International Journal of Geo-Information

MDPI

Article

A Containerized Service-Based Integration Framework for Heterogeneous-Geospatial-Analysis Models

Lilu Zhu [1,*], Yang Wang [1], Yunbo Kong [2], Yanfeng Hu [3] and Kai Huang [1]

[1] Suzhou Aerospace Information Research Institute, Suzhou 215123, China; wangyang003839@aircas.ac.cn (Y.W.)
[2] Beijing Satellite Navigation Center, Beijing 100085, China
[3] Aerospace Information Research Institute, Chinese Academy of Sciences, Beijing 100094, China
* Correspondence: zhull@aircas.ac.cn

Abstract: The integration of geospatial-analysis models is crucial for simulating complex geographic processes and phenomena. However, compared to non-geospatial models and traditional geospatial models, geospatial-analysis models face more challenges owing to extensive geographic data processing and complex computations involved. One core issue is how to eliminate model heterogeneity to facilitate model combination and capability integration. In this study, we propose a containerized service-based integration framework named GeoCSIF, specifically designed for heterogeneous-geospatial-analysis models. Firstly, by designing the model-servicized structure, we shield the heterogeneity of model structures so that different types of geospatial-analysis models can be effectively described and integrated based on standardized constraints. Then, to tackle the heterogeneity in model dependencies, we devise a prioritization-based orchestration method, facilitating optimized combinations of large-scale geospatial-analysis models. Lastly, considering the heterogeneity in execution modes, we design a heuristic scheduling method that establishes optimal mappings between models and underlying computational resources, enhancing both model stability and service performance. To validate the effectiveness and progressiveness of GeoCSIF, a prototype system was developed, and its integration process for flood disaster models was compared with mainstream methods. Experimental results indicate that GeoCSIF possesses superior performance in model management and service efficiency.

Keywords: geospatial-analysis models; model integration framework; model-servicized structure; prioritization-based orchestration; heuristic scheduling

Citation: Zhu, L.; Wang, Y.; Kong, Y.; Hu, Y.; Huang, K. A Containerized Service-Based Integration Framework for Heterogeneous-Geospatial-Analysis Models. *ISPRS Int. J. Geo-Inf.* **2024**, *13*, 28. https://doi.org/10.3390/ijgi13010028

Academic Editors: Wolfgang Kainz and Huayi Wu

Received: 17 November 2023
Revised: 1 January 2024
Accepted: 9 January 2024
Published: 12 January 2024

1. Introduction

1.1. Research Background

Geospatial-analysis models, based on principles of geography, utilize mathematical, statistical, as well as computer-science methods to simulate geographic processes and phenomena [1,2]. In recent decades, geospatial-analysis models have played an important role in urban planning, environmental protection, and natural disaster risk assessment to help decision makers solve geography-related problems. In urban planning, geospatial-analysis models can help planners gain a more scientific and comprehensive understanding of the city's development trends by collecting, processing, and analyzing a large scale of geographic data [3]. In environmental protection, geospatial-analysis models can analyze the distribution and propagation pathways of air-pollution sources. They can also predict the balance between supply and demand of water resources, and optimize the management strategies of natural reserve areas [4]. In natural-disaster risk assessment, geospatial-analysis models can quantitatively analyze the spatial distribution and impact the range of natural risks [5,6]. In recent years, advances in Earth observation technology have provided a more convenient and efficient channel for obtaining high-resolution geographic data.

However, the traditional standalone models have limitations when it comes to large-scale geospatial-analysis tasks involving multiple factors and multidimensional analysis [7]. In such scenarios, adopting integration approaches using geospatial-analysis models can effectively leverage the strengths of different models. By integrating and synergizing these models, data fusion and information sharing across domains are facilitated, leading to improved responses to complex geographical problems and providing decision makers with comprehensive and multi-perspective decision support.

1.2. Issues and Challenges

1.2.1. Characteristics of Geospatial-Analysis Models

Compared to non-geospatial models and traditional geospatial models, geospatial-analysis models have distinct characteristics and face unique challenges when it comes to integration. Non-geospatial models primarily focus on data analysis and modeling techniques, with less consideration of geographic features and spatial relationships [8]. These models can be categorized into various domains such as economics, social sciences, biology, and health, aiming to mine patterns and association rules from non-geographic data to reveal inherent laws and trends in their respective fields [9,10]. Since models within the same domain often share consistent data types and processing methods [11], integrating non-geospatial models is relatively straightforward. On the other hand, traditional geospatial models emphasize the representation of geographic facts and phenomena, providing an intuitive understanding of geospatial data [12]. They are typically categorized into point models, line models, and area models based on the types of geographic entities they represent [13]. These models present geographic data through maps, charts, and graphs, facilitating comprehension and analysis of geographic phenomena, relationships, and processes [14]. The complexity of these models is relatively minimal.

In contrast, geospatial-analysis models prioritize the discovery of underlying patterns and mechanisms behind geographic phenomena [15]. Depending on specific application objectives, geospatial-analysis models can be categorized into geospatial-decision-support models, geospatial-risk-assessment models, geospatial-environmental-simulation models, etc. [16,17]. These models utilize geospatial data and spatial relationships, employing interdisciplinary knowledge such as mathematics, statistics, and computer science to simulate and analyze geospatial processes and phenomena [18]. This involves extensive geographic-data processing and complex computations. One core challenge is of how to address model heterogeneity to facilitate model combination and capability integration.

1.2.2. Advances of Geospatial-Analysis Models

From the perspective of system integration, geospatial-analysis-model structures can be divided into monomer, component, and dependent structures [19–21]. The monomer structure integrates all functions and data from the geographic-information system (GIS). It is simple to develop and convenient to deploy, but inevitably produces significant functional redundancy [22]. The component structure develops the functions of geospatial-analysis models as plugins and couples them in a pluggable manner to solve the model-extension problem. This structure can effectively achieve functional openness and system scalability [23], but does not support cross-platform heterogeneous environments. The dependency structure develops the functions of geospatial-analysis models as micro-services and couples them in a lightweight manner [24], thus addressing the cross-platform barrier of the component structure. However, none of these model structures can be applied directly to heterogeneous-geospatial-analysis models. This will increase the complexity of model-structure representation owing to the superiority of heterogeneous-geospatial-analysis applications over modeling dimensionality.

According to different coupling methods, geospatial-analysis-model integration can be categorized into nested, modular, model library, and service-based integrations [25]. Nested integration includes the source code and function library in terms of implementation [26]. In this method, geospatial-analysis models are used as functional items in GIS. It enables

seamless integration of models with GIS, but model heterogeneity results in significant integration costs. Modular integration includes both executable and middleware [27,28]. In this method, GIS and geospatial-analysis models exist independently, and are invoked with pre-defined modules or interface protocols. This method is relatively simple and requires a low-cost development. Nevertheless, the communication costs between systems increase owing to the requirement for additional agreements on particular interaction methods. In addition, the same type of invoke interface or communication method is not universal, particularly when the model is updated and changed, and the secondary loading efficiency is reduced. Because homogeneous models in GIS have difficulty meeting the demands of complex research objectives, a combination of multisource models is required to solve practical problems. This led to an evolution in model organization from single packages to the model library. However, with the increase in model diversity and invocation complexity, the traditional model library has limitations in terms of distributed integration, process control, and function sharing. Service-based integration designs models as accessible microservices that integrate functionality through invocations and combinations. For example, Wen et al. proposed an open environment architecture to support the sharing of geographic models in cloud environments [29]. Qiao et al. integrated the container isolation mechanism into OpenGMS to eliminate the barrier of environmental heterogeneity for multisource model integration [30].

It is noteworthy that although the research on geospatial-analysis-model integration started earlier, most of them suffer from the problems of online complexity, high maintenance costs, and insufficient flexibility. Traditional methods are hardly applicable, especially in the face of geospatial-analysis models with heterogeneity in structures, dependencies, and running modes. Continuous improvements are required to accommodate the model reliability and scalability requirements.

1.2.3. Challenges of Integrating Geospatial-Analysis Models

Early integration methods of nested, modular, and model libraries have been successfully applied to some geographic-problem-analysis and -solution systems [25]. Nevertheless, the challenges presented by heterogeneity impede the collaboration and interoperability of models [31,32], restricting their deployment solely to independent servers or closed networks [29]. As a result, this impedes the seamless integration and comprehensive fusion of information on a large scale.

We find that the heterogeneities of geospatial-analysis models are reflected in three main aspects: (1) Heterogeneity in model structure: Geospatial-analysis models employ different modeling methods, data structures, interface specifications, and programming languages [7,33,34]. This diversity makes it difficult to provide a unified description and integration of the models [35], thus increasing barriers to model interoperability. (2) Heterogeneity in model dependencies: As the complexity of geospatial businesses grows, the application systems supporting them expand as well [36]. For example, in geospatial applications like smart cities [37], there may be hundreds or even thousands of small functional units of geospatial-analysis models. In such cases, sharing, discovering, and collaborating on geospatial-analysis models among different systems or platforms becomes particularly complex [38]. (3) Heterogeneity in model-execution modes: Geospatial-analysis models involve diverse business processes that typically rely on different hardware architectures, software platforms, and program dependencies [39,40]. Due to potential conflicts between operating systems or program libraries, traditional approaches struggle to deploy different models on the same server [30]. Furthermore, geospatial-analysis models require extensive processing of geographic data and complex analytical computations [41], placing higher demands on resource allocation and scheduling strategies.

1.3. Contributions

To address the aforementioned challenges of heterogeneity, we conducted thorough research and exploration, which led to the development of the GeoCSIF framework, specifi-

cally designed for the integration of large-scale heterogeneous-geospatial-analysis models. The originality of the framework is as follows:

(1) The model encapsulation component designs the model-servicized structure for the characteristics of model structural heterogeneity. With this model-servicized structure, diverse types of geospatial-analysis models can be effectively described and integrated based on standardized constraints. This approach ultimately enhances the interoperability and reusability of models across different systems and platforms.
(2) The model orchestration component designs a prioritization-based orchestration method for the characteristics of model-dependency heterogeneity. The approach prioritizes and optimizes resource discovery based on model relationships, service performance, and runtime feedback. This enables optimal combination and capability integration of large-scale heterogeneous-geospatial-analysis models.
(3) The model publication component designs a heuristic scheduling method for the characteristics of the execution-mode heterogeneity. This method utilizes containerization technology to isolate the execution of different geospatial-analysis models, thereby mitigating the effects of heterogeneous runtime environments and accommodating diverse execution modes. Additionally, it establishes optimal mapping between models and underlying computational resources, enhancing the adaptability of models to cloud environments, while improving their stability and service performance.

To validate the effectiveness and progressiveness of GeoCSIF, we also developed a prototype system. We chose heterogeneous-flood-disaster models as our research case study due to their complex dependency relationships, diverse underlying resource requirements, and heterogeneous operating environments. Comparing the integration process of the GeoCSIF with mainstream methods, our evaluation focused on the effectiveness, servicized, orchestration, and scheduling performance of the integration framework. Experimental results indicate that GeoCSIF possesses superior performance in model management and service efficiency, showcasing its strong performance in handling complex models and ensuring effective integration and utilization of resources.

1.4. Paper Organization

The rest of the current paper is organized as follows: Section 2 designs the overall framework of GeoCSIF. Component design is presented and described in Section 3 including the servicized encapsulation, prioritization-based orchestration, and adaptive publication. Section 4 presents the prototype system implementation. A case study and result analysis are carried out in Section 5. Finally, conclusions and some prospects are presented in Section 6.

2. Design of the Framework

In this section, we describe the design of a containerized service-based integration framework named GeoCSIF, as shown in Figure 1. GeoCSIF consists of three types of components: model encapsulation, orchestration, and publication. GeoCSIF can reduce the complexity of integrating and publishing heterogeneous-geospatial-analysis models as reusable model services by establishing a standardized process.

First, the model encapsulation component constructs the raw model (including executable files, data, configurations, and runtime environment) as a unified-model-service package. The model-service package is then registered to the service-package repository for centralized management. The model orchestration component obtains the model-service package from the service-package repository to assemble model dependencies. Finally, the model publishing component publishes the model to the container-based cloud environment. In particular, we accordingly designed the templated constraint method $F(\cdot)$, prioritization-based orchestration method $S(\cdot)$, and heuristic scheduling method $G(\cdot)$ in the Encapsulator, Orchestrator, and Publisher of these components. These methods can significantly reduce the integration complexity of heterogeneous-geospatial-analysis models. Among them, $F(\cdot)$ provides fine-grained decoupling and encapsulation of the model

structure. It enables models to be flexibly assembled and extended. $S(\cdot)$ can discover the optimal model-dependent resources (models, data, etc.). This significantly reduces the time and cost of dependency orchestration. $G(\cdot)$ seeks the optimal scheduling node of the models. It balances the underlying resource allocation and improves the model-service performance.

