A Containerized Service-based Integration Framework for Heterogeneous Geospatial Models

Lilu Zhu *, Yang Wang *, Yunbo Kong, Yanfeng Hu, Kai Huang

Posted Date: 13 November 2023

doi: 10.20944/preprints202311.0752.v1

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

Article

A Containerized Service-Based Integration Framework for Heterogeneous Geospatial 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
[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, wangyang003839@aircas.ac.cn

Abstract: With the rapid development of Earth observation and information technology, people are increasingly able to access geospatial models. Geospatial models, based on principles of geography, utilize mathematical, statistical, as well as computer science methods to interpret and predict geographic phenomena. These models can be applied in the fields such as urban planning, environmental protection, traffic management to help decision-makers solve geography-related problems. However, integrating different geospatial models to collaboratively solve complex geographic problems still faces significant obstacles due to heterogeneity in model structure, dependencies, and running modes. In this study, we propose a containerized service-based integration framework for heterogeneous geospatial models (GeoCSIF). GeoCSIF consists of three main components: (1) Model encapsulation. It breaks down complicated geospatial models into independently manageable model units, and builds as unified service packages with a templated constraint method. (2) Model orchestration. It achieves an optimal combination of large-scale models with complex dependencies using a prioritization-based orchestration method. (3) Model publication. It incorporates heuristics into the model scheduling process, which can provide adaptive deployment for different model runs. Finally, a prototype system was developed to validate the effectiveness and progressiveness of GeoCSIF by the integrating process of heterogeneous flood disaster models.

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

1. Introduction

In view of real-world phenomena or entities, models are the abstraction and simulation of the characteristics, states, structures, attributes, and changing laws of a research object [1]. Models can be divided into physical, mathematical, structural, and simulation models according to their presentation [2]. Geospatial models are generally of two types: mathematical models and simulation models. Geospatial mathematical models are constructed using mathematical logic methods for the calculation, analysis, and reasoning of geographical objects, such as in Figure 1a the spatial slope and aspect analysis model [3], Figure 1b the spatial decision-making support model [4], and Figure 1c the three-dimensional spatial measurement model [5]. Geospatial simulation models utilize computer algorithms and visualization techniques to simulate the running process and changing trends of geographical processes or phenomena, such as Figure 1d the satellite attitude simulation model [6], Figure 1e the Earth tide change expansion model [7], and Figure 1f the three-dimensional target simulation model [8]. In recent decades, geospatial models have played an important role in scientific research, global sustainability exploration, and commercial location-based applications. In scientific research, geospatial models can provide tools for the visualization and analysis of various phenomena and processes on Earth. They help explain the spatial relationships between the natural environment, ecosystems, and human activities. In global sustainability exploration, geospatial models can help governments and decision-makers in regional planning, resource management and environmental protection. In commercial location-based applications, geospatial models are being gradually applied to areas such as business location, market analysis and epidemic assessment.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 November 2023 doi:10.20944/preprints202311.0752.v1

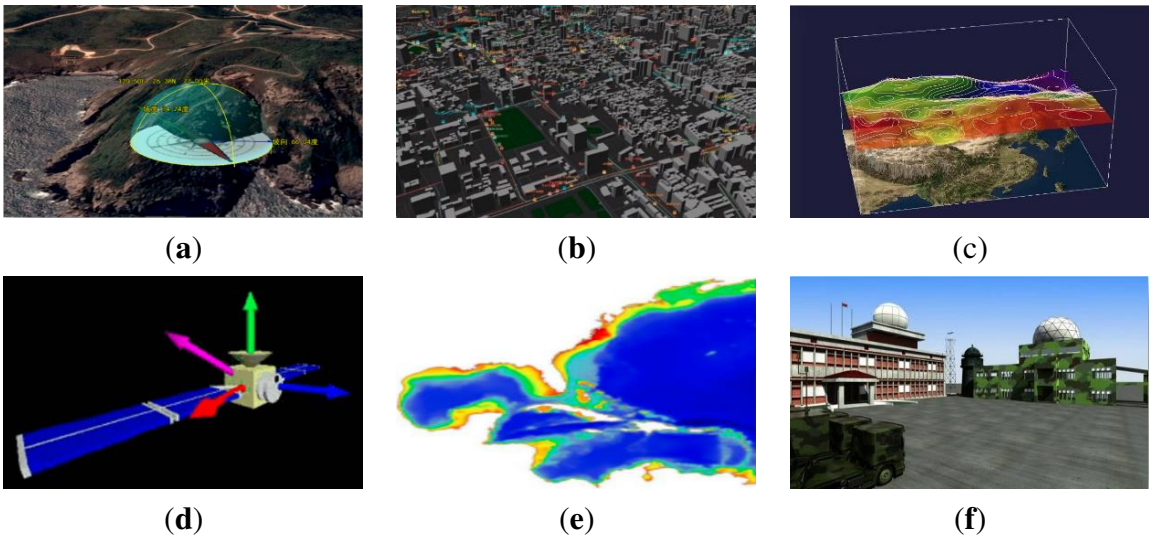


Figure 1. Illustrations of typical geospatial mathematical models: (**a**) Spatial slope and aspect analysis; (**b**) Spatial decision-making support; (**c**) Three-dimensional spatial measurement; (**d**) Satellite attitude simulation; (**e**) Earth tide change expansion; (**f**) Three-dimensional target simulation.

In recent years, advances in Earth observation technology have provided a more convenient and efficient channel for obtaining high-resolution geospatial data. However, the traditional monomer model can no longer meet the demands of users for complex business processing. For example, in large environmental and urban application scenarios, there is a need to support large-scale geospatial models. These models involve multiple factors, such as terrain, climate, and land use, which require multidimensional analysis. However, due to the difficulty of considering all factors in a single model, adopting model integration approaches can better combine the strengths of different models, thereby providing more comprehensive, accurate, and reliable analysis results. Early integration methods of nested, modular and model libraries have been successfully applied to geospatial problem analysis and solution systems [9]. Meanwhile, the proposal of some standards such as CityGML, KML, COLLADA, GeoJSON reduces the difficulty of interoperating and exchanging data between different models. However, the heterogeneity barrier limits models from being published solely on standalone servers or in closed networks, which prevents large-scale integration and information fusion [10].

We find that the heterogeneities of geospatial models are reflected in three main aspects: (1) Heterogeneity of model structures: Geospatial models usually involve different development languages, engineering architectures, and operating modes, resulting in diverse requirements for runtime environments. Owing to operating system or program dependency library conflicts, traditional methods make it difficult to deploy different models on the same server [11]. (2) Heterogeneity of model dependencies: As the complexity of geospatial business increases, the application systems continue to expand. geospatial applications such as Smart City or Digital Earth, may be composed of hundreds or even tens of thousands of tiny functional geospatial models. In such case, the sharing, discovery and collaboration of geospatial models between different systems or platforms become particularly complex. Traditional manual or script-based integration methods are not only time-consuming, but also difficult to discover high-quality models, thus affecting overall model service performance [12]. (3) Heterogeneity of model-running modes: Task diversity and computational intensity result in a wide variety of running modes. According to the running characteristics of the tasks, geospatial models can be categorized into offline processing, data access, edge preprocessing, real-time analysis, etc. According to the affinity relationship between tasks and resources, they can be categorized as CPU-intensive, memory-intensive, IO-intensive, and so on. Because the demand for underlying resources varies significantly, traditional model publishing methods are prone to skewing environmental resources and affecting service performance [11, 13].

In this study, we designed a containerized service-based integration framework (GeoCSIF) for large-scale geospatial models. The framework is divided into three main components: (1) The model

 doi:10.20944/preprints202311.0752.v1

encapsulation component designs model servicized structure for the characteristics of model structural heterogeneity. It employs a templated constraint method to build heterogeneous geospatial models as a unified model service package. (2) The model orchestration component designs a prioritization-based orchestration method for the characteristics of model-dependency heterogeneity. This enables optimal combination and capability integration of large-scale geospatial models. (3) The model publication component designs a heuristic scheduling method for the characteristics of the running mode heterogeneity. This can enhance the adaptability of the models to the cloud environment and improve the quality of service.

2. Design of the framework

2.1. Advances of geospatial models

This section illustrates the evolution of geospatial models, and summarizes the advantages and disadvantages of traditional geospatial model structures and model integration methods.

