Towards Distributed Collaborative Rendering Service for Immersive Mobile Web

Liang Li, Yakun Huang, Xiuquan Qiao, Yifa Meng, Dingguo Yu, Pei Ren and Schahram Dustdar

***Abstract*—The offloading of rendering to remote servers in resource-constrained devices is a promising strategy for developing mobile web-based immersive applications. However, achieving satisfactory delay experiences in large-scale mobile web-based immersive applications by solely offloading rendering to the cloud or edge servers can be challenging due to congestion in the core network and the lack of concurrent computing ability. To address these issues, we propose a distributed and collaborative rendering service (DCR) framework for mobile web-based immersive applications. The DCR framework constructs a collaborative rendering service architecture that includes cloud, edge, and terminal computing nodes, which can effectively alleviate the computing and transmission pressure on remote computing nodes and adapt to various immersive service environments. To achieve rendering offload between computing nodes, DCR proposes an approach of intermediate structure rendering offload based on JavaScript Object Notation (JSON) data format, which requires fewer computing and network resources. Additionally, the DCR deploys a new 3D model loading engine based on rendering intermediate JSON data communication, which provides dynamic, on-demand, asynchronous, and discrete rendering data services to the mobile web browsers.**

I. Introduction

THE widespread use of lightweight and cross-platform immersive web services, such as web augmented reality (AR) and virtual reality (VR), in mobile web-based immersive applications and metaverse has been documented. To ensure high fidelity and immersive services with real-time experience, it is essential to enable low-latency and low-energy 3D model rendering, as noted in [1]. To address this need, web rendering engines, such as Three.js and Babylon.js, have been developed to provide intensive rendering and are becoming the dominant approach. However, given the resource-constrained nature of the mobile web browser and the inefficient JavaScript computing environment, the challenge lies in providing a real-time experience with large 3D models [2]. Despite the promotion by the World Wide Web Consortium (W3C) standards organization of the WebGL standard, which enables 3D rendering acceleration using native graphics cards, it remains insufficient to render large 3D models and provide real-time interaction.

Table I provides an overview of two approaches that address the challenge of rendering intensive 3D content on resource-constrained mobile web. The first approach is native on-demand rendering, which employs level of detail (LOD) and field of view (FoV) techniques to optimize rendering data and reduce computing resource and network bandwidth demand [4]. While this approach can provide real-time and accurate rendering strategies, it requires efficient cache management and complex viewport prediction, posing challenges for mobile web developers and users. The second approach is offloading rendering tasks to the resource-rich cloud center and providing rendering services in streaming [5]. However, the increasing demand for network and computing resources due to the high concurrency of requests and the complex computing requirements of rendering services poses significant challenges. The limited core network bandwidth hampers the support of computing resources in the cloud for offloading rendering tasks [6]. Although low-delay edge computing services can potentially address this issue, personalized participation and content production requirements in immersive applications for the metaverse have created significant differences among users in terms of rendering computing content. These differences have presented significant challenges for rendering computing offloading and data delivery. Consequently, offloading-based rendering is limited to specific application scenarios, making it challenging to provide services for ubiquitous large-scale immersive applications.

With the development of 5G and Beyond 5G (B5G), mobile edge computing and device-to-device (D2D) communication have become key technologies that promise to enable distributed collaborative rendering of mobile web content. This article begins by introducing the key characteristics of immersive mobile web 3D services and subsequently analyzes the critical challenges involved in achieving real-time, low-energy consumption, and high-fidelity 3D rendering. The article presents a distributed collaborative rendering service mechanism (DCR) based on a Cloud-Edge-devices architecture, which offers several advantages over existing computing offloading approaches. Firstly, DCR requires less computing resources for rendering, resulting in a smaller volume of rendered data and reducing the dependence on network and computing environments for computing offloading. Secondly,

L. Li are with Lab of Future Imaging Technology and Application Laboratory of Zhejiang Province; Key Lab of Film and TV Media Technology of Zhejiang Province; School of Media Engineering, Communication University Of Zhejiang, Hangzhou, Zhejiang 310018, China. E-mail:liliang@cuz.edu.cn.

Y. Huang and X. Qiao (corresponding author) are with State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China. E-mail:{ykhuang, qiaoxq}@bupt.edu.cn.

Y. Meng are with Beijing Key Laboratory of Big Data in Security&Protection Industry, Beijing 100021, China. E-mail:mengyifa@126.com.

D. Yu (corresponding author) are with Key Lab of Film and TV Media Technology of Zhejiang Province; School of Media Engineering, Communication University Of Zhejiang, Hangzhou, Zhejiang 310018, China. E-mail:yudingguo@cuz.edu.cn.

P. Ren is with the AI Innovation Center, Midea Group, Beijing 100015, China. Email: renpei@midea.com.

S. Dustdar is with the Distributed Systems Group, Technische Universität Wien, 1040 Vienna, Austria. E-mail:dustdar@dsg.tuwien.ac.at.

