information

MDPI

Article

SOOCP: A Platform for Data and Analysis of Space Object Optical Characteristic

Wanjie Lu *, Qing Xu and Chaozhen Lan

Institute of Geospatial Information, Information Engineering University, Zhengzhou 450052, China; 13937169139@139.com (Q.X.); 13014511234@163.com (C.L.)

* Correspondence: lwj285149763@163.com; Tel.: +86-185-3990-8814

Received: 8 August 2019; Accepted: 24 September 2019; Published: 25 September 2019

Abstract: With the advancement of various technologies, the research and application of space object optical characteristic (SOOC), one of the main characteristics of space objects, are faced with new challenges. Current diverse structures of massive SOOC data cannot be stored and retrieved effectively. Moreover, SOOC processing and application platforms are inconvenient to build and deploy, while researchers' innovative algorithms cannot be applied effectively, thereby limiting the promotion of the research achievements. To provide a scaffolding platform for users with different needs, this paper proposes SOOCP, a SOOC data and analysis service platform based on microservice architecture. Using the hybrid Structured Query Language (SQL)/NoSQL service, the platform provides efficient data storage and retrieval services for users at different levels. For promoting research achievements and reusing existing online services, the proposed heterogeneous function integration service assists researchers and developers in independently integrating algorithmic modules, functional modules, and existing online services to meet high concurrency requests with a unified interface. To evaluate the platform, three research cases with different requirement levels were considered. The results showed that SOOCP performs well by providing various data and function integration services for different levels of demand.

Keywords: space object; optical characteristic; SQL; NoSQL; microservice; function integration

1. Introduction

The optical characteristic is one of the basic characteristics of space objects. As an important strategic resource [1], space object optical characteristic (SOOC) data can be used for state estimation, auxiliary identification, early detection, and calculation of the optical scattering cross section of space objects [2,3]. The establishment of an SOOC database is a key aspect of space object characteristic research. To effectively analyze the SOOC and achieve the objective of transforming data and algorithms into practical applications, it is necessary to not only establish stable and reliable SOOC data services [1,4–7] but also study different algorithms and applications for a comprehensive analysis [8,9]. However, with the increasing number of space objects and data acquisition methods as well as rapidly growing data volumes, users are imposing higher requirements on data services and applications of SOOC.

Owing to current technological advancements, the management of massive SOOC data is a major challenge. Multi-source heterogeneous SOOC data, such as radiant data, optical images, and infrared spectral data of materials, can be categorized into structured and unstructured data. The current mainstream solution to meet the data retrieval requirements of SOOC experiments and simulations is based on mature structured query language (SQL) databases, such as Oracle [1,4,10], Microsoft SQL Server [7,11], and MySQL [12], that manage the structured data and the metadata of the unstructured data [5,6]. Users can query structured SOOC data with different conditions through the application

programming interface (API) provided by different databases. However, unstructured SOOC data, such as videos, images, and text, are usually serialized and stored in the local file system [13]. Although the storage scheme based on SQL databases can effectively reduce data redundancy and efficiently process complex data queries for structured data [14], it is not only inconvenient for the management and retrieval of unstructured SOOC data but also faces difficulties in maintaining the association between the structured data in the SQL database and the unstructured data. As a solution to this problem, the SQL database was used to store structured data and the metadata of unstructured data while the Hadoop Distributed File System (HDFS) and Apache HBase were used to store unstructured SOOC files of different sizes [15]. Although this solution alleviates the challenges of storage and management of unstructured data to a certain extent, it actually regards unstructured data as an integral part of structured data; hence, it provides neither direct access to unstructured data nor a unified access interface for users unfamiliar with SQL or Not only SQL (NoSQL) databases.

The processing of massive SOOC data imposes higher requirements on experimental platforms and applications. Many existing platforms and applications have been developed and deployed using monolithic architecture [16–18], which is a tightly coupled architecture. In such systems or platforms, different modules are integrated and developed using a single programming language. When modifying existing functions or adding new modules, the compatibility between different modules must be considered. The maintenance and development of an application based on monolithic architecture require sufficient human and material resources; hence, such an application cannot meet the needs of continuous delivery. The characteristics of monolithic architecture have led to expansion and deployment difficulties, especially in the cloud computing environment [9,19]. Repeated deployments of applications based on monolithic architecture in different hardware platforms requires a complete operating environment, which leads to higher costs in terms of human, material, and financial resources. Furthermore, users cannot share SOOC data, algorithms, or functional modules under monolithic architecture easily and efficiently. Based on Software as a Service (SAAS), various simulation and experiment software were integrated [17] for optoelectronic testing. The deployment of each software based on SAAS requires professionals, which leads to the need for further improvement in flexibility.

The latest algorithms and research achievements related to SOOC require efficient experimentation, popularization, and practical application. In the context of rapid technological advancements, to accelerate such experimentation and application, different industries have built various scaffolding platforms. For example, Sino-InSpace [20], a platform proposed for geographical environment, provides a virtual space environment for users and researchers to build visualization scenarios and test algorithms. Songshan et al. [21] constructed a service-oriented model encapsulation strategy that allows users to share and integrate different heterogeneous geo-analysis models. Based on service-oriented architecture (SOA), the Geosciences Network (GEON) project [22] facilitates data sharing by integrating a variety of standard services and modules to meet different needs. For X-ray free electron laser (XFEL) applications, Liubov et al. [9] built a cross-platform wave optics software environment to solve a wide set of XFEL optics problems; this platform can be accessed by a variety of programming languages (e.g., MATLAB, Python, and C++). In terms of SOOC data experiments and analyses, researchers have mainly tested algorithms using existing simulation software [23]. In optical-related research, a platform was proposed [17] to integrate various software based on SAAS for on-demand requirements; however, the deployment of these software requires professionals, whereas researchers are only users, not participants. Various algorithms have been developed for processing SOOC data, such as image enhancement and contour extraction to obtain information from optical images of space objects and attitude estimation methods based on a sequence of photometric image data. Nevertheless, technologies related to the integration of modules and online services have not attracted much research attention from researchers. Therefore, the integration of existing algorithms and research achievements into existing platforms faces several obstacles. Moreover, current SOOC simulation systems can only load a small amount of data and they require a high data reading speed during experiments. These factors hinder the testing and application of algorithms.

In view of the challenges discussed above and considering the concept of X as a Service (XaaS), this paper proposes SOOCP, a service platform for SOOC data and analysis based on the microservice architecture (MSA). The objectives of this study are to provide efficient and convenient SOOC data services, algorithm experimentation services, and functional module integration services considering the rapid increase in space objects and SOOC data. Through the basic services provided, researchers can conveniently store and retrieve SOOC data, test algorithms, and transform research achievements into online services to promote the application of scientific research.

The remainder of this paper is organized as follows. Section 2 describes the architecture and orientation of the platform. Section 3 introduces the key technologies of the platform for the orientation and user requirements. To facilitate the use of the platform by different application levels, three levels of application models are designed and the corresponding case studies are presented in Section 4. Finally, Section 5 states the conclusions and explores directions for future work.

2. Platform Architecture and Orientation

SOOCP is orientated as a scaffolding platform that provides efficient storage solutions for SOOC data of different structures. In addition, it allows the integration and testing of heterogeneous algorithms, functional modules, and online services. The overall architecture of the platform is based on MSA. MSA can split a complex software system into single-function and small-grained services [24–26]. Each service can be independently developed, tested, and deployed, with a lightweight communication mechanism to exchange data [27]. MSA allows a team to build services by using appropriate programming languages and tools according to the business context; thus, it is advantageous compared to monolithic architecture [28] and SOA [29].

The platform architecture and orientation of SOOCP are shown in Figure 1. The clients (e.g., *Desktop Computer, Browser*, and *Mobile Device*) request services through the APIs in a representational state transfer (REST) architectural style [30]. The clients can use a series of display methods for the processing results, such as three-dimensional (3D) visualization. *Service Gateway* provides a unified RESTful API access interface, and all requests from the clients need to go through *Service Gateway*; thus, exposure of the internal APIs of services is avoided. Users need not be concerned with the interaction between internal services, and the cases of service upgrade, modification, and expansion do not affect user experience. *Load Balancing* is responsible for assigning requests, preventing excessive load on the background services, and improving the response speed and overall performance. When new instances of a service are added, *Load Balancing* can reasonably allocate requests to the newly added instances. All services are registered in *Service Registry*, through which each service finds the others. *Service Configuration* stores the attribute configurations of all services.

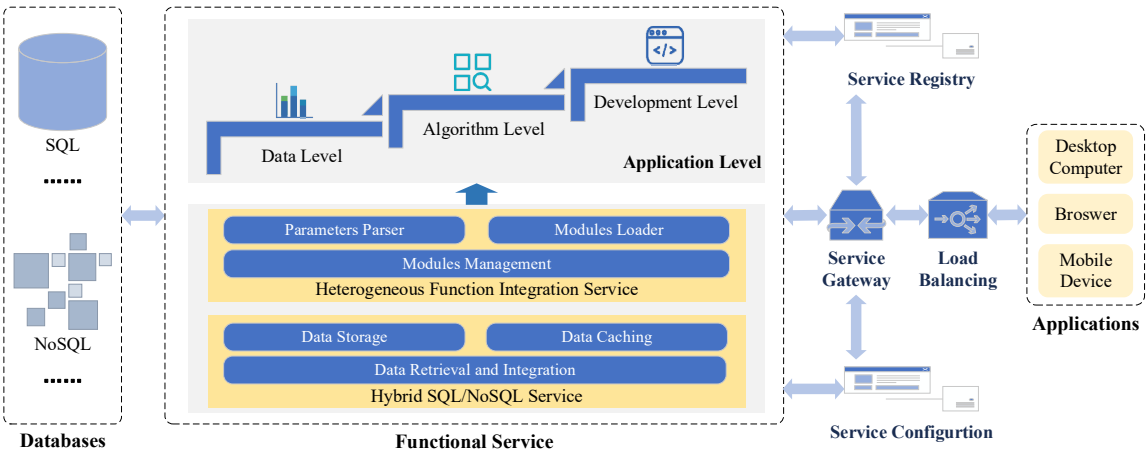

Figure 1. Platform architecture and orientation. All functional services are built on top of databases, and provide different levels of applications for different requirements and users.

Database and functional services are two essential services of a data and analysis service platform. As shown in Figure 1, the platform is oriented as an opened scaffolding platform, and only *Hybrid SQL/NoSQL Service* and *Heterogeneous Function Integration Service* are provided.

