EAI Endorsed Transactions
on Scalable Information Systems
Research Article EAI.EU

Optimization of Load Balancing and Real-Time Scheduling in Distributed Systems for Cross-Language Text Processing Tasks

Hongying Pu[1], Jixiang Wang[1*]

[1]TianShui Normal University, TianShui 741000, China

Abstract

INTRODUCTION: With the rapid expansion of cross-language text processing applications, distributed systems must support large-scale multilingual workloads with high efficiency and responsiveness. Effective load balancing and real-time scheduling are therefore essential for maintaining system performance. However, many existing approaches rely on static or general-purpose scheduling strategies, which fail to capture language-dependent workload variations, resulting in limited scalability and inefficient resource utilization.
OBJECTIVES: This study aims to improve load balancing efficiency and real-time scheduling performance in distributed systems for cross-language text processing. The focus is on addressing dynamic task variations, heterogeneous resource demands, and robustness challenges in multilingual, noisy, and fluctuating computing environments.
METHODS: A cross-language-aware dynamic load balancing and scheduling optimization framework based on reinforcement learning is proposed. The framework incorporates language pair, task type, priority, input length, estimated resource demand, and real-time node status into an adaptive scheduling strategy. Denoising mechanisms and feature fusion modules are further introduced to enhance scheduling stability and decision robustness.
RESULTS: Experimental results show that the proposed method outperforms existing approaches in response time, resource utilization, and task completion rate. The framework achieves a task completion rate of 96.5%. Under high-noise conditions, the reduction in task completion rate is approximately 50% lower than that of baseline methods, demonstrating improved robustness.
CONCLUSION: The proposed framework provides an effective solution for dynamic scheduling and load balancing in cross-language text processing systems. By coupling multilingual task characteristics with distributed resource states, it offers scalable support for multilingual distributed computing and new insights into reinforcement learning-based scheduling optimization.

Keywords: Cross-language Text Processing, Distributed Systems, Load Balancing, Real-time Scheduling, Reinforcement Learning.

Received on 02 February 2026, accepted on 17 June 2026, published on 07 August 2026

doi: 10.4108/eetsis.11807

1. Introduction

With the rapid development of information technology and computing capabilities, distributed systems have become fundamental infrastructure for large-scale data processing and global computing services. Cross-language text processing has also become increasingly important, as multilingual applications are widely used in machine translation, multilingual sentiment analysis, cross-lingual information retrieval, automatic summarization, and international content recommendation [1][2]. In this study, cross-language tasks refer to tasks that process, compare, transfer, or generate textual information across two or more languages, such as translation, cross-lingual retrieval,

*Corresponding author. Email: wjx8103@163.com

multilingual sentiment classification, and multilingual summarization.

Compared with monolingual workloads, cross-language tasks show more heterogeneous computational characteristics. Differences in tokenization patterns, vocabulary sizes, word orders, and semantic structures lead to variations in input length, preprocessing cost, and model inference demand. Machine translation often requires encoder – decoder inference and language-pair alignment, while cross-lingual retrieval and multilingual sentiment analysis rely on multilingual representation matching, resulting in different memory, CPU, and GPU requirements. Low-resource languages or noisy multilingual data may further increase execution uncertainty. These characteristics make execution time and resource consumption more variable, increasing the difficulty of load balancing and real-time scheduling [3].

When cross-language tasks are deployed in distributed environments, efficiently processing large-scale multilingual data remains challenging. The scheduler must consider heterogeneous resources while adapting to dynamic task priorities, language-dependent resource demands, and fluctuating workloads. Existing cross-language methods are often optimized for task-level performance, while distributed scheduling mechanisms are treated separately. This separation may lead to uneven resource allocation, delayed response, and low scheduling efficiency in practical multilingual applications [4]. Therefore, system-level load balancing and real-time scheduling optimization are essential for improving response time and resource utilization.

Although progress has been made in distributed scheduling, existing studies still have limitations. Traditional load balancing methods mostly rely on static strategies and neglect dynamic changes and heterogeneous resource requirements of cross-language tasks [5][6]. Recent reinforcement learning-based scheduling methods adapt better to workload changes, but most focus on general computational workloads and rarely incorporate language-related task features into scheduling decisions [7]. Moreover, many methods insufficiently consider real-time changes in task priority, estimated resource demand, node load, and workload distribution, which may cause response delays or resource wastage [8]. Thus, a flexible framework that jointly considers cross-language task characteristics and distributed resource dynamics is needed.

To address these issues, this paper proposes a cross-language characteristic-driven adaptive scheduling framework based on reinforcement learning. Unlike conventional methods that mainly rely on queue length, task size, or node utilization, the proposed framework couples multilingual task attributes with real-time distributed resource states. Specifically, language pair, task type, input length, priority, estimated and dynamically updated resource demand, and node status are incorporated into the scheduling state representation. This design enables the scheduler to handle heterogeneous multilingual workloads, improve execution efficiency, and maintain balanced resource utilization.

The innovation of this study lies in three aspects. First, it formulates cross-language text processing as a heterogeneous distributed scheduling problem rather than a general task allocation problem. Second, it introduces denoising and feature fusion mechanisms to improve scheduling stability under noisy multilingual inputs and uncertain resource estimation. Third, it develops a reinforcement learning-based adaptive scheduler that dynamically adjusts task allocation according to workload fluctuation and real-time node feedback.

This research has both theoretical and practical significance. Theoretically, it extends cross-language NLP research from task-level accuracy optimization to system-level scheduling efficiency under heterogeneous multilingual workloads. Practically, the proposed framework can support multilingual cross-border e-commerce, global information retrieval, and international content processing by improving response efficiency and resource utilization. By coupling multilingual task characteristics with real-time distributed resource states, this study offers guidance for deploying scalable cross-language NLP services.

2. Related Works

2.1. Application Scenarios and Challenges

Cross-language text processing is an important research direction in Natural Language Processing (NLP) and is widely applied to machine translation, cross-language information retrieval, automatic summarization, and sentiment analysis[9]. These tasks play an important role in multilingual information access and the global digital economy. However, when deployed in distributed computing environments, cross-language processing faces challenges arising from structural language differences, semantic inconsistencies, and the efficient coordination of heterogeneous computational resources. Efficient multilingual information transfer and effective utilization of distributed resources remain core issues[10][11][12].

Several datasets, such as Europarl, IWSLT, and TED Talks, have become important resources for cross-language research and are widely used in translation and information retrieval tasks. Europarl provides a large bilingual corpus, while IWSLT contains TED talk subtitles for multilingual translation and speech processing[13][14]. Nevertheless, limitations in language coverage and corpus scale introduce data bias, reducing their generalizability in large-scale distributed scenarios.

Common evaluation metrics include BLEU, ROUGE, METEOR, and TER, which are mainly designed for translation and text generation tasks. Although standardized, these metrics emphasize lexical overlap and overlook syntactic and semantic consistency. More importantly, they do not capture system-level performance indicators such as response latency, scheduling efficiency, or load balancing in distributed environments, limiting comprehensive performance assessment.

2.2. Overview of Mainstream Methods

Deep learning has significantly advanced cross-language text processing, particularly through Neural Machine Translation (NMT). NMT models learn language mappings in an end-to-end manner and achieve strong translation performance[15][16]. However, their reliance on large parallel corpora and high computational costs poses challenges for resource scheduling and load balancing in distributed systems, especially for low-resource languages.

Multi-task Learning (MTL) and Transfer Learning have also gained attention. MTL improves learning efficiency by sharing representations across tasks, while Transfer Learning transfers knowledge from high-resource to low-resource languages[17][18]. Despite their effectiveness, these approaches depend heavily on large pre-trained corpora and often struggle in heterogeneous distributed computing environments.

Pre-trained language models, such as BERT and Transformer-based architectures, further extend cross-language processing to sentiment analysis and text classification[19]. However, their high computational and data requirements intensify the burden on distributed scheduling and real-time resource management, particularly in large-scale settings.

2.3. Closest Related Work

In distributed load balancing and real-time scheduling, static or heuristic strategies based on task priority, queue length, or estimated execution time have been widely studied[20]. These methods work under stable workloads, but often assume fixed task profiles and stable resource availability. Such assumptions are difficult to satisfy in cross-language text processing, where task complexity, input length, language-pair characteristics, and resource demand vary over time.

Recent studies have introduced reinforcement learning-based scheduling in distributed, cloud, edge-fog-cloud, and heterogeneous computing environments[21]. These methods treat scheduling as a sequential decision-making problem, where allocation policies are learned from node load, queue length, task priority, and resource availability. Compared with static methods, reinforcement learning-based schedulers adapt better to workload changes, while deep reinforcement learning improves policy learning in large-scale environments[22][23].

However, existing reinforcement learning-based methods remain limited for cross-language text processing. Most focus on general workloads and do not explicitly model language-related features, such as language pair, tokenization complexity, semantic alignment difficulty, or multilingual inference cost. Noisy multilingual data and low-resource languages are also rarely considered.