From the perspective of system integration, traditional geospatial model structures can be divided into monomer, component, and dependent structures [14-16]. The monomer structure integrates all functions and data from the geospatial information system (GIS). It is simple to develop and convenient to deploy, but inevitably produces significant functional redundancy [17]. The component structure develops the functions of geospatial 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 [18], but does not support cross-platform heterogeneous environments. The dependency structure develops the functions of geospatial models as micro-services and couples them in a lightweight manner [19], thus addressing the cross-platform barrier of the component structure. However, none of these model structures can be applied directly to geospatial models. This will increase the complexity of model structure representation owning to the superiority of geospatial applications over modeling dimensionality.

According to different coupling methods, geospatial model integration can be categorized into nested, modular, model library, and service-based integrations [9]. Nested integration includes the source code and function library in terms of implementation [20]. In this method, geospatial models are used as functional items in GIS. It enables seamless integration of geospatial models with GIS, but model heterogeneity results in significant integration costs. Modular integration includes both executable and middleware [21,22]. In this method, GIS and geospatial models exist independently, and are invoked by pre-defined modules or interface protocols. This method is relatively simple and requires a low-cost development. Nevertheless, the communication costs between systems increase owning 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 [10]. Qiao et al. integrated the container isolation mechanism into OpenGMS to eliminate the barrier of environmental heterogeneity for multisource model integration [11].

It is noteworthy that although the research on geospatial 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 models with heterogeneity in structures, dependencies, and running modes. Continuous improvements are required to accommodate the model reliability and scalability requirements. However, the proposed

 doi:10.20944/preprints202311.0752.v1

containerized service-based integration framework can effectively solve the rapid integration problem of large-scale heterogeneous geospatial models by designing a standardized process.

2.2. Containerized service-based integration framework

In this section, we describe the design of a containerized service-based integration framework named GeoCSIF, as shown in Figure 2. GeoCSIF consists of three types of components: model encapsulation, orchestration, and publication. GeoCSIF can reduce the complexity of integrating and publishing heterogeneous geospatial 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 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 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 reduce 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.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 November 2023 doi:10.20944/preprints202311.0752.v1

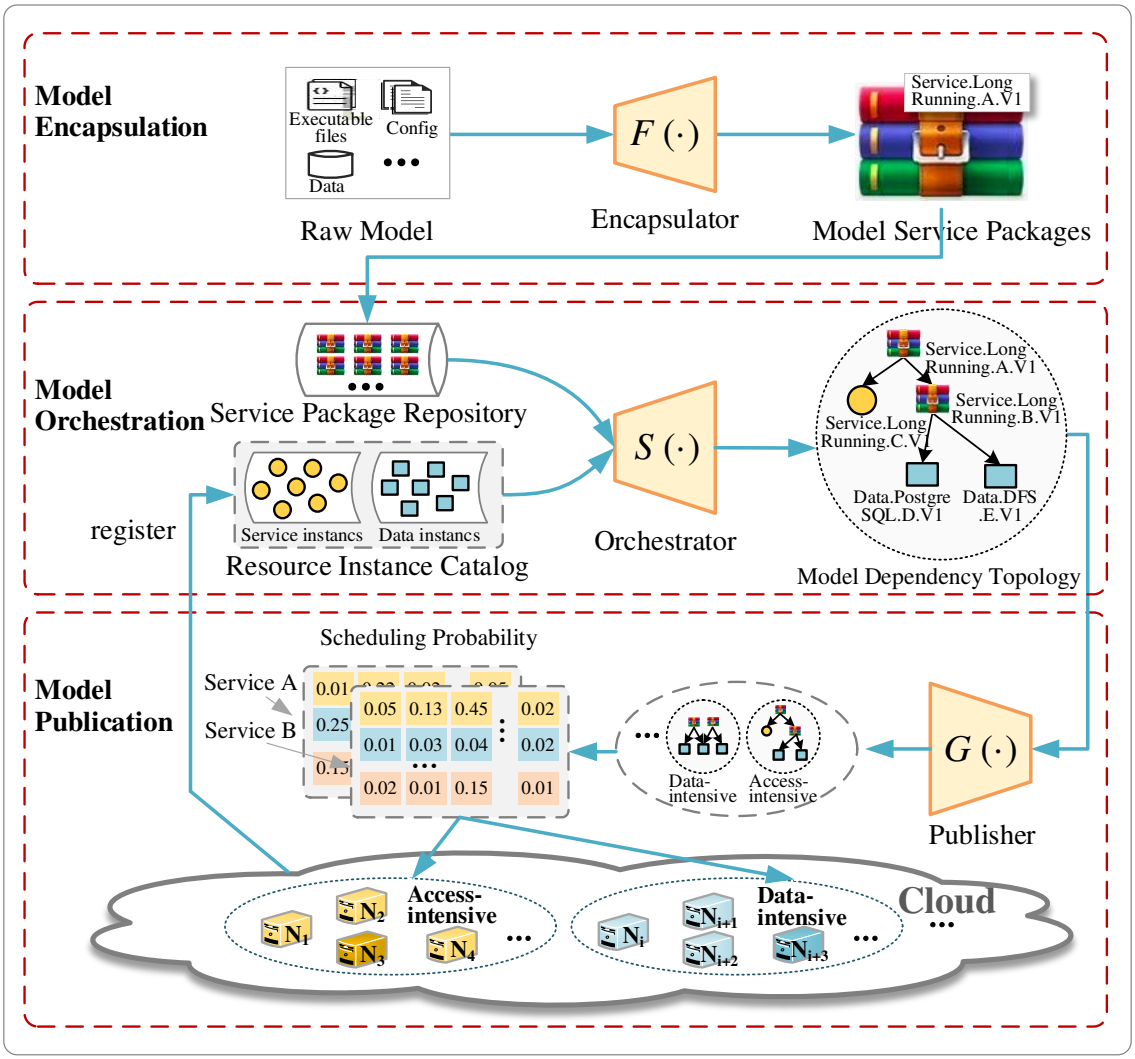


Figure 2. Overview of containerized service-based integration framework for heterogeneous geospatial models (GeoCSIF).

3. Design of the component

3.1. Servicized encapsulation of geospatial 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 unified model service package. This enables rapid publishing of geospatial models in the container-based cloud environments.

3.1.1. Servicized declaration of geospatial models

Publishing and sharing geospatial models usually require significant work, mainly owing to their complexity and disciplinary details [10]. On the one hand, geospatial 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

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 November 2023 doi:10.20944/preprints202311.0752.v1

as unified services. Therefore, structural heterogeneity must be eliminated to support open sharing and effective execution of geospatial models.

To address this issue, we propose a cloud-oriented model servicized structure, as shown in Figure 3. This model servicized structure consists of model meta-information, model business information, interface contracts, resource dependencies, and a virtualized runtime environment. Model meta-information is the basic information of geospatial models, such as name, creator, description of the model. Model business information refers to the domain information related to the business capabilities of the model, such as reference system, resolution and product level. Interface contracts are interface standards or protocols used to describe and define the interfaces that geospatial models follow when exchanging and sharing data between different systems, software or platforms. It includes standard OGC specifications and general 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 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 models, making the model integration and sharing more convenient and efficient.