Efficient and convenient data services can not only enable researchers to share and obtain data more effectively but also facilitate experiments on algorithmic and functional modules. Therefore, for SOOC data of different structures, the platform constructs *Hybrid SQL/NoSQL Service*, which provides efficient storage and retrieval for data of different structures. This service includes data storage, data caching, and data retrieval and integration. *Data Storage* is mainly responsible for importing data that meets the system's predefined format and for storing data in appropriate databases. *Data Caching* manages cached data in various processing situations and temporally stores data that need to be preserved in a persistent database. *Data Retrieval and Integration* queries data of different structures from databases on the basis of external requests and integrates the data obtained into a simple data structure that can be used directly. *Data Retrieval and Integration* avoids direct operation of the database by users as well as the writing of complex SQL statements, especially query statements for NoSQL databases.

Practitioners in the field of SOOC include data users, algorithm researchers, and development engineers, and the levels of user requirements vary considerably. For example, data users need data for analysis and visualization. Algorithm researchers study different innovative algorithms to process data. Development engineers mainly construct various business functions that can be actually used by the data users and algorithm researchers. Therefore, it is necessary to provide basic services that can meet different levels of custom requirements. *Heterogeneous Function Integration Service* is a built-in service of the platform, which allows the integration of user-defined SOOC processing and analysis functions, including algorithmic modules, functional modules, and online services; thus, users can avoid additional repetitive work by concentrating on researching the algorithms without considering how to implement the distributed deployment and high availability of online services. Owing to the loose coupling and easy deployment features of MSA, development engineers can integrate existing online services into the platform.

3. Key Technologies

3.1. SOOC Hybrid SQL/NoSQL Service

The storage scheme based on an SQL database (i) is not suitable for storing unstructured data, which leads to difficulties in data management and retrieval, (ii) does not have good scalability, resulting in complex and expensive data service clusters, and (iii) can satisfy data consistency requirements but cannot provide higher efficiency or availability for massive concurrent access. NoSQL databases, which can meet the requirement of high concurrent access, overcome the shortcomings of SQL databases to some extent. Owing to their excellent features, NoSQL databases have been used in many successful Internet applications [14,31]. NoSQL databases have many data storage models, and the most commonly used categories of data models are key-value store, wide-column store, document store and graph store [32]. Key-value databases can store data in memory and guarantee the low latency of data acquisition, which means that key-value databases are often used to handle high access loads of data, such as data cache [33]. Document databases store semi-structured and unstructured data and can handle complex data formats very well, regardless of the data schema. In document databases, each document can be formatted differently, and new structured data can be added without changing the existing documents [33]. Data can have a nested structure and document stores often use internal notations, which can be processed directly in applications [34]. For the diversity of data structures of SOOC and the frequent access of users to the same data when testing algorithms, databases of the key-value model and document model, which are used in the proposed service architecture, are suitable for storing data of various structures and formats. However, although the NoSQL databases selected [35] (i) can meet the requirements of large transaction volumes, low-latency access, and high service availability of massive data, (ii) have flexible data models, suitable for storing data of various

structures, and (iii) can update the data without affecting the existing data structure, they cannot provide better consistency, and in contrast to SQL databases, they cannot support complex data queries.

To meet the storage requirements of structured and unstructured SOOC data, the platform builds a SOOC hybrid SQL/NoSQL service based on the advantages of SQL and NoSQL databases in respective fields.

3.1.1. SOOC Hybrid SQL/NoSQL Service Architecture

The data model of the proposed hybrid SQL/NoSQL service is shown in Figure 2. The SQL database mainly stores structured data (e.g., object data, equipment data, environment data, and task data), the metadata of unstructured data, and partial processing results. The metadata of unstructured data are mainly used for retrieving data, fusing information, and building relationships with structured data. Unstructured data mainly include images, models, text, and cached data. Cached data, which were not considered in [15], include intermediate data and results generated in data processing for real-time access, or frequently accessed data for improving the access performance and reducing the load on the back-end databases. The SOOC data model can be formalized as follows:

$$\begin{aligned} \textit{OpticalCharacteristicEntity} = \{&\textit{ObjectID}, \textit{Object}, \textit{Equipment}, \textit{Environment}, \textit{Task}, \textit{Condition}, \\ &[\textit{Metadata of Unstructured data}, \textit{Unstructured data}], [\textit{Cached data}]\}, \end{aligned}$$

where [-] represents the optional parameter; *OpticalCharacteristicEntity* represents the SOOC data object; *ObjectID* is the unique identifier, i.e., the association index of different data of the same space object; *Object* contains basic information such as name, description, type, size, and ephemeris data; *Equipment*, *Environment*, *Task*, and *Condition* are parameters that mainly record the equipment, test environment, task-related information, and experimental parameters used for acquiring SOOC data, respectively; *Metadata of Unstructured Data* is the metadata of unstructured data; *Unstructured Data* includes various SOOC unstructured data, such as images, models, and text; and *Cached data* is the frequently accessed data, including processing results that need to be stored in persistent databases or data that need to be shared between different processes in real time.

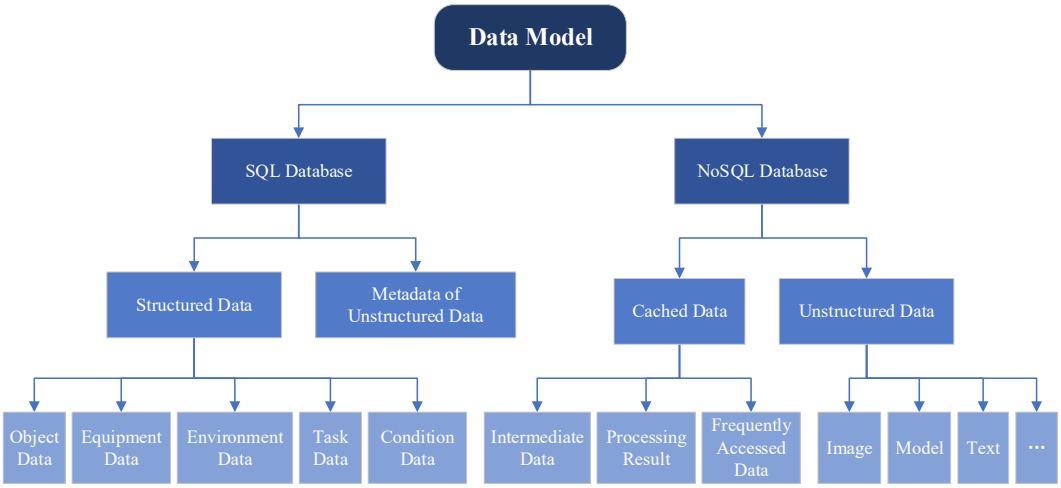

Figure 2. Data model. The storage formats of structured data and the metadata of unstructured data are predefined and fixed. The cached data changes according to requirements and the format is variable.

Through the full utilization of SQL and NoSQL databases, the data management architecture for storage and access requirements in different scenarios is designed as shown in Figure 3. The database in the architecture can be divided into three types:

1. *Cache Database* primarily stores the cached data. During the processing of SOOC data, different data need to be obtained from various databases for testing, while intermediate data and results

that need to be shared or stored in databases are generated. During experiments on algorithms, researchers mainly improve and iterate the internal processing of the algorithms, and the input data of the algorithms usually remain unchanged. Retrieving raw data from databases each time will lead to lower efficiency and exert pressure on the databases and servers; moreover, such data need to be organized and integrated to meet the requirements each time. By storing frequently accessed data in *Cache Database*, the testing data can be obtained in real time. By storing intermediate data in *Cache Database*, data sharing can be realized between multiple modules. In addition, the processing results can not only be stored in *Cache Database* temporarily but also be transferred to different persistent databases.

2. *SQL Database* provides multiple patterns of query strategies to satisfy the storage and retrieval requirements of structured data. When the data requested do not exist in *Cache Database*, they can be obtained from *SQL Database*. The metadata of unstructured data in *SQL Database* can be used as an intermediate bridge to retrieve unstructured data. *SQL Database* can also store the processing results.
3. *Unstructured Database* provides high-performance queries for unstructured data, which can be not only retrieved through the metadata in SQL databases but also queried directly from NoSQL databases. For processing results, such as images and text, *Unstructured Database* can effectively meet the storage requirements for the subsequent processing of these data.

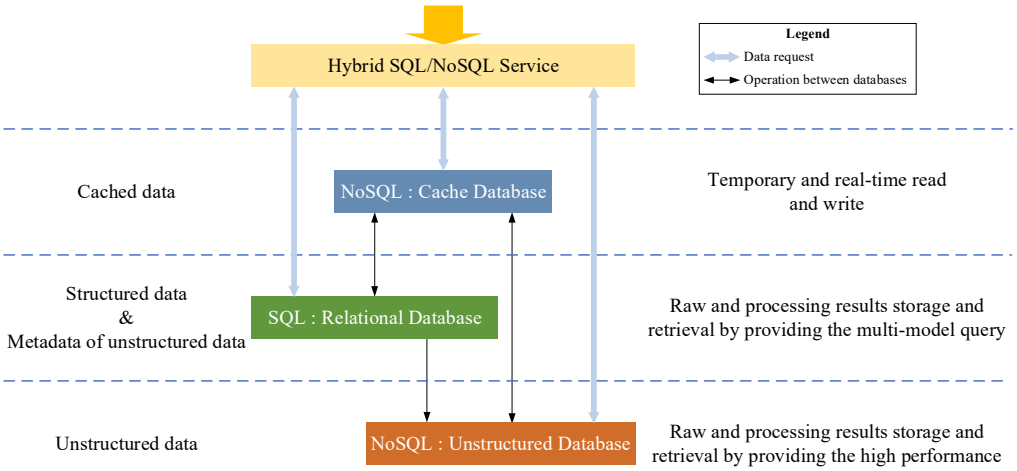

Figure 3. Hybrid storage architecture. Different types of databases store data for different structures and purposes. It should be noted that different types of NoSQL databases can be used to store different data. For example, Redis can be used as *Cache Database*, while MongoDB as *Unstructured Database.*

The hybrid SQL/NoSQL logical model and cases are shown in Figure 4. In the platform, MySQL [12] is the SQL database, MongoDB [35] is the unstructured database, and Redis [36] is the cache database. MySQL stores structured data (e.g., table *ObjectInformation*) and the metadata of unstructured data (e.g., table *MetaData*). Table *ObjectInformation* and *MetaData* associate with each other through the fields *id* and *object_id*. Each object in table *ObjectInformation* is unique and corresponds to none, one, or more of the metadata in the table *MetaData*. MongoDB is used to store the unstructured data, which can be linked to the data in MySQL through the field *collection* stored in the table *MetaData*. For the data obtained from MySQL and MongoDB, Redis is used as the cache database for caching the repeatedly accessed data to not only improve the reading and writing speed but also reduce the load on the back-end databases.

