International Journal of Geo-Information

MDPI

Article

A Dynamic Management and Integration Framework for Models in Landslide Early Warning System

Liang Liu [1,2], Jiqiu Deng [1,2,*] and Yu Tang [1,2]

[1] Key Laboratory of Metallogenic Prediction of Nonferrous Metals and Geological Environment Monitoring (Ministry of Education), Central South University, Changsha 410083, China
[2] School of Geosciences and Info-Physics, Central South University, Changsha 410083, China
* Correspondence: csugis@csu.edu.cn

Abstract: The landslide early warning system (LEWS) relies on various models for data processing, prediction, forecasting, and warning level discrimination. The potential different programming implementations and dependencies of these models complicate the deployment and integration of LEWS. Moreover, the coupling between LEWS and models makes it hard to modify or replace models rapidly and dynamically according to changes in business requirements (such as updating the early warning business process, adjusting the model parameters, etc.). This paper proposes a framework for dynamic management and integration of models in LEWS by using WebAPIs and Docker to standardize model interfaces and facilitate model deployment, using Kubernetes and Istio to enable microservice architecture, dynamic scaling, and high availability of models, and using a model repository management system to manage and orchestrate model-related information and application processes. The results of applying this framework to a real LEWS demonstrate that our approach can support efficient deployment, management, and integration of models within the system. Furthermore, it provides a rapid and feasible implementation method for upgrading, expanding, and maintaining LEWS in response to changes in business requirements.

Keywords: landslide; early warning; LEWS; model repository

Citation: Liu, L.; Deng, J.; Tang, Y. A Dynamic Management and Integration Framework for Models in Landslide Early Warning System. *ISPRS Int. J. Geo-Inf.* **2023**, *12*, 198. https://doi.org/10.3390/ijgi12050198

Academic Editors: Wolfgang Kainz, Pablo Rodríguez-González, Diego González-Aguilera

Received: 4 April 2023
Revised: 7 May 2023
Accepted: 12 May 2023
Published: 13 May 2023

1. Introduction

Landslide disaster investigation, evaluation, monitoring, and early warning aim to minimize the losses caused by disasters through effective prevention and control measures. In the widely used landslide risk management system, the monitoring and early warning are important technologies that complement engineering treatments and are integral parts of the system [1]. Establishing a landslide early warning system (LEWS) is a cost-effective approach compared to engineering treatment [2]. Over years of research, various LEWSs, and related technologies have been implemented in numerous countries and regions [3–10]. Landslide early warning can be classified into regional early warning and individual early warning [1]. Regional early warning relies on conducting a large-scale geological survey, acquiring and interpreting remote sensing data, and comprehensive analysis and evaluation of multiple data [11,12]. However, limited by the cycle, resolution, and accuracy of data acquisition, the results of regional early warning are not relevant, reliable, effective, and timely enough, and only a rough warning can be given. Individual early warning uses monitoring equipment located at the individual landslide mass and its surrounding area to monitor various landslide-inducing conditions (such as rainfall) and deformation characteristics (such as surface displacement, and crack size) [3]. The obtained monitoring data are then transmitted to the remote server where various models in LEWS quickly process and analyze them, and calculate the warning level, thus triggering an early warning. In comparison, the individual early warning has better performance in terms of relevance, reliability, validity, and velocity [13].

The LEWS is an intricate system encompassing numerous scientific aspects in its design, implementation, management, and validation [14]. According to the United Nations International Strategy for Disaster Reduction (UNISDR 2009) [8], a LEWS should consist of four procedures: monitoring, analysis and modeling, early warning, and emergency response. Among them, the prediction and modeling of landslide hazard occurrence and early warning are vital elements of LEWS [9]. Therefore, various models of these are the core of LEWS. The advancement of landslide prediction and forecasting technology, along with the application of computer technology, has allowed geological disaster researchers to develop numerous models. These models address monitoring data processing, landslide prediction, forecasting, and early warning (e.g., GM (1, 1) model [15], Saito's three-stage deformation model [16], artificial intelligence models [17]) and can be integrated into LEWS applications. It has significantly improved the application level and intelligence of LEWS.

Developing standardized early warning models is challenging, and deploying them is complex due to the diversity of programming languages (e.g., Python [12,18], Matlab [19], Java [13]), implementation methods (e.g., scripts, DLLs, functions, interfaces), and dependency environments. In previous studies, the prediction analysis and modeling components of LEWS existed as system modules, comprising multiple models, multiple computing stages, and the connection of all these models and stages. Moreover, due to the complexity of landslide prediction and forecasting problems, a single model is usually insufficient for reliable prediction and forecasting. Many relevant models are typically combined through generic early warning mechanisms to increase the accuracy of forecasting and prediction [20]. The different model implementations resulted in complex model invocation methods and strong coupling between models and system. Currently, there is limited research on a common mechanism for combining and connecting models to form new models, which would include modifying or replacing individual models and adjusting the connections within the combined models. This means modifying the source code of the system, which can lead to high costs and reduced system reliability.

The organization and integration of models in LEWS have received little attention in previous studies. A framework providing easy integration of new models would be beneficial for updating models in LEWS. Moreover, such a framework would enable free model selection, combination, and connection, allow user-defined construction of new models, and replace existing models rapidly and dynamically. Thus, it would have great significance for the development of LEWS. Therefore, this study focuses on model management and integration issues in LEWS, excluding research on data preprocessing, forecasting, and early warning methods. This study applies cloud-native technologies such as Docker, microservices, and Kubernetes to the research field of LEWS, proposes a dynamic management and integration framework for models in LEWS, and conducts research on its management and integration mechanisms. This framework covers the standardized development, deployment procedures, storage management of model information, and custom application orchestration mechanisms of models in LEWS, and supports the integration of model applications into LEWS under a unified standard protocol. Its feasibility and effectiveness are demonstrated by simulating an actual running application scenario in LEWS. Compared to traditional technical methods, this framework exhibits superior performance in terms of improving the service capability, availability, and scalability of the models in LEWS.

The paper is organized as follows: Section 2 introduces the related theories and key technologies of the research, and elaborates on the methodology of the dynamic management and integration framework for models in LEWS in detail. Section 3 provides experimental details. Section 4 presents experimental results and discusses them, as well as related studies and the direction of further exploration. Section 5 summarizes this work and outlines the potential research directions in the future.

2. Materials and Methods

2.1. Related Theories and Key Technologies

2.1.1. Models in LEWS

The following types of models are included in LEWS.

- Raw data pre-processing models: the raw monitoring data may have extreme outliers, data loss, and other anomalies because of the potential uncertainties in the monitoring and wireless transmission back to the monitoring data server [21]. Hence, pre-processing the raw data is essential to ensure the accuracy of subsequent calculations. Common data pre-processing methods include identifying anomalous data, supplementing missing values with interpolation, smoothing and fitting data, among other stages [22].
- Susceptibility assessment models: landslide susceptibility is based on the geological background environment, with reference to the impact brought by human engineering activities, and then predicts the probability of landslides occurring in a certain area [23,24]. The development of landslide susceptibility assessment model methods has evolved from qualitative analysis [25], to quantitative analysis using mathematical statistical models, machine learning methods [26–28], and deep learning models [29].
- Displacement prediction models: by employing certain mathematical methods, predicting the displacement of landslides in a future period using historical monitoring curves is a necessary prerequisite for achieving time forecasting and early warning level determination [30]. The main displacement prediction models include gray model [31], machine learning model [20], etc.
- Time forecasting models: commonly used time forecasting models include GM (1, 1) [15], Saito model [16], etc. The time forecasting model forecasts the specific time when the landslide will finally be unstabilized after it enters the accelerated deformation stage, according to the displacement prediction results. The result is an important basis for subsequent emergency response [13]. Due to the diversity and uncertainty of landslide triggers, the determination of landslide damage time is a very complicated problem.
- Warning level solver models: early warning level solver models are a class of models that classify the risk of the monitored landslide into different levels based on a series of parameters derived from displacement prediction models and time forecasting models. The common criteria for early warning level solver include improved tangent angle [32], velocity and deformation phase [33], rainfall [9,34], and combination parameters of several different indicators [10].
- Warning release models: the ultimate goal of early warning is to prevent casualties and minimize property damage. The early warning release model disseminates warning information to relevant audiences, which may include professional monitors or people within the affected area of the disaster [6], through easily received channels according to the warning level. Early warning release methods include short messaging service (SMS), portal notifications, emails, etc. [32].

