Research on Optimization of Efficient Animation Video Production Process Driven by Cloud Computing

Lili Zhang, Yixin Pan*, Qingxi Liu, Jiahui Wang

Harbin Institute of Information Technology, Harbin, 150431, China

*Corresponding author: 370212808@qq.com

Abstract. To address scattered computing power, repeated asset transfer, low material reuse and weak collaboration in traditional animation production, this paper proposes a cloud computing-driven optimization scheme. The research boundary is clarified as the workflow from asset import, online editing, task decomposition and distributed rendering to collaborative review and final output. A six-layer architecture separates user interaction, gateway access, business services, middleware support, elastic computing and external integration. A priority-aware distributed rendering scheduling policy is further formulated by considering frame complexity, deadline urgency, resource load, task dependency and material locality. The system integrates Spring Boot, Nacos, Spring Cloud Alibaba, Kubernetes, Redis, MinIO, AI-assisted generation and real-time collaboration. Compared with a centralized baseline whose average 4K frame rendering time is 31.8 s, the proposed system reduces the time to 12.5 s, lowers the P95 collaborative response time from 2.46 s to 0.91 s and keeps the final unrecovered task error rate at 0.01% in distributed node failure tests. These results show that the proposed scheme improves rendering efficiency, material access and collaboration reliability.

Keywords: Cloud Computing; Animation Video; Efficient Production; Distributed Rendering; Real-time Collaboration; Process Optimization

1 Introduction

1.1 Research Background

The rapid growth of digital media has increased the demand for animation and video content, while traditional workstation-based workflows still face high cost, fragmented computing power, repeated file transfer, version conflicts and inefficient cross-region collaboration. Existing systems usually optimize a single link such as rendering or storage, but the boundary of a full-process cloud production workflow remains insufficiently defined. Therefore, this study focuses on how cloud computing, distributed rendering, AI assistance and collaboration can be organized into a controllable animation production process[1].

R. Zur et al. (eds.), *Proceedings of the 2026 5th International Conference on Art Design and Digital Technology (ADDT 2026)*, Advances in Computer Science Research 131,
https://doi.org/10.2991/978-94-6239-737-8_72

1.2 Research Objectives and Contributions

This paper constructs a cloud computing-driven animation production system and defines its scope as online resource management, original-painting and model editing, task splitting, distributed rendering, collaborative review, version control and output management. The main contributions are: (1) a full-process production boundary is defined from material import to final delivery; (2) a six-layer architecture is mapped to production responsibilities and fault boundaries; (3) a priority-aware scheduling policy is proposed as a concrete algorithmic contribution for distributed rendering; and (4) a quantitative evaluation is added to report baseline metrics, storage-cluster specifications and distributed failure-test error rates.

2 Related Work

Cloud computing provides elastic computing power, pay-as-you-go deployment and global access, and has become important infrastructure for digital content production. Cloud rendering farms improve efficiency through cluster computing, task fragmentation and elastic scaling, while AI technologies support original-painting generation, action recommendation and shot planning. Real-time collaboration and cloud version management further enable cross-regional creation. However, existing studies often focus on isolated tools or single-stage optimization, and they rarely connect middleware governance, material hosting, rendering scheduling and fault recovery in a unified animation workflow. Interactive and explainer-video studies also emphasize structured content creation and reusable resources[4-7]. This study therefore links layered architecture with a specific scheduling policy and measurable experimental analysis. At a broader level, studies on data-element/AI integration and digital-technology empowerment in cultural industries further indicate that digital transformation is not limited to a single production step [2,3].

3 System Architecture Design and Key Technologies

3.1 Overall Architecture Design

The system adopts a six-layer architecture covering the front-end layer, access gateway layer, business microservice layer, middleware support layer, elastic computing layer and external integration layer. The front-end layer supports online drawing, model preview and collaborative interaction; the gateway layer handles identity authentication, routing and traffic control; the microservice layer encapsulates project, material, rendering and user services; the middleware layer provides message queue, cache, object storage and service registration; the elastic computing layer undertakes rendering and AI computing; and the external integration layer connects cloud storage, rendering engines and third-party AI services. The six layers are necessary because interaction, access, business logic, data communication, computing execution and ex-

ternal capability integration have different scaling rules and fault boundaries. The overall system architecture is shown in Figure 1.

