Research Article

Microservice-Based Platform for Space Situational Awareness Data Analytics

Wanjie Lu , Qing Xu, Chaozhen Lan , Liang Lyu, Yang Zhou, Qunshan Shi, and Yinghao Zhao

Information Engineering University, Zhengzhou 450052, China

Correspondence should be addressed to Chaozhen Lan; lan_chaozhen@163.com

Received 23 May 2019; Accepted 11 December 2019; Published 6 January 2020

Academic Editor: Franco Bernelli-Zazzera

Copyright © 2020 Wanjie Lu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The development, deployment, and maintenance of the current space situational awareness (SSA) information system have become increasingly complex. However, researchers cannot flexibly and conveniently apply the research results to practical applications due to the lack of basic research platforms for SSA. Inspired by X as a Service (XaaS), we propose the microservice-based platform for SSA data analytics to provide a scaffold-like platform for researchers. Based on microservice, the architecture for this platform is proposed to meet the requirements of flexible development and loosely coupled deployment. To facilitate the use of the platform, the hybrid data service layer is established to provide basic data for research and the functional service layer is designed to provide services for clients and applications. Due to the massive data processing requirements, the data analysis architecture and processing model, which can easily integrate various user-defined algorithms and significantly improve the computational efficiency, are proposed based on the Lambda architecture. To verify the platform's effectiveness, two cases are established and implemented. The results show that this platform can provide a convenient, flexible, and efficient platform for the requirements of algorithm integration, experiment, and data display from users and researchers.

1. Introduction

The importance of the space environment around the earth has increased. More than 80 countries and organizations have payloads in space [1]. Countries, organizations, and individuals worldwide increasingly depend on space with respect to positioning, navigation, communication, etc. Aerospace powers are vigorously developing space situational awareness (SSA) capabilities to maintain security and dominant position and to detect any changes or threats in space. The SSA represents the current and predictive knowledge of space events, activities, conditions, and space application status capabilities and constraints [2], which has been used for satellite tracking, communication management, orbit operation and maintenance, and threat assessment [3]. It is concerned with various space objects such as satellites, debris, and rockets. Various countries have developed and deployed a series of SSA-related systems for the acquisition, analysis, mining, and application of SSA information and data [4], which are often referred to as the SSA information system

(SSAIS). These systems have made tremendous contributions to SSA, although many challenges still remain as time and technology advances.

As countries around the world launch more spacecraft, there is growing complexity and diversity of SSA-related information and data, which have made development, deployment, and maintenance of the SSAIS more complex. Initially, SSAISs were implemented based on the monolithic architecture with tightly coupled components. The addition or modification of internal functions required consideration of their compatibility with others, which resulted in larger maintenance and development costs, longer continuous delivery cycles, and worse expansion [4, 5]. Due to the underutilization of distributed computing and network-based service architecture [6], these systems could not be effectively deployed on the cloud [7]. To address these challenges, cloud computing, big data, and service-oriented architecture (SOA) are applied to the SSAIS [5] and the SSAISs are deployed on the cloud based on various technologies such as virtualization [4]. The European Space Agency

(ESA) has established the SSA preparatory system based on SOA [8], and the Defense Advanced Research Projects Agency (DARPA) has established the OrbitOutlook project by using SOA to integrate datasets from multiple countries, organizations, and enterprises [9, 10]. To obtain an efficient and flexible SSAIS by 2020 [11], the United States is building the Joint Space Operations Center Mission System (JMS), a worldwide state-of-the-art SSAIS, based on cloud computing and SOA [12]. Almost all these systems use SOA to transform the monolithic architecture and realize a distributed deployment. However, SOA requires more manpower, material, and financial resources, and the system developed based on SOA is difficult to maintain and develop. In the era of cloud computing and big data, user requests for services are variable and resilient [13, 14]. Therefore, services need to be easily maintained and to have continuous delivery capabilities [15]. However, such properties cannot be met by the monolithic architecture and SOA.

The relationship between space and the public is growing closer. Nevertheless, there is still inconvenient and insufficient access to SSA information and knowledge. The concept of SSA has been applied by the military to space activities, and the existing SSAISs are mainly dedicated to the military and national security. However, space activities (e.g., navigation, communication, and weather) are closely related to civil life, and the public has begun to pay attention to SSA [16]. The shortage of channels for obtaining SSA information leads to the lack of access and opportunities for the public to understand outer space [3]; even a vast majority of satellite operators cannot accurately obtain information about the space environment surrounding their satellites [17]. Based on these issues, new ideas and services have emerged. Concepts of building a civil space traffic management system [18] and a publicly available global SSA information service based on open data sources have been proposed [17]. In addition to these concepts, several website portals providing services have been established. Space-Track [19] and Heavens-Above [20] are some of the portals that provide such datasets as satellite ephemeris data and SSA-related information, such as data analysis and visualization of the satellites position; however, the data distribution needs to be authorized, which limits the application of the datasets [21]. CelesTrak [22] and Gunter's Space Page [23] provide sufficient datasets without authorization restrictions and visualization [21]. SpaceBook [24] and N2YO [25] provide visualization schemes and application programming interfaces (APIs) to access object satellite ephemeris data and properties but do not provide analysis services. These projects are in part capable of providing SSA information to the public, but they do not provide comprehensive services for the public to improve SSA cognition, particularly the capabilities of helping the research institutions with academic studies and engineering applications.

In response to the above-mentioned problems and adhering to the idea of X as a Service (XaaS), we propose a microservice-based platform for SSA data analytics to promote the sharing of SSA information and provide convenient and efficient services to researchers who need to build and deploy the related algorithms and applications.

The main contributions of this paper can be summarized as follows:

- (1) Based on microservice architecture (MSA), a platform for SSA data analytics is established and the basic SSA services are constructed, including space object service, data statistic service, database management service, and batch and stream computing services. These basic services will provide support for the building of various user services
- (2) The platform proposed builds regularly updated databases based on different data sources, including the various relevant catalog object information, which can provide basic data support for various services in the platform
- (3) Aiming at the different computing requirements from spatiotemporal analysis services, the data analysis model combining batch processing and stream processing is designed based on the Lambda architecture. User-defined algorithms can be applied in this model to process SSA data, and the analysis results will be stored in the databases or the cache service for different requirements
- (4) As a scaffold-like platform, two cases of SSA analysis service are constructed and the applicability of the platform is tested, which prove that the platform can well adapt to various algorithms, and the final analysis results can be flexibly presented in different visualization schemes

The remainder of this paper is organized as follows. For the construction of this platform, several state-of-the-art technologies related to our work are introduced in Section 2. Section 3 elaborates the platform architecture based on MSA. In the platform, the data service layer and functional service layer which play important roles in the platform are discussed in detail. For the data service layer, the back-end data model is designed and constructed; for the functional service layer, the dependency and deployment approaches are designed and described. In order to meet the massive computation and analysis requirements of SSA effectively, the general data analysis architecture based on the Lambda architecture and the data processing model are designed in Section 4, and two case analyses are illustrated in Section 5 to test and verify the platform. The conclusions and outlines of our future work for the platform are drawn in Section 6.

2. Related State-of-the-Art Technologies

The state-of-the-art IT concepts and technologies needed to implement the platform are described in this section.

2.1. Microservice Architecture and Container Technology. As a new architectural style that emerged in recent years, there has been rising interest on MSA [26]. MSA can split a complex software system into single-function and small-grained services [27–29]. Each service can be independently developed, tested, and deployed. The services use

TABLE 1: Comparison between MSA and SOA.

Type	MSA	SOA
Level	Team level, bottom-up implementation	Enterprise level, up-bottom implementation
Granularity	A system is split into many fine-grained services	The service consists of multiple subsystems with large granularity
Structure	Without ESB; loose service structure	Based on ESB; centralized service structure
Integration	Simple (HTTP/REST/JSON)	Complex (ESB/WS/SOAP)
Deployment	Independent	Dependent and complex

FIGURE 1: MSA deployment method. The multiple instances wherein each microservice can be deployed on different web servers (e.g., the instances service X can be deployed in Web server 1,..., Web server n , separately).

a lightweight communication mechanism to exchange data [30]. MSA allows the team to build services by using appropriate program languages and tools according to the business context, which has advantage over the monolithic architecture [31].

Although MSA and SOA exhibit some commonalities with respect to the definition and design, they also significantly differ, as shown in Table 1 [32]. One significant difference is that the enterprise service bus (ESB) in SOA requires complex configurations, which leads to a higher latency and single failure point [7, 33]. In contrast, the highly cohesive and low-coupling MSA does not emphasize the ESB, which leads to more flexibility, enforceability, and scalability [34, 35]. Another key difference is that most services in SOA are derived from monolithic architecture applications and functions, which cannot be changed, scaled, or deployed independently [31].

