StreamWise：以大规模实时方式服务多模态生成

Haoran Qiu Gohar Irfan Chaudhry* Chaojie Zhang Íñigo Goiri Microsoft Azure Research MITCSAIL Microsoft Azure Research Microsoft Azure Research 雷德蒙德，美国 剑桥，美国 雷德蒙德，美国 雷德蒙德，美国

Esha Choukse
Microsoft Azure Research 雷德蒙德，美国

Rodrigo Fonseca
Microsoft Azure Research 雷德蒙德，美国

Ricardo Bianchini
Microsoft Azure Research 雷德蒙德，美国

arXiv:2603.05800v1 [cs.DC] 6 Mar 2026

摘要

多模态生成模型的进步正催生从故事创作到自动媒体合成等新应用。目前，大多数工作负载以批处理模式生成简单输出（例如，根据提示词生成图像），即使是基本结果也往往需要数秒。要大规模提供实时多模态工作流服务，成本高且复杂，需要在严格的延迟和资源约束下，高效协调语言、音频、图像和视频领域的各种模型（每种模型都有独特的资源需求）。

我们以实时播客视频生成为切入点，应对这些挑战，并整合大语言模型、文本转语音和视频音频生成。为满足严格的服务级别目标（SLO），我们设计了自适应模块化服务系统 StreamWise，可动态管理质量（例如，分辨率、清晰度）、模型/内容并行性以及资源感知调度。我们利用异构硬件，最大限度地提升响应速度和效率。例如，该系统可以降低视频分辨率，并为较早的场景分配更多资源。

我们量化了延迟、成本和质量之间的权衡。成本最低的配置可在 A100 GPU 上用 1.4 小时生成一段 10分钟播客视频（比实时生成慢 8.4× 倍），成本不到 25 美元。StreamWise 可在 45 美元以内实现高质量实时流式传输，且启动延迟不到一秒。

1 引言

动机。 大语言模型（LLM）的进步正在改变从开发者工具到教育等众多文本生成应用 [12, 47, 137]。在这一趋势的推动下，多模态内容生成（音频、图像和视频）日益普及 [46, 49, 52, 72, 74,97, 132]。下一个前沿方向是大规模支持面向用户的实时流式应用，从而实现动态故事讲述、个性化辅导和自动化媒体创作等全新体验 [71, 132]。

尽管近期的系统研究大幅提升了基于文本的大语言模型推理的效率和调度 [3, 65, 100], , 但这些技术并未解决其独有的多模态实时生成所带来的挑战。在严格的端到端延迟约束下，服务和编排异构多模态模型仍然成本高昂，并带来了根本性的新系统挑战。

现有流水线通常是具有高延迟的批处理工作流 [46, 49, 72, 74, 97, 132]。商业系统仍然十分昂贵（每分钟超过 2 美元），并且由于计算开销过高，将输出限制为 <10 秒 [49, 97] 。相比之下，流式多模态生成需要低延迟流水线，以便协调跨模态的昂贵人工智能模型，同时满足严格的服务级别目标（SLOs）。

这些需求给硬件和软件基础设施都带来了巨大压力。在进行扩展的同时实时协调这些各不相同的多模态模型（每个模型都具有独特的资源特征），仍然是一个尚未解决的系统挑战。

我们的工作。 我们以一个代表性应用——播客视频生成——为视角，研究实时多模态生成的系统挑战。给定一个输入（例如，一篇研究论文或新闻主题），该工作流会生成多个角色围绕该主题进行对话的视频。这需要将大语言模型、文本转语音合成、图像和视频生成以及视频与音频同步编排为一条统一的低延迟流水线。

为支持这一工作负载，我们设计并实现了一个名为 StreamWise 的模块化、自适应服务栈，在延迟、成本和质量之间取得平衡。该系统支持可调节保真度的流式传输（实时播放），以满足不同的服务级别目标和预算。我们的方法还考虑了计算资源的异构性，并支持多级并行、内容复用和质量自适应。例如，可以以较低分辨率渲染早期场景以减少启动延迟，同时确保后续片段保持高质量。

我们刻画了性能与资源之间的权衡，并量化了系统级决策对延迟、成本和质量的影响。例如，最小配置在单台 8×A100 GPU服务器上生成一段10分钟播客视频需要3.7小时，无法满足实时需求。StreamWise通过结合A100和H100 GPU，实现了亚秒级延迟；在规模化服务时，每段视频的成本低于40美元。

*在微软 Azure Research 实习期间完成的工作。

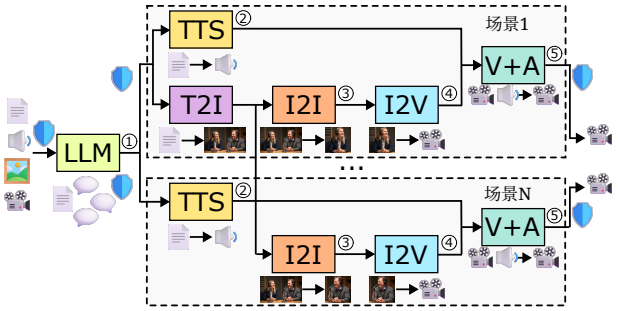

图 1. 视频播客生成的工作流。

摘要。 我们的主要贡献如下：

- 我们设计了一款实时多模态生成应用，并分析了其系统影响。
- 我们分析了延迟、成本与输出质量之间的权衡，并识别出关键瓶颈。
- 我们识别并阐述了系统级优化机会，以进一步提升效率和可扩展性。
- 我们构建了一个模块化服务系统，以支持并行执行和质量感知调度。
- 我们量化了实时约束下服务多模态生成工作流的成本。

2 播客视频生成

概述。 我们以播客视频创作为代表性用例，探讨了实时服务人工智能生成的多模态内容所面临的系统挑战（扩展了播客音频 [48]）。该工作流涵盖所有模态，并涉及在严格延迟约束下运行的多种生成模型。给定多模态输入（例如，一篇包含图表的研究论文或一部电影），系统会生成一段播客风格的视频，由多个角色讨论输入内容。图 1 展示了该工作流：（1）理解输入并生成剧本；（2）生成音频；（3）生成图像；（4）生成视频；（5）同步视频和音频。

2.1 工作流组件

剧本生成。 根据输入内容，多模态输入大语言模型会为视频生成剧本。我们通过提示词配合少样本示例（而非微调或训练）来生成场景级描述、角色和对话。剧本会描述每个场景中发言的角色以及镜头序列。例如，一个镜头同时呈现两个角色，另一个镜头则分别呈现每个角色。对于每个镜头，大语言模型都会生成口语文字稿。随着场景和文字稿的生成，下游阶段也会随之启动。

图像生成。 场景和角色描述会生成提示词，用于图像生成 [107]（文本到图像）。工作流会生成风格一致的远景镜头和特写镜头，作为基础图像，并通过多个模型和提示词（图像到图像）确保各镜头在风格、设计和场景设定上保持连贯。这还可以包括原始输入内容中的幻灯片、文本面板或图表等静态视觉素材。

音频生成。 对话通过使用不同声音配置的文本转语音（文本转语音）模型合成。在支持的情况下，我们会应用情绪和韵律控制。我们使用专用音频模型来生成非语言声音（例如环境音、音效和音乐）。

视频生成。 我们以基础图像作为关键帧，生成镜头级视频片段（图像到视频）。例如，一个 10 秒的镜头可能会展示一个角色说话、其他角色做出反应，同时与剧本时序和画面相匹配。

音视频同步。 对于每个镜头，我们使用唇形同步和镜头转场（V+A）来对齐视频和音频。将各场景中的所有镜头组合起来，最终得到一段连贯的播客视频，整合了所有多模态素材。

安全性。 AI生成内容会带来安全、伦理和法律风险。为确保负责任地生成内容，我们的工作流设置了多重保障措施，包括输入验证、质量过滤和生成后的合规性检查。

2.2 其他工作流

视频播客生成提供了一个通用模板，并可推广到许多多模态应用。表1展示了一些示例及其输入、多模态输出和计算特性。这些应用主要涉及修改大语言模型的输入（即提示设计）、添加风格修饰器（例如LoRA [55]），以及调整数据工作流。

对于短视频生成，大语言模型支持多模态输入（即视频理解），并识别可供复用或重新生成的关键片段（例如较短的场景）。对于电影生成，大语言模型会生成更长的电影剧本和视频，并注重叙事。讲座和幻灯片虚拟形象工作流使用结构化输入（例如章节或幻灯片），生成与基于文本的讲解同步的虚拟形象，并使用静态内容（例如幻灯片）。配音会生成翻译后的音频，并使用音视频同步。编辑会跳过大多数组件，并对视频生成进行条件控制（例如卡通风格）。这些示例表明，我们的工作流能够自然地扩展到广泛的多模态应用。

2.3 多模态工作流服务的目标

使用我们的工作流，以朴素方式在一台 8×A100 GPU服务器上生成一段10分钟播客（典型的 YouTube 视频长度为 [43]），分辨率为720p，需要8.3小时（比实时慢 50×），并消耗价值 ~$70 的 GPU运行时间。超过99%的时间和资源都消耗在视频和视频音频生成上。

实时。 大多数大规模多模态生成系统都以批处理模式运行，用户需要等待数分钟 [49, 97]。相比之下，我们的目标是实现实时播放：视频在数秒内开始播放，并持续流畅地播放。具体体现在：

表1。 多模态生成 工作流，包括示例输入、待生成的多模态输出以及特征 cs.

工作流	示例输入	生成输出	特征	Text	音频	图像	视频
播客 [152]	论文	角色讨论输入内容的视频播客	第2节	Low	Low	Low	High
短视频 [15]	电影	提取关键片段并生成高光视频	重型大语言模型	High	Low	–	Low
电影 [83]	高层次情节	包含角色、场景等内容的多场景电影...	长输出 [59]	Low	Low	Med	High
动画故事 [117]	故事板	包含角色和场景的全动画故事	风格 LoRA [55]	Low	Low	Low	High
讲座 [145]	教科书	包含教授和辅助视觉内容的视频	静态内容	Low	Low	Low	Med
幻灯片虚拟形象 [122]	幻灯片演示文稿	嵌入式演示者虚拟形象解说幻灯片	低分辨率	Low	Low	Low	Med
配音 [131, 150]	电视节目	翻译并实现口型同步，同时保留说话人	高级文本转语音 [86]	Low	Med	–	High
编辑 [124, 142]	视频	修改原始视频的组成部分（例如，风格）	重型视频到视频	–	–	–	High
视频聊天 [138]	机器人对话	角色说出自己在对话中的部分	短输出	Low	Low	Low	Med

• 首帧时间（*TTFF*）：从提交输入到显示首帧之间的延迟。较短的 *TTFF*能提升感知响应性。

• 帧间时间（*TBF*）：生成帧之间的间隔。稳定的 *TBF*可确保视频流传输流畅。

要以播放速度（墙钟时间每秒对应一秒视频）实现实时视频生成，我们需要：

$$\mathrm{TTFF}_{\mathrm{eff}} = \max\left(\mathrm{TTFF}, \overline{\mathrm{TBF}} \times \#\mathrm{frames} - \mathrm{video\ duration}\right)$$

例如，对于24帧/秒且TBF为50毫秒的10分钟视频，$\mathrm{TTFF}_{\mathrm{eff}}$为2分钟，即使首帧时间为30秒。

截止时间。 这也对特定内容必须就绪的时间施加了严格截止期限。例如，在24帧每秒（帧/秒）下，场景2的第172帧必须在7.2秒前就绪；若首帧时间为1秒，系统必须维持36毫秒的TBF，并在播放开始后放宽至42毫秒。

宽松的服务级别目标。 一些多模态生成请求具有更宽松的服务级别目标。例如，用户可能要求视频播客在上午8点前就绪。此类情况的截止时间宽松得多，使调度器具有更大的灵活性，并允许更有效地复用共享组件。

成本效率。大型多模态模型需要昂贵的硬件。我们力求尽量降低计算用量、最大化利用率，并考虑影响提供方与用户成本的数据中心约束（例如，功耗、散热）[120]。

输出质量。只要内容连贯且引人入胜，用户或许能够接受质量略有下降。目标是将质量保持在可接受的范围内质量。

2.4 单体集成模型与工作流

另一种方案是由单个 AI 模型端到端生成完整视频，但这需要高昂的训练成本，也会限制灵活性。因此，我们采用模块化设计，复用现有模型，从而更便于编排、自定义和升级。这样一来，我们就能快速构建新的应用（例如短视频和配音）。这种工作流让开发者无需重新训练即可升级组件（例如文本转语音或视频生成），并能迅速采用更新或效率更高的子模型。类似的模块化架构已广泛应用于生产系统 [71, 143]，也体现了模块化智能体 AI 流水线这一普遍趋势 [20]。

目前，还没有能够处理复杂、多阶段长视频生成的集成模型（无论开源还是闭源）（例如播客风格视频创作）。现有模型会根据提示词生成短片段（<25 秒的 SORA2 [97] 和 <8 秒的 Veo3.1 [49]），但不会协调对话、场景或模态。即使存在这样的单体模型，它仍会包含可解耦的子组件。例如，基于扩散的模型会包含：文本/图像编码器、注意力模块、扩散模块以及视频/音频解码器。与大语言模型的预填充/解码解耦 [100, 151]和分块 [3]类似，可以运用这些技术将视频生成划分为多个镜头（每个镜头都有各自的截止时间）。这实际上会将单体模型转变为一种工作流（类似于 VAE-DiT 解耦）[42]，并开放调度和资源分配的可调参数，无需在架构上进行模块化。尽管我们的见解针对的是模块化工作流，但也适用于单体模型。其主要局限在于无法灵活切换不同的专用模型（例如Flux [74] 对比 HiDream [19]）。

3 多模态模型特征分析

为了了解大规模实时多模态生成服务面临的系统级挑战，我们分析了播客视频生成工作流的关键组件，重点关注性能。我们着重研究图像和视频生成，这两者合计占 GPU 运行时间的 99% 以上，也是实时服务成本的主要来源。我们使用 Azure GPU 虚拟机；除非另有说明，否则使用配备 8 个 NVIDIA A100 GPU 的整节点虚拟机 [87]。

