



PDF Download  
3757374.3771507.pdf  
23 February 2026  
Total Citations: 0  
Total Downloads: 203

Latest updates: <https://dl.acm.org/doi/10.1145/3757374.3771507>

Published: 15 December 2025

[Citation in BibTeX format](#)

SA Posters '25: SIGGRAPH Asia 2025  
Posters

December 15 - 18, 2025  
Hong Kong, Hong Kong

Conference Sponsors:  
SIGGRAPH

POSTER

## A Hybrid XR Framework for Collaborative Scientific Visualization: Rendering-as-a-Service with ParaView and Real-Time Engines

**JINSUNG CHOI**, Electronics and Telecommunications Research Institute, Daejeon, South Korea

**YONGWAN KIM**, Electronics and Telecommunications Research Institute, Daejeon, South Korea

**SURAN PARK**, Electronics and Telecommunications Research Institute, Daejeon, South Korea

**KI-HONG KIM**, Electronics and Telecommunications Research Institute, Daejeon, South Korea

Open Access Support provided by:

Electronics and Telecommunications Research Institute

# A Hybrid XR Framework for Collaborative Scientific Visualization: Rendering-as-a-Service with ParaView and Real-Time Engines

Jin Sung Choi  
Electronics and  
Telecommunications  
Research Institute (ETRI)  
Deajeon, Republic of Korea  
jin1025@etri.re.kr

Yongwan Kim  
Electronics and  
Telecommunications  
Research Institute (ETRI)  
Deajeon, Republic of Korea  
ywkim@etri.re.kr

Suran Park  
Electronics and  
Telecommunications  
Research Institute (ETRI)  
Deajeon, Republic of Korea  
suepark@etri.re.kr

Ki-Hong Kim  
Electronics and  
Telecommunications  
Research Institute (ETRI)  
Deajeon, Republic of Korea  
kimgh@etri.re.kr

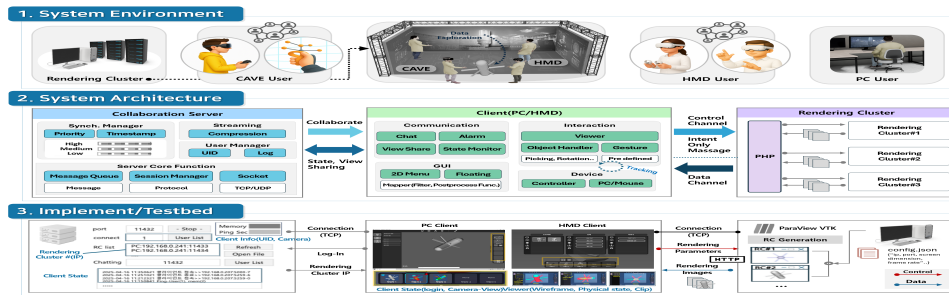


Figure 1: System architecture showing the ParaView/VTK rendering cluster, collaboration server, and Unity/Unreal clients.

## Abstract

We present a hybrid Rendering-as-a-Service (RaaS) architecture for low-latency, multi-user collaboration on TB–PB-scale scientific datasets across heterogeneous devices. It couples a server-side visualization engine (ParaView/VTK) with extended reality (XR) clients (Unity/Unreal) via a collaboration server. The design follows three principles: clear server–client role partitioning (server: high-cost filtering/rendering; clients: input/feedback), authoritative server-driven synchronization, and lightweight intent-only control. In a LAN prototype with one PC and two HMD users, per-request server time stayed under 20 ms; 20 Hz broadcasts maintained coherence of camera pose, selection, and clipping, sustaining interactive frame rates for tens-of-GB datasets. Though full user studies are beyond scope, results show the hybrid design avoids event-handling stalls common in scientific visualization and enables equivalent collaboration across heterogeneous environments.