The MSA deployment method is shown in Figure 1. The request from the external terminal is forwarded to different microservices through the service gateway, and the results of different microservices are received and transmitted to the terminal. Each microservice meets highly concurrent computing requirements by building multiple instances.

Load balancing can be used to reasonably allocate requests to make a high response speed and the overall performance certain. Different microservices can be developed using different programming languages (e.g., C++, Java, and Python) and be able to access data services as needed. This architecture and its deployment approach meet the flexible needs of users.

Because of the advantages of container technology over traditional virtualization technologies, container technology has been widely applied to software development and deployment on the cloud [36, 37]. When using traditional virtualization technologies, every service must be applied with all dependencies and a complete isolation operation system (OS) in terms of having its own process and network, which leads to the waste of the resource, space, and hardware performance. The container technology differs from traditional virtualization technologies in that services can be built in a shared host OS that only includes the applications and dependencies. Only the necessary processes need to run when services start, which means that the containers consume only the resources required for the services [38, 39]. Once a container is built, it can be deployed and run without need for modification on any host OS. These advantages make the container technology very suitable for the deployment of

microservice-based applications and allow researchers to put most of their time and effort into the innovation of algorithms and applications.

2.2. Batch and Stream Computing Platforms. Many excellent platforms and frameworks, such as Apache Hadoop [40], Apache Spark [41], Apache Storm [42], and Apache Flink [43], are available for the computation of big data. Each framework is applied in all walks of life for different needs. Apache Hadoop has been used in various big data processing fields but cannot meet the real-time computing tasks and requirements [44, 45]. Apache Storm only supports stream processing [46], Apache Spark simulates stream processing based on batch processing [47], and Apache Flink is entirely based on stream processing and simulates batch processing through stream processing [48]. Apache Flink can implement both stream processing and batch processing via a single solution, which can help prevent duplication of codes during development. The characteristics and excellent performance of Apache Flink can better meet the needs of researchers [49, 50].

2.3. SQL and NoSQL Database. Nowadays, a variety of online service applications need the appropriate databases to meet the requirements from hundreds of users at any time [51]. Simultaneously, the advances of technologies generate a huge volume of different kinds of data, which can be divided into structured data, semistructured data, and unstructured data. Relational database management system (RDBMS), or structured query language (SQL) databases, can store structured data well, but it has great limitations with massive unstructured data (e.g., images and files). Therefore, not only SQL (NoSQL) databases, which are suited for storing semistructured and unstructured data, are proposed and have achieved widespread and successful applications in the internet domain [52].

According to the storage mode, NoSQL databases can be divided into a document database (e.g., MongoDB), column-oriented database (e.g., HBase), key-value database (e.g., Redis), etc. [53]. In terms of unstructured data, NoSQL databases have a huge advantage over RDBMS [51, 54]. The NoSQL databases are well equipped to cope with the need of horizontal expansion and are mainly used in systems with intensive reading and writing requirements and systems with uncertain data structure. However, RDBMS occupies a dominant position in the systems with high requirements for data consistency and vertical scalability. For these reasons, in order to build a storage system for various data structures, the current mainstream solution is to use the RDBMS and NoSQL database side by side, which is called the hybrid SQL/NoSQL database [55–57].

2.4. 3D Visualization Platform. The Digital Earth system can effectively fuse data from different sources in a network environment [58], which has become the standard method for analyzing and displaying spatial information on different spatiotemporal scales [59]. Based on the comparison of different digital earth platforms [60], Cesium is

an appropriate solution for the visualization of SSA information and scenarios.

Cesium [61] is an open-source JavaScript library for world-class three-dimensional (3D) globes and maps based on WebGL and has become the leading 3D WebGIS platform [62, 63]. Cesium is capable of merging image layers from different data sources, loading various industrial-grade 3D model formats (e.g., KML, glTF, GeoJSON, and TopoJSON), and supporting open geospatial consortium (OGC) standards. More importantly, CZML [64], a markup language for Cesium, has been designed and developed.

CZML is a JavaScript object notation (JSON) format for the description of a time-dynamic graphical scene. It is an effective means for the network-based, real-time, data-driven analysis of dynamic objects in a 3D digital earth [62, 63]. In addition to various default properties, CZML also supports custom properties. CZML can meet all kinds of data processing needs; especially for stream computation, CZML is ideally suited for incremental streaming to Cesium for the visualization because of its unique ability to stream massive datasets and the storage of property information of different objects in multiple packets. A CZML packet is a collection of name-value pairs, which describe the geometrical, graphical, semantic, and temporal aspects of a single geospatial object [62]. The data volume in a packet depends on the needs; therefore, CZML guarantees the timeliness of each computation and transmission by controlling the data volume. Figure 2 shows the CZML data format for the scenario of the satellite coverage analysis, which includes ground area, satellite, and sensor packets.

3. Platform Framework

3.1. Platform Architecture. The architecture of the platform based on MSA is shown in Figure 3. The client (e.g., *desktop*, *browser*, and *mobile*) requests services through the APIs in representational state transfer (RESTful) architectural style [65]. The clients can use a series of display methods such as the three-dimensional (3D) visualization, for processing the results. The *service gateway* provides a unified RESTful API access interface, and all requests from the clients need to go through the *service gateway*, which avoids the exposure of internal APIs between microservices. Users do not need concern for the interaction between internal services, and the cases of service upgrading, modification, and expansion will not affect user experiences. *Load balancing* is responsible for assigning requests, preventing excessive pressure on background services, and improving the response speed and overall performance. When services add new instances, *load balancing* can reasonably allocate requests to the newly added instances. All microservices are registered in the *service registry*, and each microservice finds the others through the *service registry*. *Service configuration* stores the attribute configurations of all microservices.

In addition to the above components, the two most important parts of the platform are the *functional service layer* and *data service layer*. The *functional service layer* provides *basic services* that are currently available on the platform and *user services*; the *data service layer* provides

FIGURE 2: CZML data format for the satellite coverage analysis scenario. The CZML file can be constructed by the program development, and Cesium will display the scenarios by loading these files.

the *databases* and *cache service*. The expatiation of both layers will be provided separately in Sections 3.2 and 3.3.

The efficient operation of the platform requires a good deployment strategy. As a representative enterprise container platform for high-velocity innovation, Docker [66] can build, host, and run distributed microservices ideally. It has an advantage over traditional virtualization deployment particularly with respect to data computation-intensive applications [67]. Figure 4 shows how the functional services are deployed in the application server based on Docker. Each application server includes many containers, and each container runs a separate functional service. In each application server, the microservice in different containers exchanges messages and combines functions through a lightweight communication mechanism. *Load balancing* can distribute requests between different application servers to avoid excessive pressures on one of the servers. All services that need to request data can access the database in the *data service layer* directly or indirectly.

Generally, container technology is well suited for the distribution and deployment of application services, but the situation becomes different and complicated when databases are deployed using the container technology. Regarding the persistent data in the production database, deploying data by using the container technology can easily cause loss of data when the container fails. However, when using redis to provide the cache service, there will be no problem in deploying the cache service in the container because cache data is only stored temporarily. Based on the above analyses, in terms of the deployment of the *data service layer* in Figure 3, *databases* use the stand-alone data server for

deployment instead of containers, while the *cache service* is deployed using the container technology.

3.2. Data Service Layer. The core of the SSAIS is the database of trackable objects and relevant information [68], which needs to be continuously updated and maintained, and the space catalog object database needs to be constructed for object retrieval and data analyses, such as conjunction assessment [69]. Based on the architecture in Figure 3, the platform needs the data service layer to provide effective services for users and operators. This section will describe the construction of the data service layer in detail.

3.2.1. Acquisition of Data Sources. Current space object data of the platform include the basic information, attribute parameters, ephemeris data, photometric characteristic data, object images, and other related data, which can be obtained by integrating multiple data sources. Open datasets, which are most relevant and significant for integration, are as follows:

- (i) *CelesTrak* [70]: a website providing datasets that include current and historical orbital data, detailed interpretation of satellite ephemeris data, satellite orbital conjunction reports, and assessments of threatening encounters in space. It also provides an online query for satellite ephemeris data and a data download function for the data in the form of text documents based on different needs (e.g., launches of the last 30 days, 100 (or so) brightest).

FIGURE 3: Platform architecture. *Basic services* and *data service layer* are provided by the platform; *user services* can be developed by users with a variety of algorithms and components. *Databases* include the RDBMS and NoSQL databases.

(ii) *The Union of Concerned Scientists (UCS) satellite database* [71]: a public dataset including important information about current on-orbit satellites, such as the name, country, purpose, orbit type, orbital parameter, launch weight, and dry weight, to assist both professional and nonprofessional users in conducting research and analysis. This database is updated every few months and is less time-efficient, but it can complement other databases due to the inclusion of valid significant information about many space objects. The latest data used in this paper was updated on November 30, 2018

(iii) *Planet4589* [72]: a website maintained by Jonathan McDowell from the Harvard-Smithsonian Center for Astrophysics. It provides and updates satellite launch details, satellite lists, and geosynchronous satellite lists every couple of months

In addition to the above data sources, we obtained images, models, and characteristic data (e.g., photometric characteristic data) of the space objects through the Internet and relevant software (e.g., systems tool kit (STK) [73]), which are also used as the data sources for the platform.

