International Journal of Geo-Information

MDPI

Article

Geospatial Serverless Computing: Architectures, Tools and Future Directions

Sujit Bebortta [1,†], Saneev Kumar Das [1,†], Meenakshi Kandpal [2], Rabindra Kumar Barik [3,*] and Harishchandra Dubey [4,*]

1 Department of Computer Science and Engineering, College of Engineering and Technology, Bhubaneswar 751003, India; sujitbebortta1@gmail.com (S.B.); saneevdas.061995@gmail.com (S.K.D.)
2 School of Computer Engineering, KIIT Deemed to be University, Bhubaneswar 751024, India; meenakshikandpal14@gmail.com
3 School of Computer Applications, KIIT Deemed to be University, Bhubaneswar 751024, India
4 Centre for Robust Speech Systems, University of Texas at Dallas, Richardson, TX 75080-3021, USA
* Correspondence: rabindrafca@kiit.ac.in (R.K.B.); harishchandra.dubey@utdallas.edu (H.D.); Tel.: +91-8763293589 (R.K.B.); +1-(469)-618-2851 (H.D.)
† These authors contributed equally to this work.

Received: 15 March 2020; Accepted: 4 May 2020; Published: 7 May 2020

Abstract: Several real-world applications involve the aggregation of physical features corresponding to different geographic and topographic phenomena. This information plays a crucial role in analyzing and predicting several events. The application areas, which often require a real-time analysis, include traffic flow, forest cover, disease monitoring and so on. Thus, most of the existing systems portray some limitations at various levels of processing and implementation. Some of the most commonly observed factors involve lack of reliability, scalability and exceeding computational costs. In this paper, we address different well-known scalable serverless frameworks i.e., Amazon Web Services (AWS) Lambda, Google Cloud Functions and Microsoft Azure Functions for the management of geospatial big data. We discuss some of the existing approaches that are popularly used in analyzing geospatial big data and indicate their limitations. We report the applicability of our proposed framework in context of Cloud Geographic Information System (GIS) platform. An account of some state-of-the-art technologies and tools relevant to our problem domain are discussed. We also visualize performance of the proposed framework in terms of reliability, scalability, speed and security parameters. Furthermore, we present the map overlay analysis, point-cluster analysis, the generated heatmap and clustering analysis. Some relevant statistical plots are also visualized. In this paper, we consider two application case-studies. The first case study was explored using the Mineral Resources Data System (MRDS) dataset, which refers to worldwide density of mineral resources in a country-wise fashion. The second case study was performed using the Fairfax Forecast Households dataset, which signifies the parcel-level household prediction for 30 consecutive years. The proposed model integrates a serverless framework to reduce timing constraints and it also improves the performance associated to geospatial data processing for high-dimensional hyperspectral data.

Keywords: cloud computing; serverless framework; Cloud GIS; geoportals; scalability; latency

1. Introduction

Recent years have witnessed enormous growth in the production of massive digital data, which has facilitated the perception and cognition of several features from the physical world. The intersection of geospatial intelligence with these physical features have led to the understanding and management of geo-referenced activities [1,2]. With the technical evolution of modernized platforms and tools, the perception and representation of data have transformed considerably.

The extraction of geospatial information has contributed immensely towards social, industrial and healthcare sectors. The information obtained with the aid of these models usually pose intensive computing and storage requirements. Thus, more promising technologies like cloud computing, edge computing, fog computing and others are emerging rapidly for the management and monitoring of such massive information. Cloud computing infrastructure has provided more intuitive capabilities and has substantially transformed the way in which spatio–temporal data are represented. Towards this end, geoportals play a relevant role in the construction of Cloud Geographic Information System (GIS) platforms [3–5]. The implementation of geoportals provides a web-based platform to channelize geographic data. The geoportals in convergence with Cloud GIS framework acts as a key attribute to facilitate the exchange of geospatial data acquired from a multitude of locations over the Internet.

Cloud computing paradigm has served several complex applications and has greatly reduced the storage and computational costs. The notion of serverless computing came into existence as an anticipation to manage increasingly scaling data collection. The technological growth in capabilities of the serverless approaches have made them more versatile in handling latency issues and for delivering uninterrupted services to users [6,7]. Thus, the need for handling the diverse requirements of the system can be efficiently and reliably managed. The serverless platform substantially extends the cloud-based functionalities and provides outsourcing of infrastructures vital for performing several complex computations. Based on the evolutionary concepts brought in by the serverless platform, two fundamental categorizations can be made i.e.,

1. Back-end-as-a-Service (BaaS): It is conceptually analogous to Software-as-a-Service (SaaS). This involves placing off-the-shelf solutions at the server side by adding more modularity to the system [8]. The BaaS services represent domain generic components which can be incorporated into several user-defined external applications. Firebase is one of the Google's web-based analytics platform which employs BaaS framework for managing users' data components.
2. Functions-as-a-service (FaaS): It provides a comprehensive domain for developing and implementing server side applications [8,9]. FaaS gives more emphasis on the operations involved in the development of an application, rather than the host instances and application as a whole. Thus, FaaS makes it an event-based framework for advanced data analytics. A popular example of this framework is Amazon Web Services (AWS) Lambda, which is capable of processing billions of events in shorter intervals of time.

Figure 1 is an illustration of trade-offs between the traditional and FaaS-based approach for software deployment. FaaS can be distinguished from the traditional server side deployment as it considers different operations to develop applications without the involvement of host instances.

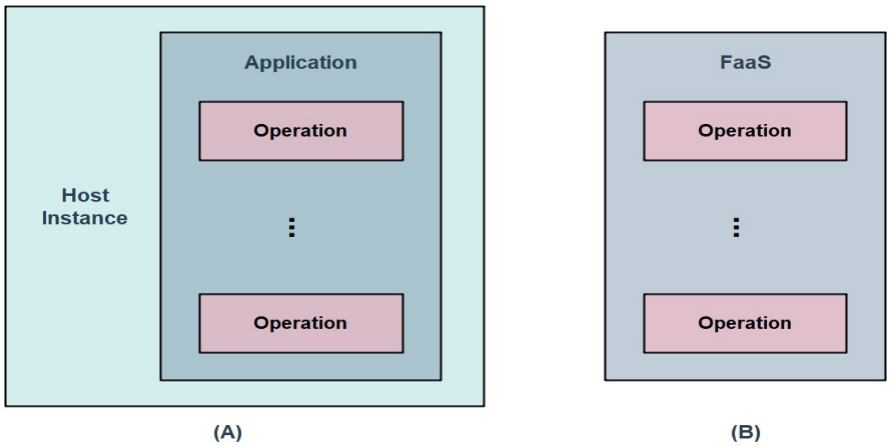

Figure 1. (**A**) represents the traditional approach for the deployment of software on the server-side; (**B**) depicts Functions-as-a-service (FaaS)-based approach for deploying the software pertaining to different operations.

In this paper, we address the applicability of serverless framework for leveraging Cloud GIS models over a distributed platform. Platforms like Amazon AWS Lambda, Microsoft Azure Functions, Google Cloud Functions and others provide an Application Programming Interface (API) for the smoother implementation of code snippets that are supported in many different programming languages. The Python programming language is common among maximum serverless frameworks. Thus, Python code snippets written for GIS implementations over Jupyter notebooks can serve as the base for our suggested framework. Many libraries exist, but proper APIs over serverless computing paradigm needs further research and development. Our model suggests such a way to apply GIS implementations over serverless framework. The datasets generally are getting larger day by day and as such, the need for GIS technology grows. High-dimensional datasets are difficult to process over local processing engines and such data are called Spatial Big Data (SBD) [10]. Serverless frameworks can aid in processing such SBD in an efficient and reliable way. The presence of distinct limitations in existing models for the handling of large-scale geospatial data prompted us to develop a relatively novel framework. The framework was developed to serve the collection, processing and representation of massive geospatial data. The proposed model encompasses all the functionalities of the cloud services along with BaaS and FaaS solutions. Some of the most important issues associated with handling geospatial data like collection, storage and dissemination over multiple platforms were addressed. The proposed system greatly reduces the waiting time as compared to the traditional cloud model. The execution time is a drawback of most server-based platforms for performing massive geospatial data analysis by providing a scalable and automated platform for their processing. The findings in this paper show the ease of developing more user-centric applications for managing and mapping the geo-referenced features with high portability and usability.

Objectives

The objectives of this paper are as follows:

- Elaborate on the trade-offs between the server-based approach and the proposed serverless approach for geospatial data.
- Some of the critical issues like latency, scalability, cost- and security-based issues are discussed for computing platforms in convergence with the proposed framework.
- Some of the prominent frameworks that have facilitated the implementation of serverless approaches are provided.
- A taxonomy of some state-of-the-art technologies and some relevant works deemed relevant to implementing Cloud GIS models are discussed.
- The proposed framework and its compliance with some available tools for the abstraction and simulation of geospatial data is addressed.
- The overlay analysis along with heatmap generation and cluster analysis to signify the complexity of geospatial data processing was visualized using two application case studies i.e., the Mineral Resources Data System (MRDS) dataset and the Fairfax Forecast Household dataset.
- The architectural and service-level limitations, future scope and concluding remarks are presented.

2. Research Methodology

Figure 2 presents the work flow behind the performed research work. The first stage signifies the study of serverless computing paradigm from various Internet sources for the creation of a knowledge base. The second stage indicates the performed literature survey on the articles that actually instantiates serverless frameworks and their applicability in various significant domains. The third stage is the tabulation of the performed survey in order to present the reader with works performed. Furthermore, the fourth and fifth stages visualize the process behind the construction of the proposed framework by merging Python libraries with GIS for smoother deployment of the Python GIS code snippets. Finally, the sixth stage presents the exploration of future challenges and research directions in the context of implementing GIS tasks in a serverless environment.

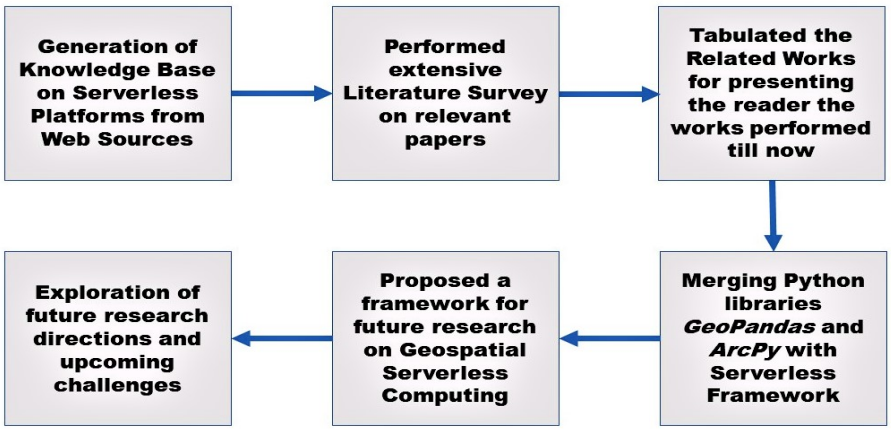

Figure 2. Visual representation of the research methodology behind our research work.

3. Related Work

Baldini et al. [7] conducted a survey on relevant platforms that exist in order to perform serverless computing. The study focused on futuristic problems that needs to be tackled in the domain of serverless computing. The authors presented a "serverless platform". The architecture suggested that servers are mandatory but the developer is not at all concerned about managing servers. This paper stated that the architecture is an automated process used to decide the number of servers and capacity required for computation. Future research directions were explored for serverless computing by presenting open research problems as a questionnaire.