3.1 生成式多模态模型

我们使用图像、视频和语音任务的公开排行榜 [9–11]，评估了 Hugging Face [35], 上精选的一批预训练开源模型。

我们使用 Elo 评分 [17] 评估模型质量。这表明，开源模型（例如，Wan [132], 混元视频 [72]）与专有模型（例如，Veo3 [49],SeeDance [46]）具有竞争力。根据现有信息 [46],，性能相近的模型通常采用相似的架构基础。

表 2 按类别（例如，图像到视频或图像到视频）、架构和规模（以

表 2. 用于播客视频生成的生成模型。

类别	模型	架构	Size	发布日期
LLM	Llama3.2 [7]	Transformer	90B	09/2024
	Gemma3 [31]	Transformer	27B	03/2025
A2T	Whisper [105]	Transformer	1.5B	11/2023
TTS	Kokoro [52]	Transformer	82M	01/2025
	XTTS [27]	Transformer	400M	2023年09月
	OpenAudio [78]	Transformer	500M	2024年11月
	Dia [76]	Transformer	1.6B	2025年5月
	VibeVoice [86]	Transformer	7B	09/2025
T2I	SD3.5 [8]	DiT	8.1B	03/2024
	Flux [74]	DiT	12B	08/2024
	HiDream-I1 [19]	DiT	17B	2025年4月
	Janus Pro [21]	Transformer	7B	01/2025
I2I	YOLO [126]	CNN	68M	2016年8月
	Real-ESRGAN [134]	CNN	16M	2021年8月
	FLUX Kontext [75]	DiT	12B	2025年7月
	Bagel [32]	MoT	14B	2025年5月
	HiDream-E1 [19]	DiT	17B	2025年7月
I2V	Wan 2.1 [132]	DiT	14B	2025年2月
	Wan 2.2 [132]	DiT+MoE	14B	2025年7月
	混元视频 [72]	DiT	13B	2024年12月
	FramePack [148]	DiT	13B	2025年4月
	LTX-Video [51]	DiT	13B	2024年12月
V+A	FantasyTalking [133]	DiT	1.2B	2025年4月
	混元头像 [22]	DiT	13B	2025年5月
	Sonic [63]	DiT	1.1B	2024年12月

FP16）。这表明该领域正在迅速发展，过去三个月已有 10 多个新模型发布。

大语言模型。 这些模型通常采用 Transformer [127]，其性能特征已得到充分研究 [100,101, 129]。推理过程包括计算密集型的预填充阶段，用于处理输入词元；随后进入受内存限制的解码阶段，逐个生成输出词元。例如，在单个 A100 GPU 上运行 Gemma3，每秒可处理超过 7000 个输入词元，并每 40 毫秒生成一个输出词元。为了扩展到多个 GPU 和服务器，大语言模型通常采用张量并行和流水线并行。

文本转语音。 现代文本转语音模型 [37] 也经常采用基于 Transformer 的架构 [127]。典型模型拥有数亿个参数，所需内存不足 10 GB，因此相对轻量。Kokoro [52], 是一种成本高效的文本转语音模型，在单个 A100 GPU 上生成一秒音频耗时不到 1 毫秒，其延迟与输入词元数量成正比（与输出时长大致呈线性关系）。它也可以在 CPU 上运行，但生成一秒音频需要 500 毫秒。较大的模型（例如Dia [76] 和 OpenAudio [78]）成本更高，运行时间会随模型规模线性增长。可以使用序列并行 [77]，将它们分配到多个 GPU 上并行运行。

图像和视频生成。图2展示了这些模型的标准架构 [36, 38], , 这些模型基于解码器——
加噪扩散 [54]。它们通常使用扩散变换器

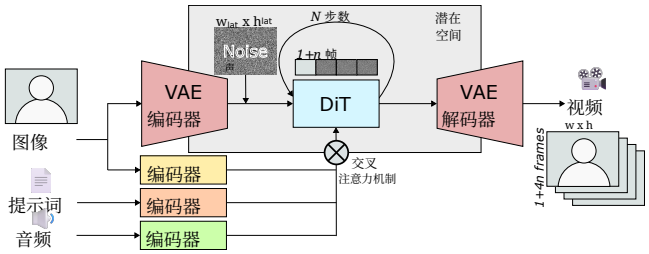

图2. 采用扩散 Transformer 和变分自编码器的视频生成架构，输入包括文本、图像和音频。

(DiT) 架构 [112],，结合变分自编码器（VAE）与文本/图像编码器 [70]。像 Wan 2.2 [132] 这样的模型采用混合专家（MoE）DiT 架构。

变分自编码器将起始帧编码为潜在噪声。大多数变分自编码器会将输入的空间分辨率压缩a 8× 倍，并将通道数从 3（RGB）扩展到 16。输入（文本、图像或音频）经过编码后，通过交叉注意力机制馈入 Transformer。DiT 会迭代地对潜在空间进行去噪，直至其接近目标输出。这些模型通常采用无分类器引导（CFG） [33],，通过在有条件和无条件情况下运行 DiT，来平衡保真度与多样性。最后，变分自编码器将潜在表示解码为最终图像/视频。

文本生成图像（*T2I*）。表现出色的开源模型 [9]，例如 FLUX [74] 和 HiDream-I1 [19]，采用 DiT 主干网络（FP16 下为 20–30 GB），根据文本提示词生成高质量图像。它们通常使用 T5 [106] 或 Llama3 等强大的文本编码器，并依靠密集注意力机制捕捉细粒度的视觉细节。

图像生成图像（*I2I*）。DiT 模型将输入图像编码为词元，并在提示词或控制信号的条件下将其转换到潜在空间中。它们使用常见的图像编码器（例如，CLIP [104]）。在这种情况下，大多数变分自编码器会将帧数压缩为原来的 1/4（第一帧保持不压缩），因此对于 1+80 帧，会将其压缩为 21 个潜在帧。示例包括 InstantCharacter [125], FluxKon-text [75], 和 HiDream-E1 [19]。ControlNet [6, 147] 等轻量级模块（例如，使用 OpenPose 或 Canny边缘）以及 IP-Adapter [139] 会将外部信号注入扩散过程中。例如，Bagel [32], 使用混合专家变换器（MoT）执行结构感知变换。

对于基本的图像生成，我们可以使用 YOLO [126] 等更便宜的模型，这些模型基于卷积神经网络（CNN）。这些较小的模型能够识别角色并生成放大的图像。

文本/图像生成视频（*T2V*和*I2V*）。这些模型 [38] 与图像生成模型采用相同的架构，但会对多帧而非单张图像进行去噪。例如，Wan2.1 [132]可以同时生成多达 1+80 帧（1帧起始帧和80帧生成帧），以每秒16帧的速度播放时长为5.1秒。Stable Video Diffusion [16] 和混元视频 [72] 也采用了类似的实现，并且经过训练以适应

特定的帧率。例如，Wan 使用每秒16帧（FPS），而混元视频使用30 FPS。
分辨率。 由于计算成本会随着像素数量增加而上升，直接生成高清内容的成本可能高得难以承受。为此，可以使用轻量级的升采样模型 [134] 来提升分辨率和时间一致性。LTX [51]采用了这一策略。

视频时长。 由于训练和推理过程都受到内存限制，大多数模型只能生成较短的视频片段（通常为 <5 秒）。一种常见的变通方法是先生成允许的最大帧数，再将末尾的几帧作为输入，用于生成下一个片段。然而，这可能会破坏时间连续性并导致视觉漂移。基于混元视频 [72], 构建的 FramePack [148], 通过在潜在空间中压缩不太重要的帧来解决这一问题，从而能够对整个视频进行注意力计算。

其他模型架构。 还有一些模型采用传统的 Transformer架构，例如 Janus [21] 和 Llama-Gen [123]。这些模型一次生成一个词元，每个词元代表图像的一个图像块。目前，这些模型既不流行 [35] 也未完全成熟；其输出质量较低 [9], ，而性能则与其他模型相当。

视频与音频同步。Sonic [63]和 RealTalk [64] 等模型专门针对视频音频同步进行训练。其他模型则添加交叉注意力层，以扩展常规视频生成模型并使其与音频对齐。例如，FantasyTalking [133] 扩展了 Wan [132]，而混元视频数字人 [22] 扩展了混元视频 [72]。它们的推理时间与普通视频生成模型相近（随视频时长、去噪步数和视频分辨率线性增长）。

3.2 性能分解

请求服务延迟。我们重点关注基于扩散的图像转视频生成（即 Wan 2.1），因为它的计算开销最大，且涵盖了其他模型。我们将在讨论中重点说明与其他模型的差异。

图 3 显示，在一块 A100 GPU 上生成时长为 ~5.1 秒的视频（以 16 帧/秒生成 + 80 帧）需要 ~93 秒（每生成一秒视频需要 ~18 秒）。其中大部分时间用于扩散（有条件和无条件 DiT），部分时间用于变分自编码器的编码/解码，而文本和图像编码的耗时相对较少。因此，
文本生成视频的延迟与图像到视频相近。

供参考，Veo 3 [49]和 Sora [97] 等商业解决方案生成 8 秒视频，较便宜的套餐需要 2–4 分钟，较贵的套餐则不到 60 秒（每生成 1 秒需 7–30 秒） [62]。

视频音频同步（例如，Fantasy Talking）会为基础模型（例如，Wan 2.1）添加音频编码和一个交叉注意力层 [103] 。音频编码甚至比文本或图像编码更快，而且新增的交叉注意力层对性能的影响可以忽略不计。

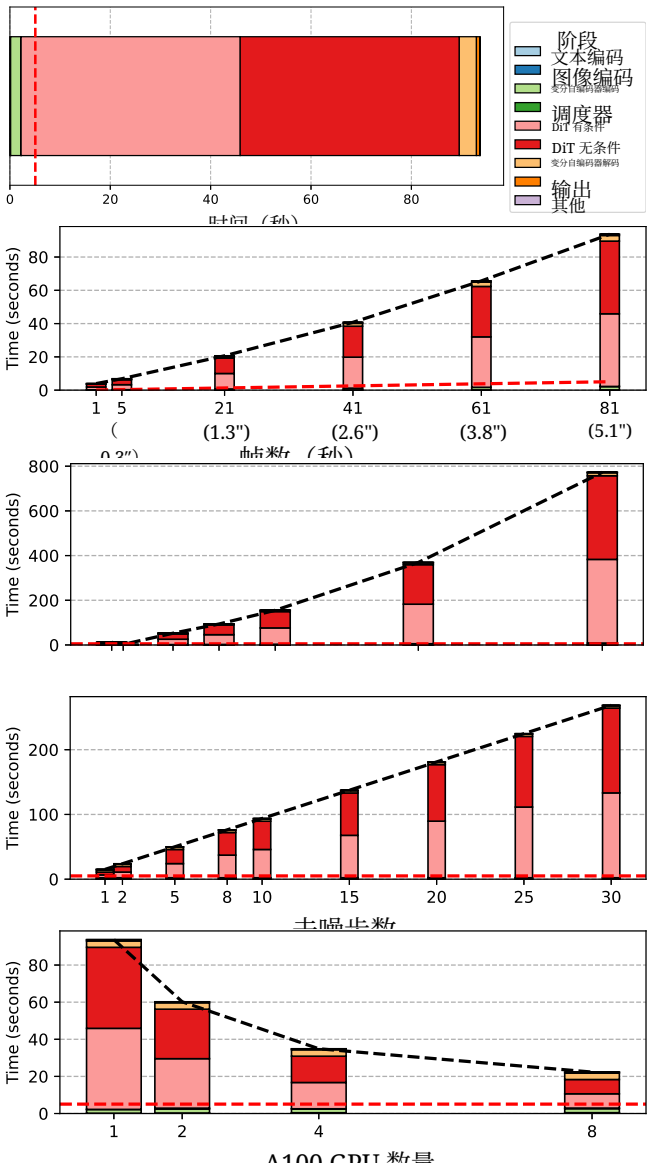

图 3. 使用 A100 GPU 和 10 个扩散步数生成一段 81 帧（以 16 帧/秒播放时长为 5.1 秒）、宽高比为 16:10 的视频所需的延迟。图中还展示了对这些参数中每一项的敏感性。红色虚线表示真实时间。

帧。 图 3 显示，推理时间与帧数几乎呈线性关系。生成 1 帧（62.5 毫秒）需要 ~66 秒/秒，生成 1.3 秒的视频需要 ~23，生成 5.1 秒的视频需要 ~18 秒/秒。生成较长的视频效率略高。生成单帧视频的延迟与图像生成相当。

分辨率。 由于变分自编码器通常采用 8:1 的压缩比（即，每 8 个输入像素对应 1 个潜像素），因此并非所有分辨率都可用。例如，16:9 宽高比下的 480p（854×480）宽度无法被 8 整除，需要进行填充或裁剪才能适配模型。相比之下，

16:10 或 5:4 宽高比的分辨率通常更容易与变分自编码器8×的下采样对齐，因此更为适合。

图 3 显示，延迟会随着像素数量（宽度 × 高度）的增加而上升。分辨率从 640×400提升到 1280×800，像素数量会增加 4×，延迟也会相应提高约 4×。请注意，较高的分辨率需要更多去噪步数才能达到相近的质量。

步数。 图 3 显示，DiT 的延迟会随去噪步数线性增加。步数越多，输出质量越高。仅使用 1–2 步时，图像质量较差；但在约 5 步时，物体便变得可辨认。大多数图像在 10 步时已十分清晰，并在 20–30 步时达到峰值质量。

GPU 数量。 为降低延迟，我们可以让这些模型在多个 GPU 上运行。并行化基于 DiT 的模型的一种常用方法是统一序列并行（USP）[41],，它结合 Ulysses [61]和环形注意力 [80]，将潜在空间分布到多个 GPU 上。如果模型采用无分类器引导（CFG），我们还可以进一步在多个 GPU 上并行执行有条件与无条件 DiT 前向过程。

图 3 显示，在单台服务器上使用多个 GPU 进行 USP，可显著缩短 DiT 的执行时间。请注意，USP 仅适用于 DiT 阶段，变分自编码器阶段保持不变。使用 8 个 GPU，与单 GPU 执行相比，DiT 时间缩短了超过 5×。

批处理。 为提高效率，可以将多个请求批处理并一同处理 [73, 141]。这种方法对图像与文本编码效果很好，因为它们几乎可以完美扩展。然而，DiT 已接近计算饱和，而变分自编码器则已完全饱和，几乎没有提升空间。将四个请求批处理后，DiT 效率提升不到 5%；对于变分自编码器，批处理会因执行开销而增加运行时间。