FIGURE 4: Service deployment. Each container is an isolated environment. Different services are deployed separately and communicate with each other to complete complex tasks.

3.2.2. Data Model. According to the platform architecture and the core design ideology of hybrid SQL/NoSQL databases, we use the storage mode in the platform as follows: the RDBMS (MySQL) is used to store structured data, mongoDB is used to store unstructured and semistructured data, and redis is used to store cached data. The schematic database diagram of this platform is shown in Figure 5.

When using the RDBMS to store the structured data, the most important table is the *satellite_list* table. The data of the table *satellite_list* that are derived from Celestrak include all orbital and decayed space objects (e.g., payloads, debris, and rockets) and relevant information such as the launch site code, owner, purpose, and operation parameters. The most pivotal field is the North American Aerospace Defense Command (NORAD) catalog number, which is a sequential 5-digit number assigned to all earth-orbiting objects in the order of identification. The other tables of the structured data are as follows:

- (i) The table *launch_site_list* includes the full name, geographic coordinates, and other information pertaining to worldwide launch sites. It is associated with each space object in the table *satellite_list* via the launch site code
- (ii) The table *purpose_list* includes codes and detailed descriptions of the satellite purpose. It is associated with the table *satellite_list* via the purpose code
- (iii) The table *owner_list* includes countries, regions, and organizations that had and have space objects. This table is associated with the table *satellite_list* via the country code

- (iv) The table *orbital_decayed_boxscore* includes the orbital and decayed counts of payloads, debris, and rockets of each country or organization and is associated with the table *owner_list* via the country code
- (v) The table *ucs_satellite_list* includes the datasets from the UCS Satellite Database and is associated with the table *satellite_list* via the NORAD number

In this platform, based on the excellent performance of mongoDB in storing unstructured data, the unstructured data of SSA are stored with mongoDB, and two examples (*photometric characteristic data* and *position data*) are presented in Figure 5. In the characteristic data of space objects, the photometric characteristic data includes imaging parameters, photometric values, and optical images. mongoDB is ideal for storing these kinds of data. For the position data of space objects within a time period, the position and time are always on a one-to-one correspondence. As shown in Figure 5, the calculation parameters and position data can be combined into data in the JSON format. mongoDB stores data in the BSON (Binary JSON) format, which means that mongoDB is well suited for storing this data.

The SSA-related information analyses have various intermediate data that will be used by different components in the execution process and analysis results of batch and stream processing that will be used by the clients, which means the cache service is indispensable. The cache service stores the cached data (e.g., analysis results and temporary data generated in data processes) temporarily within a certain period of time to improve the data access speed and reduce the request pressure on the database service. When the results requested can be obtained through the cache service, the databases will

FIGURE 5: Data model. In MySQL, each data table is associated with the others via a certain keyword. redis, which provides the cache service, stores the processing results and the data retrieved from MySQL and mongoDB that needs to be used multiple times temporarily.

no longer be accessed; otherwise, the data will be queried and processed from the databases and be stored or updated by the cache service simultaneously. As a high-performance cache database, redis can effectively store these intermediate and result data and ensure that users and various components can access these data efficiently.

3.3. Functional Service Layer. MSA allows platforms and systems to perform complex functions through the orchestration of different microservices. Simultaneously, the existing

microservices can be well integrated with other services. More importantly, MSA can solve the integration problems of heterogeneous services, so different services can be developed by using the most appropriate program language. By packaging different data analysis services, a unified interface can be provided externally by the service gateway, which can be accessed by different clients. Therefore, when building the functional service, the platform proposed in this article can split different functions and combine them to complete more complex tasks.

FIGURE 6: Service dependencies of the basic service. Examples of the applications that invoke the basic services are listed outside the basic service box.

The functional service in the platform includes basic services and user services. The basic services can provide functions and interfaces for user services; hence, they are the cornerstone and can reduce the difficulty of construction of user services. User services mainly concern various functions of SSA analysis, such as satellite coverage analysis and conjunction assessment. As a scaffold-like platform, this platform cannot provide full analytical functions. Therefore, only the basic services are covered in detail here, and the user services are illustrated using two cases in Section 5.

In MSA, each service is split according to the specific functions and the data exchange between them by using a lightweight communication mechanism. The basic services are necessary for the platform, and the current basic services of the platform and their dependencies on each other are shown in Figure 6. The detailed descriptions of each basic service are as follows:

- (1) The *data management service* manages all the data in the data service layer. This service can periodically obtain and integrate data from different sources and store them in suitable databases. It can also update the back-end databases according to the data submitted by users. This service will trigger certain functions to perform tasks when the data in databases is updated. For example, when the ephemeris data of space objects is updated, the position calculation of space objects based on the batch computing service will be called to calculate the position of updated objects at a fixed interval in a certain period of time. The results will be stored in the persistent database, and the other position-related services can avoid or

reduce the position calculation by using these results. This service is mainly called by the external data management user service, which requires the administrator authorization to log in

- (2) The *data retrieval service* constructs access to all the data in the data service layer. It can retrieve, organize, and return data based on the user's needs and requests from other related services. This service provides an easy-to-understand access interface to avoid complex data query statements, which is one of the most basic services in the platform
- (3) The *position calculation service* provides the position computation in various reference coordinate systems (e.g., J2000 coordinate system, True Equator and Mean Equinox (TEME) coordinate system) for the platform. The spatiotemporal characteristics of SSA objects make the position data into the basis of the data analysis of SSA objects. This service can be called by other services to compute position, and it accesses satellite ephemeris data through the *data retrieval service*
- (4) The *data statistic service* mainly analyzes the data of the entire back-end database and returns the results. The content of the data statistic includes the count of orbital objects (payloads, rockets, and debris), the distribution of orbital satellites according to various conditions (e.g., country, altitude, and application), the number of satellites launched in different years, and the different launch sites. All these statistics provide users with an intuitive understanding of the current space objects and the changes and trends of

the data from the past to the present. User services can use this data to build visual charts

- (5) The SSA data includes the structured data (e.g., basic information and two-line element (TLE)) and unstructured data (e.g., images, text, and media). Therefore, the *space object service* needs to retrieve both kinds of data based on request and integrates the obtained data to construct organized information that can be used by users directly
- (6) The *scenario construction service* packages the space object operating scenario or the result of SSA analysis into the data of the CZML format and visualizes scenarios by using the 3D visualization platform (e.g., Cesium), to meet the cognitive needs. Users can also add custom properties into the data for display in the scenario. The service relies on *space object service*, *data retrieval service*, and *position calculation service*, and it can provide scenario data for the visualization of various analysis scenarios, such as satellite coverage analysis, conjunction assessment, and more
- (7) The *stream computing service* can provide the timely computing service for high real-time requirements in SSA analysis, and the *batch computing service* is aimed at large-scale data computation scenarios that do not require real-time analysis. This paper builds the data analysis architecture and the processing model by integrating the stream and batch computing services, which can combine with the users' algorithms to realize the data analysis and display the results in different approaches. The data analysis architecture and processing model are illustrated in Section 4

4. Data Analysis Architecture and Processing Model

4.1. Data Analysis Architecture. As an important part of space operations, SSA requires efficient analyses of the complex space activity [74]. However, the spatiotemporal characteristic of space objects leads to large amounts of data and complex calculations in SSA data analysis. Therefore, there are needs for the batch computing to finish analyses that do not require real-time processing and for the stream computing to carry out the high real-time processing. For example, when the operational state of a certain space object changes (e.g. maneuver and attitude change), the actual impacts and consequences of the state change on relevant tasks need to be obtained and fed back to users in real time, which requires the stream processing. The spatiotemporal information of space objects is complicated, and enormous computing power is required to obtain the spatial relationship of space objects within a certain time period; this requirement can be met by employing batch processing. Therefore, it is necessary to construct a data analysis architecture that can satisfy the needs of batch and stream processing.

Based on Section 2.2, this paper builds the SSA data analysis architecture with the stream and batch processing based on the Flink and Lambda architectures, as shown in Figure 7. The Lambda architecture refers to a big data processing architecture that integrates batch and stream processing methods [75, 76]. In Figure 7, the clients include *management application* and user applications. *Management application* is used to maintain the data service of the platform. It can perform manual or automatic periodic operations, such as the update operation, on the data layer service. *Management application* interacts with the data management microservice in the box *microservices* through the API of *data management service* in the box *RESTful API*. The data that needs to be processed is passed to the microservices in the box *microservices*, and the analysis results are requested through the API provided by the API of *data retrieval service* in the box *RESTful API*.