In contrast, this study incorporates cross-language task features and real-time node resource states into scheduling decisions. Different from general RL-based scheduling, it couples language-dependent workload characteristics with distributed resource states and introduces denoising and feature fusion modules to improve scheduling robustness.

2.4. Summary and Research Gaps

Although progress has been made in cross-language text processing and distributed scheduling, most studies address them separately. Cross-language NLP studies mainly emphasize task-level performance, while reinforcement learning-based scheduling studies usually focus on general computational workloads without sufficient consideration of multilingual workload heterogeneity.

Three research gaps remain. First, existing scheduling methods rarely incorporate cross-language task attributes, such as language pair, task type, input length, tokenization complexity, and multilingual inference cost. Second, most reinforcement learning-based schedulers lack noise-aware mechanisms for noisy multilingual data or low-resource language tasks. Third, task-level cross-language processing and system-level resource scheduling are often loosely coupled, making it difficult to optimize multilingual task performance and distributed resource utilization simultaneously.

This paper addresses these gaps by proposing a reinforcement learning-based dynamic scheduling framework that jointly considers task priorities, cross-language task features, and real-time resource conditions. The novelty lies in incorporating language-related workload features into the state representation and introducing denoising and feature fusion mechanisms to improve scheduling stability under noisy multilingual conditions. This design distinguishes the proposed framework from general-purpose distributed scheduling methods and improves its relevance to cross-language text processing scenarios.

3. Methodology

3.1. Problem Formulation

In cross-language text processing tasks, the load balancing and real-time scheduling optimization problem in distributed systems aims to improve task execution efficiency and system responsiveness through efficient resource allocation. Unlike monolingual workloads, cross-language tasks involve heterogeneous attributes, such as language pair, task type, input length, tokenization complexity, priority level, and estimated rather than fixed computational demand. These attributes affect execution time, memory consumption, and scheduling priority, and should therefore be considered in the scheduling formulation.

Consider a distributed system with N computational nodes and M cross-language text processing tasks. The computational capacity of node iis denoted as C_i, and the task set assigned to node iis denoted as $R_j(t)$. For each task t_j, its estimated and dynamically updated resource demand is denoted as R_j, and its priority is represented by p_j, where

values closer to 1 indicate higher priority. Instead of assuming that resource demand is fully known in advance, $R_j(t)$ is updated according to input length, language-pair complexity, estimated inference cost, and real-time execution feedback. Each task t_j is also associated with a cross-language feature vector x_j, including language pair, task type, input length, tokenization complexity, and estimated model inference cost. This vector characterizes multilingual workload heterogeneity and supports adaptive scheduling decisions.

The objective is to dynamically schedule tasks and allocate resources to maximize resource utilization and minimize response time. Resource utilization is expressed as:

$$\max U = \frac{\sum_{i=1}^{N} \sum_{t_j \in S_i} R_j}{\sum_{i=1}^{N} C_i} \quad (1)$$

where S_i is the set of tasks assigned to node i. To ensure timely processing of high-priority tasks, weighted response time is minimized as follows:

$$\min WRT = \sum_{t_j \in T} p_j \cdot RT_j \quad (2)$$

where T denotes the complete task set, and RT_j represents the response time of task t_j. This formulation gives higher scheduling importance to urgent tasks while maintaining overall response efficiency.

During optimization, several constraints must be satisfied. The total resource consumption assigned to each node should not exceed its available capacity; each task should be assigned to an appropriate node or decomposed into subtasks for multi-node parallel execution when necessary; and the response time of each task should remain below its predefined threshold. For cross-language tasks, this threshold may vary by task type and priority. For example, real-time multilingual sentiment analysis usually requires a shorter response threshold than offline cross-lingual document retrieval. In addition, practical factors such as node load fluctuation, communication overhead, and temporary resource unavailability are considered through real-time node-state feedback.

By combining these objectives and constraints, this paper formulates cross-language task scheduling as a dynamic resource allocation problem under heterogeneous multilingual workloads. The scheduler jointly considers task priority, language-related computational characteristics, dynamically estimated resource demand, and real-time node resource status to improve execution efficiency and resource utilization in distributed cross-language text processing systems.

3.2. Overall Framework

The proposed distributed load balancing and real-time scheduling optimization framework consists of three core modules: the task allocation module, the scheduling optimization module, and the resource monitoring module. These modules work together to ensure efficient task scheduling, maximization of resource utilization, and minimization of system response time. Figure 1 illustrates the data flow and interactions between these modules. The overall workflow includes task profiling, priority-based initial allocation, reinforcement learning-based rescheduling, real-time resource monitoring, and multi-node execution support for computation-intensive tasks.

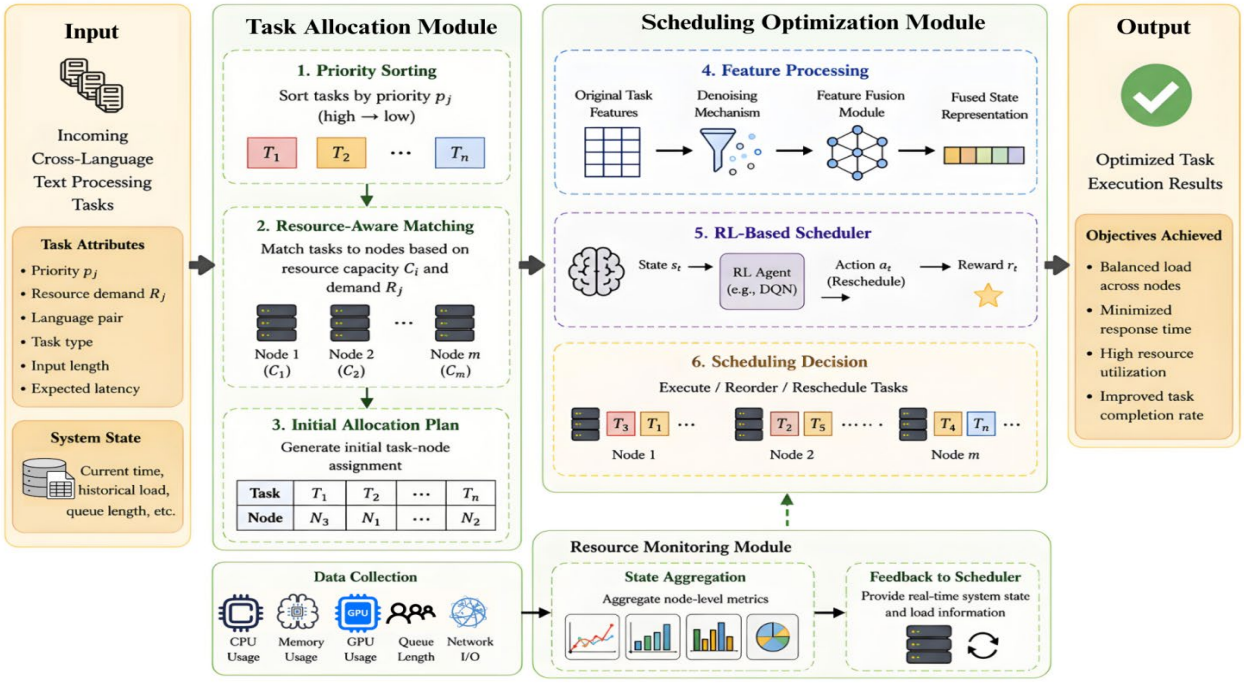

Figure 1. System architecture for real-time task scheduling and load balancing

The task allocation module preliminarily assigns tasks to computational nodes based on task priority p_j, dynamically estimated resource demand R_j, and node capacity C_i, forming an initial task allocation plan and passing it to the scheduling optimization module. For computation-intensive cross-language tasks, the module also supports subtask-level allocation when multi-node execution is required.

The scheduling optimization module dynamically adjusts the task allocation strategy using a reinforcement learning-based scheduler, ensuring that high-priority tasks are processed first while optimizing resource utilization. This module adjusts scheduling based on real-time feedback to meet response time requirements and load balancing goals. Its input includes fused task-node features, and its output is the scheduling action, such as assigning, delaying, or rescheduling a task.

The resource monitoring module continuously monitors the resource usage of each computational node, providing feedback to the scheduling optimization module to ensure that reasonable decisions are made based on the current resource status, avoiding overload or resource wastage. The monitored indicators include CPU utilization, memory utilization, GPU utilization, queue length, node load, and task execution status.

The collaborative mechanism of these three modules ensures that the system can flexibly respond to dynamic task changes, achieving real-time scheduling optimization for cross-language text processing tasks.

Task Allocation Module

The primary goal of the task allocation module is to allocate tasks reasonably based on their priority, resource requirements, and the computational capabilities of the nodes, ensuring system load balancing and improving task execution efficiency.