TABLE I: Summary of rendering techniques for the mobile web

Methods		Characteristics	Disadvantages
On-demand rendering	LOD	The pressure of 3D model loading and transmission is relieved by the gradual optimization of the triangular network structure [3].	Not suitable for the immersive mobile web, which relies on efficient local cache resource management and under-supported by computing resources.
	FOV	Adopts progressive transmission and on-demand loading based on the human eye viewport to achieve efficient transmission and rendering of large scenes. [4].	Due to the necessity of providing a real-time and accurate on-demand loading strategy based on the dynamic context, rendering techniques are a source of difficulties and uncertainties for an interactive user experience.
Rendering offloading	Cloud	The ECs of rich computing resources and scalability provide rendering services in the form of a stream. [5]	The ability to provide adaptive services for diversified environments is compromised because core network congestion will affect the implementation of the offloading of the computing.
	Edge	This technique provides reliable rendering offloading services for resource-constrained high-bandwidth and low-latency mobile networks through edge cloud computing resources. [6].	It is costly to deploy computing resources under limited scalability for the high concurrency requirements of mobile web and advanced GPU computing.
	Hybrid	This technique uses cloud-edge collaboration to effectively supplement the limited and expensive computing resources of the edge cloud, which significantly improves the concurrency of mobile user access. [7].	Over-reliance on core computing nodes and networks will cause service interruption when core computing nodes and organizations fail.

DCR proposes a novel approach to loading discrete rendered data with low latency, thereby reducing the dependence on stable network and computing resources. Finally, DCR integrates D2D collaborative computing with the cloud-edge-devices collaborative rendering computing approach, making it adaptable to various service environments. To validate the feasibility of DCR, we have implemented a preliminary prototype that renders complex 3D model services for the mobile web, improving efficiency through collaborative offloading. We evaluate the performance of the prototype and discuss future research directions regarding immersive application rendering offloading services.

II. Characteristics and challenges

This section delves into the features of rendering offloading, which are illustrated in Figure 1. Additionally, it examines the obstacles encountered by current methods.

A. Characteristics

1. Limited computing capability of mobile web browsers. In order to maintain consistency of page data, the interpretation threads of JavaScript and the GUI threads in mobile web browsers are mutually exclusive. In accordance with this mechanism, any thread tasks with a large pre-sequence delay consumption in the queue can result in delays in computing response of post-sequence threads, which may even lead to loading blocks. This issue is particularly problematic for delay-sensitive and high computing resource-consuming applications, such as mobile VR/AR, as loading blocks can severely impair the user's interactive experience. To improve the intensive computing performance, WebAssembly can be used to quickly load and instantiate precompiled code, which is much more efficient than JavaScript interpretation computing. However, due to the mobile browser's security mechanisms and underlying architecture, there may be insufficient computing resources available [8].

2. High demands for high-quality media presentation and interaction. Immersive applications impose a higher demand for rendering quality in 3D scenes due to their close interaction and wider visual field [9]. To overcome the impact of low resolution and fast motion on the user experience caused by monocular domain and close-up interaction, ultra-high resolution and frame rate are required [10]. However, the use of image data with ultra-high resolution entails more complex rendering computation and greater communication bandwidth requirements. Thus, achieving ultra-low latency and real-time rendering is dependent on rich computing and network resources, and is considered a crucial factor for rendering offloading in immersive experience services.

3. Continuous service delivery requires a stable network environment. It is imperative for mobile web immersive applications to respond promptly to users' motion and interaction requests with low Motion-to-Photons Latency (MTP, less than 20ms). With frequent user interaction, immersive applications that rely on continuous image rendering have more stringent requirements for continuous data provision than traditional applications. This continuous data supply is highly reliant on stable and uninterrupted high frame rate data. The advent of 5G network services provides a chance to better fulfill the requirements of immersive services in terms of computing throughput and transmission delay [11]. However, mobile web browsers face limitations in storing the presented data in local storage hardware with larger storage capacity due to security restrictions and lack of underlying resources. Additionally, advance rendering based on server cache cannot withstand the pressure of massive data capacity in multi-user scenarios due to differences in content delivery of multi-user channels. Therefore, mobile devices require a continuous and stable

network esource that supports low-latency continuous content to ensure uninterrupted and reliable access to immersive applications.

4. The dynamics of services eager an intelligent distributed resource orchestration. In light of their convenience and universality, web services exhibit a marked degree of randomness and spatial aggregation, particularly in the context of mobile AR. This randomness engenders considerable uncertainty with regard to the distribution of network and computing resources available to the mobile web [10]. Notably, the rendering of these services necessitates the use of computing nodes external to the application device, with data exchanged via a network (e.g., a D2D communication channel). This process is heavily reliant on the quality of the external network and computing environment. Given the inherent variability of network and computing resources, the rendering of immersive mobile web services via computing nodes is inherently complex.

B. Challenges

Rendering offloading based on computing nodes imposes higher requirements on the network infrastructure and service environments, such as a flexible rendering node structure, a lightweight rendering environment, and an efficient multi-source rendering mechanism.

1. Lack of sufficient computing and network resources support in complex environments.

To meet the demands of high-quality media presentation and multi-user channel interaction, rendering offloading requires significant computing resources and ample delivery bandwidth. Utilizing a central server allows for low latency through the use of advanced frameworks, such as machine learning and deep neural networks, which can provide superior performance for a limited number of users. However, when there is a high degree of request concurrency among users, the efficiency of data delivery can be compromised. Decentralized rendering based on peer-to-peer computing nodes, such as mobile devices, can address this issue by providing ultra-low latency rendering and computing offloading capabilities, dependent upon sufficient computing nodes. Therefore, effectively deploying rendering computing offload services in unfamiliar service environments presents an important challenge for immersive applications.

2. Low latency, high quality interaction rendering delivery. In current approaches to offloading rendering, computing nodes transmit immersive application renderings as video streams. Mobile web browsers must either load these streams asynchronously or store them in cache for rendering. However, hardware limitations may hinder mobile web browsers from efficiently indexing and loading multimedia data that has been remotely delivered and stored in cache. In high-frequency interaction scenarios, mobile web browsers struggle to load video streams with low latency, presenting significant challenges to the user's experience. Furthermore, the convergence of computing and rendering data from multiple synchronous sources will pose additional obstacles to the low-latency computing capabilities of mobile web browsers.