2.1.2. Key Technologies

(1) WebAPIs

WebAPIs serve as interfaces that facilitate communication between distinct systems over HTTP/HTTPS protocols, without exposing the source code or implementation details of the service providers [35]. These abstract concepts can be implemented and utilized across a range of platforms and programming languages, including Java, NET, Python, and more. WebAPIs provide a flexible and versatile way to create client-oriented services for diverse applications. The design quality of WebAPIs influences their adoption and usability among developers.

(2) Docker

Docker is a container engine that enables developers to create and deploy applications as lightweight and portable images that contain all the necessary dependencies [36]. These

images can be run as containers on any machine that supports Docker. Containers are isolated from each other by a sandbox mechanism and have minimal overhead in terms of creation and destruction. Unlike virtual machines, which emulate hardware and require a full operating system, containers share the same kernel but only include specific applications and libraries [37].

Docker provides an official image repository called Docker Hub, where users can store and retrieve images [38]. Alternatively, users can also set up their private image repositories using tools such as Registry or Harbor. This can improve the network performance and security of image management.

(3) Microservices

Microservices are a distributed application solution that decomposes a single application into multiple small services that collaborate to deliver value to users. Each service operates in its process and communicates with others using lightweight mechanisms. Each service focuses on a specific business domain and can be deployed and scaled independently and dynamically [39]. The first generation of microservices frameworks, such as Spring Cloud and Dubbo, rely on embedded proxies for service registration and discovery. This introduces coupling between microservices and proxies and makes service governance invasive to business code. Moreover, these frameworks have limited support for microservices written in different programming languages [40].

(4) Kubernetes and Istio

Kubernetes (also known as K8s) is a system that automates and manages containerized applications [41]. It is widely used for deploying microservices [42], but it does not provide all the functionalities required by microservices. A service mesh can complement Kubernetes by adding a communication agent to each service instance and handling all the network interactions of the service [43], such as service discovery and routing. Istio is a service mesh implementation for Kubernetes that uses a sidecar container approach. A sidecar container is a container that runs alongside the model service container in the same pod (Figure 1), sharing its IP, lifecycle, resources, network, and storage. This allows Istio to implement microservice features in containers without affecting the service code or language.

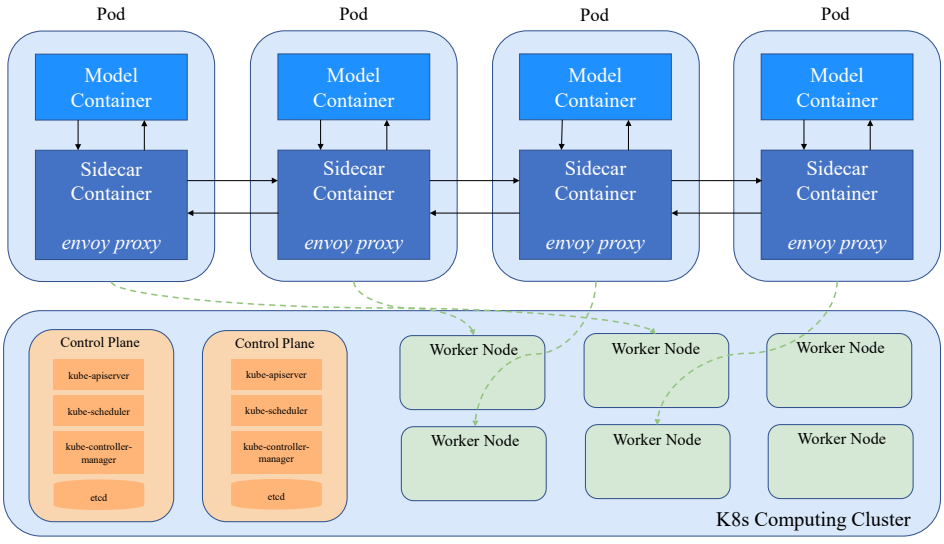


Figure 1. Istio service mesh architecture diagram.

2.2. Dynamic Management and Integration Framework for Models in LEWS

We propose a dynamic management and integration framework that deploys standardized and containerized LEWS models as WebAPIs on Kubernetes and Istio for microservices. LEWS can invoke models and user-defined early warning business processes through the

interface provided by the model repository management system. Figure 2 shows the technology roadmap.

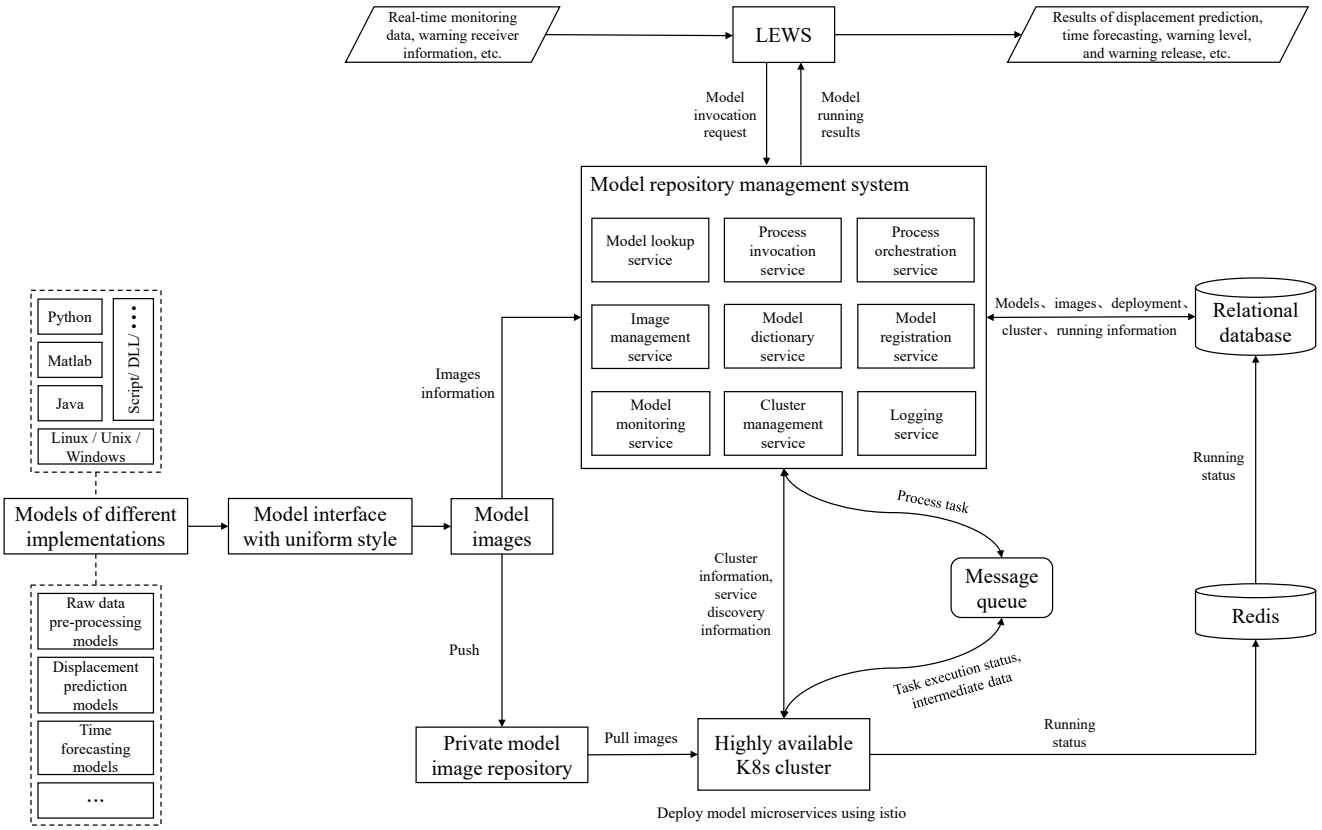


Figure 2. Technology roadmap.

2.2.1. Standardized Deployment of Models in LEWS

LEWS models (e.g., for monitoring data pre-processing, displacement prediction, time forecasting, warning level solving, and warning release) are often integrated as scripts, functions, libraries, or Web services with different languages and platforms. Many models have more diverse languages and runtimes than the technology stack of LEWS. For example, LEWS is built using Java Web technology, while some models (e.g., for data pre-processing) are implemented in Python and accessed through Python scripts. This leads to deep coupling and low flexibility. We propose an approach that uses Kubernetes and Istio as the runtime platform for various models, develops them as WebAPIs, and deploys them in containers.

Specifically, models are developed as WebAPIs using programming languages developers are familiar with (e.g., Python, Java), and using JavaScript object notation (JSON) data format for data interaction. WebAPI models have different technologies and deployment methods, which cause low efficiency and complex dependencies. We use Docker to package WebAPI models as images and deploy them in containers, build a private image repository for hosting model images, deploy a Kubernetes cluster with Istio, and use their microservice capabilities for model deployment, container orchestration, service discovery, and more.

