动态点云自适应流媒体的可扩展实时转码

Michael
R
udolph
michael.rudolph@ikt.uni-hannover.de 莱布尼茨汉诺威大学 汉诺威，德国

Matthias
DeFrémattthias.defre@ugent.be
IDLab，根特大学 - imec 根特，比利时

Finn Schnier
finn.schnier@stud.uni-hannover.de 莱布尼茨汉诺威大学 汉诺威，德国

Tim Wauters
tim.wauters@ugent.be
IDLab，根特大学 - imec 根特，比利时

Amr Rizk
a
mr.rizk@ikt.uni-hannover.de 莱布尼茨汉诺威大学 汉诺威，德国

arXiv:2603.08417v1 [cs.MM] 9 Mar 2026

摘要
对动态点云序列进行实时转码，可降低存储需求，并在按需流媒体场景中，实质性地增加可用表示的数量。然而，实时转码会给媒体提供商的基础设施带来额外负载。虽然通过重新编码底层视频比特流，可以高效地转码 V-PCC 编码内容，并且这一过程可从硬件加速的视频编解码器实现中显著受益，但此类系统的可扩展性仍不明确。本文介绍并评估了一种利用实时转码的动态点云流媒体系统。我们探究该系统在请求完成时间方面的可扩展性极限，并重点评估用户感知的体验质量。我们通过实证展示，缓存和推测式转码如何显著降低转码负载，从而支持更多并发客户端。

CCS 概念
• **信息系统** → **多媒体流媒体**。

关键词
点云、自适应码率流媒体、即时转码、MPEG-DASH

1 引言
面向增强现实与虚拟现实（AR/VR）应用的大规模沉浸式内容分发，为网络多媒体系统带来了新的挑战。点云已成为沉浸式流媒体的一种热门表示，因为它们易于采集和渲染 [30]，同时支持灵活且开销低的处理方式，例如视点相关分割 [24] 或基于采样的 [4]质量自适应。然而，逼真场景的点云表示通常每帧包含数百万个点，导致原始数据速率高得难以承受。因此，业界涌现出多种压缩方法 [15, 16] ，以支持带宽受限的终端用户设备。

虽然压缩降低了所需比特率，但波动的网络状况要求灵活且可扩展的传输机制。对于按需流媒体场景，自适应码率（ABR）流媒体通过客户端自适应实现可扩展的内容分发，同时依托基于 HTTP 的传输确保广泛的可访问性。因此，已有工作将这一概念应用于沉浸式多媒体，特别是动态点云流媒体 [11, 28]。简而言之，点云视频被划分为可以独立下载和解码的时间片段。由于客户端仅下载并缓冲一部分内容，这既能确保流媒体会话的启动延迟较短，又能兼顾跳过或提前停止等用户交互。用户规模上的可扩展性通过客户端质量自适应来保证。客户端针对每个分片从一组表示中进行选择，从而在码率与画质之间提供不同的权衡。这种自适应基于对网络状况的当前估计、缓冲区水位，以及沉浸式内容可能的可见性估计 [24, 28]。

内容提供商会提供一组以不同比特率编码的表示，称为码率梯级。因此，客户端可以为每个分片选择码率梯级中的一种表示，每种表示都提供独特的画质与码率权衡。显然，在码率梯级中提供更多内容表示，就能实现更精细的调整和更平滑的画质转换。

已观察到，流式传输的媒体中只有一小部分广受欢迎，即内容观看量的分布在经验上遵循幂律 [3]。由于大多数内容并不热门，可能仅被

观看少数几次，这会导致因保存同一分片的多种表示而浪费存储。实时转码 [17] 能够解决这一问题：内容提供商无需存储所有表示，而只需存储其中一部分（通常是以最高速率编码的表示）。收到请求后，可通过转码从高速率表示导出较低比特率的表示。对于采用基于视频的点云压缩（V-PCC）编码的内容，已证明可使用仅作用于已编码内容的视频比特流的高效转码技术 [22] 来实现这一点。