3. Reuse of continuous rendered data from multiple sources. In mobile web-based immersive applications, high-quality media presentation and interactions are necessary, making multi-source rendering the leading solution. Multi-source offloading rendering services requires the optimization of data reuse between multiple computing nodes to improve the utilization of rendered data and reduce network delivery latency. Moreover, redundancy check mechanisms of multiple sources can address packet loss and frame skipping caused by network fluctuations. However, existing remote rendering approaches in continuous video stream structures may create challenges in multiplexing and aggregating multi-source rendering computations. Firstly, stream-based data reuse relies on efficient cache computation and adequate cache resources. Secondly, continuous content data structures require precise synchronization of the rendered computing data from multiple sources.

4. Flexible network and computing resource structure. The provision of an immersive service environment is a

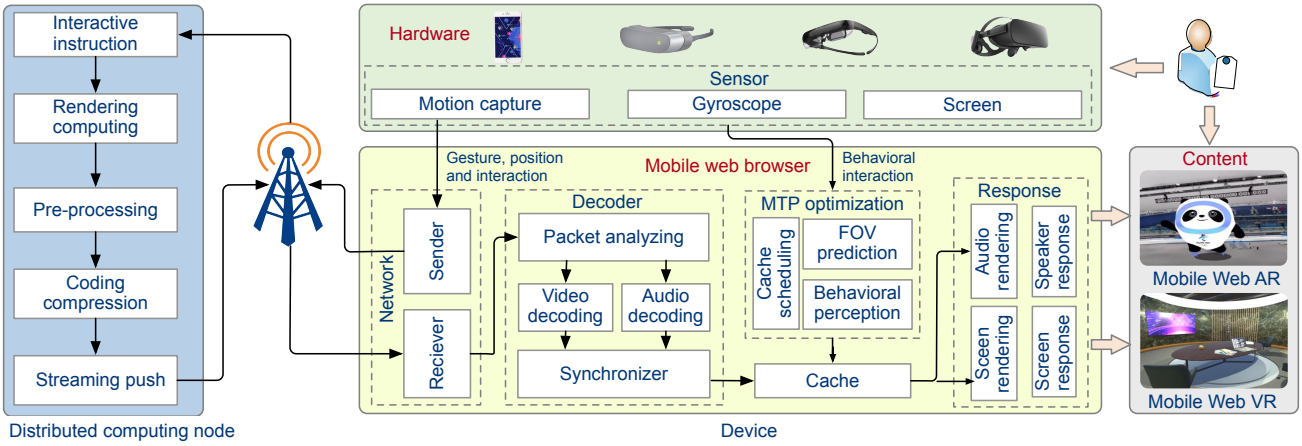


Fig. 1. Network node-based rendering system

complex and highly variable process that results in dynamic changes in the distribution of network computing nodes and access to rendered computing data. However, the existing central computing nodes, consisting of cloud servers and edge servers, are affected by the bandwidth of the core network and the resources of the core computing nodes. The effective transfer of computing tasks from mobile web browsers to computing nodes is contingent upon the high performance and reliability of the network and mobile edge servers. Unfortunately, the current distribution of computing nodes impedes the decentralization of rendering, thereby limiting the effectiveness of existing approaches to offloading rendering to computing nodes. This constraint further exacerbates the challenges in achieving widespread adoption of such approaches for the changing service environments.

III. Methods of the DCR framework

To meet these challenges, developing a collaborative rendering service framework composed of multiple computing nodes is wise. We will indicate how the proposed DCR framework addresses these challenges.

A. Distributed collaborative rendering solution

The DCR framework offers rendering services for mobile web-based immersive applications by means of offloading the rendering process. The proposed solution is distinct from current practices in several ways. First, we utilize various computing nodes, including edge servers and mobile devices, within the network to offload rendering and provide these services for mobile web-based immersive applications. To address the challenge of creating an efficient rendering environment, we use a lightweight rendering environment based on JSON exchange data. Second, the centralized and decentralized computing nodes cooperate in providing computing services based on the offloaded multi-node rendering service. This solution can switch the service between different computing nodes based on the change in the service environment, providing a flexible network resource structure. Third, we adopt a JSON data format that is customized to exchange rendered data between computing nodes and application devices. With only minimal computing resources on the web side, this data format can reduce the burden of rendering and push images to the multimedia elements of the web browser. Finally, in multi-source collaborative rendering, we utilize a granularity computing task scheduling and rendered data reused approach to minimize the redundancy of data and the dependence on specific computing nodes.

B. Construction of collaborative rendering offloading

The DCR framework is primarily implemented for the purpose of building distributed computing nodes. In a distributed rendering architecture, effective management of computing nodes, task scheduling, and data exchange are crucial for addressing the pressure on centralized rendering. Consequently, this work will elaborate on the offloading of distributed rendering.

To begin with, computing nodes are deployed on the mobile edge server and the mobile devices surrounding the application device. The DCR approach establishes a rendering offloading service using a Docker container on the mobile edge server. This rendering offloading service functions as a microservice on the mobile devices. The main advantage of this approach is that the mobile browser requires minimal computing, making it suitable for various application device hardware scenarios.