In the analysis architecture, *batch computing* performs two kinds of roles. First, when the client initiates a data processing request without real-time requirements, it can be processed by *batch computing*. Next, the space objects (e.g., satellites and debris) display obvious spatiotemporal characteristics, and it is necessary to calculate the positions of the space object during the data analysis, such as conjunction assessment. Because SSA analyses are usually time-consuming, computing the positions of the space objects over a period of time and storing them in databases will effectively improve analytical efficiency. The data analyses of SSA also include real-time processing requirements, which can be met by *stream computing*. For example, when the maneuver of a space object occurs, users need to immediately perform a conjunction assessment to determine whether this maneuvered object will be a threat to the other objects. The batch and stream computing services receive the data from the client and process them by using user-defined algorithms.

When the position data of the objects exist in the *cache service*, the computing services can retrieve these data from the *cache service*, which prevents redundant calculations from occurring. After the completion of the computing process, the data that needs to be persisted can be stored in the *persistence databases*, and the analysis result required by the user is stored in the *cache service*. Finally, the clients obtain the data stored in the *persistence databases* and the analysis result in the *cache service* through the *data retrieval service*.

4.2. Data Processing Model. Due to the spatiotemporal characteristic of SSA, a general spatiotemporal-oriented data processing model for space objects is proposed in order to process the data in a better way. As shown in Figure 8, the execution flow of the processing model is as follows:

- (1) Getting the input dataset. The input dataset includes the time range $T(0)-T(n)$ and the calculation objects *targets*. Each *target* contains a set of objects that can complete the calculation, for example, in the

FIGURE 7: Data analysis architecture. In the box *microservices*, the data analysis and result storage are completed through the cooperation of different microservices.

FIGURE 8: Data processing model and execution flow. In this model, the format of the dataset generated in each step is indicated in the dotted box. The time range $T(0)-T(n)$ is divided equally into n parts, where $m = 0, 1, 2, \dots, n-1$. *Target x* is a subset of *targets* in the dataset.

- conjunction assessment, *Target x* includes one primary and one subordinate space object
- (2) According to the time range and a certain time interval (e.g., one day), the dataset from step (1) is divided into new datasets. Each new dataset contains a sub-time range $T(m)-T(m+1)$ and calculation object *targets*
 - (3) The *targets* of the datasets from step (2) are split into a separate object *Target x* to form new datasets. Each new dataset contains a sub-time range $T(m)-T(m+1)$ and calculation object *Target x*
 - (4) This step contains a series of processing maps, and each processing map contains a user-defined algorithm (e.g. A.1, A.2,..., A.n in Figure 8). The data from step (3) will be processed through a series of maps, and the results obtained include the unique identifier generated by *Target x* and the sub-time range $T(m)-T(m+1)$. Simultaneously, each result can be constructed by adding the other information and custom properties for special needs
 - (5) The analysis results from step (4) will be combined in this step according to the unique identifier in chronological order. The *result* finally formed is the

TABLE 2: The details of all nodes.

Nodes	Node IP	Operation system	Configuration
Master node	169.254.240.201	Ubuntu 16.04 LTS 64 bit	Memory: 8G Storage: 500G
Slave node 1	169.254.240.202	Ubuntu 16.04 LTS 64 bit	Memory: 8G Storage: 500G
Slave node 2	169.254.240.203	Ubuntu 16.04 LTS 64 bit	Memory: 8G Storage: 500G
Slave node 3	169.254.240.204	Ubuntu 16.04 LTS 64 bit	Memory: 8G Storage: 500G
Data server node	169.254.240.205	Ubuntu 16.04 LTS 64 bit	Memory: 8G Storage: 500G

analysis result of the object *Target x* in the time range $T(0)-T(n)$

- (6) The results obtained from step (5) are stored separately in the *persistence database* and *cache service* according to different requirements. The client and users obtain the results through the RESTful API

5. Case Studies and Implementation

This article uses two cases to test the proposed platform. The experimental system environment includes 5 nodes, and the details of all nodes are shown in Table 2. The service gateway, load balancing, service registry, and configuration services are deployed in the master node; functional services are deployed in the slave nodes; all databases and cache services of the service layer are deployed in the data server node.

5.1. Case I: Microservice-Based Satellite Coverage Analysis. As an important application efficiency indicator of earth observation satellites, the satellite coverage capability is of great significance for the mission planning, constellation design, and acquisition of data [77]. Common algorithms mainly include analytical methods [78], grid point methods [79, 80], and geometric operation methods [81]. The analytical method is only applied to single satellites and regions. The calculation efficiency and accuracy of the grid point method are greatly affected by the grid size. These algorithms are computationally intensive and time-consuming. In view of these problems, we propose a satellite coverage analysis algorithm based on the extended minimum bounding rectangle (EMBR) of the ground area, called EMBR-SCA, to verify and test the platform proposed in this paper.

5.1.1. Algorithm Principle and Flow. The calculation of the satellite coverage area is complex, which is one of the main factors affecting the computational efficiency. Within one satellite orbital period, the satellite coverage area only has a limited time to join with the ground area. Preventing the calculation of the satellite coverage area in the disjoint time period will improve the efficiency.

The points, lines, and polygons used for the judgment of spatial relationships are converted to suit the plane coordinate system. Simultaneously, the rectangular centrum sensor

model is taken as an example, and other sensor models can be analyzed based on this method. Figure 9 shows that polygon *A* is the satellite coverage area, point *P* represents the intersection of the satellite sensor centerline with the earth ellipsoid, the dotted line represents the motion track of point *P*, polygon *B* is the ground area, polygon *C* is the minimum bounding rectangle (MBR) of *B*, and polygon *D* is the EMBR of *B*. The EMBR is obtained by extending the MBR of the ground area by a certain angle in the southeastern and north-western directions, which will be described in Section 5.1.2. The algorithm principle is as follows: within the satellite orbital period, the maximum central angle of the satellite coverage area on the earth ellipsoid will not exceed a certain value $\alpha_{\max}$. Polygon *D* is then built by using $\alpha_{\max}$ as the certain angle. If polygon *D* and point *P* are disjointed, polygons *A* and *B* must be disjointed. In that case, there is no need to calculate polygon *A*, and this is implemented to avoid an excessive calculation of the satellite coverage area. The algorithm flow is as follows: first, angle $\alpha_{\max}$ is calculated during the satellite orbital period. Angle $\alpha_{\max}$ only needs to be calculated once during the entire satellite orbital period. Angle $\alpha_{\max}$ is used to extend polygon *C* to obtain polygon *D*. Figure 9(a) shows that polygons *A* and *B* must be disjointed if polygon *D* and point *P* are disjointed. In this case, there is no need to calculate polygon *A*. Figures 9(b)–9(e) show that polygon *A* and its spatial relationship with polygon *B* must be calculated if polygon *D* contains point *P*. Figure 9(f) shows that polygons *A* and *B* are disjointed if polygon *D* does not contain point *P*. In this case, point *P* must be calculated without calculating polygon *A*, and it must be determined if polygon *D* contains point *P*. The algorithm only calculates polygon *A* during a certain part of the satellite orbital period, which reduces the calculation and increases the efficiency. The algorithm records the time period and coordinates of the satellite coverage area intersecting with or containing the ground area; these data can be used to obtain other satellite coverage parameters such as the coverage ratio.

5.1.2. Establishment of the EMBR of the Ground Area. Attitude maneuvering of the satellite ontology will lead to a complex calculation of $\alpha_{\max}$. To balance simplicity and accuracy, a special method is proposed.

FIGURE 9: Schematic diagram of the algorithm principle. (a) Polygon D does not contain point P ; (b) polygon D contains point P , polygon B disjoint with polygon A ; (c) polygon B intersects with polygon A ; (d) polygon B intersects with polygon A ; (e) polygon D contains point P , polygon B disjoint with polygon A ; (f) polygon D does not contain point P .