2.2.2. Unified Management of Models in LEWS

Each model requires a unified management strategy to enable easy discovery, management, and integration of various models. For example, users can search by model name to get the interface and parameter information of a Verhulst prediction model. Each standardized encapsulated model corresponds to a Docker image, and each model has

different definitions of input and output parameters. After standardized deployment, a model may run with multiple container instances on different Kubernetes cluster nodes. We abstract the above model organization rules into an E-R (entity-relationship) data model (Figure 3).

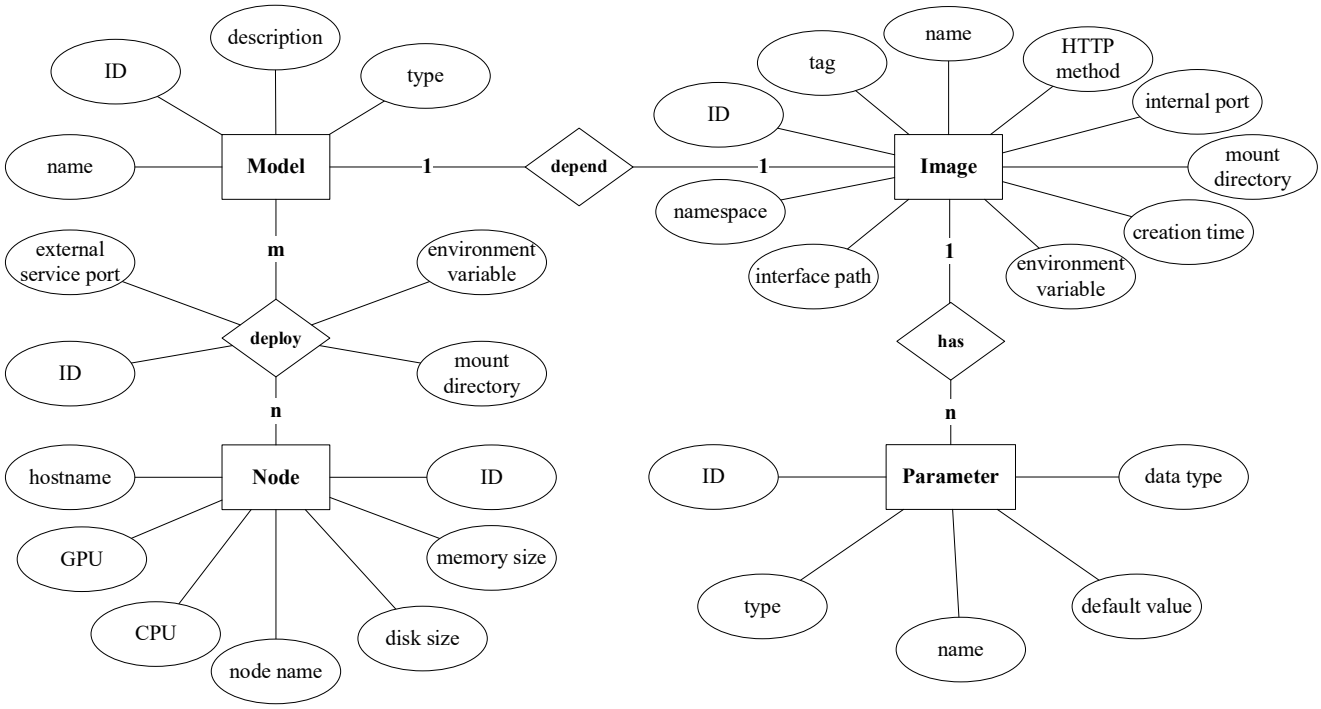


Figure 3. The conceptual model for managing LEWS models.

Each entity has the following meaning:

- Node: a node in the Kubernetes cluster with attributes such as hostname, IP address, and resource information.
- Image: a model service image with attributes such as name and tag.
- Parameter: a parameter includes the type (input/output), data type (text, number, boolean, list, or dictionary), and default values.
- Model: a deployed model instance with attributes such as name, description, model type, associated image, and node.

Each relationship has the following meaning:

- Depend: a model depends on an image.
- Deploy: a model can be deployed on multiple nodes with external service ports. Each node can host multiple models.
- Has: an image has input and output parameters of different types.

The design of the database is based on the above data model. The Kubernetes API from the control plane in Kubernetes and the Registry API from the mirror repository provide the ability to manipulate Kubernetes objects (Pod, Namespace, Node, etc.) and image repository objects, respectively. These APIs and the model repository database enable the unified management of model information.

2.2.3. Early Warning Business Application Process Orchestration

LEWS involves multiple subtasks, sequential constraints, and data transfer. For example, in the loess landslide warning system in Heifangtai terrace (Figure 4), data are processed, standard parameters are calculated for the multi-criteria warning model, and warning information is sent to different receivers based on thresholds and criteria [32]. Inspired based on the workflow of [44], we abstract the model-based application process

as a directed acyclic graph (DAG). The data structure of the model application process is defined as shown in Figure 5. In DAG, nodes contain model information such as model ID, and edges contain IDs of connected nodes and data binding between models. The user-defined process description is stored in JSON format.

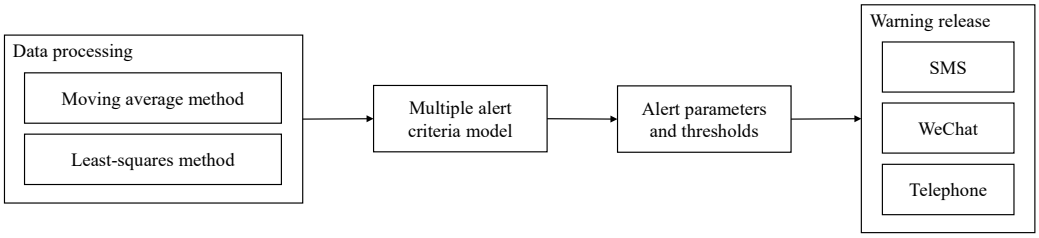


Figure 4. Flowchart of real-time early warning system for loess landslide on the Heifangtai terrace [32].

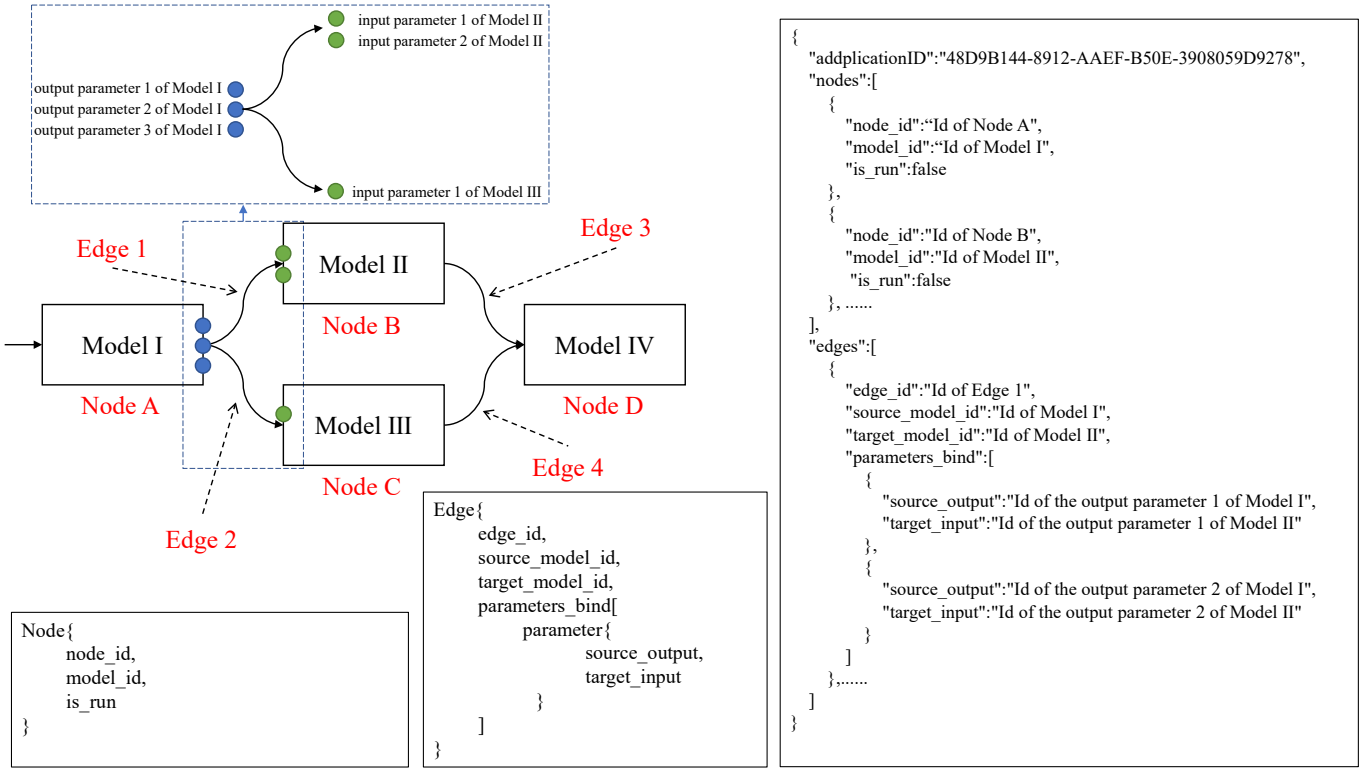


Figure 5. Data structure definition of early warning business process based on models connection.