在本研究中，我们提出了一种基于即时转码范式的动态点云自适应比特率流媒体系统。我们通过实验探究了在转码资源有限的情况下，随着客户端数量增加，该系统的可扩展性。

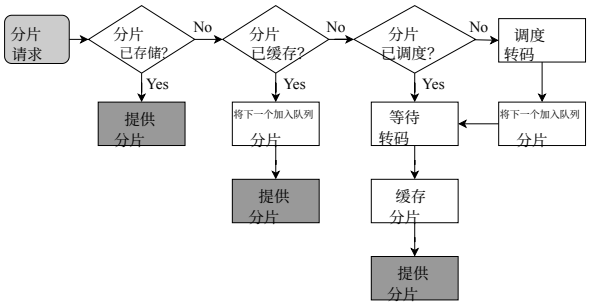


图 1：转码决策过程：服务器收到分片后，要么从存储中提供内容，要么将请求转发到后端，由后端管理转码任务。

2 相关工作

沉浸式内容流式传输系统的一项关键设计决策是选择传输协议。在 [20], 中，采用了基于TCP的协议基于HTTP的动态自适应流媒体传输（DASH），因为该用例要求将完整数据流可靠地传送给所有用户。相比之下，在对延迟敏感的场景中，通常更倾向于使用基于UDP的协议，例如WebRTC和QUIC，因为它们优先保证较低的端到端延迟，而非严格的可靠性 [7, 8]。

由于活跃用户数量众多，沉浸式流媒体系统可能会遭遇网络拥塞，从而降低每位用户可用的带宽 [32]。带宽波动可能导致播放中断，迫使用户等待后续帧接收完毕 [19]。这些卡顿事件会显著降低用户的整体体验质量（QoE）。为缓解这一问题，通常会采用自适应流媒体技术，动态降低内容画质以避免播放卡顿 [21]。正如 [25], 所示，画质下降对体验质量的影响不如卡顿事件那么严重。许多最初为传统视频流传输和360度视频流式传输开发的自适应流媒体技术，现已被改造用于支持动态点云流媒体。在 [11], 中，作者扩展了DASH [23] 以支持点云内容，并采用采样策略，为存储在媒体服务器上的动态点云场景生成多个表示。配备

对此， [28] 的作者针对包含多个动态点云的场景提出了码率分配方法，以优化用户视口中的视觉质量，这一点后来得到了 [27]中客观质量与主观质量研究的支持。类似地，其他方法通过点云分块策略 [24], 扩展了这些概念，从而能够基于用户视口实现更细粒度的画质自适应。其他方法 [4, 31]

则通过点云采样创建多种不同的画质表示来降低编码时延，这些表示可在接收端进行组合。此外，这些系统还可结合预测算法，提前估计最优画质表示，从而进一步降低端到端时延 [12, 33]。

利用自适应比特率流媒体传输点云序列，需要采用高效的方法来准备和压缩多种质量表示。大多数先前关于按需点云流传输的工作 [10, 24, 27, 28] 都依赖V-PCC标准 [16], ，因为它在动态点云序列上实现了前所未有的率失真性能，但代价是较高的编码延迟 [2]。尽管已有研究提出了快速的 V-PCC编解码器实现 [6] ，编码时延仍然令人担忧。这使得相关应用仅限于按需场景：内容以离线方式预先编码，此时延迟并不那么关键。因此，如视频流传输 [17], 中所探索的实时转码仍然面临挑战。这与一篇关于体积视频流传输领域最新进展的综述 [29],形成对照，该综述强调了流媒体系统中转码的必要性。同样， [18] 的作者也强调，体积视频流传输系统需要低延迟、低计算量转码。与对 V-PCC 编码内容进行完整编码和解码不同，仅对已编码内容中的视频子流进行转码 [22] 能够降低复杂度，并有助于弥合这一差距。

3 方法

3.1 系统描述

接下来，我们介绍一种用于 HTTP 自适应流传输的按需流媒体系统。转码服务由媒体服务器前端和转码器后端组成，其处理请求的决策流程如图 1 所示。

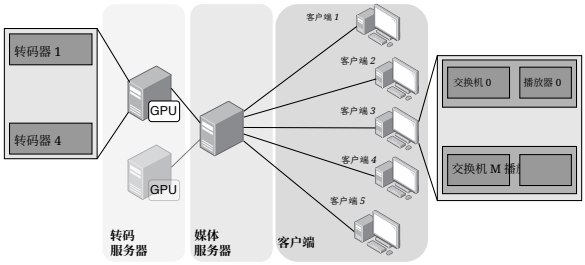


图 2：实验设置：中心节点充当媒体服务器。我们使用 1 或 2 个节点（以较低的不透明度表示）作为转码后端，并使用 5 个客户端节点，每个节点模拟可配置数量的播放器。后端中的每个转码节点都配备了图形处理器，支持 4 个并行转码器。为模拟播放器，我们通过软件交换机容器转发流量，该容器会为每个播放器模拟来自 [26] 的网络轨迹。

