambda and EC2 comparison.

AWS service	Average time [hours]	Max time [hours]	Min time [hours]	Cost [US dollars]
Lambda	0.00335	0.0042	0.0025	0.06
EC2	1.358	1.380	1.341	0.061

To ensure accurate and precise NDVI calculations in our application, it is necessary to validate the results obtained by SNDVI with a standard NDVI computation tool. We compared the image generated by SNDVI with a baseline NDVI image generated from the same Dataset using QGIS (QGIS Development Team, 2023) Version 2.18.17 deployed on a Lenovo T480s Laptop (Intel Core i7 running Ubuntu OS 18.04.6 LTS). QGIS is a free and open-source cross-platform desktop geographic information system (GIS) application that supports viewing, editing, printing, and analysis of geospatial data. This application is widely used by agriculture specialists and scientists. In our work, we used the raster calculator of QGIS to do the NDVI computation and to generate the baseline tiff image for comparison.

Although the comparison with QGIS validated the quality of the NDVI computation obtained with SNDVI, a direct comparison of their computational performance is not feasible due to their differing nature. QGIS is a desktop application that runs locally, whereas SNDVI runs on remote services, leading to different execution conditions and monetary costs. Therefore, in this paper, we only focus on comparing the performance of SNDVI on similar cloud services.

Figure 3, illustrates the final result of SNDVI (see Figure 3A), and QGIS (see Figure 3B).

As can be seen in Figure 3, the SNDVI result is the same as the QGIS result, which indicates that the processing performed by SNDVI does not change the final result.

4.2.2. Experiment two. Results

Table 4 shows a comparison of the results obtained on Amazon EC2 with the Lambda results. Row 1 corresponds to the execution of SNDVI on Lambda, and row 2 to the execution on EC2. Column 2 details the average time required to compute 1,000 images on each platform. It should be noted that the startup time required to start the instance on EC2, as well as the time required for data transfer between the local machine and S3, and between S3 and EC2 (or Lambda), have not been included in this time. Columns 3 and 4 present the maximum and minimum time required to compute the 1,000 images. Finally, column 5 indicates the average costs of executing SNDVI on both platforms.

As can be observed from Table 4, the processing costs were approximately the same in both cases. The main difference between EC2 and Lambda was the execution time. The time required for the computation of 1,000 images on EC2 was approximately 411 times greater than in Lambda. This is because Lambda allows the execution of up to 1,000 functions in parallel, which on EC2 is not possible, as more instances and specific Python parallelism techniques would need to be utilized. It can be concluded that Lambda offered significantly lower execution times than the EC2 t2.medium instance with similar costs.

In our experiments, we have conducted tests using sequential execution on Amazon EC2 instances, specifically focusing on evaluating the capabilities of Amazon Lambda and EC2 at equivalent price points. It is important to note that in order to fully exploit the parallelization potential of EC2, larger and more expensive instances would be required, which would ultimately result in a higher overall cost for the system architecture. As Lambda allows for the execution of up to 1000 functions concurrently, it provides a more cost-effective solution for parallel processing when compared to EC2, which would need the use of more instances and specialized Python parallelism techniques.

5. Conclusions and future works

In this paper, we presented SNDVI, a new scalable native serverless framework for NDVI calculation. SNDVI was fully implemented within the AWS environment using Lambda functions for processing and S3 buckets for storage.

To fully implement SNDVI in Lambda, we took a thoughtful approach to leverage Lambda's capabilities effectively. Here are the steps we followed:

- Designing SNDVI components: We began by designing the different components required for NDVI calculation, taking into consideration how Lambda could be utilized. We analyzed the existing NDVI Python code and made necessary adaptations to ensure compatibility with Lambda.
- Handling dependencies: One challenge was incorporating the required libraries, such as GDAL, into the Lambda environment. To overcome this, we explored alternative approaches, including a "black box" solution, which involved packaging the necessary libraries along with the NDVI computation software. This allowed us to utilize the required dependencies seamlessly within the Lambda environment.
- Edge data processing: We also developed additional components to enhance the SNDVI implementation. For example, we designed software that could crop datasets on the edge, optimizing data processing efficiency. Furthermore, we utilized the Amazon CLI to facilitate smooth data uploads to S3, as Lambda has direct integration with S3, enabling seamless data transfer between the two services.
- Leveraging Lambda-S3 integration: We chose to use S3 as the storage solution due to its direct integration with Lambda. This integration allowed us to take advantage of the extensive capabilities provided by both Lambda and S3, enabling efficient data storage, retrieval, and processing.

In summary, our implementation involved adapting a standard NDVI computation code based on GDAL to run effectively in the Lambda environment, harnessing the benefits it offers. We designed and developed the necessary components, such as edge data processing and S3 integration, to fully leverage Lambda's capabilities. This approach allowed us to execute SNVDI computations efficiently and utilize Lambda's scalability and serverless architecture to handle incoming data requests effectively.