Crespo et al. [11] presented the challenges faced when bio-informatics applications are implemented using a serverless platform. They explored the concept of CloudDmetMiner, which is a framework designed using the Drug Metabolism Enzymes and Transporters(Dmet)-Miner algorithm. Furthermore, the Amazon AWS Lambda platform was used to run on Single Nucleotide Polymorphism (SNP) datasets, which are highly relevant to bio-informatics applications. The proposed work was verified and validated through examples successfully running over considered SNP datasets for preliminary tests. A comparison of the advantages provided by CloudDmetMiner framework over a serverless ecosystem and a pure cloud environment was provided. It was observed that infrastructure is not a matter of concern in case of serverless computing.

Niu et al. [12] presented a proof-of-concept case study for biomedical research. This paper focused on the alignment of around 20,000 protein sequences using the serverless computing paradigm. The resource constraints that actually existed while using a serverless paradigm includes disk space, duration of run-time, primary memory space and number of virtual processing units. They used a distributed approach in order to deal with the resource constraints in a smarter way. Tools compared in order to process a huge bio-informatics data with lesser execution time in serverless platform were AWS Lambda and Google Cloud Functions. It was observed that AWS Lambda has much higher efficiency as compared to that of Google Cloud Functions.

Kim et al. [13] proposed a framework, Flint, for visualizing the concept of big data analytics using serverless computing. From a developer point of view the usage of PySpark with Flint was presented. The complete implementation, associated challenges and performance analytics were addressed. It was inferred that big data analytics in a serverless platform are completely feasible and analytical queries can be handled well. It was inferred that the Flint architecture is a perfect choice for the analysis of big data, overcoming the challenges of Spark defaults for ad-hoc analysis and many more data analysis tasks.

Ishakian et al. [14] suggest that cloud suppliers and enterprise corporations are implementing Artificial Intelligence (AI) to either isolate themselves, or deliver valuable services to customers. The authors appraised the perfection of a serverless computing environment by speculating on massive neural network prototypes. Experimental calculations were implemented on a AWS Lambda

environment by means of the MxNet deep learning framework, which showed the speculating latency within a satisfactory range. The results highlighted the latency of warm requests in a practical range when compared to the latency of cold requests, which are very high in range. Hence, the authors proclaimed that this issue has to be resolved if applications of AI and customized service level agreements are to be merged.

Anand et al. [15] discussed key aspects to constructing a GPS tracking mechanism for vehicles using serverless computing paradigm. It was comprised of low power system, real-time tracking, auto-theft notification by the practices of geofences to increase the proficiency of storing and analyzing the data. It enhanced the characteristics and guaranteed safe and improved services in context to vehicle tracking systems. The authors proposed a pro-active GPS tracker that was equipped with a apt mobile/web application for observing real-time outcomes, which made it more user friendly. It could be observed on the website that was exclusively built for the corresponding vehicle tracking solution. To overcome security aspects, if a user was identified outside of the geographical boundary, notifications were mailed to the user. These geographical boundaries or geofences could be modified based on the user's wish.

Hellerstein et al. [9] focused on the significant gaps between first generation serverless computing and modern computing. The paper stated that AWS, a public cloud, offers a fundamental new computing platform: the biggest collection of data competence and distributed computing, with power constantly available to the common public, managed as a service. The article also explained that serverless computing offers the eye-catchy notion of a platform in the cloud where developers merely upload their code, and the platform executes the codes on their behalf as required. It was observed that developers should not worry themselves with operational servers and they pay only for the resources they use. This paper instantiated serverless as FaaS supported by a standard library, the various multi-tenanted, auto scaling services provided by the vendor. The limitations presented were that the vendors pay no attention to the significance of efficient data processing and hinder the expansion of distributed systems, which is the hub for most of the innovation of modern computing. This paper focused on the following shortcomings in FaaS as: limited lifetime, communication through slow storage and others.

Malawski et al. [16] conducted a study on serverless infrastructures for the purpose of background processing required in applications of Web and Internet of Things (IoT). It presented the prime characteristics of serverless computing infrastructure and compared with the Infrastructure-as-a-Service (IaaS) cloud model. A prototype that included three distinct platforms: AWS Lambda, Hyperflow workflow engine and Google Cloud Functions in the context of scientific workflows was presented. The authors validated the proposed approach by using Montage workflow, which signified an astronomic application in a real-time environment. This work provided some effective ways to manage resources since resources are a matter of concern for scientific workflows.

Varghese et al. [17] discussed the advancements of cloud computing infrastructures towards a decentralized approach of computing where the data centers are not centrally located. In order to implement next generation infrastructure for cloud computing, it focused on challenges that needs to be tackled. The prime focus of this paper was on discussing models of cloud computing which needed peer interest towards shaping the future generation of cloud computing.

Lee et al. [18], claimed the dynamic behavior of serverless computing for parallel execution of partitioned tasks over small function instances. The results corresponding to CPU performance, memory and disk utilization for performing computationally intensive tasks on serverless environments were provided. Four notable serverless computing environments were evaluated: Google Cloud Serverless Functions, Amazon AWS Lambda, Apache OpenWhisk and Microsoft Azure Serverless Functions. A set of functions were deployed for concurrent processing of data to illustrate the performance trade-offs between virtual machines and serverless computing environments. It was inferred from the experimentation that the serverless platforms provided an explicit processing of the data. The potential scalability traits of the serverless platforms were analyzed in context to resource

utilization and cost effectiveness. It can be deduced that the serverless computing platforms in their present form are usually associated with working on ephemeral workloads, which in the future may find more space with enhanced configurations to handle complex workloads.

Table 1 provides a comprehensive study for the different works discussed along with the significant contributions made in each towards assessing the capabilities of serverless platforms.

Table 1. A summary of some notable works, along with their contributions for assessing the capabilities of serverless platforms.

Author(s)	Contributions
Baldini et al. [7]	Emphasized on vital research directions in serverless computing by presenting some open research problems as a questionnaire.
Crespo et al. [11]	Used the concept of CloudDmetMiner for bio-informatics applications using Amazon AWS Lambda platform.
Niu et al. [12]	Performed alignment of around 20,000 protein sequences using serverless computing paradigms.
Kim et al. [13]	Presented the usage of PySpark with the proposed Flint architecture.
Ishakian et al. [14]	Studied the aptness of serverless computing environments for speculating massive neural network prototypes.
Anand et al. [15]	Provided GPS-based tracking system for real-time management of vehicles with the aid of serverless computing paradigm for theft detection on a web-based platform using mobile applications.
Hellerstein et al. [9]	Focused on the significant gaps between first generation serverless computing with modern computing.
Malawski et al. [16]	Presented a comprehensive study on serverless environments for processing of Web based applications.
Varghese et al. [17]	Discussed the advancement of cloud computing infrastructures towards and provided solutions for handling the challenges associated with implementing next generation infrastructure for cloud computing.
Lee et al. [18]	Studied the dynamic behaviour of serverless computing environments for parallel execution of partitioned tasks over small function instances. The results corresponding to the CPU performance, memory and disk utilization for performing computationally intensive tasks on serverless environments were presented.

4. Background

4.1. Distributed Computing for Geospatial Information Sharing

The advancements in modern GIS systems and diverse data collection frameworks have substantially led to the generation of bulk geospatial data. The computing and data collection resources associated with these systems are usually distributed over a large geographic area [19]. The processing of bulk geospatial data generated from heterogeneously collocated devices impose challenges on the conventional databases and computing resources for performing advanced querying and processing of complex geospatial data. Hence, the need for a distributed computing platform becomes inevitable for managing the requirements of different remotely located clients for geospatial data processing. A distributed computing platform facilitates the sharing of geospatial information over multiple clients and diverse computing platforms. Several studies have focused on the efficacy achieved by implementing distributed computing platforms for geospatial information querying, processing and sharing [20–22]. This framework was observed to overcome most of the challenges associated with local computing resources and databases by advantageously facilitating the collection and distribution of

heterogeneous geospatial data. Distributed computing framework leverages the cloud infrastructure for integration of the locally distributed computing resources. It can aid the rendering of a computationally dynamic framework for the logical sharing of the computing resources on a global scale. This essentially solves the complex geospatial data processing associated challenges. Traditional distributed computing platforms have evolved drastically with the increase in computational demands. In order to facilitate improved delivery of services, different distinct classes namely, mobile computing, fog computing and edge computing frameworks came into existence [22–25].

4.2. Edge Computing for Geospatial Analytics

Edge computing is a relatively newer adaptation of the distributed computing paradigm for providing an efficient utilization of computing power by facilitating computing services on the edge. This provides computing resources to heterogeneous devices associated with a network in close proximity with geospatial data sources, or computing devices [22–24]. Edge computing overcomes some of the crucial challenges associated with conventional distributed computing paradigms. Management of latency issues in large networks, securing the users privacy and secure data sharing, providing improved connectivity, low power consumption and so on are some of the issues that can be efficiently managed with edge computing. These capabilities are extremely beneficial for large-scale distributed geospatial systems for achieving a secure, reliable, and energy efficient framework. A major advantage of using edge computing for geospatial information sharing and processing is that it facilitates high service discoverability and availability by providing the computing services on the edge. Hence, the connected computing devices do not have to wait for longer periods to avail the services. Further the capabilities of edge computing framework can be explicitly enhanced by the integration of some cutting-edge architectures like the peer-to-peer framework to produce a highly hybridized architecture [25].

4.3. Server-Based Approach

The processing of rich geospatial information over cloud based servers for extracting and representing useful spatio-temporal patterns is called server-based processing [7]. In a server-based approach, the server is often remotely located from the client and is responsible for managing and executing the applications on the client side. Thin client computing is a popular example of server-based processing, which involves light-weight computing devices (referred to as thin clients) which are able to represent remote applications. These clients typically establish a connection with the server and can provide an account of various applications running over the server. Some prominent real-world applications using the SBA are Citrix, Tarantella, GraphOn, and many more.

4.4. Serverless Approach

The Serverless Approach (SLA) does not indicate the processing of data in the absence of servers, but leverages more operational capabilities. Scalability, maintenance, fault tolerance and management of resources are a few to mention [6,16]. SLA offers much more convenience in terms of cost for administering physical high computing devices. It automatically provisions the appropriate resources as the computational demands scale-up, hence providing seamless capabilities to the hosted clients. Hence, SLA would be a superior alternative to the SBA. Some popular examples of SLA are AWS lambda, OpenLambda, Google Cloud function and many such.

Figure 3 presents the comparison of Google Trends data for the keywords “Server based” and “Serverless” computing based on performed research works over the last five years in a worldwide fashion. The blue line indicates the growth in serverless computing domain whereas the red line signifies the fall in server-based approach.