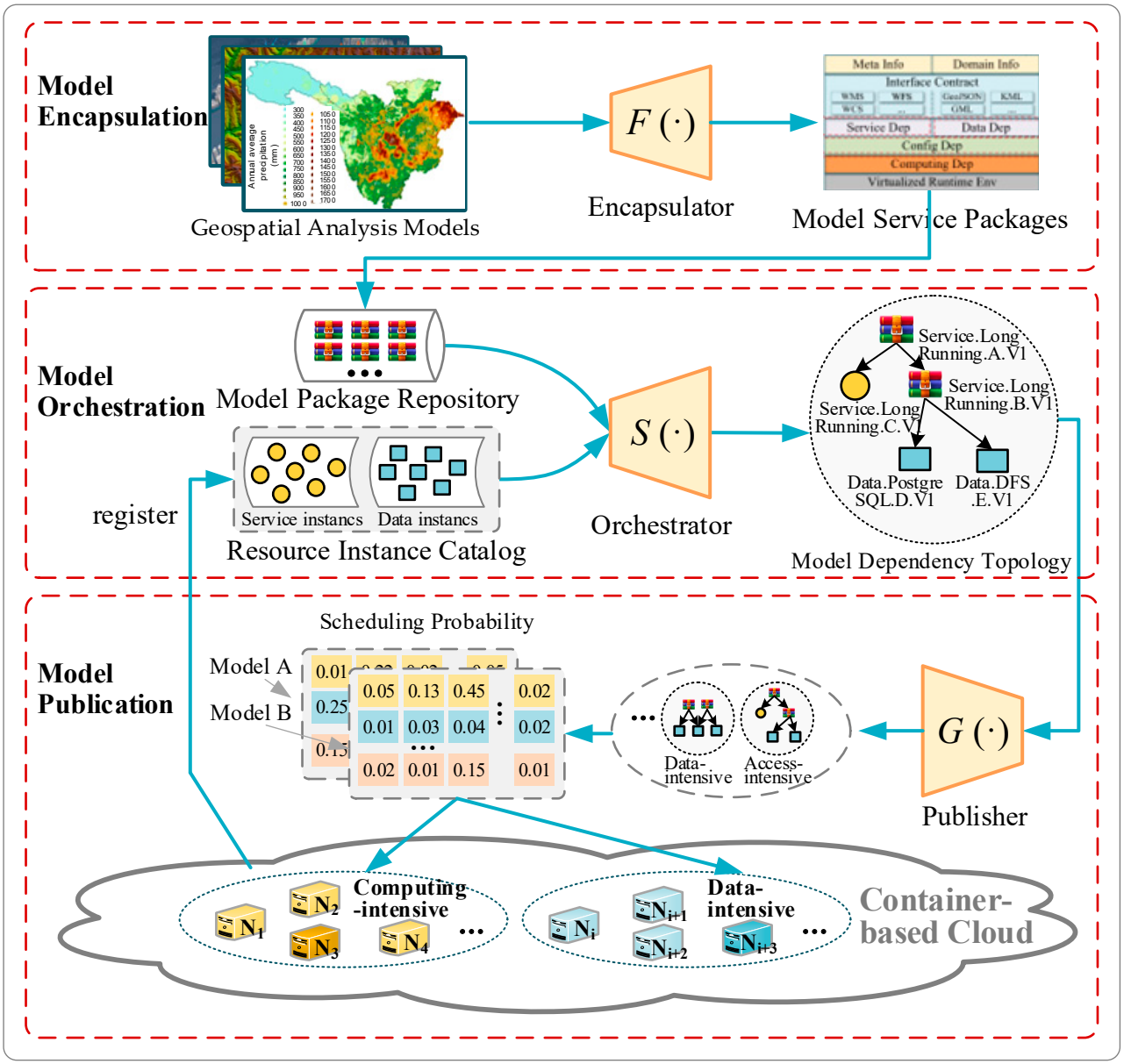


Figure 1. Overview of containerized service-based integration framework for heterogeneous-geospatial-analysis models (GeoCSIF).

3. Design of the Component

3.1. Servicized Encapsulation of Geospatial-Analysis Models

In the model encapsulation component, we first design a cloud-oriented model-servicized structure. It can effectively shield the model structural heterogeneity in terms of the development language, engineering architecture, and runtime environment. Then, based on the model-servicized structure, we further propose a templated constraint method $F(\cdot)$ to decouple the raw model into declarative service, data, configuration, and environment templates. Finally, container virtualization technology is employed to encapsulate these template files and model business images as a unified-model-service package.

This enables rapid publishing of geospatial-analysis models in the container-based cloud environments.

3.1.1. Servicized Declaration of Geospatial-Analysis Models

Publishing and sharing geospatial-analysis models usually require significant work, mainly owing to their complexity and disciplinary details [25]. On the one hand, geospatial-analysis models vary widely in terms of discipline fields, modeling approaches, and data organization, making it difficult to represent them in a fixed model. On the other hand, because these models depend on different hardware architectures, software platforms and programming languages, it is difficult to share them as unified services. Therefore, structural heterogeneity must be eliminated to support open sharing and effective execution of geospatial-analysis models. To address this issue, we propose a cloud-oriented model-servicized structure, as shown in Figure 2.

This model-servicized structure consists of model meta-information, model domain information, interface contracts, resource dependencies, and a virtualized runtime environment. Model meta-information is the basic information of geospatial-analysis models, such as name, creator, and description of the model. Model domain information refers to the business-capability information of the model, such as the reference system, business type, and product level. Interface contracts are interface standards or protocols used to describe and define the interfaces that geospatial-analysis models follow when exchanging and sharing data between different systems, software, or platforms. It includes OGC specifications, data-format standards such as Shapefile, GeoJSON, KML, as well as customized interface protocols. Resource dependencies are descriptions of the resource information required for running a model, including model services (one model may depend on multiple other models), data storages, configuration files, and computing resources. A virtualized runtime environment is a unified encapsulation of the operating system, program-dependent libraries, and other runtimes. By using virtualization technology to shield the operating-system and dependency-library conflicts, different models can run on the same servers and model interoperability becomes more accessible. Overall, the model-servicized structure provides a fine-grained decoupling of the geospatial-analysis models, making the model integration and sharing more convenient and efficient.

We adopted declarative template files to describe the model-servicized structure, including the service, data, configuration, and environment templates. The syntax format of these templates can be JSON, XML, or YAML, and an example of the XML format is shown in Figure 2. The service template is a formatted description of the model meta-information, business information, interface contracts, and its dependency resources. The data template is a formatted description of the model-dependent storages, including the storage type, capacity, and access mode. The configuration template is a formatted description of model configurations. We abstract the raw configuration files into configuration templates, by adjusting the values of the configuration items, model configuration files applicable to different business scenarios can be produced. The environment template is a declaration of the model runtime environment, including operating system (OS), CPU architecture, and program dependency libraries. By separating the model runtime environment from business logic and dynamically assembling them at runtime, it not only reduces the volume of the model-service package, but also improves the reusability of the runtime environment.

3.1.2. Standardized Constraints of Model Services

As mentioned earlier, we describe the model-serviced structure through declarative templates. However, owing to the wide range of sources of geospatial-analysis models, the lack of uniform template field constraints may lead to difficulties in model assembly. To this end, we propose a templated constraint method $F(\cdot)$ with multiple constraint rules to ensure that the models can be assembled correctly. On top with this, the models are uniformly encapsulated through a standardized-service-package structure.

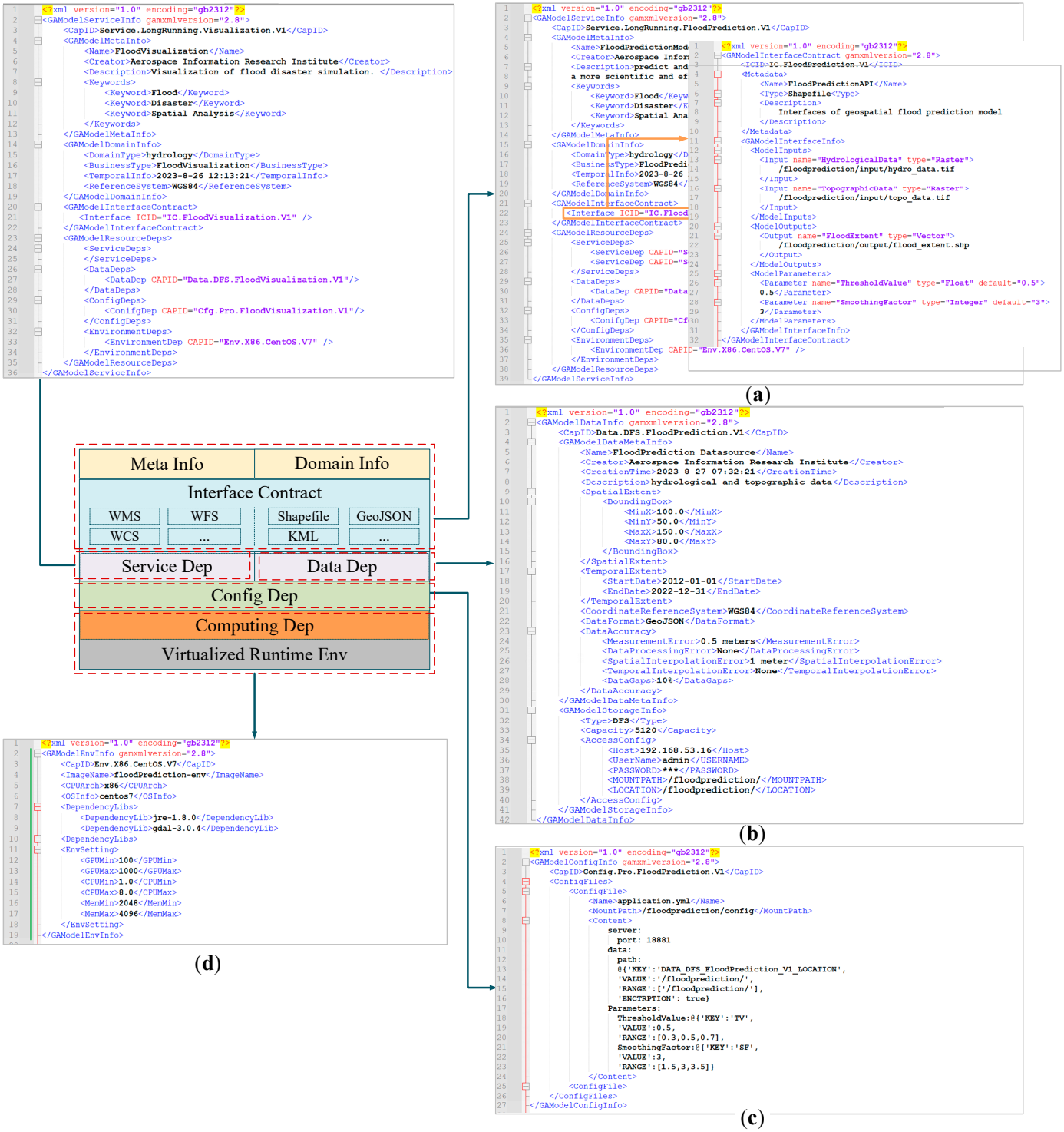


Figure 2. Model-serviced structure with declarative templates: (**a**) service template; (**b**) data template; (**c**) configuration template; and (**d**) environment template.