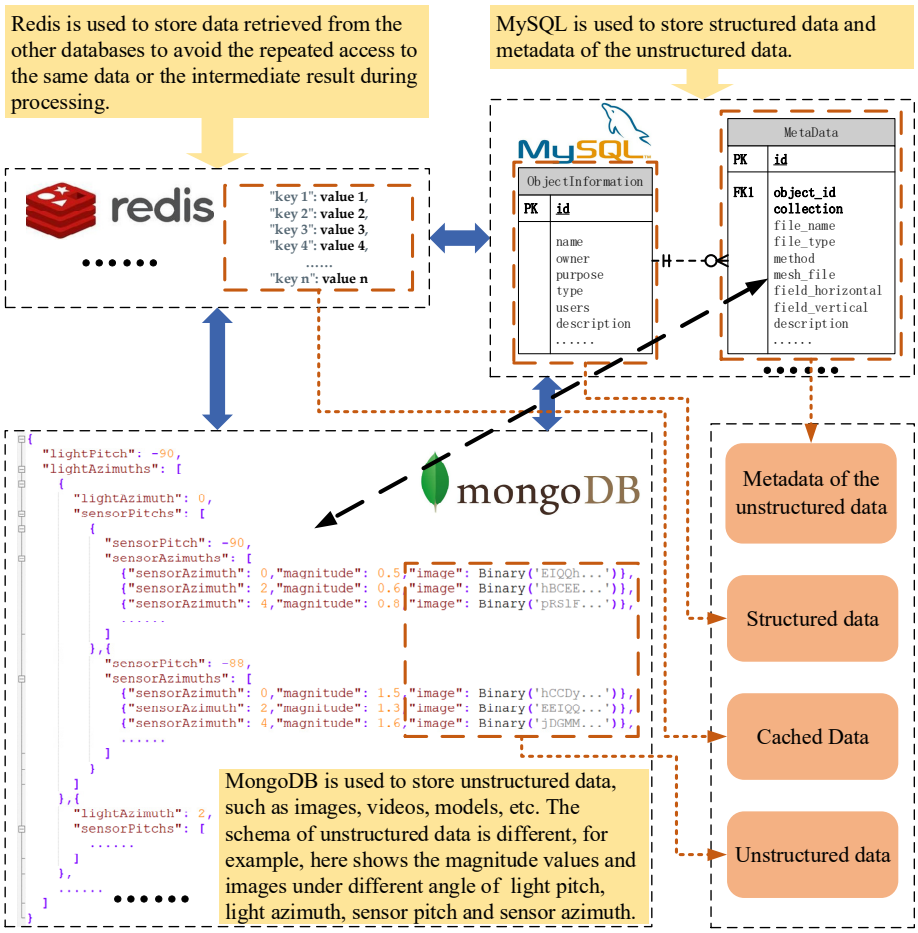

Figure 4. Hybrid SQL/NoSQL logical model and cases. Different databases store different data and collaborate with each other to provide a unified data service.

3.1.2. SOOC Hybrid SQL/NoSQL Data Access Flow

By building an appropriate data access flow, the platform can provide efficient and convenient data services for various requirements. The hybrid SQL/NoSQL data access flow is shown in Figure 5. As a part of the hybrid SQL/NoSQL service, *Data Retrieval and Integration* provides data integration as well as a unified access interface for the data in different databases. The data access flow is divided into the following parts:

Part 1 Retrieve cached data. When the module *Data Retrieval and Integration* receives the data query request (1: Request), it first queries data from the cache database (2: Request cached data), returns the result (3: Return), and judges whether cached data exists (4: Cached data exists?). If the cached data exists, the result will be returned to the user (5: Return cached data) without requesting data from MySQL or MongoDB; otherwise, **Part 2** will be executed.

Part 2 Retrieve structured data and metadata of structured data. When the user requests structured data, these data can be obtained from MySQL (6: Query structured data and metadata) and returned to the module *Data Retrieval and Integration* (7: Return). The frequently accessed raw data can be stored in the cache database (8: Store result in cache database) to ensure data access efficiency. Owing to different user requirements, it is necessary to judge whether to retrieve unstructured data in *Data Retrieval and Integration*. When it is not necessary to obtain unstructured data, the result is directly returned to the users (12: Return); otherwise, **Part 3** will be executed on the basis of the user requirements or the metadata of the unstructured data.

Part 3 Retrieve unstructured data. When unstructured data are requested, the query parameters can be obtained according to the user requirements or the metadata of the unstructured data acquired from **Part 2**. The unstructured data obtained from MongoDB (9: Query unstructured data) will be processed in *Data Retrieval and Integration* (10: Return) for ease of use and returned to the users (12: Return). The frequently accessed raw data can be stored in the cache database (11: Store result in cache database) to ensure data access efficiency.

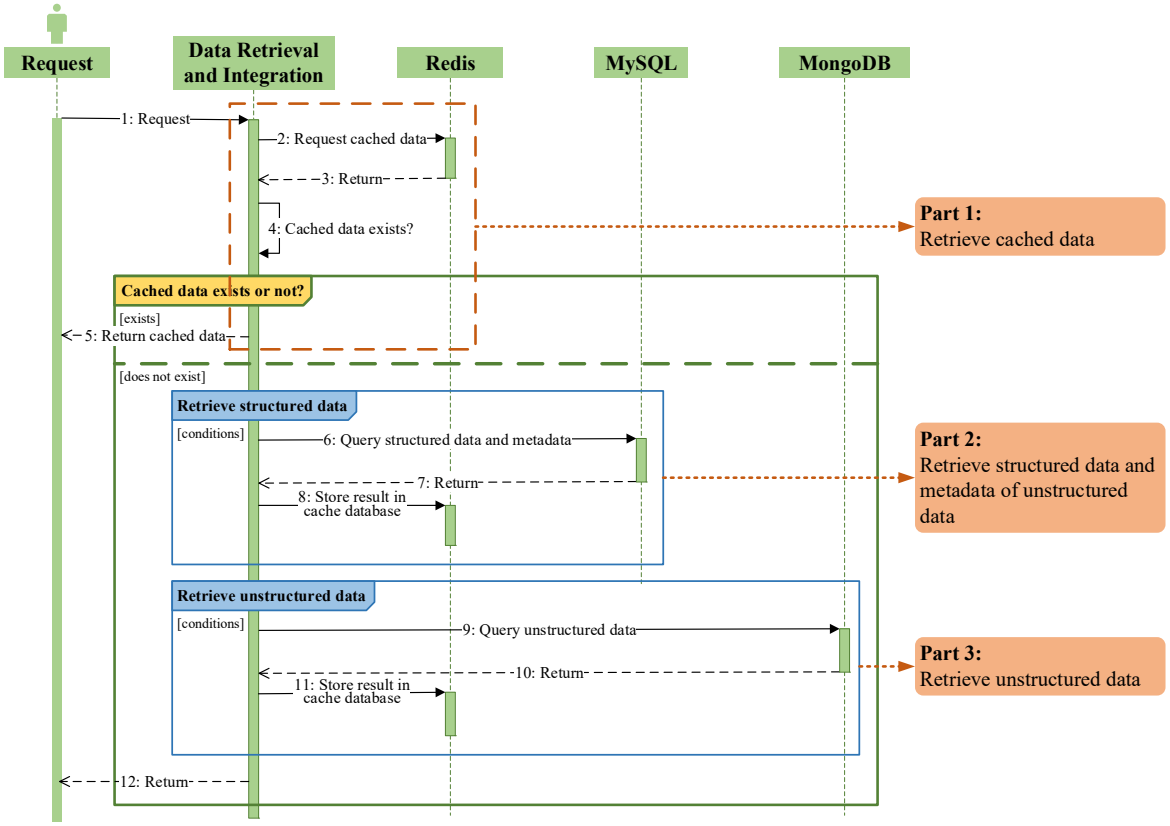

Figure 5. Hybrid SQL/NoSQL access flow. [*conditions*] represents the parameters for retrieving different data from MySQL or MongoDB.

The final query results, especially those from structured and unstructured data, need to be integrated by *Data Retrieval and Integration* according to the format of *OpticalCharacteristicEntity*.

3.1.3. Comparison of Different SOOC Data Services

Many SOOC data services are based on SQL databases, and Yanqi et al. [15] used Oracle to store structured data and metadata of unstructured data, Hadoop (Apache Hbase and HDFS) to store unstructured SOOC files, and Redis to provide cache service. As analyzed previously, services based on SQL databases cannot meet research requirements. The service based on Oracle and Hadoop is not easy to deploy and maintain. As a commercial database, Oracle has higher hardware requirements, and Apache Hbase or HDFS are not flexible enough for different data formats. By contrast, MySQL is a lightweight database, and MongoDB can handle complex data formats regardless of the data schema. Simultaneously, MySQL and MongoDB are easy to deploy and maintain. The detailed comparisons between hybrid SQL/NoSQL service and the service based on RDBMS and Hadoop proposed in Reference [15] are as shown in Table 1. Since SQL databases cannot meet the requirements, the solutions based on SQL databases are no longer listed. For hybrid SQL/NoSQL service and service based on RDBMS and Hadoop, only the different items are compared. The comparison results in Table 1 indicate that hybrid SQL/NoSQL service is more suitable for researchers and research institutions.

Table 1. Comparison of different SOOC data services.

Item	Service Based on RDBMS and Hadoop	Hybrid SQL/NoSQL Service
Deployment	Not easy to deploy and not user-friendly	Easy to deploy and user-friendly
Maintenance	Difficult and the updates of Oracle and Hadoop are complex	Easy to maintain and update
Data Support	Support data of different structures, but unable to support variable data schemas	Support data of different structures and schemas, and also support new formats in existing documents
Access Interface	Do not provide a direct or unified access interface for users unfamiliar with SQL or NoSQL databases	Support to access all data through a unified interface or data of different structures separately
Cost	May need more money and time	Based on opensource software with less money and time
Purpose	Mainly for enterprise business applications and medium or large companies	Mainly for startup or smaller scientific research teams and companies
Flexibility	Moderate	High

3.2. SOOC Heterogeneous Function Integration Service

As a scaffolding platform, SOOCP does not provide specific SOOC analysis functions; instead, it provides the ability to integrate multi-source heterogeneous functions. Using this platform, researchers can concentrate on the innovation and experimentation of algorithms. The heterogeneous function integration service provided by the platform mainly (i) integrates algorithmic and functional modules developed by researchers in different programming languages and (ii) offers online service integration for development engineers.