Secondly, in order to implement rendering offloading services within the D2D communication on computing nodes around mobile browsers, a central controller was deployed on the edge server. To integrate new computing nodes into the cooperative rendering network, we adopted a registration mechanism in the central controller, which registered the rendering computing status of the computing nodes to the DCR.We introduced a reference approach [12] to achieve awareness of the multi-user multiple input and multiple output environment and cooperative rendering scheduling. Additionally, we borrowed algorithms from literature [13] and utilized deep neural networks and resource scheduling methods to appropriately schedule collaborative rendering computing services to achieve collaborative optimization of latency and energy. However, the edge-based registration mechanism requires a central controller to optimize the geographically distributed computing nodes, and may not scale well for larger distributed rendering computing due to excessive overhead and delays caused by status reports of the computing nodes. In such cases, optimal scheduling results can be achieved through the coordination of cloud and edge scheduling mechanisms by relying on sufficient computing resources of the cloud server. Finally, computing node discovery and task allocation in the D2D communication can be realized.

Thirdly, DCR employs a dynamic rendering offloading scheduling strategy on the edge server, which determines the current rendering approach and strategy. This dynamic scheduling enables the framework to adapt to diverse network and application devices' computing resource environments by adjusting the rendering and network connection approaches accordingly. The mobile edge server is responsible for determining a context-aware rendering offloading strategy and the appropriate data accessing approach based on the prevailing context. To execute this policy, the following steps are taken:

- DCR employs a dynamic deployment strategy for granularity rendering tasks based on the computing capability of each computing node. Furthermore, we establish a redundancy reused mechanism for sharing computing resources and rendered data, taking into account the distribution of computing nodes with services.
- In cases where the remote server's computing resources or network access is obstructed, DCR utilizes an end-to-end approach to offload rendering, with data shared within the D2D communication. Conversely, when the server's resources and network access are available, the edge cloud and mobile devices collaborate to offload rendering and share data through both the access network

and D2D communication.

In DCR architecture, computing nodes situated on edge servers or mobile devices fulfill distinct roles. When network resources are scarce, a higher degree of latency may be observed in DCR computing nodes deployed on mobile devices. Conversely, when network resources are abundant but available mobile devices for computing offloading are limited, mobile edge servers exhibit superior performance. In such cases, it is advantageous to deploy on mobile devices. To optimize performance, computing nodes can be strategically placed on both mobile edge servers and mobile devices in proximity to the application device, thereby establishing a redundant mechanism for the computing nodes. This configuration allows rendering computing offloading scheduling to swiftly adapt to user distribution in the current service environment, ensuring low-latency rendering computing services. Simultaneously, in scenarios where network resources are limited and few mobile devices surround the application device, the browser-based rendering computing approach can promptly deliver an immersive application rendering response. Consequently, by implementing a dynamic computing offloading method that encompasses browser-based rendering computing techniques, DCR computing nodes situated on edge servers, and DCR computing nodes deployed on mobile devices, the initial rendering computing delay of the DCR can be rendered less susceptible to network bandwidth fluctuations.

Finally, the offloading method presented herein utilizes a JSON-based data format for exchanging information between the mobile web immersive application and computing nodes. This approach differs from traditional model loading techniques as the computing nodes in DCR directly load the model file as a model object. Due to its smaller execution environment and direct loading capabilities, this method is readily deployable within the microservices of mobile devices. Moreover, the JSON-based data format offers several advantages, including reduced bandwidth consumption and increased loading efficiency, in comparison to the multimedia stream data formats employed by existing techniques. To facilitate seamless and uninterrupted data service between the multi-computing nodes and the application device, DCR incorporates asynchronous communication and on-demand loading. Further details on this approach can be found in Section III-C.

C. Low-MTP loading by JSON-based data

By adopting an active and continuous data service approach, browsers can efficiently and reliably access rendered data, facilitating low latency and low energy consumption in rendering services. Computing nodes converts JSON data to string data and sends it back to mobile web-based immersive applications via access and D2D communications. The rendered data is parsed by the mobile web browser and added directly to the scene, resulting in low latency. In DCR, interactive JSON data is defined as a class with a vertex object list element named 'vertices' and a face list element named 'faces'. The 'vertices' and 'faces' lists consist of classes that respectively describe the normal vectors and faces of triangular meshes. Each element of the 'vertices' list contains the x-, y-, and z-coordinates of a point or face's normal vector. An element of the 'faces' list contains the point index of the triangular mesh, a collection of the face's vertices, color description, and material index. DCR advance renders and bakes the material data on the mobile edge server or shares rendered JSON data between computing nodes to reduce browser computing resource consumption. On-demand loading optimizes the textures data of the 3D model as attributes when the browser loads. When the mobile browser requests texture data, it sends a request to the edge server. The edge server responds with device status data under a registration mechanism, which enables the DCR to establish communication links between computing devices. The mobile browser can then access the cached data stored in the cache of other computing nodes or the edge server. The browser performs other rendering computations, such as lights. The key advantage of this approach is its minimal computing demand on the mobile web browser, making it applicable to various application device hardware scenarios.

The rendering and loading of 3D models in mobile web-based immersive applications are affected by the web browser's indirect computing resource scheduling and JavaScript interpretation execution mechanism. Therefore, it is necessary to adjust the approach to loading the 3D model to the browser while optimizing the rendering. To achieve this, we optimize the model data retrieval process from a simple interface to enable asynchronous caching and on-demand loading through the web browser's multi-threading and caching. Asynchronous browser threads and database technology are used to cache and synchronize the model data on-demand on the mobile web browser. The use of asynchronous threads enables browsers to open up new loading threads besides the main thread, allowing JavaScript to run in the background. This approach ensures the loading thread's independence from other scripts and does not affect the web's performance, enabling the quick loading of required resources and reducing waiting time. Consequently, multi-threaded parallel loading is an effective way to solve the problem of excessive resource consumption and long loading delays.