This module allocates tasks based on task priority p_j, dynamically estimated resource demand $R_j(t)$, and node capacity C_i. A priority sorting and resource matching approach is employed, assigning higher priority tasks to nodes with greater available resources. During allocation, task-node compatibility is considered to ensure that computational resources are not overloaded. The compatibility score is determined by comparing the estimated task demand with the current available CPU, memory, GPU, and queue capacity of each node.

The task allocation module first sorts tasks based on priority, then selects the most appropriate node for allocation based on the node's resource status and the task's requirements. If a task exceeds the capacity of a single node or is suitable for parallel execution, it can be divided into subtasks or micro-batches and assigned to multiple nodes. This design relaxes the one-task-one-node constraint and supports multi-node parallel execution for computation-intensive cross-language tasks. The task allocation plan is subsequently passed to the scheduling optimization module for further optimization. The task allocation module is illustrated in Figure 2.

3.3. Module Descriptions

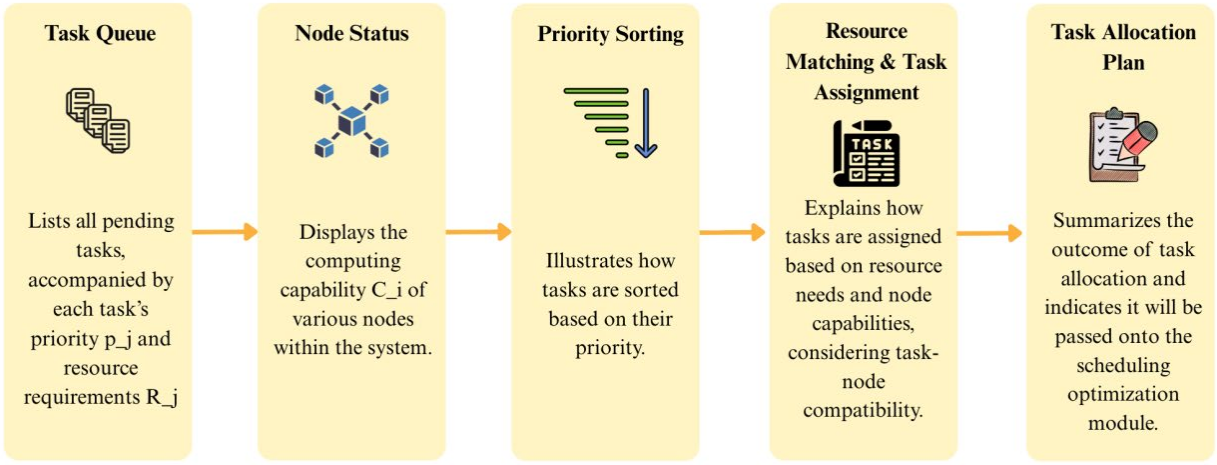

Figure 2. Task Allocation Module: Priority-Based Sorting and Resource-Aware Task Assignment Workflow

Scheduling Optimization Module

The scheduling optimization module dynamically adjusts task allocation to improve performance in dynamic environments. It updates scheduling decisions based on task progress, resource utilization, queue length, and cross-language task characteristics. Before decision-making, a denoising mechanism reduces noisy multilingual input interference, while a feature fusion module integrates task-

level cross-language features with node-level resource states. The reinforcement learning scheduler then optimizes decisions through a reward mechanism that balances resource utilization and response time.

The module adjusts task order and allocation based on real-time feedback. Q-learning is used for low-dimensional states, while DQN is used for larger state spaces. The state includes task priority, estimated resource demand, language pair, task type, input length, tokenization complexity, CPU, memory and GPU utilization, queue length, node load, and node availability. The action space includes task assignment, low-priority task delay, overloaded task rescheduling, and multi-node decomposition for computation-intensive tasks. The reward considers resource utilization and response time, while penalizing overload, deadline violation, excessive communication overhead, and assignment to unavailable nodes. The denoising module is implemented as a two-layer MLP with a hidden dimension of 128, ReLU activation, and a dropout rate of 0.2. The feature fusion module projects task and node representations into the same 128-dimensional latent space, and the fused representation is used as the RL state input.

Specifically, for task t_j, the original cross-language feature vector is denoted as x_j, including language pair, task type, input length, tokenization complexity, and estimated inference cost. The denoising mechanism maps it into a cleaner representation:

$$\tilde{x}_j = f_d(x_j; \theta_d) \quad (3)$$

where $f_d(\cdot)$ denotes the denoising function, θ_d represents its parameters, and $\tilde{x}_j$ denotes the denoised task representation. The denoised representation is then fused with the real-time resource state r_i of node i:

$$z_{ij} = \phi(W_x \tilde{x}_j + W_r r_i + b) \quad (4)$$

where r_i includes CPU utilization, memory utilization, GPU utilization, queue length, and current node load. The fused representation z_{ij} is used as part of the reinforcement learning state input, enabling the scheduler to jointly consider multilingual task heterogeneity and real-time resource conditions. The final scheduling decision is selected according to the learned policy, and the allocation plan is updated when node overload, task delay, or resource fluctuation occurs. The scheduling optimization module is illustrated in Figure 3.

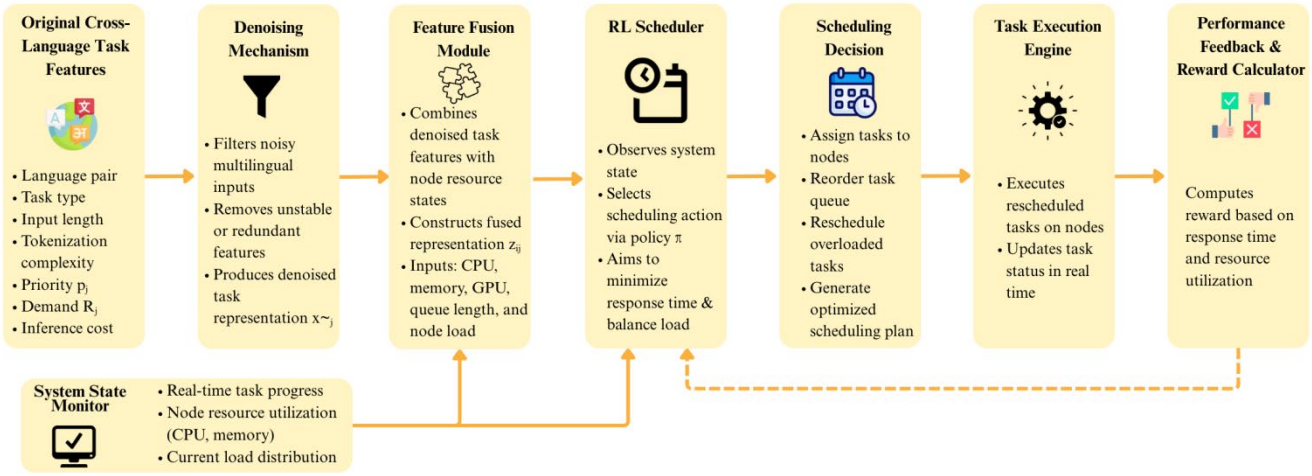

Figure 3. Scheduling Optimization Module: Denoising and Feature Fusion Enhanced Reinforcement Learning–Driven Dynamic Task Rescheduling Framework

Resource Monitoring Module

The resource monitoring module continuously monitors the resource usage of each computational node to ensure that the scheduling optimization module can make reasonable decisions based on the current resource status.

This module collects resource usage data, such as CPU, memory, GPU utilization, queue length, task completion status, and node availability, from computational nodes and passes this information to the scheduling optimization module. The monitoring data helps the scheduling module adjust task allocation in real time, avoiding resource waste or overload.

The resource monitoring module collects real-time resource data from each node via system calls or APIs, processes the data, and feeds it back to the scheduling optimization module. The collected data are normalized into node state vectors for the reinforcement learning scheduler. The monitoring interval is set according to workload intensity, and abnormal states such as overload, delayed execution, or temporary node unavailability trigger rescheduling. These monitoring results are also used to update task resource estimation and node availability during online scheduling. When GPU utilization, queue length, or response delay exceeds a predefined threshold, the task can be delayed, migrated, or reassigned. Based on the current load of each node, the scheduling module can reschedule tasks to ensure

reasonable resource allocation. The resource monitoring architecture is illustrated in Figure 4.