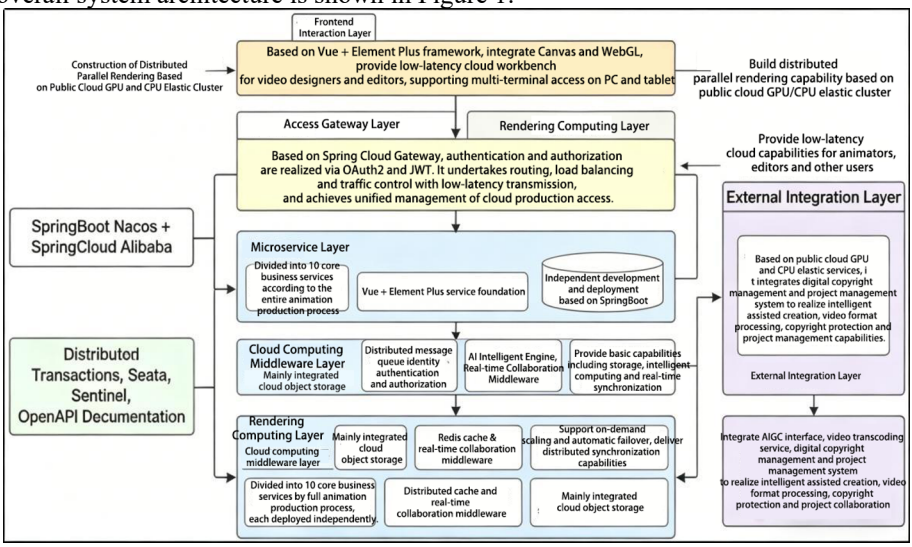

Fig. 1. Overall System Architecture Design

The design boundary is limited to the animation production workflow from asset import to final video output. Spring Boot, Nacos and Spring Cloud Alibaba are used for service governance, configuration management, routing and load balancing; they do not replace the rendering algorithm, AI model or storage engine. This distinction separates business orchestration from computing execution and makes the proposed design boundary explicit.

3.2 Key Technical Mechanisms

The system includes four key technical modules: cloud-based process reconstruction, distributed rendering scheduling, AI-assisted production and real-time collaborative management. Cloud-based reconstruction defines each production stage; distributed rendering decomposes shots and frames into executable blocks; AI assistance supports image generation, lens recommendation and material retrieval; and collaboration management records operations, permissions and versions. The architecture of the core technical mechanism is shown in Figure 2.

The distributed rendering scheduling policy is defined as follows. After a shot is imported, frames are divided into blocks according to shot boundary, resolution, material dependency and estimated complexity. The scheduler calculates priority score $S(i)=aC(i)+bU(i)+cP(i)+dD(i)$, where C(i), U(i), P(i) and D(i) denote frame complexity, deadline urgency, user priority and dependency readiness, and $a+b+c+d=1$. Candidate GPU nodes are filtered by available memory, plug-in environment, queue length and data locality. Each block is assigned to the node with the smallest estimated completion time $E(i,j)=Q(j)+W(i)/R(j)+T(i,j)$, where Q(j) is queue waiting time,

W(i) is block workload, R(j) is node rendering capability and T(i,j) is material-transfer cost. Failed tasks are automatically re-queued, and completed blocks are merged according to frame order and checksum verification. This formulation turns distributed scheduling into a repeatable algorithmic contribution rather than a general system description.

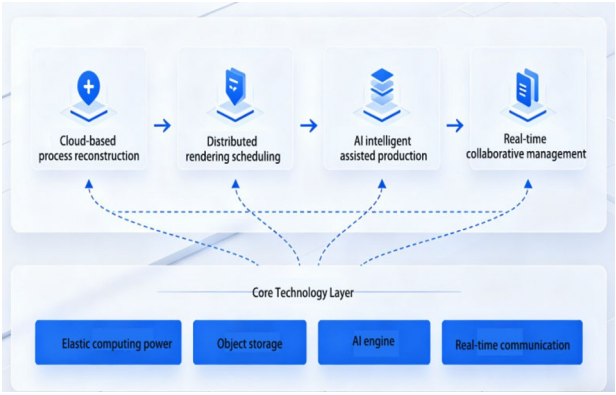

Fig. 2. Architecture of Core Technical Mechanism

4 System Implementation and Functional Modules

4.1 Core Functional Modules

The system covers the entire life cycle of animation and video production, including 8 core functional modules such as original painting design, model management, animation binding, rendering scheduling, material management and collaborative service support. These modules correspond to the six-layer architecture and jointly support asset creation, process control, rendering execution and final output. The core functional modules of the system are shown in Table 1.