As shown in Figure 3, the templated constraint method is divided into resource classification, rules definition, and hybrid constraint rule processing. The method dynamically matches constraint rules based on the types of geospatial-analysis models and their dependent resources to construct declarative templates. The detailed steps are as follows:

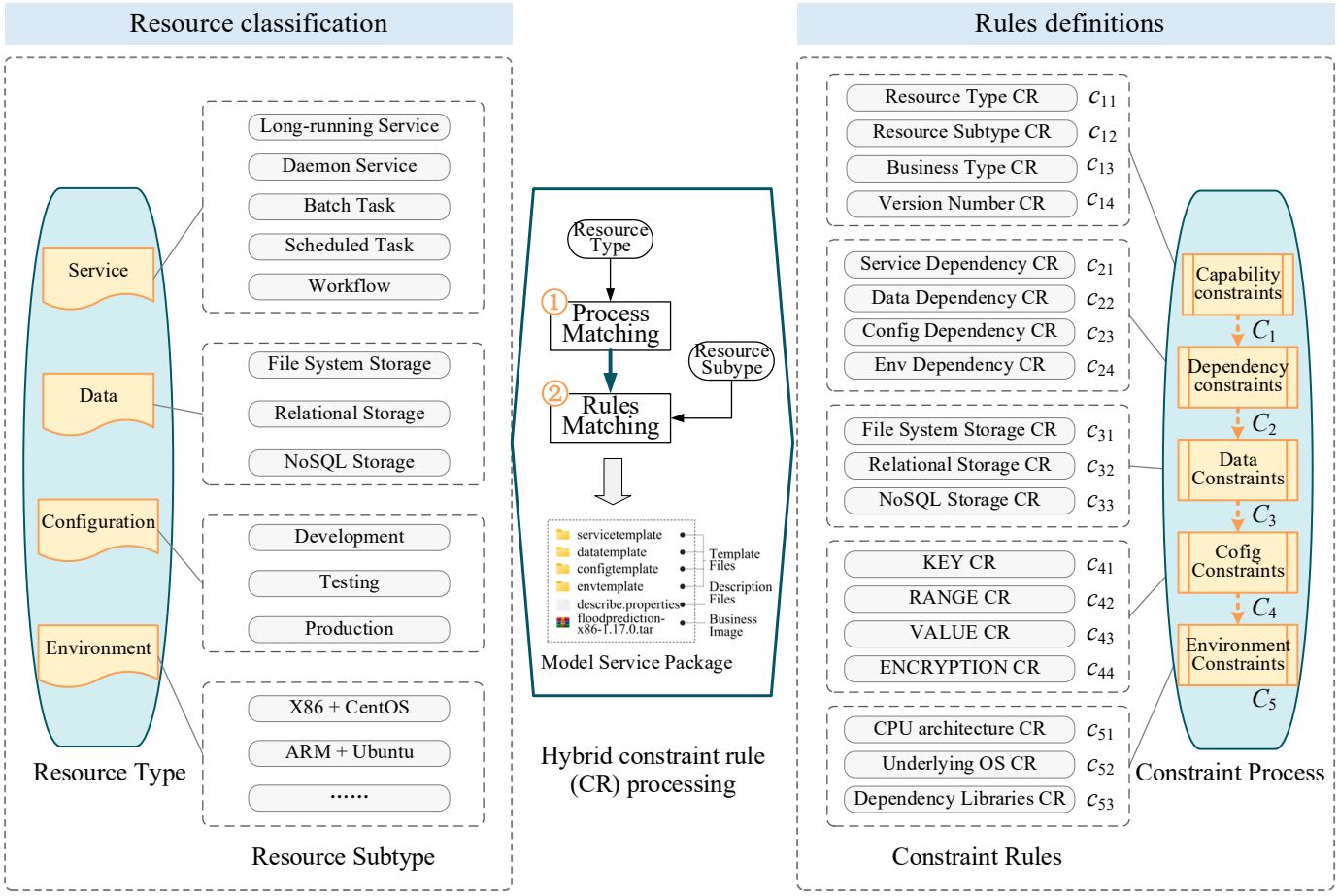


Figure 3. Principle of templated constraint method $F(\cdot)$.

Resource classification. As shown in Figure 3 (left), we categorized the composition of geospatial-analysis model into service, data, configuration, and environment, to facilitate resource discovery and assembly. Based on the running mode, service resources are subdivided into long-running services, daemon services, batch tasks, scheduled tasks, and workflows. Based on the storage type, data resources are subdivided into distributed file system DFS; relational storage, such as MySQL, PostgreSQL and DM; and NoSQL storage, such as HBase. Configuration resources are subdivided into development, testing, and production based on the application scenarios. Runtime environment resources are subdivided into different combinations of CPU architecture and the operating system (OS).

Rule definitions. As shown in Figure 3 (right), we define five types of constraints C_1–C_5 to normalize declarative template fields. They are subdivided into multiple constraint rules (CR) described using regular expressions. Among them, C_1 employs a capability identifier (CapID) to describe a resource's business capability. CapID is divided into a four-field structure, with resource type, resource subtype, business type, and version number in order from left to right, such as "Service.LongRunning.FloodPrediction.V1". The values of the CapID field are specifically constrained using c_{11}–c_{14}. Note that CapID is the basis for resource discovery and assembly. Dependencies between resources are established through the CapID of resources. C_2 establishes associations between the models and dependent resources. c_{21}–c_{24} are used to query and reference the CapID of the model's dependent resources. C_3 verifies the model requirements for the storage resources, where c_{31} verifies that the storage capacity requested by the model does not exceed the storage-system quota, and c_{32} and c_{33} primarily verify the availability of storage access addresses. C_4 verifies the compliance of the configuration template. Configuration templates are generated by inserting special placeholders into the raw configuration files

of geospatial-analysis models. This special placeholder is in the form of @{KEY, VALUE, RANGE, ENCRYPTION}, which is used to identify configuration items that are open for modification. By constraining the values of configuration items, it prevents business disasters caused by incorrect modifications. Among them, c_{41} and c_{42} verify the names and values of configuration items. c_{43} indicates the range of values. c_{44} identifies whether the sensitive configuration items are encrypted. C_5 represents the runtime environment of the model. c_{51}–c_{53} describe the CPU architecture, operating system, and program-dependent libraries, respectively, to ensure that the underlying environment can provide the appropriate running support.

Hybrid constraint rule processing. As shown in Figure 3 (middle), we establish a mapping between the resource types and constraint rules, which in turn regularizes the declarative templates. First, the overall servicized constraint process is constructed as $C_1 \rightarrow C_2 \rightarrow C_3 \rightarrow C_4 \rightarrow C_5$. The specific constraint rules are matched based on the resource subtypes. For all geospatial-analysis models, c_{11}–c_{14} are used to establish CapID according to the resource type, resource subtype, business type, and version number. c_{21}–c_{24} are employed to query and reference the CapID of model-dependent resources. In particular, these capability identifiers are accordingly injected into the "ServiceDeps", "DataDeps", "ConfigDeps", and "EnvironmentDeps" fields of the service template, as show in Figure 2a. For models with data dependencies, constraint rules c_{31}–c_{33} are matched based on the subtypes of the dependent storage. The requested capacity and access address of the dependent storage are injected into the fields "Capacity" and "AccessConfig" of the data template, as show in Figure 2b. For models with configuration dependencies, c_{41}–c_{44} are used to verify the compliance of the configuration template, as show in Figure 2c. We must ensure that the values of the configuration items are in accordance with the range specification. For all models, c_{51}–c_{53} were employed to construct virtualized runtime environments, which are described as environment templates, as show in Figure 2d. Finally, executable files and launch scripts of the raw model are constructed as business images based on the virtualization technology. These declarative template files, along with business images are further encapsulated into unified service packages that can be quickly published in cloud environments.

3.2. Prioritization-Based Orchestration of Geospatial-Analysis Models

Model orchestration refers to the assembly of models, data, configurations, and runtime environment of the current model according to the dependencies between model-service packages. The open sharing of geospatial-analysis models significantly improves the efficiency of model integration and publication. However, owing to the large number of resources, wide sources and different structures, the discovery of high-quality resources is extremely challenging. In this section, we design a prioritization-based orchestration method $S(\cdot)$, which consists of the pre-selection sub-method $s_1(\cdot)$ and priority-selection sub-method $s_2(\cdot)$. The method can proactively discover optimal resources, thus significantly reducing labor costs.

3.2.1. Priority Selection of Dependent Resources

Most early orchestration methods employed a weighted sum of QoS attributes to evaluate and screen resources. These methods are simple and feasible but ignore the relationship between multi-dimensional attributes and the impact of attribute interactions on resource discovery. This may result in higher rankings for resources with a single better attribute. To this end, we propose an improved two-stage prioritization-based orchestration method $S(\cdot)$, as shown Figure 4.

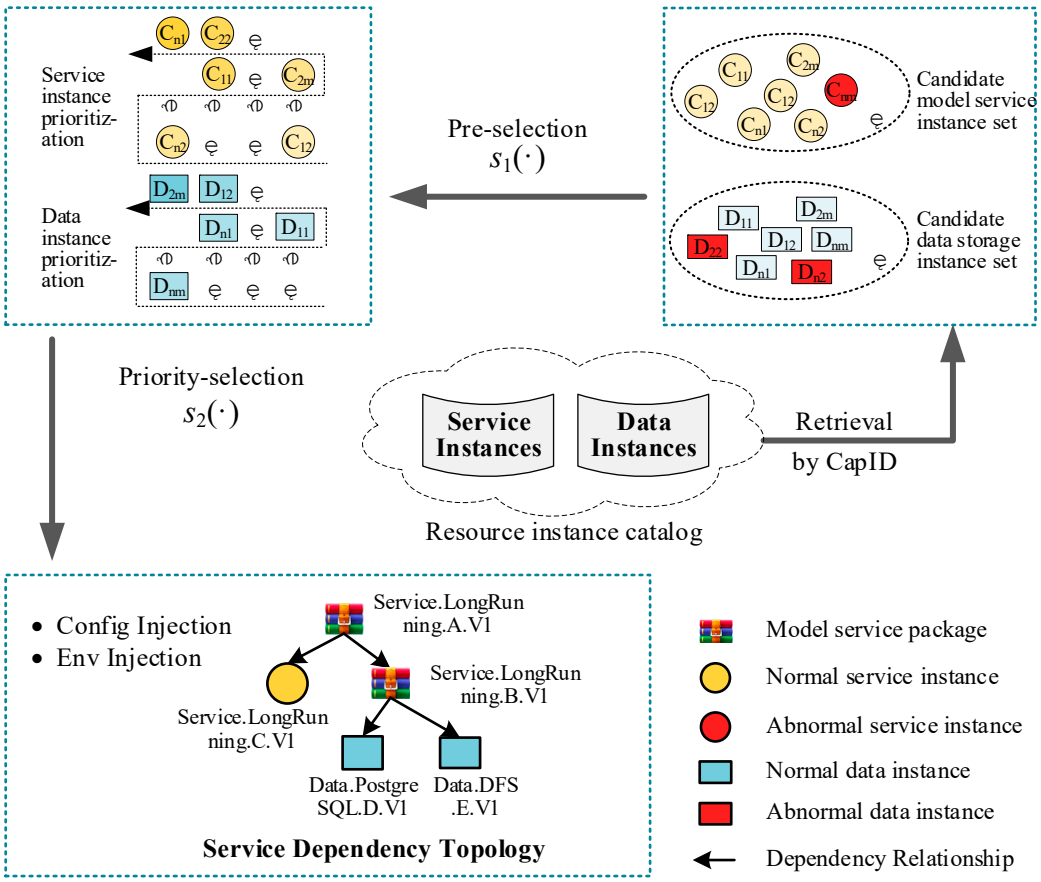


Figure 4. Flow of prioritization-based orchestration method $S(\cdot)$.

The proposed method initially retrieves resources (models, data storages) from the resource instance catalog using the CapID of dependent resources. The retrieval results are marked as candidate resource sets, including the candidate-model-service instance set *MSS* and the candidate-data-storage instance set *DSS*. Owing to the large number of candidate resource instances with varying performance, the following two-step screening was performed.

Dependency pre-selection $s_1(\cdot)$. Abnormal instances in the candidate resource sets were eliminated using an isolation forest (iForest) [42]. This can reduce the risk of participation of abnormal instances in orchestration. For model-service instances, we construct an isolation forest using metrics such as service running time (SRT), number of service reboots (NSR), number of service migrations (NSM), and number of service updates (NSU). The feature space is found using Formula (1).

$$F_i = (srt_i, nsr_i, nsm_i, nsu_i). \tag{1}$$

For data-storage instances, we adopt metrics such as the storage quota (SQ), storage utilization (SU), number of host reboots (NHR), and number of storage reboots (NSTR).

The isolation forest acts as an aggregate of isolation trees (iTree). Different isolation trees play different roles as anomaly-identification specialists, which identify resource instances with shorter paths as anomalies. Specifically, the isolated forest detects the abnormal resources by introducing an outlier function, as shown in Formula (2).

$$\begin{aligned} &g(F_i, \varphi) = 2^{-\frac{E(h(F_i))}{c(\varphi)}}, \\ &s.t.\ c(\varphi) = 2(\ln(\varphi - 1) + \varepsilon) - (2(\varphi - 1)/\varphi), \end{aligned} \tag{2}$$

where $E(h(F_i))$ is the mathematical expectation of the depth of leaf node F_i in an isolated forest subtree. $c(\varphi)$ is the average depth of isolated trees containing φ sample data in an isolated forest; ε is the Euler constant. When $E(h(F_i))/c(\varphi) \to 0$, that is, when the evaluation score is close to 1, the resource is judged to be abnormal. We evaluate and mark the abnormal status of resources in the candidate resource sets using Formula (2), with "1" indicating normal and "−1" otherwise. Furthermore, the pre-selected service instance set MSS' and the pre-selected data instance set DSS' were obtained after removing abnormal resources.