## ACM Reference Format:

Jin Sung Choi, Yongwan Kim, Suran Park, and Ki-Hong Kim. 2025. A Hybrid XR Framework for Collaborative Scientific Visualization: Rendering-as-a-Service with ParaView and Real-Time Engines. In *SIGGRAPH Asia 2025 Posters (SA Posters '25)*, December 15–18, 2025, Hong Kong, Hong Kong. ACM, New York, NY, USA, 2 pages. <https://doi.org/10.1145/3757374.3771507>

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

SA Posters '25, Hong Kong, Hong Kong

© 2025 Copyright held by the owner/author(s).

ACM ISBN 979-8-4007-2134-2/25/12

<https://doi.org/10.1145/3757374.3771507>

## 1 Introduction

Large-scale scientific simulations produce TB–PB-scale datasets requiring visualization pipelines with filtering and post-processing. Tools like ParaView and VisIt excel at large-data processing but are optimized for single-user desktop workflows, limiting low-latency, multi-user interaction in immersive VR [Liu et al. 2022]. In ParaView-based rendering, compute-intensive stages can block event handling, reducing interactivity for remote collaboration across PC/HMD/CAVE systems [Ayachit 2015]. In contrast, Unity and Unreal use event-driven loops with low-latency input. We propose and validate a hybrid Rendering-as-a-Service (RaaS) architecture coupling a ParaView-based rendering cluster with Unity/Unreal clients (Fig. 1), mediated by a collaboration server coordinating local and remote participants [Porcino et al. 2022]. The design follows three core principles: explicit server–client role partitioning, synchronization via an authoritative server state, and lightweight intent-only control transmitting user intent instead of dense interaction traces.

## 2 System overview and workflow

The framework comprises three components (Figs. 1–2): (1) a rendering cluster (ParaView/VTK) performing data filtering, transformation, and image compositing; (2) clients (Unity/Unreal) that capture input, display streamed images, and provide local feedback (e.g., cursors, widgets, avatars); and (3) a collaboration server that manages sessions, roles, conflict arbitration, and consistent viewpoints across participants. Control and data are decoupled (Fig. 2). Clients send low-latency state updates to the collaboration server at 20 Hz, which aggregates them into an authoritative state and propagates updates

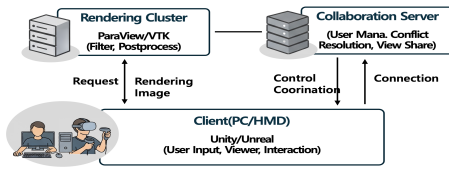


Figure 2: Component Role and Processing Flow.

via 20 Hz broadcasts, aligning viewpoints and selections across heterogeneous devices (PC/HMD/CAVE). For rendering, clients issue HTTP requests to the cluster as needed; the cluster applies filters (e.g., clip/slice/threshold), encodes results, and streams images at up to 30 fps. Transport is split—TCP/WebSocket for control and state, HTTP for rendering and image delivery—balancing latency and robustness. This separation prevents compute-heavy frame generation from blocking input handling, reducing contention between rendering and interaction. Workflow summary (Fig. 3): clients join → report conditions at 20 Hz → server arbitrates and forwards intents → cluster applies filters and streams frames → server broadcasts authoritative state and relays chat/alerts. This mediation, combined with intent-only lightweight control, minimizes bandwidth and sustains consistent multi-user collaboration in asymmetric PC/HMD/CAVE environments.

Session joins and state mediation: Clients join via the collaboration server (Fig. 3), completing a lightweight handshake to obtain a user index, role (host, follower, observer), and available cluster endpoints. After joining, clients send 20 Hz condition packets—camera pose, selection, clipping, and alerts—to the server, which aggregates updates and arbitrates conflicts using role priority with timestamp tie-breakers, maintaining an authoritative state broadcast to all participants. When users interact, clients send intent messages (navigate/select/clip); the server validates and propagates resulting states, while clients directly request rendering operations from the cluster over HTTP and display streamed images.