2.2.4. Invocation Mechanism for Individual Models and Application Process

In the framework proposed in this paper, the instances of each deployed model (e.g., preprocessing, displacement prediction, time forecasting, etc.) are dynamically assigned resources and access addresses and need service discovery for service lookup. Under the microservices framework integrated with Kubernetes and Istio, Istio could follow the service discovery function of Kubernetes [45] and could view the real-time information of services and pods in Istio, including IP addresses, ports, and other access information and running status.

To call a specific model, it is essential to first obtain the model instance's URL. The URL consists of the deployed node's IP address, the external service port, and the dependency image's interface route. To acquire the IP address and port, use service discovery by providing the model ID and name. Meanwhile, the interface path and parameter information can be gathered through a related query.

We designed an interface that can be used to launch the early warning application processes. Firstly, by entering the user-defined model application process identifier, the corresponding application process topology relationship is queried from the database. Then, the task order relationship of each model is parsed, and the invocation of individual models is performed sequentially to realize the data interaction between the two models until the final result is returned.

2.2.5. Integration of Model or Application Process into LEWS

Individual models or user-defined application processes could be integrated into LEWS with a few steps. The model repository management system stores model information, so the model ID sufficed to get the model instance address and parameter definition. Access is based on the interface, identifier, and input data for application processes. Both individual models and application processes used the HTTP protocol for easy and quick integration into LEWS.

2.2.6. Development of Model Repository Management System

In this section, we developed a model repository management system based on the proposed methodology, with its structure illustrated in Figure 6. The system primarily consists of the following components: user interface, application services, basic components, and infrastructure platform.

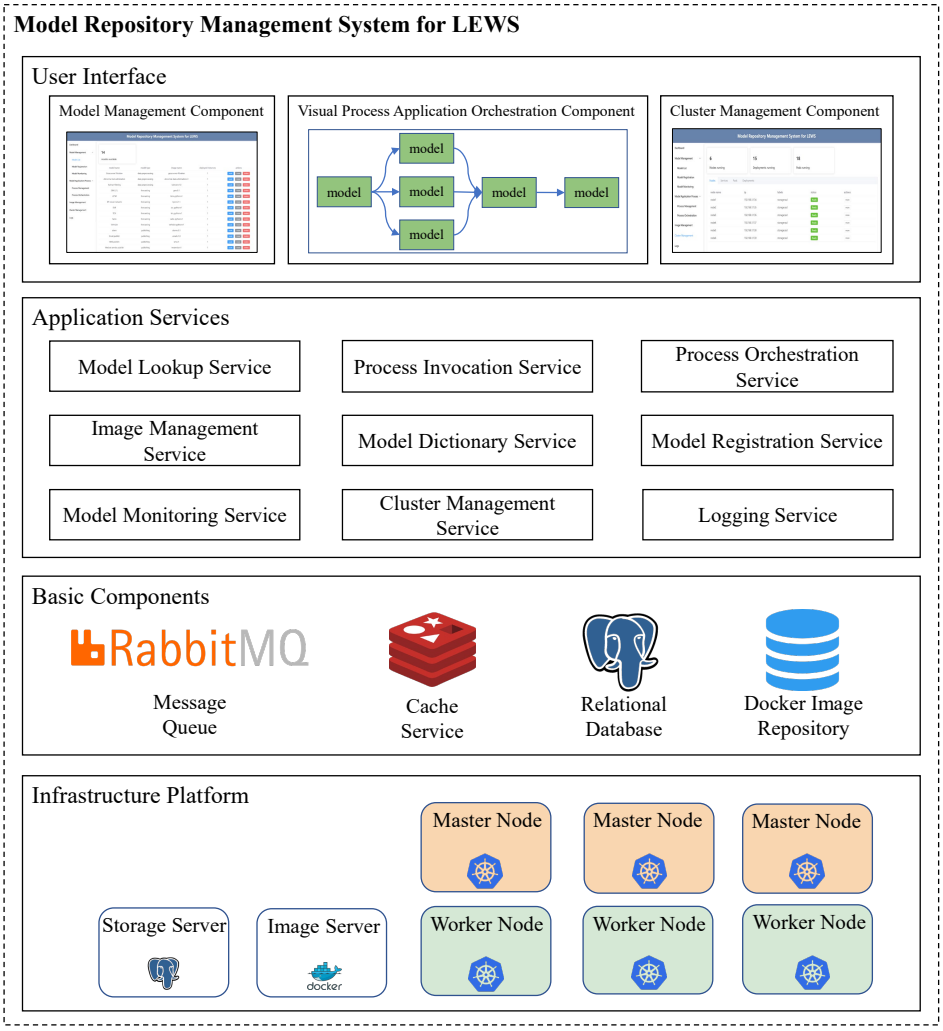


Figure 6. Diagram of the system structure.

The user interface supports model management, process orchestration, and cluster management. It could support modifying the number of instances of a model running through the UI, as well as process orchestration in the form of visualization, and combining models by dragging and dropping connections to achieve the creation of complex processes and the binding of parameters.

Application services comprise nine crucial services: model lookup service, process invocation service, process orchestration service, image management service, model dictionary service, model registration service, model monitoring service, cluster management service, and logging service. Among them, the logging service stores information such as the running status of each model, the result of each run, and the computation time spent, and provides a query interface.

Basic components include RabbitMQ message queue, Redis caching service, PostgreSQL database, private Docker image repository, etc.

The infrastructure platform includes a highly available Kubernetes cluster, a storage server, and an image server.

3. Experiments

3.1. Experiments Environment

In the experiments, eight CentOS 7.9 machines were utilized. Six of these machines (nodes 1–6) composed a highly available Kubernetes 1.25.5 cluster, featuring three manager nodes and three worker nodes. We deployed Istio 1.16.1 as the model runtime environment within our model repository management system. All machines were equipped with Docker 20.10.22 and OpenSSH 7.4.

3.2. Case Study

An operating early warning system in Huaihua City, Hunan Province will be presented as a case study. The system features functionalities such as a GIS map, monitoring data processing and analysis, warning mechanisms, and warning information release (as depicted in Figure 7). It integrates various models, including data pre-processing, prediction, and forecasting algorithms, threshold-based warning level solver models, and SMS/email release models. This system exemplifies a typical LEWS that necessitates the collaboration of multiple models. The early warning process based on surface displacement monitoring data consists of the following steps:

(1) Data pre-processing of surface displacement data (time-displacement curves) using the Kalman filtering model.
(2) Prediction of surface displacement data for the next 24 h, 48 h, and 72 h, as well as deformation rate, deformation acceleration, deformation rate increment, improved tangent angle, deformation phase, etc., using the long short-term memory (LSTM) model trained based on historical monitoring data.
(3) Utilizing an improved backpropagation (BP) network to forecast the expected destabilization damage times, based on the displacement prediction results.
(4) Warning level determination using a solver model based on time forecasting results. Generation of warning content and retrieval of warning recipient information based on the warning level correlation query.
(5) Sending early warning information to the recipient by SMS.