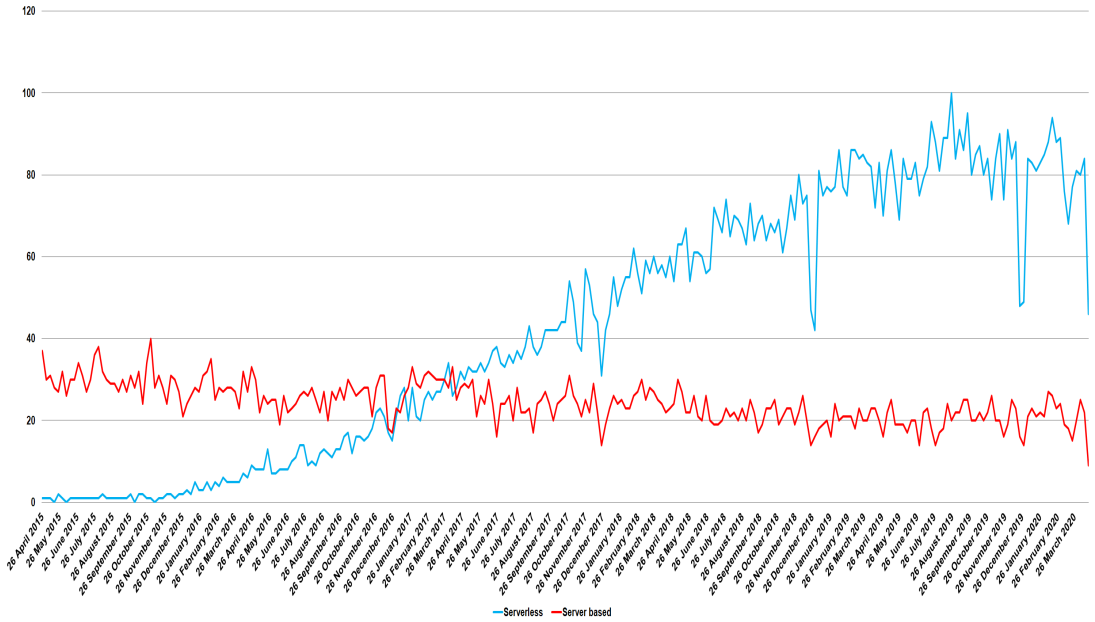

Figure 3. Trade-offs between serverless and server-based approach obtained from Google Trends.

4.5. State Based Properties

The management of states in a serverless platform is quite challenging due to lack of persistence among the functions. Thus, in context to SLA, we are more concerned with the scalability of the resources, therefore the statelessness property attributes in providing concurrency of execution [6]. This feature greatly assists in extending the capabilities of the system for processing geospatial data. However, in order to preserve any information beyond a specific course of time, the applications must channelize the information into some stateful components. The AWS Lambda function is one such example of a stateless platform that facilitates smoother concurrent execution of the instances of any component, without the necessity of providing each instance with more resources (Figure 4).

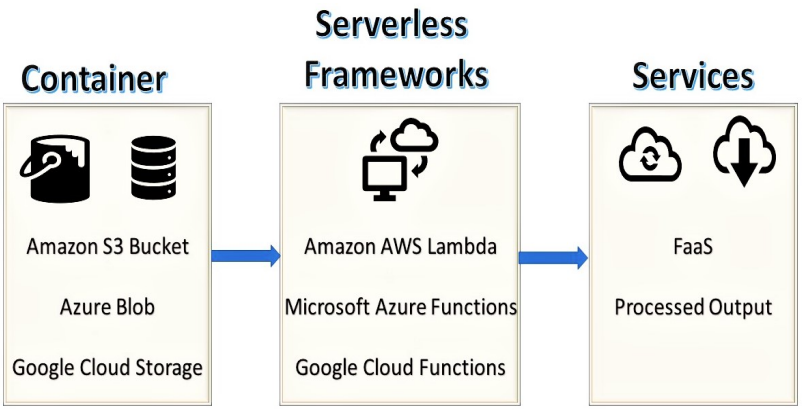

Figure 4. Generalized architecture for existing serverless computing frameworks.

4.6. Managing Latency

As the resources scale-up exceedingly in serverless environments, the interaction among the components present in the system increases further introducing latency. This increases the overall complexity of the system. In an SBA application, the latency induced among the components can be mitigated by co-locating the components, or by importing them into the same process. Here co-location of the components involves placing the components on the same host instances. These techniques assist in optimizing the performance of the system and can be applied concurrently or individually

to the components. Further, optimization can be obtained by augmenting the performance of the communication channel, by employing some specialized protocols and appropriate data formats.

4.7. Amazon AWS Lambda

This platform of serverless computing is the most popular among all existing platforms. The reason is the versatility of this platform in supporting code snippets of Node.js, Python, Java and C#. It is a containerized concept where each instance acts as a container produced by Amazon Linux AMIs with required amount of server specifications allocated for the code snippet to run in the environment. The raw code cannot be run directly in the serverless platform and thus an S3 bucket is used to pack all the required contents to execute the code snippet. This framework is completely stateless. The efficiency of this framework is commendable. The maximum execution time per request is 900 s in this platform [26]. In our context, running geospatial applications coded in Python using GeoPandas or ArcPy is a difficult yet efficient way with futuristic importance to visualize the maps within a shorter interval of time. The cost optimization in context to geospatial serverless computing is a critical issue and needs further research.

4.8. Google Cloud Serverless Platform

This serverless platform is widely used due to the trust of Google involved in it. Furthermore, going deep into the technical aspects, the platform supports code snippets written using Python and Node.js presently. The runtime environment that is Knative built with the aid of Kubernetes allows use of this serverless framework for multiple purposes. This paradigm comes with the ease of coding and comparatively is new to these serverless functionalities. The maximum execution time per request is by default 60 s and it can be extended up to 540 s which is comparatively very low further proving its efficacy [26]. The disadvantage is the maximum number of functions allowed per project is 1000 by default.

4.9. Microsoft Azure Serverless Platform

Azure Serverless platform supports code snippets written in C#, Python, Java, F#, PHP and Node.js. Its unique feature is its concurrent execution being completely dependent on triggers and bindings with no specific limits. The scaling is metered or manual in type. This platform can run a code snippet of high memory within a few minutes. Furthermore, the scaling is so perfect that user don't have to focus on volume of the workload. This platform has an attractive code editor which improves its user base. Two of its prime features include Continuous Integration (CI) and Continuous Delivery (CD). The monitoring of code snippet submitted is well interfaced. The maximum execution time per request is 300 s by default which is up-gradable to 600 s in this platform [26]. This framework in context to geospatial serverless applications can prove itself to be the best since it aids in dealing with complex orchestration problems and is based on triggers and bindings. The process of building followed by debugging then deploying and finally monitoring is made simple and easy in this platform.

4.10. Terminologies Used

4.10.1. Overlay Analysis

The term overlay analysis in GIS is used to signify superimposition of multiple features as vector maps [27]. In our context, overlay analysis refers to data points being overlaid over the Open Street Map. Weighted overlay analysis involves reclassifying the spatial layers depending on the influence of each layer defined by weights. This analysis is much important in order to perform further visualizations. Various symbols can be used to plot data points over the map based on latitude and longitude values. Overlay operations can be computationally intensive for traditional computing systems to handle SBD [27]. Hence, serverless framework can be a good choice for handling real-time requirements of most GIS applications.

4.10.2. Point-cluster Analysis

Point-cluster analysis is a symbolic presentation which is used to reduce and precise large number of data points. The analysis of geographical patterns is an indispensable task and it can be performed through point-cluster analysis. Finding areas with low or high concentration is possible through point-cluster analysis. A point pattern may be thought of as a set of geographical coordinates over a defined region where the events of interest are recorded [28]. In case of SBD, data points are densely packed and to analyze the geographical patterns and trajectories is a hectic task. Thus, point-cluster analysis makes the process easier to detect the patterns by clustering dense points into individual clusters.

4.10.3. Heatmap Analysis

To find the highly dense regions over the map, that is in some cases we require to restrict our analysis to a specific region where density of points is very high. The heatmap plays an important role in specifying the regions based on density and differentiated by color depths. The heatmap needs to be properly analyzed and parameter for weight distribution is required to be specified.

4.10.4. Cluster Analysis

Spatial Partitioning is a term used to partition the spatial data points based on proximity which is often called Spatial Clustering [29]. Spatial Clustering is a method to cluster the similar data points where similarity is based on features and the points in the same cluster are different from points in the other clusters [30]. The cluster analysis can be performed using two well-known clustering techniques used in machine learning namely Density-Based Spatial Clustering of Applications with Noise (DBSCAN) Clustering and K-means Clustering. Both the techniques can be used over the heatmap or the overlay to allot specific cluster IDs to various data points based on the features. Furthermore, we need to specify the number of clusters we want to visualize in order to generate that much cluster IDs and also the clustering points. In DBSCAN, we need to specify the border points to be treated as noise. The cluster density is determined by two parameters namely *eps*, which signifies the radius of the circle and *minpts*, which specifies the minimum number of points within that circle [31].

4.11. Geospatial Big Data Processing

The geospatial data can be represented as three primary models namely raster data which signifies grids of pixels, vector data signifying lines or polygons of pixels and graph data which represents the spatial networks [32]. The geospatial data inthe present-day scenario are increasing in aspects such as volume, variety and processing complexity, which makes the available spatial processing tools unable to process and analyze the data which leads to an SBD [10]. Spatial data mining plays a pivot role in the process of analyzing the SBD. Some algorithms which can aid in the processing of such data are Spatial Auto-regressive (SAR) Model, Markov Random Field Classifiers, Gaussian Process Learning or the hybridization of these algorithms [33]. The era of real-time computing demands the need of geospatial data processing for real-time environments and thus requires the methods to handle SBD. Spatial Hadoop, GIS on Hadoop, Declustering and Space Partitioning are some computational paradigms to handle SBD [34].

5. Proposed Framework

The fundamental motivations behind integrating the serverless platform for geospatial analytics lies in achieving high operational capabilities and near real-time processing of geospatial big data. As discussed in the previous section that the SLA adds more scalability and maintainability in a highly heterogeneous computing environment, thereby providing a low cost alternative for processing massive geospatial data. It also provisions the users, or clients to process their data on high-end computing platforms by virtualizing the computing resources. This architecture removes the requirement for physically deploying the devices and assists to meet the dynamic requirements of the

clients by implementing a pay-as-you-go model. Hence towards this end we propose the applicability of the serverless framework for geospatial big data analytics. Figure below represents the proposed serverless framework for geospatial data analytics. Here the web-based geoportals are responsible for geospatial data discovery and access which can be captured from different heterogeneously dispersed computing devices. The geospatial data acquired from different heterogeneous sources may be channelized over the internet gateways to respective client/server for processing the data. The geoportals may further assist in geospatial information creation and visualization for remotely acquired geospatial data by making optimal use of GIS platforms. The information obtained from this framework plays a crucial role towards the development of Spatial Data Infrastructures (SDI). The application servers facilitate the clients with a server to run their requests in a multi-client environment. The clients associated with this framework comprise of thin clients, thick clients, and mobile clients. The thin clients may be the users associated with web servers for accessing the geospatial object (i.e., a collection of attributes and instances constituting a geospatial database [19]). These clients may further wish to display and manipulate the geospatial objects as per their requirements using some GIS tools or application software referred to as thick clients. The mobile clients are capable of performing the operations on geospatial data locally through mobile computing devices like tablets, personal computers (PCs), mobile phones and so on.

For the purpose of storing the acquired geospatial objects Amazon simple storage service (S3), buckets can be efficiently used. In relation to the geospatial information sharing platform, the S3 bucket can provide high discoverability of the resources. Uploads from the S3 bucket, internet gateways, application servers and geoportals trigger the AWS lambda function (or, simply known as the lambda function) to forward the jobs to the cloud service providers for processing. This gives rise to the concept of FaaS by making use of the serverless framework. Lambda functions can be invoked by clients or users, depending on the requirements of the application it is deemed to serve. The serverless platform makes use of microservices for processing of complex application requirements generated by application server module. The microservice architecture disintegrates large complex application modules into simpler sub-modules which can be quickly executed. Further this provides the clients more flexibility so as to add new modules or to make modifications to existing ones as per the requirements of specific geospatial application. The serverless framework provides an efficient and convenient way for processing of geospatial information acquired from heterogeneous computing devices and also for handling the increasing computational demands in a multi-client environment.