In addition, the browser database, as a local database, can be created and used through JavaScript, providing convenient operations such as finding an interface and establishing an index suitable for this scenario. The combination of the browser database and browser cache provides fast rendering data index and services. With this approach, users who enter the same page can directly read the corresponding model data through the browser database among computing nodes, avoiding the reliance on the network to request the corresponding model resources. This method improves the efficiency of any subsequent loading.

D. Collaborative and stable offloading for rendering service

To reduce the computing delay caused by repetitive tasks, we propose a granularity computing task scheduling and rendered data redundancy reused approach. Specifically, we utilize DCR to preprocess or process the model in real-

time on the central server based on the physical structure of 3D objects. This allows us to divide the JD model into several model blocks, which can be encapsulated along with business-related information (such as angle and interaction) into fine-grained rendering task descriptors. By leveraging user demand prediction, we can then map multiple fine-grained rendering sub-tasks to establish the service logic of mobile web immersive application. Assuming that the immersive application's specific service environment contains n independent 3D scenes denoted by $S = [S_0, S_1, ..., S_{n-1}]$, we can divide the rendering of each scene into m fine-grained subtasks represented by $T^f = [T_0^f, T_1^f, ..., T_{m-1}^f]$. Additionally, we can combine the model data blocks corresponding to each fine-grained subtask into $[M_0, M_1, ..., M_{m-1}]$. The rendering task for each scene can be described as $T^c = [t_0^c, T_1^c, ..., T_{n-1}^c]$, where $T_i^c = [T_{f(i)}^f]$ and $T_{f(j)}^f \in T^f$ with $f(i)$ representing the mapping function between the ith scene and T^f. By utilizing this approach, we can effectively reduce the computing delay caused by repetitive tasks and optimize the rendering process of the immersive application.

The rendering tasks' granularity organization in the remote cloud server is represented by a matrix $A = [ST_{i,j}]$, where i and j are indices that range from 0 to $n-1$ and 0 to $m-1$, respectively. The symbol of subtask T_j^f is expressed by $ST_{i,j}$ if $T_j^f \in T_i^c$, otherwise, it is represented as $'null'$.

When the service in the regional network is triggered by the application device, the fine-grained computing task and the matrix A are migrated to the edge cloud for collaborative control. The edge cloud assigns computing tasks to the collaborative computing nodes based on the current scene's time sequence and logical relations. Once a rendering subtask is completed by a computing node in the network, it sends the rendered data to the application device via short-range communication links, such as D2D. Additionally, the computing node requests the edge cloud to update the computing task state matrix A, which describes the storage index of the current subtask rendered data (including computing node information). During the allocation of subtask T_i^f, the edge cloud traverses the state of the jth column subtask in matrix A. If T_i^f has already been completed (i.e., the $ST_{i,j}$ state is the storage index of subtask computing output data), the edge cloud notifies the application device to communicate directly with the corresponding computing node. To render a 3D scene, DCR divides the large JD file into model file blocks and renders them on multiple computing nodes. This enables multiple mobile devices to collaborate on rendering a large JD model file simultaneously, resulting in faster rendering of the complete model. The server can merge multiple fragments of a 3D model into a complete model and return it to the browser. Alternatively, the browser can initiate multiple requests to push the model fragments individually to the scene, and the server can request other model fragments in parallel to avoid the idle state of the mobile browser while waiting for network IO. Finally, the server executes the rendering process using an approach based on docker or microservice.

IV. Performance Evaluation

This section outlines our experimental methodology that serves three distinct purposes. Firstly, we aim to evaluate the efficacy of the offloaded service environment that was constructed using the DCR to illustrate its advantages. Secondly, we aim to verify the latency and quality of the interaction in the rendering when the environment undergoes changes. Lastly, we seek to verify the benefits of the rendering reuse approach employed in the DCR.

A. Effectiveness of the collaborative rendering offloading

We aim to assess the effectiveness of different approaches, including Browser (rendering in the browser), DCR-Edge (employing DCR that computing nodes deployed on the edge server), and DCR-Devices (employing DCR that computing nodes deployed on mobile devices), by measuring the response delays under different 3D model file sizes. To achieve this, we utilized ten dockers on mobile devices with a 1.6 GHz and 4 GB RAM single-core processor and a mobile edge server with six dockers as computing nodes. This setup was chosen because mobile devices often serve multiple purposes, including acting as computing nodes, and therefore have limited computing resources. By utilizing low computing resources, user participation can be enhanced without negatively affecting other services, especially in environments where resources are limited. The browser rendering was performed using Chrome simulation with an 8-core CPU and 8GB RAM, and the bandwidth of the D2D channel was set to 300 Mbps. The response delays for each approach are presented in Table II.

TABLE II: The effectiveness in initial response delay (ms)

Model volume (MB)		1.505	4.511	10.698	22.227	32.623	53.416
Browser	τ_{tran}	80	137	447	685	903	1322
	τ_{load}	430	1068	1701	2168	2892	4182
	τ_{sum}	510	1205	2148	2853	3795	5504
DCR-Edge	τ_{com}	3	4	7	8	10	13
	τ_{tran}	61	188	404	930	1320	2290
	τ_{load}	21	45	95	190	252	467
	τ_{sum}	85	237	506	1128	1582	2770
DCR-Devices	τ_{com}	4	4	7	12	20	35
	τ_{tran}	17	54	126	266	378	656
	τ_{load}	21	45	95	190	252	467
	τ_{sum}	42	103	228	468	650	1158

Here, τ_{tran}, τ_{load}, and τ_{com} denote the transmission delay, browser loading and rendering delay, and rendering computing offloading delay, respectively. The total delay, denoted by τ_{sum}, is described as $\tau_{sum} = \tau_{load} + \tau_{tran}$ or $\tau_{sum} = \tau_{load} + \tau_{tran} + \tau_{com}$.