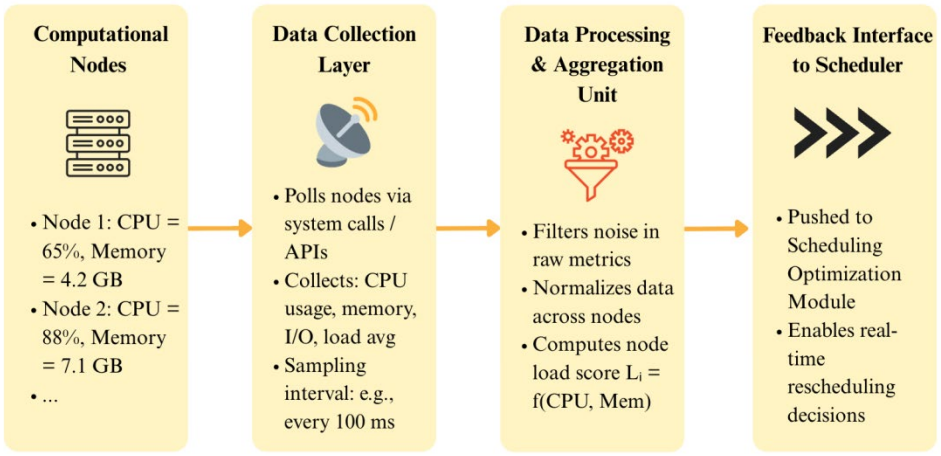

Figure 4. Architecture of the Resource Monitoring Module: Real-Time Node-Level Resource Data Collection and Feedback Pipeline

3.4. Objective Function & Optimization

In the distributed load balancing and real-time scheduling optimization of cross-language text processing tasks, the core objective is to improve resource utilization, reduce task response time, and ensure timely processing of high-priority tasks. Unlike static scheduling formulations, the proposed optimization considers dynamically estimated resource demand, real-time node status, communication overhead, and node availability.

Objective Functions

The goal of this study is to maximize resource utilization while minimizing the weighted response time of tasks.

Resource utilization measures the efficiency of resource use across computational nodes, with the goal of maximizing system resource utilization. Let C_i be the computational capacity of node i, T_i be the set of tasks assigned to node i, and $R_j(\mathrm{t})$ be the dynamically estimated resource demand of task t_j, then resource utilization is defined as:

$$Utilization = \frac{\sum_{i=1}^{n} \sum_{t_j \in T_i} R_j}{\sum_{i=1}^{n} C_i} \tag{5}$$

Task response time is a key factor influencing system performance. Let p_j be the priority of task t_j and T_j be its response time. The weighted response time is defined as::

$$Weighted\ Response\ Time = \sum_{t_j \in T} p_j \cdot T_j \tag{6}$$

To reflect inter-node data transfer and synchronization costs, communication overhead is introduced as:

$$Com = \sum_{i=1}^{n} \sum_{t_j \in T} x_{ij} \cdot Com_{ij}(t) \tag{7}$$

where $Com_{ij}(t)$ denotes the communication cost when task t_j, or its subtask, is assigned to node i, and x_{ij} indicates the assignment decision.

Optimization Objective

The optimization goal is to maximize resource utilization and minimize the weighted response time, with the objective function expressed as:

$$\max\ Utilization\ s.t. \quad \min\ Weighted\ Response\ Time \tag{8}$$

where L_{imb} denotes load imbalance among nodes, and λ_1, λ_2, and λ_3 are weight factors. This objective better reflects practical distributed scheduling than considering utilization and response time alone.

Constraints

To ensure rational scheduling, resource, assignment, response-time, and node-availability constraints are considered. The resource consumption of each node must not exceed its available capacity:

$$\sum_{t_j \in T_i} R_j(t) \leq A_i(t) \cdot C_i, \forall i \tag{9}$$

where $A_i(t) \in \{0,1\}$ denotes node availability. If $A_i(t) = 0$, the node is unavailable due to failure, overload, or temporary interruption.

For task assignment, both single-node and multi-node execution are supported. For non-decomposable tasks:

$$\sum_{i=1}^{n} x_{ij} = 1, \forall t_j \in T \quad (10)$$

For computation-intensive or decomposable tasks:

$$\sum_{i=1}^{n} x_{ij} \geq 1, \forall t_j \in T_{parallel} \quad (11)$$

where $T_{parallel}$ denotes tasks suitable for subtask or micro-batch parallel execution. The response time of each task must not exceed its threshold:

$$T_j \leq Threshold_j, \forall t_j \in T \quad (12)$$

The threshold may vary according to task type, language pair, and priority level.

Optimization Method

To solve this problem, a reinforcement learning-based scheduler is adopted. The agent interacts with the distributed environment and learns an optimized policy according to dynamic task changes and real-time resource feedback.

The state space s includes CPU utilization, memory utilization, GPU utilization, queue length, node load, node availability, task priority, $R_j(t)$, language pair, task type, input length, and tokenization complexity. The action space $a$includes assigning task $t_j$to node i, rescheduling overloaded tasks, delaying low-priority tasks, or decomposing computation-intensive tasks for multi-node execution.

Let $S = \{s_1, s_2, \dots, s_m\}$be the scheduling strategy, where s_j represents the node or node set assigned to task t_j. The optimal strategy is:

$$S^* = \arg\max(Utilization - \lambda_1 \cdot Weighted\ Response\ Time - \lambda_2 \cdot Com - s_3 \cdot L_{imb}) \quad (13)$$

The reward gives positive feedback for higher utilization, shorter response time, and better load balance, while penalizing overload, excessive communication cost, deadline violation, failed node assignment, and unavailable-node allocation.

The reinforcement learning update formula is:

$$Q(s,a) = Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right] \quad (14)$$

where $Q(s,a)$ is the value of taking action a in state s, r is the reward, γ is the discount factor, and α is the learning rate. During training, the agent updates the policy through exploration and exploitation under different workloads and resource states. In online scheduling, the overhead mainly comes from state construction and policy inference. Because compact task and node features are used, the scheduler introduces limited overhead while improving applicability to practical distributed environments.

4. Experiment and Results

4.1. Experimental Setup

To validate the proposed load balancing and scheduling optimization framework, experiments were conducted using a custom-built dataset and publicly available benchmark datasets (see Table 1). The custom-built dataset was designed for cross-language text processing tasks and covers multiple language pairs and task types to ensure workload diversity.

The dataset contains five language pairs: Chinese-English, French-English, German-English, Spanish-English, and Japanese-English. It includes machine translation, sentiment analysis, and text classification tasks. These tasks differ in input length, inference cost, memory demand, and latency sensitivity, making them suitable for evaluating heterogeneous cross-language scheduling. Each language pair contains over 100,000 samples.

Raw corpora were collected from public translation databases, social media, and news websites, followed by denoising, tokenization, annotation, and quality control through manual verification and automated tools. WMT2014 and SST-2 were also used as benchmark datasets, with WMT2014 for machine translation and SST-2 for sentiment classification. For WMT2014 and SST-2, sampled subsets were used to maintain comparable experimental scale and reduce computational cost. Compared with these public datasets, the custom-built dataset provides greater task diversity and multilingual workload complexity, especially for low-resource language-pair scenarios.

To reflect distributed scheduling, experiments were conducted in a simulated distributed cluster based on the hardware platform shown in Table 2. The cluster contained N=8 computational nodes with heterogeneous CPU, memory, GPU, queue length, initial-load, and availability states. Communication overhead and temporary node failure were randomly introduced during scheduling to approximate practical distributed environments. During each run, cross-language tasks were dynamically submitted to the task queue, and the scheduler allocated them according to task priority, dynamically estimated resource demand, and real-time node status. Each experiment was repeated 10 times, and the mean and standard deviation were reported.

Table 1. Dataset Overview

Dataset Name	Task Type	Language Pairs	Sample Size	Data Features
Custom Dataset	Cross-language Text Processing	ZH-EN, FR-EN, DE-EN, ES-EN, JA-EN	100,000+	Multi-task, Multi-language, Multi-domain
WMT2014	Machine Translation	English-German	45,000+ sampled sentence pairs	Bilingual, Standard Dataset
SST-2	Sentiment Analysis	English	15,000 sampled	Sentence-level

	examples	Sentiment Analysis

Table 2 shows the hardware and software configurations used as the base platform for the simulated distributed cluster. The software environment includes common deep learning frameworks and libraries to support reproducibility.

For reproducibility, all experiments used fixed parameters. The reinforcement learning scheduler adopted an exploration – exploitation strategy with a learning rate of 0.001, discount factor of 0.95, batch size of 64, 500 training episodes, initial exploration rate of 0.9, minimum exploration rate of 0.05, and exploration decay rate of 0.995. For the DQN-based scheduler, the hidden layer size was 128 and the optimizer was Adam. Reward weights for resource utilization, response time, communication overhead, and load imbalance were tuned on the validation set and fixed during testing. All baselines and the proposed method used the same dataset split, task queue, cluster configuration, and evaluation metrics.