The proposed framework is based on the implementation strategy required to merge the geospatial services with serverless computing paradigm. The existing frameworks for serverless computing supports inputs as code snippets written in Node.js, C#, Java and Python. Thus, in order to implement geospatial services in a serverless ecosystem we need a novel and workable architecture. As we know, geospatial data formats include raster format and vector format, which can be processed using tools like QGIS, ArcGIS and many more. Geospatial applications require high level processing but is a completely centralized concept. Our novelty towards this domain is to perform the processing in a decentralized and distributed platform. Many platforms and libraries of Python are important in the processing of geospatial data. Thus, our proposed architecture aims at using Python code snippets written for GIS implementation to be used in a serverless computing framework. The proposed architecture can further lead to a widespread use of GIS technology in a serverless platform which shall exempt the developers from the high-level of pre-processing that is currently required in all geospatial applications.

Figure 5 represents a high-level three-layered architecture of the proposed framework signifying data processing in the first layer. The second layer deals with transformation of processed output to Python code snippets, which is practically possible since many GIS tools working presently use Python at its back-end. The third layer signifies the use of serverless frameworks, which supports Python code snippets to work over the framework. Figure 6 signifies a generalized model to present the client and serverless computing interaction. It can be realized through different geospatial data

access and generation mechanisms. This includes geoportals, internet gateways, application servers and Amazon S3 bucket corresponding to different client models working on different platforms with the essence of serverless computing paradigm.

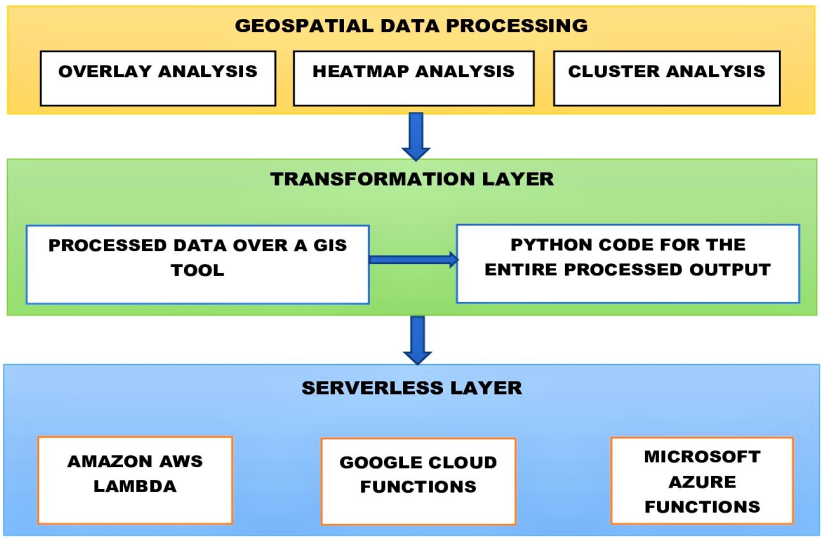

Figure 5. Proposed framework for deploying geospatial serverless computing.

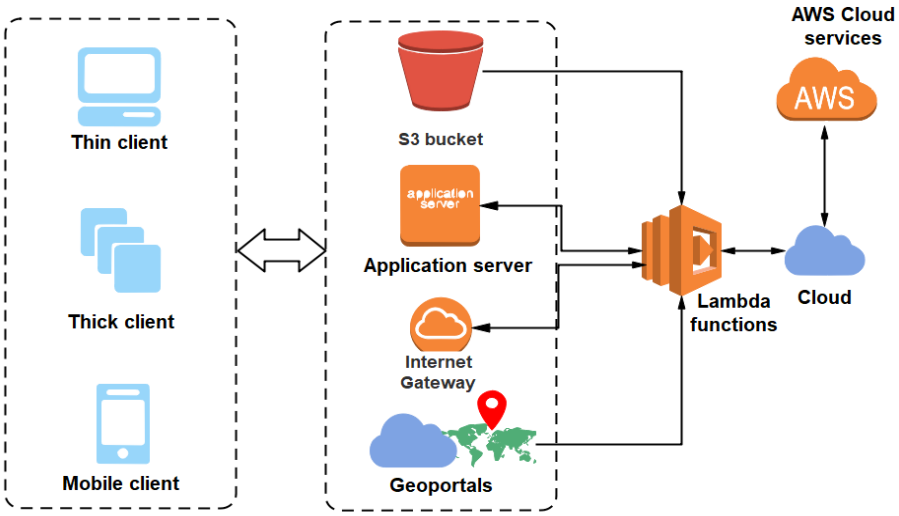

Figure 6. Serverless framework for geospatial data analytics with Amazon Web Services (AWS) Lambda embedded to perform specific geospatial data analytics activation functions.

6. Performance Evaluation

In Table 2, a comparison of some of the performance metrics for different emerging technologies which are commonly used for modeling geospatial data are presented. Here, we compared four computing paradigms namely, the Server-Based Approach (SBA), edge computing, distributed computing and SLA [35–37]. A traditional SBA usually portrays very low scalability and reliability metrics due to the static nature of system. High implementation and maintenance costs are involved if one has to work with the existing system and the data resources scale-up continually, thereby putting more load upon the data centers. The highly distributed architecture of the edge computing platform further increases the chances of security threats, as a web-based platform is used. The distributed computing approach may not always process the massive geospatial data in real time [38–40]. This is due

to the complexity in the architecture of distributed platforms which involve more number of servers and operation sites, which often leads to complex management and high maintenance costs [22,41]. In contrast to these limitations, the SLA provides a highly scalable, reliable and comparatively low cost framework with minimal requirements for managing the servers. It also provides minimum latency for processing large-scale geospatial data. Thus, it can be inferred that the proposed SLA outperforms most of the computing paradigms in terms of reliability, scalability, processing speed and cost (Figure 7).

Table 2. Comparison of the performances for different emerging technologies in context to the performance metrics.

Technology	Scalability	Reliability	Real-time	Speed	Cost	Security
SBA	L	L	H	M	H	H
Edge Computing	H	H	H	H	M	L
Distributed Computing	H	H	M	H	H	M
SLA	H	H	H	H	M	M

L: Low; M: Medium; H: High.

The performance metrics corresponding to each of the discussed technologies can be evaluated by assigning numeric values from 1 to 3 for the criteria on the basis of which each technology's performance was assessed. Here, value 1 denotes a low or insufficient performance, 2 denotes a medium or moderate performance level and 3 signifies a high performance level. All these measures were inferred by qualitatively analyzing the respective Telecommunication Standardization Sector of the International Telecommunications Union (ITU-T) documentations and from the above studies [22,35–41].

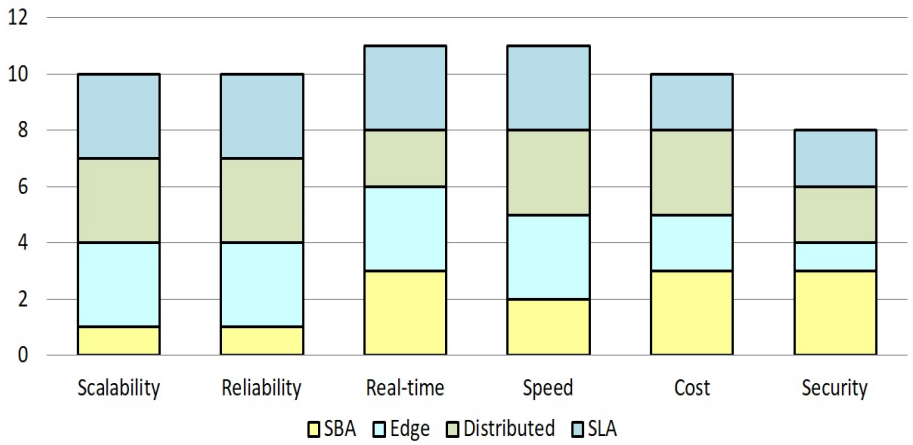

Figure 7. A qualitative analysis of different Ouality of Service (QoS) parameters for the studied technologies with the proposed Serverless Approach (SLA) framework.

7. Experimental Setup

The visualization of geospatial big data was a complex task and takes a lot of time to implement using local engines. Thus, we proposed a serverless framework for big data visualization in a geospatial context. The current packages and libraries were not sufficient enough to perform the visualization. Python is a common language for most serverless architectures. The serverless architecture is expected to run Python code snippets. We presented the visualizations over a local engine to signify the necessity of serverless computing paradigm to be merged with GIS technology. Since the datasets considered were complex and large in size, it is computationally intensive to label the data points over the map using local engines. The Python libraries and packages that can be used for the serverless computing paradigm are as follows:

The library of **GeoPandas** in Python consists of the following packages:

1. CartoPy: This package aids in visualization of maps and is a cartographer's tool. It is an object-oriented approach towards geospatial analysis and well-known for processing of raster and vector formats [42].
2. Shapely: Sometimes we deal with two-dimensional data for some geospatial applications and this package acts perfectly for manipulation of such geometric data in a Cartesian plane [43].
3. Fiona: This package has a clear API, which attracts users to read as well as write various geospatial data as well as zipped contents. Geospatial data can be collected in single-layers inside multi-layer file formats [44].
4. PyProj: As the name suggests PROJ refers to projections and changes related to geospatial data. This package is used only to transform coordinates and well project it [45].
5. RTree: For indexing of geospatial data, this package is used and it supports many spatial indexing features like multi-dimensional loading, bulk loading, disk serialization, nearest neighbor search and many more [46].
6. Descartes: It works like the popular plotting library matplotlib and works on geometric data for plotting various essential paths which are of interest for geospatial application development [47].
7. Rasterio: In order to use data in geospatial raster formats, it becomes essential to input the file formats of type GeoTIFF and others. This package reads and writes such raster data and outputs an API based on N-dimensional arrays [48].

The following modules can also be used for the generation of Python code snippets:

ArcPy: This module is used to manipulate the ArcGIS tool in order to directly achieve the efficiency of Python in GIS tasks.

PyGRASS: This Python package works on the popularly known GRASS GIS 7 to work with geospatial data. Thus, PyGRASS library signifies the efficiency of the GRASS tool in a Python environment. As we studied, there exists various Python modules to work on geospatial data. Thus, our proposed architecture aims at using Python code snippets written for GIS implementation to be used in a serverless computing framework. The proposed framework can further lead to a widespread use of GIS technology in a serverless platform, which shall exempt the developers from the high-level of pre-processing that is currently required in all geospatial applications.

Assisting Tools

The tool used for visualization purpose was the QGIS software package, where it required us to insert the datasets with latitude and longitude values in order to plot it over the Open Street Map. The overlay analysis was performed to visualize location wise data points for both the datasets. The overlay analysis was followed by the point-cluster generation where we signified the points based on distance, represented by red triangles. The heatmap analysis was performed next to signify the densely populated data points which refers to the regions of interest. The heatmap analysis was then followed by the DBSCAN clustering and K-means clustering, one in each dataset with 50 cluster IDs each. The relevant statistical plots were generated using QGIS Processing Toolbox to validate the results of visualization.

8. Results and Discussions

In this section, the geospatial big data was visualized after processing with multiple functionalities. The dataset considered may be supposed to act as a spatial big data. The proposed framework suggests using the serverless computing paradigm; currently, processing geospatial data over serverless architecture is a difficult task. We visualized the outputs over QGIS software package, which uses Python at its back-end. The considered tool was suitable for implementation over a serverless framework.