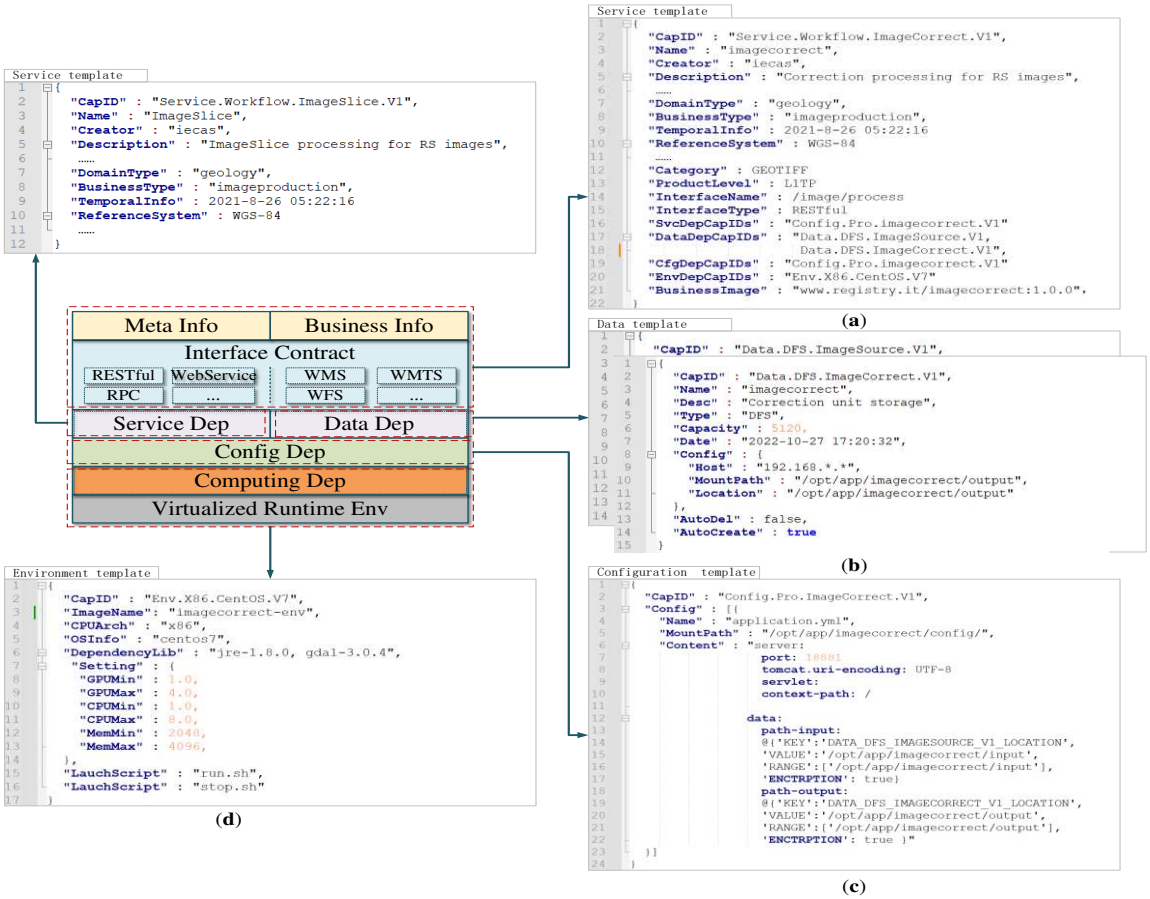


Figure 3. Model serviced structure with declarative templates: (**a**) Service template; (**b**) Data template; (**c**) Configuration template; (**d**) Environment template.

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 JSON format is shown in Figure 3. 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

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 November 2023 doi:10.20944/preprints202311.0752.v1

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, owning to the wide range of sources of geospatial 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.

As shown in Figure 4, 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 models and their dependent resources to construct declarative templates. The detailed steps are as follows:

Step 1: Resource classification. As shown in Figure 4 (left), we categorized the composition of geospatial 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 operating system (OS).

Step 2: Rules definition. As shown in Figure 4 (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 by 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 by 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, c_{32} and c_{33} primarily verify the availability of storage access addresses. C_4 verify the compliance of the configuration template. Configuration templates are generated by inserting special placeholders into the raw configuration files of geospatial models. This special placeholders in the form of @{KEY, VALUE, RANGE, ENCRYPTION}, which are used to identify configuration items that are open for modification. By constraining the values of configuration items to prevent 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.

Step 3: Hybrid constraint rule processing. As shown in Figure 4 (middle), we establish a mapping between the resource types and constraint rules, which in turn regularizes the declarative

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 November 2023 doi:10.20944/preprints202311.0752.v1

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 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 "SvcDepCapIDs", "DataDepCapIDs", "CfgDepCapIDs", "EnvDepCapIDs" fields of the service template, as show in Figure 3a. 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 "Config" of the data template, as show in Figure 3b. For models with configuration dependencies, c_{41}-c_{44} are used to verify the compliance of the configuration template, as show in Figure 3c. 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 3d. 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.

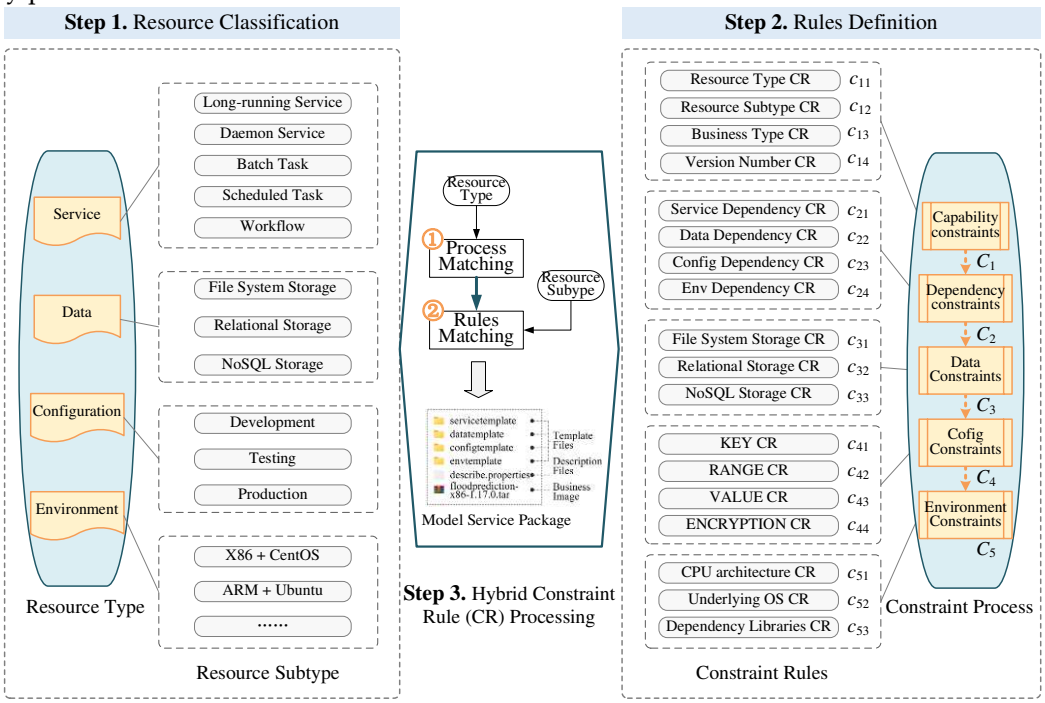


Figure 4. Principle of templated constraint method $F(\cdot)$.

3.2. Prioritization-based orchestration of geospatial models

Model orchestration refers to the assembly of models, data, configurations, and runtime environment that the current model according to the dependencies between model service packages. The open sharing of geospatial models significantly improves the efficiency of model integration and publication. However, owning 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 weighted a sum of QoS attributes to evaluate and screen resources. These methods are simple and feasible but ignore the relationship between multi-

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 November 2023 doi:10.20944/preprints202311.0752.v1

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 5.