除了转码器之外，我们还提出了两项扩展，旨在减少所需的转码任务数量。首先，我们为转码片段引入最近最少使用（LRU）缓存，即淘汰标记时间最早的分片。此外，一旦收到请求 n，我们就会将序列中的分片$n + 1$ 加入队列。这基于 QoE 调优假设：客户端只会在必要时切换画质，并且通常会尽量持续播放相同画质的分片。

图 1 总结了逐分片的决策流程。服务器收到片段请求后，如果存储中有所请求的片段，则直接提供；否则，将请求转发给后端。验证请求后，后端首先检查缓存。如果片段已缓存，则调度一个针对下一片段的预测性转码任务，并返回所请求的片段。如果没有缓存，后端会检查是否已有针对同一片段的转码任务正在进行。如果有，则等待任务完成；否则，调度一个新的转码任务。无论哪种情况，都会对后续片段触发推测式转码。请求的片段完成转码后，会被加入缓存，后端随后将转码后的片段返回给客户端。

3.2 实验设置

我们在 imec Virtual Wall [13] 测试平台上开展实验，共使用 8 个裸机计算节点。实验拓扑如图 2 所示。一个由 NGINX 提供前端服务的中央媒体服务器，通过 FastAPI 后端处理传入的媒体请求，并通过 Redis 支持的任务队列分发转码任务。容器化 GPU 工作节点异步获取任务，并使用 RABBIT 转码器 [22]独立转码 V-PCC 码流。

我们使用基于 iStream DASH 播放器 [1] 的客户端容器。播放器的目标缓冲区设为 12 s，安全缓冲区水位设为 8 s，紧急缓冲区水位设为 2 s。卡顿后恢复播放所需的最短缓冲时长设为 2 s，启动缓冲阈值设为 3 s。控制脚本会为每个流媒体会话随机选择 4 个编码点云视频中的一个。实验中各流媒体会话的启动时间遵循参数为 $\lambda = 0.1$ 的指数分布，以避免同步启动。我们使用交换机容器为每个客户端调整带宽，并使用 tc 根据 [26], 中给出的带宽轨迹模拟链路，其中每个客户端都分配有独立的带宽轨迹。我们发现，这些轨迹与编码测试序列的码率梯级十分契合，形成了具有挑战性的条件，迫使客户端使用所提供表示的完整范围。

实验评估使用 8iVFBv2 数据集 [5] 中的全部四个序列，并通过循环播放帧将其扩展至 80 秒。扩展序列采用 [14] 中定义的 R5 V-PCC 配置进行编码。转码工作进程使用 nvdec/nvenc 对预编码比特流中的几何信息和属性比特流进行重新编码，将其转换为 R1-R4 [14] 码率配置，并使用 *p2* 预设。

每次流媒体实验运行 600 秒，后端使用 1 或 2 个 GPU 节点。由于每个 GPU 节点都配备 NVIDIA GeForce GTX1080 显卡，因此可以同时运行 8 个视频编码器，即每个节点有 4 个转码工作进程。DASH 客户端分布在物理客户端节点上，每个物理客户端节点运行 4 或 8 个 DASH 客户端。每个 DASH 客户端都使用各自的带宽仿真。

我们比较以下实验变体：

- **基线（B）**：作为实验的基线，我们预先编码所有表示，并将其存储在媒体服务器上。因此，无需进行转码。
- **转码（T）**：此变体会在收到请求时进行转码，不使用额外的缓存、预编码或预测编码。

此外，我们还探索了对转码系统的以下改进：

- **缓存（+C）**：媒体服务器配有一个小型缓存（128 MB），可存储部分转码片段以便直接响应。
- **预测性转码（+P)**：一旦请求某个分片，我们不仅会触发该分片的转码任务，还会为采用相同编码配置的后续分片触发转码任务。

