Cutting the Cord: Designing a High-quality Untethered VR System with Low Latency Remote Rendering （Mobisys2018）

链接: https://dl.acm.org/doi/10.1145/3210240.3210313

1 内容概述

这篇文章介绍了一个适用于端到端的VR系统，
主要用到两种方法：

- 并行化渲染和流处理（Parallel Rendering and Streaming mechanism, PRS）
- 远程垂直同步渲染技术（Remote VSync Driven Rendering technique, RVDR）

在无线网络条件下，满足高质量VR的延迟和视频质量的需求，因此作者称之是“无所束缚的”,"Cutting the Cord"这个标题也是蛮生动形象的。

结果显示该VR系统可以在60Ghz无线网络、2160x1200的分辨率、90Hz刷新率的条件下达到16ms以内的延迟，4K分辨率可以达到20ms以内，同时保证给用户展示出无损的画质。

2 背景

VR的设备主要分为两种，一种是**结合式**的，另一种是**独立式**的。

- **结合式VR**需要将头戴设备（*Head Mounted Display, HMD*）和PC连接起来，头戴设备通过USB线将传感器数据传输给PC，然后PC通过HDMI线将渲染的画面传回给头戴设备，这种方式可以是VR

的画面质量更高清保真。
- **独立式VR**只是在单独一个设备上对画面进行操作，这种方式摆脱了“绳子的束缚”。

存在的问题

- **结合式VR**有“绳索的束缚”，用户行动不方便，甚至有安全隐患（比如被绳子缠绕脖子就麻烦了。。。hhh）
- **独立式VR**计算资源有限，没法做到高质量的画面渲染，有的人尝试用无线传输的方式将一些计算任务迁移到PC上，然而延迟会对画面帧率造成影响，使VR用户产生眩晕的感觉，严重影响用户体验。

3 延迟分析

对于无线的VR设备来说，端到端的延迟主要有以下几个组成部分：

$$T_{e2e} = T_{sense} + T_{render} + T_{stream} + T_{display}$$

其中，

$$T_{stream} = T_{encode} + T_{trans} + T_{decode}$$
$$T_{trans} = \frac{FrameSize}{Throughout}$$