3.2.1. SOOC Algorithmic and Functional Module Integration

The algorithms built by researchers in different programming languages and operating environments are multi-source heterogeneous algorithms. By using the integration service, researchers can integrate the multi-source heterogeneous algorithmic and functional modules into the platform and thus test and practically apply the algorithms. To assist researchers in integrating multi-source heterogeneous modules autonomously, the platform adopts algorithmic and functional module integration, which mainly consists of *Module Manager*, *Module Loader*, and *Parameter Parser*, as shown in Figure 6.

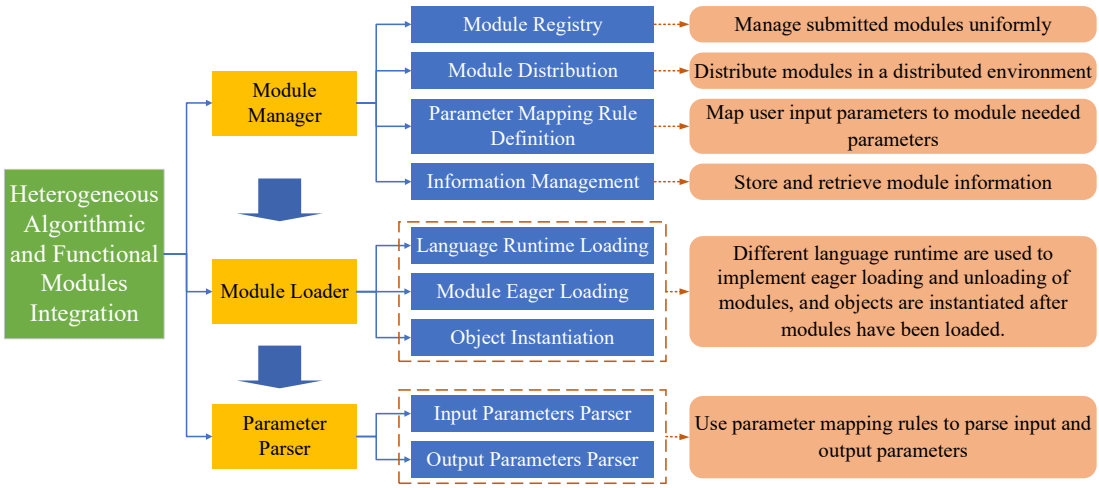

Figure 6. Heterogeneous algorithmic and functional module integration. The combination of module manager, module loader, and parameter parser completes the integration.

Module Manager manages the modules submitted by various researchers. The submitted modules need to be registered and the related information needs to be stored in databases to facilitate data retrieval for use of the modules. High concurrency requests from researchers require the construction of distributed applications. *Module Manager* can distribute each module on different platforms to meet the high concurrency requirements through load balancing. Different algorithms require ordered input and output parameters, and the user input parameters need to match with the required parameters of these modules on the basis of the parameter mapping rule. *Module Manager* also provides information management, i.e., it manages the basic information about each module as well as the requirements for the input and output parameters. On the basis of the required information, researchers can construct corresponding parameters to drive different modules for data analysis. Usually, the heterogeneous algorithmic and functional modules are stored in the functional server, and the input and output parameters of each module are stored in the database, as shown in Figure 7.

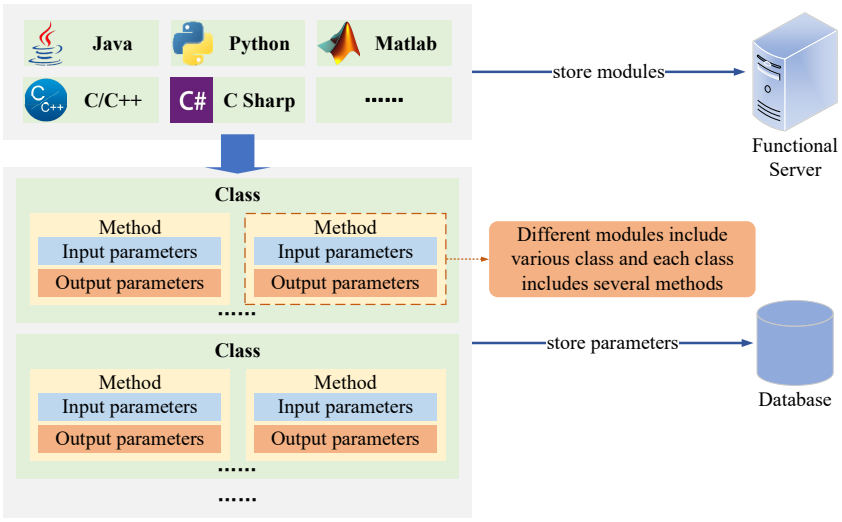

Figure 7. Storage of algorithmic and functional modules, components, and parameters. The various modules are stored on the functional server ready to be called, and the corresponding parameters are stored in the database for retrieving.

Through *Module Loader*, different algorithmic and functional modules implement eager loading, as shown in Figure 8. To support modules developed in different languages, *Module Loader* provides language runtime loading for different programming languages. Before different modules are loaded, the language runtime of each module will be run, and each module will execute eager loading after the running environment is established. Each module usually includes at least one class, and each class includes at least one method. *Module Loader* constructs the instantiated objects by the class name of the module and calls methods through these instantiated objects.

Different modules have different input and output parameters. The input parameters from researchers need to be in one-to-one correspondence with the input parameters of the module, which can be completed by the input parameter parser in *Parameter Parser*, and the process results of the modules can be parsed by the output parameter parser in *Parameter Parser*.

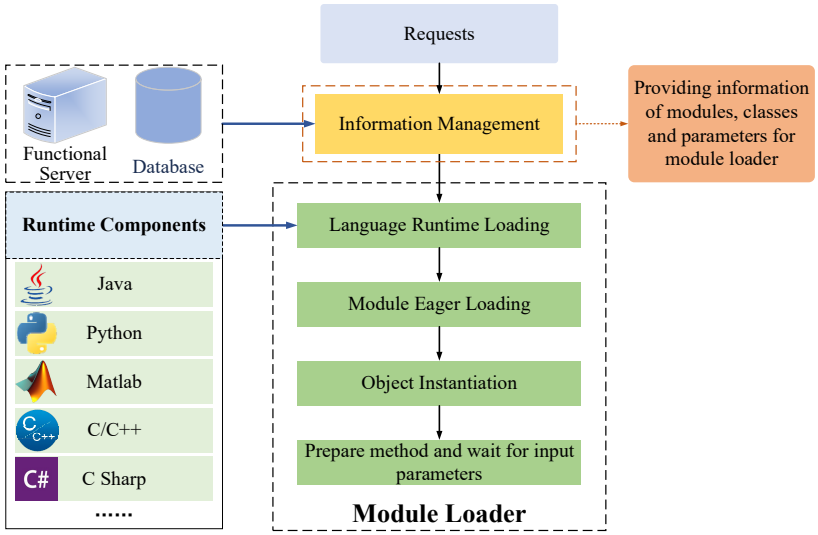

Figure 8. Work flow of *Module Loader*. The runtime components include different language running environments.

3.2.2. SOOC Online Service Integration

Existing online services need to be split by functions or even re-programmed to use the integration method described in Section 3.2.1, which is difficult and expensive. Therefore, for development engineers with relevant online service development capabilities, SOOCP provides the integration of various independent online services on the basis of the sidecar model [37] in MSA.

Based on the sidecar model, the platform can add services for different languages in separate processes without affecting other services. By attaching different services to the platform using the sidecar model, the functions of the platform can be extended and enhanced. All the integrated online services can share the basic services, such as service registry, load balancing, and service gateway, provided by the platform. Each heterogeneous online service is not only loosely coupled to the platform but also developed and deployed independently and autonomously using the sidecar model. The sidecar model can exploit existing online services, while reducing the duplication of components and increasing service availability. The online service integration method based on the sidecar model is shown in Figure 9. Each existing online service needs (i) to provide a status link for the platform to check its health status and (ii) a dedicated integration service (e.g., C/C++ Service, Nodejs Service in Figure 9). The online service can be accessed normally through the unified access interface provided by the service gateway when the health check meets the requirements. By load balancing, the online services can be easily scaled horizontally.

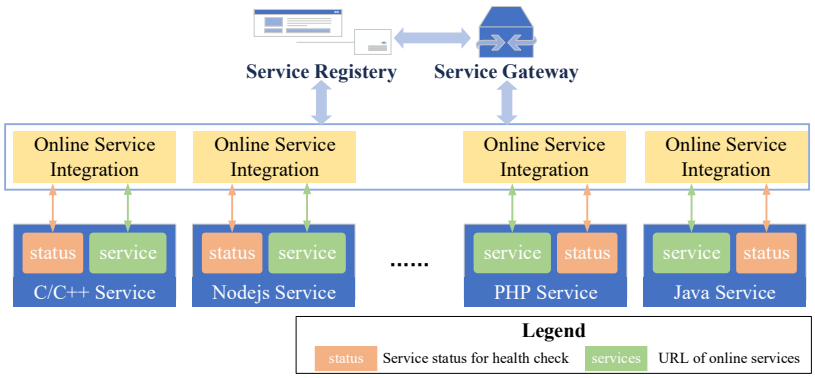

Figure 9. Sidecar model of the platform. Each service can be accessed after passing the health status check.

4. Platform Application Modes and Case Study

Different levels of users in the field of SOOC have diverse requirements. As a scaffolding platform, the proposed platform provides appropriate services to meet the needs of each level. As shown in Figure 10, the design orientation of this platform mainly aims to meet the requirements of data level, algorithm level, and development level.

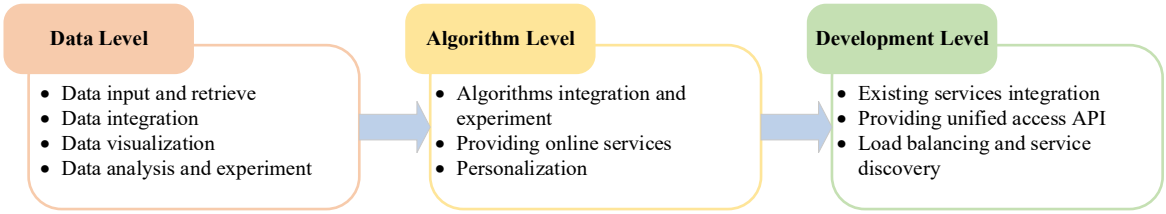

Figure 10. Design of platform application modes. Different levels have different requirements and application modes.