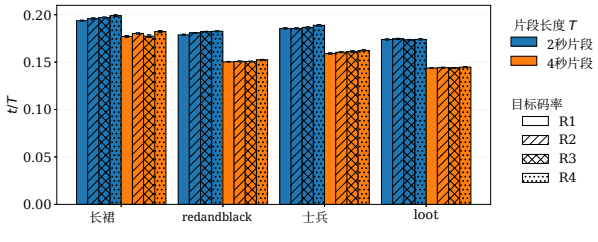
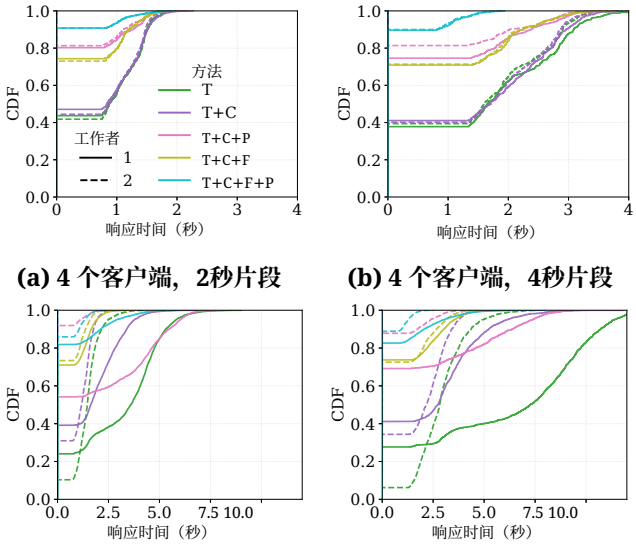


图3：按片段长度 T 归一化的转码时间 t，用于从码率配置R5转码到R1–R4。对每个循环序列的20个连续转码任务取平均。置信区间仍然很小。

- **回退预编码（+F）**：除了仅预编码并存储最高码率表示外，我们还会在媒体服务器上存储最低码率表示。由于当客户端的缓冲区水平低于阈值时会回退到最低码率表示，因此无需预先执行转码任务，服务器即可立即响应。



(a) 4 个客户端，2秒片段 **(b) 4 个客户端，4秒片段**

(c) 24 个客户端，2秒片段 **(d) 24 个客户端，4秒片段**

图 4：响应时间（从接收到请求到服务器端响应）的累积分布函数。零延迟处的累积分布函数值（左侧纵轴）表示即时处理的请求所占比例。

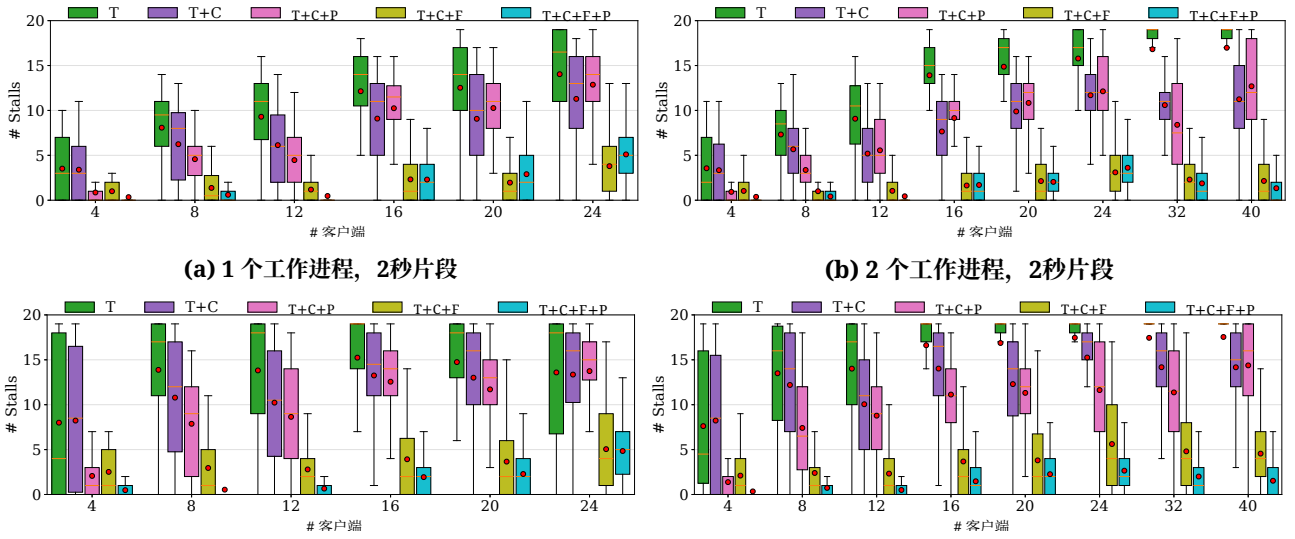
4 评估

4.1 服务器端响应时间

由于实时转码需要在提供所请求的媒体之前对其进行处理，因此会引入额外的响应延迟。我们在图3中可视化了测试平台上转码延迟的独立时间测量结果，这些结果是对20次运行取平均得到的。为了能够比较不同的片段长度，我们通过将测得的转码时间 t 除以片段长度 T 来计算归一化转码时间。尽管较长的分段往往会产生更低的归一化转码时间 t/T（即更高的转码吞吐量），但我们实验中考虑的两种片段长度都可以足够快地完成转码，从而支持实时转码。