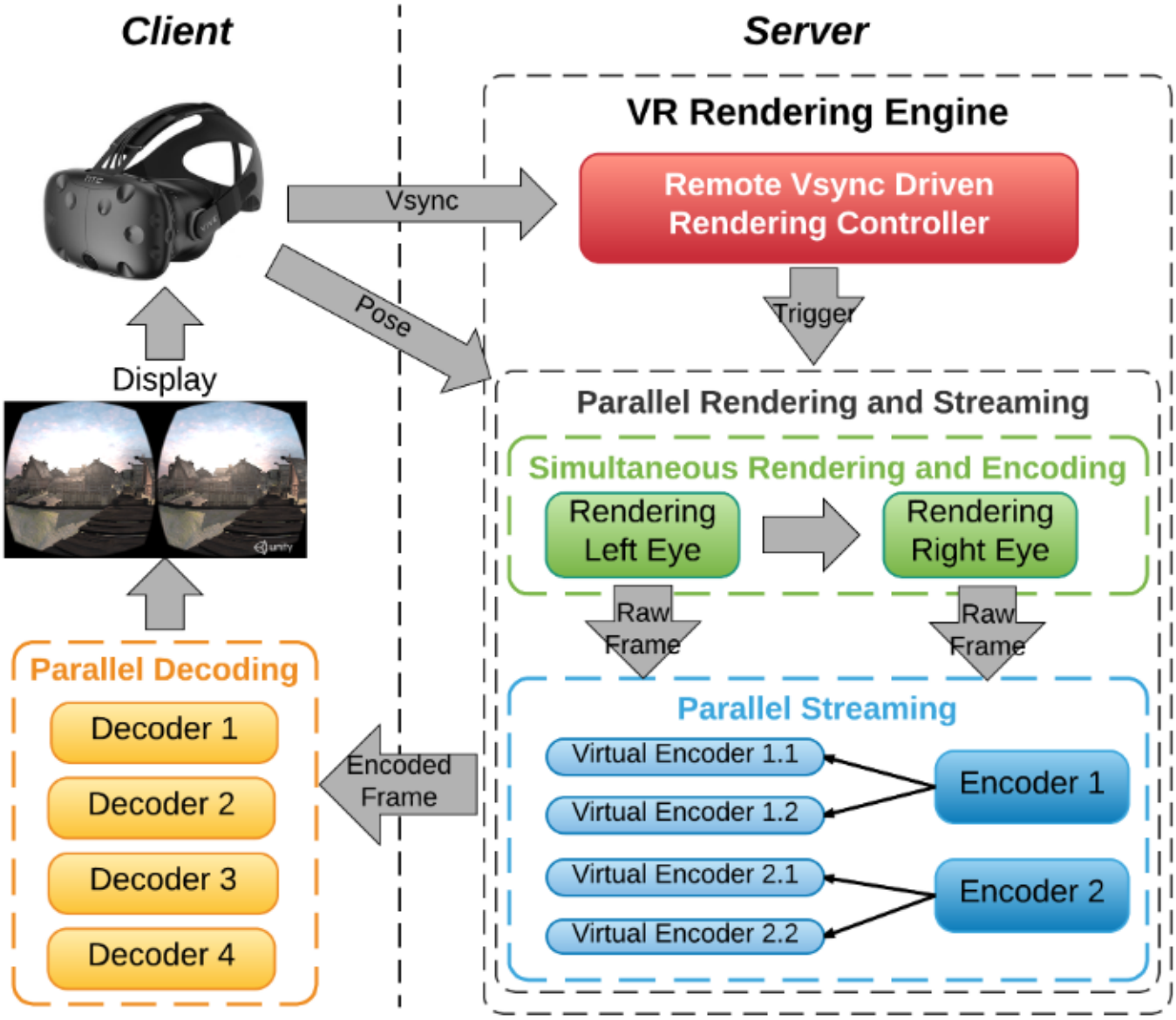
每个部分的含义如下：

- T_{e2e}: 端到端的延迟时间（从一个视频帧的产生再到它被显示出来的时间）
- T_{sense}: 服务器接收到HMD传感数据的时间
- T_{render}: 服务器根据传感数据生成一个新视频帧的时间
- T_{stream}: 将新视频帧从服务器传输到HMD的时间
  - T_{encode}: 在服务器上压缩（编码）一个新视频帧的时间
  - T_{trans}: 从服务器将压缩的帧通过无线网络传输到HMD的时间
  - T_{decode}: 在HMD上解压（解码）一个新视频帧的时间
- $T_{display}$: HMD将新视频帧显示出来的时间

值得注意的是，$T_{display}$被考虑在了端到端的延迟中，这是因为，在现在的图像系统中，一个视频帧的显示是通过**VSync信号**驱动的，VSync信号是通过屏幕刷新率周期性生成的。如果一个帧错过了当前的VSync信号，那么它就需要在缓存队列中等待，直到下一次的VSync信号传来，才可以显示在屏幕上，对于90Hz的刷新率来说，平均等待时间是5.5ms，这个等待时间会对整体的延迟造成一定影响，所以这个时间也需要被优化。

T_{sense}和T_{render}是不能改变的，这篇文章主要针对通过优化T_{stream}（PRS）和$T_{display}$（RVDR）来减小整体端到端的延迟。

4 系统架构



作者使用了**WiGig**无线连接，这是一种60Hz的无线通信方式。
[系统概述，待完成]

5 并行化渲染和流处理

这部分主要是为了减少上文所说的T_{stream}延迟，主要包含两部分：

- 同时渲染与编码
- 并行流处理

5.1 同时渲染与编码

渲染一个高质量的视频帧需要比较长的时间，甚至在一个高性能的显卡上也需要超过5ms的时间，我们又不能降低图像质量，因此需要寻找其他方法去减少这一部分的耗时，提出了一种同时渲染和编码