8.1. Case-Study 1: World Mineral Resources Mapping and Analytics

8.1.1. Dataset Specification

The Mineral Resources Data System (MRDS) dataset provides information regarding worldwide mineral resources, which includes both metallic and non-metallic resources. It constitutes different attributes, like geographic location, types of minerals and their names, along with other physical and geographical traits. In this study, the worldwide mineral resource potential of different regions were analyzed and the corresponding geospatial mappings are provided. The potential of mineral commodities pertaining to a specific geographical area indicates the likelihood of occurrence for the mineral resources across a specific area [49,50]. The information for the mineral deposits can be obtained over a massive spatial domain and can be used in developing geo-database for performing some specialized tasks. These include querying, visualizing, analyzing and updating geospatial information. The customized overlay maps for different mineral commodities were generated using the QGIS tool for the MRDS dataset. This facilitated the study of the distribution of various mineral resources over the extensive spatial domain. The identification of different geographic boundaries on the basis of availability of the mineral resources can be performed. In this paper, the DBSCAN [51] algorithm, a density-based clustering technique popularly used for identification of dense information from a given feature space was implemented. Here, DBSCAN technique was employed to identify the geographic areas with dense mineral resources in contrast to those with lower density of occurrences. The technique is quite popular for the discovery of densely scattered features in large spatial domains [52,53] as it can adaptively handle the noises associated with data without affecting the performance of the algorithm. This framework can lead to the forecasting of future mineral productions and their spatial analysis by integrating the locally available geospatial data with remote geospatial datasets. The integration of these services with the serverless platform can result in lower computational complexities as well as reduced waiting times, which is usually desirable for processing of massive geospatial data in near real-time applications. The total data present in the dataset is 304,633 rows which signifies it is a high-dimensional data.

8.1.2. Visualization of MRDS Dataset

The weighted overlay specifying the data points generated for the instances of the MRDS dataset over the Open Street Map is represented in Figure 8. The generation of weighted overlay is the primary task to perform in order to further visualize the data points with numerous visualization strategies. Figure 9 presents the point-cluster analysis for the MRDS dataset. Small brown circles and red triangles represent points and clusters respectively. The point-cluster analysis was followed by the generation of heatmap which is illustrated in Figure 10. The heatmap identifies highly dense regions where maximum data points lie. The heatmap allowed us to specify the regions of interest which can be differentiated based on color intensity. The clustering mechanism used over this dataset was DBSCAN clustering, which was implemented using three parameters *eps*, *minpts* and the number of clusters. Here, the number of clusters was set to 50. The results for DBSCAN clustering is depicted in Figure 11 and the corresponding highly dense region with allotted cluster ID 1 is shown in Figure 12. To validate the clustering mechanism, the statistical bar plot for the attributes Country and Cluster ID was generated to visualize 50 clusters corresponding to countries represented in Figure 13. The mean and standard deviation plot is shown in Figure 14, which is plotted between the attributes Region and MRDS-ID. It signifies the mineral resources present in various regions worldwide. The corresponding maximum value, mean value and minimum value are visualized. The labels could not be provided in the maps as the dataset is acting as an SBD, and it is practically a tough task to analyze such data over local engines.

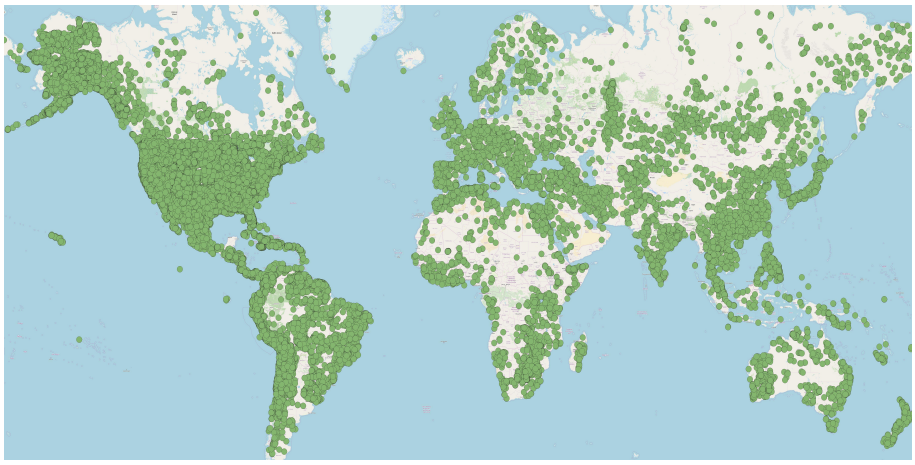

Figure 8. Weighted overlay analysis of the Mineral Resources Data System (MRDS) dataset over Open Street Map without labels.

Figure 9. Point cluster analysis with clusters in red triangles and points in brown small circles.

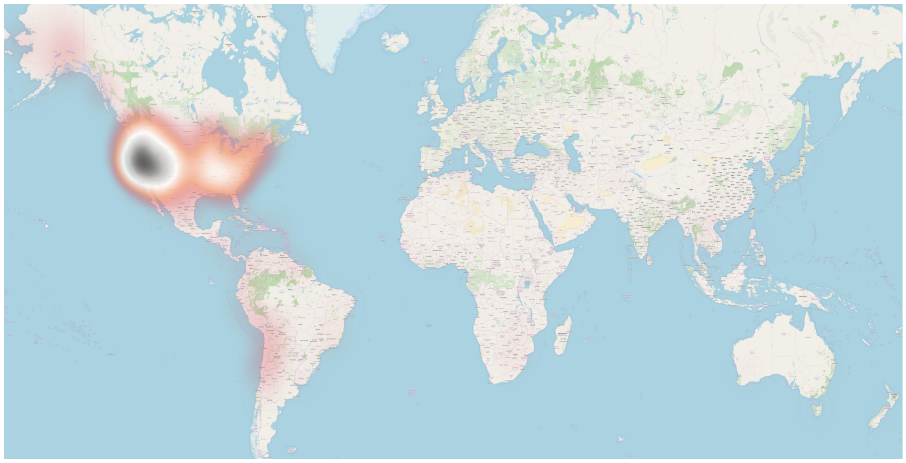

Figure 10. Heatmap analysis to determine the highly dense region by means of varying colors as per the features.

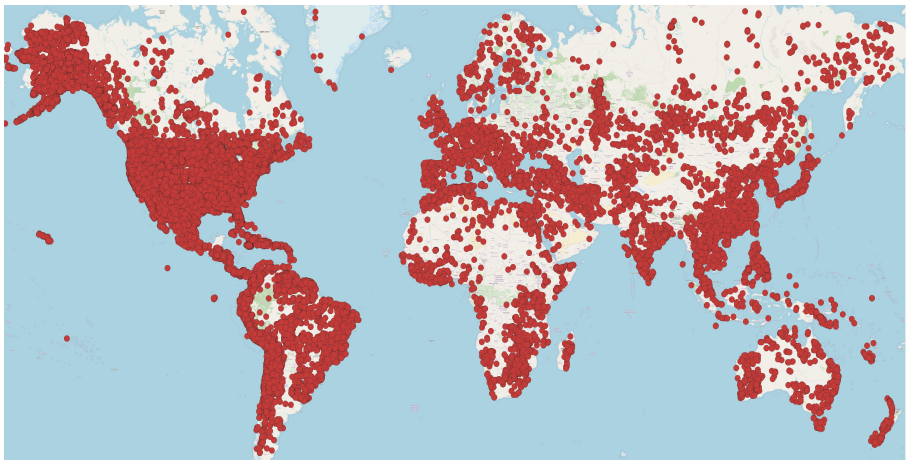

Figure 11. Cluster analysis with the aid of Density-Based Spatial Clustering of Applications with Noise (DBSCAN) clustering over the generated map.

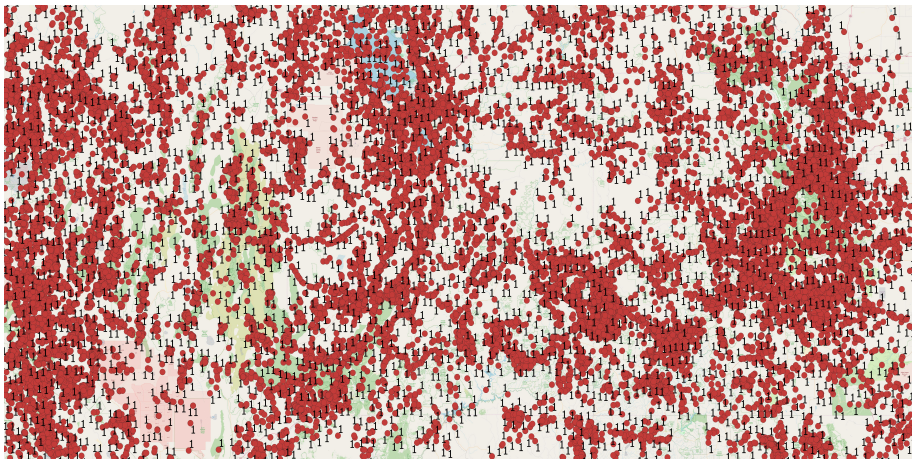

Figure 12. The visualization of a single cluster with cluster ID 1, which is the highly dense region, United States as per the MRDS dataset.

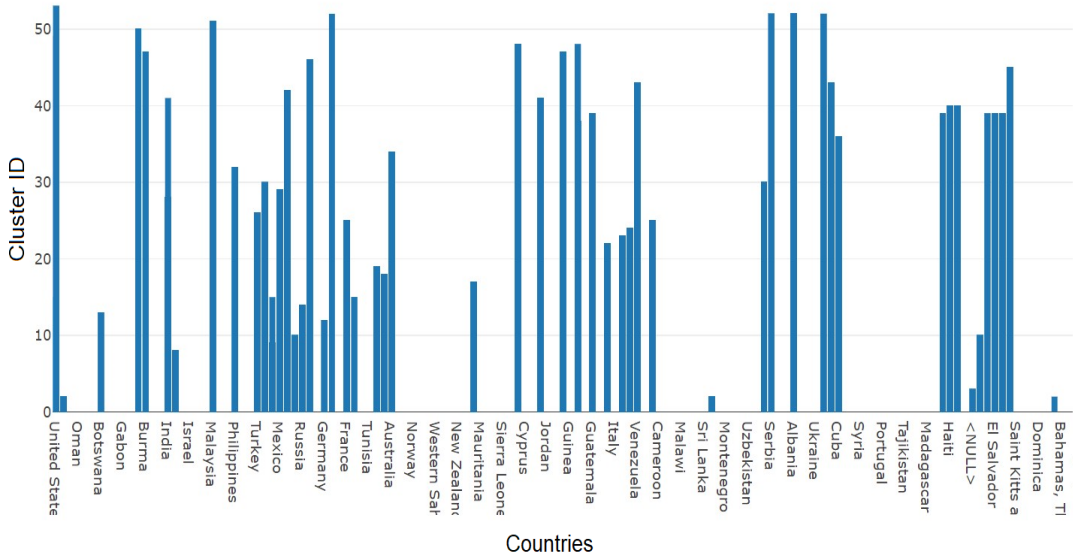

Figure 13. Bar plot generated between attributes "country" and "Cluster ID" to visualize which country belonging to the DBSCAN cluster.