从系统角度来看，转码任务延迟本身仅占响应延迟的一部分，即从请求到达服务器到媒体服务器发回响应之间的时间。在图 4a 和图 4b 中，考察有 4 个客户端时独立转码器系统（T；绿色）的响应时间累积分布函数可以发现，服务器接收到的所有请求的响应时间变化很大。大约 40% 的请求对应的分片已存储在媒体服务器上，因此几乎可以即时处理。相比之下，其余请求会触发转码任务，由于转码器池中存在额外的排队延迟，因而导致响应时间有所变化。对于这种情况，使用 1 个或 2 个转码工作进程的 4 客户端流媒体实验之间的差异很小，表明两种配置的资源都已足够。引入缓存（T+C）后，可以显著提高能够直接处理的请求比例，即减少转码任务的数量，如图 4 所示。对于预测编码（T+C+P），我们也观察到类似效果：在假设客户端不会切换当前流的画质时，系统会针对相同配置下的后续分片触发推测式转码任务，从而进一步提高直接响应的比例。相比之下，在配置 T+C+F 中对最低画质表示进行预编码，得到的直接响应略少于 T+C+P。不过需要注意的是，这需要在媒体服务器上额外存储最低码率表示。最后，配置为 T+C+P+F 的系统仅会延迟 10% 的传入请求，其余响应均可直接从缓存或存储中处理。

并发客户端数量越多，请求速率也越高；如图 4c 和图 4d 所示，服务器端响应延迟随之增加。由于资源限制，转码任务只能由数量有限的工作进程处理，因此请求激增会导致任务的排队延迟增加。对于 24 个客户端，使用 1 个或 2 个转码工作进程的差异十分明显。



(a) 1 个工作进程，2秒片段　(b) 2 个工作进程，2秒片段

(c) 1 个工作进程，4秒片段　(d) 2 个工作进程，4秒片段

图5：每个流媒体会话的平均卡顿事件数：将最低质量表示预编码作为备用方案（F），在减少同时连接的客户端数量增加时出现的卡顿方面效果最佳；而在负载较高时，缓存（C）效果更好。对于客户端数量较少的情况，预测编码（P）可以减少卡顿事件，但会更快耗尽资源，因此客户端数量增加时会导致更多卡顿。

4.2 客户端质量

响应延迟的高度变化性会显著影响流媒体播放器的行为。服务器端响应延迟较高会影响客户端的缓冲区填充，导致基于缓冲区填充的质量选择出现画质变化；当播放器从缓冲区中消耗片段的速度快于缓冲区填充速度时，最终会发生卡顿事件。图 5 展示了所有实验配置下每个流媒体会话发生的卡顿事件数量分布。

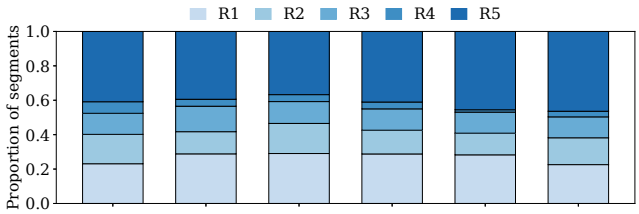
仅转码：我们发现，仅使用转码器（T）即使在客户端数量较少时也会导致大量卡顿事件。这与使用的工作进程数量无关。即使资源充足，较高的服务器端响应延迟（参见图4）也已严重影响客户端的缓冲区管理。

转码与缓存（T+C）：增加缓存（T+C）有助于在客户端数量较少时略微减少卡顿次数。客户端数量增多时，缓存的效果更加明显：与仅采用转码（T）相比，采用缓存的系统在 40 个客户端时能有效减少卡顿事件（图 5b 和图 5d）。在大规模场景下，缓存可以减少转码任务的数量，并提高可直接处理的请求比例。