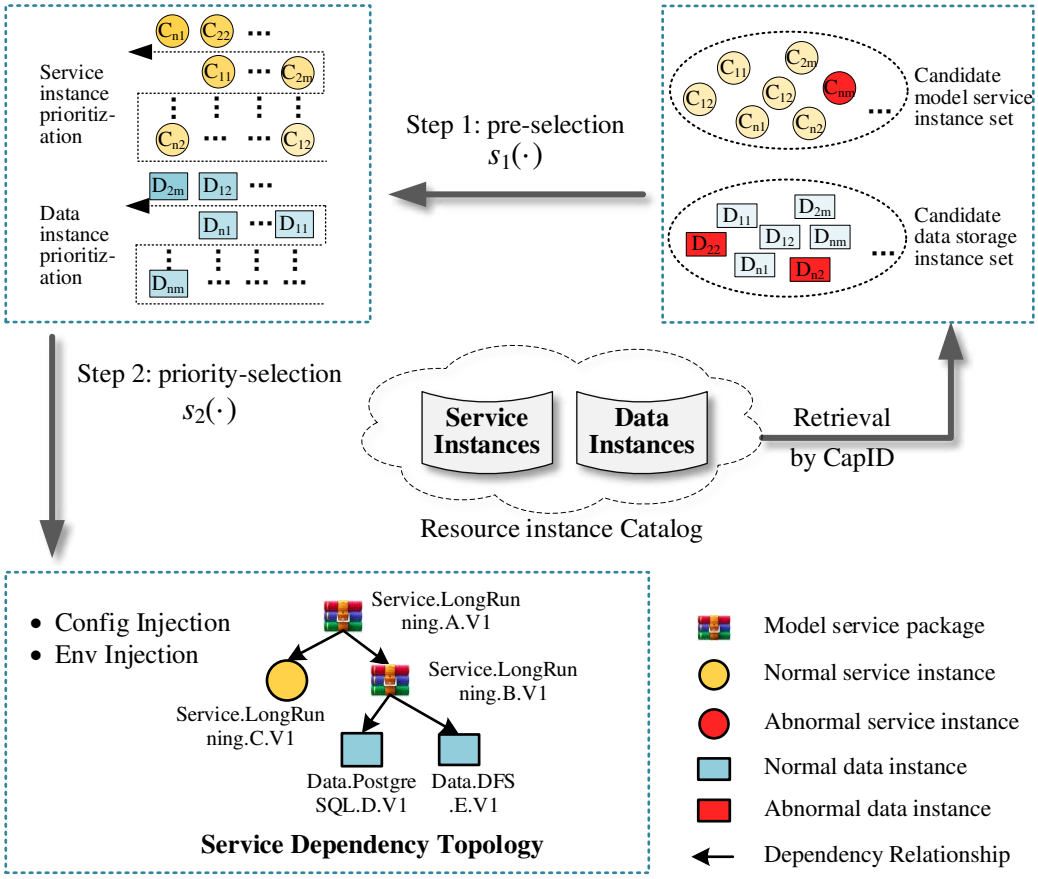


Figure 5. Flow of prioritization-based orchestration method $S(\cdot)$.

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. Owning to the large number of candidate resource instances with varying performance, the following two-step screening was performed.

Step 1: Dependency pre-selection $s_1(\cdot)$. Abnormal instances in the candidate resource sets were eliminated using an isolation forest (iForest) [24]. 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 given by 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).

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

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 November 2023 doi:10.20944/preprints202311.0752.v1

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.

Step 2: Dependency 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 [25] by entropy weighting method, as shown in Formula (3). This can eliminate the influence of traditional subjective weights on the accuracy evaluation of resources.

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

where DM denotes the resource decision matrix of dimension $n \times m$ and its row vector is the resource decision vector $D_i (0 \le i \le n\text{-}1)$. 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, $p_{ij} = d_{i,j} / \sum_{j=1}^{n} d_{i,j}$ $(0 \le i \le n\text{-}1, 0 \le j \le m\text{-}1)$ is the normalized probability, $d_{i,j}$ is an element of vector D_i. c denotes the resource evaluation vector with dimensions $n \times 1$. 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 $rt_i\text{-}\delta$ to the service response time rt_i, that is, $sla_i = (rt_i\text{-}\delta) \,/\, rt_i$. err_i denotes the service error rate, that 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, $iops_i$ 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 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

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 November 2023 doi:10.20944/preprints202311.0752.v1

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 6 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 by 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 [23]. 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.

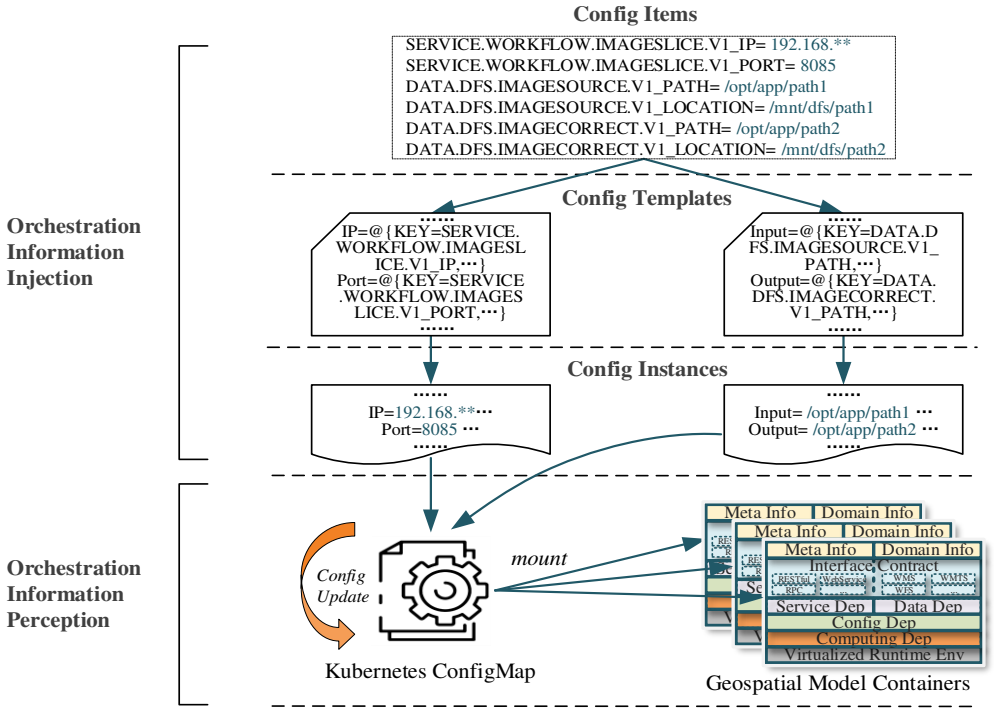


Figure 6. Principle of orchestration information injection and perception.

3.3. Adaptive publication of geospatial models

On the basis of dependency orchestration, the model publication component further publishes geospatial models as model services. Owing to the fact that geospatial models from 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 model feature parameterization sub-method $g_1(\cdot)$ and 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.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 November 2023 doi:10.20944/preprints202311.0752.v1

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 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 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 7. The flow is as follows:

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 by the similar model requirements resources Q_i^2. Coordination coefficient α is employed to balance the two. In the evaluation of model similarity, we categorized geospatial 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).

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

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 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.

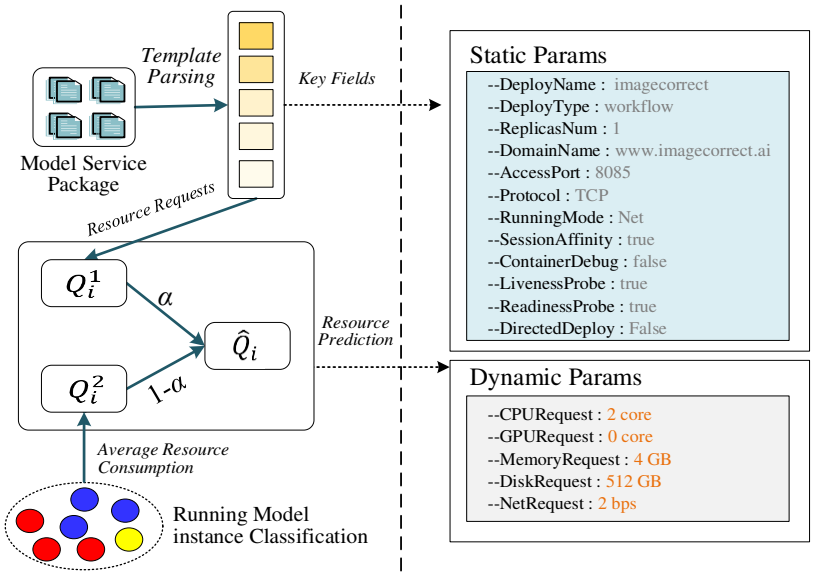


Figure 7. Flow of model feature parameterization sub-method $g_1(\cdot)$.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 November 2023 doi:10.20944/preprints202311.0752.v1

3.3.2. Heuristic model scheduling policies

Geospatial models are characterized by task and data diversity, with significant differences in resource and performance requirements between models. For example, the portal applications of earth systems have the characteristics of high concurrency and intensive access. They can be deployed in cluster environments with a high bandwidth and low latency. Earth 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 scene 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 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 models.

As shown in Figure 8, 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).

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

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 models are scheduled to the nodes with maximum scheduling probability $h(i,j)$.