The design, implementation and validation of SNDVI allowed us to make the following contributions to the development of scalable cloud applications related to image processing:

- The combination of a serverless architecture, cloud storage and a compiled binary (containing the GDAL library, Scipy, and other packages) is an effective solution in terms of monetary costs and computing times.
- SNDVI architecture can be adapted to process other types of data such as time series.
- The binary compilation methodology can be followed by researchers to compile and run binaries with other Python packages (e.g., Pandas) to deploy their applications on serverless computing.
- Our main contribution to the agrotech ecosystem is to enable small and medium farmers access to cutting-edge technologies without prohibitive costs. This will enable them to have the tools to monitor and improve the health of their crops in the same way that large agricultural enterprises do.

We evaluated SNDVI performance through two experiments by analysing the computing time and the monetary costs. In the first experiment, we analyzed the overall functioning of our architecture, its performance and the quality of the output generated by SNDVI. The experiment was done by executing SNDVI with a Satellite Multispectral Dataset. In the second experiment, we conducted a benchmarking study between SNDVI and an EC2-based architecture in order to compare the performance of both approaches.

On the one hand, results from the first experiment demonstrated that the processing times ranged from 9 to 15 seconds for the entire architecture. The compiled binary used in the system was identified as the limiting factor, as further optimization was beyond the scope of our study. The cost analysis revealed that storing all the data in the S3 bucket for one month without deletion would amount to approximately 4.19 USD in total AWS service costs for the parallel execution of 1,000 images. Also, the NDVI obtained after SNDVI execution was validated through a comparison with the NDVI computed from the same dataset using QGIS (a widely-used free and open-source desktop GIS application). This comparison showed that the NDVI calculated with our architecture is the same as the one calculated by QGIS.

On the other hand, the benchmark between SNDVI and the architecture based on EC2 instances revealed that while monetary costs are similar, there was a significant difference in execution time. The time required to compute 1,000 images on SNDVI was approximately 411 times faster than on the EC2 architecture. This is due to Lambda's ability to execute up to 1,000 functions in parallel, a capability not easily available on EC2 without the use of more instances and specialized Python parallelism techniques. Therefore, Lambda offered significantly lower execution times with similar costs to the EC2 platform.

For future works, we will continue comparing the performance of serverless computing with other Amazon EC2 instances (tX.large and other tX.medium) and combinations of them. Also, we will analyse SNDVI performance on datasets larger than the one used in this paper (e.g., composed of high-resolution multispectral images acquired with Drones). Furthermore, we will evaluate the workload balance on SNDVI by deploying more NDVI computing functions and binaries on Lambda, to enable more than 1,000 executions in parallel. In addition, we will enhance SNDVI execution on EC2 by identifying computationally intensive tasks and executing them independently. Using Python's multiprocessing or threading modules, we will parallelize these tasks. Synchronization mechanisms like locks, semaphores, or queues will be employed to ensure data integrity. Monitoring and profiling tools will optimize performance, resulting in a faster and improved SNDVI execution on EC2.

Finally, we plan to extend this paper by adapting our framework to other binaries and datasets like time series data processing, e.g., localized frost prediction in agriculture.

Data availability statement

The raw data supporting the conclusions of this article will be made available by the authors, without undue reservation.

Author contributions

Conceptualization: LI and JV-P. Methodology: JV-P. Software, validation, and writing—original draft: LI and DP. Formal analysis and investigation: LI, DP, and JV-P. Resources and data curation: LI. Writing—review and editing: JV-P. All authors contributed to the article and approved the submitted version.

Funding

DP and JV-P acknowledge support through the IN-TIME (Grant Agreement 823934) and EYE (Grant Agreement 101007638) projects from the European Commission, the EDGECLOUD (RTI2018-096465-B-I00) and EDGEDATA (S2018/TCS-4499) projects. This research was partially funded by Know-Center GmbH. Know-Center was funded by the Austrian COMET Program–Competence Centers for Excellent Technologies—under the auspices of the Austrian Federal Ministry of Transport, Innovation and Technology, the Austrian Federal Ministry of Economy, Family and Youth and the State of Styria. COMET was managed by the Austrian Research Promotion Agency FFG.

Conflict of interest

LI was employed by Know-Center.

The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

JV-P declared that they were an editorial board member of Frontiers, at the time of submission. This had no impact on the peer review process and the final decision.

Publisher's note

References

Amazon (2023a). *Amazon Lambda*. Available online at: https://aws.amazon.com/lambda/ (accessed August 1, 2023).

Amazon (2023b). *Amazon Simple Storage Service*. Available online at: https://aws.amazon.com/s3/ (accessed August 1, 2023).

Amazon (2023c). *Amazon Web Services*. Available online at: https://aws.amazon.com/what-is-aws/ (accessed August 1, 2023).