Table 2. Hardware and Software Configuration

Device Name	Model	Description	Software Environment	Version
CPU	Intel Xeon Gold 6248	20 cores, supports large-scale parallel computing	Operating System	Ubuntu 20.04 LTS
GPU	NVIDIA A100	40GB VRAM, suitable for deep learning training	Deep Learning Framework	TensorFlow 2.6
Memory	256GB DDR4	Large memory to support large-scale data processing	Major Libraries	NumPy 1.21.0, SciPy 1.7.1, Pandas 1.3.2, Matplotlib 3.4.2
Storage	2TB SSD	High-speed storage for large-scale dataset loading		

To comprehensively evaluate the model's performance, we used both task-level and system-level metrics. BLEU and ROUGE were adopted for machine translation and text generation tasks, Accuracy was used for sentiment analysis and text classification, and Response Time and Resource Utilization were used to evaluate real-time scheduling efficiency and load balancing performance. Distributed scheduling metrics were also added, including scheduling latency, throughput, load imbalance, deadline violation rate, and communication overhead. These metrics jointly reflect task accuracy, execution efficiency, and system robustness.

4.2. Baselines

To evaluate the load balancing and scheduling optimization framework proposed in this study, we selected two classical methods and two state-of-the-art (SOTA) methods as baseline comparisons. The classical methods include Round-Robin (RR) scheduling and Shortest Job First (SJF) scheduling, while the SOTA methods are reinforcement learning-based dynamic scheduling and Deep Reinforcement Learning (DRL) models. These methods were chosen as benchmarks to understand their performance in a distributed environment, especially in cross-language text processing tasks.

RR scheduling periodically allocates tasks, which is simple to implement but cannot dynamically adjust task priorities, leading to resource wastage or uneven load distribution. SJF prioritizes the execution of shorter tasks, reducing response time, but ignores task priority and load variations, performing poorly in complex scheduling scenarios[24]. Reinforcement Learning-Based Dynamic Scheduling dynamically adjusts task allocation through agent-environment interactions, adapting to load changes, but requires large amounts of data and computational resources. DRL combining deep learning and reinforcement learning, DRL is suitable for large-scale task scheduling but has high computational demands, complex training, and poor interpretability[25].

For fair comparison, all baselines used the same dataset split, task queue, simulated cluster configuration, and evaluation metrics. RR assigns tasks cyclically, and SJF assigns shorter estimated tasks first. The RL baseline uses node load, queue length, task priority, and resource utilization as inputs, while the DRL baseline adopts a DQN-based scheduler with task priority, queue length, and node utilization. Unlike these baselines, the proposed method further incorporates language-pair features, tokenization complexity, estimated inference cost, denoising, and feature fusion.

Each of these baseline methods has its strengths and weaknesses. Classical methods are simple but fail to address dynamic task and resource changes, while SOTA methods are flexible but computationally expensive. In comparison, the scheduling optimization framework proposed in this paper outperforms these methods in resource utilization, response time, and system stability, especially in tasks involving low-

resource language pairs, effectively reducing computational demands while maintaining high performance.

4.3. Quantitative Results

To validate the effectiveness of the load balancing and scheduling optimization framework proposed in this study, this section presents the comparative results with classical methods and SOTA methods. The superiority of the proposed method is quantified through statistical data, charts, and significance testing. Table 3 shows the performance comparison of the proposed framework with baseline methods on key evaluation metrics. Distributed scheduling indicators, including scheduling latency, throughput, load imbalance, deadline violation rate, and communication overhead, were also evaluated. Scheduling latency reflects scheduler decision time, throughput represents completed tasks per minute, load imbalance measures the variance of node utilization, and communication overhead is reported as a normalized relative cost.

Table 3. Key Performance Comparison

Method	Resource Utilization	Response Time	Task Completion Rate	Scheduling Latency	Throughput	Load Imbalance	Deadline Violation Rate	Communication Overhead	p-value RT	p-value RU
RR	72.5% ± 3.4%	105.2s ± 10.5s	85.4% ± 5.1%	17.9ms ± 2.8ms	43.6 ± 4.7 tasks/min	0.238 ± 0.041	12.6% ± 2.4%	1.00 ± 0.13	Baseline	Baseline
SJF Scheduling	78.3% ± 4.1%	98.4s ± 8.3s	88.6% ± 4.4%	13.7ms ± 2.5ms	45.1 ± 4.0 tasks/min	0.196 ± 0.036	9.4% ± 2.1%	0.97 ± 0.12	0.014	0.045
Reinforcement Learning Scheduling	84.9% ± 2.8%	90.3s ± 6.1s	92.2% ± 3.6%	15.4ms ± 1.7ms	52.4 ± 3.9 tasks/min	0.127 ± 0.026	7.8% ± 1.4%	0.86 ± 0.09	0.001	0.002
DRL	88.6% ± 2.5%	85.6s ± 5.4s	94.3% ± 3.2%	12.1ms ± 1.9ms	55.6 ± 3.6 tasks/min	0.112 ± 0.022	6.9% ± 1.6%	0.79 ± 0.10	0.005	0.009
Proposed Method	91.2% ± 2.1%	79.8s ± 4.3s	96.5% ± 2.8%	11.4ms ± 1.5ms	60.9 ± 3.2 tasks/min	0.081 ± 0.018	4.7% ± 1.1%	0.68 ± 0.08	<0.001	<0.001

As shown in Table 3, the proposed method outperforms classical and SOTA methods in resource utilization, response time, task completion rate, and distributed scheduling indicators. Specifically, it achieves a response time of 79.8s, reducing response time by approximately 11.6% and 6.8% compared with reinforcement learning scheduling and DRL, respectively. Its resource utilization reaches 91.2%, improving by 7.4 percentage points and 2.9 percentage points compared with reinforcement learning scheduling and DRL. In addition, the proposed method achieves lower scheduling latency, load imbalance, deadline violation rate, and communication overhead, while maintaining higher throughput. These results indicate that incorporating cross-language features into reinforcement learning-based scheduling improves both task-level performance and system-level stability under heterogeneous distributed conditions.

The statistical results further indicate that the improvements of the proposed method over the RR baseline in response time and resource utilization are significant and consistent across repeated runs. The large effect sizes further support the practical significance of the observed improvements.

Table 4. Statistical Support for Paired t-Test Results (n = 10)

Metric	Comparison	Mean Difference	95% CI	Cohen's d	p-value
Response Time	Proposed vs RR	−25.4s	[−32.1, −18.6]	1.32	<0.001
Resource Utilization	Proposed vs RR	+18.7%	[12.4, 25.1]	1.41	<0.001

Figure 5 shows the scheduling objective convergence curves of different methods. The scheduling objective value of the proposed framework decreases rapidly within the first 50 iterations and then gradually stabilizes, indicating that the reinforcement learning scheduler can learn a stable task allocation policy within fewer iterations. This faster convergence is mainly attributed to the feature fusion module, which integrates cross-language task attributes and node-level resource states to provide more informative state

representations for policy learning. The proposed method achieves a lower final objective value (0.298) than the DRL model (0.540), while maintaining a smoother convergence trend. The convergence behavior is evaluated by observing both the decreasing objective trend and the final stable objective value, which reflect the training efficiency and stability of the learned scheduling policy. These results suggest that incorporating cross-language features into the reinforcement learning state representation improves policy learning efficiency and scheduling stability.

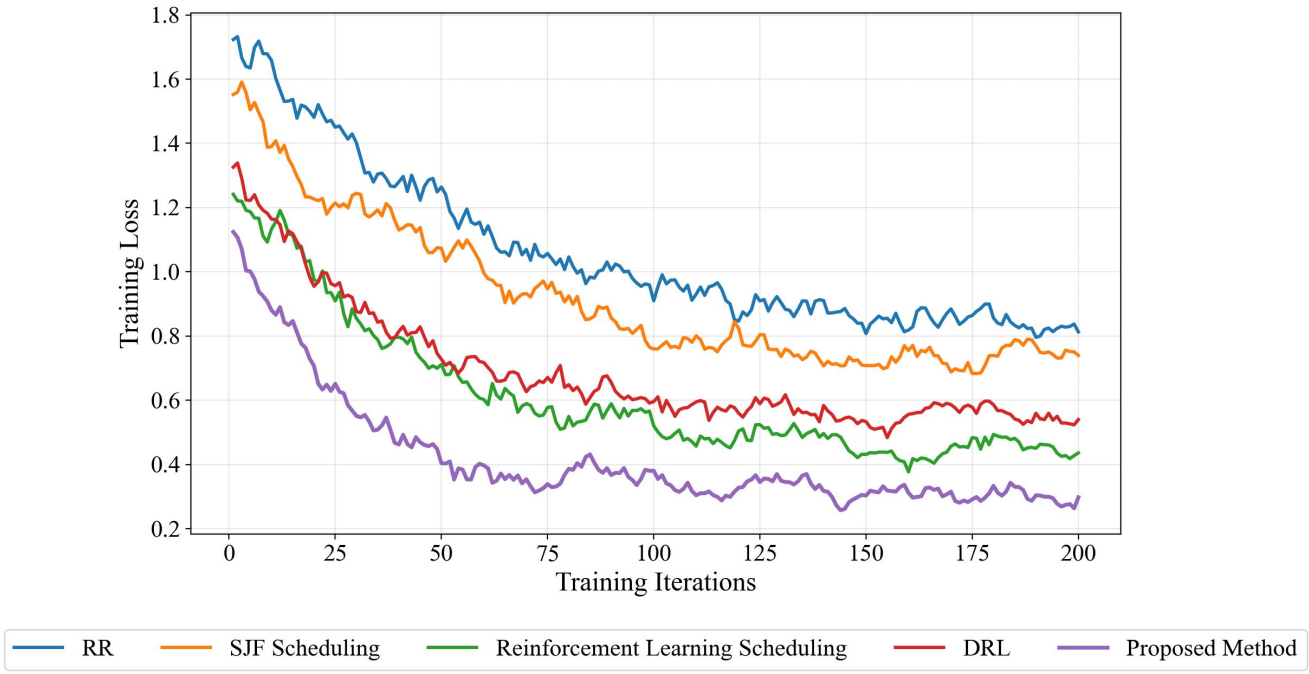

Figure 5. Scheduling Objective Convergence Curves of Different Methods

4.4. Qualitative Results

This section presents the advantages and limitations of the scheduling optimization framework in cross-language text processing tasks through visual case studies. It combines data analysis of successful and failed cases and compares them with baseline methods, such as reinforcement learning-based scheduling.