### 3 Implementation and prototype

Our prototype consists of a server-side rendering cluster built on ParaView/VTK, a multi-user collaboration server developed with WPF (C#), and Unity (C#) clients. Clients communicate with the rendering cluster over HTTP for image and visualization requests, while exchanging state, chat, and alerts with the TCP-based collaboration server. Cluster configuration—IP/port, display resolution, frame rate, and related parameters—is managed via a configuration file (e.g., config.json) to ensure consistent deployment and reproducibility (Figs. 1–3).

Responsibilities are strictly partitioned: clients explore rendered images from the cluster—applying filters (e.g., clip, slice), zooming, and rotating—sending only user intent to the collaboration server. The server validates requests, propagates an authoritative state to all participants, and clients invoke corresponding operations on the rendering cluster via HTTP. The cluster updates the scene and streams resulting frames back to the requesting client over a dedicated image channel. The UI is optimized for both PC and HMD (Figs. 1–3): PCs use a menu-centric 2D interface, while HMDs employ controller-driven floating menus with ray-cast selection.

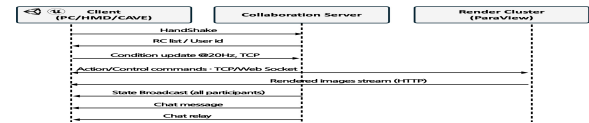


Figure 3: Workflow of Client-Server-Rendering Cluster.

Both share a unified interaction vocabulary—navigate, select, manipulate, clip—to maintain functional equivalence across devices. Each client includes a session panel showing active participants and the rendering status of their current view, enhancing collaborator awareness.

### 4 Results and discussion

We evaluated responsiveness and collaboration fidelity on a LAN with one PC client and two HMD users, using tens-of-GB datasets and exploration/selection/clipping tasks.

*Performance.* Per-request server time remained under 20 ms across all tasks, with 20 Hz state broadcasts maintaining alignment of camera pose and selections. Frame rates by resolution were  $800 \times 600: \geq 60$  fps;  $1280 \times 720: \sim 30/45$  fps (cluster/client);  $1920 \times 1280: \sim 15/20$  fps. Control traffic was minimal ( $\sim 100$  B  $\times 20$  Hz  $\approx 2$  KB/s per client). With up to three concurrent clients, the system ran stably for 120 continuous hours. Limitations and outlook. We present a preliminary, interoperable framework linking scientific-visualization tools with Unity/Unreal for multi-user collaboration, verified on a LAN prototype. Peer feedback highlighted three priorities: (i) scaling the collaboration server for  $\geq 20$  users; (ii) developing adaptive data-exploration methods tuned to task and dataset characteristics; and (iii) refining UI/UX through structured usability studies to ensure parity and efficiency across PC/HMD/CAVE systems.

### 5 Conclusion and future work

We introduced a hybrid XR visualization architecture and reference implementation based on separation of responsibilities, authoritative synchronization, and intent-driven control. Future work will extend validation to WAN settings and integrate natural-language or voice interaction by connecting LLMs to ParaView via the Model Context Protocol (MCP) and Python API, mapping user utterances directly to intents.

### Acknowledgments

This research was supported by the National Research Council of Science & Technology (NST) grant by the Korea government (MSIT) (No. GTL24031-000)

### References

- Utkarsh Ayachit. 2015. *The Paraview Guide: A Parallel Visualization Application*. Kitware, Inc.
- Richen Liu, Min Gao, Lijun Wang, Xiaohan Wang, Yuzhe Xiang, Aolin Zhang, Jiazhi Xia, Yi Chen, and Siming Chen. 2022. Interactive Extended Reality Techniques in Information Visualization. *IEEE Transactions on Human-Machine Systems* 52, 6 (2022), 1338–1351. doi:10.1109/THMS.2022.3211317
- Thiago Porcino, Seyed Adel Ghaeinian, Juliano Franz, Joseph Malloch, and Derek Reilly. 2022. Design of an extended reality collaboration architecture for mixed immersive and multi-surface interaction. In *International Conference on Entertainment Computing*. Springer, 112–122. doi:10.1007/978-3-031-20212-4\_9