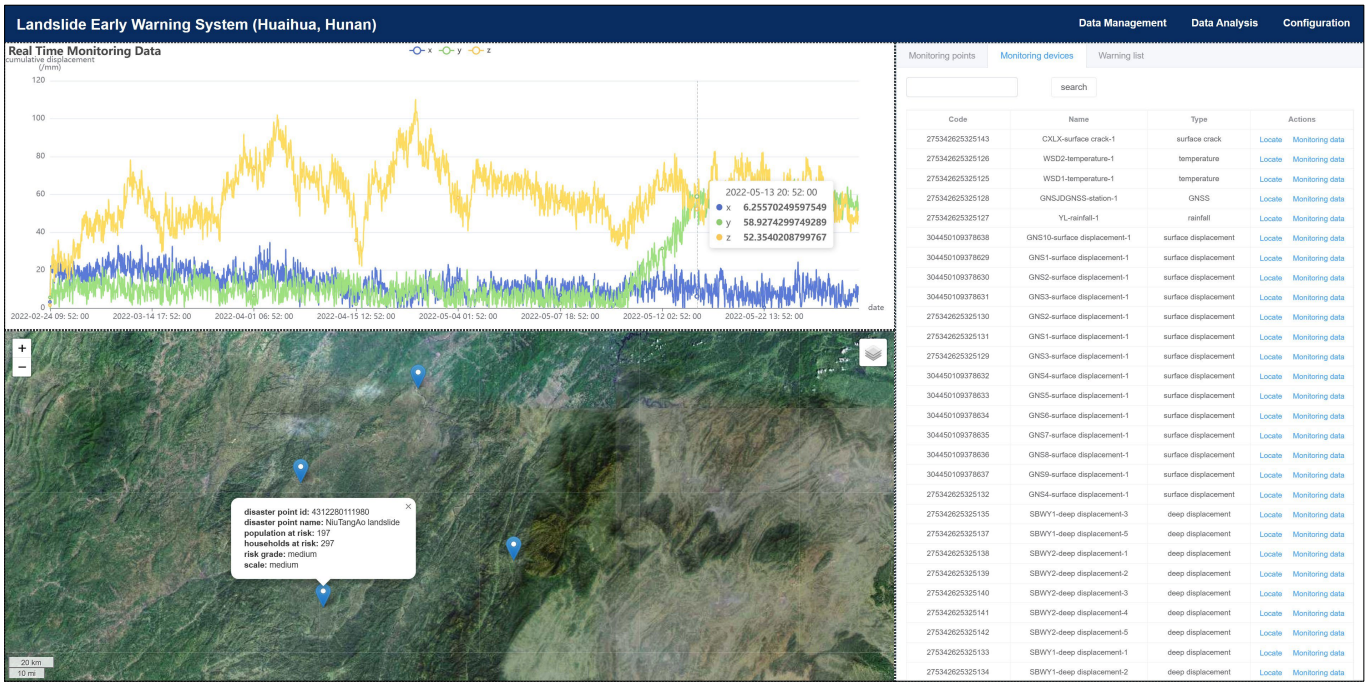


Figure 7. Home page of Huaihua LEWS.

4. Results and Discussion

4.1. Results

4.1.1. Deployment and Management of Models

In the Huaihua LEWS, the Kalman filter preprocessing model, LSTM displacement prediction model, and BP neural network time forecasting model were developed using Python. Meanwhile, the threshold-based warning level solver model and SMS publishing model were implemented in Java. The models developed in Python were encapsulated using Flask, a lightweight web application framework in Python. Similarly, the models implemented in Java were encapsulated using Spring Boot, a Java web framework, and packaged into a jar package via Maven, a software project management and comprehension tool. These standardized models were built into Docker images and uploaded to the image repository. Orchestration files were written to deploy the models in the Kubernetes cluster, and finally, the model information was added to the model repository management system, and the model and parameter metadata were enhanced.

4.1.2. Construction of User-Defined Early Warning Business Application Process

Following the original business logic of Huaihua LEWS, a test warning application process was established in the model repository management system. Next, the appropriate models were selected from the model list and dragged onto the canvas. Model connections and parameter correspondences were then set, completing the model application creation (as shown in Figure 8).

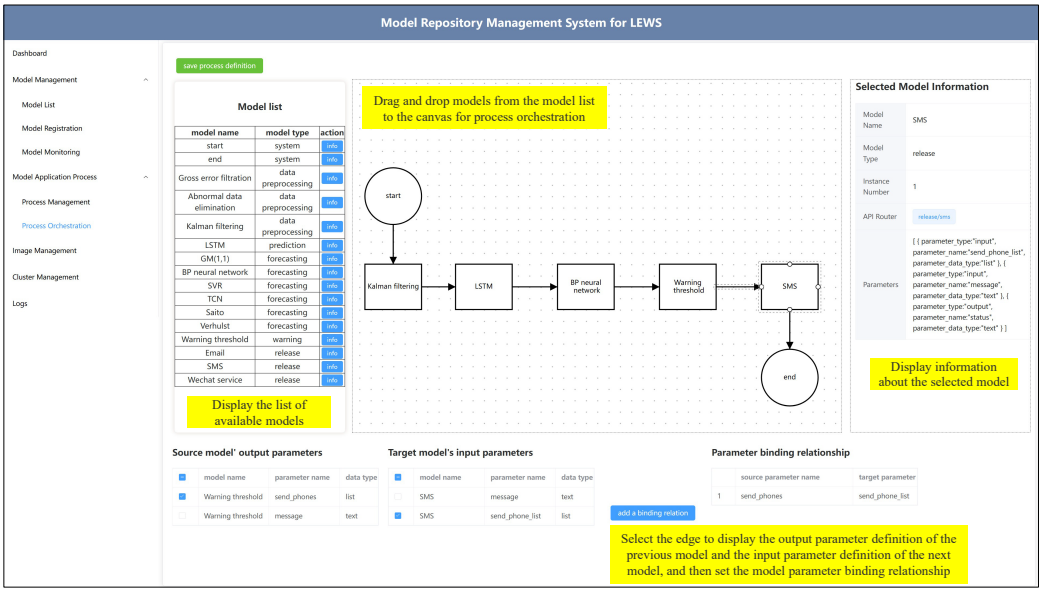


Figure 8. Process orchestration page.

4.1.3. Task Execution Test

Within the application process management user interface, the identifier of the previously established application process was retrieved, and the application process calling interface was seamlessly integrated into the Huaihua LEWS. Surface displacement monitoring data from two separate landslides, locations, and time frames were employed for early warning task execution experiments, respectively. The execution outcomes for each model involved in the application process indicated that each task in the customized application process operated accurately, including accomplishing data pre-processing (Figure 9), displaying the results of the warning calculation (Figure 10), and conducting the final warning information release (Figure 11).