The results in Table II indicate that DCR is generally more efficient in achieving an initial response than other approaches. However, the optimization efficiency of DCR on the mobile edge server is not significant with the increase in the model file capacity. The advance rendering of the DCR framework results in a 4x capacity increase compared to the original 3D

model, which may cause substantial delays when transmitted between computing nodes. To overcome this limitation, DCR can deploy computing nodes around the application device to avoid access network blocking, as shown in the DCR-Devices section of Table II.

Next, we verify the effectiveness of the DCR architecture in immersive services under a 3D model volume of 10.698 MB with different application device computing capabilities. We use the Chrome tool to simulate application devices with different computing abilities. The delays with different browser computing capability for each approach are presented in Table III. As a result, DCR has lower requirements for browser computing capability than existing methods, making it more suitable for pervasive application devices. Consequently, an efficient and low-latency rendering environment can be easily constructed through the collaborative rendering offloading service of edge servers and mobile devices.

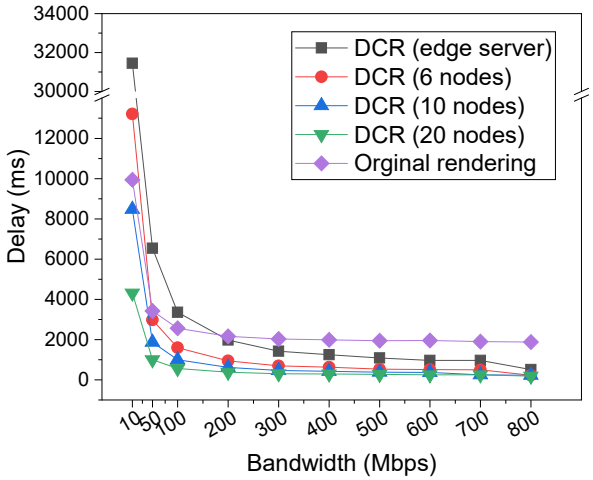
TABLE III: The delays with different browser computing (ms)

Browser computing		Original	$\frac{1}{4}$ Original	$\frac{1}{6}$ Original
Browser	τ_{load}	1701	6216	9180
	τ_{sum}	1886	6401	9365
DCR-Edge	τ_{load}	95	390	612
	τ_{sum}	506	801	1023
DCR-Devices	τ_{load}	95	390	612
	τ_{sum}	306	601	823

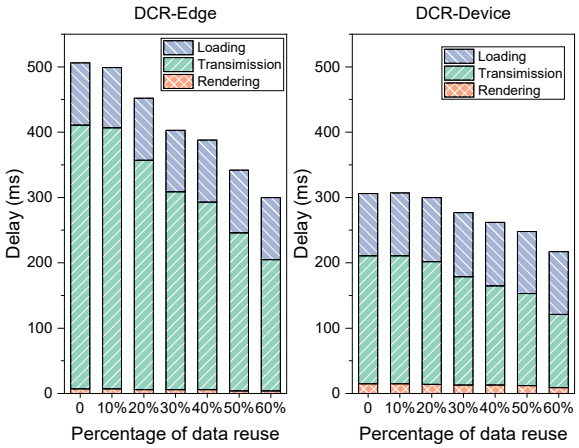
B. Latency and interaction quality in different services

This subsection also employs the Chrome tool to simulate various connection bandwidths in order to examine the effectiveness of DCR in different network environments. Figure 2a illustrates that DCR deployed on edge server computing nodes yields a worse rendering delay than browser-based rendering when the bandwidth is low, i.e., less than 20 Mbps. This outcome can be attributed to the advance rendering process that generates greater data volume, as discussed previously. To address this issue, we conducted experiments with different numbers of mobile devices as computing nodes to evaluate the initialization response delay. Figure 2a shows that deployment on mobile devices results in lower latency than both browser-based rendering and computing nodes deployed on mobile edge servers. DCR sends the rendered data to the application device through the D2D channel instead of the access network, thus minimizing the impact of network bandwidth. Furthermore, as more computing nodes are deployed, the optimization of the DCR in initialization response delay becomes more pronounced.

The MTP is a crucial index for assessing the quality of user experience. In this study, we utilized the stats.js tool (from https://github.com/mrdoob/stats.js) to measure the MTP, which is typically expressed by the interactive frame rate, of two different approaches: the DCR framework deployed on edge and traditional browser-based approaches. The results of our experiments are shown in Table IV, where f_{min} represents



(a) DCR's delay in various network bandwidth



(b) DCR-Edge with reused (c) DCR-Devices with reused

Fig. 2. In-depth analysis of DCR in different environments.

the minimum interactive frame rate, f_{max} represents the interactive maximum frame rate, and f_{sta} represents the stable interactive frame rate. It indicates that DCR-Edge and browser rendering approach achieve similar MTP scores. This outcome can be attributed to the fact that when users request interactive content, the animation data is rendered directly in the mobile device cache through WebGL. In contrast, third-party servers (such as server-based rendering approaches) provide the data service. The advance rendered data is directly inserted into the current scene using the 'add' approach, thereby avoiding the extra delay that is caused by the data stream index and cache. Overall, these findings demonstrate that DCR provides similar interactive frame rates to browser-based rendering approaches, as a result of its ability to directly render data in the mobile device cache, which bypasses the need for a third-party server's data service.