Carlson, T. N., and Ripley, D. A. (1997). On the relation between NDVI, fractional vegetation cover, and leaf area index. *Rem. Sens. Environ.* 62, 241–252. doi: 10.1016/S0034-4257(97)00104-1

Castro, P., Ishakian, V., Muthusamy, V., and Slominski, A. (2019). The rise of serverless computing. *Commun. ACM* 62, 44–54. doi: 10.1145/3368454

Chu, L., Raghu, K. G., and Srivatsa, M. (2020). *IBM Cloud for Serverless Remote Sensing Data*. Available online at: https://www.ibm.com/cloud/blog/ibm-cloud-for-serverless-remote-sensing-data (accessed August 1, 2023).

Clark, J. A. (2019). *Pillow (PIL Fork)*. Available online at: https://github.com/python-pillow/Pillow (accessed August 1, 2023).

Cortesi, D. (2022). *PyInstaller Manual*. Available online at: https://pyinstaller.org/ (accessed August 1, 2023).

Drusch, M., Del Bello, U., Carlier, S., Colin, O., Fernandez, V., Gascon, F., et al. (2012). Sentinel-2: Esa's optical high-resolution mission for gmes operational services. *Rem. Sens. Environ*. 120:25–36. doi: 10.1016/j.rse.2011.11.026

Earth Resources Observation and Science (EROS) Center. (2013). *Collection-2 Landsat 8-9 OLI (Operational Land Imager) and TIRS (Thermal Infrared Sensor) Level-2 Science Products*. U.S. Geological Survey. doi: 10.5066/P9OGBGM6

Eismann, S., Scheuner, J., Van Eyk, E., Schwinger, M., Grohmann, J., Herbst, N., et al. (2020). A review of serverless use cases and their characteristics. *arXiv preprint arXiv*:2008.11110.

ESA (2023). *Copernicus Sentinel-2*. Available online at: https://sentinel.esa.int/web/sentinel/missions/sentinel-2 (accessed August 1, 2023).

Ferreira, K., Queiroz, G., Camara, G., Souza, R., Vinhas, L., Marujo, R., et al. (2020). "Using remote sensing images and cloud services on aws to improve land use and cover monitoring," in *2020 IEEE Latin American GRSS ISPRS Remote Sensing Conference (LAGIRS)* (IEEE), 558–562. doi: 10.1109/LAGIRS48042.2020.9165649

Iacono, L. E., Poletti, J. L. V., Garino, C. G., and Llorente, I. M. (2018). Performance models for frost prediction in public cloud infrastructures. *Comput. Inform.* 37, 815–837. doi: 10.4149/cai_2018_4_815

Jiang, Z., Huete, A. R., Chen, J., Chen, Y., Li, J., Yan, G., et al. (2006). Analysis of ndvi and scaled difference vegetation index retrievals of vegetation fraction. *Rem. Sens. Environ*. 101, 366–378. doi: 10.1016/j.rse.2006.01.003

NASA (2023). *MODIS*. Available online at: https://modis.gsfc.nasa.gov/ (accessed August 1, 2023).

QGIS Development Team (2023). *QGIS Geographic Information System*. QGIS Association.

Shahrad, M., Balkind, J., and Wentzlaff, D. (2019). "Architectural implications of function-as-a-service computing," in *Proceedings of the 52nd annual IEEE/ACM International Symposium on Microarchitecture*, 1063–1075. doi: 10.1145/3352460.3358296

USGS (2023). *Landsat 8*. Available online at: https://www.usgs.gov/landsat-missions/landsat-8 (accessed August 1, 2023).

Vazquez-Poletti, J. L., and Llorente, I. M. (2018). Serverless computing: from planet mars to the cloud. *Comput. Sci. Eng*. 20, 73–79. doi: 10.1109/MCSE.2018.2875315

Warmerdam, F. (2008). "The geospatial data abstraction library," in *Open Source Approaches in Spatial Data Handling* (Springer) 87–104. doi: 10.1007/978-3-540-74831-1_5

Wu, J., Wu, M., Li, H., Li, L., and Li, L. (2022). A serverless-based, on-the-fly computing framework for remote sensing image collection. *Rem. Sens*. 14, 1728. doi: 10.3390/rs14071728

Yang, G., Liu, J., Qu, M., Wang, S., Ye, D., and Zhong, H. (2021). "Faasrs: Remote sensing image processing system on serverless platform," in *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)* (IEEE), 258–267. doi: 10.1109/COMPSAC51774.2021.00044

Zhai, Y., Liu, M., Zhai, J., Ma, X., and Chen, W. (2011). "Cloud versus in-house cluster: evaluating amazon cluster compute instances for running mpi applications," in *State of the Practice Reports*, 1–10. doi: 10.1145/2063348.2063363

Zhang, J., Huang, Y., Pu, R., Gonzalez-Moreno, P., Yuan, L., Wu, K., et al. (2019). Monitoring plant diseases and pests through remote sensing technology: A review. *Comput. Electr. Agric*. 165, 104943. doi: 10.1016/j.compag.2019.104943