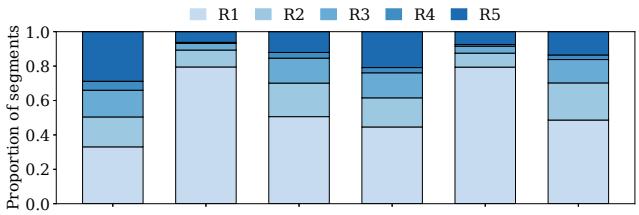
转码、缓存与预测转码（T+C+P）：当资源充足时，加入预测编码机制（T+C+P）可缩短少量客户端的响应时间但资源耗尽且任务队列已满后，对连续片段进行预测编码的效果会降低，因为这会增加转码工作负载并填满任务队列。

预编码最低质量（+F）：最后，我们发现，预编码最低质量表示（T+C+F 和 T+C+P+F）在客户端数量增加时，最能减少卡顿次数。这两种方案都能大幅减少服务器端的实时转码任务，使系统在客户端数量增加时具备显著更强的扩展能力。

由于所有实时转码变体都会影响客户端的自适应比特率算法选择，我们展示了它们对流式传输码率表示的影响，例如图 6 中序列*redandblack*的情况。图 6a 展示了 4 个客户端的画质选择，以及所有实时转码配置的结果；结果表明，与服务器上具备所有画质的基线（B）相比，画质选择仍然相似。在这种情况下，转码具有显著优势：无需像基线（B）那样保留所有画质，同时流式传输片段的质量分布保持不变。请注意，客户端运行无线网络轨迹的独立样本路径，因此可用带宽会有所变化，进而导致请求码率不同。接下来，我们观察图 6b 中 24 个并发客户端的画质下降。我们发现，资源耗尽后，播放器无法将缓冲区提升至安全水平，



(a) 4 个客户端



(b) 24 个客户端

图6：序列 *redandblack* 在片段长度为 4 秒、使用 2 个工作进程时，流式传输片段质量的占比。堆叠条形图表示实验中所有 DASH 客户端接收到的各种质量所占比例，红线表示平均表示级别。

从而无法切换到更高码率的表示。缓存、预测性预编码以及预编码最低质量（+C，+P+F）等附加技术有助于改善码率分布，使其接近基线，但仍可明显看出一定的性能损失。

4.3 讨论

在我们实验所面临的严苛流式传输条件下，实时转码带来的额外服务器端响应延迟会直接影响 DASH 客户端的行为。若仅使用转码器（T），而不采取其他服务器端措施来缩短响应时间，则不足以支持大规模实时转码，因为随着客户端数量增加，客户端播放时会出现严重卡顿。

在计算资源充足的前提下，预测性预编码结合缓存（T+C+P）可以通过直接在服务器上处理大量请求，大幅减少卡顿次数。然而，在大规模场景下，对从未被请求的分片进行推测性转码会增加服务器负载，却没有任何收益。

最后，我们的结果表明，实时转码能够提供与服务器上具备所有画质时相同的流式比特率分布。这将显著降低为自适应码率流媒体存储动态点云场景的需求。从平均卡顿事件数和流式比特率分布可以看出，随着客户端数量增加而扩展实时转码，能够从缓存、预测性预编码以及对最低质量进行预编码等服务器端辅助措施中显著受益。

5 结论

在本工作中，我们提出并评估了一种面向大规模动态点云序列自适应码率流媒体的实时转码系统。研究结果表明，仅部署转码器来实现实时转码范式并不足够，尤其是在缓冲区填充目标较小且网络波动性较高的情况下。为此，我们提出了对转码系统的系统级扩展，与仅使用转码器的方法相比，可大幅减少卡顿次数。

与所有表示均已在服务器端就绪的传统流媒体场景相比，带有服务器端扩展的实时转码能够在并发客户端数量不断增加的情况下，于资源耗尽之前提供相同的码率分布。在转码资源有限的条件下，随着并发客户端数量增加，我们观察到额外的卡顿，以及所选码率表示降低。

我们指出了若干有前景的未来工作方向：通过预测转码时间并选择性地以较低码率表示进行响应，将服务器端画质决策纳入其中，或可避免作业队列堆积和过长的响应时间 [9]。此外，目前客户端并不知道片段请求是可以立即得到提供，还是需要服务器端处理。在媒体呈现描述（MPD）中公开转码时间估计，供感知转码的ABR算法使用，可以缓解卡顿，但代价是视觉质量下降。