模型加载时间。 大规模提供模型服务时，必须考虑 GPU 加载时间。例如，拥有 14B 个参数的 Wan，其图像大小为 ~90 GB。将其 FP16 权重加载到 GPU 需要 ~30 秒，而第一个预热请求（触发编译）则需要 ~80 秒。加载完成后，该模型会占用 ~48GB 的 GPU 内存。

加载时间随模型规模而变化。例如，Flux（12B 参数）需要 ~10 秒加载。不过，它需要~3 分钟预热，并占用 ~33GB 图形处理器内存。

3.3 硬件的影响

这些模型的性能在很大程度上取决于底层硬件。我们在四代图形处理器上对它们进行了基准测试，并以 CPU 作为参考（表3）。

类型与生成。 我们重点关注英伟达图形处理器，因为它们供应更广泛，软件生态也更成熟。尽管其他加速器（例如AMD MI300 或谷歌张量处理单元）理论上也能支持这些模型，但由于软件支持不佳，实际应用仍然有限。另一方面，CPU运行中等规模模型的速度太慢，效率不足。它们可以运行Kokoro或YOLO等规模更小、成本更低的模型，但速度最多会降低 60×。

表3. 硬件、每小时价格和功耗。

Type	模型	Year	#GPU	预留	Spot	TDP
CPU	英特尔 EMR	2024	-	$2.33	$0.83	350 瓦
GPU	V100 [90]	2017	8	10.79 美元	3.97 美元	8×300 瓦
GPU	A100 [91]	2020	8	14.42 美元	8.52 美元	8×400 周
GPU	小时100 [92]	2022	8	43.16 美元	32.22 美元	8×700 瓦
GPU	高200 [92]	2024	8	$45.22	$33.76	8×700 瓦
GPU	GB200 [94]	2025	4	$57.67	$43.04	4×1200 瓦

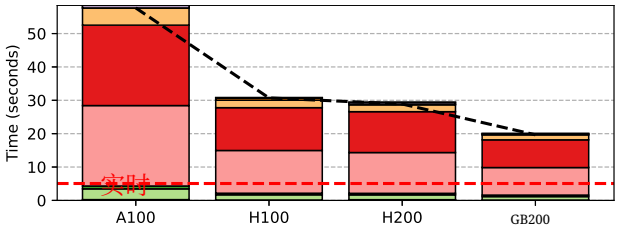

图 4. 在单台服务器中配备 4 个图形处理器时，对英伟达图形处理器代际的延迟敏感性（限制为与 GB200 相匹配）。

可用性。 GPU需求旺盛，且受到区域限制。在某云服务提供商的60个主要区域中，只有1个区域提供V100，而A100、H100和H200分别在18、14和5个区域可用；目前GB200则普遍无法使用。即使在这些区域，配额也受到严格限制。鉴于这些限制，我们将在后文探讨混用多个区域的不同GPU类型。

特性。 大多数基于DiT的模型（包括 Flux 和 Wan）都采用 FlashAttention [29, 30, 116], 这一系列针对英伟达 Ampere [91]及更新一代图形处理器优化的注意力内核。这些优化带来了超过 20× 的性能提升，但与 V100级硬件不兼容 [90]。具体而言，FlashAttention-3 专为 Hopper 及更新架构设计 [116],，可进一步提升速度。

量化。 现代图形处理器包含专用于 FP32 以外数据类型的计算单元，例如 FP16、FP8（Hopper+ [92]）和 FP4（Blackwell+ [94]）。大多数开源模型使用 FP16 权重，不过一些非官方实现支持 FP8 [25]。量化可减少图形处理器内存占用并降低延迟 [108]。例如，在 H100 图形处理器上使用 FP8 运行 Flux，可将推理延迟降低 30–40%。

延迟。图 4 显示，将硬件从 A100 升级到 H100 可使 Wan 2.1 的延迟降低 ~1.9×。H200 凭借更高的内存带宽和容量，进一步提升了 5% 的性能。与 A100 相比，GB200 可进一步降低延迟 ~2.9×（请注意，我们没有针对 Blackwell 使用特定优化，而是采用默认的 FP16 ）。这些改进惠及所有阶段，尤其是时间最紧迫的阶段：DiT 去噪步骤以及 VAE 编码/解码。模型加载时间也有所缩短。

成本。表 3 列出了三大云服务提供商的最佳图形处理器价格：EC2 [114], Azure [14], 和 GCP [23],，并通过第三方聚合平台 [24]进行验证。所有价格

均对应最长的可用预留期限（例如，3 年）。作为更便宜的选择，我们还列出了竞价型虚拟机的价格。价格因地区而异，差异显著（通常为 >2×）。

A100 GPU 的成本效率比 H100 高 ~60%，比 GB200 高 ~2.7× 。在以低成本实现实时服务时，这种权衡十分重要。我们还需要将模型加载时间和整个系统中的 GPU 空闲时间等固定成本纳入考量。作为比较，Veo 3 [49] 等商用视频生成模型，以 720p [62]、24 帧/秒生成最长 8 秒的视频，费用约为 0.20 美元。

能耗。 与 A100 相比，H100 和 H200 分别将总能耗降低了 7% 和 12%，而 GB200 的能耗则增加了 2%，尽管其速度几乎快了 3× 。请注意，这里仅考虑了 GPU，而未计入系统的其他部分（CPU、内存、NVLink 等）。这表明，性能提升本身并不会直接转化为能耗节省，尤其是因为功耗未必会随代际变化而线性缩放。这些趋势说明，频率调节是一种切实可行的机制，能够在不牺牲实时延迟的情况下提高能源效率。

频率调节。为完整起见，我们分析了 GPU 频率调节对 A100 的影响，将频率从 200 MHz 调至最高 1.4 GHz。DiT 和变分自编码器阶段都表现出很高的敏感性：频率降低 15% 会使运行时间增加 8%，而更大幅度地限制频率（降低 45%）则会导致运行速度下降 52%。

A100 GPU 的热设计功耗（TDP）为 400W。功耗大致随频率的平方变化，范围从空闲时的 63W 到峰值时的 400W。频率降低 15% 会使峰值功耗降低 23%，降低 45 % 则会使其降低 48%。在最高频率下，平均功耗与峰值相差不到 10%，这表明 GPU 持续保持高利用率，几乎没有空闲时间。能效最高的频率范围为 800–1000MHz，可将能耗降低 20% 以上。其他 GPU 世代在按热设计功耗归一化后也呈现出类似趋势（表3）。

GPU 温度与功率使用密切相关，并会随着时间推移逐渐稳定。我们观察到 GPU 间热差异：由于物理位置和制造工艺不同，一些 GPU 的运行温度始终较高 [120]。

3.4 跨多台服务器的分布式推理

尽管有所改进，延迟仍低于实时要求。我们利用 InfiniBand [13] 连接性，以便在单台服务器之外进行并行化 [100]。图 5 展示了延迟如何从配备 8×H200 的一台服务器扩展到配备 80 个 GPU 的 10 台服务器。使用五台 H200 服务器（40 个 GPU），并对变分自编码器阶段进行流水线处理，即可实现实时 DiT 性能。然而，效率仍然很低：要实现低于 18×延迟的加速比，需要 40× 更多资源（即，从 1 个 H200 GPU 增至 40 个）。

并行性限制。USP 同时受模型架构和输入规模的限制。例如，Ulysses [41] 沿注意力头进行并行化（例如，Wan 中有 40 个），这使其

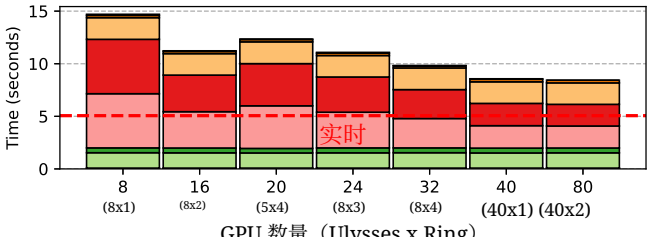

图 5. 使用 USP、Ulysses 和 Ring 注意力机制时，H200 GPU 数量及其分布在不同服务器上的敏感性。

与使用两台 8×GPU服务器等配置不兼容。此外，序列长度（与输出分辨率相关）必须满足整除约束，以避免填充或裁剪。它必须能被注意力头数量和 GPU 数量整除。结合变分自编码器压缩（例如， 8×），这些约束缩小了有效的 16:9 分辨率范围。为提高调度灵活性，我们采用 16:10（8:5）和 5:4 等替代宽高比。

4 多模态生成服务

为了实时服务我们的视频播客生成工作流，我们提出了潜在的系统策略，并设计了一种端到端架构，优化多模态生成的各个阶段，以平衡延迟、成本和输出质量。

4.1 提升效率的系统机遇

基于工作流的需求及其基础多模态模型的特性，我们概述了系统层面的策略，以平衡延迟、成本和质量。

降低延迟。要实时生成多模态内容，最大限度地降低延迟至关重要。

截止时间感知调度。严格的截止期限（尤其是早期镜头）要求优先处理对延迟敏感的片段。截止时间感知调度会预先为这些片段分配资源，同时放宽对后续片段的限制，从而确保响应及时。

并行性。 为了降低延迟，我们可以在多个层面利用多台 GPU：（1）独立的阶段 （例如，音频和图像生成）可以在前序阶段（例如，剧本生成）完成后并行运行；（2）场景和镜头 可以独立生成，从而支持在服务器之间进行空间和时间分配；（3）大型模型 可通过序列 [89], 张量 [118], 和流水线并行实现横向扩展 [58]。

解耦。 为了实现流水线处理并降低延迟，可以将模型组件解耦 [100, 151]。例如，将 DiT 和变分自编码器阶段分离后，即使 DiT 仍在进行去噪，也可以在潜在空间中开始解码。这种重叠处理可提升吞吐量和响应速度。

成本最小化。 我们的工作流提供了灵活性，可降低成本并提高资源效率。

硬件感知。 轻量级任务可以在 CPU 或较旧的 GPU 上运行，而延迟关键阶段则使用较新的 GPU。

组件可以独立扩展，并且可以为时间敏感片段选择更新的硬件。

细粒度扩展。模型可以根据工作负载需求独立预配。此外，解耦使组件能够独立扩展，从而可以对延迟敏感路径的资源分配进行精确控制。例如，我们可以为 I2V DiT 预配 16 个实例、为 I2V VAE 预配 2 个，为文本转语音预配 1 个。

竞价实例和多区域部署。 为进一步降低成本，我们可以积极使用 Spot资源，并通过超额配置 [140]来降低驱逐风险。为了利用多个区域中可用的资源，我们可以将这些资源汇总起来，并据此编排工作流 [65, 82], , 同时考虑带宽和延迟约束。

多路复用。 同时为多个用户提供服务可以提高资源利用率，并摊销固定成本，从而进一步提升系统整体效率。批处理可以提高吞吐量，但也可能增加延迟 [141]。经过微调的批处理策略可以在不让计算密集型组件过载的情况下，兼顾响应能力和效率。

频率调优。 调整 GPU 频率可使调度兼顾功耗、能耗和热限制。这降低了基础设施需求，并减少托管成本。

质量。 为平衡响应速度、用户体验和成本，我们还可以调节输出质量。

配置选择。 在每个阶段选择合适的模型和配置（例如，去噪步数、分辨率）至关重要。高质量设置可预留给关键或视觉上突出的镜头，而较轻量的配置则在其他地方维持实时性能。

自适应质量。 早期镜头可采用较低保真度的模型或降低分辨率以满足截止时间，并在资源允许时提高保真度。这使得系统能够根据系统负载和用户期望，实时调整脚本长度、分辨率和模型复杂度。

静态内容。预先制作的片头、背景或插值画面可以掩盖延迟、降低负载并提高响应速度。在时间关键片段中，静态元素（例如，幻灯片、广告）可吸收延迟，而轻量叠加层则实现低成本个性化。

4.2 StreamWise 概述

我们提出 StreamWise，一种面向规模化多模态内容生成服务的端到端解决方案。它旨在支持高效的实时流式传输，并应对独特的
通过借鉴前几节概述的挑战与机遇，满足这些工作负载的需求。

StreamWise 建立在广为人知的技术之上（例如，有向无环图调度、解耦），但其创新之处在于针对实时、基于扩散的多模态生成工作流对这些技术进行调整和编排。这类工作负载比以往的大语言模型

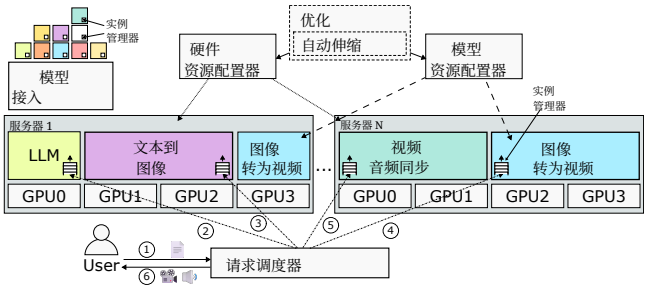

图 6。 StreamWise 的架构概览，展示了一个跨四个模型实例进行请求调度的示例。

服务优化所依赖的预填充和解码两阶段工作流更加复杂、计算密集，且对延迟更加敏感 [65, 82, 100, 151]）。

图 6 展示了 StreamWise 的高层架构，其中包括：(1) 接入多模态生成模型，(2) 在集群中预配资源（即服务器和 GPU 的数量）及模型（即两个 Flux 副本和五个 Fantasy Talking 副本），(3) 在模型实例之间调度和编排多模态生成请求，以满足延迟要求（例如实时处理），以及 (4) 在本地模型实例中执行请求。

4.3 多模态模型接入

每周都会发布性能与成本权衡各异的新模型 [19, 52, 74, 75, 135]（表 2）。StreamWise 支持快速接入：许可和使用限制会预先处理；模型完成打包后，只需极少的工作即可部署并提供服务。为实现快速集成和接入，我们将每个模型与实例管理器 一同打包，该管理器提供标准的远程接口。

特征。 在接入过程中，我们会在元数据文件中描述模型的特征，供 StreamWise 后续进行模型选择。这些特征包括质量（Elo排名）、帧率（帧/秒）、最大帧数（视频长度）、注意力头数量、VAE压缩比、支持的分辨率以及其他相关属性。