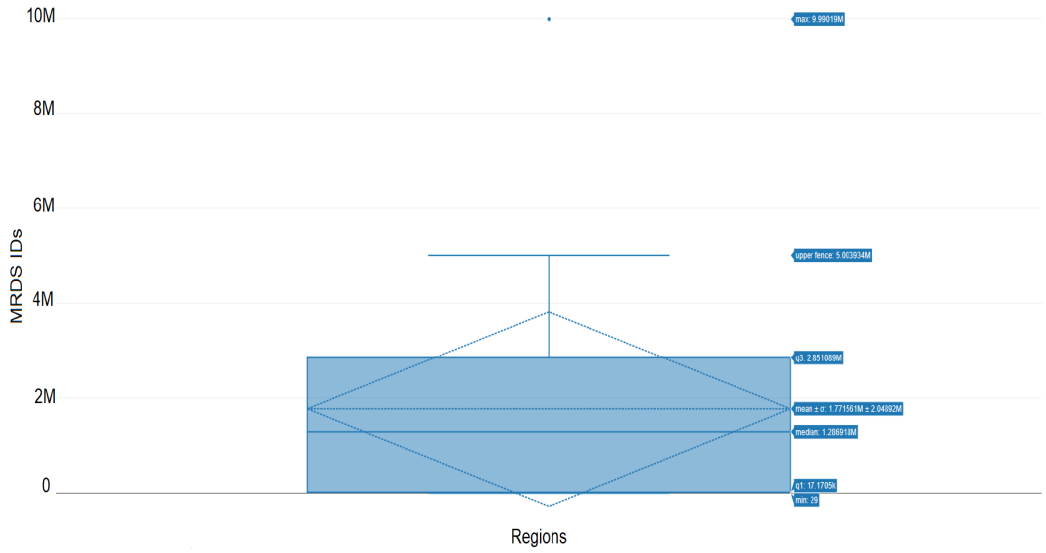

Figure 14. Mean and standard deviation plot between attributes "region" and MRDS-ID.

8.2. Case-Study 2: Household Forecasting for Fairfax County

8.2.1. Dataset Specification

We considered the Forecast Household dataset published by Fairfax county GIS, which provides 30 year prediction of households over the Fairfax region. The dataset consists of latitude and longitude values for the purpose of geospatial prediction [54]. The housing units were tracked at the parcel level. The dataset consists of the parcel IDs, Maximum Likelihood Estimates (MLE) of the recently developed households, Land Use Code (LUC), housing unit type (i.e., single family, multiplex unit, townhouse unit, duplex unit and so on), area in acres and the years corresponding to the housing units' establishment. The total data present in the dataset is 342,308 rows. which signifies that it is a high-dimensional data.

8.2.2. Visualization of Fairfax Forecast Households Dataset

The weighted overlay analysis for the 30 year households forecast of Fairfax County was generated over the Open Street Map. The data points over the map were visualized where each data point refers to each row or alternatively a feature of the dataset. The weighted overlay is illustrated in Figure 15. The point-cluster analysis for the considered dataset is shown in Figure 16 where small brown circles and red squares signify points and clusters respectively. The heatmap generated with various color intensities to designate the highly dense region is shown in Figure 17. The K-means clustering technique applied with 50 clusters was processed early as compared to case study 1 and can be visualized in Figure 18. The plot between the object-ID and the predicted households for year-30 was a bar-plot which can be seen in Figure 19. Furthermore, the statistical mean and standard deviation plot for cluster ID versus the predicted households for year 30 showing the 50 cluster IDs in diverse colors is presented in Figure 20. This dataset also acts as an SBD.

These visualizations take much time to implement and so the proposed framework provides a solution to tackle the time constraint with the use of serverless computing paradigm to reduce the time associated to analyze such big data and concurrently produce faster results. We analyzed the two geospatial big datasets using the QGIS open source package with a future research direction towards converging the visualization tool with a serverless computing environment.

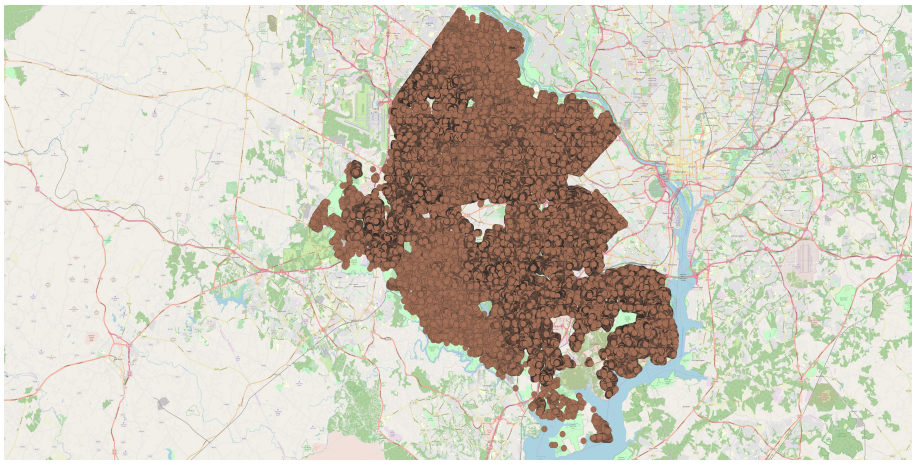

Figure 15. A weighted overlay analysis for data points plotted over the Open Street map.

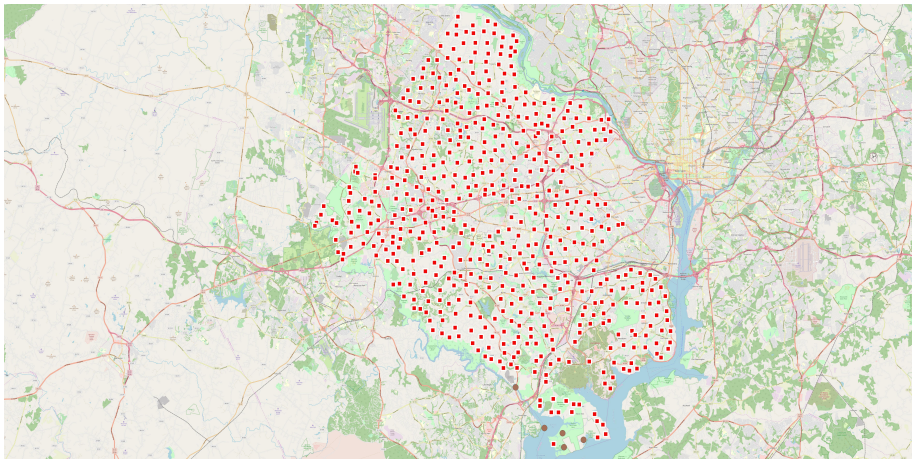

Figure 16. Point-cluster analysis where points are represented by brown circles and clusters by red squares signify the households over the region.

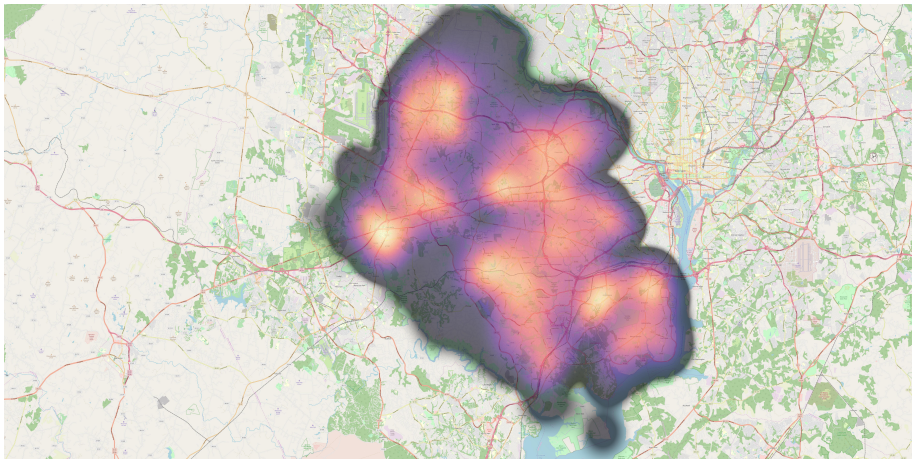

Figure 17. Heatmap analysis performed to designate regions with high density of households.

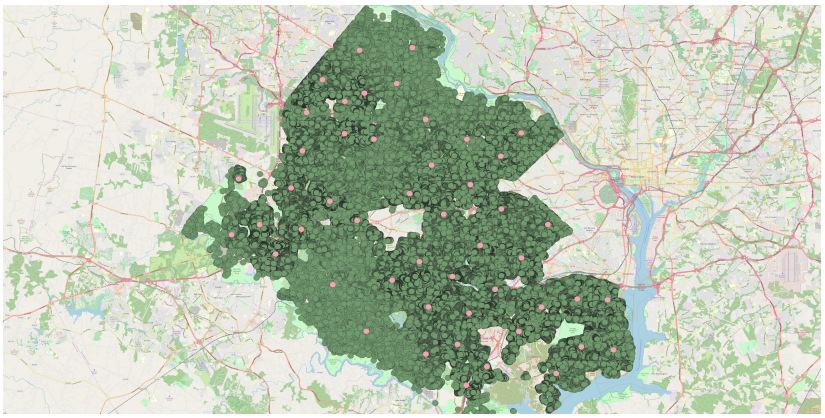

Figure 18. Cluster analysis with the aid of K-means clustering over the generated map with 50 cluster IDs.

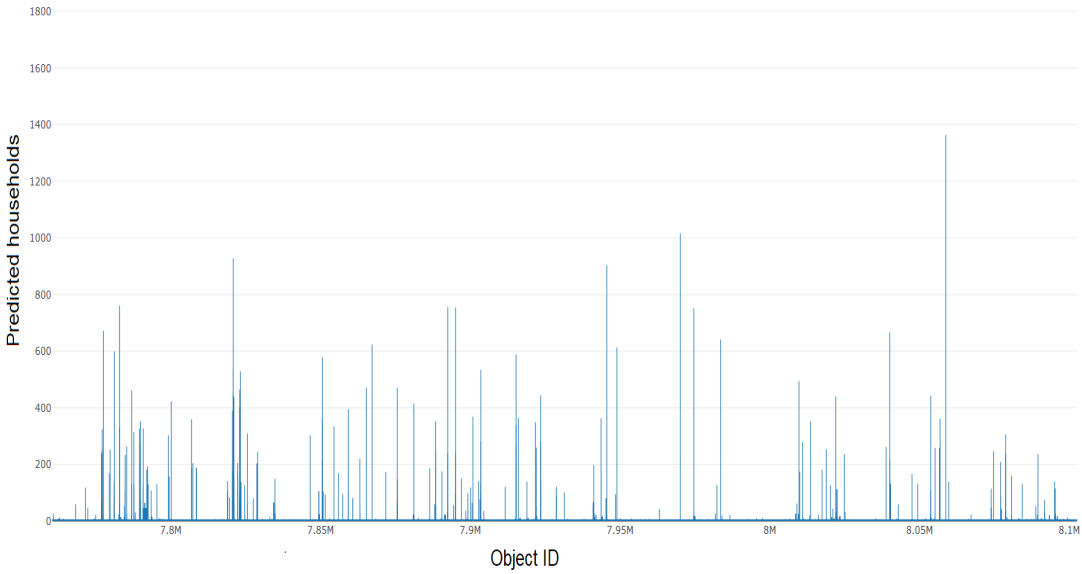

Figure 19. Bar-plot to signify the predicted households for year 30 based on Object-ID.

Figure 20. Mean and standard deviation plot to signify the K-means cluster IDs associated to the predicted households for year 30.