C. Effectiveness of the reuse from multiple sources

This subsection examines the efficacy of rendered data reused within the DCR framework. Specifically, the model processing module partitions the original model using a granularity segmentation approach. This approach involves the reuse

TABLE IV: The interactive frame rate (FPS)

Model volume (MB)		4.511	10.698	22.227	32.623	53.416
Browser	f_{min}	28	15	10	20	19
	f_{max}	60	60	45	37	35
	f_{sta}	60	60	43	33	31
DCR-Edge	f_{min}	27	21	10	20	19
	f_{max}	60	60	45	37	35
	f_{sta}	60	60	42	33	31

of repetitive model segmentation units within a given scene, which are then superimposed based on semantic information. To evaluate the effectiveness of this approach, we varied the proportion of reusable scenes within the overall scene (i.e., model file ratio) and measured the initial rendering delay, as illustrated in Figure 2b and Figure 2c. Figure 2b is the performance of DCR's edge with different data reuse, and Figure 2c is the performance of DCR's mobile devices with different data reuse. The results indicate that the efficiency of DCR optimization increases as the reuse ratio increases. In real service environments, as the increase in user scale, there is greater potential for reusing rendered data across different mobile browsers. Therefore, DCR can offer a superior user experience for mobile network immersive applications compared to existing methods when involving dense users. These findings highlight the suitability of DCR for showcasing large-scale and complex service environments.

D. Discussion

Our approach raises several issues that we need to consider in order to better understand the contribution of this paper. Firstly, our experiment did not consider the server-based rendering computing offloading method. This is because in an actual service environment, the rendering computing offloading service is typically deployed on a remote cloud server with abundant computing resources, and may even require a specialized software and hardware environment for rendering computing services. However, this would require service providers or operators to spend additional capital and energy costs to support rendering computing offloading, which is currently not cost-effective. Additionally, while the server-based rendering computing method offers the advantage of low computing offloading delay for the remote server and low loading delay for the application device, the rendered multimedia data from the server-based rendering computing is much larger than that of the original 3D model and DCR framework method. When network resources are limited, this can cause unacceptable delays. Furthermore, the efficiency of interactive response is also a major factor that we did not consider when examining the server-based rendering computing offloading method. Thus, considering both service cost and user experience, DCR method is a practical approach to rendering computing offloading services.

Secondly, it is beneficial to integrate it with existing studies and technologies to enhance the performance of DCR. Although these technologies are not novel, it is essential to discuss their integration to bolster the credibility of DCR. Optimization technologies in web browsers, such as browser database cache and asynchronous thread communication, can significantly improve the latency of DCR, which is discussed in Section III-C. Moreover, DCR's discrete data loading approach can proficiently organize cached data to provide users with seamless and high-quality interactive experiences. Browser-based asynchronous communication and on-demand loading can reduce the amount of data rendered and loaded by the browser, further contributing to smoother user experiences. When combined with DCR's continuous data service mechanism and discrete loading method, this approach can produce a more polished interaction experience, surpassing what existing rendering computing offloading methods can achieve. However, these techniques are not reflected, so it is challenging to evaluate their impact accurately in comparison to other optimization methods.

Thirdly, we would like to clarify that although some works are highly relevant to this paper, they may be relatively complex. However, various scholars have already conducted extensive research on these works, resulting in significant findings. For instance, Yang Li et al. have proposed a hybrid model of non-orthogonal multiple access (NOMA) and frequency division multiple access to facilitate multi-user dual computation offloading [14], which provides a comprehensive reference model for real-time scheduling of distributed collaborative rendering. Additionally, optimizing time delay and collaborative energy offload computing is one of the primary research areas in computing offload optimization. Literature such as (e.g., [15]) has proposed more advanced mathematical models that establish a balance between delay and energy between the device and multi-access computing node. Moreover, while we have adopted a relatively simple and rough method, i.e., segmentation based on physical structure in the reuse of rendered data, more advanced and novel research methods exist in relevant studies, such as fine-grained segmentation methods for 3D models and business demand prediction. In particular, the feature extraction method using the depth neural network for fine-grained segmentation is more scientific and effective, which can better ensure the smoothness of the 3D model. These works are highly significant to the DCR framework, yet they are not our innovations, and we have not introduced or verified them in detail in this work.

V. Conclusions and Future Work

In this work, we propose the DCR, which provides a rendering service for large-scale mobile web-based immersive applications. We have made several key findings. Firstly, we have demonstrated that increasing the number of mobile devices deployed as DCR computing nodes can enhance the efficiency of optimizing the response delay. However, we must acknowledge the issue of user privacy protection and security that arises when offloading computing tasks between untrusted mobile devices. To address this concern, we propose researching solutions such as blockchain technology or offloading between personal mobile devices in future work. Secondly, we have

found that DCR computing nodes deployed on edge servers or mobile devices play different roles. When the network resources are insufficient, mobile devices with DCR computing nodes will provide a shorter delay. Conversely, when the network resources are sufficient, and the mobile devices available for offloading computing are limited, mobile edge servers will perform better. We recommend deploying computing nodes on both mobile edge servers and mobile devices around the application device and establishing a redundancy mechanism for the computing nodes. This way, the offloaded rendering can be scheduled quickly according to the user distribution in the current service scenario to provide low latency. In scenarios where network resources are insufficient and few mobile devices are available, the browser-based rendering approach can provide a quick rendering response. To make DCR less affected by the properties of the service environment, we propose a dynamic computing offloading approach in future work. This approach will include browser-based rendering, DCR edge-based rendering, and DCR mobile devices-based rendering. By incorporating these rendering methods, DCR can provide a more flexible and efficient rendering service for large-scale mobile web-based immersive applications.