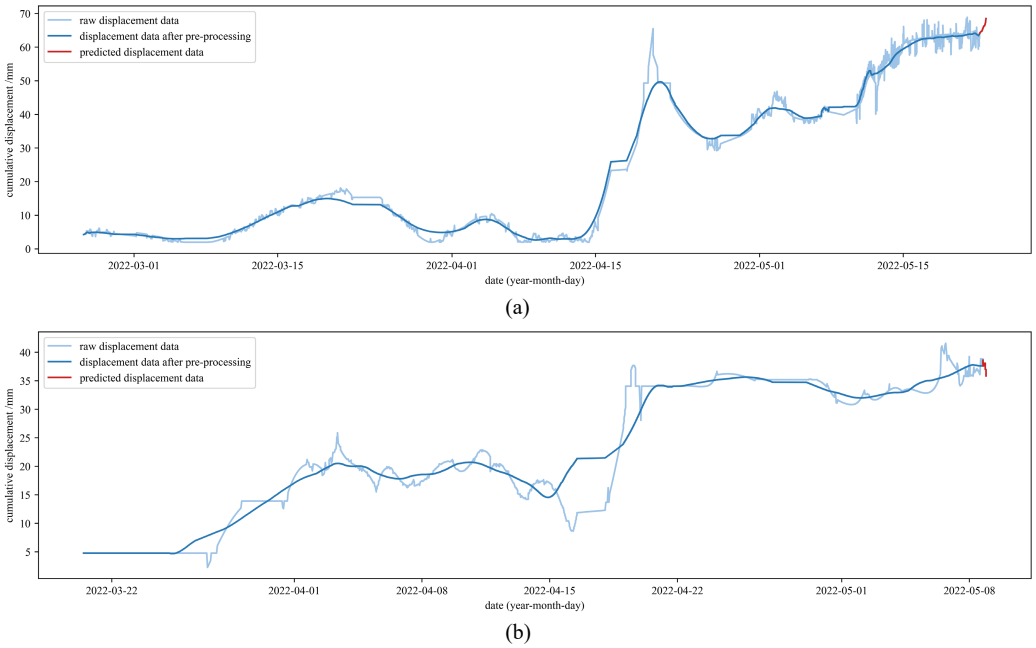
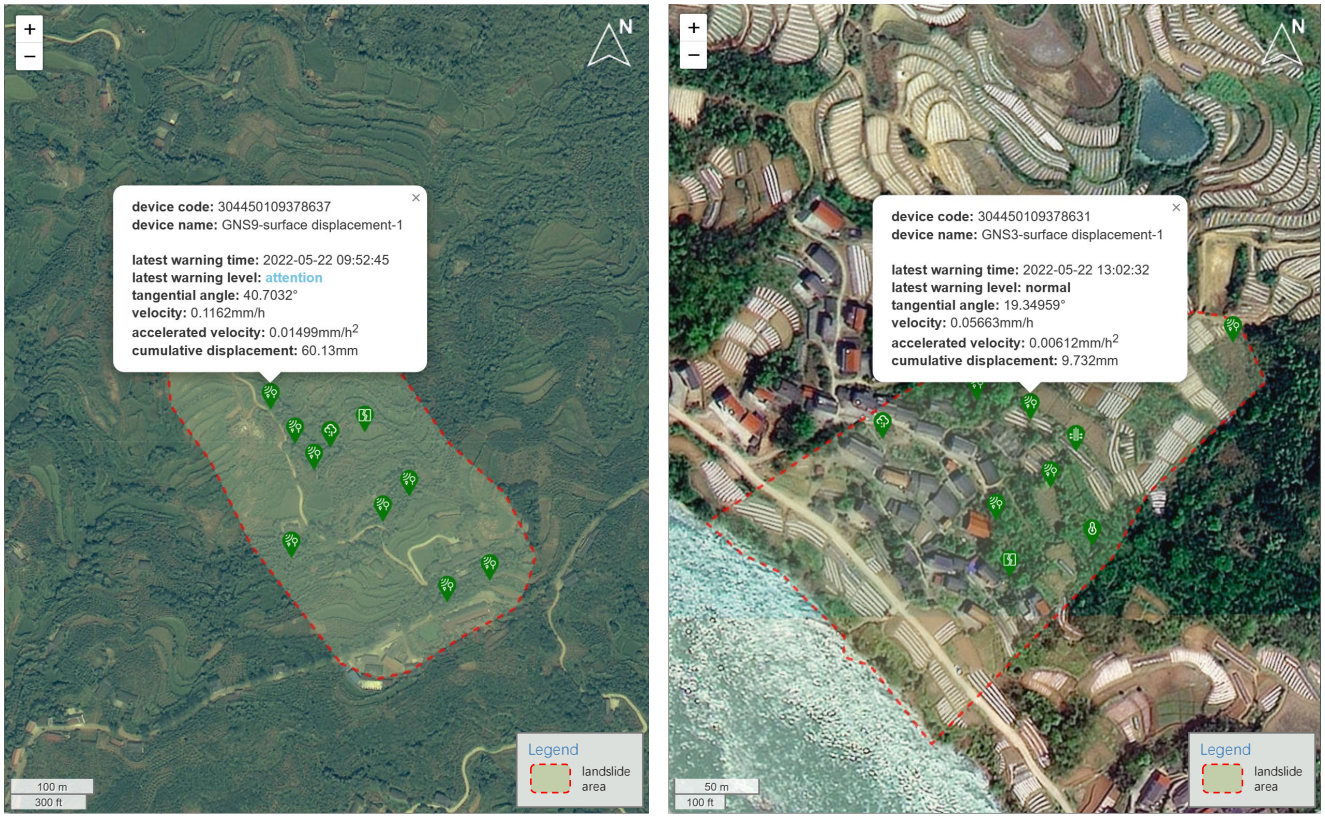


Figure 9. Pre-processing and prediction task results. (**a**) Pre-processing and prediction results of monitoring data of NiutangAo landslide. (**b**) Pre-processing and prediction results of monitoring data of Jiangdong Village landslide.



(a) (b)

Figure 10. Map visualization of warning results at different landslide monitoring points. (**a**) NiutangAo landslide. (**b**) Jiangdong Village landslide.

Figure 11. Release NiuTangAo landslide warning information through SMS.

4.1.4. Scalability Test

Taking the BP neural network model in the above execution task test experiment as an example, the number of its instances was set from 1 to 3 through the model management user interface. By observing the cluster management service user interface (Figure 12 bottom), we could find that the number of pods of the BP neural network model has changed from 1 to 3, and the two newly scaled instances were successfully scaled and deployed to node 4 and node 5.

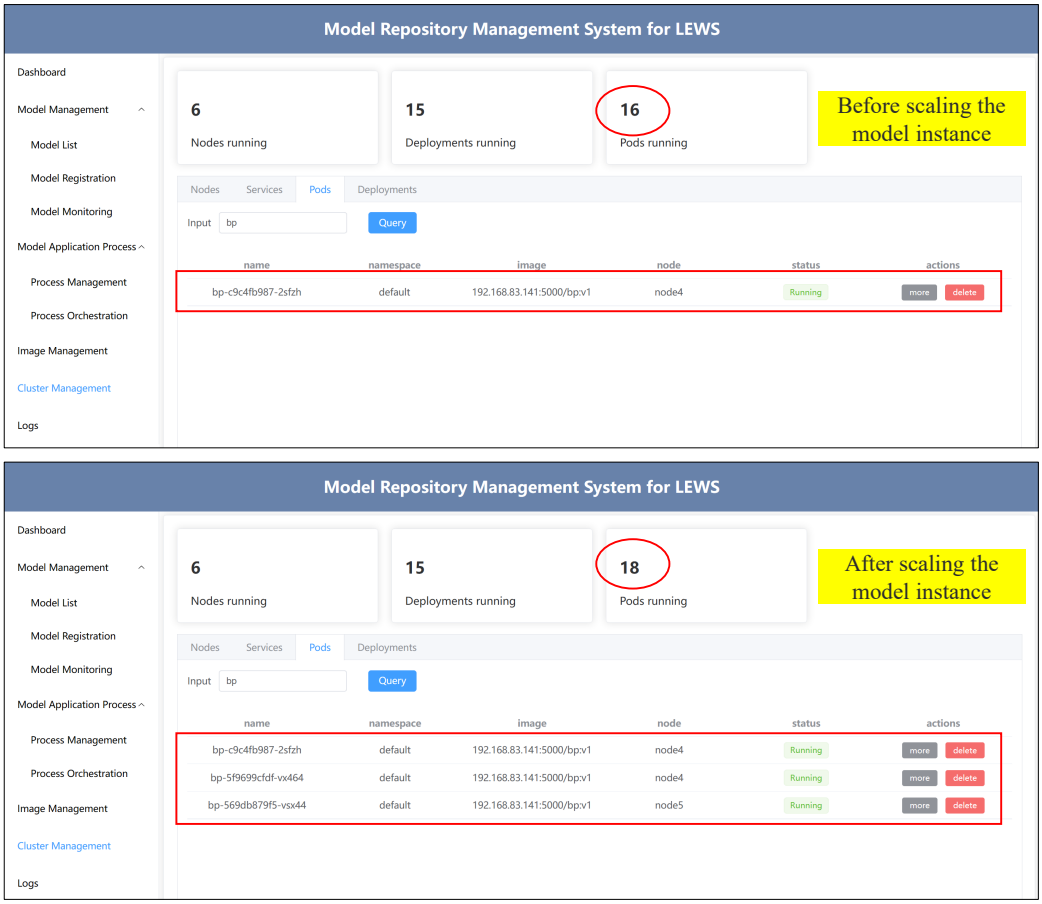


Figure 12. Scaling of a model instance.

4.1.5. High Availability Test

During the above execution test, nodes 1 and 4 were directly shut down to simulate a failure scenario in the cluster. The application process operation changes could be observed through the cluster management user interface and the model monitoring service user interface. As shown in Figure 13, the status of nodes 1 and 4 was changed from Ready to NotReady, but the early warning computation process was still able to run successfully. After restarting the nodes that were down, the service could be restored in seconds.

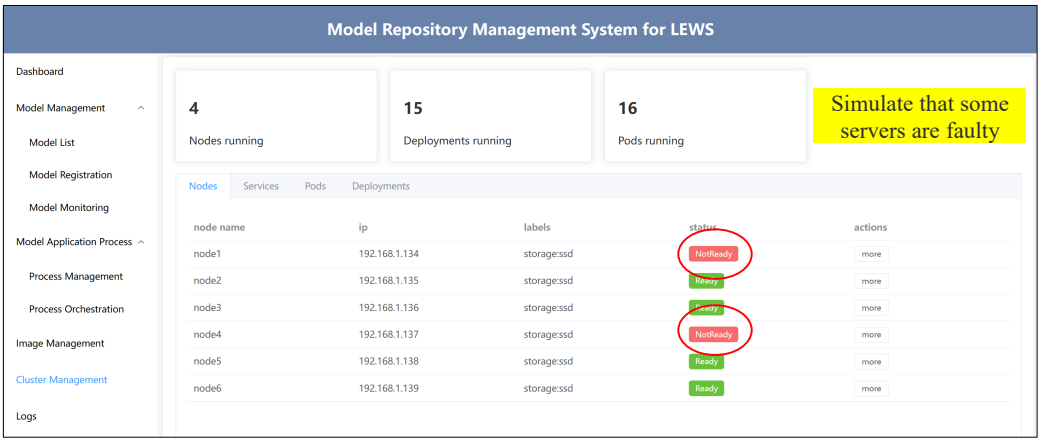


Figure 13. Illustration of nodes status changes on the nodes list page while executing the test.