Figure 6 shows a successful scheduling case in the French-English translation task. Compared to the RR baseline (resource utilization: 72.5%; response time: 105.2s), the proposed framework achieved a resource utilization of 91.2% ± 2.1%, an increase of 18.7 percentage points, and reduced response time to 79.8s ± 4.3s, a 24.1% improvement. The experiment demonstrated that the framework effectively avoided resource wastage and ensured high-priority tasks were completed on time by dynamically adjusting task allocation. The visualization shows that the model successfully aligned the key content between the source and target languages, and the load was evenly distributed across nodes, optimizing system efficiency. Compared to the reinforcement learning-based scheduling method, the proposed method performed better in both response time and resource utilization.

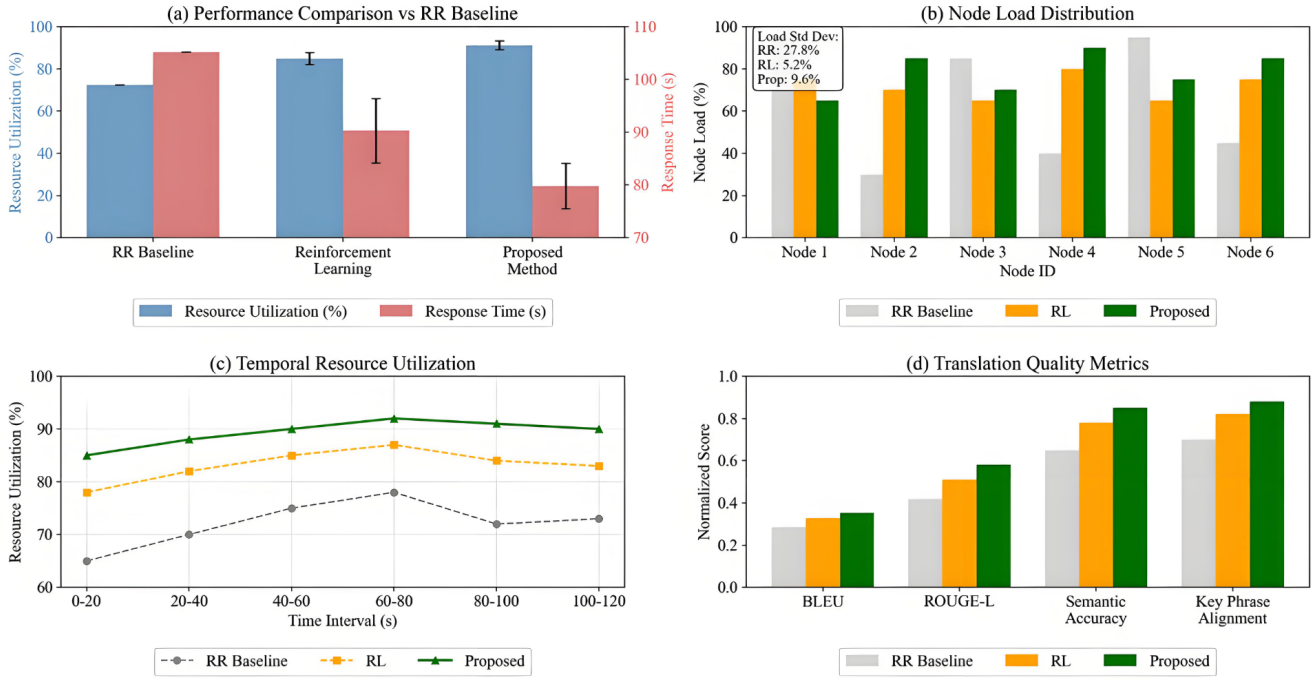

Figure 6. Comprehensive Analysis of Scheduling Performance in French-English Translation Task: (a) Resource Utilization and Response Time Comparison; (b) Node Load Distribution; (c) Temporal Evolution of Resource Utilization; (d) Translation Quality Metrics

Figure 7 illustrates a typical scheduling failure case. In this scenario, the scheduling strategy failed to respond to the dynamic load changes of the nodes in a timely manner, causing low-priority tasks to occupy excessive computational resources. As a result, the response time for a high-priority task increased by 8.3%, reaching 96.5s ± 3.2s, compared to its theoretical optimal completion time. Although the overall average task completion rate remained 96.5%, this delay in a critical task negatively impacted overall workflow efficiency. This case highlights that the current scheduling strategy lacks agility in resource reallocation when facing sudden load fluctuations. By contrast, the reinforcement learning-based scheduling method showed a response time increase of over 12% under the same conditions. This underscores the challenge of achieving efficient and robust scheduling in extreme dynamic environments.

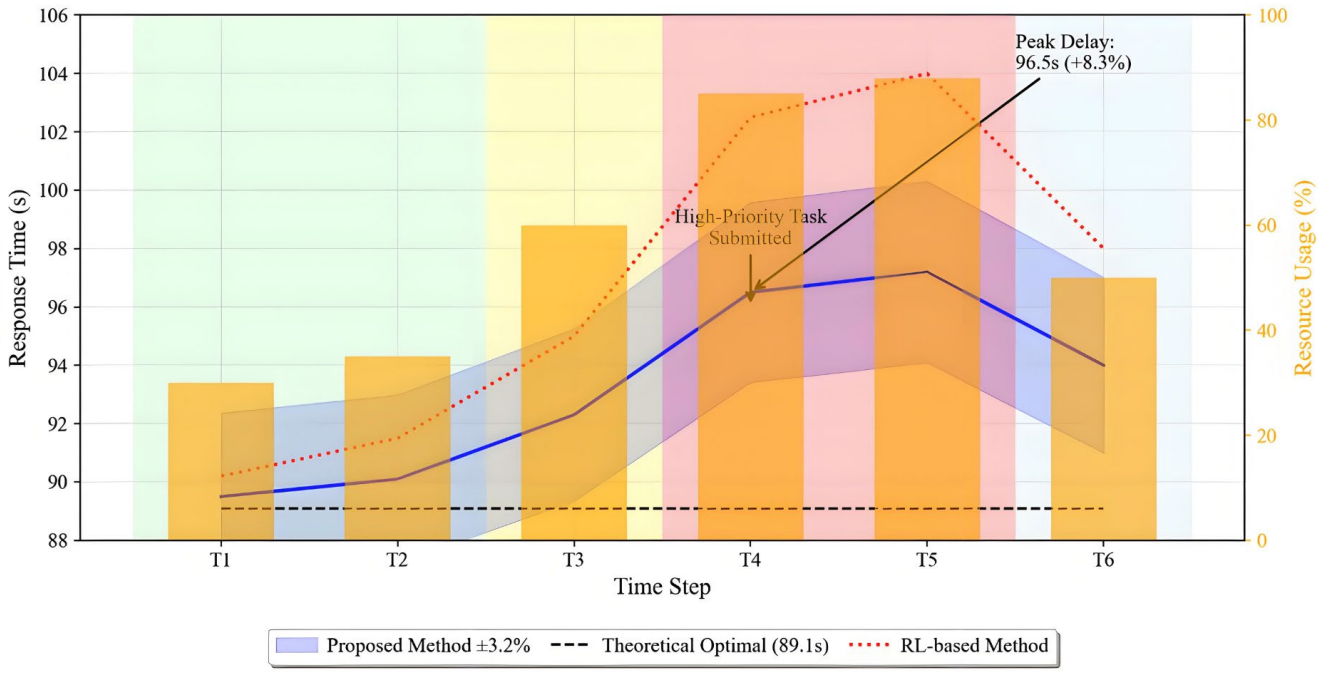

Figure 7. Scheduling Failure Case: Load Surge and High-Priority Task Delay

The successful case demonstrates that the proposed method significantly improved resource utilization (from 72.5% to 91.2%) and reduced response time (from 105.2s to 79.8s) under typical scenarios compared to the RR baseline. It also outperformed the reinforcement learning-based scheduling method in multi-language task scheduling. The failed case reveals that the flexibility of the current scheduling strategy still needs improvement when handling abrupt load changes. Future improvements should focus on enhancing the framework's adaptability to dynamic loads to ensure efficient and robust task scheduling.