的方法。

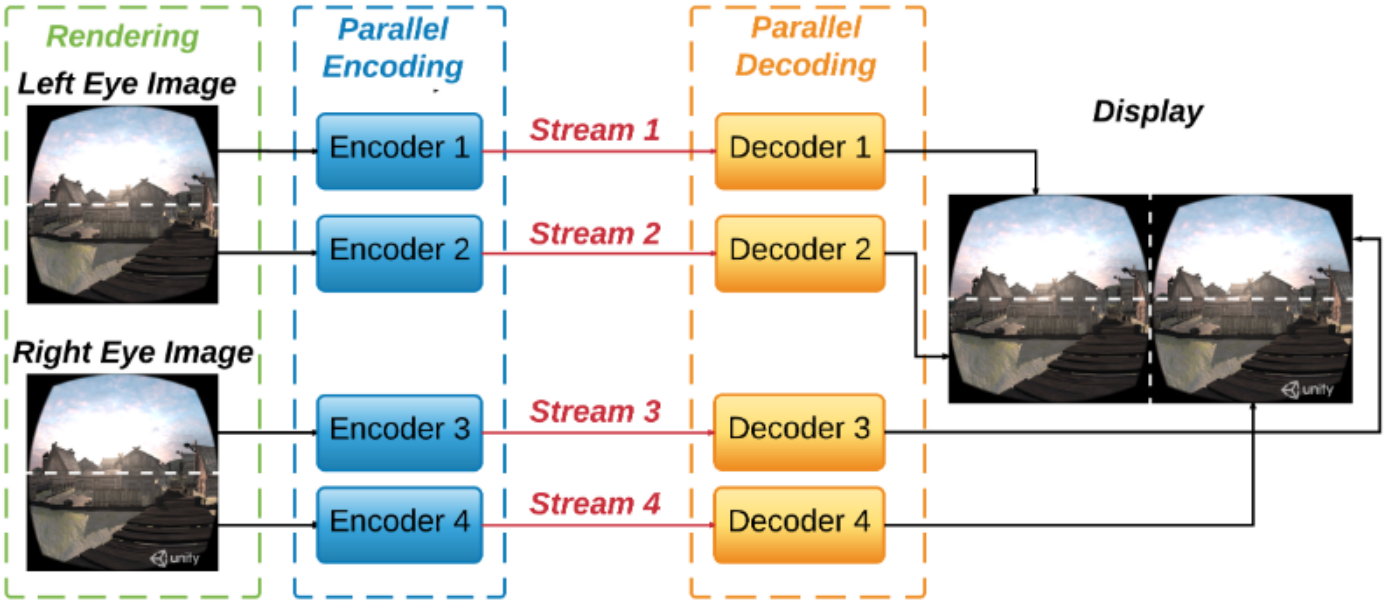
可行性主要有两点：

1. VR上的渲染通常分三步骤进行：（1）渲染左眼图像（2）渲染右眼图像（3）在整个视频帧加上镜头模糊，因此这种顺序渲染提供了一个机会，让我们可以在渲染右眼图像的同时开始对渲染好的左眼图像进行编码。
2. 现在的好多GPU有专用的硬件编解码部分，他们和渲染部分是相互独立的，因此即使利用GPU的编解码部分，也不会对渲染产生影响。