Dependency on priority selection $s_2(\cdot)$. Owing to the large number of pre-selected resources in MSS' and DSS', prioritization is required to further filter out the optimal resources. We propose an improved multi-criteria decision-making method to achieve a more comprehensive resource evaluation. The method objectively assigns weights to the decision-making factors of TOPSIS [43] using an entropy weighting method, as shown in Formula (3). This can eliminate the influence of traditional subjective weights on the accuracy evaluation of resources.

$$\begin{aligned} &C = TOPSIS(DM, W), \\ &s.t.\ W = \left(w_1, w_2, \cdots, w_j, \cdots w_m\right), \\ &w_j = \tfrac{1-E_j}{n-\sum E_j}, E_j = -\ln(n)^{-1} \sum_{i=1}^{n} p_{ij} \ln p_{ij}, \end{aligned} \tag{3}$$

where DM denotes the resource decision matrix of dimension $n \times m$, and its row vector is the resource decision vector $D_i (0 \le i \le n-1)$. n is the number of pre-selected resources; m is the dimension of the resource decision vector. W is a decision weight vector with dimensions of $1 \times m$. Vector element w_j denotes the weight coefficients of the decision factors. E_j denotes the information entropy of decision factor j; $p_{ij} = d_{i,j} / \sum_{j=1}^{n} d_{i,j} (0 \le i \le n-1,\ 0 \le j \le m-1)$ is the normalized probability; and $d_{i,j}$ is an element of vector D_i. c denotes the resource evaluation vector with dimensions $n \times 1$. The element of c represents the resource evaluation value.

In terms of service resources, we focus on four types of QoS attributes as decision factors, as shown in Formula (4).

$$D_i = (rt_i, th_i, sla_i, err_i), \tag{4}$$

where rt_i denotes the average response time of model-service i under the concurrent access. th_i denotes the service throughput, that is, the number of service requests that are successfully processed. We define the service response time exceeding threshold δ as a violation. The service violation rate sla_i is expressed as the ratio of the violation time $rt_i - \delta$ to the service response time rt_i, that is, $sla_i = (rt_i - \delta)/rt_i$. err_i denotes the service error rate, which is the ratio of error requests under concurrent access to the total number of requests. In terms of data resources, we focus on four types of QoS attributes as decision factors, as shown in Formula (5).

$$D_i = (cap_i, qps_i, tps_i, iops_i), \tag{5}$$

where cap_i, qps_i, tps_i, and $iops_i$ denote the storage capacity, number of queries per second, number of transactions per second, and number of disk I/O operations per second, respectively.

Finally, all pre-selected resource instances are prioritized using Formula (3). Resource instances with the highest priority rankings are selected for orchestration. The access addresses of these priority-selected resources are injected into the target geospatial-analysis models.

3.2.2. Implementation of Dependency Orchestration

We provide configuration injection and environment-variable injection to support the loading and usage of orchestration information. Orchestration information refers to the access address of the priority-selected model services or data storages. For configuration injection, orchestration information is employed as a configuration item. With configuration mapping, orchestration information is injected into the configuration files. These models can be used in their original way without code modification. For the environment-variable injection, orchestration information is injected into the environment variables of the model containers. The models can consume them by reading environmental variables. Once the model obtains the access addresses of the dependent resources, it can interoperate based on the interface contracts of the dependent resources or by reading and writing the same data storage.

Figure 5 illustrates the principle of orchestrating information injection and perception. First, we built configuration items in the format of "CapID_Keywords", such as "SERVICE.WORKFLOW.IMAGESLICE.V1_IP". Then, configuration items are constructed as configuration instances through the mapping of "Configuration Items—Configuration Templates—Configuration Instances". Note that the placeholder "@{}" in the configuration templates will be replaced by the value of the configuration items accordingly. Once the configuration items have been altered, a new configuration instance is built. Finally, ConfigMap is adopted to carry configuration instances in the underlying containerized environment. ConfigMap is a configuration-management solution for containerized applications provided by the open-source container orchestration system, Kubernetes [44]. It supports persistent storage of configuration contents as key-value strings and maps these contents into configuration files within the model containers via virtualized mounts. Models can consume configuration files in their original manner without code modification. Furthermore, once the configuration contents in ConfigMap are modified, the configuration files were automatically updated accordingly. The models were dynamically updated using Kubernetes' rolling upgrade mechanism for hot awareness of configuration changes.

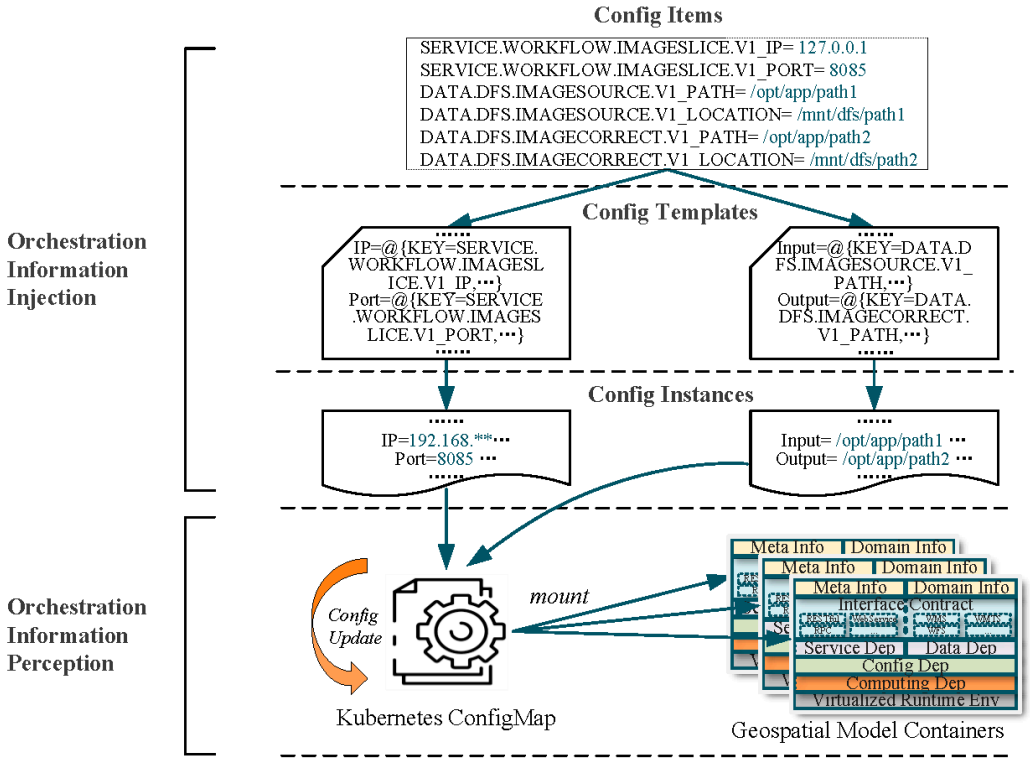


Figure 5. Principle of orchestration information injection and perception.

3.3. Adaptive Publication of Geospatial-Analysis Models

On the basis of dependency orchestration, the model publication component further publishes geospatial-analysis models as model services. Owing to the fact that geospatial-analysis models form a wide range of sources with variable resource requirements, an affinity mapping between the models and the underlying computational resources needs to be established so that the models can be published to the optimal server nodes. To this end, we designed a heuristic model-scheduling method $G(\cdot)$, which consists of the model-feature-parameterization sub-method $g_1(\cdot)$ and the heuristic-policies sub-method $g_2(\cdot)$. Among them, $g_1(\cdot)$ extracts model features and maps them into publishing parameters; $g_2(\cdot)$ schedules the models to the optimal server nodes via affinity-aware heuristics.

3.3.1. Model-Feature Parameterization

The primary challenge of model publication is properly configuring the publishing parameters to support the multi-modal running of geospatial-analysis models. To adapt the highly stochastic container-based cloud environments, we designed two categories of publishing parameters. Static parameters are used to configure the running modes of geospatial-analysis models. Dynamic parameters are employed to adaptively tune the underlying resource provisioning. Further, we designed a model-feature-parameterization sub-method $g_1(\cdot)$ to adaptively configure model publishing parameters, as shown in Figure 6. The flow is as follows:

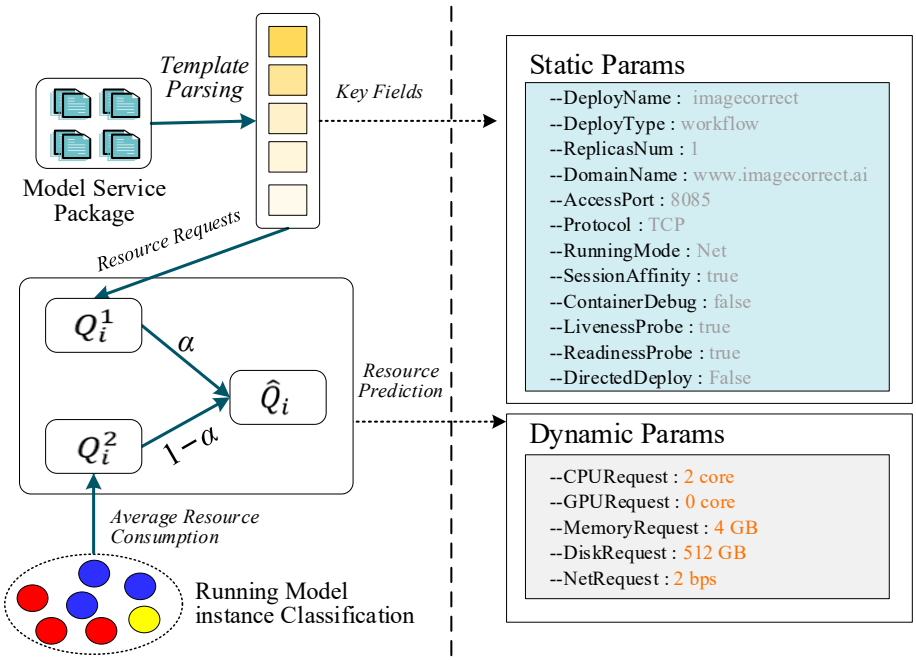


Figure 6. Flow of model-feature-parameterization sub-method $g_1(\cdot)$.

First, key fields are parsed from declarative templates to extract the static publishing parameters. Then, the dynamic publishing parameters are adaptively calculated by combining the resource requests set by the users and the actual resource consumption of similar models, as in Formula (6).

$$\hat{Q}_i = \alpha \cdot Q_i^1 + (1-\alpha) \cdot Q_i^2, \tag{6}$$

where $\hat{Q}_i = (\hat{q}_{i,1}, \hat{q}_{i,2}, \cdots, \hat{q}_{i,K})$ denotes the actual allocated resources; K is the number of resource types. Q_i^1 denotes the user-set resources. Considering that Q_i^1 may differ significantly from the actual requirements of the model, we optimize it using the similar model-requirement resources Q_i^2. Coordination coefficient α is employed to balance the two. In the evaluation of model similarity, we categorized geospatial-analysis models as access-intensive, data-intensive, computation-intensive, and non-intensive, based on

underlying resource requirements. Among them, access-intensive tends to consume network bandwidth resources; data-intensive tends to consume memory and disk resources; and computing-intensive tends to consume CPU and GPU resources. The probability of resource demand is defined in Formula (7).

$$\begin{aligned} &P_i = \left(p_{i,1}, p_{i,2}, \ldots, p_{i,k}, \ldots, p_{i,K}\right), \\ &s.t.\ \ p_{i,k} = s_{i,k} / \textstyle\sum_{k=1}^{K} s_{i,k}, s_{i,k} = q_{i,k} / m_k, \end{aligned} \tag{7}$$

where $p_{i,\,k}$ denotes the probability of model i demanding resource k. $q_{i,k}$ is the resource request of model i; m_k is the maximum quota of resource k on the server nodes of a container-based cloud cluster. We define the resource type with the maximum demand probability as the resource tendency category of the corresponding model. Models with the same resource-tendency category were identified as similar. Taking the actual resource consumption of similar models as a reference for the resource allocated of the current model, it can more accurately reflect the actual resource demand of the current model and realize the reasonable allocation of the underlying resources.

3.3.2. Heuristic Model Scheduling Policies

Geospatial-analysis models are characterized by task and data diversity, with significant differences in resource and performance requirements between models. For example, the portal applications of the smart city have the characteristics of high concurrency and intensive access. They can be deployed in cluster environments with a high bandwidth and low latency. Geographic big-data analysis tasks and real-time stream processing tasks have intensive I/O requirements. They can be deployed in different nodes of loosely-coupled computing clusters. Machine learning tasks for geo-scenario data analysis and mining require large amounts of CPU and GPU computation resources. They can be deployed to tightly coupled computing clusters with distributed computing capabilities. Thus, establishing affinity mapping between geospatial-analysis models and underlying resource nodes is another challenge for model publication. In this section, a heuristic policy sub-method, $g_2(\cdot)$, is designed to adaptively allocate the optimal scheduling nodes for geospatial-analysis models.