4.5. Robustness and Sensitivity Analysis

To verify the stability and sensitivity of the proposed scheduling framework under non-ideal conditions, we designed multi-task, high-noise, and multi-dataset scenarios. In this study, noise refers to disturbances in multilingual text processing and distributed scheduling, including partial text loss, abnormal input length, labeling errors, and inaccurate resource demand estimation. These factors simulate imperfect multilingual inputs and uncertain execution conditions in real-world distributed environments.

In the multi-task scenario, the model handles translation, sentiment analysis, and text classification tasks. Noise was simulated by randomly corrupting a proportion of task samples or scheduling-related attributes. Specifically, partial tokens were removed, some labels were randomly changed, and estimated resource demands were perturbed. Noise intensity was defined as the proportion of affected samples or attributes; for example, 10% noise means that 10% of task inputs or scheduling attributes were disturbed, while 50% represents a high-noise condition. The method was evaluated on the custom-built dataset, WMT2014, and SST-2.

Figure 8 presents the sensitivity analysis under different noise intensities. When noise increased from 10% to 50%, the task completion rate of the proposed method decreased from approximately 95.2% to 88.7%, a drop of 6.5 percentage points. In contrast, the reinforcement learning-based dynamic scheduling method decreased from approximately 93.1% to 80.4%, a drop of 12.7 percentage points. The response time of the proposed method also increased more slowly.

The stable performance is mainly attributed to the denoising mechanism and feature fusion module. The denoising mechanism reduces the interference of noisy multilingual inputs and unstable task attributes, while feature fusion improves tolerance to incomplete task information by integrating task-level features with node-level resource states. These settings reflect realistic multilingual distributed scenarios, where texts may be incomplete, labels imperfect, and resource demand different from initial estimates.

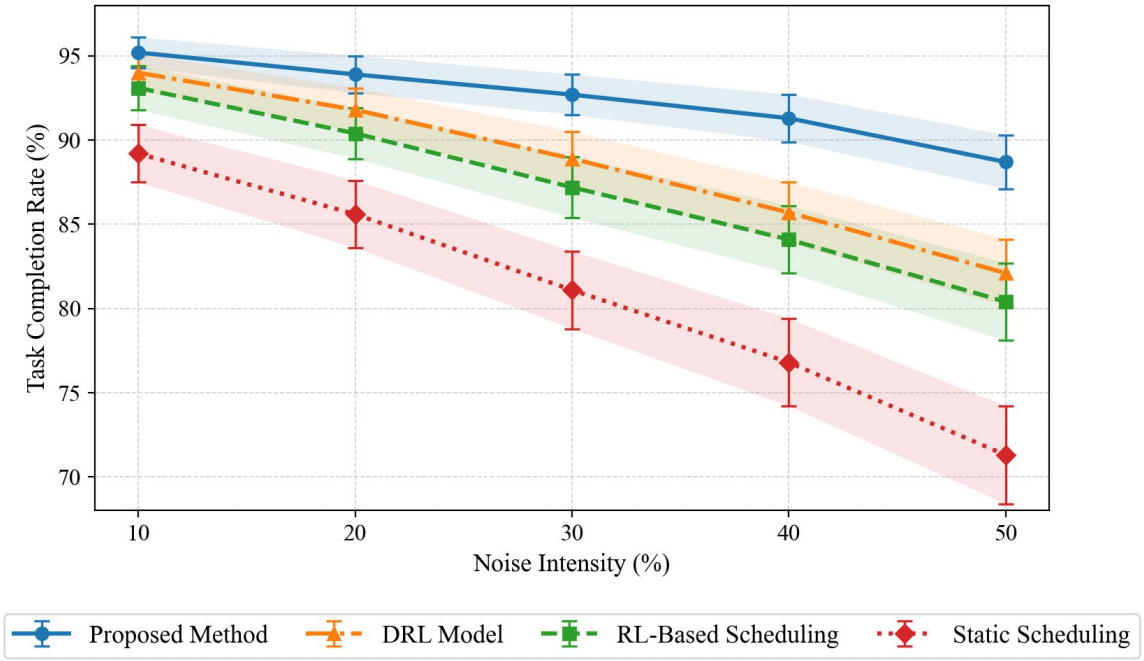

Figure 8. Impact of Noise Intensity on Performance in Multi-Task Scenario

Compared with baseline methods, the proposed method performs more stably in high-noise and multi-task environments. This indicates that the proposed framework is less sensitive to noisy multilingual inputs and uncertain resource estimates, demonstrating its robustness in dynamic distributed scheduling environments.

4.6. Ablation Study

To demonstrate the individual contributions of the main components, we analyze the effects of the reinforcement learning-based scheduler, denoising mechanism, and feature fusion module. The contribution of reinforcement learning is reflected by the comparison between static scheduling methods and reinforcement learning-based scheduling methods in Table 3. To further quantify the effects of the denoising mechanism and feature fusion module, an ablation study was conducted. In this experiment, the reinforcement learning scheduler without denoising and feature fusion was used as the baseline. The two modules were then added separately and jointly to observe their effects on scheduling performance. Table 5 shows the impact of different module configurations.

Table 5. Ablation Study Results

Module or Component	Resource Utilization (Mean ± Std)	Response Time (Mean ± Std)	Task Completion Rate (Mean ± Std)	p-value (Response Time)	p-value (Resource Utilization)
RL Scheduler Only	84.9% ± 2.8%	90.3s ± 6.1s	92.2% ± 3.6%	Baseline	Baseline
RL + Denoising Mechanism	87.1% ± 2.9%	86.7s ± 5.8s	93.4% ± 3.3%	0.031	0.044
RL + Feature Fusion Module	89.0% ± 2.6%	84.1s ± 5.2s	94.6% ± 3.1%	0.018	0.026
RL + Denoising + Feature Fusion (Proposed Method)	91.2% ± 2.1%	79.8s ± 4.3s	96.5% ± 2.8%	<0.001	0.002

From the ablation results, both the denoising mechanism and feature fusion module improve scheduling performance. Compared with the RL scheduler only, adding the denoising mechanism increases resource utilization from 84.9% to 87.1%, reduces response time from 90.3s to 86.7s, and improves task completion rate from 92.2% to 93.4%. This indicates that denoising helps reduce the influence of noisy multilingual inputs and uncertain resource estimates.

Adding the feature fusion module produces stronger improvements, increasing resource utilization to 89.0%, reducing response time to 84.1s, and improving task completion rate to 94.6%. This suggests that integrating cross-language task attributes with node-level resource states provides more informative scheduling representations. The complete framework achieves the best performance, with 91.2% resource utilization, 79.8s response time, and 96.5% task completion rate. These results confirm the complementary effects of denoising and feature fusion in improving robustness, resource allocation, and scheduling stability.

Overall, the ablation study verifies the individual and combined contributions of the proposed components. Reinforcement learning provides the basic dynamic scheduling ability, denoising improves robustness under noisy multilingual conditions, and feature fusion enhances state representation for more effective task allocation.

5. Discussion

The dynamic load balancing and scheduling optimization framework based on reinforcement learning demonstrates clear performance improvements in cross-language text processing tasks. Experimental results show that the proposed method outperforms classical and SOTA methods in response time, resource utilization, and task completion rate. In high-noise environments, it also shows stronger robustness, with a smaller decline in task completion rate than baseline methods.

The results indicate that the adaptive scheduling strategy, denoising mechanism, and feature fusion module jointly improve system performance. The denoising mechanism reduces noisy multilingual input interference, while the feature fusion module integrates cross-language task features and node-level resource states to support more stable scheduling decisions. Compared with conventional reinforcement learning-based scheduling, the proposed method performs better in dynamic load environments because it considers both resource feedback and language-dependent workload heterogeneity.

The computational overhead mainly comes from state construction, policy inference, and policy updating during training. In online scheduling, compact task-level and node-level features are used instead of full text representations, so the additional overhead is relatively limited. This improves engineering feasibility because online scheduling does not require full-text semantic encoding for every decision. However, in very large clusters, scheduler-node communication, task migration, multi-node synchronization, and state updating may introduce extra deployment costs. Thus, the current framework improves cross-language-aware dynamic allocation, but further engineering optimization is still needed for very large-scale industrial clusters.