4.1. Data Level

At this level, the platform stores various SOOC data in the databases, which can be accessed through the network. Thus, it is convenient for users to store and retrieve SOOC data. For SOOC data, users are mainly concerned with data input and retrieval. Therefore, data input and retrieval experiments are conducted for testing, and some applications of the data level are presented.

4.1.1. Data Input and Retrieval Efficiency

The experimental data are the historical orbital ephemeris data and the simulated optical characteristic data of the Worldview 1 satellite, as shown in Figure 11. Historical orbit ephemeris data, current as of May 19, 2019, 00:00:00 UTC, are obtained from Space-Track.org [38] in the format of two line element (TLE). The optical characteristic data are the full-angle simulated data, which include four angles: optical source pitch, optical source azimuth, detector pitch, and detector azimuth. The sampling interval of the four angles is 2°. The experimental data include 180 magnitude data files and the corresponding optical image files for each set of the four angles at an equivalent distance of 500 km. The data volume of the magnitude data files is 5.58 GB, including approximately 260 million pieces of magnitude data, and the data volume of the optical image files is 1245.6 GB, including approximately 260 million image files. The experimental hardware platform is a PC with an Intel® 8-core CPU and 32 GB of memory. The operating system was Ubuntu 16.04 LTS and the databases are MySQL (version: 8.0.15), MongoDB (version: 4.0.6), and Redis (Version: 4.0.13). Redis is a real-time cache database and is used only in the cases of temporary data reading and writing, which will not be compared here.

The experimental data are divided into two parts: (i) numerical data and (ii) numerical and image data. The numerical data include the historical orbit ephemeris data and the magnitude data, and the image data include the optical image files. The contrasting storage strategies are shown in Table 2.

Table 2. Storage strategies and experimental data.

Experiment	Storage Strategy	
	MySQL	**Hybrid SQL/NoSQL Service**
Data input of numerical data	Storing all the numerical data	MySQL stores the historical orbit ephemeris data and MongoDB stores the magnitude data
Data input of numerical and image data	Storing all the numerical data and image files	MySQL stores the historical orbit ephemeris data and MongoDB stores the magnitude data and image files

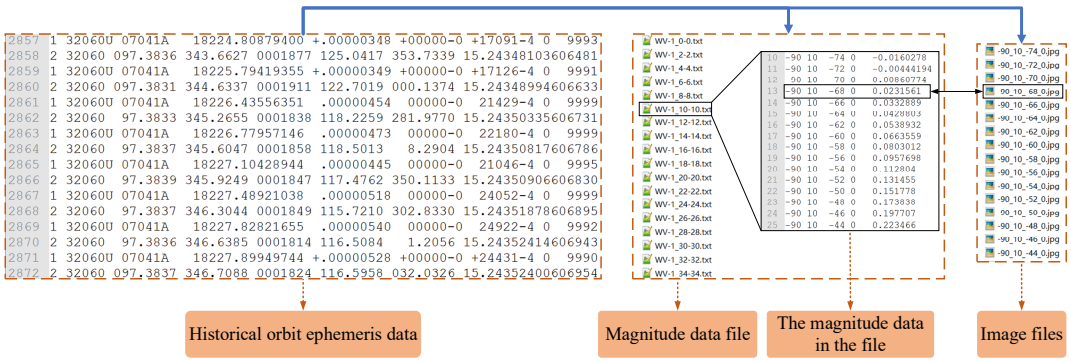

Figure 11. Experimental data. Each magnitude data file includes the magnitude data under different directions of the optical source and detector. Similarly, the image files are the optical images under different directions of the optical source and detector. Historical orbit ephemeris data is used to calculate the position of the satellite, and the direction of the optical source and detector can be obtained by the relative position of the satellite, the detector, and the sun. The corresponding magnitude data and image file can be retrieved by the angle of the optical source and detector, for example, when the optical source pitch is −90°, the optical source azimuth is −10°, the detector pitch is −68°, and the detector azimuth is −0°, the value of magnitude is 0.0231561 and the image file is "−90_10_−68_0.jpg".

(1) Efficiency comparison of data input

Figure 12 compares the data input efficiency of MySQL and the hybrid SQL/NoSQL service under different data volume inputted each time. The results show that the data input efficiency of MySQL is significantly lower than that of the hybrid SQL/NoSQL service when the same data volume is inserted each time. The hybrid SQL/NoSQL service is at least 17 (resp. 34) times more efficient than MySQL in the case of input numerical data (resp. input numerical data and image files) as shown in Figure 12a (resp. Figure 12b). Thus, Figure 12 shows the excellent performance of the hybrid SQL/NoSQL service when storing massive heterogeneous data. In particular, when storing unstructured data, such as image files, the hybrid SQL/NoSQL service shows better performance than MySQL.

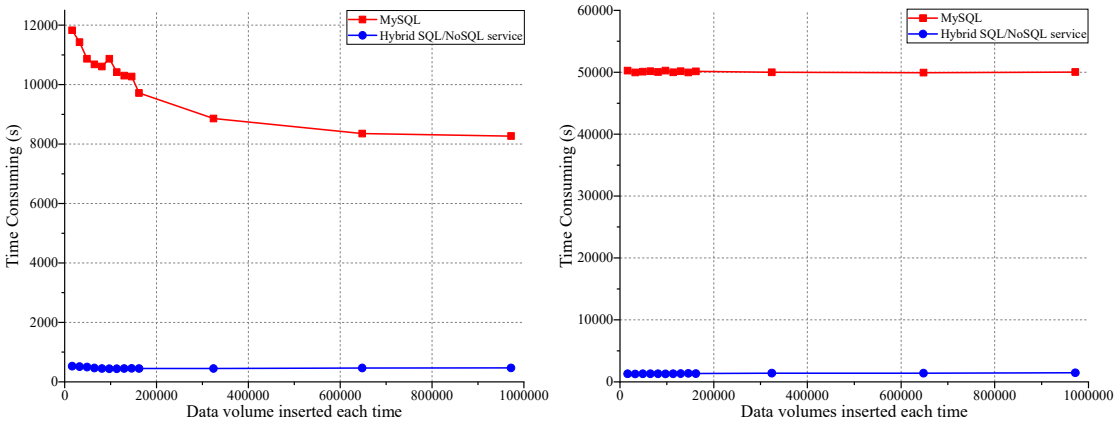

Figure 12. Efficiency comparison of data input: (**a**) numerical data; (**b**) numerical data and image files.

(2) Efficiency comparison of data retrieval

The main purpose of storing SOOC data in databases is to provide data services for application scenarios such as experiments, analyses, and simulations. Therefore, meeting the data access requirements, especially in high concurrency situations, is extremely important. Apache JMeter [39]

was used to test the retrieval efficiency of MySQL and the hybrid SQL/NoSQL service; their efficiencies in terms of responses to concurrent requests are compared in Figure 13. The comparison shows that with an increase in concurrent requests, the response time of MySQL increases rapidly and exceeds that of the hybrid SQL/NoSQL service. Thus, the proposed hybrid SQL/NoSQL service can meet the data retrieval requirements in a highly concurrent environment.

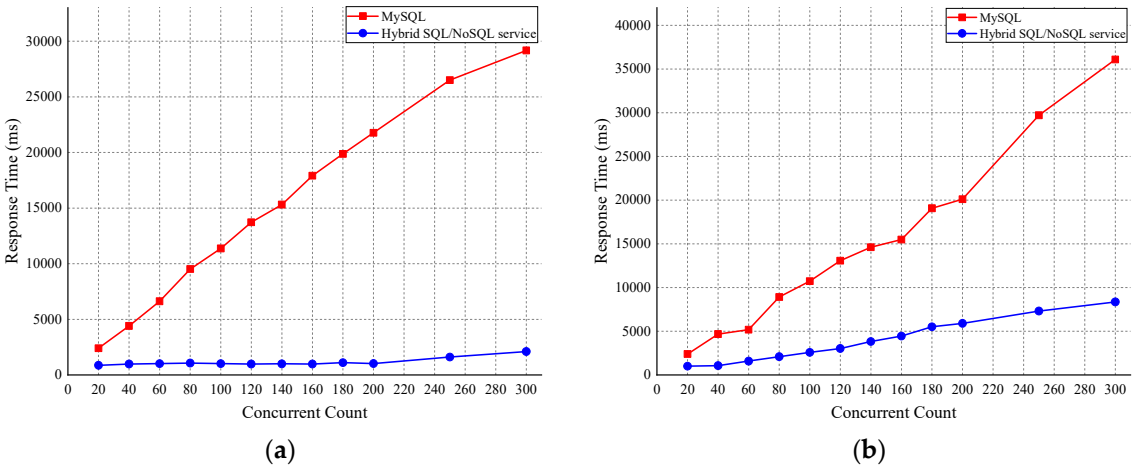

Figure 13. Efficiency comparison of responses to concurrent requests: (**a**) data retrieval of numerical data; (**b**) data retrieval of numerical data and image files.

4.1.2. SOOC Data Visualization

The platform uses Echarts [40], an advanced visualization component, to provide visualization of the data level cases. Users obtain different visualization results, such as optical scattering magnitude, optical scattering intensity, and optical scattering images, by selecting different data and parameters. By taking the visualization of optical scattering magnitude as an example, users need to select one object from the object list as well as the data file of the selected object that needs to be visualized. The object list and the data file name of each object are retrieved from the database. The selected data in the following cases are the simulated photometric data of the Worldview 1 at the equivalent distance of 500 km.

Figure 14 shows the planar heatmap of the magnitude under different detector pitch and azimuth values to better display the relationship between various angles and magnitudes, and the 3D heatmap is shown in Figure 15. By moving the mouse on the 3D heatmap, the magnitude under different detector pitch and azimuth values can be intuitively perceived when the optical angle is fixed; thus, the cognitive level of users is improved. Figure 16 shows how the magnitude changes with the detector azimuth when the optical source pitch, optical source azimuth, and detector pitch are fixed.

In addition to the visualization results discussed above, users can obtain richer visualization results based on the comprehensive data retrieval interface provided, and they can use the retrieved data for SOOC analyses in different algorithms and applications.