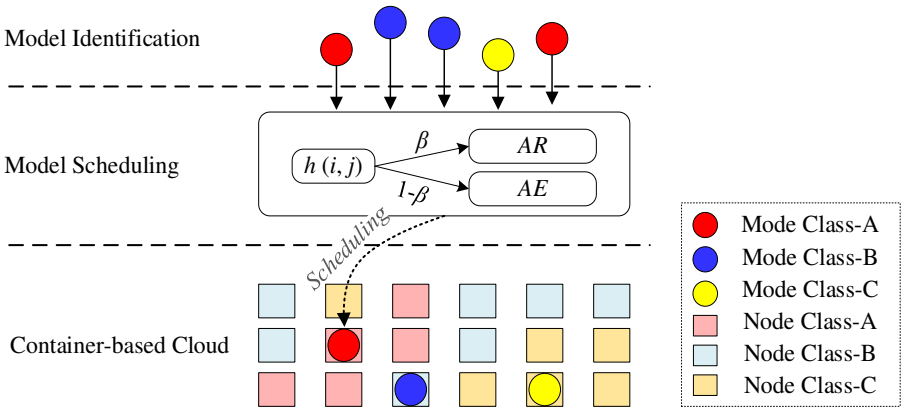


Figure 8. Flow of heuristic policies sub-method $g_2(\cdot)$.

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 bottom to top, as shown in Figure 9. The

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 November 2023 doi:10.20944/preprints202311.0752.v1

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 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 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 models in the form of HTTP RESTful, including the model information retrieval, resource evaluation pushing, service running control interfaces. The presentation layer provides users with human-computer interaction views, including model management, orchestration management, and service management.

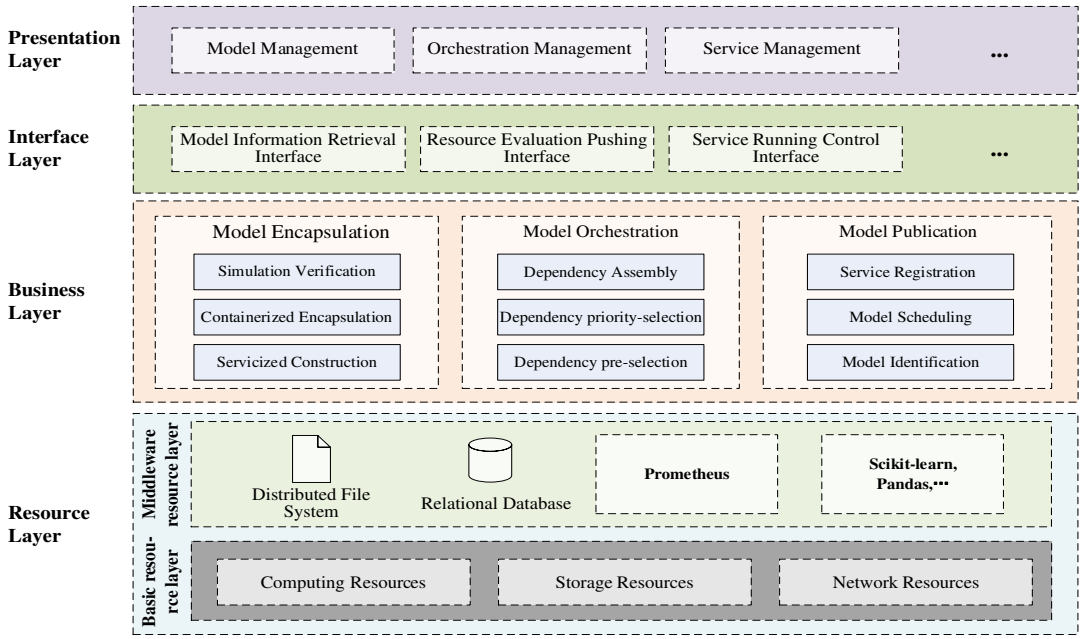


Figure 9. System architecture of GeoCSIF.

Figure 10 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.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 November 2023 doi:10.20944/preprints202311.0752.v1

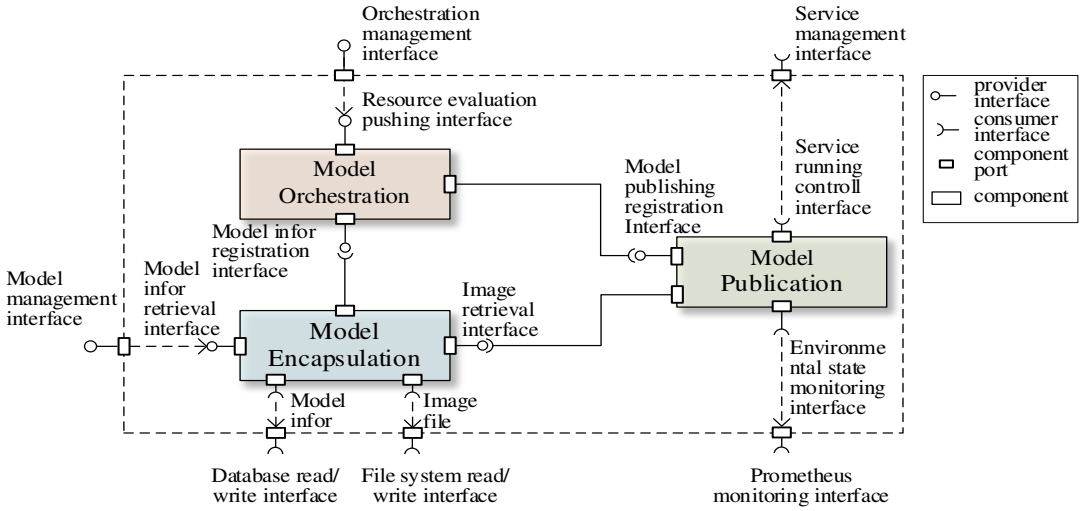


Figure 10. Component invocation relationships of prototype system.

4.2. System implementation and function introduction

In this section, we implemented a prototype GeoCSIF system with human-machine interface, as shown in Figures 11–13.

Figure 11 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 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.

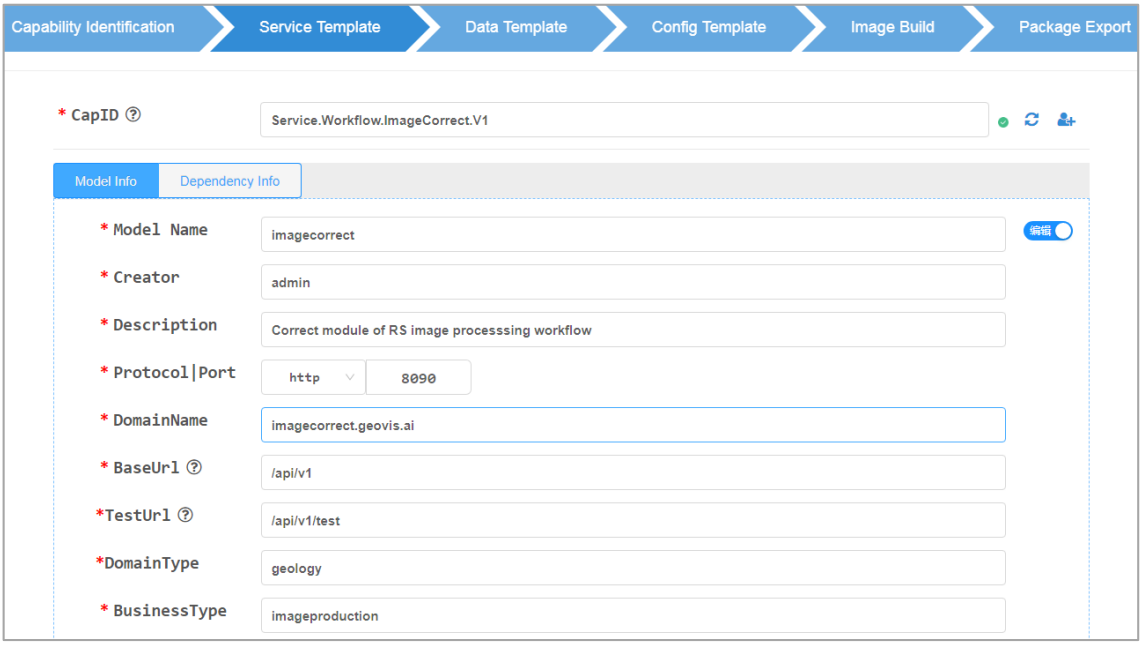


Figure 11. Human-machine interface of model encapsulation component.

Figure 12 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 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

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 November 2023 doi:10.20944/preprints202311.0752.v1

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 13). Note that the static deployment parameters were automatically populated by extracting key fields from the service template of geospatial 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 reuse support.