性能剖析。 我们会生成简单的模型画像，以估算运行时间和资源使用，因为关键参数（例如像素数、帧数）会按比例变化。我们会对具有代表性的配置进行基准测试（例如，1+16帧、10 步、640×400 分辨率），并通过其他测试点进行验证。我们还会测量峰值功耗、能耗和温度。这些数据可用于构建预测模型，估算不同配置下的性能、成本和质量。

4.4 硬件和模型预配

我们将为某项工作负载（例如，一段时长 10 分钟、质量中等的视频播客）选择硬件和模型视为一个优化问题。选定配置后，硬件和模型配置器会相应地处理设置。

自动伸缩。为适应不断变化的需求，StreamWise 会定期运行优化。我们从当前配置出发，并计入预配资源的成本以及加载模型所需的额外时间。这样，StreamWise 可以在白天横向扩展至十台 H200 服务器，夜间则缩减至一台 A100 服务器。

优化流程。 它分为两个阶段：

初始配置。 我们从经济高效的基线配置开始，采用价格低廉的模型和成本较低的 GPU。每个模型实例（例如，用于图像生成的 Flux、用于视频生成的混元 FramePack）分配一块 GPU（例如，A100）。贪心算法会模拟用户请求在此配置下的处理过程。它将请求表示为有向无环图（DAG），并优先沿关键路径将节点分配给可用资源。延迟和成本估算基于接入过程中生成的模型画像。

迭代优化。 接下来，我们通过系统地探索延迟-成本权衡空间来改进初始配置。对于每种设置，我们都使用贪心算法估算延迟和成本。如果某个方案不可行（例如，没有图像生成模型），就会将其舍弃。此优化过程包括：(1) 添加或移除硬件资源（包括竞价实例）；(2) 切换 GPU 类型；(3) 为任务切换模型；(3) 调整模型实例数量；以及 (4) 修改每个实例的 GPU 分配（即模型并行）。我们还加入领域特定启发式方法来引导演进。如果成本超出预算，就切换到竞价虚拟机并缩减虚拟机规模。如果延迟过高，就扩容并尝试更快的 GPU。

优化目标。 我们将成本 × 首帧时间（$ ×秒）降至最低。针对特定服务级别目标（例如，首帧时间为 10 秒的实时场景），我们会引导演化过程朝着满足这些目标的配置发展。如果找不到可行解，则返回最接近的解。在探索过程中，我们会在延迟与成本之间的帕累托前沿上进行搜索。默认情况下，系统会平衡这两个目标，但可以调整此行为以优先考虑其中一个目标。我们还支持针对能耗、首帧时间及其他组合进行优化（例如，能耗 × 首帧时间）。

优化扩展。 为了支持多种高级改进，我们扩展了基本优化框架。

质量。 StreamWise 可以选择直接生成高质量视频，也可以先生成较低分辨率的版本，再使用单独的模型进行超分辨率放大。例如，我们可以以 640×400 运行 Fantasy Talking，然后使用 Real-ESRGAN 将其放大至 1280×800。

解耦。我们可以将 DiT 和变分自编码器等计算密集型模型组件解耦为独立组件。FramePack DiT 将潜在输出流式传输至变分自编码器进行解码，随后再传递给音频同步模型（例如，图 7）。这使流水线执行和独立

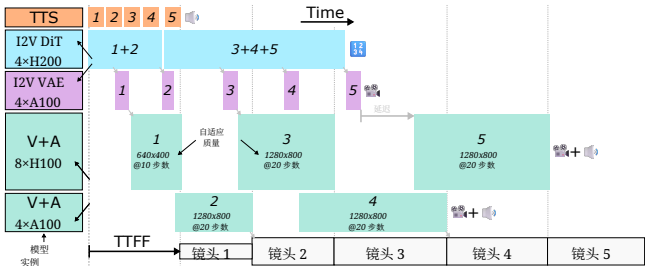

图 7。 在 A100、H100 和 H200 上，对解耦组件采用截止时间感知调度和自适应质量。

扩展以及细粒度资源分配成为可能。解耦后，每个组件都表示为一个独立的有向无环图节点，供预配和优化算法使用。

竞价实例。 这些实例可显著降低成本（例如最高可达 50 %），因此优化过程自然会倾向于使用它们。不过，这些实例可能会被驱逐。我们根据驱逐风险按比例增加分配的资源数量，从而确保成本能够反映这一风险，在经济性与可靠性之间取得平衡。

多区域。 由于图形处理器在不同区域的可用性会随时间变化，StreamWise 会持续监控并汇总这些可用性。在计算生成延迟和成本时，我们会同时考虑区域间延迟和带宽。例如，小型图像传输可以容忍区域间延迟，但出于性能考虑，DiT 和变分自编码器等组件应部署在同一区域。

4.5 跨模型实例调度请求

硬件和模型配置完成后，请求调度器 会依据请求DAG，采用我们贪心算法的在线迭代版本来编排执行。

多个请求。遵循 YARN 理念 [128]，每个用户请求都由一个请求调度器管理。为协调多个请求，模型实例会维护本地队列，按截止时间为任务排序。例如，如果新请求中较早场景的截止时间更紧，图像生成模型就可能先处理该场景，而不是较早请求中较晚的场景。调度器 会监控这些队列，并在资源可用时重新调度。

DAG生成。DAG的大部分内容都是在运行时间生成的。例如，在剧本生成之前，我们并不知道场景和镜头的数量。我们从DAG的草图开始（例如，10分钟视频，每个镜头持续30秒），并随着各阶段的生成不断更新DAG。

截止时间。 请求调度器 根据服务等级目标和预期运行时间，为每个有向无环图节点计算截止时间。例如，首帧时间为 5 秒、时长为 10 分钟的实时视频播客，会将最终节点的截止时间设为$t_{now} + 605$，并递归地调度其依赖节点。

执行从无依赖节点开始，优先处理截止时间较早的节点。节点完成后，会触发其依赖节点。例如，剧本生成先于音频和图像生成；由于图像的截止时间更紧，因此优先处理图像。请求调度器 还会优先处理较早的场景和镜头（图 7）。每次提交都会附上截止时间，以便进行细粒度的实例级调度。

实例选择。 每个有向无环图节点都会分配给预期运行时间最短的模型实例（例如，图 7 中的 8×H200 与4×A100）。轻量级任务（例如，提示词过滤、安全检查）通常会分配给成本较低的 GPU 或 CPU，而早期阶段的任务则可能使用高端实例。如果没有可用实例，请求调度器 会将请求排入队列，并在资源释放后重新调度。

自适应质量。 调度从目标质量开始（例如， 1280×800，20 个去噪步数），如果可能无法满足截止时间，则逐步降低质量（例如， 640×400，图 7 中的 10 步）。初始阶段和中间阶段可以采用较低质量运行，而最终输出则以更高分辨率渲染。如果仍然不够，我们就切换到静态内容。

模型约束。 我们的贪心算法会考虑特定于模型的限制（例如，最大生成长度）。例如，Fantasy Talking 在发生漂移前，最多支持生成 3.5 秒、帧率为 23 帧/秒的视频/音频。为了处理更长的镜头，我们先用 FramePack 以中等质量生成完整片段（例如，30 秒），再根据语音停顿对其分段，并使用 Fantasy Talking 重新同步音频。我们还会考虑分辨率限制，以确保宽高比合适。

驱逐与故障。竞价实例通常会提前 30 秒发出驱逐通知 [140]。收到通知后，我们会停止向受影响资源上的实例发送新请求。对于其他故障，我们会监控模型实例的运行状态，避免向无响应的实例发送请求。在故障资源上运行的请求会重新提交。

缓存。StreamWise 支持在不同请求之间复用中间结果，以提高效率。常用资源（例如静态背景、文本/图像嵌入或先前生成的片段）会被缓存并重复使用。此功能还可扩展，以支持扩散感知缓存方法（例如AdaCache [67], NIRVANA [2], MoDM [136]）。

4.6 在模型实例中执行请求

每个模型都配备一个本地实例管理器，它提供标准化的 REST 接口，并触发本地模型推理。该管理器还负责请求批处理，并调整 GPU频率以优化资源使用。

批处理。 传入的请求会被放入本地队列，以进行批处理 和细粒度调度 [141],，并依据截止时间调整顺序。StreamWise 聚合请求以提高吞吐量，但收益有限：DiT 和变分自编码器等计算密集型阶段已使 GPU 达到饱和，几乎没有空间用于并行性。较轻量的组件（例如文本/图像编码器）会更频繁地进行批处理，但收益并不显著。

频率管理。为满足数据中心基础设施的限制，实例管理器可以降低 GPU 频率并引入短暂延迟，以减少：(1) 峰值功率 [101, 120], ；(2) 能耗 [121], ； 以及 (3) 热负载 [120]。

4.7 实现

我们基于 Kubernetes (K8s) 集群 [18]构建了 StreamWise：这是一种广泛采用的集群管理器，支持模块化部署、自动伸缩、服务发现和容错。所有控制器均使用 Python 实现，我们的实现代码和部署脚本将开源。StreamWise 通过硬件与模型配置器、实例管理器以及执行器扩展了 Kubernetes；执行器负责定义、调度和编排多模态实时生成工作流。

替代方案。StreamWise 也可以在 Ray [88], 、YARN [128], 或 Knative [69]等平台上实现。这些平台的默认配置使用不设截止时间的先进先出调度器。我们的调度器在类似 Ray 和 YARN 的机制（例如本地队列、任务窃取）之上增加了截止时间感知能力。在工作流规范方面，可以使用 ComfyUI [26], 等工具，但这些工具仍需要手动拆分模型组件。我们选择 Kubernetes 是因为它简单灵活，同时仍保留了与其他系统相同的基本执行模型。

模型接入。 我们将表 2 中的每个模型打包为 Docker [84] 容器，容器基于配备图形处理器驱动程序和运行时工具 [96]的英伟达镜像。每个容器都内置了我们的实例管理器，用于统一执行推理请求的接口。我们将现有推理代码（通常来自 Hugging Face）适配到该接口，并将其与模型权重打包在一起。使用 Diffusers [130] 的模型可在 <30 分钟内完成接入；其他模型可能需要 1–2 小时。

所有模型都在 PyTorch [99], 上运行，并可选择使用量化 [108] 和 TeaCache [79]等优化。对于大语言模型（例如 Llama、Gemma），我们使用搭载 vLLM 0.9.1 [73] 的 Docker 镜像，并通过 OpenAI API [98] 提供服务。已接入模型的元数据由一个集中式 JSON 文件维护。

并行性。许多扩散模型原生支持多 GPU 推理（例如 Wan [132]）。对于不支持的模型，我们使用来自 xDiT [40]的 USP [41] 。我们已为四个模型启用并行性（例如 Fantasy Talking、混元 FramePack），每个模型所需的工作时间都不到两小时。xfuser [40] 代码库提供了相关示例，而基于大语言模型的编程智能体可以简化这一流程。这与添加到 SGLang [115]中的扩散优化类似。

性能剖析。 我们使用 Scikit-learn [102] 拟合线性模型。我们的运行时间和成本概况的准确率超过 99.9%。

质量。 接入模型时，StreamWise 会使用公开排行榜中的 Elo 排名 [9, 11]。

硬件配置器。 我们使用虚拟机作为 K8s 集群的底层资源池。硬件配置器通过标准 API 与云服务提供商（例如，EC 2 [114], Azure [14]）交互，以便向集群添加或移除虚拟机。它支持异构硬件（例如，V100、A100、H100、H200）和竞价虚拟机，这些资源均通过虚拟网络互连。在适用的情况下，我们会配置 InfiniBand [13]和 NCCL [28]，并设置网络对等连接以实现跨区域。我们将所有这些特性（图形处理器、区域、可驱逐）公开为 K8s 节点标签，以便通过亲和性规则用于 Pod 调度。

模型配置器。 配置优化在 ~90 毫秒内完成，并输出待部署的模型实例（例如，在 8× 个 H100 和 2× 个 A100 上部署两个 FantasyTalking 实例）。资源配置器使用 K8s 接口在指定硬件上启动这些实例。此过程可能需要数分钟，因为它涉及拉取 Docker 镜像、加载模型权重以及执行预热运行。为降低启动延迟，我们在每个区域缓存常用镜像。模型到硬件的映射通过 K8s Pod 亲和性规则进行管理。我们还支持部分图形处理器部署，允许轻量级模型（例如，Kokoro）使用 MPS [93] 和 MIG [95] 共享一块图形处理器。

实例管理器。 它通过 Quart [66] 提供简单的 HTTP REST 接口。它实现了一个截止时间感知请求队列，用于本地批处理，从而支持在多个请求调度器之间进行全局协调。该管理器还通过英伟达接口设置图形处理器频率。

StreamCast。 我们将播客视频生成实现为一项 K8s 服务：StreamCast。它通过 REST API [66] 向用户提供接口，以便提交请求（例如，“以中等质量为这篇论文生成一个 10分钟视频”）。它根据各个阶段、场景和镜头构建工作流有向无环图（例如，场景 2 中镜头 3 的音频）。它从剧本节点开始，随着大语言模型生成场景，将节点添加到有向无环图中。它首先创建一个草图有向无环图（例如，一个 5 分钟视频，镜头时长为 45 秒），并设定大致的截止时间，之后再根据实际生成的节点进行细化（例如，场景 2 的 0:37–1:12 部分必须在协调世界时 19:39 前完成）。

StreamCast 使用请求调度器 库来发现 K8s 集群中所有已部署的实例模型及其硬件资源和位置。随后，调度器将有向无环图节点分派到相应的模型实例。如果某个镜头无法满足其截止时间，则会降低其质量。最后，StreamCast 使用 FFmpeg [44]拼接各项输出，并将结果流式传回用户。

其他应用。 除 StreamCast 外，我们还实现了表1中的其他应用工作流：StreamShort、StreamMovie、StreamAnimated、StreamLecture、StreamPersona、StreamDub、StreamEdit 和 StreamChat。我们主要修改大语言模型的输入（内容和包含少样本示例的提示词）以及有向无环图生成。其余部分则使用相同的组件和库。

5 评估

方法论。 我们纳入了一个朴素的非优化基线，代表开箱即用的实现。该基线根据模型的最大有效并行性，将模型静态分区到可用的 GPU 上。例如，在由四台服务器组成、每台配备 8 个 A100 GPU 的部署中，它会按照模型的运行时间比例分配 GPU，但受各模型并行性上限的限制（例如，Flux 最多使用 8 个 GPU）。无法并行化的模型（例如，YOLO）则固定在单个 GPU 上。该基线在单一区域使用按需（非竞价实例）A100 GPU，不拆分模型，并以最高质量执行每个请求（不进行超分辨率处理）。我们还与其他基线进行比较，这些基线将现有的大语言模型优化方法移植到多模态生成工作流中。