Table 1. System Core Functional Modules

No.	Core Modules	Module Description
1	Original Painting Design	Cloud online original painting drawing, line draft editing, coloring and layer management
2	Model Management	3D model uploading, preview, binding, modification and version control
3	Animation Binding	Character skeleton binding, key frame setting and automatic action generation
4	Scene Editing	Cloud scene construction, light setting, lens adjustment and special effect addition
5	Rendering Scheduling	Rendering task creation, fragmentation, distribution, progress monitoring and AI-assisted recommendation
6	Post Editing	Cloud video editing, transition, dubbing, subtitle and filter processing

7	Material Management	Cloud material storage, intelligent retrieval, classification, permission control and reuse
8	Collaboration	Real-time collaboration, computing scheduling, distributed communication, permission control and data security

4.2 Database Design

The database adopts a read-write separation architecture of 1 master and 3 slaves to improve data access efficiency and reliability. MySQL is used as the main database and Redis as the cache database. The core data tables are shown in Table 2.

Table 2. Core Data Tables

Table Name	Core Fields
project	project_id (primary key), project_name, type, create_user, status, create_time, update_time
material	material_id (primary key), name, type, style, size, storage_path, upload_time
user	user_id (primary key), name, role, department, phone, email, create_time
render_task	task_id (primary key), project_id, frame_num, resolution, status, start_time, end_time
collaborate_record	record_id (primary key), project_id, user_id, operation, time, version
model	model_id (primary key), name, type, poly_num, binding_status, update_time
video	video_id (primary key), project_id, resolution, duration, format, output_time

4.3 Deployment Architecture

The system relies on containerization and Kubernetes elastic deployment, equipped with dual gateway nodes and eight redundant microservice nodes. Spring Boot implements service units, Nacos provides registration and configuration governance, and Spring Cloud Alibaba supports gateway routing, load balancing and service fault tolerance. The rendering cluster executes GPU rendering tasks, the Redis cluster stores scheduling states and distributed locks, and the MinIO object-storage cluster hosts original paintings, models, textures, audio and video clips. In the experimental deployment, the storage cluster uses four nodes with erasure-coded storage; each node provides a 16-core CPU, 128 GB memory, two 1.92 TB NVMe cache disks and eight 8 TB enterprise disks through a 25 GbE internal network. This configuration clarifies the material-hosting hardware boundary and separates asset storage from rendering execution. The system deployment topology is shown in Figure 3.

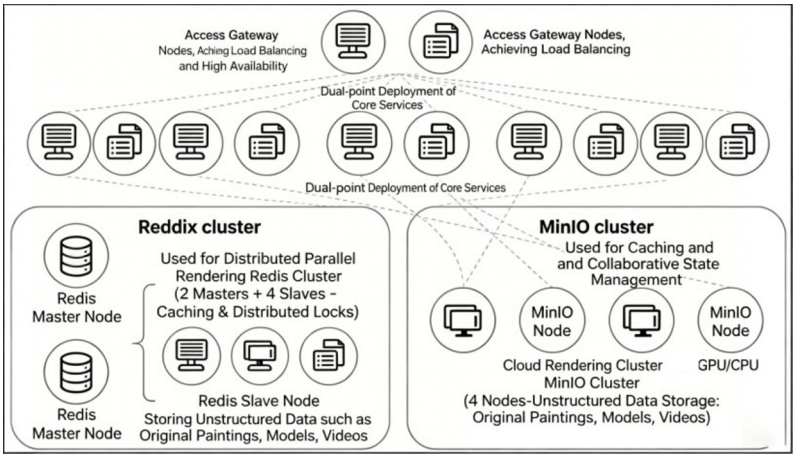

Fig. 3. System Deployment Topology

5 Experimental Design and Result Analysis

5.1 Experimental Environment and Baseline Configuration

The experiment was designed to evaluate rendering performance, material access, collaboration load and fault recovery under the same 4K animation scene. The baseline system used a centralized file server and a non-elastic rendering queue, while the proposed system used Kubernetes-based GPU scheduling, Redis state caching and MinIO object storage. Both systems rendered the same 300-frame test sequence with identical resolution, plug-ins, material packages, lighting settings and output format, so the 12.5 s rendering result is compared with a clear baseline rather than reported as an isolated value. The experimental environment and baseline configuration are summarized in Table 3.

Table 3. Experimental Environment

Hardware / Software	Configuration Description
Rendering nodes	8 GPU nodes; each node uses Intel Xeon Platinum 8369B CPU, NVIDIA A100 40 GB GPU, 256 GB DDR4 memory and 1.92 TB NVMe local cache
Storage cluster nodes	4 MinIO object-storage nodes for animation materials; each node has a 16-core CPU, 128 GB memory, 2 x 1.92 TB NVMe cache disks and 8 x 8 TB enterprise disks
Network	25 GbE internal network for render-node, Redis and MinIO communication
Operating system	CentOS 7.9 with Kubernetes container deployment
Database / cache	MySQL 8.0 with one master and three slaves; Redis 7.0 cluster
Testing tools	JMeter and a self-developed rendering efficiency testing tool