致谢

本项目由欧盟资助（SPIRIT，101070672）。然而，文中表达的观点和意见仅代表作者本人，并不一定反映欧盟的观点。欧盟及资助机构均不对这些观点和意见承担责任。SPIRIT 项目还获得了瑞士国家教育、研究与创新秘书处（SERI）的资助。

参考文献

[1] Akram Ansari 和 Mea Wang。2023 年。iStream Player：一种多功能视频播放器框架。载于 第 *33* 届数字音频与视频网络及操作系统支持研讨会论文集。65–71。[2] Mai Bui、Lin-Ching Chang、Hang Liu、Qi Zhao 和 Genshe Chen。2021年。三维点云压缩方法比较研究。载于 *2021*年 *IEEE* 大数据国际会议（大数据）。IEEE，5859–5861。[3] Meeyoung Cha、Haewoon Kwak、Pablo Rodriguez、Yong-Yeol Ahn 和 Sue Moon。2009年。分析大规模用户生成内容系统的视频流行度特征。*IEEE/ACM*网络汇刊 17，5（2009），1357–1370。[4] Matthias De Fré、Jeroen van der Hooft、Tim Wauters 和 Filip De Turck。2024年。基于多描述编码的可扩展体积视频传输：用于实时一对多 WebRTC 会议。载于 第 *15* 届 *ACM*多媒体系统会议论文集。121–131。[5] Eugene d’Eon、Bob Harrison、Taos Myers 和 Philip A. Chou。2017年。8i体素化全身——一个体素化点云数据集。ISO/IEC JTC1/SC29 联合 WG11/WG1（MPEG/JPEG）输入文件 WG11M40059/WG1M74006（2017）。[6] Louis Fréneau、Guillaume Gautier、Alexandre Mercat 和 Jarno Vanne。2025年。uvgVPCCenc：用于快速 V-PCC 压缩的实用开源编码器。载于 第 *16* 届 *ACM*多媒体系统会议论文集。235–241。[7] Simon NB Gunkel、Rick Hindriks、Yonatan Shiferaw、Sylvie Dijkstra-Soudarissanane 和 Omar Niamut。2024年。基于 VP9 码流的分块多点控制单元：在沉浸式三维通信系统中扩展同步 RGBD 用户流。载于 第 *15* 届 *ACM*多媒体系统会议论文集。23–33。[8] Muhammad Haseeb、Eugene Chai 和 Matteo Varvello。2025年。Octavius：通过自适应编码和 QUIC 实现三维点云高效传输。载于 第 *3* 届新兴多媒体系统研讨会论文集。19–24。[9] Felix Hechler 、Michael Rudolph 和 Amr Rizk。2024年。360度视频流的准时转码。载于 第 *15* 届 *ACM*多媒体系统会议论文集。89–99。[10] Hadi Heidarirad 和 Mea Wang。2025年。VV-DASH：体积视频 DASH 流传输框架。载于 第 *16* 届 *ACM*多媒体系统会议论文集。256–262。[11] Mohammad Hosseini 和 Christian Timmerer。2018年。动态自适应点云流传输。载于 第 *23* 届分组视频研讨会论文集。25–30。[12] Kaiyuan Hu、Haowen Yang、Yili Jin、Junhua Liu、Yongting Chen、Miao Zhang 和 Fangxin Wang。2023年。了解用户观看体积视频时的行为：数据集、分析与预测。载于 第 *31* 届 *ACM*国际多媒体会议论文集。1108–1116。[13] imec iLab.t。无日期。Virtual Wall测试床文档。https://doc.ilabt.imec.be/ilabt/virtualwall/ 访问日期：2026年1月16日。[14] ISO/IEC JTC 1/SC 29/WG 11。2020年。V-PCC 通用测试条件。（2020）。https://www.mpeg.org/standards/MPEG-I/5/ 第 131 次 MPEG 会议。[15] Ge Li、Wei Gao 和高文。2024年。MPEG 基于几何的点云压缩（基于几何的点云压缩）标准。载于 点云压缩：技术与标准化。施普林格·自然，135–165。[16] Ge Li、Wei Gao 和高文。2024年。MPEG 基于视频的点云压缩（V-PCC）标准。载于 点云压缩：技术与标准化。施普林格·自然，199–218。[17] Xiangbo Li、Mohsen Amini Salehi、Magdy Bayoumi、Nian-Feng Tzeng 和 Rajkumar Buyya。2018年。利用异构云服务实现经济高效且稳健的按需视频转码。*IEEE*并行与分布式系统汇刊 29，3（2018年3月），556–571。[18] Zhi Liu、Qiyue Li、Xianfu Chen、Celimuge Wu、Susumu Ishihara、Jie Li 和 Yusheng Ji。