However, this study still has limitations. First, the scalability of the framework under extremely high-load conditions and very large clusters requires further validation, especially in environments with hundreds or thousands of nodes. Second, reinforcement learning-based scheduling involves training costs, and the interpretability of the learned policy remains limited. Long-term

deployment may also face model drift when workload patterns, language distributions, or resource conditions change. Third, real-world systems may involve communication overhead, node failures, task migration costs, multi-node parallel execution constraints, and fairness issues in multilingual task allocation. Although the revised formulation supports subtask-level and multi-node execution, its effectiveness in physical large-scale LLM deployment still requires validation. Lastly, the dataset mainly focuses on cross-language text processing tasks, and future work should extend the framework to other domains.

Future research could optimize reinforcement learning algorithms to reduce overhead and improve interpretability. Enhancing scalability, incorporating communication-aware scheduling constraints, introducing fault-tolerant mechanisms, multi-node parallel deployment support, drift detection, and fairness-aware scheduling are also important directions. In addition, extending the framework to cross-modal tasks may broaden its application scope. In summary, this study provides useful insights into cross-language-aware load balancing and real-time scheduling for distributed text processing systems, and offers a system-level perspective for improving the efficiency, robustness, and deployability of multilingual NLP services.

6. Conclusion

This study addresses load balancing and real-time scheduling for cross-language text processing tasks in distributed systems and proposes a cross-language-aware dynamic scheduling framework based on reinforcement learning. By integrating adaptive scheduling, denoising, and feature fusion mechanisms, the framework improves scheduling efficiency and robustness under multi-task and noisy multilingual environments. Experimental results show that the proposed method outperforms classical and SOTA baselines, reducing response time by 6.7% compared with the DRL baseline, improving resource utilization from 88.6% to 91.2%, and achieving a task completion rate of 96.5%.

The main contributions are threefold. First, a reinforcement learning-based scheduling strategy is developed to adapt to task changes, load fluctuations, and real-time node states. Second, denoising and feature fusion modules are introduced to enhance robustness under noisy multilingual inputs and uncertain resource estimation. Third, cross-language task characteristics are coupled with distributed resource states to support more effective resource allocation for heterogeneous multilingual workloads. The results confirm the advantages of the proposed method in response time, resource utilization, task completion rate, and system stability.

Academically, this study extends cross-language NLP research from task-level performance optimization to system-level scheduling efficiency. By incorporating multilingual task features into the reinforcement learning state representation and reward design, it provides insights into domain-aware resource management in distributed systems. Practically, the framework offers a learnable scheduling paradigm for dynamic multilingual task flows and supports the integration of task semantics, language-pair complexity, data quality, and real-time resource feedback into scheduling decisions.

Future work will validate the framework in larger-scale and more realistic distributed environments. Key directions include reducing computational and communication overhead, improving fault tolerance, supporting large-scale multi-node parallel execution, and enhancing policy interpretability. Overall, this study provides a cross-language-aware scheduling solution for distributed text processing and offers a practical foundation for future multilingual NLP service deployment.

Acknowledgements

This work was sponsored by Tianshui Normal University High-level Talent Research Project “Translation of Cultural Phenomena with Regional Characteristics in Countries Along the‘Belt and Road' Routes From the Perspective of Cultural Exchanges and Mutual Learning”(KYQ2023-11); sponsored by Tianshui Normal University School-Level Research Project “ Study on Chinese Discourse in the Complete Works of a Global History”（GJB2024-10）

References

[1] Nawaz, A. (2025). Advancements in Natural Language Processing for Multilingual Information Retrieval Systems. Multidisciplinary Research in Computing Information Systems, 5(3), 197-216.

[2] Park, G., Park, J., & Lee, H. (2025). Cross-Lingual Summarization for Low-Resource Languages Using Multilingual Retrieval-Based In-Context Learning. Applied Sciences, 15(14), 7800.

[3] Jain, D. (2025). Multilingual and Cross-Linguistic Challenges in NLP. In Transformative Natural Language Processing: Bridging Ambiguity in Healthcare, Legal, and Financial Applications (pp. 157-177). Cham: Springer Nature Switzerland.

[4] Shubham, M. (2024). Breaking Language Barriers: Advancements in Machine Translation for Enhanced Cross-Lingual Information Retrieval. J. Electrical Systems, 20(9s), 2860-2875.

[5] Zhou, G., Tian, W., Buyya, R., Xue, R., & Song, L. (2024). Deep reinforcement learning-based methods for resource scheduling in cloud computing: A review and future directions. Artificial Intelligence Review, 57(5), 124.

[6] Albalawi, N. S. (2025). Dynamic scheduling strategies for cloud-based load balancing in parallel and distributed systems. Journal of Cloud Computing, 14(1), 33.

[7] Li, T., Ying, S., Zhao, Y., & Shang, J. (2023). Batch jobs load balancing scheduling in cloud computing using distributional reinforcement learning. IEEE Transactions on Parallel and Distributed Systems, 35(1), 169-185.

[8] Zhang, S., Zhao, Z., Liu, C., & Qin, S. (2023). Data-intensive workflow scheduling strategy based on deep reinforcement learning in multi-clouds. Journal of Cloud Computing, 12(1), 125.

[9] Devi, N., Dalal, S., Solanki, K., Dalal, S., Lilhore, U. K., Simaiya, S., & Nuristani, N. (2024). A systematic literature review for load balancing and task scheduling techniques in cloud computing. Artificial Intelligence Review, 57(10), 276.

[10] Liu, J., Li, K., Zhu, A., Hong, B., Zhao, P., Dai, S., ... & Su, H. (2024). Application of deep learning-based natural language processing in multilingual sentiment analysis. Mediterranean Journal of Basic and Applied Sciences (MJBAS), 8(2), 243-260.

[11] Huang, Z., Yu, P., & Allan, J. (2023, February). Improving cross-lingual information retrieval on low-resource languages via optimal transport distillation. In Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining (pp. 1048-1056).

[12] Xu, N., & Wu, Y. (2024, October). Progress in Cross Language Text Sentiment Analysis Based on Deep Learning. In 2024 3rd International Conference on Data Analytics, Computing and Artificial Intelligence (ICDACAI) (pp. 321-325). IEEE.

[13] Agostinelli, V., Alumäe, T., Anastasopoulos, A., Bentivogli, L., Bojar, O., Borg, C., ... & Züfle, M. (2025, July). Findings of the IWSLT 2025 evaluation campaign. In Proceedings of the 22nd International Conference on Spoken Language Translation (IWSLT 2025) (pp. 412-481).

[14] Papi, S., Gaido, M., Karakanta, A., Cettolo, M., Negri, M., & Turchi, M. (2023). Direct speech translation for automatic subtitling. Transactions of the Association for Computational Linguistics, 11, 1355-1376.

[15] Mondal, S. K., Zhang, H., Kabir, H. D., Ni, K., & Dai, H. N. (2023). Machine translation and its evaluation: a study. Artificial Intelligence Review, 56(9), 10137-10226.

[16] NLLB Team. (2024). Scaling neural machine translation to 200 languages. Nature, 630, 841–846.

[17] Guo, X., Adnan, H. M., & Abidin, M. Z. Z. (2024). Detecting offensive language on malay social media: A zero-shot, cross-language transfer approach using dual-branch mbert. Applied Sciences, 14(13), 5777.

[18] Nguyen, N., Takeda, H., & Karunathilake, L. (2025). Empowering Weak Languages Through Cross-Language Hyperlink Recommendation. Information, 16(9), 749.

[19] Gardazi, N. M., Daud, A., Malik, M. K., Bukhari, A., Alsahfi, T., & Alshemaimri, B. (2025). BERT applications in natural language processing: a review. Artificial Intelligence Review, 58(6), 1-49.

[20] Raeisi-Varzaneh, M., Dakkak, O., Fazea, Y., & Kaosar, M. G. (2024). Advanced cost-aware Max–Min workflow tasks allocation and scheduling in cloud computing systems. Cluster computing, 27(9), 13407-13419.

[21] Esmaeili, M. E., Khonsari, A., Sohrabi, V., & Dadlani, A. (2024). Reinforcement learning-based dynamic load balancing in edge computing networks. Computer Communications, 222, 188-197.

[22] Wang, J., Li, S., Zhang, X., Wu, F., & Xie, C. (2024). Deep reinforcement learning task scheduling method based on server real-time performance. PeerJ Computer Science, 10, e2120.

[23] Mangalampalli, S., Karri, G. R., Ratnamani, M. V., Mohanty, S. N., Jabr, B. A., Ali, Y. A., ... & Abdullaeva, B. S. (2024). Efficient deep reinforcement learning based task scheduler in multi cloud environment. Scientific Reports, 14(1), 21850.

[24] Shen, W., Lin, W., Wu, W., Wu, H., & Li, K. (2025). Reinforcement learning-based task scheduling for heterogeneous computing in end-edge-cloud environment. Cluster Computing, 28(3), 179.

[25] Jalali Khalil Abadi, Z., Mansouri, N., & Javidi, M. M. (2024). Deep reinforcement learning-based scheduling in distributed systems: a critical review. Knowledge and Information Systems, 66(10), 5709-5782.