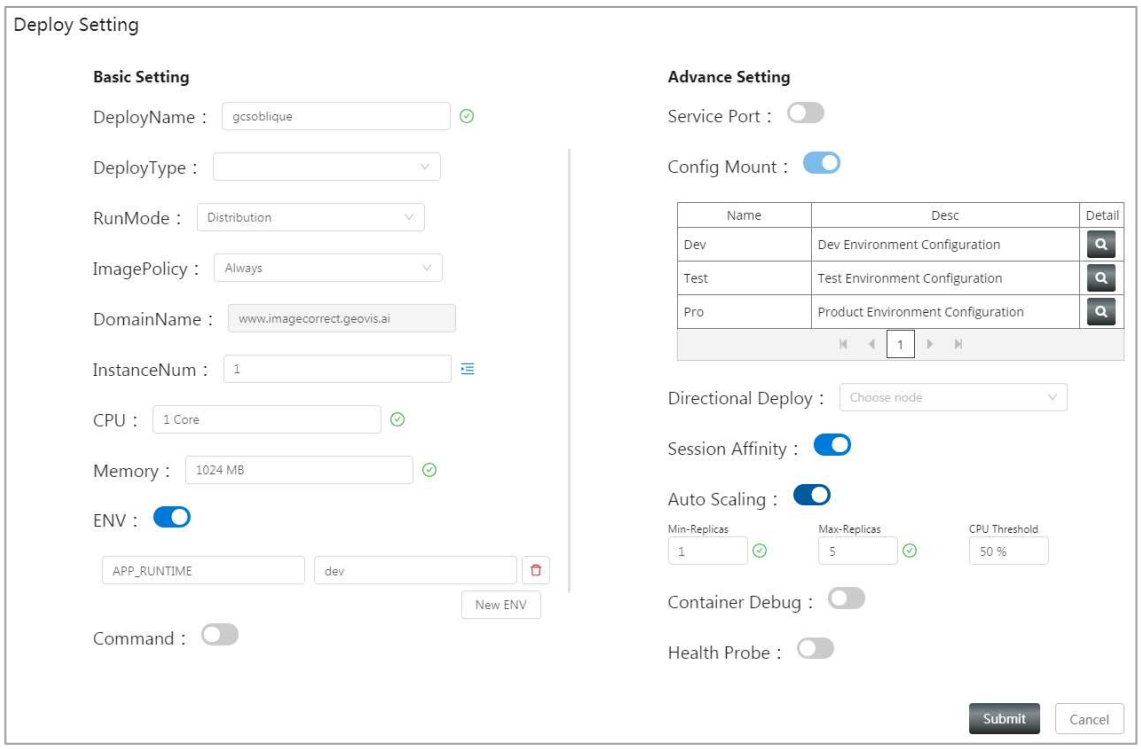


Figure 12. Human-machine interface of model orchestration component.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 November 2023 doi:10.20944/preprints202311.0752.v1

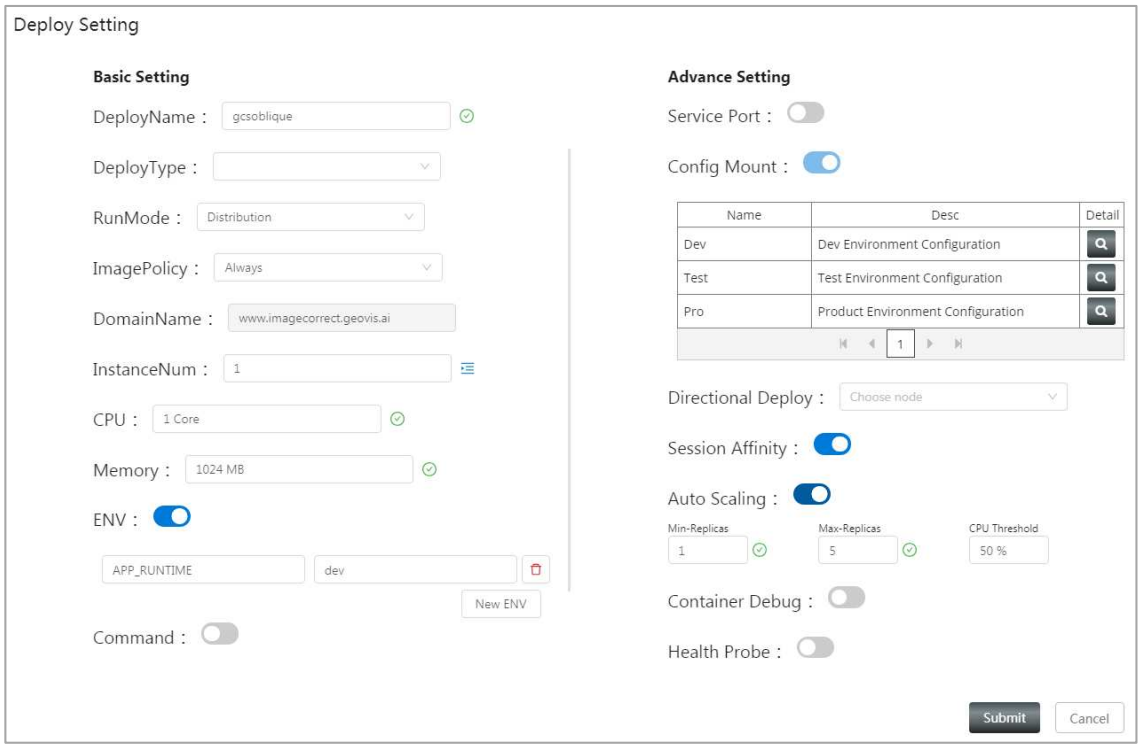


Figure 13. Human-machine interface of model publication component.

5. Case Studies

In this section, we introduce the GeoCSIF process with a practical case. Moreover, the effectiveness of the framework was evaluated through 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.

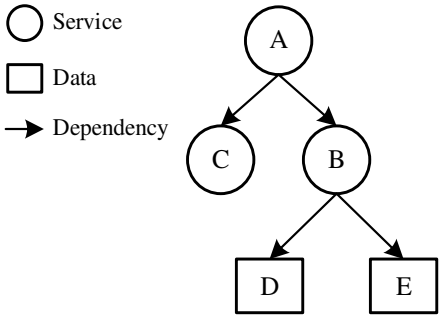
5.1. geospatial models for flood disaster prediction

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 14a. 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) Owning 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-

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 November 2023 doi:10.20944/preprints202311.0752.v1

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, which are difficult to cope with using traditional methods. These models rely on different operating systems, dependent libraries, and Python environments, as shown in Figure 14b. It can be difficult to deploy multiple models on the same server owning 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 14. flood disaster prediction models: (**a**) Model dependencies; (**b**) Model details.

5.2. Practical verification of geospatial model integration

In this section, the effectiveness of GeoCSIF is first verified by using the flood disaster models shown in Figure 14. 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 2, the integration process includes: (1) Declare capability identifiers for each model, construct service, data, configuration, and environment templates, build business images and encapsulate 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 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) Access and validate the availability of flood disaster prediction models.

As show in Figure 15, 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

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 November 2023 doi:10.20944/preprints202311.0752.v1

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 models.