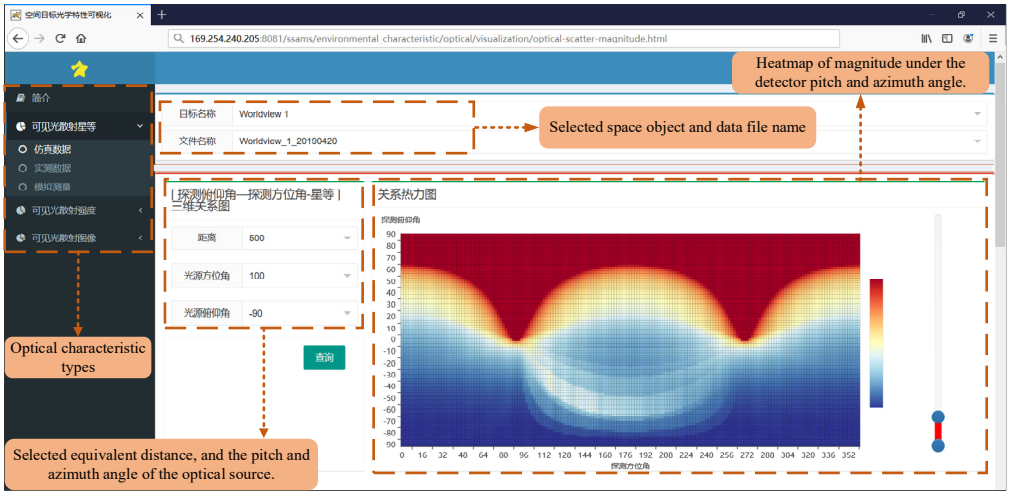

Figure 14. Home page of the visualization. The visualization displayed is the planar heatmap of the magnitude of the Worldview 1 satellite. The data of the space object needed to be visualized are obtained by selecting the space object and the data file name. By selecting the optical source pitch and azimuth, the planar heatmap under different optical sources can be obtained. In the heatmap, the *x*-axis is the detector azimuth, the *y*-axis is the detector pitch, and the value is the magnitude value. In this figure, the selected space object and the data file name are "Worldview 1" and "Worldview_1_20190420" respectively; the values of optical source pitch, optical source azimuth, and equivalent distance are −90°, 100°, and 500 km, respectively.

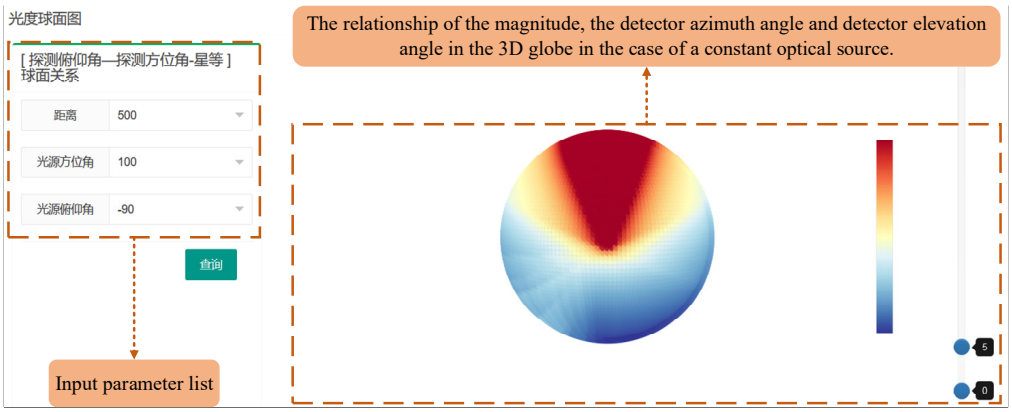

Figure 15. 3D heatmap of the magnitude of the Worldview 1 satellite. In this figure, the values of optical source pitch, optical source azimuth, and equivalent distance are −90°, 100°, and 500 km, respectively.

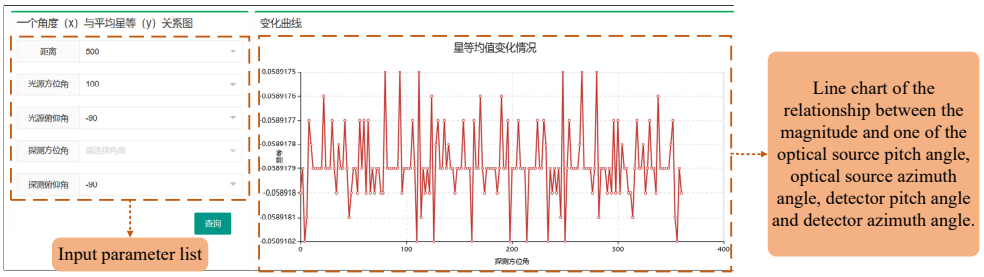

Figure 16. Line chart of magnitude changes of the Worldview 1 satellite with the detector azimuth when the optical source pitch, optical source azimuth, and detector pitch are fixed. In this figure, the values of optical source pitch, optical source azimuth, detector pitch, and equivalent distance are −90°, 100°, −90°, and 500 km, respectively. In the heatmap, the *x*-axis is the detector azimuth and the *y*-axis is the magnitude value.

4.2. Algorithm Level

Researchers in the field of SOOC often place greater emphasis on the research and experimentation of algorithms, while they are not proficient in sharing the research achievements externally. In general, professional developers are required to build and integrate the algorithms and research achievements; thus, the promotion of the latest innovative algorithms is restricted. Based on the proposed SOOC heterogeneous function integration service, researchers can conveniently build distributed online services for publishing the algorithms and research achievements externally.

In the case of the algorithm level, the SOOC data analysis algorithm developed in MATLAB is taken as an example to illustrate algorithm integration and application. The test images, which have been converted into grayscale images, are obtained from the observed video of the International Space Station (ISS) [41], and the tested algorithms are image enhancement and contour extraction.

The purpose of image enhancement is to improve the visual effect of images, highlight the meaningful information for human or machine analysis, and suppress useless information. To highlight the grayscale interval of interest and relatively suppress other grayscale intervals, three-stage linear transformation is used, as shown in Figure 17. In contour extraction, based on the threshold determination algorithm [42], the grayscale image is converted into a binary image, and a two-dimensional eight-connected neighborhood is then constructed to determine the contour.

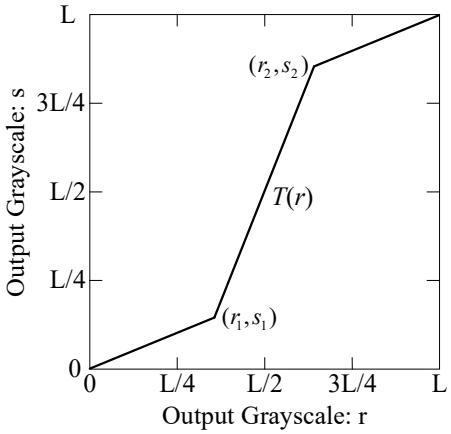

Figure 17. Three-stage linear transformation. As a typical linear transformation, the position of the points can control the shape of the transformation function.

The algorithmic module can be packaged using the library compiler provided by MATLAB and integrated using the heterogeneous function integration service. As shown in Figure 18, the version needs to be set, and the component needs to be selected initially. Furthermore, it is important to list the input and output parameters in detail. Function submission can be performed when all the parameters are correct.

Information on the various algorithmic and functional modules submitted can be retrieved from the databases. By selecting the information in the module information list shown in Figure 19, the format of the access link and result can be obtained. The format of the access link is as follows:

http://ip:port/{language}/{runtime-version}/{module}/{version}/{class}/{method}/{input-parameters},

where *ip* and *port* are the IP address and port of the platform, respectively; {*language*} and {*runtime-version*} are the developing language and the runtime version used for the algorithm; {*module*} is the name of the module selected; {*version*} is the version of the currently selected function; {*class*} and {*method*} build the specific algorithm to be implemented; and {*input-parameters*} is the list of parameters required to execute the algorithm. As shown in Figure 19, the format of the result indicates the content of the processing result by accessing the link of the selected algorithm.

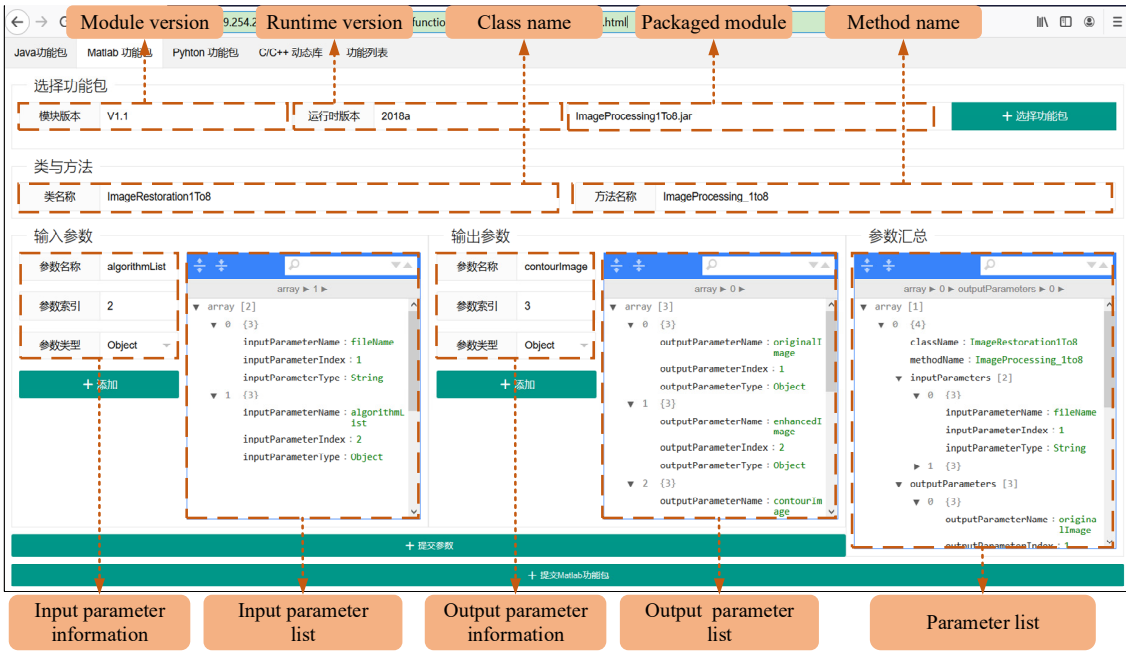

Figure 18. Submission of algorithmic and functional modules. Each parameter includes the name, index, and type of parameter. As the order of each parameter, the index is crucial.