硬件。 我们在 Azure 虚拟机上使用 Azure Kubernetes 服务 (AKS) [85] 运行主要实验。我们评估了一种由单台配备 8× A100 GPU 的服务器组成的小规模设置，以及多种组合了 A100、H100、H200 和 GB200 服务器的配置。这些配置的服务器数量从 8 台到 40 台不等，最多包含 320 个 GPU，共计 10 种配置。我们使用这些代表性数据点来验证延迟和成本估算器，然后使用经过验证的模型来模拟更多配置。

5.1 模型配置

我们首先评估 StreamWise 为 StreamCast 进行的预配，包括：(1) 一种低成本设置，即一台配备 8 个 GPU 的 A100 服务器；(2) 一种成本高效设置，最多使用 32×8×台 A100 和 8×8×台 H200 服务器（320 个 GPU）。表 4 显示了为每个模型分配的 GPU。对于 A100 设置，轻量级模型（Kokoro 和 YOLO）共享一个图形处理器，而计算最密集的阶段（即 FantasyTalking）分配了两个 GPU。相比之下，朴素 基线在不进行 FramePack 解耦的情况下为每个模型分配一个图形处理器，使首帧时间从 5 小时增加到 8 小时以上。

成本高效的配置使用 12 个 Fantasy Talking 实例（96 张 A100 和 50 张 H200），以满足延迟要求。朴素 方法配置了 200 张 H100 GPU，但端到端延迟仍高出 9.7× 。此配置包含 4 个 Fantasy Talking 实例，使用 4×8× 张 H200（96 张 GPU），由于流水线处理能力有限，效果明显较差。朴素基线消耗相同数量的 GPU，但将首帧时间从 20 秒延长至 4 分钟以上。

根据模型配置，StreamCast 遵循图2中的工作流：(1) Gemma [31] 生成剧本，触发第一个镜头的生成；(2) Kokoro [52]合成语音音频；(3) Flux [74] 生成高质量基础图像；(4) YOLO [126] 生成放大的图像裁剪；(5) Hunyuan+FramePack [72, 148] 使用这些图像在潜在空间中生成草图视频；(6) 同时，Hunyuan+FramePack VAE 从潜在表示中解码草图视频；(7) Fantasy Talking [133](基于

表 4. StreamCast 生成 10 分钟播客视频的资源配置与生成时间（秒）。该视频由 43 个镜头组成，采用 1280×800，使用 20 个扩散步数。

	低成本 8×A100			成本高效 32×8×A100 + 8×8×H200		
模型	#GPU	TTFF	Time	#GPU	TTFF	Time
StreamCast	-	1.2	3.5	-	1.2	3.5
Gemma	1	6.6	31.8	8+0	1.5	9.7
Flux	1	9.8	9.8	16+0	0.9	1.0
YOLO	0.5	0.1	0.6	0.5+0	0.1	0.1
Kokoro	0.5	0.6	25.8	0.5+0	0.6	25.8
FramePack	1	8.6	1486.2	41+8	0.9	27.1
FramePack VAE	1	2.0	343.0	20+4	0.9	52.4
Fantasy Talking	2	78.8	13588.7	96+50	14.6	131.1
Real-ESRGAN	1	15.4	2663.4	74+2	2.0	34.4

Wan [132])在生成过程中以中等质量合并视频和音频；(8) Real-ESRGAN [134] 将输出提升至高质量；(9) StreamCast 将各个镜头拼接起来，并流式传输最终视频。

多个区域。 若将部署限制在单一区域，可用硬件会减少，而且无法混用 H200 和 A100 硬件。H200 部署在美国东部，A100 部署在美国西部。StreamWise 在预配模型时会考虑带宽和延迟。这与 Helix [82] 为大语言模型所做的类似。此外，使用多个区域可降低 Spot 虚拟机驱逐事件之间的相关性，从而提高对 Spot 虚拟机驱逐的容忍度。

5.2 多模态生成服务请求

表 4 展示了使用我们的 8×A100 和混合 A100+H200 配置，以 1280 × 800 和 23 帧 / 秒生成一段关于本文、时长为 10 分钟的视频播客所需的首帧时间（TTFF）和总生成时间。在 8×A100 GPU 上，首帧可在 123 秒后生成，但直到 3.8 小时后才会生成最后一帧。若要实现不间断的实时流式传输，首帧时间为 3.7 小时。256×A100+64×H200 配置将首帧时间缩短至 <22 秒。其余帧会在 10 分钟内生成，因此首帧时间结束后即可开始实时流式传输。

延迟与成本。图 8 展示了在改变已配置硬件的情况下，以高质量生成一段 10 分钟视频播客的首帧时间和成本。 8×A100 GPU 配置的成本为 <$25，但延迟较高。作为参考，在 8×A100 上以朴素方式运行，不采用解耦、超分辨率和感知截止时间的调度，速度会慢 2.2×，且成本会高出 >$50。H100+H200 配置的成本略高，约为 $45，但首帧时间不到 22 秒。我们在真实的 Azure Kubernetes 集群上评估了这两种配置。

64×A100 的首帧时间为 2 分钟，成本约为 $25。相比之下， 8×A100 的成本几乎高出 2×，是 16 的.....

×A100 的执行时间较长。混合使用 A100 和 H200 可将首帧时间缩短至 1 分钟以内，同时将成本控制在 50 美元以下。总体而言，A100 的成本效率最高，而混合

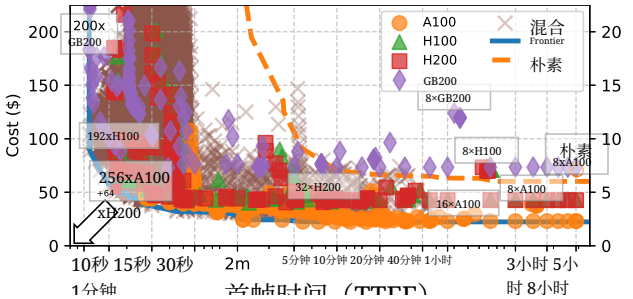

图 8. 使用不同硬件配置生成 10 分钟高质量视频播客的首帧时间与 成本。左下角是成本最低、速度最快的配置。

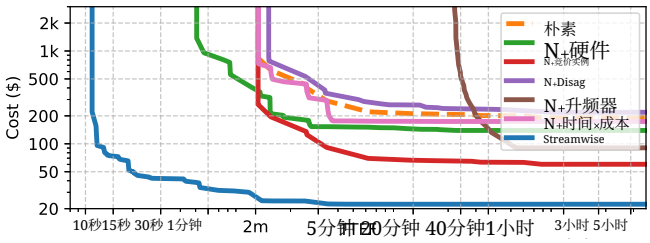

图 9. StreamWise 在高质量下使用的各项技术的贡献，包括朴素 基线。

A100+H100 配置提供了最佳的延迟-成本权衡。与 H100 相比，H200 配置带来的性能提升微乎其微，因此成本效益较低。GB200 能进一步缩短首帧时间，但其每小时成本显著更高，因此除非目标是实现极低延迟（即，低于 15 秒），否则与 H100 相比缺乏竞争力。StreamWise 配置少量高成本 GPU（例如，GB200）以缩短首帧时间，并将剩余容量分配给成本较低的 GPU（例如，A100）。StreamWise 选择成本高效的方案，采用 256×A100 和 64×H200。

宽松的服务等级目标。在服务没有实时截止时间的请求时，StreamWise 会在这一延迟空间中进行权衡，并选择成本最低的配置。例如，如果视频无需在 2 小时内生成，它会使用 2 台服务器和 8×A100 GPU，将成本降至 25 美元以下。直到首帧时间的截止时间收紧至 2 分钟，成本才会开始上升。

消融研究。图 9 展示了相对于朴素 基线，各项 StreamWise 技术的贡献。硬件 可同时降低成本和延迟。解耦 可进一步降低成本，而竞价实例 则能在延迟相近的情况下进一步降低成本。针对时间×成本 进行优化，可显著降低成本，同时对延迟影响甚微。升频器 可大幅缩短首帧时间，但对成本影响不大。StreamWise 综合运用所有技术，实现了最佳的成本-延迟权衡。

图10 展示了从 StreamWise 中移除各项技术所造成的性能损失。不使用硬件 （仅使用*A100*）时，首帧时间无法低于 30 秒。不使用竞价实例时，成本会大幅上升，使前沿向上移动。

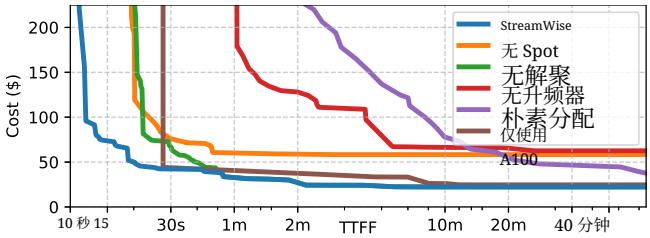

图10。 在以高质量生成 10 分钟播客时，禁用 StreamWise 中各项技术所造成的性能损失。

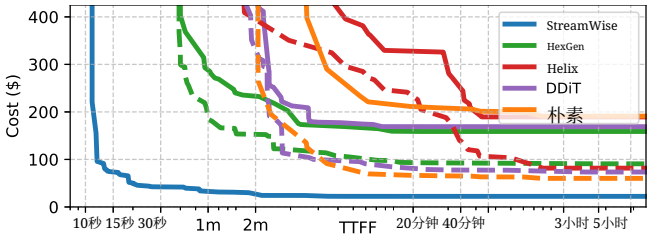

图11. 高效大语言模型服务系统的适用性。虚线表示使用竞价虚拟机来降低成本。

不采用解耦时，系统无法利用流水线处理，导致首帧时间更长。不使用升频器时，成本和延迟都会大幅增加。用朴素分配器 替代贪心启发式也会显著增加成本。

总的来说，这些结果表明，单靠任何一项技术都不足以实现高效的实时生成。

移植大语言模型服务优化。应用大语言模型的预填充/解码解耦 [100, 151]并不能改善多模态工作流的首帧时间或成本，因为大语言模型所占的总执行时间不到 1%。因此，单独针对大语言模型服务的优化对端到端效果几乎没有影响。尽管如此，我们还是改造了两个有代表性的系统（HexGen [65] 和 Helix [82]），并针对我们的场景重新设计。我们还进一步扩展了这两个系统，使其能够选择性地利用竞价型虚拟机，从而提升其与 StreamWise 的竞争力。图 11 对比了它们与 StreamWise 的首帧时间和成本。

HexGen [65] 使用张量并行和流水线并行对大语言模型进行分区，并通过遗传算法在异构 GPU 上搜索部署位置和并行化策略，以实现高吞吐量。我们将其扩展为把多模态流水线中的每个组件都视为可分区单元，并利用其搜索来确定部署位置、资源分配和并行性。尽管进行了这项调整，竞价实例上的 HexGen 成本仍高出 3×，且 TTFF 比 StreamWise 慢 5×。这是因为 HexGen 优化的是单个模型的吞吐量，而非降低异构阶段之间的端到端关键路径延迟。

Helix [82] 将大语言模型服务建模为最大流问题，并使用混合整数线性规划计算优化后的部署位置和并行化策略。我们扩展了其建模方法，使其能够联合优化多模态流水线中的所有组件。然而，竞价实例上的 Helix 在 TTFF 方面表现更差，因为它在全局预算内独立优化每个模型，却没有在流水线阶段之间平衡资源。因此，它甚至比朴素方法表现更差：它为某些模型过度预配资源，却为另一些模型分配不足，导致阶段失衡和关键路径延长。这种针对单个模型的优化无法捕捉跨阶段依赖关系，从而降低了端到端性能。

解耦。 近期的系统（如 DDiT [56] 和 StreamDiT [71]）主张对基于扩散的模型进行解耦，以提高利用率和可扩展性。图 11包括

这种应用于播客生成工作流的 DiT 解耦设计。然而，仅分离 DiT 和变分自编码器组件，还不足以实现高效的实时性能。虽然解耦提高了资源的灵活性，但并不会降低关键路径延迟。

成本影响也很大程度上取决于部署规模。在小规模下，解耦会增加成本，因为 DiT 和变分自编码器必须分别运行在不同的图形处理器上，而不能共置于同一设备上。然而，在较大规模下，解耦可使各组件独立扩展（例如，DiT 使用 32 个图形处理器，而变分自编码器仅使用 2 个），从而使容量更好地匹配需求并降低成本。尽管 DiT/变分自编码器解耦能够提升大规模场景下的弹性，但仅靠解耦还不足以实现高效的实时多模态生成。这一观察与图10一致，图中展示了未采用解耦的 StreamWise。

贪心启发式。 图12将 StreamWise 的贪心分配启发式与一种最优方案进行了比较。该方案使用 Gurobi [50]求解混合整数规划，以将多模态人工智能组件分配给图形处理器。对于较宽松的延迟目标（即，首帧时间超过一分钟），贪心启发式与最优方案效果相当。在更严格的延迟约束下，其成本仍在最优成本的 20% 以内，并且在最严格的首帧时间目标下收敛至相同方案。

不过，这些算法的计算开销差异显著。贪心启发式可在 100 毫秒内完成，从而支持在线重配置。相比之下，求解完整的混合整数规划可能需要 10 分钟以上，因此不适用于实时控制。如果需要更高的准确性，可以使用缓存的最优解为贪心启发式提供热启动。

自适应质量。图13展示了以三种质量生成内容的效果：(1) 高质量，采用 20 个去噪步数，1280×800（与图 8 相同）；(2) 中等质量，采用 10 步，640×400；以及 (4) 低质量，采用 5 步，320×200。采用低质量时，我们的首帧时间为 <3 秒，费用为每分钟 <$0.5。

自适应。图13 展示了在内容生成过程中调整质量如何降低延迟。流式传输以低质量开始，首帧时间不到 3 秒；10–20 秒后切换为中等质量，并在 45 秒内达到高质量。