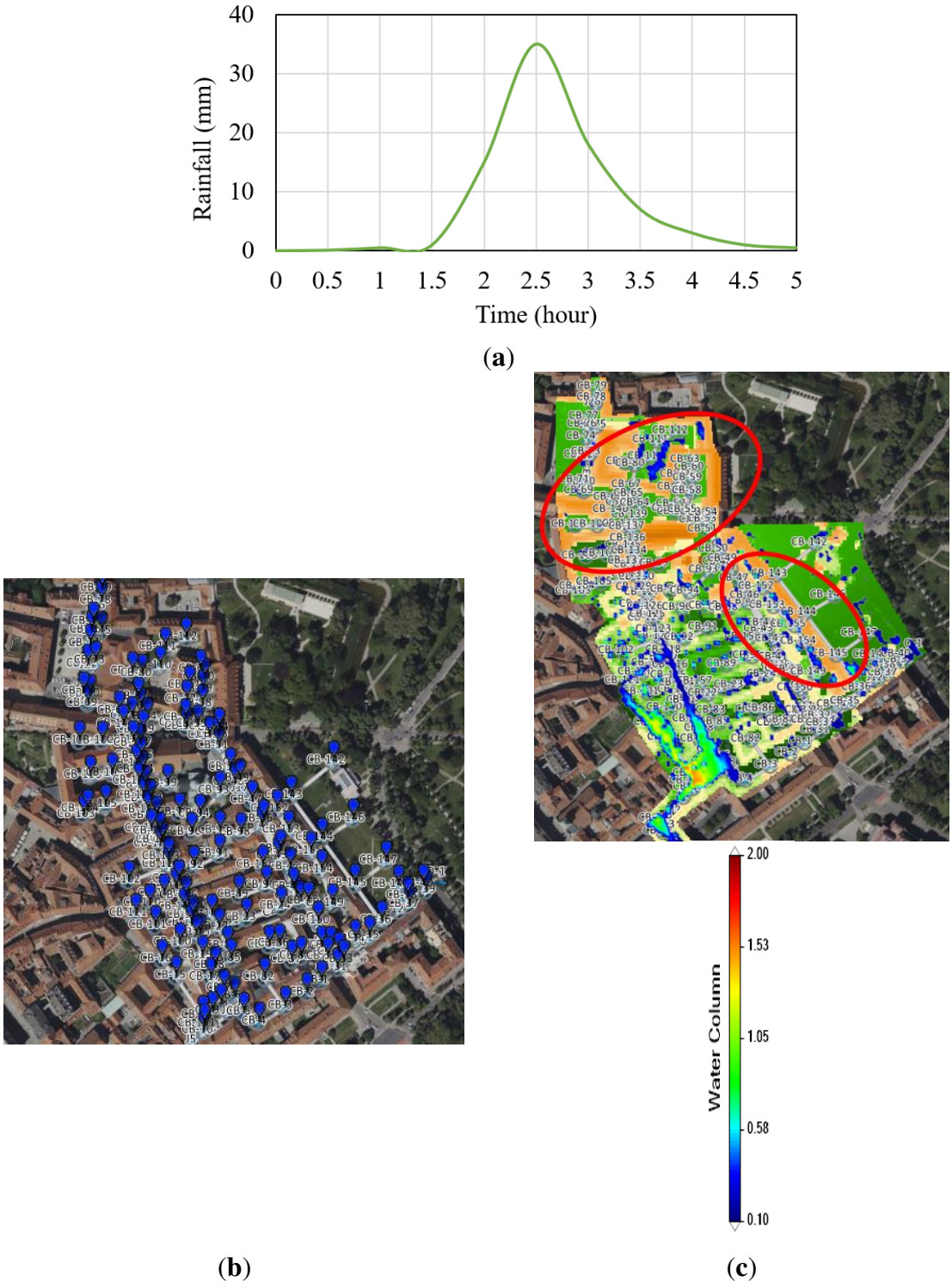


Figure 15. Results of models in flood prediction: (**a**) Simulation of urban heavy rainfall; (**b**) Underground drainage network; (**c**) Surface flood distribution.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 November 2023 doi:10.20944/preprints202311.0752.v1

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 16 a,b, the native structure for traditional standalone deployment includes a dependency library, executable file, and install script. The proposed servicized structure for cloud deployment includes service template, data template, configuration template, environment template, business image and description file with a more fine-grained model structure.

Figure 16. Comparison of model structures: (**a**) Native structure; (**b**) Servicized structure.

Figure 17 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 models in a modular way and encapsulate them by using virtualization with separation of business and environment. That is, the larger running environment and dependent libraries are not encapsulated in the model service package, but rather 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.

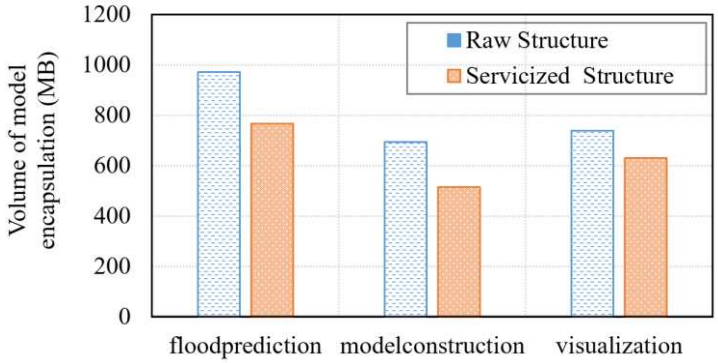


Figure 17. Comparison of model volume under different encapsulation methods.

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 [26]. The experimental results are shown in Figure 18a–d. We compared the orchestration completion time, service response time, throughput, and SLA violation rate under different orchestration methods. (a) 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, (b)-(d) show that the model has better QoS (quality of service) performances under the prioritization-based orchestration. Compared to GA-Par, the response time,

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 November 2023 doi:10.20944/preprints202311.0752.v1

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.

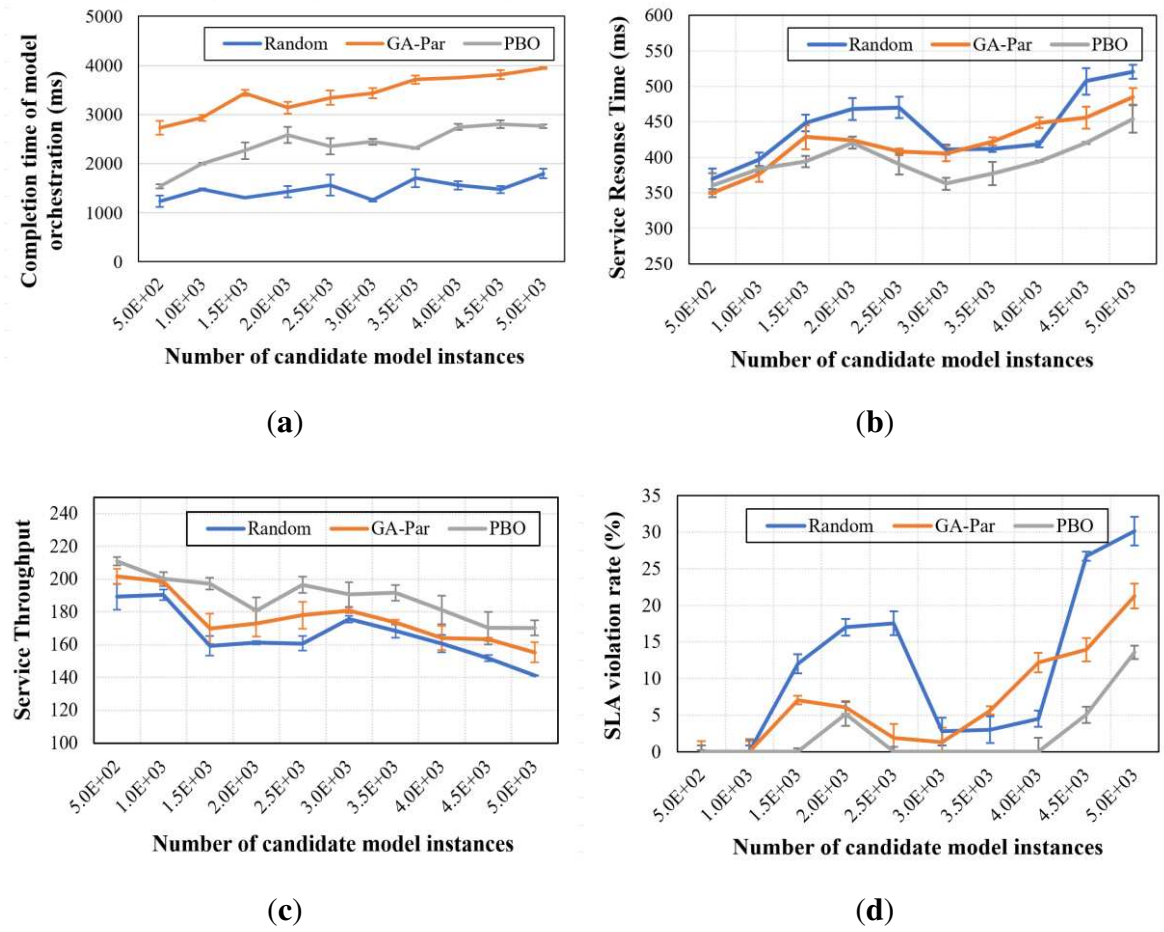


Figure 18. Comparison of model orchestration performance: (**a**) Completion time of model orchestration; (**b**) Service response time; (**c**) Service throughput; (**d**) SLA violation rate.