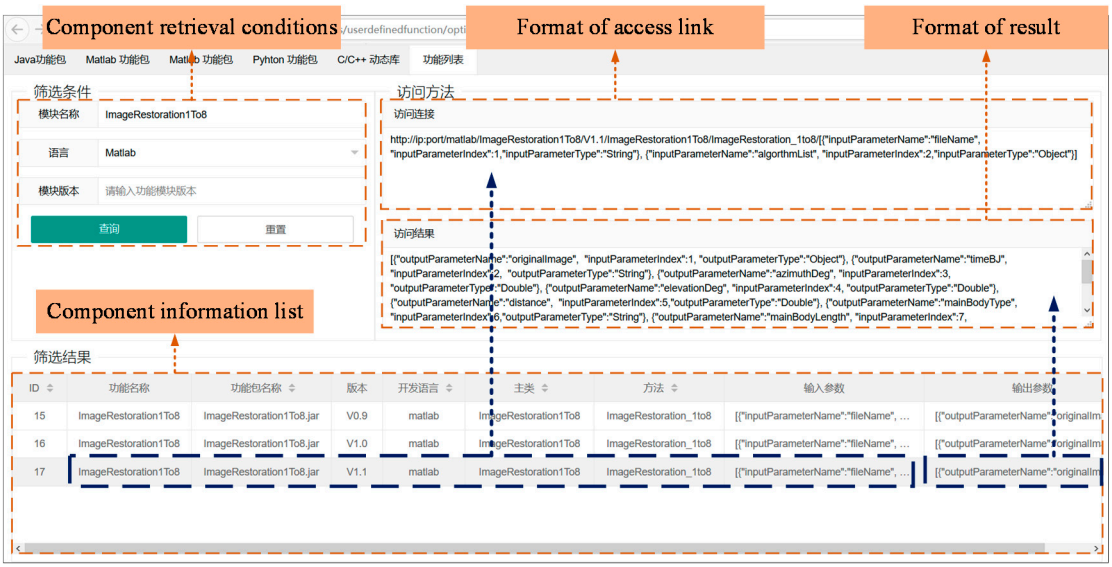

Figure 19. Module list and information retrieval. The format of the access link and the processing result of the selected algorithm can be obtained.

Based on the SOOC algorithmic module and the format of the access link and processing result, the SOOC processing web page shown in Figure 20 is constructed as a case. In the web page, the target image, 3D model, image list, and selected image are displayed. By selecting the processing algorithm from the algorithm list, the selected image will be processed in the functional server and the results of image enhancement and contour extraction will be displayed on the web page. The applications of various algorithms are not limited to the displayed case, which simply serves as a reference.

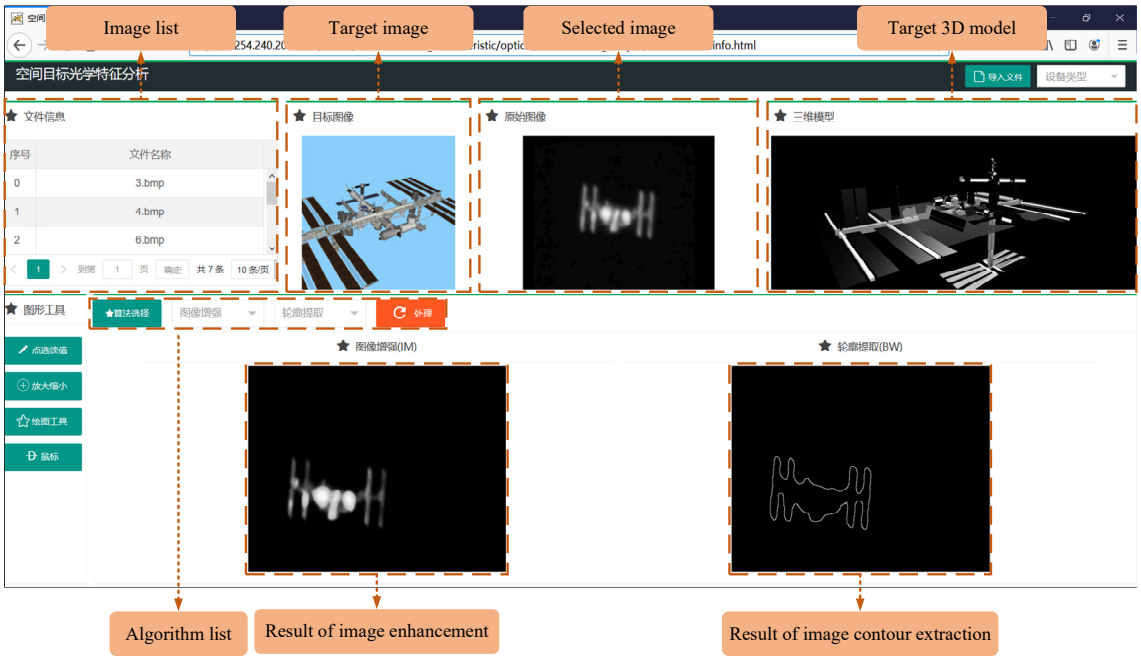

Figure 20. Web page of the algorithm level case. Based on the MATLAB algorithms of image enhancement and contour extraction, various online services can be proposed.

4.3. Development Level

Existing online services provide specific functions separately, and to integrate existing online services, advanced unified service integration is required. Using the service registry and service gateway in SOOCP, the development level can integrate services and provide unified management and standard formatted access links. In addition, based on load balancing, the service accesses in a high concurrency environment can be fulfilled well with satisfactory responses. At the same time, a unified workflow can be built by seamlessly connecting functions between different integrated services.

At the development level, the integration of online services into the platform requires the services to provide a health status check interface. The uniform resource locator (URL) of the health status check interface and the check result provided by the online service are shown in Table 3.

Table 3. URL of health check interface and check result.

Item	Content
Health status check URL	http://address/health.json

Result	{
	"status": "UP"
	}

An online service developed in C++ is considered for the case study. As shown in Figure 21, the online service provides the health check link, the result of the health status, and the server port. The service instance of the online service can be discovered in the service registry by integrating the health check link and the server port into the sidecar model. As seen in the configuration file of the service instance, the service has maintained the access link of the online service, which means that the heterogeneous online service has been integrated into the platform.

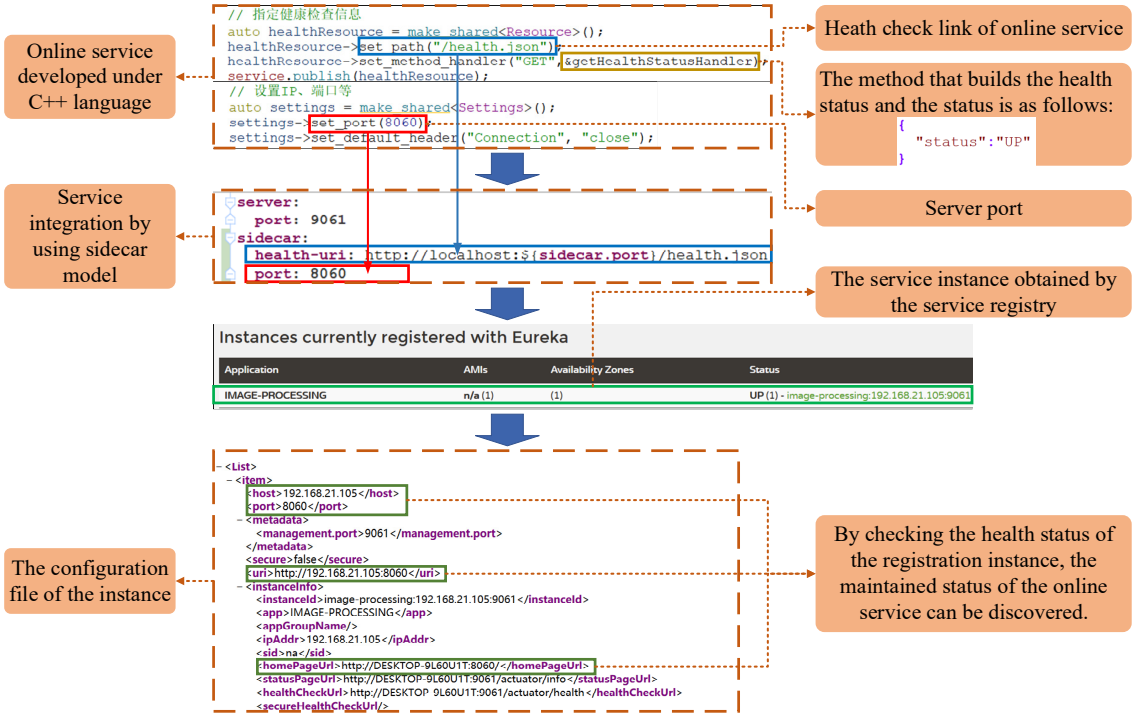

Figure 21. The integration of online services. Based on the unified access interface provided by the platform, users can use the instance of the online service through *Service Gateway*.

5. Conclusions and Future Work

Advances in technology have led to a rapid increase in SOOC data. In addition, they have resulted in higher requirements for the storage and application of SOOC data as well as for the innovation of algorithms and development of SOOC functions. To meet these diverse requirements, this paper proposes SOOCP, a microservice-based SOOC data and analysis platform. By exploiting the advantages of SQL and NoSQL, the hybrid SQL/NoSQL service provides appropriate data storage and retrieval services for different data storage and application scenarios. By comparing with related research, the hybrid SQL/NoSQL service can better meet the needs of research teams. Owing to the different processing methods of massive SOOC data, to promote algorithm application and research, the proposed platform uses the heterogeneous function integration service, which can seamlessly integrate and deploy algorithmic and functional modules built in different languages in the distributed environment and fuse existing online services on the basis of the sidecar model. The heterogeneous function integration service not only satisfies the requirements of algorithm testing and highly concurrent requests but also effectively promotes the latest research. To evaluate the platform for different levels of requirements in the field of SOOC, three cases were considered. The results confirmed the excellent performance of the proposed platform in terms of providing various data and function integration services for different levels of demand.

The platform proposed in this paper provides efficient data service and heterogeneous function integration service for users with different needs, which promotes the dissemination of innovative research achievements. However, as a scaffolding platform, the proposed platform requires further improvement. First, the current hybrid SQL/NoSQL service simply stores data that meet the predefined data formats, especially unstructured data. To better adapt to various requirements, the platform needs to build a capable service for user-defined data without predefined data formats. The users can autonomously create formats in the database and input the data, which is a major challenge. Second, the heterogeneous function integration service requires researchers to package the algorithms and components supported by the platform. The platform should provide the online packaging function based on SAAS, which will package the source code of the algorithms created by the researchers.

Third, the analysis of SOOC data is a complex process involving different aspects. In particular, data processing usually requires a combination of multiple modules. The platform should provide a complete workflow mode to realize the interconnection of various functional modules. Finally, the current platform mainly concentrates on the space object optical characteristic; the characteristics of other related fields, such as the electromagnetic scattering characteristic, electromagnetic radiation characteristic, and environmental characteristics, need to be considered in the future.