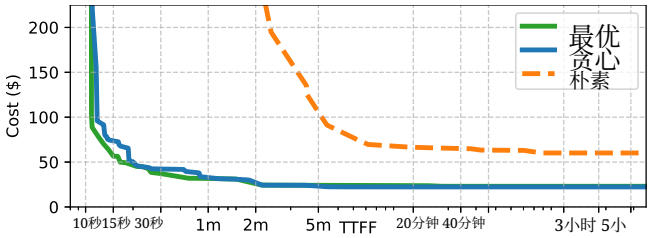

图12. 贪心启发式相对于最优分配的质量。最优方案耗时是贪心方案的 100×倍。

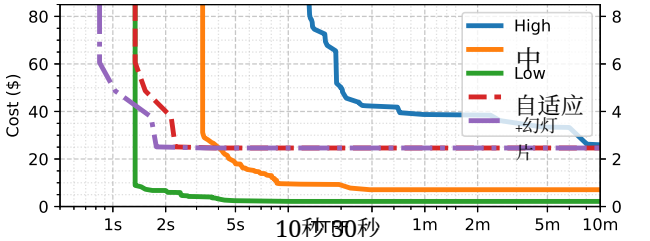

图 13. 首帧时间 与 高、中和低输出质量下的成本帕累托前沿。包含我们的自适应质量策略。

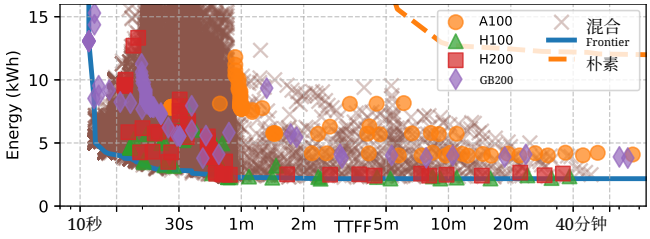

图 14。视频播客生成的能源效率。

这种方法的成本不到 50 美元，同时可使视频中超过 90% 的部分达到高质量。

非生成内容。我们巧妙地以一张持续 500 毫秒的标题页（例如，展示论文标题和首页）开场，并配上主持人的画外音介绍视频，从而进一步将首帧时间缩短至 1 秒以内。

能源效率。图 14 展示了生成一段 10分钟播客所需的能耗。在这种情况下，单独使用 A100 GPU 的效率不如 H100，能耗为其的 2× 倍。GB200 的能源效率与 A100 相近。将 H100 与几块 A100 配合使用，能耗为 ~2 kWh，首帧时间不到 1 分钟；StreamWise 会选择这一配置。相比之下，朴素基线即使在能源效率最高的配置下也会消耗超过 10 kWh，而在速度最快、首帧时间超过 2 分钟的配置下则会消耗超过 50 kWh。

其他工作流。 图15评估了表1中的所有工作流。StreamWise 始终优于朴素策略，延迟降低 10.4×，成本节省 17.5×。成本效益最高的应用是幻灯片，因为它的分辨率要低得多。

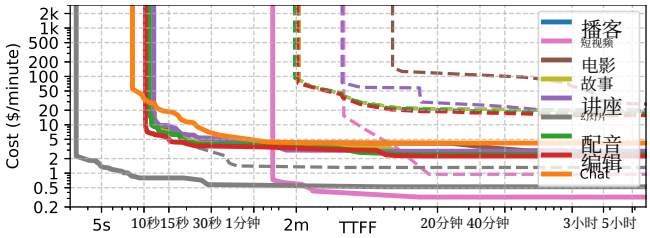

图15。对多模态工作流的敏感性。虚线表示朴素方案，实线表示 StreamWise。

成本最高的应用是*Chat*，因为它对交互性有要求。这表明，在延迟和质量约束下运行的生成式多模态模型所构成的应用，能够从 StreamWise 的抽象和调度机制中受益。

单体模型。由于不存在端到端的播客生成模型（第2.4节），我们使用单个扩散模型（即 Fantasy Talking）来模拟该模型，生成与 StreamCast 匹配的 10分钟视频。尽管 StreamWise 仍采用截止时间感知调度、解耦和竞价实例，但它放弃了模型专用化、跨阶段异构硬件以及选择性放大等模块化优势。该受限版本的 StreamWise 与 StreamDiffusionV2 [42]高度相似。

即使在这种受限设置下，StreamWise 仍将成本降低了 3.5×，相较于单体基线。在完全模块化的 StreamCast 工作流中，与单体方案相比，StreamWise 进一步将成本降低了 2.7×，并将延迟降低了 4.3×，量化了工作流级模块化所带来的性能和效率提升。

5.3 服务多个请求

图16展示了随着每分钟查询数（QPM）增加、根据所需模型副本数量计算的每小时成本。它从图13中的单个 请求开始，使用 256×A100和 64×H200（成本最高效）。在较低速率下（例如，1 QPM），最少的副本数量即可满足需求，将成本控制在每小时 1.2K 美元以内。随着负载增加，更多处理中的请求需要额外的副本，以避免排队和延迟。

轻量级模型（例如，Kokoro）可受益于实例共享和GPU共享，从而提高成本效率。将 Kokoro 从 1 QPM 扩展到 100 QPM，成本仅增加 43×。Flux 每个请求所需的激活值较少，因此吞吐量高，单个实例即可服务多个请求。Fantasy Talking 为满足实时服务级目标，需要为每个请求配备专用的 GPU副本，这限制了多路复用并推高了成本。

要维持相同的吞吐量和延迟，朴素 方法需要付出 5.6× 倍的成本。StreamWise 会动态调配资源，以高效处理并发的多模态生成。共享轻量级模型可最大限度地减少开销，而较重的模型仍是可扩展性的主要瓶颈。

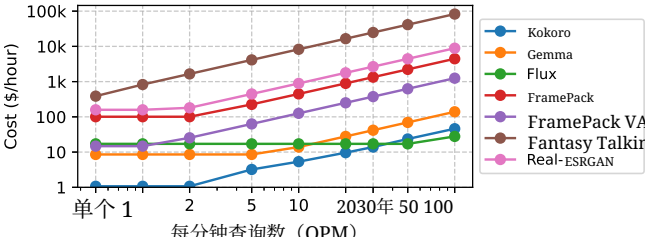

图16. 使用我们的成本高效自适应质量策略服务视频生成请求时，各模型的成本明细。

放宽服务级别目标。 我们将工作负载扩展为具有不同延迟要求的场景：三分之一为实时请求，三分之一的服务级别目标放宽 50%，另三分之一则没有服务级别目标（即批处理）。我们的截止时间感知调度器利用这部分宽裕时间，优先处理实时请求并减少副本数量；与延迟要求一致、均为实时服务级别目标的场景相比，成本进一步降低 37.9%。相比之下，朴素 方法无法利用这种异质性，仍会过度配置资源。总体而言，与基线相比，我们将成本降低了 9.1×。

6 相关工作

工作流与有向无环图调度。 Tez [111]、Ray [88]、Spark [144]、YARN [128]、Borg [18], 等通用集群调度器 [18, 53, 60, 113], 为分布式调度和资源管理奠定了基础。然而，它们需要经过调整才能满足多模态生成的需求（例如，StreamWise 基于 K8s 构建）。

推理即服务。 FastFlow [149], 、INFaaS [109]等早期机器学习推理系统面向单模态或粗粒度批处理工作流，通常假设有较简单的有向无环图。相比之下，现代多模态生成流水线引入了细粒度有向无环图，其中各组件紧密耦合（例如，DiT、变分自编码器、超分辨率、文本转语音），且各自具有独特的延迟和资源需求。Llama [110], Serverless-LLM [45],和 VIVA [68] 可优化成本、延迟和资源利用率。然而，它们假设算子图的动态性较低，且交互性有限。

大语言模型服务。 以往的工作主要聚焦于优化大语言模型的执行。这类模型有两个定义明确的阶段（即预填充和解码）以及键值缓存 [3, 65, 82, 100, 151]。Split-Wise [100] 和 DistServe [151] 提出将这两个阶段拆分，并将它们部署在不同的硬件上。Helix [82] 和 HexGen [65] 提出通过管理细粒度并行性，在异构硬件上以及跨区域提供大语言模型服务。Medha [4] 则进行大语言模型专用优化，例如自适应分块和截止时间感知调度。相比之下，多模态生成的工作流更加复杂（例如涉及文本/图像编码器、变分自编码器、DiT、文本转语音和大语言模型），可调参数也更多（例如分辨率、帧数和步数）。

多模态服务。 近期研究探讨了服务多模态输入模型所面临的挑战 [103]。Mora [143]、StreamDiT [71],和 StreamDiffusionV2 [42] 等系统开始着手解决组合式和流式文本生成视频的问题。然而，针对多模态模型实时编排的全面解决方案仍然有限。Veo 3 [49]、Runway [34], 、SeeDance [46], 和 Sora [97] 等商业系统展示了大规模多模态生成的潜力，但其系统架构和调度策略大多属于专有信息，外界无从得知。

扩散优化。 近期工作通过推理复用和架构专用化降低扩散成本。对于文生图，ApproximateCaching [1]、TeaCache [79]、MoDM [136],和 DiffServe [5]等方法会缓存内部状态或调整质量，但仅限于静态的单模态设置。对于视频，AdaCache [67]、FasterCache [81],和 Vchitect [39] 利用潜变量缓存和并行性来应对批处理瓶颈。个性化和适配通常使用 LoRA [55] 及相关技术 [57, 119]。尽管 FlashAttention [29, 30] 和 SAGE [146]等优化提高了阶段效率，但它们并未解决端到端调度或实时多模态服务问题。

7 结论

我们介绍了 StreamWise，这是一个面向播客视频创作等工作流的实时多模态生成系统。其模块化设计支持自适应扩展、并行性以及跨不同硬件的部署。我们确定了在严格延迟约束下为这些模型提供服务所面临的关键挑战，并量化了延迟-成本-质量权衡。未来的工作可以通过模型专用化以及更紧密的模型-调度器协同设计来提升效率。

参考文献

[1] Shubham Agarwal、Subrata Mitra、Sarthak Chakraborty、Srikrishna Karanam、Koyel Mukherjee 和 Shiv Saini。《利用近似缓存高效服务扩散模型》。*arXiv:2312.04429*，2023年。[2] Shubham Agarwal、Subrata Mitra、Sarthak Chakraborty、Srikrishna Karanam、Koyel Mukherjee 和 Shiv Kumar Saini。《利用近似缓存高效服务文生图扩散模型》。载于*NSDI*，2024年。[3] Amey Agrawal、Nitin Kedia、Ashish Panwar、Jayashree Mohan、Nipun Kwatra、Bhargav Gulavani、Alexey Tumanov 和 Ramachandran Ramjee。《使用 Sarathi-Serve 驯服大语言模型推理中的吞吐量—延迟权衡》。载于*OSDI*，2024年。[4] Amey Agrawal、Haoran Qiu、Junda Chen、Íñigo Goiri、Chaojie Zhang、Rayyan Shahid、Ramachandran Ramjee、Alexey Tumanov 和 Esha Choukse。《Medha：无需近似，高效服务数百万上下文长度的大语言模型推理请求》。*arXiv:240917264*，2024年。。[5] Sohaib Ahmad、Qizheng Yang、Haoliang Wang、Ramesh K Sitaraman 和 Hui Guan。《DiffServe：利用查询感知模型缩放高效服务文生图扩散模型》。*arXiv:2411.15381*，2024年。[6] Jasper AI。Flux.1-dev-Controlnet-Upscaler，2025年。[7] Meta AI。《Meta Llama 3：开放基础模型与指令微调语言模型》。https://github.com/meta-llama/llama3，2024年。