9. Execution-Level Performance of the Experimental Study

The hardware specifications of the local engine over which experimentation was performed include an Intel Core i5-8265U CPU, 8.00 GB RAM, 64-bit Windows 10 Home Single Language 2019 operating system, a 256 GB Solid State Drive and a 1 TB Hard Disk Drive. In this section, we compare the execution-level performance for each operations performed using QGIS tool. For the two application case studies, we performed different operations such as weighted overlay analysis, point cluster analysis, heatmap generation and spatial cluster analysis. The execution times for performing above operations were compared. Figure 21 provides the comparison between execution time (considered in seconds) for performing weighted overlay analysis, point cluster analysis and heatmap generation corresponding to the two case studies. Due to substantial variance in scale, the execution times for performing cluster analysis are shown separately in Figure 22. Here, execution time corresponding to the two case studies have been provided for two different clustering techniques i.e., K-means clustering and DBSCAN algorithm. It is worth noting that the MRDS data considered in case study 1 is computationally intensive for performing geospatial operations using local engines as the time taken to form the spatial clusters using the DBSCAN technique amounts to 23 min and 10 s (≈1390 s). Hence, the serverless computing platform is inevitable towards performing such complex computations.

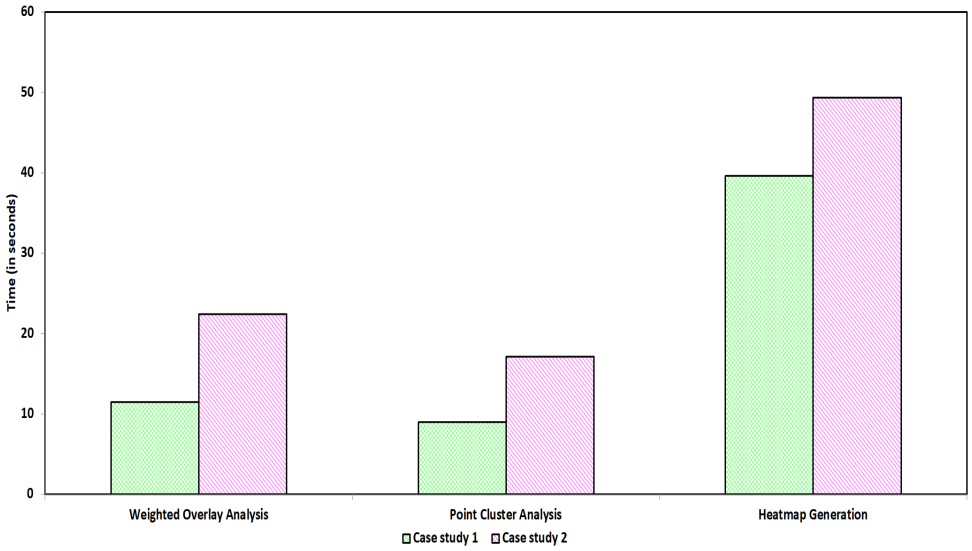

Figure 21. Execution time for the different operations performed in case studies 1 and 2.

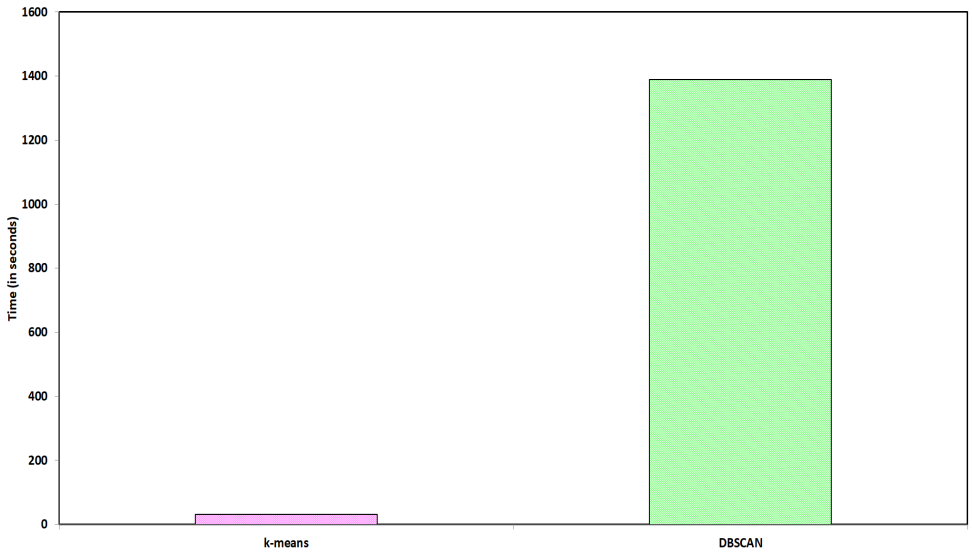

Figure 22. Execution time for K-means clustering and DBSCAN algorithms.

10. Architectural and Service Level Challenges

In this paper, we presented an insight towards integration of serverless platform with GIS framework for advanced processing of geospatial big data. Although different global enterprises like Amazon, Google and Microsoft have adopted the serverless working model to meet advanced computational demands of their users; however, there are still numerous architectural as well as service level challenges which requires consideration before completely adopting these platforms [8]. Below we discuss some important issues and challenges for integration of serverless computing paradigms in association with geospatial data analytics:

- The complexity associated with the processing of massive geospatial data for creating client specific application poses architectural challenges on local engines. We seek the use of serverless platforms for their execution. As discussed in Section 5, the microservice architecture can be efficiently used in this scenario to implement complex applications by dividing them into simpler sub-modules thereby providing the users with more flexibility and quicker outcomes; however, this architecture in the serverless scenario is still a much debated topic as no specific standards have been provided on integrating these outcomes to achieve fully usable geospatial applications.
- The GIS tools are currently being used for the creation of geospatial databases; however, their visualization lacks the proper definitive packages to support the serverless architecture.
- As the computing requirements scale up with the increase in acquired geospatial data, more functions are required to be invoked for performing the desired operations. Most of the available serverless computing platforms lack the availability of sufficient number of functions to execute the increasing number of incoming requests. For instance the Google Cloud serverless platform provisions only 1000 functions per 297 projects.
- Since the geospatial data may necessarily constitute of geolocated information which may encompass real world location information, hence the security and privacy of these information requires to be preserved. The serverless architecture, due to its decentralized nature, lacks a centralized control over the activities in the network. Thus making the data susceptible to security breaches.

11. Future Research Directions

The domain of serverless computing has proven itself to be a highly efficient FaaS solution. Handling complex data may be one of the limitations in a serverless computing paradigm. Serverless computing signifies a Pay-as-you-Go model; however, the time complexity associated during the deployment of complex blocks of code snippets needs further optimizations in order to satisfy customer needs. SBD needs complex processing and in order to exempt the developer from processing the code snippets, we require increased competency of the serverless frameworks. The future research directions can be summarized as follows:

- Geospatial data optimization is a prime concern in the context of the deployment of GIS data in a serverless platform. The reason being the cost one pays to process and deploy unoptimized code snippets over any serverless framework.
- Resource utilization techniques need further research in order to improve the size of code snippets that can be run at a single time slot in a serverless platform.
- The extensions of codes that are currently supported needs to be upgraded to implement code snippets of at least those extensions which are popular in today's computing society.
- Geospatial applicability in fields of agriculture, crime analysis, satellite imagery and many such needs further research to provide an exclusive platform based on Python for easier implementation in any serverless computing paradigm.
- Code optimization is a major challenge which needs to be dealt with in such a way that implementing GIS applications in a serverless platform would be worthy.

12. Conclusions

Serverless computing is going to be a dominant platform in upcoming years. In this paper, we proposed a framework for geospatial applications to run over a serverless computing paradigm. The required geospatial application computations are complex. Many existing serverless platforms can build, debug, deploy and run code snippets written in Node.js, Java, C# and Python. The problem with geospatial applications is that it uses specific tools to execute like QGIS, ArcGIS and others. The proposed framework suggests the use of Python libraries at the back-end for processing geospatial data. Since code snippets in Python are acceptable for AWS Lambda, Google Cloud Functions and Microsoft Azure Functions, the proposed framework provides a future research direction towards implementing geospatial code snippets coded in Python to be built successfully in a serverless environment. The presented results focus on geospatial data analytics with the aid of a tool where Python works at the back-end. The tool can be improvised to integrate serverless computing frameworks for diminishing the associated timing constraints. The competency of considered serverless platforms is well-known to many researchers, we tried to provide a novel approach towards reducing the processing complexity and timing constraints associated with geospatial applications by embedding it in a serverless computing paradigm. In this paper, the architectural and service-level challenges are presented to make the reader aware of the limitations posed by the proposed framework. The computational performances are also analyzed to show the present-day necessities due to which switching to serverless paradigm is highly inevitable.

Author Contributions: Sujit Bebortta: writing—original draft preparation and visualization. Saneev Kumar Das: writing—review and editing, methodology and validation. Meenakshi Kandpal: visualization and investigation. Rabindra Kumar Barik: conceptualization, resources and project administration. Harishchandra Dubey: conceptualization, writing—review and editing. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: We are extremely grateful to the reviewers for their valuable remarks in improving the overall quality and presentation of the manuscript.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

GIS	Geographic Information Systems.
MRDS	Mineral Resources Data System.
BaaS	Back-end as a Service.
Faas	Function as a Service.
DMET	Drug Metabolism Enzymes and Transporters.
AWS	Amazon Web Services.
SNP	Single Nucleotide Polymorphisms.
GPS	Global Positioning System.
IaaS	Infrastructure as a Service.
CPU	Central Processing Unit.
SBA	Server-based Approach.
SLA	Serverless Approach.
CI	Continuous Integration.
CD	Continuous Delivery.
SDI	Spatial Data Infrastructure.
Amazon S3	Amazon Simple Storage Service.
DBSCAN	Density-Based Spatial Clustering of Applications with Noise.
SBD	Spatial Big Data.

SAR Spatial Autoregressive Model.
QGIS Quantum Geographic Information Systems.
API Application Programming Interface.
ITU-T Telecommunication Standardization Sector of the International Telecommunications Union.
QoS Ouality of Service.
MLE Maximum Likelihood Estimation.
LUC Land Use Code.