Under the rapid development of technologies, such as cloud computing and artificial intelligence, users need platforms that can meet more needs to realize innovation and promote the spread of technologies. Different industries have different requirements, and in the field of space object optical characteristic, in addition to improving the shortcomings of the platform, there are still several contents worthy of deep consideration and discussion for future development, such as improving the operation and computing efficiency of modules in the platform and introducing components and basic algorithms of artificial intelligence into the platform for users, which are all potential demands that the platform needs to consider for further requirements. We hope that this paper can promote the development and dissemination of the technologies of space object optical characteristic and provide some references for different application requirements.

Author Contributions: W.L. and Q.X. conceived and designed the research; W.L. and C.L. developed the platform and participated in data collection together; W.L. wrote the paper.

Funding: This research was funded by National Natural Science Foundation of China, grant number 41701463.

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. Meanwhile, the authors would like to thank the organizations and individuals who share the data for free. We would also like to thank the anonymous reviewers and editors for their constructive comments and suggestions.

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

References

1. Yusheng, J.; Shu, Y.; Hua, Z.; Xiaodan, X. Research on Target characteristic Database Synchronization Method Based on Thrift. *Procedia Comput. Sci.* **2019**, *147*, 542–549. [CrossRef]
2. Yi Ping, Z.; Chang Yin, Z.; Xiao Xiang, Z.; Yi Ding, P.; Chen, Z. Statistics and Analysis of LEO Objects' Luminosities. *Chin. Astron. Astrophys.* **2014**, *55*, 322–337. [CrossRef]
3. Xu, C.; Li, Z.; Zhang, F. A GEO Satellite Working State Detection Method Based on Photometric Characteristics. In Proceedings of the Optical Design and Testing VIII, Beijing, China, 11–13 October 2018; p. 1081511. [CrossRef]
4. Zhao Cheng, Y.; Lei, N.; Xiang, L. Design and Retrieval Method for All Attitude RCS Database of Radar Target. *Applic. Res. Comput.* **2009**, *26*. [CrossRef]
5. Dong Ning, L. Study on Photometric of Space Target. Ph.D. Thesis, University of Chinese Academy of Sciences, Changchun, China, 2015.
6. Jun, X. Creation and Application of the Natural Environment Database in Laser Semi-active Guided Simulation System. Master's Thesis, Xidian University, Xi'an, China, 2017.
7. Xue Mei, J. Database Construction of Missile & Satellite and the Abstract of Radar & Infrared Targets RCS Characteristic. Master's Thesis, Lanzhou University, Lanzhou, China, 2008.
8. Riddle, R.L.; Burse, M.P.; Law, N.M.; Tendulkar, S.P.; Baranec, C.; Rudy, A.R.; Sitt, M.; Arya, A.; Papadopoulos, A.; Ramaprakash, A. The Robo-AO Software: Fully Autonomous Operation of a Laser Guide Star Adaptive Optics and Science System. In Proceedings of the Adaptive Optics Systems III, Amsterdam, The Netherlands, 1–6 July 2012; p. 84472O. [CrossRef]
9. Samoylova, L.; Buzmakov, A.; Geloni, G.; Chubar, O.; Sinn, H. Cross-platform Wave Optics Software for XFEL Applications. In Proceedings of the Advances in Computational Methods for X-Ray Optics II, San Diego, CA, USA, 21–24 August 2011; p. 81410A. [CrossRef]
10. Oracle Database Documentation. Available online: https://docs.oracle.com/en/database/oracle/oracle-database/index.html (accessed on 13 September 2019).

11. SQL Server Documentation. Available online: https://docs.microsoft.com/en-us/sql/sql-server/sql-server-technical-documentation?view=sql-server-2017 (accessed on 13 September 2019).
12. MySQL 8.0 Reference Manual. Available online: https://dev.mysql.com/doc/refman/8.0/en/ (accessed on 13 September 2019).
13. Becla, J.; Hanushevsky, A.; Nikolaev, S.; Abdulla, G.; Szalay, A.; Nietosantisteban, M.; Thakar, A.; Gray, J. Designing a Multi-petabyte Database for LSST. In Proceedings of the SPIE—The International Society for Optical Engineering, Orlando, FL, USA, 24–31 May 2006. [CrossRef]
14. Jain, V.; Upadhyay, A. MongoDB and NoSQL Databases. *IJCA* **2017**, *167*, 16–20. [CrossRef]
15. Yanqi, W.; Yusheng, J.; Xiaodan, X. Research of Target Characteristics Storage Based on RDBMS and Hadoop. In Proceedings of the 2016 International Conference on Identification, Information and Knowledge in the Internet of Things (IIKI), Beijing, China, 20–21 October 2016; pp. 417–420. [CrossRef]
16. Adaptive Optics Softwares & Real Time Computer. Available online: https://www.alpao.com/adaptive-optics/ao-softwares.html (accessed on 21 May 2019).
17. Webster, S.; Miller, G.; Mayott, G. Software as a Service Approach to Sensor Simulation Software Deployment. In Proceedings of the Modeling and Simulation for Defense Systems and Applications VII, Baltimore, MD, USA, 24 April 2012; p. 84030I. [CrossRef]
18. Application-Specific Optical Design. Available online: https://www.synopsys.com/content/dam/synopsys/optical-solutions/documents/whitepapers/application-specific-design.pdf (accessed on 21 May 2019).
19. Villamizar, M.; Garcés, O.; Castro, H.; Verano, M.; Salamanca, L.; Casallas, R.; Gil, S. Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud. In Proceedings of the Computing Colombian Conference, Bogotá, Colombia, 21–25 September 2015. [CrossRef]
20. Lyu, L.; Xu, Q.; Lan, C.; Shi, Q.; Lu, W.; Zhou, Y.; Zhao, Y. Sino-InSpace: A Digital Simulation Platform for Virtual Space Environments. *Isprs. Int. J. Geoinf.* **2018**, *7*, 373. [CrossRef]
21. Yue, S.; Chen, M.; Wen, Y.; Lu, G. Service-oriented model-encapsulation strategy for sharing and integrating heterogeneous geo-analysis models in an open web environment. *Isprs. J. Photogramm.* **2016**, *114*, 258–273. [CrossRef]
22. Baru, C.; Chandra, S.; Lin, K.; Memon, A.; Youn, C. The GEON service-oriented architecture for Earth Science applications. *Int. J. Digit. Earth* **2009**, *2*, 62–78. [CrossRef]
23. Zhang, L.; Ma, D.; Niu, C. Simulation and Analysis of ACB'S Photometric Signature Based on STK. In Proceedings of the 5th International Symposium on Advanced Optical Manufacturing and Testing Technologies: Optoelectronic Materials and Devices for Detector, Imager, Display, and Energy Conversion Technology, Dalian, China, 26–29 April 2019; p. 76585H. [CrossRef]
24. Microservices a Definition of This New Architectural Term. Available online: https://martinfowler.com/articles/microservices.html (accessed on 28 April 2019).
25. Alshuqayran, N.; Ali, N.; Evans, R. A Systematic Mapping Study in Microservice Architecture. In Proceedings of the 2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA), Macau, China, 4–6 November 2016; pp. 44–51. [CrossRef]
26. Di Francesco, P.; Lago, P.; Malavolta, I. Architecting with microservices: A systematic mapping study. *J. Syst. Softw.* **2019**, *150*, 77–97. [CrossRef]
27. Pahl, C.; Jamshidi, P. Microservices: A Systematic Mapping Study. In Proceedings of the International Conference on Cloud Computing & Services Science, Rome, Italy, 23–25 April 2016. [CrossRef]
28. Baškarada, S.; Nguyen, V.; Koronios, A. Architecting Microservices: Practical Opportunities and Challenges. *J. Comput. Inform. Syst.* **2018**, 1–9. [CrossRef]
29. Dragoni, N.; Lanese, I.; Larsen, S.T.; Mazzara, M.; Mustafin, R.; Safina, L. Microservices: How to make your application scale. In Proceedings of the International Andrei Ershov Memorial Conference on Perspectives of System Informatics, Moscow, Russia, 26–29 June 2017; pp. 95–104. [CrossRef]
30. Mazzetti, P.; Nativi, S.; Caron, J. RESTful implementation of geospatial services for Earth and Space Science applications. *Int. J. Digit. Earth* **2009**, *2*, 40–61. [CrossRef]
31. Pokorny, J. NoSQL databases: A step to database scalability in web environment. *Int. J. Web Inf. Syst.* **2013**, *9*, 69–82. [CrossRef]
32. Chen, J.-K.; Lee, W.-Z. An Introduction of NoSQL Databases Based on Their Categories and Application Industries. *Algorithms* **2019**, *12*, 106. [CrossRef]

33. Hecht, R.; Jablonski, S. NoSQL evaluation: A use case oriented survey. In Proceedings of the International Conference on Cloud & Service Computing, Hong Kong, China, 12–14 December 2011.
34. Khazaei, H.; Fokaefs, M.; Zareian, S.; Beigi-Mohammadi, N.; Ramprasad, B.; Shtern, M.; Gaikwad, P.; Litoiu, M. How Do I Choose the Right Nosql Solution? a Comprehensive Theoretical and Experimental Survey. *Big Data Inf. Analyt.* **2017**, *1*, 185–216.
35. The MongoDB 4.0 Manual. Available online: https://docs.mongodb.com/v4.0/ (accessed on 13 September 2019).
36. Redis Documentation. Available online: https://redis.io/documentation (accessed on 13 September 2019).
37. Spring Cloud Netflix. Available online: https://cloud.spring.io/spring-cloud-netflix/multi/multi_spring-cloud-netflix.html (accessed on 21 May 2019).
38. Space-Track.org. Available online: https://www.space-track.org (accessed on 19 May 2019).
39. User's Manual. Available online: http://jmeter.apache.org/usermanual/index.html (accessed on 13 September 2019).
40. Li, D.; Mei, H.; Shen, Y.; Su, S.; Zhang, W.; Wang, J.; Zu, M.; Chen, W. ECharts: A declarative framework for rapid construction of web-based visualization. *Vis. Inf.* **2018**, *2*, 136–146. [CrossRef]
41. The International Space Station Through My Telescope. Available online: https://www.youtube.com/watch?v=me_fbGVuwy8 (accessed on 21 May 2019).
42. Otsu, N. A Threshold Selection Method from Gray-Level Histograms. *IEEE Trans. Syst. Man Cybern.* **1979**, *9*, 62–66. [CrossRef]