[8] Stability AI。《Stable Diffusion 3.5：高保真文生图生成》。https://stability.ai/news/introducing-stable-diffusion-3 5，2025年 - 。[9] Artificial Analysis。图像竞技场排行榜，2025年。[10] Artificial Analysis。文本转语音竞技场，2025年。[11] Artificial Analysis。视频竞技场排行榜，2025年。[12] Anthropic。Claude Code 概览。https://docs.anthropic.com/en/ docs/claude-code/overview，2025年。[13] InfiniBand 行业协会。InfiniBand 架构规范。载于*InfiniBand* 行业协会，2001年。版本 1.0。[14] Microsoft Azure。Azure 虚拟机规格——GPU 加速型，2025年。[15] Aadit Barua、Karim Benharrak、Meng Chen、Mina Huh 和 Amy Pavel。《Lotus：通过抽象式和抽取式摘要将长视频制作成短视频》。载于*IUI*，2025年。[16] Andreas Blattmann、Tim Dockhorn、Sumith Kulal、Daniel Mendele-vitch、Maciej Kilian、Dominik Lorenz、Yam Levi、Zion English、Vikram Voleti、Adam Letts 等。《Stable Video Diffusion：将潜空间视频扩散模型扩展至大型数据集》。*arXiv:2311.15127*，2023年。[17] Meriem Boubdir、Edward Kim、Beyza Ermis、Sara Hooker 和 Marzieh Fadaee。《Elo 揭秘：语言模型评估的稳健性与最佳实践》。*NeurIPS*，2024年。[18] Brendan Burns、Brian Grant、David Oppenheimer、Eric Brewer 和 John Wilkes。《Borg、Omega 和 Kubernetes：十年来三种容器管理系统的经验教训》。*Queue*，14(1)，2016年。[19] Qi Cai、Jingwen Chen、Yang Chen、Yehao Li、Fuchen Long、Yingwei Pan、Zhaofan Qiu、Yiheng Zhang、Fengbin Gao、Peihan Xu 等。《HiDream-I1：采用稀疏扩散 Transformer 的高效图像生成基础模型》。*arXiv:2505.22705*，2025年。[20] Gohar Irfan Chaudhry、Esha Choukse、Haoran Qiu、Íñigo Goiri、Rodrigo Fonseca、Adam Belay 和 Ricardo Bianchini。《Murakkab：云平台中资源高效的智能体工作流编排》。*arXiv:2508.18298*，2025年。[21] Xiaokang Chen、Zhiyu Wu、Xingchao Liu、Zizheng Pan、Wen Liu、Zhenda Xie、Xingkai Yu 和 Chong Ruan。《Janus-Pro：通过数据和模型缩放实现统一的多模态理解与生成》。*arXiv:2501.17811*，2025年。[22] Yi Chen、Sen Liang、Zixiang Zhou、Ziyao Huang、Yifeng Ma、Junshu Tang、Qin Lin、Yuan Zhou 和 Qinglin Lu。《混元视频数字人：面向多个角色的高保真音频驱动人物动画》。*arXiv:2505.20156*，2025年。[23] Google Cloud。GPU 机器类型，2025年。[24] CloudPrice。CloudPrice，2025年。[25] Comfy.org。Flux FP8。https://huggingface.co/Comfy-Org/flux1-dev，2025年。[26] ComfyUI。ComfyUI。https://github.com/Comfy-Org/ComfyUI，2025年。[27] Coqui.ai。XTTS-v2：跨语言语音克隆与多语言文本转语音。https://huggingface.co/coqui/XTTS-v2，2024年。[28] NVIDIA Corporation。NCCL：NVIDIA 集合通信库。https://developer.nvidia.com/nccl，2017年。[29] Tri Dao。《FlashAttention-2：通过更优的并行性与工作划分实现更快的注意力机制》。载于*ICLR*，2024年。[30] Tri Dao、Daniel Y. Fu、Stefano Ermon、Atri Rudra 和 Christopher Ré。《FlashAttention：快速且内存高效的精确注意力机制，具备 I/O 感知能力》。载于*NeurIPS*，2022年。[31] Google DeepMind。Gemma 3：谷歌推出的多模态开放模型。https://huggingface.co/google/gemma-3-1b-it，2025年。[32] Chaorui Deng、Deyao Zhu、Kunchang Li、Chenhui Gou、Feng Li、Zeyu Wang、Shu Zhong、Weihao Yu、Xiaonan Nie、Ziang Song、Guang Shi 和 Haoqi Fan。《统一多模态预训练中的新兴特性》。*arXiv:2505.14683*，2025年。[33] Prafulla Dhariwal 和 Alexander Nichol。《扩散模型在图像合成方面胜过生成对抗网络》。*NeurIPS*，2021年。[34] Patrick Esser、Johnathan Chiu、Parmida Atighehchian、Jonathan Granskog 和 Anastasis Germanidis。利用扩散模型进行结构与内容引导的视频合成。载于*ICCV*，2023年。[35] Hugging Face。热门任务，2025年。[36] Hugging Face。文生图，2025年。[37] Hugging Face。文本转语音，2025年。[38] Hugging Face。文生视频，2025年。[39] Weichen Fan、Chenyang Si、Junhao Song、Zhenyu Yang、Yinan He、Long Zhuo、Ziqi Huang、Ziyue Dong、Jingwen He、Dongwei Pan 等。Vchitect-2.0：用于扩展视频扩散模型的并行Transformer。*arXiv:2501.08453*，2025年。[40] Jiarui Fang、Jinzhe Pan、Xibo Sun、Aoyu Li 和 Jiannan Wang。xDiT：具备大规模并行能力的扩散Transformer（DiT）推理引擎。*arXiv:2411.01738*，2024年。[41] Jiarui Fang 和 Shangchun Zhao。USP：面向长上下文生成式人工智能的统一序列并行方法。*arXiv:240507719*，2024。。[42] Tianrui Feng、Zhi Li、Shuo Yang、Haocheng Xi、Muyang Li、Xiuyu Li、Lvmin Zhang、Keting Yang、Kelly Peng、Song Han 等。StreamDiffusionV2：用于动态交互式视频生成的流式系统。*arXiv:2511.07399*，2025年。[43] Chinasa Ferderick。YouTube搜索引擎优化研究：哪些技术能提升2025年视频排名?https://adilo.com/blog/youtube-seo-study/，2025年。[44] FFmpeg 开发者。FFmpeg。https://ffmpeg.org/，2025年。[45] Yao Fu、Leyang Xue、Yeqi Huang、Andrei-Octavian Brabete、Dmitrii Ustiugov、Yuvraj Patel 和 Luo Mai。ServerlessLLM：面向大语言模型的低延迟无服务器推理。载于*OSDI*，2024年。[46] Yu Gao、Haoyuan Guo、Tuyen Hoang、Weilin Huang、Lu Jiang、Fangyuan Kong、Huixia Li、Jiashi Li、Liang Li、Xiaojie Li 等。Seedance 1.0：探索视频生成模型的边界。*arXiv:2506.09113*，2025年。[47] GitHub。GitHub Copilot：你的 AI结对程序员。https://github.com/features/copilot，2021年。[48] 谷歌。NotebookLM，2025年。[49] Google DeepMind。Veo 3：高质量视频-音频生成模型。https://deepmind.google/models/veo，2025年。[50] Gurobi Optimization, LLC。Gurobi优化器参考手册。https://www.gurobi.com，2024年。[51] Yoav HaCohen、Nisan Chiprut、Benny Brazowski、Daniel Shalem、Dudu Moshe、Eitan Richardson、Eran Levin、Guy Shiran、Nir Zabari、Ori Gordon、Poriya Panet、Sapir Weissbuch、Victor Kulikov、Yaki Bitterman、Zeev Melumian 和 Ofir Bibi。LTX-Video：实时视频潜在扩散。*arXiv:2501.00103*，2024年。[52] Hexgrad。Kokoro。https://github.com/hexgrad/kokoro，2025年。[53] Benjamin Hindman、Andy Konwinski、Matei Zaharia、Ali Ghodsi、Anthony D Joseph、Randy Katz、Scott Shenker 和 Ion Stoica。Mesos：用于数据中心细粒度资源共享的平台。载于*NSDI*，2011年。[54] Jonathan Ho、Ajay Jain 和 Pieter Abbeel。去噪扩散概率模型。*NeurIPS*，2020年。[55] Edward J. Hu、Yelong Shen、Phillip Wallis、Zeyuan Allen-Zhu、Yuanzhi Li、Shean Wang 和 Weizhu Chen。LoRA：大语言模型的低秩适配，2021年。[56] Heyang Huang、Cunchen Hu、Jiaqi Zhu、Ziyuan Gao、Liangliang Xu、Yizhou Shan、Yungang Bao、Sun Ninghui、Tianwei Zhang 和 Sa Wang。DDiT：扩散Transformer模型服务的动态资源分配。*arXiv:2506.13497*，2025年。[57] Lianghua Huang、Wei Wang、Zhi-Fan Wu、Yupeng Shi、Huanzhang Dou、Chen Liang、Yutong Feng、Yu Liu 和 Jingren Zhou。扩散Transformer的上下文LoRA。*arXiv:2410.23775*，2024年。[58] Yanping Huang、Youlong Cheng、Ankur Bapna、Orhan Firat、Mia Chen、Zhifeng Chen、HyoukJoong Lee、Jiquan Ngiam、Quoc V. Le 和 Yonghui Wu。GPipe：利用流水线并行高效训练巨型神经网络。载于*NeurIPS*，2019年。

[59] IMDb。IMDb 数据集。https://www.imdb.com/interfaces/，2025年。[60] Michael Isard、Mihai Budiu、Yuan Yu、Andrew Birrell 和 Dennis Fetterly。Dryad：由顺序构建模块组成的分布式数据并行程序。载于*EuroSys*，2007年。[61] Sam Ade Jacobs、Masahiro Tanaka、Chengming Zhang、Minjia Zhang、Shuaiwen Leon Song、Samyam Rajbhandari 和 Yuxiong He。Deep-Speed Ulysses：支持极长序列Transformer模型训练的系统优化。*arXiv:2309.14509*，2023年。[62] Jane Marie Laroya。Veo 3 Pro 与 Ultra 对比：谷歌AI套餐究竟哪个更值得？https://www.softlist.io/veo-3-pro-vs-ultra-which- plan-is-worth-it，2025年。[63] Xiaozhong Ji、Xiaobin Hu、Zhihong Xu、Junwei Zhu、Chuming Lin、Qingdong He、Jiangning Zhang、Donghao Luo、Yi Chen、Qin Lin 等。Sonic：聚焦人像动画中的全局音频感知。*arXiv:2411.16331*，2024年。[64] Xiaozhong Ji、Chuming Lin、Zhonggan Ding、Ying Tai、Junwei Zhu、Xiaobin Hu、Donghao Luo、Yanhao Ge 和 Chengjie Wang。RealTalk：借助3D面部先验引导的身份对齐网络，实现实时逼真的音频驱动人脸生成。*arXiv:2406.18284*，2024年。[65] Youhe Jiang、Ran Yan、Xiaozhe Yao、Yang Zhou、Beidi Chen 和 Binhang Yuan。HexGen：异构环境下的大语言模型生成式推理。2024年。[66] Phillip Jones。Quart：一个与 Flask 具有相同 API 的 Python ASGI Web 微框架。https://github.com/pgjones/quart，2023年。[67] Kumara Kahatapitiya、Haozhe Liu、Sen He、Ding Liu、Menglin Jia、Chenyang Zhang、Michael S Ryoo 和 Tian Xie。通过扩散Transformer自适应缓存加速视频生成。*arXiv:241102397*，2024。。[68] Daniel Kang、Francisco Romero、Peter D Bailis、Christos Kozyrakis 和 Matei Zaharia。VIVA：用于交互式视频分析的端到端系统。载于*CIDR*，2022年。[69] Nima Kaviani、Dmitriy Kalinin 和 Michael Maximilien。迈向作为基础设施的无服务器：Knative 案例。载于*WoSC*，2019年。[70] Diederik P Kingma、Max Welling 等。自编码变分贝叶斯，2013年。[71] Akio Kodaira、Tingbo Hou、Ji Hou、Masayoshi Tomizuka 和 Yue Zhao。StreamDiT：实时流式文本生成视频，2025年。[72] Weijie Kong、Qi Tian、Zijian Zhang、Rox Min、Zuozhuo Dai、Jin Zhou、Jiangfeng Xiong、Xin Li、Bo Wu、Jianwei Zhang 等。Hunyuan-Video：大型视频生成模型的系统化框架。*arXiv:2412.03603*，2024年。[73] Woosuk Kwon、Zhuohan Li、Siyuan Zhuang、Ying Sheng、Lianmin Zheng、Cody Hao Yu、Joseph E. Gonzalez、Hao Zhang 和 Ion Stoica。利用分页注意力高效管理大语言模型服务的内存。载于*SOSP*，2023年。[74] Black Forest Labs。Flux。https://github.com/black-forest-labs/flux，2024年。[75] Black Forest Labs、Stephen Batifol、Andreas Blattmann、Frederic Boe- sel、Saksham Consul、Cyril Diagne、Tim Dockhorn、Jack English、Zion English、Patrick Esser、Sumith Kulal、Kyle Lacey、Yam Levi、Cheng Li、Dominik Lorenz、Jonas Müller、Dustin Podell、Robin Rombach、Harry Saini、Axel Sauer 和 Luke Smith。FLUX.1 Kontext：潜在空间中用于上下文图像生成与编辑的流匹配，2025年。[76] Nari Labs。Dia，2025年。[77] Shenggui Li、Fuzhao Xue、Chaitanya Baranwal、Yongbin Li 和 Yang You。序列并行：从系统角度进行长序列训练。*arXiv:2105.13120*，2021年。[78] Shijia Liao、Yuxuan Wang、Tianyu Li、Yifan Cheng、Ruoyi Zhang、Rongzhi Zhou 和 Yijin Xing。Fish-Speech：利用大语言模型实现先进的多语言文本转语音合成。*arXiv:2411.01156*，2024年。[79] Feng Liu、Shiwei Zhang、Xiaofeng Wang、Yujie Wei、Haonan Qiu、Yuzhong Zhao、Yingya Zhang、Qixiang Ye 和 Fang Wan。时间步

Embedding Tells：是时候为视频扩散模型进行缓存了。*arXiv:2411.19108*，2024年。[80] Hao Liu、Matei Zaharia 和 Pieter Abbeel。借助分块 Transformer 实现近乎无限上下文的环形注意力。*arXiv:2310.01889*，2023年。[81] Zhengyao Lv、Chenyang Si、Junhao Song、Zhenyu Yang、Yu Qiao、Ziwei Liu 和 Kwan-Yee K. Wong。FasterCache：无需训练的高质量视频扩散模型加速。2024年。[82] Yixuan Mei、Yonghao Zhuang、Xupeng Miao、Juncheng Yang、Zhihao Jia 和 Rashmi Vinayak。Helix：通过最大流在异构 GPU 和网络上服务大语言模型。发表于*ASPLOS*，2025年。[83] Yihao Meng、Hao Ouyang、Yue Yu、Qiuyu Wang、Wen Wang、Ka Leong Cheng、Hanlin Wang、Yixuan Li、Cheng Chen、Yanhong Zeng、Yujun Shen 和 Huamin Qu。HoloCine：电影式多镜头长视频叙事的整体生成。*arXiv:251020822*，2025年。。[84] Dirk Merkel 等。Docker：用于一致开发和部署的轻量级 Linux 容器。*Linux Journal*，2014年。[85] 微软。Azure Kubernetes 服务 (AKS)。https://azure.microsoft.com/en-us/services/kubernetes-service/，2025年。[86] 微软。VibeVoice，2025年。[87] Microsoft Azure。NDasrA100_v4 规模系列。https: //learn.microsoft.com/en-us/azure/virtual-machines/sizes/gpu-accelerated/ndasra100v4-series，2025年。[88] Philipp Moritz、Robert Nishihara、Stephanie Wang、Alexey Tumanov、Richard Liaw、Eric Liang、Melih Elibol、Zongheng Yang、William Paul、Michael Jordan 和 Michael Isard。Ray：面向新兴 AI 应用的分布式框架。发表于*OSDI*，2018年。[89] Dhruv Narayanan、Mohammad Shoeybi、Jared Casper、Mostofa Pat- wary、Raul Puri、Patrick LeGresley、Vikas Korthikanti 和 Bryan Catanzaro。使用 Megatron-LM 在 GPU 集群上高效训练大规模语言模型。*arXiv:2104.04473*，2021年。[90] 英伟达。Tesla V100。https://www.nvidia.com/en-us/data-center/ tesla-v100/，2017年。[91] 英伟达。Ampere GPU 架构。https://www.nvidia.com/en-us/data-center/ampere-architecture/，2020年。[92] 英伟达。Hopper GPU 架构。https://www.nvidia.com/en-us/data-center/technologies/hopper-architecture/，2022年。[93] 英伟达。CUDA 多进程服务。https://docs.nvidia.com/deploy/mps/index.html，2024年。[94] 英伟达。GB200 NVL72。https://www.nvidia.com/en-us/data-center/ gb200-nvl72/，2025年。[95] 英伟达。多实例 GPU 用户指南。https://docs.nvidia.com/datacenter/tesla/mig-user-guide/index.html，2025年。[96] 英伟达。NVIDIA CUDA Docker 镜像。https://hub.docker.com/r/ nvidia/cuda，2025年。[97] OpenAI。Sora：OpenAI 的文本转视频生成模型。https://openai.com/index/sora/，2024年。[98] OpenAI。OpenAI 平台。https://platform.openai.com/docs/ overview，2025年。[99] Adam Paszke、Sam Gross、Francisco Massa、Adam Lerer、James Bradbury、Gregory Chanan、Trevor Killeen、Zeming Lin、Natalia Gimelshein、Luca Antiga、Alban Desmaison、Andreas Kopf、Edward Yang、Zachary DeVito、Martin Raison、Alykhan Tejani、Sasank Chil- amkurthy、Benoit Steiner、Lu Fang、Junjie Bai 和 Soumith Chintala。PyTorch：一种命令式风格的高性能深度学习库。*NeurIPS*，2019年。[100] Pratyush Patel、Esha Choukse、Chaojie Zhang、Íñigo Goiri、Aashaka Shah、Saeed Maleki 和 Ricardo Bianchini。Splitwise：通过阶段拆分实现高效的生成式大语言模型推理。发表于*ISCA*，2024年。[101] Pratyush Patel、Esha Choukse、Chaojie Zhang、Íñigo Goiri、Brijesh Warrier、Nithish Mahalingam 和 Ricardo Bianchini。POLCA：大语言模型云服务提供商的电力超额配置，2023年。