References

1. Dold, J.; Groopman, J. The future of geospatial intelligence. *Geo-Spat. Inf. Sci.* **2017**, *20*, 151–162. [CrossRef]
2. Soille, P.; Burger, A.; De Marchi, D.; Kempeneers, P.; Rodriguez, D.; Syrris, V.; Vasilev, V. A versatile data-intensive computing platform for information retrieval from big geospatial data. *Future Gener. Comput. Syst.* **2018**, *81*, 30–40. [CrossRef]
3. Iosifescu-Enescu, I.; Matthys, C.; Gkonos, C.; Iosifescu-Enescu, C.; Hurni, L. Cloud-based architectures for auto-scalable web Geoportals towards the Cloudification of the GeoVITe Swiss academic Geoportal. *ISPRS Int. J. Geo-Inf.* **2017**, *6*, 192. [CrossRef]
4. Barik, R.K.; Kandpal, M.; Dubey, H.; Kumar, V.; Das, H. Geocloud4GI: Cloud SDI Model for Geographical Indications Information Infrastructure Network. In *Cloud Computing for Geospatial Big Data Analytics*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 215–224.
5. Barik, R.K.; Dubey, H.; Mankodiya, K.; Sasane, S.A.; Misra, C. GeoFog4Health: A fog-based SDI framework for geospatial health big data analysis. *J. Ambient Intell. Humaniz. Comput.* **2019**, *10*, 551–567. [CrossRef]
6. Roberts, M.; Chapin, J. *What is Serverless?* O'Reilly Media Incorporated: Sebastopol, CA, USA, 2017.
7. Baldini, I.; Castro, P.; Chang, K.; Cheng, P.; Fink, S.; Ishakian, V.; Mitchell, N.; Muthusamy, V.; Rabbah, R.; Slominski, A.; et al. Serverless computing: Current trends and open problems. In *Research Advances in Cloud Computing*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 1–20.
8. Taibi, D.; El Ioini, N.; Pahl, C.; Niederkofler, J.R.S. Patterns for Serverless Functions (Function-as-a-Service): A Multivocal Literature Review. In Proceedings of the 10th International Conference on Cloud Computing and Services Science, CLOSER 2020, Prague, Czech Republic, 7–9 May 2020.
9. Hellerstein, J.M.; Faleiro, J.; Gonzalez, J.E.; Schleier-Smith, J.; Sreekanti, V.; Tumanov, A.; Wu, C. Serverless computing: One step forward, two steps back. *arXiv* **2018**, arXiv:1812.03651.
10. Shekhar, S.; Gunturi, V.; Evans, M.R.; Yang, K. Spatial big-data challenges intersecting mobility and cloud computing. In Proceedings of the Eleventh ACM International Workshop on Data Engineering for Wireless and Mobile Access, Scottsdale, AZ, USA, 20 May 2012; pp. 1–6.
11. Crespo-Cepeda, R.; Agapito, G.; Vazquez-Poletti, J.L.; Cannataro, M. Challenges and Opportunities of Amazon Serverless Lambda Services in Bioinformatics. In Proceedings of the 10th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics, Niagara Falls, NY, USA, 7–10 September 2019; pp. 663–668.
12. Niu, X.; Kumanov, D.; Hung, L.H.; Lloyd, W.; Yeung, K.Y. Leveraging serverless computing to improve performance for sequence comparison. In Proceedings of the 10th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics, Niagara Falls, NY, USA, 7–10 September 2019; pp. 683–687.
13. Kim, Y.; Lin, J. Serverless data analytics with flint. In Proceedings of the 2018 IEEE 11th International Conference on Cloud Computing (CLOUD), San Francisco, CA, USA, 2–7 July 2018; pp. 451–455.
14. Ishakian, V.; Muthusamy, V.; Slominski, A. Serving deep learning models in a serverless platform. In Proceedings of the 2018 IEEE International Conference on Cloud Engineering (IC2E), Orlando, FL, USA, 17–20 April 2018; pp. 257–262.
15. Anand, S.; Johnson, A.; Mathikshara, P.; Karthik, R. Real-time GPS tracking using serverless architecture and ARM processor. In Proceedings of the 2019 11th International Conference on Communication Systems & Networks (COMSNETS), Bangalore, India, 7–11 January 2019; pp. 541–543.
16. Malawski, M.; Gajek, A.; Zima, A.; Balis, B.; Figiela, K. Serverless execution of scientific workflows: Experiments with hyperflow, AWS lambda and Google cloud functions. *Future Gener. Comput. Syst.* **2017**. [CrossRef]

17. Varghese, B.; Buyya, R. Next generation cloud computing: New trends and research directions. *Future Gener. Comput. Syst.* **2018**, *79*, 849–861. [CrossRef]
18. Lee, H.; Satyam, K.; Fox, G. Evaluation of production serverless computing environments. In Proceedings of the 2018 IEEE 11th International Conference on Cloud Computing (CLOUD), San Francisco, CA, USA, 2–7 July 2018; pp. 442–450.
19. Goodchild, M.F.; Fu, P.; Rich, P. Sharing geographic information: an assessment of the Geospatial One-Stop. *Ann. Assoc. Am. Geogr.* **2007**, *97*, 250–266. [CrossRef]
20. Shashi, S. *Spatial Databases*; Pearson Education: Bengaluru, India, 2007.
21. Yang, C.; Li, W.; Xie, J.; Zhou, B. Distributed geospatial information processing: Sharing distributed geospatial resources to support Digital Earth. *Int. J. Digit. Earth* **2008**, *1*, 259–278. [CrossRef]
22. Escamilla-Ambrosio, P.; Rodríguez-Mota, A.; Aguirre-Anaya, E.; Acosta-Bermejo, R.; Salinas-Rosales, M. Distributing Computing in the internet of things: Cloud, fog and edge computing overview. In *NEO 2016*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 87–115.
23. Barik, R.K.; Dubey, H.; Mankodiya, K. SOA-FOG: Secure service-oriented edge computing architecture for smart health big data analytics. In Proceedings of the 2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP), Montreal, QC, Canada, 14–16 November 2017; pp. 477–481.
24. Higashino, T. Edge computing for cooperative real-time controls using geospatial big data. In *Smart Sensors and Systems*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 441–466.
25. Cao, X.; Madria, S. Efficient Geospatial Data Collection in IoT Networks for Mobile Edge Computing. In Proceedings of the 2019 IEEE 18th International Symposium on Network Computing and Applications (NCA), Cambridge, MA, USA, 26–28 September 2019; pp. 1–10.
26. Simform. AWS Lambda vs. Azure Functions vs. Google Cloud Functions: Comparing Serverless Providers. 2018. Available online: www.simform.com/aws-lambda-vs-azure-functions-vs-google-functions (accessed on 1 May 2020).
27. Wang, Y.; Liu, Z.; Liao, H.; Li, C. Improving the performance of GIS polygon overlay computation with MapReduce for spatial big data processing. *Clust. Comput.* **2015**, *18*, 507–516. [CrossRef]
28. Gatrell, A.C.; Bailey, T.C.; Diggle, P.J.; Rowlingson, B.S. Spatial point pattern analysis and its application in geographical epidemiology. *Trans. Inst. Br. Geogr.* **1996**, *21*, 256–274. [CrossRef]
29. Jiang, Z.; Shekhar, S. Spatial and spatiotemporal big data science. In *Spatial Big Data Science*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 15–44.
30. Shekhar, S.; Evans, M.R.; Kang, J.M.; Mohan, P. Identifying patterns in spatial information: A survey of methods. In *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*; Wiley: Hoboken, NJ, USA, 2011; Volume 1, pp. 193–214.
31. Zhou, C.; Frankowski, D.; Ludford, P.; Shekhar, S.; Terveen, L. Discovering personally meaningful places: An interactive clustering approach. *ACM Trans. Inf. Syst. (TOIS)* **2007**, *25*, 12-es. [CrossRef]
32. Cugler, D.C.; Oliver, D.; Evans, M.R.; Shekhar, S.; Medeiros, C.B. Spatial Big Data: Platforms, Analytics, and Science. Available online: https://pdfs.semanticscholar.org/c64e/b7f733cf78573e962c6b5df24860eed3aabe.pdf (accessed on 1 May 2020).
33. Vatsavai, R.R.; Ganguly, A.; Chandola, V.; Stefanidis, A.; Klasky, S.; Shekhar, S. Spatiotemporal data mining in the era of big spatial data: Algorithms and applications. In Proceedings of the 1st ACM SIGSPATIAL International Workshop on Analytics for Big Geospatial Data, Redondo Beach, CA, USA, 6 November 2012; pp. 1–10.
34. Evans, M.R.; Oliver, D.; Yang, K.; Zhou, X.; Ali, R.Y.; Shekhar, S. Enabling spatial big data via CyberGIS: Challenges and opportunities. In *CyberGIS for Geospatial Discovery and Innovation*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 143–170.
35. Barik, R.K.; Dubey, H.; Samaddar, A.B.; Gupta, R.D.; Ray, P.K. FogGIS: Fog Computing for geospatial big data analytics. In Proceedings of the 2016 IEEE Uttar Pradesh Section International Conference on Electrical, Computer and Electronics Engineering (UPCON), Varanasi, India, 9–11 December 2016; pp. 613–618.
36. Barik, R.K.; Tripathi, A.; Dubey, H.; Lenka, R.K.; Pratik, T.; Sharma, S.; Mankodiya, K.; Kumar, V.; Das, H. Mistgis: Optimizing geospatial data analysis using mist computing. In *Progress in Computing, Analytics and Networking*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 733–742.

37. Blower, J.D. GIS in the cloud: Implementing a Web Map Service on Google App Engine. In Proceedings of the 1st International Conference and Exhibition on Computing for Geospatial Research & Application, Washington, DC, USA, 21–23 June 2010; p. 34.
38. Apostu, A.; Puican, F.; Ularu, G.; Suciu, G.; Todoran, G. Study on advantages and disadvantages of Cloud Computing—The advantages of Telemetry Applications in the Cloud. *Recent Adv. Appl. Comput. Sci. Digit. Serv.* **2013**, *2103*, 118–123.
39. Yang, C.; Huang, Q. *Spatial cloud computing: A practical approach*; CRC Press: Boca Raton, FL, USA, 2013.
40. Sykora, P.; Schnabel, O.; Enescu, I.I.; Hurni, L. Extended cartographic interfaces for open distributed processing. *Cartogr. Int. J. Geogr. Inf. Geovis.* **2007**, *42*, 209–218. [CrossRef]
41. Kazemitabar, S.J.; Banaei-Kashani, F.; McLeod, D. Geostreaming in cloud. In Proceedings of the 2nd ACM SIGSPATIAL International Workshop on GeoStreaming, Chicago, IL, USA, 1 November 2011; pp. 3–9.
42. PyPI. A Cartographic Python Library with Matplotlib Support for Visualisation. 2018. Available online: pypi.org/project/Cartopy (accessed on 1 May 2020).
43. PyPI. Geometric Objects, Predicates, and Operations. 2018. Available online: pypi.org/project/Shapely (accessed on 1 May 2020).
44. PyPI. Fiona Reads and Writes Spatial Data Files. 2019. Available online: pypi.org/project/Fiona (accessed on 1 May 2020).
45. PyPI. Python Interface to PROJ (Cartographic Projections and Coordinate Transformations Library). 2019. Available online: pypi.org/project/pyproj (accessed on 1 May 2020).
46. PyPI. R-Tree Spatial Index for Python GIS. 2016. Available online: pypi.org/project/Rtree (accessed on 1 May 2020).
47. PyPI. Use Geometric Objects as Matplotlib Paths and Patches. 2017. Available online: pypi.org/project/descartes (accessed on 1 May 2020).
48. PyPI. Fast and Direct Raster I/O for Use with Numpy and SciPy. 2019. Available online: pypi.org/project/rasterio (accessed on 1 May 2020).
49. McLemore, V.T. Mineral-Resource Potential of Sabinoso Wilderness Area and Rio Grande Del Norte National Monument in Northeastern New Mexico. 2018. Available online: https://geoinfo.nmt.edu/publications/openfile/details.cfml?Volume=599 (accessed on 1 May 2020).
50. Zhou, W.; Chen, G.; Li, H.; Luo, H.; Huang, S.L. GIS application in mineral resource analysis—A case study of offshore marine placer gold at Nome, Alaska. *Comput. Geosci.* **2007**, *33*, 773–788. [CrossRef]
51. Ester, M.; Kriegel, H.P.; Sander, J.; Xu, X. A density-based algorithm for discovering clusters in large spatial databases with noise. *Kdd* **1996**, *96*, 226–231.
52. Bryant, A.; Cios, K. RNN-DBSCAN: A density-based clustering algorithm using reverse nearest neighbor density estimates. *IEEE Trans. Knowl. Data Eng.* **2018**, *30*, 1109–1121. [CrossRef]
53. Zeng, Y.; Zhang, H.; Fang, Z.; Liu, X.; Crociani, L.; Vizzari, G. Lane-formation in counter-flow based on DBSCAN. In Proceedings of the 4th ACM SIGSPATIAL International Workshop on Safety and Resilience, Seattle, WA, USA, 6 November 2018; pp. 1–6.
54. GIS, F.C. Forecast Households. 2017. Available online: https://catalog.data.gov/dataset/forecast-households-371f2 (accessed on 1 May 2020).