4.2. Discussion

This study presents a framework that unifies model information management and integrates user-defined model application processes (Figure 8) into existing LEWS. The framework operates the warning calculation process in a manner consistent with the original LEWS, sequentially invoking models and accomplishing corresponding tasks (Figures 9–11). Figure 10 shows how the same customized process applies to two cases with different warning outcomes. Our framework determines whether to issue a warning based on different levels in the results. For example, the NiuTangAo landslide reaches the attention level and triggers SMS notifications to monitors, while the Jiangdong Village landslide remains at the normal monitoring level and requires no warning release. These results match the expectations of the case study (Figure 11).

This study aims to standardize the deployment, management, and integration of various models in LEWS rather than propose a new software architecture to refactor LEWS completely [39]. The proposed framework minimizes the impact on non-core system functions such as data management and visualization, and is providing a non-invasive approach to enhancing the capabilities of the system. We deploy various models in LEWS by microservice implementation and containerization. This approach standardizes the model development and deployment process, which facilitates subsequent integration and application of the models. Although it adds a small amount of workload compared to the original integration approach, the benefits outweigh the additional effort.

This work establishes a model repository management system that consolidates the storage and management of model information, image information, and cluster information while offering manipulation capabilities. The system supports the formation of application processes by combining and connecting models (Figure 8). The users can replace corresponding models on the graphical interface according to changing business requirements for landslide early warning systems, such as switching from a BP neural network model to another time forecasting model. The proposed framework provides individual models or user-defined model application processes to LEWS as interfaces, which abstract away the details of each model invocation and implementation. Changing the warning models in the process only requires modifying the process definition, not the system source code. The framework invokes models in a uniform approach in LEWS, which facilitates system integration and adapts to dynamic changes in early warning business and integrated models.

The experimental results show that the method ensures the normal serviceability of the model. The model is deployed in a distributed and scalable form on a highly available cluster (Figure 12), which improves the service availability and maintains the model's service capability even under partial node failure (Figure 13). This enhances the stability of the early warning service. Our framework also presents the potential for achieving high parallelizability of model tasks and supporting high-performance parallel computation when multiple warning processes are executed simultaneously. This capability distinguishes it from the original general mode, enabling more efficient and concurrent task processing.

The proposed framework divides the early warning process in LEWS into separate models, which are independently developed and deployed as manageable services, effectively decoupling them from the core early warning system. It differs from the monolithic or modular development approach in terms of dynamic management of models, dynamic orchestration of the warning process, and integration with the system. However, it also has a drawback. The smaller the granularity of the independently deployed and dynamically managed models in the early warning process, correspondingly, the higher the cost of model operation, maintenance, and inter-model communication, yet this is a common drawback of cloud-native applications. Nevertheless, if many models exist in LEWS, if different warning processes need to be defined based on different types of monitoring data, or if models or warning processes need to be updated frequently. Then it is signifi-

cantly beneficial to follow the model encapsulation, deployment, and integration strategy proposed in this paper.

5. Conclusions

In this paper, we present a framework for dynamic management and standardized integration of models in the landslide early warning system (LEWS) to address the challenges of updating, replacing, deploying, and integrating various models in LEWS. The framework uses containerization technology for model release and deployment, Kubernetes and Istio for model microservices governance, a model repository for model base information management, and a visual drag-and-drop editor for model application process orchestration. Models and user-defined processes are encapsulated as standardized WebAPIs for system integration. We apply the framework to two monitoring points (NiuTangAo and Jiangdong Village) of the Huaihua LEWS, build a warning calculation process based on surface displacement data, and perform system integration. The results demonstrate that this framework optimizes model deployment, unifies the management integration approach, improves scalability, availability, and scalability of models, and has the prospect of increasing computational parallelism, which is constructive to promote the upgrade and development of LEWS.

A natural progression of this work is to develop continuous integration and continuous deployment (CI/CD) workflows to automate the operation and maintenance during model development, deployment, and integration in LEWS. We will also explore Kubernetes scheduling of model computation tasks for real-time landslide warning to achieve efficient parallelism of massive warning tasks.

Author Contributions: Conceptualization, Jiqiu Deng; Funding acquisition, Jiqiu Deng; Investigation, Yu Tang; Methodology, Liang Liu; Software, Liang Liu; Validation, Yu Tang; Visualization, Liang Liu; Writing—original draft, Liang Liu; Writing—review & editing, Jiqiu Deng and Yu Tang. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by National Natural Science Foundation of China (Grant Number: 42172330).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The authors would like to thank three anonymous peer reviewers for helpful and constructive reviews of the manuscript's content, and Mohammad Naser Lessani from the University of South Carolina for helping to polish the manuscript.

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

References

1. Pecoraro, G.; Calvello, M.; Piciullo, L. Monitoring strategies for local landslide early warning systems. *Landslides* **2019**, *16*, 213–231. [CrossRef]
2. Wicki, A.; Lehmann, P.; Hauck, C.; Seneviratne, S.I.; Waldner, P.; Stähli, M. Assessing the potential of soil moisture measurements for regional landslide early warning. *Landslides* **2020**, *17*, 1881–1896. [CrossRef]
3. Michoud, C.; Bazin, S.; Blikra, L.H.; Derron, M.H.; Jaboyedoff, M. Experiences from site-specific landslide early warning systems. *Nat. Hazards Earth Syst. Sci.* **2013**, *13*, 2659–2673. [CrossRef]
4. Ju, N.; Huang, J.; Huang, R.; He, C.; Li, Y. A Real-time monitoring and early warning system for landslides in Southwest China. *J. Mt. Sci.* **2015**, *12*, 1219–1228. [CrossRef]
5. Liu, Y.; Yin, K.; Chen, L.; Wang, W.; Liu, Y. A community-based disaster risk reduction system in Wanzhou, China. *Int. J. Disaster Risk Reduct.* **2016**, *19*, 379–389. [CrossRef]
6. Piciullo, L.; Calvello, M.; Cepeda, J. Territorial early warning systems for rainfall-induced landslides. *Earth-Sci. Rev.* **2018**, *179*, 228–247. [CrossRef]
7. Tao, Z.; Zhang, H.; Zhu, C.; Hao, Z.; Zhang, X.; Hu, X. Design and operation of App-based intelligent landslide monitoring system: the case of Three Gorges Reservoir Region. *Geomat. Nat. Hazards Risk* **2019**, *10*, 1209–1226. [CrossRef]