As shown in Figure 7, the flow of the heuristic policy sub-method $g_2(\cdot)$ is mainly divided into two stages. The model identification stage identifies the resource tendency category and predicts the resource consumption of the current models based on sub-method $g_1(\cdot)$. The model scheduling stage determines the optimal scheduling nodes using the heuristic probability function $h(i,j)$, as shown in Formula (8).

$$\begin{aligned} &h(i,j) = \textstyle\sum_{k=1}^{K} w_k \cdot h_{i,j}^k, \\ &s.t.\ \ h_{i,j}^k = \beta \times AR_{i,j}^k + (1-\beta) \times AE_i^k, \\ &AR_{i,j}^k = \frac{r_{j,k} - u_{j,k} - \hat{q}_{i,k}}{r_{j,k} - u_{j,k}}, \end{aligned} \tag{8}$$

where $h(i,j)$ denotes the probability of model i scheduling to node j. $h_{i,j}^k$ is the resource affinity probability; w_k is the weighting coefficient. $AR_{i,j}^k$ and AE_i^k are the calculated value and expert experience value of resource affinity. $r_{j,k}$ and $u_{j,k}$ denote the quota and usage of resource k on node j. $\hat{q}_{i,k}$ denotes the predicted resource consumption as in Formula (6). The coordination coefficient β was employed to balance $AR_{i,j}^k$ and AE_i^k. The geospatial-analysis models are scheduled to the nodes with maximum scheduling probability $h(i,j)$.

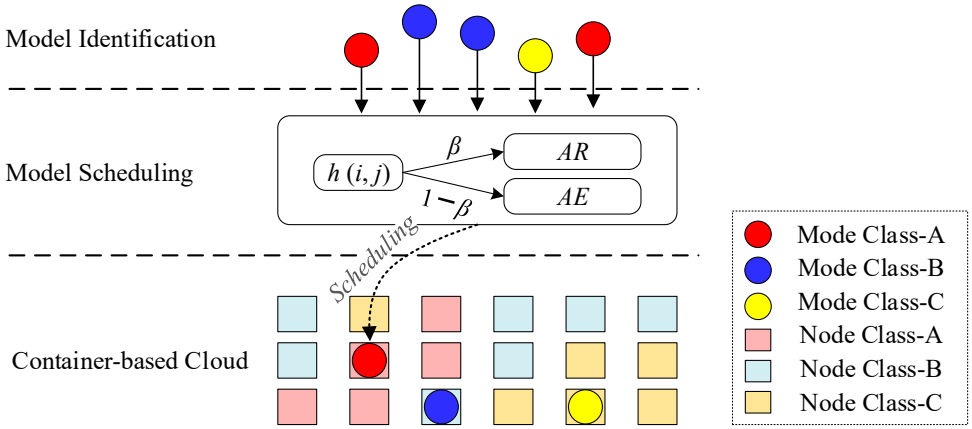


Figure 7. Flow of heuristic policies sub-method $g_2(\cdot)$.

4. Design of the System

In this section, we introduce the prototype GeoCSIF system. First, the system architecture was designed, and the interface dependencies between the system components were detailed. Subsequently, a prototype system is developed, and its application process is introduced.

4.1. System Architecture and Its Component Relationships

From the perspective of system architecture, the prototype GeoCSIF system can be divided into resource, business, interface, and presentation layers from the bottom to top, as shown in Figure 8. The resource layer mainly contains the basic and middleware resources required for running the system. The basic resources provide the necessary computing, storage, and network resources for the continuous integration of geospatial-analysis models. Middleware resources provide storage and learning environments. Among them, the storage environment employs a distributed file system for persistent-model-service packages and a relational database for persistent model information. The learning environment provides the Prometheus component to sample the environmental state, and Python libraries to build model integration methods. The business layer includes three components: model encapsulation, orchestration, and publication. The model encapsulation component builds raw models into unified-model-service packages. The model orchestration components provide visual interfaces to assemble large-scale geospatial-analysis models efficiently. The model publication component adaptively publishes model services to enable resource sharing. The interface layer exposes interfaces for the integration management of geospatial-analysis models in the form of HTTP RESTful, including the model information retrieval, resource evaluation pushing, and service running control interfaces. The presentation layer provides users with human–computer interaction views, including model management, orchestration management, and service management.

Figure 9 illustrates the component dependencies of the prototype system. First, the model encapsulation component transmits model information and model-service packages to the relational database and distributed file system by invoking the database read/write interface and the file system read/write interface. In addition, the model information registration interface is invoked to synchronize the model-servicized information to model the orchestration component. Then, the model orchestration component assembles model-dependency resources and registers the orchestrated models to the model publication component using the model-publishing-registration interface. Finally, the model publication component samples the environment state to compute the optimal scheduling nodes using the environment-state-monitoring interface. Meanwhile, it accomplishes the scheduling deployment of model services using the service-runtime-control interface.

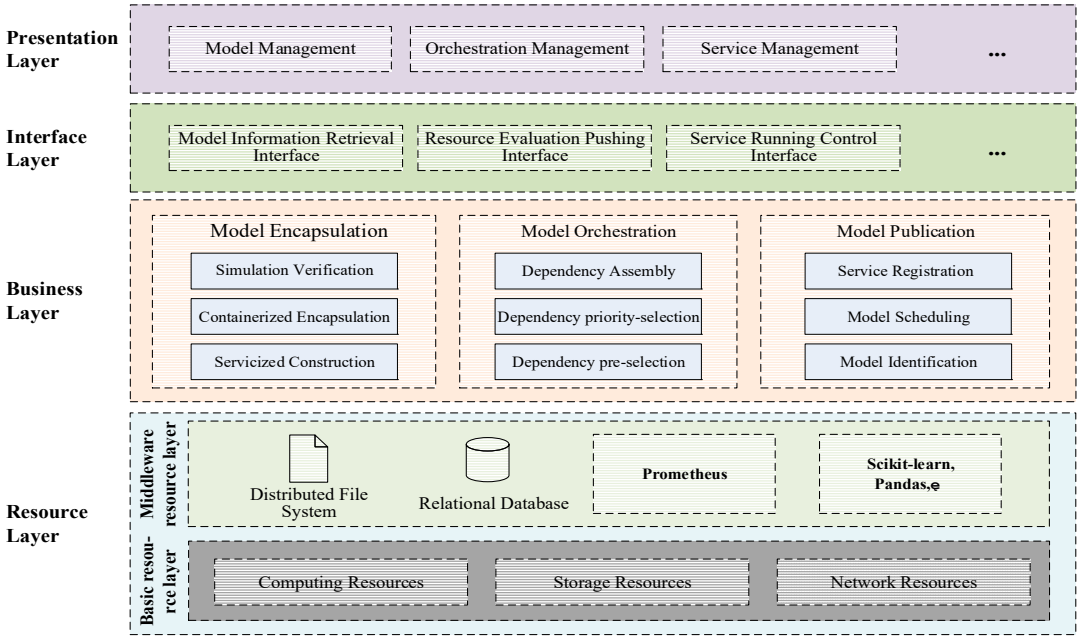


Figure 8. System architecture of GeoCSIF.

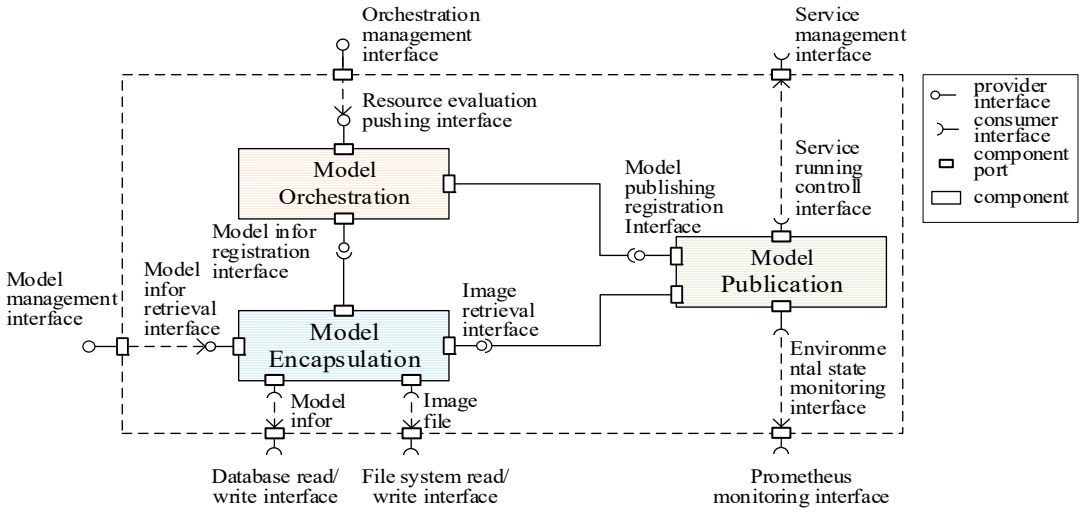


Figure 9. Component invocation relationships of prototype system.

4.2. System Implementation and Function Introduction

In this section, we implemented a prototype GeoCSIF system with a human–machine interface, as shown in Figures 10–12.

Figure 10 shows the human–machine interface of the model encapsulation component, including steps of capability identification, template construction, image build, and package export. The capability identification step defines the capability identifiers of geospatial-analysis models and their dependent resources based on the resource type, resource subtype, business type, and version number. The template construction step constructs service, data, configuration and environment templates based on the standardized constraints of the model service. The image-build step builds a model executable file as a docker business image. The package-export step organizes the template files and business image into a unified-model-service package.

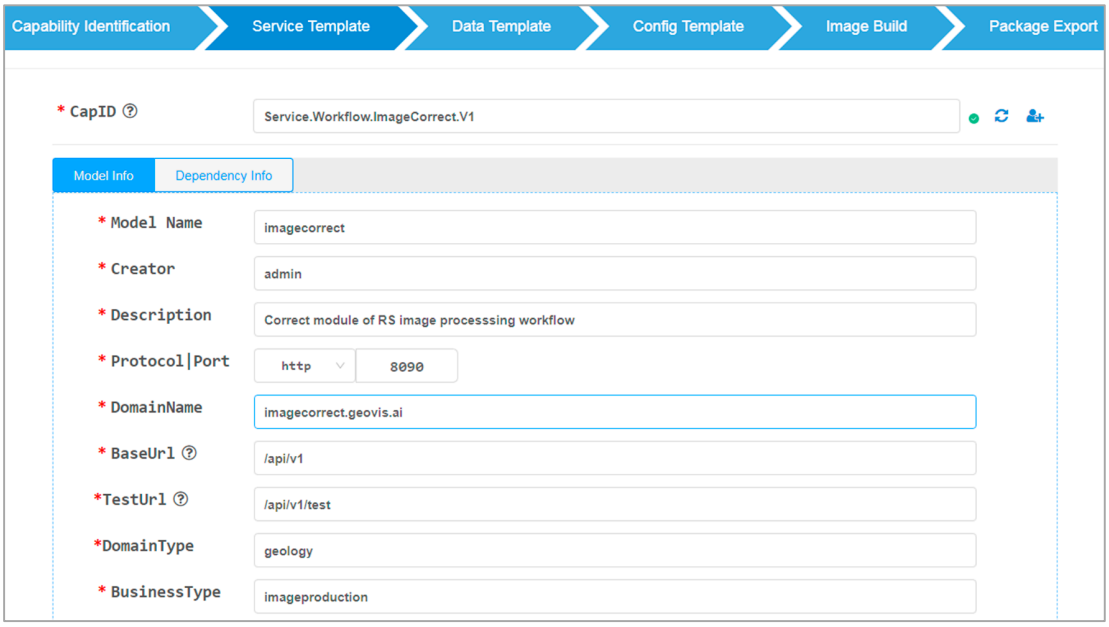


Figure 10. Human–machine interface of model encapsulation component.

Figure 11 provides a graphical interface for model-dependent resources orchestration. The model-dependency topology on the left side of Figure 12 shows the business-dependency relationships of the geospatial-analysis models. The right is the recommendation result of the dependent resources based on prioritization. By using the symbol star to represent the recommendation level of dependent resources, five stars represent the highest priority and one star otherwise. The component supports the automatic selection of the highest-priority resources for assembly, while users can also reselect manually.

Once all dependencies have been orchestrated, click "New Deploy" to enter the human interface of the model publication component (Figure 12). Note that the static deployment parameters were automatically populated by extracting key fields from the service template of geospatial-analysis models. It also supports online adjustments by users. Once all deployment parameters are correctly configured, users can click the "Submit" button to publish models. Subsequently, the models were automatically dispatched to the optimal nodes of the container cloud cluster. The access addresses of the model services are registered in the resource instance catalog for sharing and reusing support.