The cut-away view along the meridian in Figure 10 shows that points E and F are the intersections of the ground coverage of the satellite S and earth ellipsoid at time t , respectively; the corresponding central angle of the arc $\widehat{EF}$ is $\angle EOF$. If the field of view $\theta = \angle ESF$ is fixed, $\angle EOF$ reaches the maximum when the boundary ray SE of the satellite sensor is tangential to the earth ellipsoid. The calculation of angle $\angle EOF$ is not simple; therefore, line OE' is created, with $OE' \perp SE$. Because $\angle EOF$ is smaller than $\angle E'OF$ and the calculation of $\angle E'OF$ is simple, we obtain $\alpha_{\max}$ by calculating $\angle E'OF$. The angle $\angle E'OF$ can be calculated in

$$\begin{aligned} \angle E'OF = & \arccos(\mu) - \arcsin\left[\frac{OS}{OF} \sin(\arcsin(\mu) - \theta)\right] \\ & + \arcsin(\mu) - \theta, \end{aligned} \quad (1)$$

where $\mu = OE'/OS$. The partial derivative of Equation (1) with respect to parameters OE' , OS , OF , and θ shows that $\angle E'OF$ decreases with OE' and increases with OS , OF , and θ . The following relationship exists:

$$\begin{cases} OE' > E_{\text{major}}, \\ OF \leq E_{\text{minor}}, \\ OS \leq O_{\text{apogee}}, \end{cases} \quad (2)$$

FIGURE 10: Sectional view of the area coverage of the satellite.

where E_{major} is the semimajor axis of the earth ellipsoid and $E_{\text{major}} = 6378137.0$ m, E_{minor} is the semimajor axis of the earth ellipsoid and $E_{\text{minor}} = 6356752.3142$ m, O_{apogee} is the distance between the satellite apogee and the center of the earth, and $OE' > E_{\text{major}}$, $OF \leq E_{\text{minor}}$, and $OS \leq O_{\text{apogee}}$. Taking the rectangular centrum sensor model as an example, Figure 11 shows that the horizontal half angle $\theta_H = \angle M_4SP$ and the vertical half angle $\theta_V = \angle N_2SP$. The maximum θ can be calculated in

$$\theta_{\max} = \angle M_1SN_3 = \arctan\left(\sqrt{\tan^2(\theta_H) + \tan^2(\theta_V)}\right). \quad (3)$$

Therefore, the angle $\alpha_{\max}$ can be calculated by using Equation (4), and the EMBR of the ground area can then

FIGURE 11: Rectangular cone model.

be achieved by using $\alpha_{\max}$ to extend the MBR of the ground area.

$$\alpha_{\max} = \arccos\left(\frac{E_{\min}}{O_{\text{apogee}}}\right) - \arcsin\left[\frac{O_{\text{apogee}}}{E_{\text{major}}} \sin\left(\arcsin\left(\frac{E_{\min}}{O_{\text{apogee}}}\right) - \theta_{\max}\right)\right] + \arcsin\left(\frac{E_{\min}}{O_{\text{apogee}}}\right) - \theta_{\max}. \quad (4)$$

5.1.3. Comparison and Analysis of the Calculation Accuracy and Efficiency. To verify the computational efficiency of EMBR-SCA based on the proposed data analysis and processing model, we compare the results of the satellite coverage by using EMBR-SCA and STK [73].

The experiment dataset details are shown in Table 3. The ground area data in Table 3 are the geodetic coordinates of the points that make up the area; the calculation time step of the satellite position is 1 s; θ_H is the horizontal half angle; θ_V is the vertical half angle; and pitch, yaw, and roll are the attitude maneuvering parameters of the satellite ontology. The satellite ephemeris data are represented by the two-line orbital element (TLE).

The comparison of the intersection time windows of the satellite coverage with the ground area is shown first. We use the differences in the start time, end time, and time length separately for the comparison. The results are shown in Figure 12. The range of the differences is less than 1 s, which indicates that the calculation accuracy of EMBR-SCA is close to that of STK.

To compare the coverage rates, the grid size of the ground area in STK is set to 0.05° . The results in Table 4 show that the coverage rate differences between EMBR-SCA and STK are small and meet the actual needs.

To compare the calculation efficiency by using the proposed platform, the traditional algorithm, EMBR-SCA, and EMBR-SCA are tested based on the data processing model with different calculation thread counts. The results in Table 5 show that EMBR-SCA based on the data processing model achieves a more than six times larger speed-up ratio. The time consumption associated with different thread

counts in Figure 13 shows that the more the thread counts, the smaller the data quantities per segment and the shorter the calculation time. When the thread count is constant, the calculation time is proportional to the number of satellites, which is expected. The number of ground areas has minimal impact on the calculation time because the datasets are decomposed into a combination of one satellite with all ground areas. The parameters of the satellite for different ground areas must be calculated only once.

5.1.4. Visualization of the Case. Based on the proposed algorithm, this paper integrates the case into the platform as a demonstration demo. When using this demo, as shown in Figure 14, the time range and time step should be first selected. Then, the earth observation satellite can be constructed, which includes the name and TLE data; following that, a sensor with different parameters should be added. The option to set the attitude of the satellite is also provided in consideration of the attitude change. After that, all the parameters can be displayed in the form of JSON in the right list widget. After the data analysis is performed, the results can be presented in 3D scenario by using Cesium or chart visualization.

5.2. Case II: Microservice-Based Space Object Conjunction Assessment. The increasing number of orbital space objects (satellites, rockets, debris, etc.) has led to a deterioration of the space environment and constituted a serious threat to the normal operation of spacecraft, space missions, and future space developments [69]. The United States and Russia satellite collision that happened on February 10, 2009, is a powerful proof.

The operational safety of space objects requires robust and effective space object information [82]. Preventing the collision between space objects has become an urgent problem that requires a solution, and the conjunction assessment is an evaluation indicator for the collision risk of space objects [83]. Beyond that, the conjunction assessment can be applied to various analyses of space operations, such as the timing determination of space-based target imaging.

To improve the efficiency, the conjunction assessment typically uses a series of filters to screen objects. The most common screening processes are proposed by Hoots et al. [83] and Rodríguez et al. [84], but some filters in the two methods are not robust [85]. In response to these problems, different software and solutions [70, 86, 87] have been proposed to improve the processes. Through the algorithms' experiments and the screening steps, this paper proposes a robust data filter process and uses the process as a base case to test the platform and data processing model proposed here.

5.2.1. Filters of Conjunction Assessment. The filters that have been implemented in this case are as follows:

(1) *Perigee/Apogee Filter.* This is the most robust filter in the classical prescreening schemes, which can filter out lots of space objects. But according to the test from Alfano and Finkleman [88], it is necessary to add a pad of 50 km to the

TABLE 3: Experiment dataset details.

Type	Parameter
Time range	2018-12-01T00:00:00–2018-12-11T00:00:00
ZY-3 2	
Ephemeris	1 41556U 16033A 18360.16266073 .0000446 00000-0 23037-4 0 9996 2 41556 97.4159 75.2737 0003224 104.0789 336.3495 15.21266840142925
Sensor	$\theta_H = 1^\circ$, $\theta_V = 3^\circ$, pitch = 10° , yaw = 10° , roll = 10°
Gaofen 5	
Ephemeris	1 43461U 18043A 18360.11744794-.00000015 00000-0 66166-5 0 9996 2 43461 98.1459 296.8281 0000886 308.5684 51.5438 14.57785406 33726
Sensor	$\theta_H = 2^\circ$, $\theta_V = 5^\circ$, pitch = 20° , yaw = 10° , roll = 20°
WorldView-4	
Ephemeris	1 41848U 16067A 18360.22122525 .00001879 00000-0 22607-3 0 9999 2 41848 97.8946 75.8578 0001575 8.2585 351.8677 14.85111481114924
Sensor	$\theta_H = 3^\circ$, $\theta_V = 3^\circ$, pitch = 20° , yaw = 30° , roll = 20°
Ground area 1	(89.3584°, 31.3039°), (89.2117°, 27.345°), (95.5168°, 27.1495°), (95.5657°, 30.9618°)
Ground area 2	(-110.60°, 43.32°), (-104.75°, 42.83°), (-104.15°, 45.85°), (-104.45°, 49.13°), (-109.68°, 50.54°), (-110.18°, 46.94°)

FIGURE 12: Comparison of calculation results: (a) ground area 1; (b) ground area 2.

TABLE 4: Comparison of the coverage rate.

Result	Coverage rate of ZY-3 2		Coverage rate of Gaofen 5		Coverage rate of WorldView-4	
	Ground area 1	Ground area 2	Ground area 1	Ground area 2	Ground area 1	Ground area 2
STK	39.71%	40.30%	69.85%	90.81%	73.94%	68.62%
EMBR-SCA	39.70%	40.16%	69.83%	90.82%	73.92%	68.88%

TABLE 5: Comparison of the calculation efficiency.

Dataset	Traditional algorithm	Time consumption (ms)	
		EMBR-SCA	EMBR-SCA based on the data processing model (thread count is 8)
Satellite count: 1, ground area count: 1	13192	8579	2025
Satellite count: 1, ground area count: 2	14012	8725	2223
Satellite count: 3, ground area count: 1	38679	27162	4604
Satellite count: 3, ground area count: 2	38821	27295	4828

FIGURE 13: Comparison of time consumption strategies with different parallel counts.

conjunction threshold distance for distance screening based on probability-based actions. The subordinate object cannot pass this filter if Equation (5) is met:

$$\text{Perigee}_{\max} - \text{Apogee}_{\min} > D + 50 \text{ km}, \quad (5)$$

where $\text{Perigee}_{\max}$ is the larger of the two perigees, $\text{Apogee}_{\min}$ is the smaller of the two apogees, and the conjunction threshold distance $D = 5 \text{ km}$.