5.2 Result Analysis and Baseline Comparison

The baseline metrics were recorded before applying the proposed cloud optimization. Its average rendering time for a single 4K frame was 31.8 s, the 95th-percentile collaboration response time was 2.46 s, and the average material retrieval latency was 2.90 s. After optimization, the corresponding values decreased to 12.5 s, 0.91 s and 0.80 s, respectively. Therefore, the 12.5 s rendering result represents a 60.7% reduction relative to the baseline. The response-time and material-access improvements also indicate that gateway routing, Redis state caching and MinIO object storage reduce collaboration congestion and repeated asset transfer. (Table 4)

Table 4. Baseline Comparison and Failure-Test Results

Metric	Baseline system	Proposed system	Result
Average 4K frame rendering time	31.8 s	12.5 s	60.7% shorter
P95 response under 500 creators	2.46 s	0.91 s	63.0% shorter
Material retrieval latency	2.90 s	0.80 s	72.4% shorter
Final unrecovered task error rate	0.62%	0.01%	98.4% lower

5.3 Distributed Node Failure and Error-Rate Analysis

In the distributed node failure test, 10,000 rendering blocks were executed while GPU worker interruption, render-node reboot, network timeout and storage I/O delay were manually injected. Before retry, the observed error rates were 0.31% for GPU-worker interruption, 0.18% for network timeout between render nodes and storage, 0.09% for storage I/O timeout, 0.04% for checksum or frame-merge exception and 0.02% for duplicate scheduling. The total pre-recovery exception rate was 0.64%. After automatic re-queuing, dependency checking and checksum verification, only one block remained unrecovered, giving a final unrecovered error rate of 0.01%. The results show that the fault-tolerant scheduling mechanism effectively converts most transient node and network exceptions into recoverable tasks.

6 Conclusion

This study proposed and verified a cloud computing-driven optimization scheme for efficient animation video production. By defining the workflow from material import to final output, the study clarifies the roles of Spring Boot, Nacos, Spring Cloud Alibaba, distributed rendering, AI assistance and real-time collaboration. The six-layer architecture separates user interaction, gateway access, business services, middleware support, elastic computing and external integration, thereby improving scalability and maintainability. The scheduling model converts frame complexity, urgency, resource load and dependency readiness into concrete dispatching factors. The result analysis shows that, compared with the centralized baseline, the proposed

system reduces average 4K frame rendering time from 31.8 s to 12.5 s, lowers P95 collaborative response time from 2.46 s to 0.91 s, reduces material retrieval latency from 2.90 s to 0.80 s and decreases the final unrecovered error rate to 0.01%. These results demonstrate that the proposed scheme can shorten production cycles, improve material reuse and enhance collaborative reliability.

References

1. Guo Haichun. Research on Intelligent Upgrade of Data Center UPS System Driven by Cloud Computing[J]. Telecom Power Technology, 2026, 43(1):119-121. DOI:10.19399/j.cnki.tpt.2026.01.040.
2. Zhu Jiaqi, Ren Jianxin. The Impact of the Integration of Data Elements and Artificial Intelligence Technology on the Regional Economic Development Gap——Empirical Research Based on 30 Provinces[J]. Science & Technology Progress and Policy, 2025, 42(15):1-10. DOI:10.6049/kjjbydc.D42025030370.
3. Wu Qiong. Research on the Development Path of Red Cultural Industry Empowered by Digital Technology——Taking City J in Jiangxi Province as an Example[J]. Reform & Opening, 2025,(15):42-49. Available at: https://www.cnqkgw.com/index/content/content?con_id=7657&id=115.
4. Rekik, Ghazi,Belkhir, Yosra,Jarraya, Mohamed.Searching to improve learning from complex animated basketball scenes: when decreasing the presentation speed is more efficient than using segmentation[J].Technology, Pedagogy and Education,2021,30(3):393-407. DOI:10.1080/1475939X.2021.1893804.
5. Haripottawekul, Ariyaporn,Epstein, Ethan,Ren, Jolie, et al.Innovative Approaches for Student-Led Creation of Animated and Interactive Videos in an Undergraduate Introductory Chemistry Course[J]. Journal of Chemical Education,2025,102(8):3644-3652. DOI:10.1021/acs.jchemed.5c00375.
6. Sharmin, Nazlee,Houshyar, Shahram,Stevenson, Thomas R., et al.Using PowerPoint and H5P to Create Interactive Animated Instructional Videos[J].The Clinical Teacher,2025, 22(3).DOI:10.1111/tct.70094.
7. Calvert, Clare,Barber, Vicki S.,Appelbe, Duncan, et al.Developing generic clinical trial animated explainer videos in the UK: results of a survey and case study[J].TRIALS, 2025,26(1):25-10.DOI:10.1186/s13063-024-08687-5.