[102] Fabian Pedregosa、Gaël Varoquaux、Alexandre Gramfort、Vincent Michel、Bertrand Thirion、Olivier Grisel、Mathieu Blondel、Peter Pret- tenhofer、Ron Weiss、Vincent Dubourg、Jake Vanderplas、Alexandre Passos、David Cournapeau、Mathieu Brucher、Matthieu Perrot 和 Édouard Duchesnay。Scikit-learn：Python 中的机器学习。《机器学习研究杂志》，12，2011年。[103] Haoran Qiu、Anish Biswas、Zihan Zhao、J ayashree Mohan、Alind Khare、Esha Choukse、Íñigo Goiri、Zeyu Zhang、Haiying Shen、Chetan Bansal 等。ModServe：可扩展且资源高效的大型多模态模型服务。发表于*SoCC*，2025年。[104] Alec Radford、Jong Wook Kim、Chris Hallacy、Aditya Ramesh、Gabriel Goh、Sandhini Agarwal、Girish Sastry、Amanda Askell、Pamela Mishkin、Jack Clark 等。通过自然语言监督学习可迁移视觉模型。发表于*ICML*，2021年。[105] Alec Radford、Jong Wook Kim、Tao Xu、Greg Brockman、Christine McLeavey 和 Ilya Sutskever。通过大规模弱监督实现稳健语音识别。发表于*ICML*，2023年。[106] Colin Raffel、Noam Shazeer、Adam Roberts、Katherine Lee、Sharan Narang、Michael Matena、Yanqi Zhou、Wei Li 和 Peter J Liu。探索统一文本到文本 Transformer 的迁移学习极限。《机器学习研究杂志》，2020年。[107] Aditya Ramesh、Prafulla Dhariwal、Alex Nichol、Casey Chu 和 Mark Chen。基于 CLIP 潜变量的分层文本条件图像生成。*arXiv:2204.06125*，2022年。[108] Babak Rokh、Ali Azarpeyvand 和 Alireza Khanteymoori。深度神经网络图像分类模型量化综合综述。智能系统与技术汇刊，14，2023年。[109] Francisco Romero、Qian Li、Neeraja J Yadwadkar 和 Christos Kozyrakis。INFaaS：自动化无模型推理服务。发表于*USENIXATC*，2021年。[110] Francisco Romero、Mark Zhao、Neeraja J Yadwadkar 和 Christos Kozyrakis。Llama：用于自动调优视频分析流水线的异构无服务器框架。发表于*SoCC*，2021年。[111] Bikas Saha、Hitesh Shah、Siddharth Seth、Gopal Vijayaraghavan、Arun Murthy 和 Carlo Curino。Apache Tez：用于建模和构建数据处理应用的统一框架。发表于*SIGMOD* 数据管理国际会议，2015年。[112] Chitwan Saharia、William Chan、Huiwen Chang、Chris Lee、Jonathan Ho、Tim Salimans、David Fleet 和 Mohammad Norouzi。Palette：图像到图像扩散模型。发表于*SIGGRAPH*，2022年。[113] Malte Schwarzkopf、Andy Konwinski、Michael Abd-El-Malek 和 John Wilkes。Omega：面向大型计算集群的灵活、可扩展调度器。发表于*EuroSys*，2013年。[114] 亚马逊云科技。Amazon EC2 实例类型，2025年。[115] SGLang。SGLang Diffusion：加速视频和图像生成。https://lmsys.org/blog/2025-11-07-sglang-diffusion/，2025年。[116] Jay Shah、Ganesh Bikshandi、Ying Zhang、Vijay Thakkar、Pradeep Ramani 和 Tri Dao。FlashAttention-3：通过异步执行和低精度实现快速、准确的注意力机制。*NeurIPS*，37，2024年。[117] Haoyuan Shi、Yunxin Li、Xinyu Chen、Longyue Wang、Baotian Hu 和 Min Zhang。AniMaker：借助 MCTS 驱动的片段生成实现自动化多智能体动画故事创作。*arXiv:2506.10540*，2025年。[118] Mohammad Shoeybi、Mostofa Patwary、Raul Puri、Patrick LeGres- ley、Jared Casper 和 Bryan Catanzaro。Megatron-LM：使用模型并行训练数十亿参数语言模型。*arXiv:1909.08053*，2019年。[119] James Seale Smith、Yen-Chang Hsu、Lingyu Zhang、Ting Hua、Zsolt Kira、Yilin Shen 和 Hongxia Jin。持续扩散：借助 C-LoRA 持续定制文生图扩散模型。*arXiv:2304.06027*，2023年。[120] Jovan Stojkovic、Chaojie Zhang、Íñigo Goiri、Esha Choukse、Haoran Qiu、Rodrigo Fonseca、Josep Torrellas 和 Ricardo Bianchini。TAPAS：云平台中面向大语言模型推理的热能与功耗感知调度。发表于*ASPLOS*，第1266–1281页，2025年。

[121] Jovan Stojkovic、Chaojie Zhang、Íñigo Goiri、Josep Torrellas 和 Esha Choukse。《DynamoLLM：面向性能与能源效率设计大语言模型推理集群》。发表于*HPCA*，2025年。[122] Aolan Sun、Xulong Zhang、Tiandong Ling、Jianzong Wang、Ning Cheng 和 Jing Xiao。《Pre-avatar：一种利用会说话的虚拟形象自动生成演示文稿的框架》。发表于*ICTAI*，2022年。[123] Peize Sun、Yi Jiang、Shoufa Chen、Shilong Zhang、Bingyue Peng、Ping Luo 和 Zehuan Yuan。《自回归模型胜过扩散模型：用于可扩展图像生成的 Llama》。*arXiv:2406.06525*，2024年。[124] Wenhao Sun、Rong-Cheng Tu、Jingyi Liao 和 Dacheng Tao。《基于扩散模型的视频编辑：综述》。*arXiv:2407.07111*，2024年。[125] Jiale Tao、Yanbing Zhang、Qixun Wang、Yiji Cheng、Haofan Wang、Xu Bai、Zhengguang Zhou、Ruihuang Li、Linqing Wang、Chunyu Wang 等。《InstantCharacter：基于可扩展扩散变换器框架个性化任意角色》。*arXiv:2504.12395*，2025年。[126] Ultralytics。《YOLOv11：Ultralytics 实时目标检测模型》。https://docs.ultralytics.com/，2024年。[127] Ashish Vaswani、Noam Shazeer、Niki Parmar、Jakob Uszkoreit、Llion Jones、Aidan N Gomez、Lukasz Kaiser 和 Illia Polosukhin。《Attention Is All You Need》。*NeurIPS*，2017年。[128] Vinod Kumar Vavilapalli、Arun C Murthy、Chris Douglas、Sharad Agarwal、Mahadev Konar、Robert Evans、Thomas Graves、Jason Lowe、Hitesh Shah、Siddharth Seth 等。《Apache Hadoop YARN：又一个资源协商器》。发表于*SoCC*，2013年。[129] vLLM。《vLLM 基准测试》。https://github.com/vllm-project/vllm/tree/main/benchmarks，2025年。[130] Patrick von Platen、Suraj Patil、Anton Lozhkov、Pedro Cuenca、Nathan Lambert、Kashif Rasul、Mishig Davaadorj、Dhruv Nair、Sayak Paul、William Berman、Yiyi Xu、Steven Liu 和 Thomas Wolf。《Diffusers：最先进的扩散模型》。https://github.com/huggingface/diffusers，2022年。[131] Alexander Waibel、Moritz Behr、Dogucan Yaman、Fevziye Irem Eyiokur、Tuan-Nam Nguyen、Carlos Mullov、Mehmet Arif Demirtas、Alperen Kantarci、Stefan Constantin 和 Hazim Kemal Ekenel。《视频配音：口型同步且保留原声的翻译》。发表于*ICASSPW*，2023年。[132] Ang Wang、Baole Ai、Bin Wen、Chaojie Mao、Chen-Wei Xie、Di Chen、Feiwu Yu、Haiming Zhao、Jianxiao Yang、Jianyuan Zeng 等。《Wan：开放且先进的大规模视频生成模型》。*arXiv:2503.20314*，2025年。[133] Jinfeng Wang、Yifan Zhang、Zhen Li 等。《FantasyTalking：通过连贯的运动合成实现逼真的说话人像生成》。*arXiv:2504.12345*，2025年。[134] Xintao Wang、Liangbin Xie、Chao Dong 和 Ying Shan。《Real-ESRGAN：利用纯合成数据训练真实世界盲超分辨率模型》。发表于*ICCVW*，2021年。[135] Chengyue Wu、Xiaokang Chen、Zhiyu Wu、Yiyang Ma、Xingchao Liu、Zizheng Pan、Wen Liu、Zhenda Xie、Xingkai Yu、Chong Ruan 等。《Janus：通过解耦视觉编码实现统一的多模态理解与生成》。*arXiv:2410.13848*，2024年。[136] Yuchen Xia、Divyam Sharma、Yichao Yuan、Souvik Kundu 和 Nishil Talati。《MoDM：通过混合扩散模型实现高效图像生成服务》。*arXiv:2503.11972*，2025年。[137] Hanyi Xu、Wensheng Gan、Zhenlian Qi、Jiayang Wu 和 Philip S. Yu。《教育领域的大语言模型：综述》。*arXiv:240513001*，2024年。。[138] Yichao Yan、Zanwei Zhou、Zi Wang、Jingnan Gao 和 Xiaokang Yang。《DialogueNeRF：迈向逼真的虚拟形象面对面交谈视频生成》。*Visual Intelligence*，2(1):24，2024年。[139] Hu Ye、Jun Zhang、Sibo Liu、Xiao Han 和 Wei Yang。《IP-Adapter：兼容文本的图像提示适配器，用于文本到图像扩散模型》。*arXiv:2308.06721*，2023年。[140] Sangho Yi、Derrick Kondo 和 Artur Andrzejak。《通过检查点降低 Amazon Elastic Compute 竞价实例的成本

Cloud》。发表于*CLOUD*，2010年。[141] Gyeong-In Yu、Joo Seong Jeong、Geon-Woo Kim、Soojeong Kim 和 Byung-Gon Chun。《Orca：用于基于 Transformer 的生成模型的分布式服务系统》。发表于*OSDI*，2022年。[142] Shoubin Yu、Difan Liu、Ziqiao Ma、Yicong Hong、Yang Zhou、Hao Tan、Joyce Chai 和 Mohit Bansal。《Veggie：通过基于依据的生成进行指令式编辑和视频概念推理》。发表于*CVF*，2025年。[143] Zhengqing Yuan、Yixin Liu、Yihan Cao、Weixiang Sun、Haolong Jia、Ruoxi Chen、Zhaoxu Li、Bin Lin、Li Yuan、Lifang He 等。《Mora：通过多智能体框架实现通用视频生成》。*arXiv:2403.13248*，2024年。[144] Matei Zaharia、Mosharaf Chowdhury、Tathagata Das、Ankur Dave、Justin Ma、Murphy McCauly、Michael J Franklin、Scott Shenker 和 Ion Stoica。《弹性分布式数据集：一种用于内存集群计算的容错抽象》。发表于*NSDI*，2012年。[145] Dengsheng Zhang。《借助人工智能驱动的视频讲座变革高等教育》。*arXiv:2511.20660*，2025年。[146] Jintao Zhang、Jia Wei、Haofeng Huang、Pengle Zhang、Jun Zhu 和 Jianfei Chen。《SageAttention：用于即插即用推理加速的精准 8 位注意力》。*arXiv:2410.02367*，2024年。[147] Lvmin Zhang 和 Maneesh Agrawala。为文生图扩散模型添加条件控制，2023年。[148] Lvmin Zhang、Lingdong Kong、Feng Liu、Yujun Wang、Zhen Li 等。《FramePack：为视频生成中的下一帧预测模型打包输入帧上下文》。*arXiv:2505.07719*，2025年。[149] Peiyuan Zhang、Yongqi Chen、Runlong Su、Hangliang Ding、Ion Stoica、Zhengzhong Liu 和 Hao Zhang。《通过滑动瓦片注意力实现快速视频生成》。*arXiv:2502.04507*，2025年。[150] Yue Zhang、Zhizhou Zhong、Minhao Liu、Zhaokang Chen、Bin Wu、Yubin Zeng、Chao Zhan、Yingjie He、Junxin Huang 和 Wenjiang Zhou。《MuseTalk：通过时空采样实现实时高保真视频配音》。*arXiv:2410.10122*，2024年。[151] Yinmin Zhong、Shengyu Liu、Junda Chen、Jianbo Hu、Yibo Zhu、Xu- anze Liu、Xin Jin 和 Hao Zhang。《DistServe：分离预填充与解码，实现吞吐量优化的大语言模型服务》。发表于*OSDI*，2024年。[152] Zeyu Zhu、Kevin Qinghong Lin 和 Mike Zheng Shou。《Paper2video：从科学论文自动生成视频》。*arXiv:251005096*，2025年。