(2) r^2 Filter. This filter is improved by Chen et al. [89]. The time period $[t, t + \Delta t]$ is selected, and the distance between two objects is compared in time t . If the distance between two objects is greater than $R_{th,1}$ in Equation (6) at all the given analysis intervals, the current subordinate object cannot pass this filter:

$$R_{th,1} = D + v_{\text{escape}} \Delta t, \quad (6)$$

where v_{escape} is the escape speed and Δt is usually 3% of the average orbital period of the two objects [89].

(3) *Minimum Filter and Fine r^2 Filter*. Minimum filter and fine r^2 filter are proposed by Rodríguez et al. [84].

Integrating the above filters, the object filter process is as shown in Figure 15.

5.2.2. Comparison and Analysis of the Calculation Accuracy and Efficiency. The conventional method is less efficient when conjunction assessment is carried out between the primary object and all the other orbital objects in the back-end database, a process referred to as the one-to-all conjunction assessment. The one-to-all conjunction assessment is a typical SSA analysis task, and the data processing model proposed in this paper is very suitable for the conjunction assessment. The time range in the one-to-all conjunction assessment can be segmented by step (1), and each filter can be constructed as a separate map by step (4) in Section 4.2. After all the filters are completed, the objects that are close to the primary object can be obtained by comparing their distance and the conjunction threshold distance.

This study combines the conjunction assessment filters with the proposed data processing model and conducts the

FIGURE 14: Execution flow and visualization of the satellite coverage analysis.

FIGURE 15: The robust object filter process in this case based on different literature and experiments.

related experiments. First, this study calculates the minimum distance and the closest moment between the primary object and the given subordinate objects. The comparative data represents the top 10 objects by minimum range from CelesTrak, and the comparison are shown in Table 6. The start computation time is April 28, 2019, 00:00:00, and the stop computation time is May 5, 2019, 00:00:00. The conjunction threshold distance is 5 km. The ephemeris data of all the objects is obtained from CelesTrak. The result shows that the minimum distance and the closest moment obtained in this paper are very close to the results given by CelesTrak, which indicates that the accuracy of the conjunction assessment can meet the needs.

On the basis of meeting the accuracy, this paper carries out the calculation efficiency test of the one-to-all conjunction assessment. This paper chooses three primary objects for experiment: FENGYUN 1C DEB (NORAD catalog

number: 30879), IRIDIUM 33 DEB (NORAD catalog number: 35619), and MICROSAT-R DEB (NORAD catalog number: 44201). To each primary object, the subordinate objects are all the other orbital objects in the database. The count of the subordinate objects is 17854 (data current as of April 28, 2019, 00:06:00 UTC). The running environments involved in the comparison are listed as follows: the single thread running on a single machine, the stand-alone service instance, and the distributed service instances (three nodes) running the data analysis architecture and process model proposed. As shown in Figure 16, the result shows that the computational efficiency can be greatly improved based on the data analysis architecture and process model proposed in this paper.

5.2.3. Visualization of the Case. Based on the conjunction assessment, the visualization of the case is constructed here,

TABLE 6: Comparison of the result. Data current as of April 28, 2019, 00:06:00 UTC.

NORAD catalog number	Name	Result from CelesTrak		Result from this article	
		Minimum distance (km)	Time (UTC)	Minimum distance (km)	Time (UTC)
41977	FLOCK 3P-58	0.009	2019 Apr 28 00:00:56.197	0.009	2019 Apr 28 00:00:55.862
43901	FLOCK 3K-8				
04363	TOPO 1	0.013	2019 May 02 14:43:45.108	0.014	2019 May 02 14:43:44.745
30879	FENGYUN 1C DEB				
08799	METEOR 1-24	0.041	2019 May 02 03:06:00.511	0.041	2019 May 02 03:06:00.158
34066	COSMOS 2251 DEB				
23606	CERISE	0.046	2019 Apr 29 23:30:16.909	0.045	2019 Apr 29 23:30:16.511
30734	FENGYUN 1C DEB				
29052	FORMOSAT-3 FM2	0.047	2019 May 04 18:41:54.020	0.047	2019 May 04 18:41:52.925
36657	FENGYUN 1C DEB				
39492	AIST-1 (RS-41)	0.058	2019 Apr 28 08:13:56.817	0.057	2019 Apr 28 08:13:56.378
41118	ARIANE 44LP DEB				
43574	IRIDIUM 154	0.060	2019 May 01 14:00:10.375	0.060	2019 May 01 14:00:10.030
36424	COSMOS 2251 DEB				
44083	ASTROCAST 0.2	0.060	2019 May 04 01:50:49.557	0.060	2019 May 04 01:50:48.595
30512	FENGYUN 1C DEB				
39091	BRITE-AUSTRIA	0.072	2019 Apr 30 13:49:05.091	0.072	2019 Apr 30 13:49:04.875
42338	NOAA 16 DEB				
42032	FLOCK 3P-47	0.074	2019 Apr 29 14:59:00.907	0.074	2019 Apr 29 14:59:00.473
43855	CHOMPTT

FIGURE 16: The runtime of one-to-all conjunction assessment in different running environments.

FIGURE 17: Execution flow and visualization of the conjunction assessment. In the scenario, by selecting different object combinations, the conjunction assessment results (the NORAD catalog number of objects, the minimum distance, and the closest time) and the change of distance between the objects can be displayed in a different way.

as shown in Figure 17. The algorithms of the filters and distance computation have been integrated with the data analysis architecture and processing model proposed. For a one-to-all analysis, the primary object can be selected at the front end. After the analysis task is submitted, the server side begins the conjunction assessment based on the data analysis architecture and the processing model. The analysis results are stored in the cache service, and the users get them through the data retrieval service. By constructing a 3D scenario, the conjunction assessment can be visualized.

6. Conclusion and Future Plans

The development, deployment, and maintenance of the current SSA service systems have become more complex, which hinders the application of innovative algorithms and the promotion and dissemination of the easy-to-understand SSA knowledge to the public. In response to these problems, we propose the microservice-based platform for SSA data analytics, which is designed for SSA promotion and for innovation researchers to test and integrate their algorithms. This platform allows sustainable delivery and is easy to deploy. To

facilitate the usage and development, different dataset sources are merged and the SSA data service is constructed. The platform builds a series of basic services and describes the deployment method. Aiming at the massive computing needs in SSA, this paper proposes the data analysis architecture and processing model. Based on the above, the platform is deployed and tested via two typical case studies. In order to better improve the cognition of analysis results, different schemes are used for the data visualization. The results show that the platform proposed can easily integrate various analysis algorithms and use efficient architecture and calculation strategies to satisfy various needs of the public and scientific institutions.

In the future, we will reinforce the prototype platform to create greater convenience and benefits. To meet the needs of SSA applications and improve the practicability of the platform, more practical applications and cases need to be integrated and tested. Simultaneously, the platform will provide users with more comprehensive service access interfaces in the future to meet the integration of heterogeneous data and user-defined algorithms. Effective and appropriate visualization methods can significantly reduce the cognitive

difficulty of SSA information, which shows that more comprehensive visualization solutions need to be integrated. The increase of system services requires the construction of the overall system monitor. Furthermore, the rapid development of artificial intelligence (AI) and machine learning (ML) also opens up new research directions for SSA, and the future platform will build user-oriented machine learning microservices that allow users to apply data, models, and algorithms for the cognitive improvement of SSA.

Data Availability

The authors declare that all data underlying the findings of the manuscripts can be shared with researchers, and the authors allow all readers to be able to access the data supporting the conclusions of the study. There is no unavailable data that cannot be released.

Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

Authors' Contributions

Wanjie Lu and Qing Xu conceived and designed the research; Wanjie Lu, Chaozhen Lan, Liang Lyu, and Qunshan Shi developed the platform together; Yang Zhou provided the algorithms for building cases; Wanjie Lu and Yinghao Zhao participated in data collection; Wanjie Lu and Chaozhen Lan wrote the paper.

Acknowledgments

We would like to thank all individuals who contributed to the development of the platform, including the students who assisted in improving the platform by testing and reporting bugs. We would like to thank Editage (<https://www.editage.cn>) for English language editing. This research was funded by the National Natural Science Foundation of China, grant number 41701463.