8. Wu, Y.; Niu, R.; Wang, Y.; Chen, T. A fast deploying monitoring and real-time early warning system for the Baige Landslide in Tibet, China. *Sensors* **2020**, *20*, 6619. [CrossRef]
9. Abraham, M.T.; Satyam, N.; Shreyas, N.; Pradhan, B.; Segoni, S.; Abdul Maulud, K.N.; Alamri, A.M. Forecasting landslides using SIGMA model: A case study from Idukki, India. *Geomat. Nat. Hazards Risk* **2021**, *12*, 540–559. [CrossRef]
10. Huang, M.; Weng, H.; Hong, C.; Xu, X.; Tao, Z.; Li, C.; Huang, Y. Novel Intelligent Approach for the Early Warning of Rainfall-Type Landslides Based on the BRB Model. *Int. J. Geomech.* **2022**, *22*, 06022027. [CrossRef]
11. Song, Y.; Fan, W.; Yu, N.; Cao, Y.; Jiang, C.; Chai, X.; Nan, Y. Rainfall Induced Shallow Landslide Temporal Probability Modelling and Early Warning Research in Mountains Areas: A Case Study of Qin-Ba Mountains, Western China. *Remote Sens.* **2022**, *14*, 5952. [CrossRef]
12. Liu, Y.; Huang, J.; Xiao, R.; Ma, S.; Zhou, P. Research on a Regional Landslide Early-Warning Model Based on Machine Learning—A Case Study of Fujian Province, China. *Forests* **2022**, *13*, 2182. [CrossRef]
13. Huang, H.; Ni, J.; Zhang, Y.; Qian, T.; Shen, D.; Wang, J. Web3DGIS-based system for reservoir landslide monitoring and early warning. *Appl. Sci.* **2016**, *6*, 44. [CrossRef]
14. Guzzetti, F.; Gariano, S.L.; Peruccacci, S.; Brunetti, M.T.; Marchesini, I.; Rossi, M.; Melillo, M. Geographical landslide early warning systems. *Earth-Sci. Rev.* **2020**, *200*, 102973. [CrossRef]
15. Huang, C.; Cao, Y.; Zhou, L. Application of optimized GM (1, 1) model based on EMD in landslide deformation prediction. *Comput. Appl. Math.* **2021**, *40*, 1–21. [CrossRef]
16. Wu, S.; Hu, X.; Zheng, W.; Berti, M.; Qiao, Z.; Shen, W. Threshold definition for monitoring Gapa Landslide under large variations in reservoir level using GNSS. *Remote Sens.* **2021**, *13*, 4977. [CrossRef]
17. Niu, H. Smart safety early warning model of landslide geological hazard based on BP neural network. *Saf. Sci.* **2020**, *123*, 104572. [CrossRef]
18. Zhang, D.; Yang, J.; Li, F.; Han, S.; Qin, L.; Li, Q. Landslide Risk Prediction Model Using an Attention-Based Temporal Convolutional Network Connected to a Recurrent Neural Network. *IEEE Access* **2022**, *10*, 37635–37645. [CrossRef]
19. Intrieri, E.; Bardi, F.; Fanti, R.; Gigli, G.; Fidolini, F.; Casagli, N.; Costanzo, S.; Raffo, A.; Di Massa, G.; Capparelli, G.; et al. Big data managing in a landslide early warning system: Experience from a ground-based interferometric radar application. *Nat. Hazards Earth Syst. Sci.* **2017**, *17*, 1713–1723. [CrossRef]
20. Li, J.; Wang, W.; Han, Z. A variable weight combination model for prediction on landslide displacement using AR model, LSTM model, and SVM model: A case study of the Xinming landslide in China. *Environ. Earth Sci.* **2021**, *80*, 386. [CrossRef]
21. Thirugnanam, H.; Ramesh, M.V.; Rangan, V.P. Enhancing the reliability of landslide early warning systems by machine learning. *Landslides* **2020**, *17*, 2231–2246. [CrossRef]
22. Manconi, A.; Giordan, D. Landslide failure forecast in near-real-time. *Geomat. Nat. Hazards Risk* **2016**, *7*, 639–648. [CrossRef]
23. Spinetti, C.; Bisson, M.; Tolomei, C.; Colini, L.; Galvani, A.; Moro, M.; Saroli, M.; Sepe, V. Landslide susceptibility mapping by remote sensing and geomorphological data: Case studies on the Sorrentina Peninsula (Southern Italy). *GISci. Remote Sens.* **2019**, *56*, 940–965. [CrossRef]
24. Palenzuela Baena, J.A.; Scifoni, S.; Marsella, M.; De Astis, G.; Irigaray Fernández, C. Landslide susceptibility mapping on the islands of Vulcano and Lipari (Aeolian Archipelago, Italy), using a multi-classification approach on conditioning factors and a modified GIS matrix method for areas lacking in a landslide inventory. *Landslides* **2019**, *16*, 969–982. [CrossRef]
25. Thiery, Y.; Maquaire, O.; Fressard, M. Application of expert rules in indirect approaches for landslide susceptibility assessment. *Landslides* **2014**, *11*, 411–424. [CrossRef]
26. Sharma, L.; Patel, N.; Ghose, M.; Debnath, P. Development and application of Shannon's entropy integrated information value model for landslide susceptibility assessment and zonation in Sikkim Himalayas in India. *Nat. Hazards* **2015**, *75*, 1555–1576. [CrossRef]
27. Wang, Q.; Wang, Y.; Niu, R.; Peng, L. Integration of information theory, K-means cluster analysis and the logistic regression model for landslide susceptibility mapping in the Three Gorges Area, China. *Remote Sens.* **2017**, *9*, 938. [CrossRef]
28. Huan, Y.; Song, L.; Khan, U.; Zhang, B. Stacking ensemble of machine learning methods for landslide susceptibility mapping in Zhangjiajie City, Hunan Province, China. *Environ. Earth Sci.* **2023**, *82*, 1–18. [CrossRef]
29. Lv, L.; Chen, T.; Dou, J.; Plaza, A. A hybrid ensemble-based deep-learning framework for landslide susceptibility mapping. *Int. J. Appl. Earth Obs. Geoinf.* **2022**, *108*, 102713. [CrossRef]
30. Huang, F.; Yin, K.; Zhang, G.; Gui, L.; Yang, B.; Liu, L. Landslide displacement prediction using discrete wavelet transform and extreme learning machine based on chaos theory. *Environ. Earth Sci.* **2016**, *75*, 1–18. [CrossRef]
31. Wang, W.; Li, J.; Qu, X.; Han, Z.; Liu, P. Prediction on landslide displacement using a new combination model: A case study of Qinglong landslide in China. *Nat. Hazards* **2019**, *96*, 1121–1139. [CrossRef]
32. Xu, Q.; Peng, D.; Zhang, S.; Zhu, X.; He, C.; Qi, X.; Zhao, K.; Xiu, D.; Ju, N. Successful implementations of a real-time and intelligent early warning system for loess landslides on the Heifangtai terrace, China. *Eng. Geol.* **2020**, *278*, 105817. [CrossRef]
33. Chen, H.; Li, G.; Fang, R.; Zheng, M. Early warning indicators of landslides based on deep displacements: Applications on Jinping landslide and Wendong landslide, China. *Front. Earth Sci.* **2021**, *9*, 747379. [CrossRef]
34. Piciullo, L.; Gariano, S.L.; Melillo, M.; Brunetti, M.T.; Peruccacci, S.; Guzzetti, F.; Calvello, M. Definition and performance of a threshold-based regional early warning model for rainfall-induced landslides. *Landslides* **2017**, *14*, 995–1008. [CrossRef]

35. Segura, S.; Parejo, J.A.; Troya, J.; Ruiz-Cortés, A. Metamorphic testing of RESTful web APIs. In Proceedings of the 40th International Conference on Software Engineering, Gothenburg, Sweden 27 May–3 June 2018; p. 882. [CrossRef]
36. Zaragozí, B.M.; Trilles, S.; Navarro-Carrión, J.T. Leveraging container technologies in a giscience project: A perspective from open reproducible research. *ISPRS Int. J. Geo-Inf.* **2020**, *9*, 138. [CrossRef]
37. Wan, X.; Guan, X.; Wang, T.; Bai, G.; Choi, B.Y. Application deployment using Microservice and Docker containers: Framework and optimization. *J. Netw. Comput. Appl.* **2018**, *119*, 97–109. [CrossRef]
38. Zhao, N.; Tarasov, V.; Albahar, H.; Anwar, A.; Rupprecht, L.; Skourtis, D.; Paul, A.K.; Chen, K.; Butt, A.R. Large-scale analysis of docker images and performance implications for container storage systems. *IEEE Trans. Parallel Distrib. Syst.* **2020**, *32*, 918–930. [CrossRef]
39. Bai, D.; Tang, J.; Lu, G.; Zhu, Z.; Liu, T.; Fang, J. The design and application of landslide monitoring and early warning system based on microservice architecture. *Geomat. Nat. Hazards Risk* **2020**, *11*, 928–948. [CrossRef]
40. Jamshidi, P.; Pahl, C.; Mendonça, N.C.; Lewis, J.; Tilkov, S. Microservices: The journey so far and challenges ahead. *IEEE Softw.* **2018**, *35*, 24–35. [CrossRef]
41. Rossi, F.; Cardellini, V.; Presti, F.L.; Nardelli, M. Geo-distributed efficient deployment of containers with Kubernetes. *Comput. Commun.* **2020**, *159*, 161–174. [CrossRef]
42. Zhao, H.; Deng, S.; Liu, Z.; Yin, J.; Dustdar, S. Distributed redundant placement for microservice-based applications at the edge. *IEEE Trans. Serv. Comput.* **2020**, *15*, 1732–1745. [CrossRef]
43. Minna, F.; Massacci, F. SoK: Run-time security for cloud microservices. Are we there yet? *Comput. Secur.* **2023**, *127*, 103119. [CrossRef]
44. Pakdil, M.E.; Çelik, R.N. Serverless geospatial data processing workflow system design. *ISPRS Int. J. Geo-Inf.* **2021**, *11*, 20. [CrossRef]
45. Larsson, L.; Tärneberg, W.; Klein, C.; Elmroth, E.; Kihl, M. Impact of etcd deployment on kubernetes, istio, and application performance. *Softw.-Pract. Exp.* **2020**, *50*, 1986–2007. [CrossRef]