2021年。点云视频流传输：挑战与解决方案。*IEEE Network* 35，5（2021年9月），202–209。[19] J-M Martinez-Caro 和 M-D Cano。2021年。关于识别和预测卡顿事件以提升视频流传输体验质量的研究。*Electronics* 10，6（2021年），753。[20] De Mi、Joe Eyles、Tero Jokela、Swen Petersen、Roman Odarchenko、Ece Öztürk、Duy-Kha Chau、Tuan Tran、Rory Turnbull、Heikki Kokkinen 等。2020年。在5G广播和组播测试网络中展示沉浸式媒体传输。*IEEE Transactions on Broadcasting* 66，2（2020年），555–570。[21] Chunyu Qiao、Jiliang Wang 和 Yunhao Liu。2020年。超越体验质量：边缘视频流传输中的多样性自适应。*IEEE/ACM*网络汇刊 29，1（2020年），289–302。[22] Michael Rudolph、Stefan Schneegass 和 Amr Rizk。2025年。实时转码V-PCC点云流。*ACM Trans. MultimediaComput. Commun. Appl.* 21，9（2025年9月），250:1–250:22。[23] Iraj Sodagar。2011年。用于互联网多媒体流传输的 MPEG-DASH标准。*IEEE MultiMedia* 18，4（2011年4月），62–67。[24] Shishir Subramanyam、Irene Viola、Alan Hanjalic 和 Pablo Cesar。2020年。通过低复杂度切片实现以用户为中心的动态点云自适应流媒体。载于第*28*届*ACM*国际多媒体会议论文集。3669–3677。[25] Babak Taraghi、Minh Nguyen、Hadi Amirpour 和 Christian Timmerer。2021年。Intense：深入研究HTTP自适应流传输中的卡顿事件和质量切换及其对体验质量的影响。*IEEE Access* 9（2021年），118087–118098。[26] Jeroen van der Hooft、Stefano Petrangeli、Tim Wauters、Rafael Huysegems、Patrice Rondao Alface、Tom Bostoen 和 Filip De Turck。2016年。基于HTTP/2的自适应流媒体：通过4G/LTE网络传输HEVC视频。*IEEE Communications Letters* 20，11（2016年11月），2177–2180。[27] Jeroen van der Hooft、Maria Torres Vega、Christian Timmerer、Ali C. Begen、Filip De Turck 和 Raimund Schatz。2020年。自适应点云流传输的客观与主观体验质量评估。载于*2020*年第十二届国际多媒体体验质量会议（*QoMEX*）。1–6。[28] Jeroen van der Hooft、Tim Wauters、Filip De Turck、Christian Timmerer 和 Hermann Hellwagner。2019年。通过点云压缩实现6DoF HTTP自适应流媒体。载于第*27*届*ACM*国际多媒体会议论文集。2405–2413。[29] Irene Viola 和 Pablo Cesar。2023年。体积视频流传输：当前方法与实现。载于沉浸式视频技术。Academic Press，425–443。[30] Irene Viola、Jack Jansen、Shishir Subramanyam、Ignacio Reimat 和 Pablo Cesar。2023年。VR2Gather：面向自适应多方实时通信的协作式社交虚拟现实系统。*IEEE MultiMedia* 30，2（2023年4月），48–59。[31] Jingrou Wu、Haoxian Liu、Jin Zhang、Dan Wang 和 Jing Jiang。2025年。P2VS：网络动态环境下基于渐进式分区的体积视频流传输。载于第*33*届*ACM*国际多媒体会议论文集。12064–12073。[32] Xinlei Yang、Hao Lin、Zhenhua Li、Feng Qian、Xingyao Li、Zhiming He、Xudong Wu、Xianlong Wang、Yunhao Liu、Zhi Liao 等。2022年。实践中的移动接入带宽：测量、分析及影响。载于*ACM SIGCOMM 2022*会议论文集。114–128。[33] Ran Zhang、Jiang Liu、Fangqi Liu、Tao Huang、Qinqin Tang、Shangguang Wang 和 F Richard Yu。2021年。基于缓冲区感知的虚拟现实视频流传输：个性化与隐私保护的视口预测。*IEEE Journal on Selected Areas in Communications* 40，2（2021年），694–709。