References

- [1] B. Lal, A. Balakrishnan, B. M. Caldwell, R. S. Buenconsejo, and S. A. Cariosciam, "Global trends in space situational awareness (SSA) and space traffic management (STM)," April 2019, <https://www.ida.org/-/media/feature/publications/g/gl/global-trends-in-space-situational-awareness-ssa-and-space-traffic-management-stm/d-9074.ashx>.
- [2] J. A. Kennewell and B. N. Vo, "An overview of space situational awareness," in *Proceedings of the 16th International Conference on Information Fusion*, pp. 1029–1036, Istanbul, Turkey, July 2013.
- [3] G. H. McCall and J. H. Darrah, "Space situational awareness: difficult, expensive and necessary," *Air & Space Power Journal*, vol. 28, no. 6, pp. 6–16, 2014.
- [4] B. C. Weeden and P. J. Cefola, "Computer systems and algorithms for space situational awareness: history and future development," in *12th International Space Conference of Pacific-basin Societies*, pp. 28–30, Montreal, Canada, July 2018.
- [5] D. Richmond, "Space situational awareness architecture vision," in *Advanced Maui Optical and Space Surveillance Technologies Conference*, pp. 10–13, Maui, Hawaii, September 2013.
- [6] P. J. Cefola, B. Weeden, and C. Levit, "Open source software suite for space situational awareness and space object catalog work," in *4th International Conference on Astrodynamics Tools Techniques*, pp. 3–6, Madrid, Spain, May 2010.
- [7] M. Villamizar, O. Garcés, H. Castro et al., "Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud," in *2015 10th Computing Colombian Conference (10CCC)*, pp. 21–25, Bogota, Colombia, September 2015.
- [8] M. Sarkarati, M. Spada, S. Moulin, and D. Fischer, "Leveraging advanced software technologies for implementing the European space situational awareness ground data systems," in *SpaceOps 2010 Conference*, p. 1922, Huntsville, Alabama, April 2010.
- [9] M. Czajkowski, A. Shilliday, N. Lofaso, A. Dipon, and D. Van Brackle, "The orbitoutlook data archive," in *Advanced Maui Optical and Space Surveillance Technologies Conference*, pp. 20–23, Maui, Hawaii, September 2016.
- [10] J. Raley, R. M. Weisman, C. Chow, M. Czajkowski, and K. Sotzen, "Orbitoutlook: autonomous verification and validation of non-traditional data for improved space situational awareness," in *Advanced Maui Optical and Space Surveillance Technologies Conference*, pp. 20–23, Maui, Hawaii, September 2016.
- [11] K. D. Pham, B. Liu, Y. Chen et al., "Cloud-based space situational awareness: initial design and evaluation," in *Sensors and Systems for Space Applications VI*, p. 87390M, Baltimore, MD, USA, May 2013.
- [12] M. Morton and T. Roberts, "Joint Space Operations Center (JSpOC) Mission System (JMS)," in *Advanced Maui Optical and Space Surveillance Technologies Conference*, pp. 11–14, Maui, Hawaii, September 2011.
- [13] K. Walter, "Development of an early warning information infrastructure using spatial web services technology," *International Journal of Digital Earth*, vol. 3, no. 4, pp. 384–394, 2010.
- [14] S. Law, J. Dorny, J. Stallings, and J. Loving, "Cloud services for space situational awareness," in *Advanced Maui Optical and Space Surveillance Technologies Conference*, pp. 19–22, Maui, Hawaii, September 2017.
- [15] A. Balalaie, A. Heydarnoori, and P. Jamshidi, "Microservices architecture enables devOps: migration to a cloud-native architecture," *IEEE Software*, vol. 33, no. 3, pp. 42–52, 2016.
- [16] S. A. Kaiser, "Legal and policy aspects of space situational awareness," *Space Policy*, vol. 31, pp. 5–12, 2015.
- [17] B. Weeden, "Global space situational awareness sensors," in *Advanced Maui Optical and Space Surveillance Technologies Conference*, pp. 14–17, Maui, Hawaii, September 2010.
- [18] M. A. Skinner, "Conceptual development of a civil space traffic management system capability," in *Advanced Maui Optical and Space Surveillance Technologies Conference*, pp. 11–14, Maui, Hawaii, September 2018.
- [19] SAIC, "Space-Track," April 2019, <https://www.space-track.org>.
- [20] C. Peat, "Heavens-Above," April 2019, <https://www.heavens-above.com>.
- [21] T. Chow, "Space situational awareness sharing program: an SWF issue brief," April 2019, https://swfound.org/media/3584/ssa_sharing_program_issue_brief_nov2011.pdf.
- [22] T. S. Kelso, "CelesTrak," April 2019, <http://celestrak.com/>.

- [23] G. D. Krebs, "Gunter's space page," April 2019, <https://space.skyrocket.de>.
- [24] AGI, "SpaceBook," April 2019, <http://apps.agi.com/SatelliteViewer>.
- [25] N2YO, "N2YO.com," April 2019, <http://www.n2yo.com>.
- [26] N. Kratzke, "A brief history of cloud application architectures," *Applied Sciences*, vol. 8, no. 8, pp. 1368–1393, 2018.
- [27] M. Fowler and J. Lewis, "Microservices a definition of this new architectural term," April 2019, <https://martinfowler.com/articles/microservices.html>.
- [28] N. Alshuqayran, N. Ali, and R. Evans, "A systematic mapping study in microservice architecture," in *2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA)*, pp. 44–51, Macau, China, November 2016.
- [29] P. Di Francesco, P. Lago, and I. Malavolta, "Architecting with microservices: a systematic mapping study," *Journal of Systems and Software*, vol. 150, pp. 77–97, 2019.
- [30] C. Pahl and P. Jamshidi, "Microservices: a systematic mapping study," in *Proceedings of the 6th International Conference on Cloud Computing and Services Science*, pp. 23–25, Rome, Italy, April 2016.
- [31] S. Baškarada, V. Nguyen, and A. Koronios, "Architecting microservices: practical opportunities and challenges," *Journal of Computer Information Systems*, vol. 58, pp. 1–9, 2018.
- [32] N. Dragoni, I. Lanese, S. T. Larsen, M. Mazzara, R. Mustafin, and L. Safina, "Microservices: how to make your application scale," in *Perspectives of System Informatics*, pp. 95–104, Springer, Cham, Switzerland, 2017.
- [33] J. Hutchinson, G. Kotonya, J. Walkerdine, P. Sawyer, G. Dobson, and V. Onditi, "Evolving existing systems to service-oriented architectures: perspective and challenges," in *IEEE International Conference on Web Services (ICWS 2007)*, pp. 9–13, Salt Lake City, UT, USA, July 2007.
- [34] D. Namiot and M. Sneps-Sneppé, "On micro-services architecture," *International Journal of Open Information Technologies*, vol. 2, no. 9, pp. 24–27, 2014.
- [35] C. Xia, Y. Zhang, L. Wang, S. Coleman, and Y. Liu, "Micro-service-based cloud robotics system for intelligent space," *Robotics and Autonomous Systems*, vol. 110, pp. 139–150, 2018.
- [36] D. Bernstein, "Containers and cloud: from LXC to Docker to Kubernetes," *IEEE Cloud Computing*, vol. 1, no. 3, pp. 81–84, 2014.
- [37] C. Esposito, A. Castiglione, and K.-K. R. Choo, "Challenges in delivering software in the cloud as microservices," *IEEE Cloud Computing*, vol. 3, no. 5, pp. 10–14, 2016.
- [38] A. Bucchiarone, N. Dragoni, S. Dustdar, S. T. Larsen, and M. Mazzara, "From monolithic to microservices: an experience report from the banking domain," *IEEE Software*, vol. 35, no. 3, pp. 50–55, 2018.
- [39] D. Kim, H. Muhammad, E. Kim, S. Helal, and C. Lee, "TOSCA-based and federation-aware cloud orchestration for Kubernetes container platform," *Applied Sciences*, vol. 9, no. 1, pp. 191–203, 2019.
- [40] Apache Software Foundation, "Apache Hadoop," April 2019, <http://hadoop.apache.org>.
- [41] Apache Software Foundation, "Apache Spark," April 2019, <http://spark.apache.org>.
- [42] Apache Software Foundation, "Apache Storm," April 2019, <http://storm.apache.org>.
- [43] Apache Software Foundation, "Apache Flink," April 2019, <https://flink.apache.org>.
- [44] L. Gu and H. Li, "Memory or time: performance evaluation for iterative operation on Hadoop and Spark," in *2013 IEEE 10th International Conference on High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing*, pp. 721–727, Zhangjiajie, China, November 2013.
- [45] A. Verma, A. H. Mansuri, and N. Jain, "Big data management processing with Hadoop MapReduce and Spark technology: a comparison," in *2016 Symposium on Colossal Data Analysis and Networking (CDAN)*, pp. 18–19, Indore, India, March 2016.
- [46] M. A. Lopez, A. Lobato, and O. C. M. B. Duarte, "A performance comparison of open-source stream processing platforms," in *2016 IEEE Global Communications Conference (GLOBECOM)*, pp. 4–8, Washington, DC, USA, December 2016.
- [47] O.-C. Marcu, A. Costan, G. Antoniu, and M. S. Perez-Hernandez, "Spark versus Flink: understanding performance in big data analytics frameworks," in *2016 IEEE International Conference on Cluster Computing (CLUSTER)*, pp. 433–442, Taipei, China, September 2016.
- [48] P. Carbone, A. Katsifodimos, S. Ewen, V. Markl, S. Haridi, and K. Tzoumas, "Apache Flink: stream and batch processing in a single engine," *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, vol. 36, no. 4, pp. 28–38, 2015.
- [49] Z. Ni and Z. Xia, "Design and implementation of the parallel Apriori algorithm on Flink platforms," *Journal of China Jiliang University*, vol. 29, no. 2, pp. 175–180, 2018.
- [50] M. Hanif, E. Kim, S. Helal, and C. Lee, "SLA-based adaptation schemes in distributed stream processing engines," *Applied Sciences*, vol. 9, no. 6, pp. 1045–1065, 2019.
- [51] C. Györödi, R. Györödi, I. Andrada, and L. Bandici, "A comparative study between the capabilities of MySQL vs. mongoDB as a back-end for an online platform," *International Journal of Advanced Computer Science and Applications*, vol. 7, no. 11, pp. 73–78, 2016.
- [52] J. Pokorny, "NoSQL databases: a step to database scalability in web environment," *International Journal of Web Information Systems*, vol. 9, no. 1, pp. 69–82, 2013.
- [53] V. Jain and A. Upadhyay, "MongoDB and NoSQL databases," *International Journal of Computer Applications*, vol. 167, no. 10, pp. 16–20, 2017.
- [54] C. Györödi, R. Györödi, G. Pecherle, and A. Olah, "A comparative study: mongoDB vs. MySQL," in *2015 13th International Conference on Engineering of Modern Electric Systems (EMES)*, pp. 1–6, Oradea, Romania, June 2015.
- [55] C. Nance, T. Lossner, R. Iype, and G. Harmon, "NoSQL vs RDBMS-why there is room for both," in *Proceedings of the Southern Association for Information Systems Conference*, pp. 111–116, Savannah, GA, USA, March 2013.
- [56] R. Lawrence, "Integration and virtualization of relational SQL and NoSQL systems including MySQL and mongoDB," in *2014 International Conference on Computational Science and Computational Intelligence*, pp. 285–290, Las Vegas, NV, USA, March 2014.
- [57] S. Bjeladinovic, "A fresh approach for hybrid SQL/NoSQL database design based on data structuredness," *Enterprise Information Systems*, vol. 12, no. 8-9, pp. 1202–1220, 2018.