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 comparison 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 (1min) conforms to the Poisson distribution, and simulates model arrival time at a frequency of λ=50. The experimental results are shown in Figure 19. Where, (a) shows that the resource utilization under HMS is more balanced. Numerical analysis shows that compared with Random and MRSF, cluster resource utilization under HMS increases by 15.7% and 10.3%. (b) 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.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 November 2023 doi:10.20944/preprints202311.0752.v1

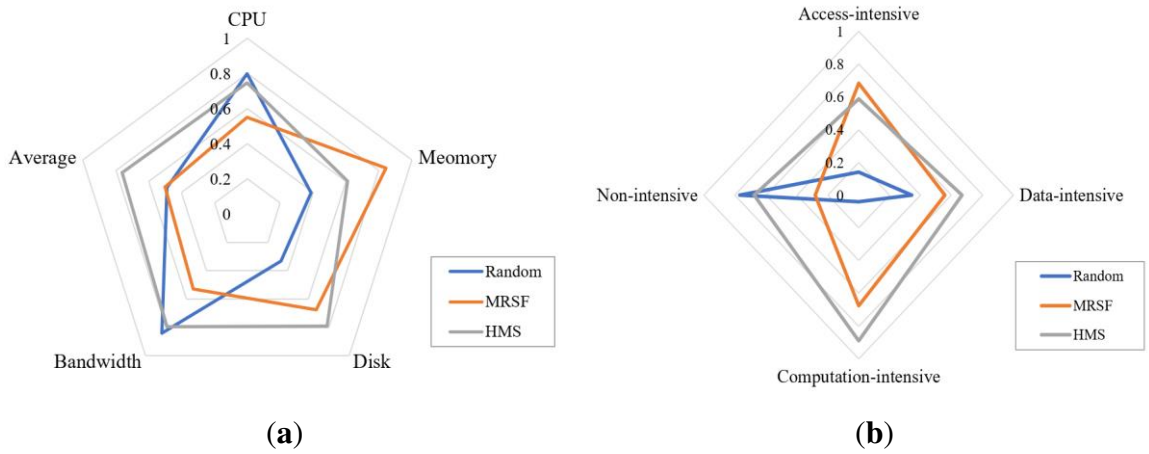


Figure 19. Comparison of scheduling performance: (**a**) Resources utilization; (**b**) Matching rate of models to resources.

6. Conclusions and future work

This study proposes a method for publishing multisource heterogeneous geospatial models as model services in container-based cloud environments. This is of great significance for the reuse and sharing of multi-disciplinary and cross-domain geospatial models. All types of resources can be integrated more conveniently by using the proposed containerized service-based integration framework. First, the model encapsulation component employs a templated constraint method to encapsulate heterogeneous geospatial models in unified service packages. Then, aiming at the dependency heterogeneity of multisource geospatial models, the model orchestration component adopts a prioritization-based orchestration method to realize the automatic discovery and optimal combination of model-dependent resources. Finally, the model publication component provides a heuristic scheduling method in response to the variability of the model running modes and the uncertainty of the model runtime environments. Although much work has been conducted on heterogeneous geospatial model integration, the proposed framework still needs to be continuously deepened to cope with more complex application scenarios. Future work will involve cross-scale and cross-temporal model integration, uncertainty modeling and inference, and deepening and expansion of application domains. These studies will help further improve the accuracy, robustness, and application effectiveness of multisource heterogeneous geospatial 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 & editing, Yang Wang.

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 conflict of interest.

References

1. Zhu, C.Q.; Shi, W.Z. *Spatial analysis modelling and principle*; Science Press: Beijing, China, **2006**; pp. 15–18.
2. Zhao, X.Y. *System modelling and simulation*; National Defense Industry Press: Beijing, China, **2015**; pp. 3–4.
3. Bennie, J.; Huntley, B.; Wiltshire, A.; Hill, M. O. Baxter, R. Slope, aspect and climate: Spatially explicit and implicit models of topographic microclimate in chalk grassland. *Ecological Modelling*. **2008**, 216, pp. 47-59.
4. Ham, Y.; Kim, J. Participatory sensing, and digital twin city: Updating virtual city models for enhanced risk-informed decision-making. *Journal of Management in Engineering*. **2020**, 36, pp. 1-12.
5. Ahmad, M. A.; Tvoroshenko, I.; Baker, J. H.; Kochura, L.; Lyashenko, V. Interactive geoinformation three-dimensional model of a landscape park using geoinformatics tools. *International Journal on Advanced Science Engineering Information Technology*. **2020**, 10, pp. 2005-2013.

6. Narkiewicz, J.; Sochacki, M.; Zakrzewski, B. Generic model of a satellite attitude control system. *International Journal of Aerospace Engineering*. **2020**, pp.1-17.
7. Yan, J.; Zhao, S.Y.; Su, F.Z.; Du, J.X.; Feng, P.F.; Zhang, S.X. Tidal flat extraction and change analysis based on the RF-W model: A case study of Jiaozhou bay, east China. *Remote Sensing*. **2021**, 13, pp. 1436-1453.
8. Guo, C.J.; Xu B.L.; Jing, T.; Zhang, Y.; Hu, H.R.; Wang, L.H. Three-dimensional visual perception technology and simulation of individual soldier in combat. In Proceedings of the 2021 28th International Conference on Geoinformatics, Nanchang, China, 3-5 Nov. **2021**; pp. 1–15.
9. Zhang, X.L.; Ren, Z.G.; Cao, Y.B. Research on seamless integration of spatial analysis model and GIS. *Geospatial Information*. **2014**, 12, pp. 156-158+12.
10. 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. *International Journal of Digital Earth*. **2013**, 6, pp 356-382.
11. 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. *International Journal of Digital Earth*. **2021**, 14, pp. 1067-1086.
12. 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. *International Journal of Digital Earth*, **2017**, 10, pp. 405-425.
13. Yue, S.S.; Chen, M.; Wen, Y.N.; Lu, G.N. Service-oriented model-encapsulation strategy for sharing and integrating heterogeneous geo-analysis models in an open web environment. *ISPRS Journal of Photogrammetry and Remote Sensing*, **2016**, 114, pp. 258-273.
14. Stoimenov, L.; Stanimirovi, A.; Djordjevi, S.; Kajan. Realization of component-based GIS application framework. *7th AGILE Conference on Geographic Information Science*. **2004**, pp. 113-120.
15. Liu, S.H.; Liu, L.P.; Yu, H.L. XML based GIS application model definition and description language GBMDL. *China Management Informationization*. **2009**, 12, pp.93-98.
16. Zhou, L.L.; Wang, R.J.; Cui, C.Y.; Xie, C.J. GIS application model based on cloud computing. *Network Computing and Information Security: Second International Conference*. **2012**, pp. 130-136.
17. Ou, S. J. Research and application of geographic information system development based on component architecture. PhD diss., Jilin University, Jilin, **2003**.
18. Jia, J. Research on key technologies of geographic information system development. *Construction & Design for Project*. **2022**, 9, pp. 142-144.
19. Dontsov, A. A.; Sutorikhin, I. A.; Development of a geographic information system for data collection and analysis based on microservice architecture. *CEUR Workshop Proceedings*. **2021**, 3006, pp. 280-287.
20. Xu, Z.H. GIS functional component library and functional integration. *Geomatics and Information Science of Wuhan University*, **2001**, 04, pp. 303-309.
21. 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. *24th International Conference on Geoinformatics*. **2016**, pp. 1-5.
22. Wang, S.H.; Zhong, Y.; Wang, E.Q. An integrated GIS platform architecture for spatiotemporal big data. *Future Generation Computer Systems*, **2019**, 94, pp. 160-172.
23. Burns, B.; Grant, D.; Oppenheimer; Brewer, E.; Wilkes, J. Borg, Omega, and Kubernetes. *Communications of the ACM*, **2016**, 59, pp. 50-57.
24. 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 Dec. **2008**; pp. 413–422.
25. Papathanasiou, J.; Ploskas, N.; Papathanasiou, J.; Ploskas, N. Topsis. Multiple Criteria Decision Aid: Methods, Examples and Python Implementations, **2018**, 136, pp. 1-30.
26. 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 Transactions on Parallel and Distributed Systems*. **2020**, 31, pp. 129-143.