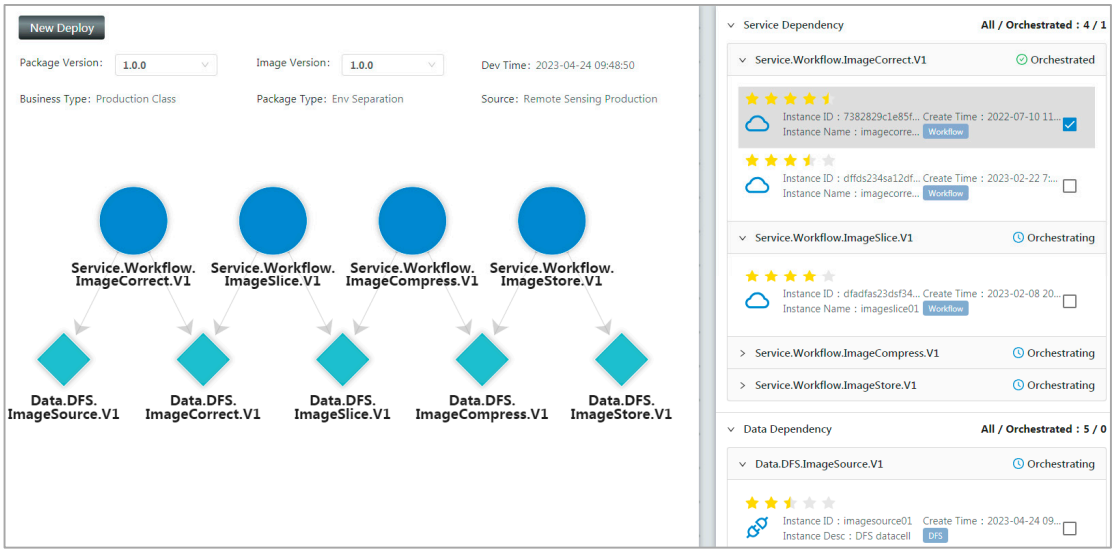


Figure 11. Human–machine interface of model orchestration component.

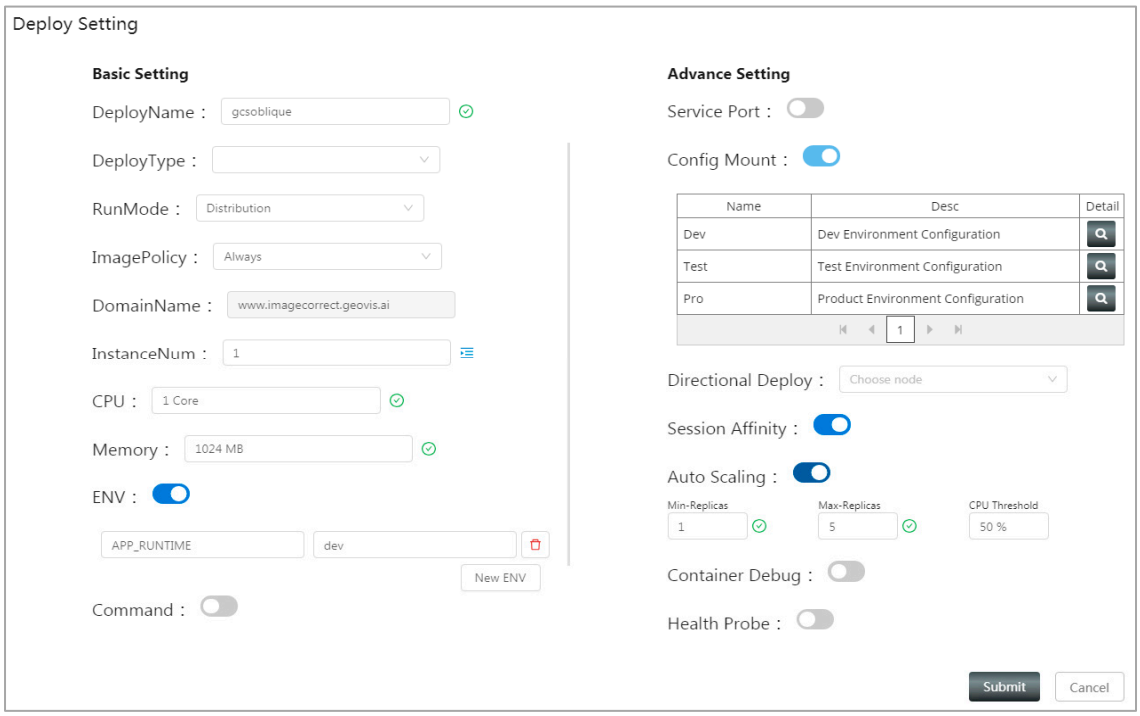


Figure 12. Human–machine interface of model publication component.

5. Case Studies

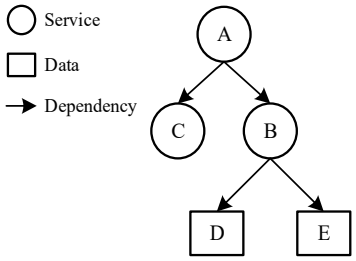
In this section, we introduce the GeoCSIF process with a practical case. Moreover, the effectiveness of the framework was evaluated using comparative experiments. We adopted flood-disaster models as the object of this study. The main reasons are the complexity of its dependencies, wide variation in underlying resource requirements, and heterogeneity of runtime environments. Owing to the conflict of program-dependent libraries, traditional methods have not successfully published these models. Therefore, we attempted to publish them through the proposed GeoCSIF framework.

5.1. Geospatial-Analysis Models for Flood-Disaster Prediction

Flood-disaster models are used to predict and assess the occurrence and evolution of floods, providing a more scientific and effective basis for disaster prevention and mitigation. The selected flood-disaster models consist of three model-service units and two data units; the dependencies between the models are shown in Figure 13a. The units were coupled to each other through lightweight HTTP communication. Among them, Unit A is the core model used to simulate the propagation of floods in the drainage and surface networks. Unit B is a data source service that constructs and provides data including a digital-terrain model and an engineering-drainage-network model for Unit A. Unit B relies on a relational database and a distributed file system to persist model meta-information and entity files. Unit C is a visualization service that provide visualization of simulation results.

The challenges for publishing the model using traditional stand-alone methods lie in the following three aspects: (1) Because the model depends on multiple types of resources, the traditional manual orchestration method is time-consuming and labor-intensive. In this case, B and C are deployed in advance of A and their access addresses are injected into the configuration file of A. If multiple instances of B and C are available, manually filtering the optimal instances is not only tedious but also lacks professionalism. (2) Owing to the large difference in the demand for underlying computing resources among different models, traditional methods are prone to resource fragmentation and resource waste. In this case, A, B, and C are computation, data, and access intensive, respectively. They must be deployed on servers with appropriate resource types. Once the matching fails, one

resource is exhausted, while the others remain, making it impossible to continue deploying other services. (3) Heterogeneity of runtime environments are difficult to cope with using traditional methods. These models rely on different operating systems, dependent libraries, and Python environments, as shown in Figure 13b. It can be difficult to deploy multiple models on the same server owing to conflicting dependencies. In conclusion, this case presents a great challenge to traditional methods while better evaluating the integration effectiveness of the proposed GeoCSIF framework.



(**a**)

Name	CapID	Service Dependencies	Data Dependencies	Environment Dependencies
A	Service.LongRunning.FloodPrediction.V1	B, C	-	CentOS 7.8, Fortran90, g++,gfortran,netcdf4.2,1, etc.
B	Service.LongRunning.ModelConstruction.V1	-	D, E	Ubuntu12.04, Python2.7, GDAL3.0.4, Pandas2.0, etc.
C	Service.LongRunning.Visualization.V1	-	-	Ubuntu14.02 Node6.8.1, Python3.6, etc.
D	Data.DFS.Source.V1	-	-	-
E	Data.PostgreSQL.Source.V1	-	-	CentOS8, PostgreSQL10

(**b**)

Figure 13. Flood-disaster-prediction models: (**a**) model dependencies, and (**b**) model details.

5.2. Integration Verification of Heterogeneous Geospatial-Analysis Models

In this section, the effectiveness of GeoCSIF is first verified by using the flood disaster models shown in Figure 13. Then, comparative experiments were conducted to evaluate the performance of GeoCSIF by simulating large-scale model-integration scenarios.

Experiment A evaluates the effectiveness of GeoCSIF. According to the service-based integration framework shown in Figure 1, the integration process includes (1) declaring capability identifiers for each model, constructing service, data, configuration, and environment templates, building business images, and encapsulating model-service packages. (2) Based on the model-service-package dependencies, the GeoCSIF automatically orchestrates the model and data-storage resources on which the current model depends. The access addresses of these dependent resources are written to the current model's configuration files. (3) The GeoCSIF system configures model publishing parameters, assembles the model runtime environment, as well as program dependency libraries. Depending on the model type and current environment state, the models are published to the optimal server nodes. (4) It includes accessing and validating the availability of flood-disaster-prediction models.

As show in Figure 14, the flood-disaster-prediction models can work together properly. We simulated heavy rainfall over a period of time through Model A, as shown in (a), and viewed the flood-simulation results through Model C, as shown in (b) and (c). It can be seen that after five hours of sustained heavy rainfall, the eastern and northwestern regions have seen a large amount of standing water. This is mainly because some underground pipes run at full capacity. Therefore, it is necessary to increase the number of pipelines and inlets in this region to cope with the threat of flooding caused by heavy rainfall. The experimental result shows that the GeoCSIF framework can effectively cope with the integration of complex geospatial-analysis models.

Experiment B evaluates the model-servicized performance of GeoCSIF. We compared and analyzed the efficiency of service-package management under different encapsulation modes. As can be seen from Figure 15a,b, the native structure for traditional standalone deployment includes a dependency library, executable file, and installation script. The proposed servicized structure for cloud deployment includes a service template, data template, configuration template, environment template, business image, and description file with a more fine-grained model structure.

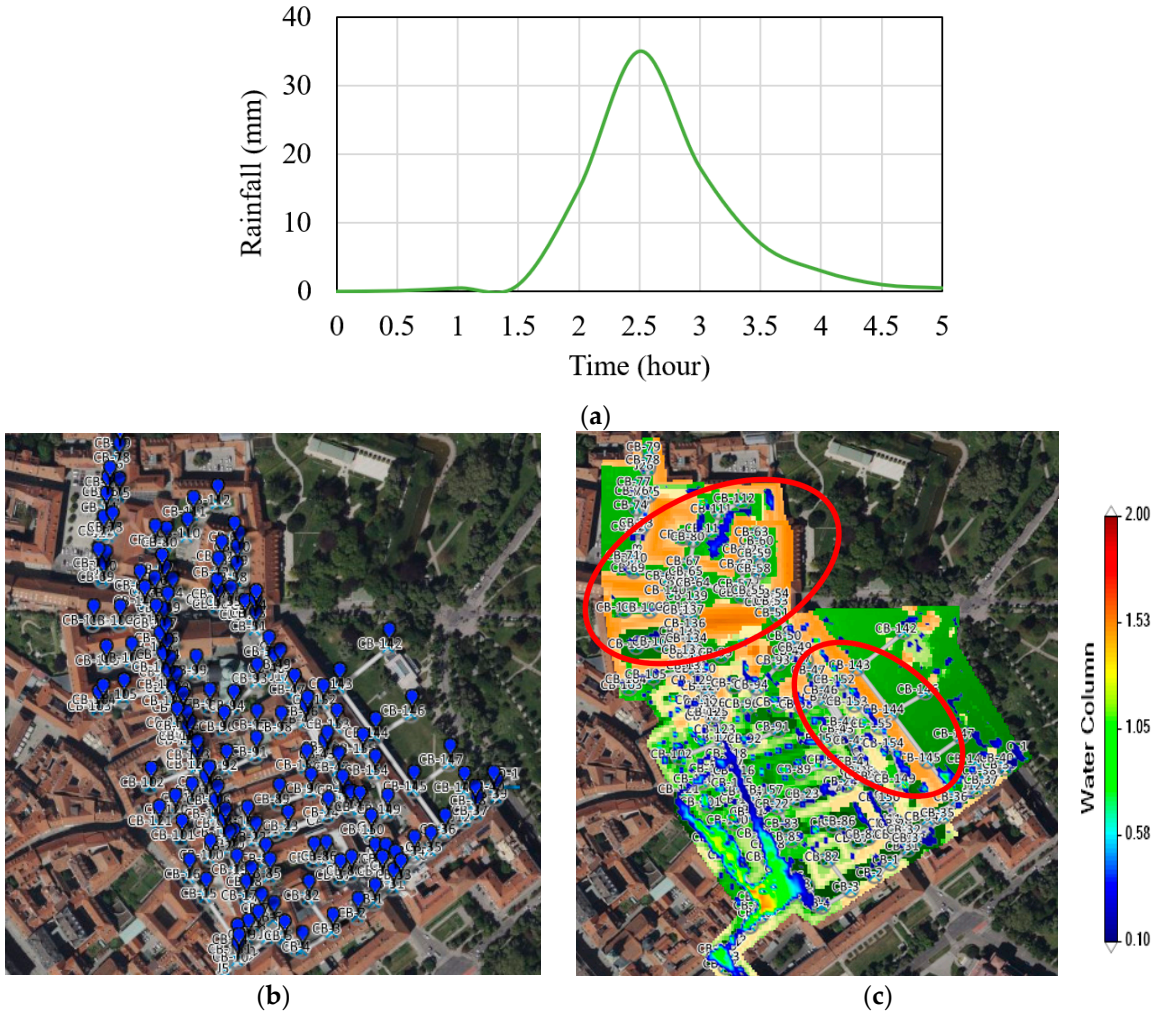


Figure 14. Results of models in flood prediction: (**a**) simulation of urban heavy rainfall; (**b**) underground drainage network; and (**c**) surface flood distribution.