- [58] D. Li, Y. Yao, Z. Shao, and L. Wang, "From digital earth to smart earth," *Chinese Science Bulletin*, vol. 59, no. 8, pp. 722–733, 2014.
- [59] D. G. De Paor, M. M. Dordevic, P. Karabinos, S. Burgin, F. Coba, and S. J. Whitmeyer, "Exploring the reasons for the seasons using Google Earth, 3d models, and plots," *International Journal of Digital Earth*, vol. 10, no. 6, pp. 582–603, 2016.
- [60] J. Hunter, C. Brooking, L. Reading, and S. Vink, "A web-based system enabling the integration, analysis, and 3D sub-surface visualization of groundwater monitoring data and geological models," *International Journal of Digital Earth*, vol. 9, no. 2, pp. 197–214, 2016.
- [61] AGI, "Cesium," April 2019, <https://cesiumjs.org/>.
- [62] L. Zhu, Z. Wang, and Z. Li, "Representing time-dynamic geospatial objects on virtual globes using CZML—part i: overview and key issues," *ISPRS International Journal of Geo-Information*, vol. 7, no. 3, pp. 97–117, 2018.
- [63] L. Zhu, Z. Li, and Z. Wang, "Representing time-dynamic geospatial objects on virtual globes using CZML—part ii: impact, comparison, and future developments," *ISPRS International Journal of Geo-Information*, vol. 7, no. 3, pp. 102–121, 2018.
- [64] AGI, "Cesium language (CZML) guide," April 2019, <https://github.com/AnalyticalGraphicsInc/czml-writer/wiki/CZML-Guide>.
- [65] P. Mazzetti, S. Nativi, and J. Caron, "RESTful implementation of geospatial services for earth and space science applications," *International Journal of Digital Earth*, vol. 2, no. 1, pp. 40–61, 2009.
- [66] Docker, "Docker," April 2019, <https://www.docker.com>.
- [67] D. Liu and L. Zhao, "The research and implementation of cloud computing platform based on Docker," in *2014 11th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*, pp. 475–478, Chengdu, China, December 2014.
- [68] O. Kalden and C. Bodemann, "Building space situational awareness capability," in *Proceedings of 5th International Conference on Recent Advances in Space Technologies - RAST2011*, pp. 650–654, Istanbul, Turkey, June 2011.
- [69] X. Bai, "Survey on collision assessment and warning techniques for space object," *Journal of Astronautics*, vol. 34, no. 8, pp. 1027–1039, 2013.
- [70] T. S. Kelso, D. A. Vallado, J. Chan, and B. Buckwalter, "Improved conjunction analysis via collaborative space situational awareness," in *European Conference on Space Debris*, pp. 453–461, Darmstadt, Germany, April 2009.
- [71] Union of Concerned Scientists, "UCS satellite database," April 2019, <https://www.ucsusa.org/nuclear-weapons/space-weapons/satellite-database>.
- [72] J. McDowell, "Jonathan's space home page," April 2019, <http://www.planet4589.org/space/>.
- [73] AGI, "Systems tool kit," April 2019, <https://www.agi.com>.
- [74] D. Greenly, M. Duncan, J. Wysack, and F. Flores, "Space situational awareness data processing scalability utilizing Google Cloud services," in *Advanced Maui Optical and Space Surveillance Technologies Conference*, pp. 15–18, Maui, Hawaii, September 2015.
- [75] L. Rodríguez-Pupo, C. Granell, and S. Casteleyn, "An analytics platform for integrating and computing spatio-temporal metrics," *ISPRS International Journal of Geo-Information*, vol. 8, no. 2, pp. 54–73, 2019.
- [76] B. Zhang, L. Yu, Y. Feng, L. Liu, and S. Zhao, "Application of workflow technology for big data analysis service," *Applied Sciences*, vol. 8, no. 4, pp. 591–613, 2018.
- [77] H. Wang, D. Qiu, X. Huang, M. Ma, and J. Zhu, "Research on schedule technique for area-detecting electronic reconnaissance satellite," *Acta Armamentarii*, vol. 32, no. 11, pp. 1365–1372, 2011.
- [78] J. Wei and R. Zhang, "Optimization design and simulation analysis of earth reconnaissance satellite constellation," *Journal of Xidian University*, vol. 40, no. 2, pp. 138–147, 2013.
- [79] X. Shen, D. Li, and H. Yao, "A fast algorithm for imaging time window prediction of optical satellites considering J2 perturbation for imaging mission scheduling," *Geomatics and Information Science of Wuhan University*, vol. 37, no. 12, pp. 1468–1471, 2012.
- [80] H. Liu, B. Cai, and Q. Zhu, "Self-adaptive planning method of imaging reconnaissance area coverage," *Geomatics and Information Science of Wuhan University*, vol. 42, no. 12, pp. 1719–1724, 2017.
- [81] R. Wang, "Analysis algorithm for satellite regional coverage based on polygonal Boolean operation," *Journal of Equipment Academy*, vol. 27, no. 2, pp. 83–87, 2016.
- [82] R. Hall, S. Alfano, and A. Ocampo, "Advances in satellite conjunction analysis," in *Advanced Maui Optical and Space Surveillance Technologies Conference*, pp. 14–17, Maui, Hawaii, September 2010.
- [83] F. R. Hoots, L. L. Crawford, and R. L. Roehrich, "An analytic method to determine future close approaches between satellites," *Celestial Mechanics*, vol. 33, no. 2, pp. 143–158, 1984.
- [84] J. R. A. Rodríguez, F. M. Fadrique, and H. Klinkrad, "Collision risk assessment with a "smart sieve" method," *European Space Agency Esa Sp*, vol. 486, no. 486, pp. 159–164, 2002.
- [85] J. Woodburn, V. Coppola, and F. Stoner, "A description of filters for minimizing the time required for orbital conjunction computations," *Advances in the Astronautical Sciences*, vol. 135, no. 2, pp. 1157–1173, 2009.
- [86] B. Abernathy, D. Surka, S. Harvey, and M. O'Connor, "The CAOS-D architecture for conjunction analysis," in *Infotech@Aerospace 2011*, pp. 1434–1444, St. Louis, MO, USA, March 2011.
- [87] L. M. Healy, "Close conjunction detection on parallel computer," *Journal of Guidance, Control, and Dynamics*, vol. 18, no. 4, pp. 824–829, 1995.
- [88] S. Alfano and D. Finkleman, "On selecting satellite conjunction filter parameters," *Acta Astronautica*, vol. 99, pp. 193–200, 2014.
- [89] L. Chen, X.-Z. Bai, Y.-G. Liang, and K.-B. Li, *Orbital Data Applications for Space Objects*, Springer, Singapore, Singapore, 1st edition, 2017.