ACKNOWLEDGMENT

This work is supported in part by the National Natural Science Foundation of China under Grant 62202065, in part by the Zhejiang provincial natural science foundation under Grant LTGG23F020001, in part by the Key Lab of Film and TV Media Technology of Zhejiang Province under Grant 2020E10015, in part by the Project funded by China Postdoctoral Science Foundation 2022TQ0047 and 2022M710465.

REFERENCES

[1] C. Liu, W. T. Ooi, J. Jia, and L. Zhao, "Cloud baking: Collaborative scene illumination for dynamic Web3D scenes," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 14, no. 3s, pp. 1–20, 2018.

[2] X. Qiao, P. Ren, S. Dustdar, L. Liu, H. Ma, and J. Chen, "Web AR: A promising future for mobile augmented reality—state of the art, challenges, and insights," *Proceedings of the IEEE*, vol. 107, no. 4, pp. 651–666, 2019.

[3] Z. Lv, X. Li, H. Lv, and W. Xiu, "BIM big data storage in WebVRGIS," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 4, pp. 2566–2573, 2019.

[4] Y. Yang, L. Feng, X. Que, F. Zhou, and W. Li, "Energy- and quality-aware task offloading for WebVR service in terminal-aided mobile edge network," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 8, pp. 8825–8838, 2022.

[5] T. Kämäräinen, M. Siekkinen, J. Eerikäinen, and A. Ylä-Jääski, "CloudVR: Cloud accelerated interactive mobile virtual reality," in *Proceedings of the 26th ACM international conference on Multimedia*, 2018, pp. 1181–1189.

[6] L. Lin, X. Liao, H. Jin, and P. Li, "Computation offloading toward edge computing," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1584–1607, 2019.

[7] Y. Huang, X. Qiao, S. Dustdar, J. Zhang, and J. Li, "Toward decentralized and collaborative deep learning inference for intelligent iot devices," *IEEE Network*, vol. 36, no. 1, pp. 59–68, 2022.

[8] D. Lehmann, J. Kinder, and M. Pradel, "Everything Old is New Again: Binary Security of WebAssembly," in *29th USENIX Security Symposium (USENIX Security 20)*, 2020, pp. 217–234.

[9] X. Hou, Y. Lu, and S. Dey, "Wireless VR/AR with edge/cloud computing," in *2017 26th International Conference on Computer Communication and Networks (ICCCN)*. IEEE, 2017, pp. 1–8.

[10] Y. Siriwardhana, P. Porambage, M. Liyanage, and M. Ylianttila, "A survey on mobile augmented reality with 5G mobile edge computing: architectures, applications, and technical aspects," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 2, pp. 1160–1192, 2021.

[11] J. Du, F. R. Yu, G. Lu, J. Wang, J. Jiang, and X. Chu, "MEC-assisted immersive VR video streaming over terahertz wireless networks: A deep reinforcement learning approach," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9517–9529, 2020.

[12] C. Ding, J.-B. Wang, H. Zhang, M. Lin, and J. Wang, "Joint mu-mimo precoding and resource allocation for mobile-edge computing," *IEEE Transactions on Wireless Communications*, vol. 20, no. 3, pp. 1639–1654, 2021.

[13] F. Jiang, K. Wang, L. Dong, C. Pan, W. Xu, and K. Yang, "Deep-learning-based joint resource scheduling algorithms for hybrid mec networks," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6252–6265, 2020.

[14] Y. Li, Y. Wu, M. Dai, B. Lin, W. Jia, and X. Shen, "Hybrid noma-fdma assisted dual computation offloading: A latency minimization approach," *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 5, pp. 3345–3360, 2022.

[15] D. Song, T.-B. Li, W.-H. Li, W.-Z. Nie, W. Liu, and A.-A. Liu, "Universal cross-domain 3d model retrieval," *IEEE Transactions on Multimedia*, vol. 23, pp. 2721–2731, 2021.

Liang Li is currently a Professor at School of Media Engineering, Communication University Of Zhejiang, Lab of Future Imaging Technology and Application Laboratory of Zhejiang Province, and Key Lab of Film and TV Media Technology of Zhejiang Province, Hangzhou, China. His research interests lie in augmented reality, virtual reality, and services computing.

Yakun Huang is currently a Postdoctoral Researcher at the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China. His current research interests include video streaming, mobile computing, and augmented reality.

Xiuquan Qiao is currently a Full Professor with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China. His current research interests include the future Internet, services computing, computer vision, distributed deep learning, augmented reality, virtual reality, and 5G networks.

Yifa Meng is currently a senior engineer with Beijing Key Laboratory of Big Data in Security&Protection Industry, Beijing, China. His current research interests include system architecture design, artificial intelligence.

Dingguo Yu was born in 1976 and is the director of the Key Lab of Film and TV Media Technology of Zhejiang Province, China. He received his M.S. and Ph.D. degrees in computer application technology from Tongji University, China in 2005 and 2011, respectively. His research focuses on media fusion technology, big data and artificial intelligence for media, and so on.

Pei Ren received his Ph.D. degree in Computer Science from State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, China. He is currently working with the Midea Group. His current research interests include machine learning, augmented reality, edge computing, and 5G networks.

Schahram Dustdar (Fellow, IEEE) is a Full Professor of Computer Science and is heading the Distributed Systems Research Division at the TU Wien. He is an ACM Distinguished Scientist, ACM Distinguished Speaker, IEEE Fellow, and Member of Academia Europaea.