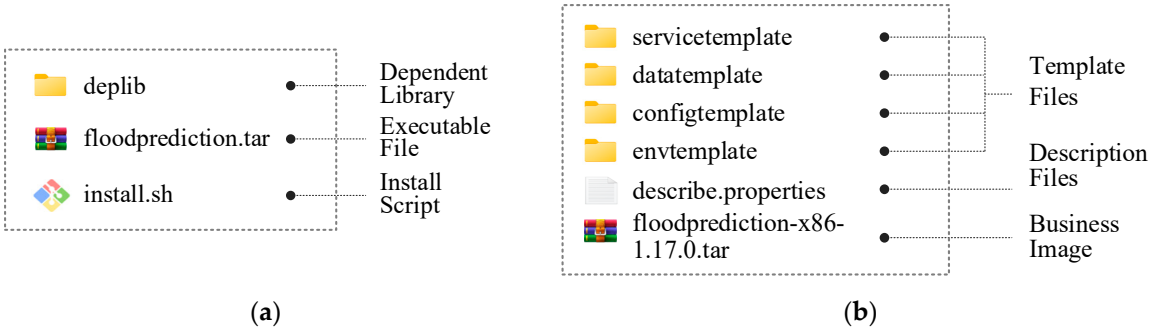


Figure 15. Comparison of model structures: (**a**) native structure; (**b**) servicized structure.

Figure 16 shows a comparison of model-service-package volumes under different encapsulation modes. Compared to the native structure, the proposed servicized structure has an average volume reduction of over 21%, indicating that servicized encapsulation is more lightweight. Thanks to the fact that we decompose and refine the geospatial-analysis models in a modular way and encapsulate them by using virtualization with the separation of business and environment. That is, the larger running environment and dependent libraries are not encapsulated in the model-service package, but rather they declare references to them. The runtime assembles business executable files, runtime

environment, and dependency libraries together through dynamic loading. This not only effectively reduces the volume of the package, but can also adapt to the high reliability and scalability of container-based cloud environments.

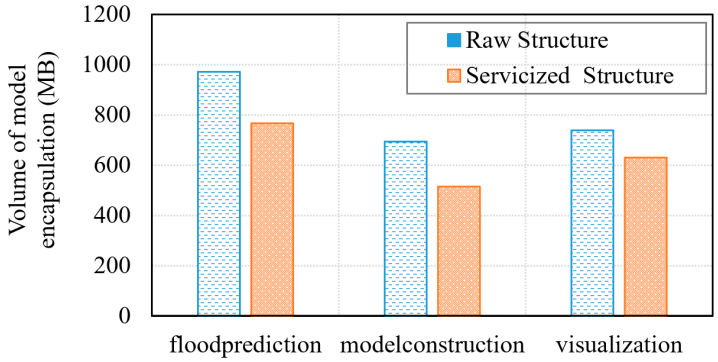


Figure 16. Comparison of model volume under different encapsulation methods.

Experiment C evaluates the model orchestration performance of GeoCSIF. By simulating heterogeneous orchestration scenarios with different candidate model instance scales, the proposed prioritization-based orchestration method (PBO) is comprehensively compared with the traditional methods such as Random, meta-heuristic GA-Par [45]. The experimental results are shown in Figure 17a–d. We compared the orchestration completion time, service response time, throughput, and SLA violation rate under different orchestration methods. Figure 17a shows that the proposed PBO is intermediate between Random and meta-heuristic GA-Par in terms of orchestration efficiency. This is mainly owing to the tedious optimization process of prioritization-based orchestration, which brings some time overhead. However, Figure 17b–d show that the model has better QoS (quality of service) performances under the prioritization-based orchestration. Compared to GA-Par, the response time, throughput, and SLA violation rate of the model services under PBO improve by about 8.2%, 10.7%, and 14.2%, respectively. Thanks to the prioritization-based orchestration employing a two-phase dependency filtering, the pre-selection phase excludes anomalous instances to reduce the probability of model-service failures, while the priority-selection phase further filters high-quality instances to improve the model-service performance.

Experiment D evaluates the model scheduling performance of GeoCSIF. By simulating large-scale model-scheduling scenarios, the efficiency of the proposed heuristic-model-scheduling method (HMS) in utilizing underlying computing resources is evaluated. The comparative algorithms are selected as traditional Random scheduling and More Resource Surplus First (MRSF) scheduling. We assume that the number of models arriving at the GeoCSIF system in a time slot (1 min) conforms to the Poisson distribution, and simulates model arrival time at a frequency of $\lambda = 50$. The experimental results are shown in Figure 18. In where, Figure 18a shows that the resource utilization under HMS is more balanced. A numerical analysis shows that compared with Random and MRSF, cluster resource utilization under HMS increases by 15.7% and 10.3%. Figure 18b shows that HMS can match the models to the corresponding server nodes according to their types, with a higher matching rate. Thanks to the proposed heuristic scheduling, the resource consumption types of the models are effectively identified and divided into access intensive, data intensive, computation intensive and non-intensive. Moreover, the mapping relationship of resource affinity based on environment and expert experience is established, which can match the models to the optimal resource nodes and realize the global optimization of cluster resource.

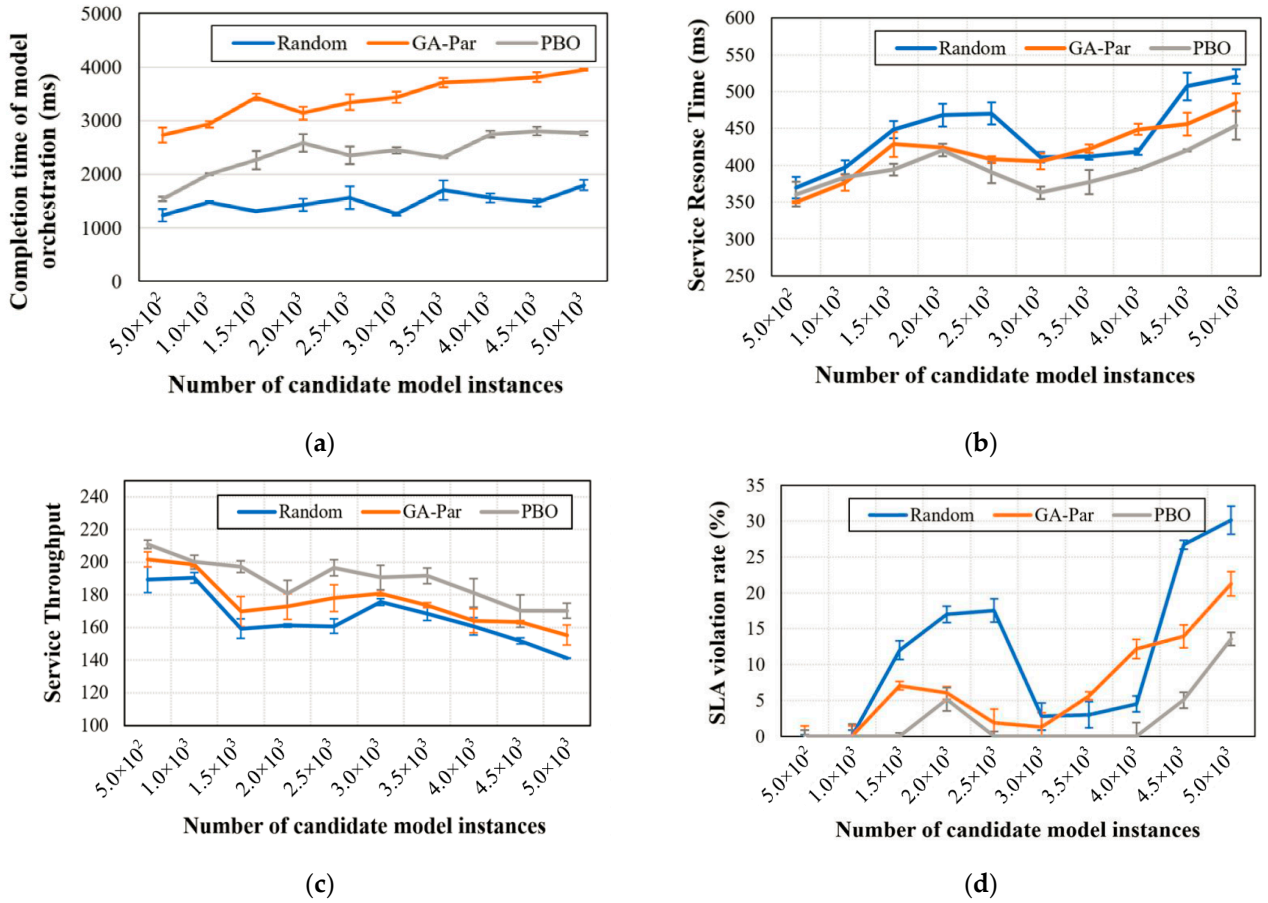


Figure 17. Comparison of model orchestration performance: (**a**) completion time of model orchestration; (**b**) service response time; (**c**) service throughput; and (**d**) SLA violation rate.

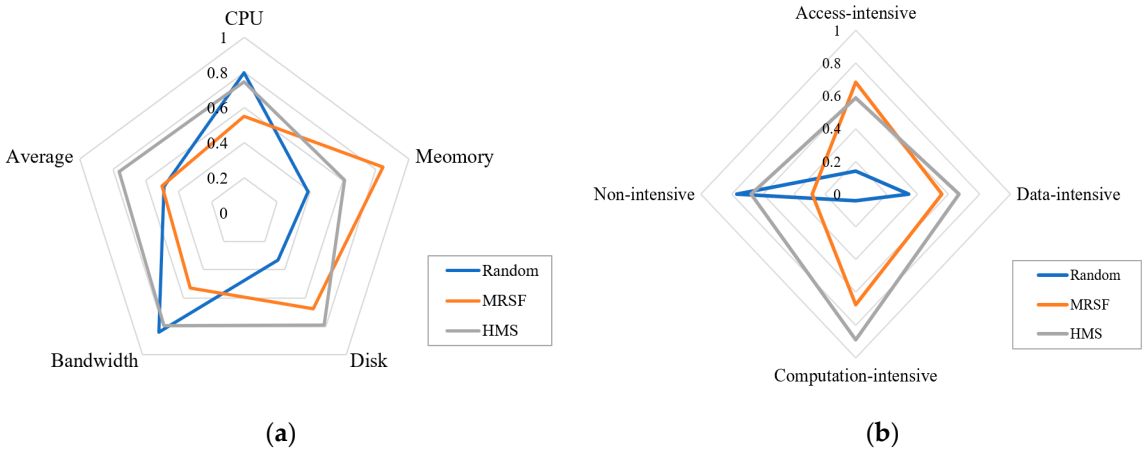


Figure 18. Comparison of scheduling performance: (**a**) resources utilization; and (**b**) matching rate of models to resources.

6. Conclusions and Future Work

This study proposes a method for publishing large-scale heterogeneous-geospatial-analysis models as model services in container-based cloud environments. This is of great significance for the reusability and sharing of geospatial-analysis models. Geospatial-analysis models encounter greater challenges compared to non-geospatial and traditional geospatial models due to the intricate computations and extensive processing of geographic

data involved, thus giving rise to a multitude of issues. One core issue is how to eliminate model heterogeneity to facilitate model combination and capability integration.

To address these challenges, this study conducted thorough research and exploration, leading to the development of the GeoCSIF framework. The GeoCSIF framework can facilitate the seamless integration of diverse geospatial-analysis-model resources, and utilizes container-based cloud environments as the runtime infrastructures to address complex geo-related problems. At first, a model-servicized structure was designed to shield model structure heterogeneity. Geospatial-analysis models can be integrated on the basis of adhering to standardized constraints. Based on the mode-servicized structure, a templated constraint method was employed to encapsulate diverse geospatial-analysis models as unified-model-service packages. Then, to address the challenge of combining potentially hundreds or even thousands of geospatial-analysis-model units in complex geospatial applications, like in the smart city, a prioritization-based orchestration method was proposed. This method enables the automatic discovery and optimized combination of model-dependent resources. Meanwhile, addressing the challenges of diverse runtime environments and intensive model computations, a heuristic scheduling method was proposed. This method effectively schedules geospatial-analysis models to optimal server nodes based on their execution characteristics, thereby enhancing model stability and service performance. Lastly, a prototype system was developed, and its integration process for heterogeneous-flood-disaster models was compared with mainstream methods. Experimental results indicate that GeoCSIF possesses superior performance in model management and service efficiency including model servicized, orchestration, and scheduling.

Although much work has been conducted on heterogeneous-geospatial-analysis-model integration, the proposed framework still needs to be continuously expanded to cope with more complex application scenarios. Future work will involve uncertainty modeling and inference, and deepening and expansion of application domains. These studies will help further improve the accuracy, robustness, and application effectiveness of heterogeneous-geospatial-analysis-model integration.

Author Contributions: Conceptualization, Lilu Zhu; data curation, Yang Wang and Kai Huang; formal analysis, Yunbo Kong and Yanfeng Hu; methodology, Lilu Zhu; project administration, Yanfeng Hu; resources, Yang Wang and Yunbo Kong; software, Lilu Zhu; supervision, Yanfeng Hu; validation, Yang Wang; visualization, Kai Huang; writing—original draft, Lilu Zhu; writing—review and editing, Yang Wang. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Suzhou Cutting-edge Technology Research Project, grant number SYG202335.

Data Availability Statement: The data that support the findings of this study are available from the corresponding author, Zhu L., upon reasonable request.

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

References

1. Yue, S.; Wen, Y.; Chen, M.; Lu, G.; Hu, D.; Zhang, F. A data description model for reusing, sharing and integrating geo-analysis models. *Environ. Earth Sci.* **2015**, *74*, 7081–7099. [CrossRef]
2. Zhang, F.; Chen, M.; Ames, D.P.; Shen, C.R.; Yue, S.S.; Wen, Y.N.; Lü, G.N. Design and development of a service-oriented wrapper system for sharing and reusing distributed geoanalysis models on the web. *Environ. Model. Softw.* **2019**, *111*, 498–509. [CrossRef]
3. Ustugova, S.; Parygin, D.; Sadovnikova, N.; Yadav, V.; Prikhodkova, I. Geoanalytical system for support of urban processes management tasks. In Proceedings of the Creativity in Intelligent Technologies and Data Science: Second Conference, CIT&DS 2017, Volgograd, Russia, 12–14 September 2017; pp. 430–440.
4. Wang, J.; Chen, M.; Lü, G.; Yue, S.; Chen, K.; Wen, Y. A study on data processing services for the operation of geo-analysis models in the open web environment. *Earth Space Sci.* **2018**, *5*, 844–862. [CrossRef]
5. Zhang, B.C.; Chen, M.; Ma, Z.Y.; Zhang, Z.; Yue, S.S.; Xiao, D.W.; Zhu, Z.Y.; Wen, Y.N.; Lü, G.N. An online participatory system for SWMM-based flood modeling and simulation. *Environ. Sci. Pollut. Res.* **2022**, *29*, 7322–7343. [CrossRef] [PubMed]
6. Lin, Q.; Lin, B.; Zhang, D.; Wu, J. Web-based prototype system for flood simulation and forecasting based on the HEC-HMS model. *Environ. Model. Softw.* **2022**, *158*, 105541. [CrossRef]

7. Fang, Z.; Yue, P.; Zhang, M.D.; Xie, J.B.; Wu, D.J.; Jiang, L.C. A service-oriented collaborative approach to disaster decision support by integrating geospatial resources and task chain. *Int. J. Appl. Earth Obs. Geoinf.* **2023**, *117*, 103217–103230. [CrossRef]
8. Bhandari, P.; Anastasopoulos, A.; Pfoser, D. Are large language models geospatially knowledgeable? In Proceedings of the 31st ACM International Conference on Advances in Geographic Information Systems, Hamburg, Germany, 13–16 November 2023; pp. 1–4.
9. Qi, Y.; Jiang, H.; Li, S.; Cao, J. ConvLSTM coupled economics indicators quantitative trading decision model. *Symmetry* **2022**, *14*, 1896. [CrossRef]
10. Islam, T.; Rahman, M.R.; Khan, A.; Moni, M.A. Integration of Mendelian randomisation and systems biology models to identify novel blood-based biomarkers for stroke. *J. Biomed. Inform.* **2023**, *141*, 104345–104381. [CrossRef]
11. Torres, W.; Van den Brand, M.G.J.; Serebrenik, A. A systematic literature review of cross-domain model consistency checking by model management tools. *Softw. Syst. Model.* **2021**, *20*, 897–916. [CrossRef]
12. Li, M.; Stefanakis, E. Geospatial operations of discrete global grid systems—A comparison with traditional GIS. *J. Geovisualization Spat. Anal.* **2020**, *4*, 1–21. [CrossRef]
13. Abdulrahman, I.; Radman, G. Power system spatial analysis and visualization using geographic information system (GIS). *Spat. Inf. Res.* **2020**, *28*, 101–112. [CrossRef]
14. Kraak, M.J.; Ormeling, F. *Cartography: Visualization of Geospatial Data*; CRC Press: Boca Raton, FL, USA, 2020.
15. Pu, H.; Wan, X.J.; Song, T.R.; Schonfeld, P.; Li, W.; Hu, J.P. A geographic information model for 3-D environmental suitability analysis in railway alignment optimization. *Integr. Comput. Aided Eng.* **2023**, *30*, 67–88. [CrossRef]
16. Jena, R.; Pradhan, B.; Naik, S.P.; Alamri, A.M. Earthquake risk assessment in NE India using deep learning and geospatial analysis. *Geosci. Front.* **2021**, *12*, 101110–101125. [CrossRef]
17. Manna, P.; Bonfante, A.; Colandrea, M. A geospatial decision support system to assist olive growing at the landscape scale. *Comput. Electron. Agric.* **2020**, *168*, 105143–105153. [CrossRef]
18. Mind'je, R.; Li, L.; Kayumba, P.M.; Mindje, M.; Ali, S.; Umugwaneza, A. Integrated geospatial analysis and hydrological modeling for peak flow and volume simulation in Rwanda. *Water* **2021**, *13*, 2926. [CrossRef]
19. Stoimenov, L.; Stanimirovi, A.; Djordjevi-Kajan, S. Realization of component-based GIS application framework. In Proceedings of the 7th AGILE Conference on Geographic Information Science, Heraklion, Greece, 29 April–1 May 2004; pp. 113–120.
20. Liu, S.H.; Liu, L.P.; Yu, H.L. XML based GIS application model definition and description language GBMDL. *China Manag. Inform.* **2009**, *12*, 93–98.
21. Zhou, L.L.; Wang, R.J.; Cui, C.Y.; Xie, C.J. GIS application model based on cloud computing. In Proceedings of the Network Computing and Information Security: Second International Conference, NCIS 2012, Shanghai, China, 7–9 December 2012; pp. 130–136.
22. Ou, S.J. Research and Application of Geographic Information System Development Based on Component Architecture. Ph.D. Thesis, Jilin University, Jilin, China, 2003.
23. Jia, J. Research on key technologies of geographic information system development. *Constr. Des. Project* **2022**, *9*, 142–144.
24. Dontsov, A.A.; Sutorikhin, I.A. Development of a geographic information system for data collection and analysis based on microservice architecture. *CEUR Workshop Proc.* **2021**, *3006*, 280–287.
25. Zhang, X.L.; Ren, Z.G.; Cao, Y.B. Research on seamless integration of spatial analysis model and GIS. *Geospat. Inf.* **2014**, *12*, 156–158+12.
26. Xu, Z.H. GIS functional component library and functional integration. *Geomat. Inf. Sci. Wuhan Univ.* **2001**, *4*, 303–309.
27. Lv, D.; Ying, X.X.; Gao, X.B.; Tao, W.D.; Cui, Y.J.; Hua, T.T. A WebGIS platform design and implementation based on open source GIS middleware. In Proceedings of the 24th International Conference on Geoinformatics, Galway, Ireland, 14–20 August 2016; pp. 1–5.
28. Wang, S.H.; Zhong, Y.; Wang, E.Q. An integrated GIS platform architecture for spatiotemporal big data. *Future Gener. Comput. Syst.* **2019**, *94*, 160–172. [CrossRef]
29. Wen, Y.N.; Chen, M.; Lu, G.N.; Lin, H.; He, L.; Yue, S.S. Prototyping an open environment for sharing geospatial analysis models on cloud computing platform. *Int. J. Digit. Earth* **2013**, *6*, 356–382. [CrossRef]
30. Qiao, X.H.; Li, Z.Y.; Zhang, F.Y.; Ames, D.P.; Chen, M.; Nelson, E.J.; Khattar, R. A container-based approach for sharing environmental models as web services. *Int. J. Digit. Earth* **2021**, *14*, 1067–1086. [CrossRef]
31. Jiang, B. Geospatial analysis requires a different way of thinking: The problem of spatial heterogeneity. *GeoJournal* **2015**, *80*, 1–13. [CrossRef]
32. Jat, M.K.; Choudhary, M.; Saxena, A. Application of geo-spatial techniques and cellular automata for modelling urban growth of a heterogeneous urban fringe. *Egypt. J. Remote Sens. Space Sci.* **2017**, *20*, 223–241. [CrossRef]
33. Patanè, G.; Spagnuolo, M. *Heterogeneous Spatial Data: Fusion, Modeling, and Analysis for GIS Applications*; Springer Nature: Berlin, Germany, 2022.
34. Hu, N.; Meng, X.; Liu, H. Exploring geospatial data visualization based on python. In Proceedings of the 2022 2nd International Conference on Control and Intelligent Robotics, Nanjing, China, 24–26 June 2022; pp. 858–862.
35. Sun, K.; Zhu, Y.Q.; Pan, P.; Hong, Z.W.; Wang, D.X.; Li, W.R.; Song, J. Geospatial data ontology: The semantic foundation of geospatial data integration and sharing. *Big Earth Data* **2019**, *3*, 269–296. [CrossRef]

36. Mena, M.; Corral, A.; Iribarne, L.; Criado, J. A progressive web application based on microservices combining geospatial data and the internet of things. *IEEE Access* **2019**, *7*, 104577–104590. [CrossRef]
37. Anthopoulos, L.; Janssen, M.; Weerakkody, V. A Unified Smart City Model (USCM) for smart city conceptualization and benchmarking. *Int. J. Electron. Gov. Res.* **2016**, *12*, 247–264. [CrossRef]
38. Wen, Y.N.; Chen, M.; Yue, S.S.; Zheng, P.B.; Peng, G.Q.; Lu, G.N. A model-service deployment strategy for collaboratively sharing geo-analysis models in an open web environment. *Int. J. Digit. Earth* **2017**, *10*, 405–425. [CrossRef]
39. Fricke, A.; Asche, H. Geospatial database for the generation of multidimensional virtual city models dedicated to urban analysis and decision-making. In Proceedings of the Computational Science and Its Applications—ICCSA 2019: 19th International Conference, Saint Petersburg, Russia, 1–4 July 2019; pp. 711–726.
40. Wan, W.; Du, X.Q.; Zhao, X.W.; Yang, Z.K. A cloud-enabled collaborative hub for analysis of geospatial big data. In Proceedings of the 2021 IEEE 6th International Conference on Cloud Computing and Big Data Analytics (ICCCBDA), Chengdu, China, 24–26 April 2021; pp. 1–5.
41. Scheider, S.; Nyamsuren, E.; Kruiger, H.; Xu, H.Q. Geo-analytical question-answering with GIS. *Int. J. Digit. Earth* **2021**, *14*, 1–14. [CrossRef]
42. Liu, F.T.; Ting, K.M.; Zhou, Z.H. Isolation Forest. In Proceedings of the 2008 Eighth IEEE International Conference on Data Mining, Pisa, Italy, 15–19 December 2008; pp. 413–422.
43. Papathanasiou, J.; Ploskas, N.; Papathanasiou, J.; Ploskas, N.T. *Multiple Criteria Decision Aid: Methods, Examples and Python Implementations*; Springer: London, UK, 2018; Volume 136, pp. 1–30.
44. Burns, B.; Grant, B.; Oppenheimer, D.; Brewer, E.; Wilkes, J. Borg, Omega, and Kubernetes. *Commun. ACM* **2016**, *59*, 50–57. [CrossRef]
45. Wen, Z.Y.; Lin, T.; Yang, R.Y.; Ji, S.L.; Ranjan, R.; Romanovsky, A.; Lin, C.T.; Xu, J. GA-Par: Dependable microservice orchestration framework for geo-distributed clouds. *IEEE Trans. Parallel Distrib. Syst.* **2020**, *31*, 129–143. [CrossRef]