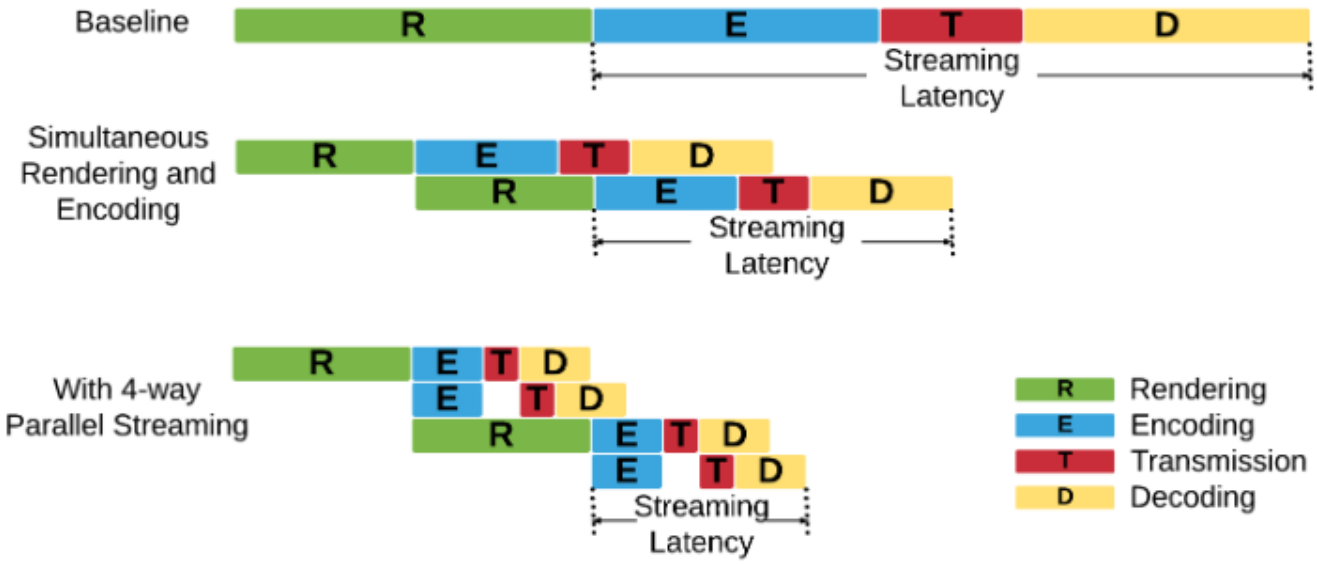
5.2 并行流处理

为了进一步减小视频流处理的时间，文中提出了一种多线程视频流处理的技术，利用多线程对每只眼睛的图像进行编码。

文中将每只眼的图像有分成两个slide，分别对应图像的上半部分和下半部分，因此整个视频帧对应4个slide，对应有四个编码线程和解码线程，结合前面提到的同时渲染与编码，过程如下图所示。



文中也解释了为什么这种分slide处理可以节约视频流处理的时间，如下图所示，



看到这里可能有人会问，岂不是线程数越多越好?

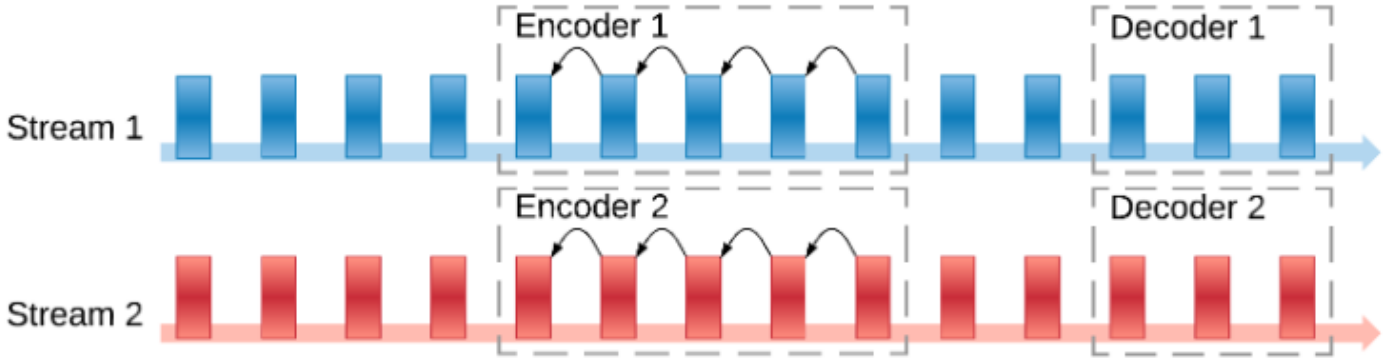
然而并不是这样，因为GPU上编解码部分并行的资源是有限的，而且分的slide太多会影响H.264编码的性能，对延迟造成影响，系统的实现也会变得更加复杂。

编码器的多路复用

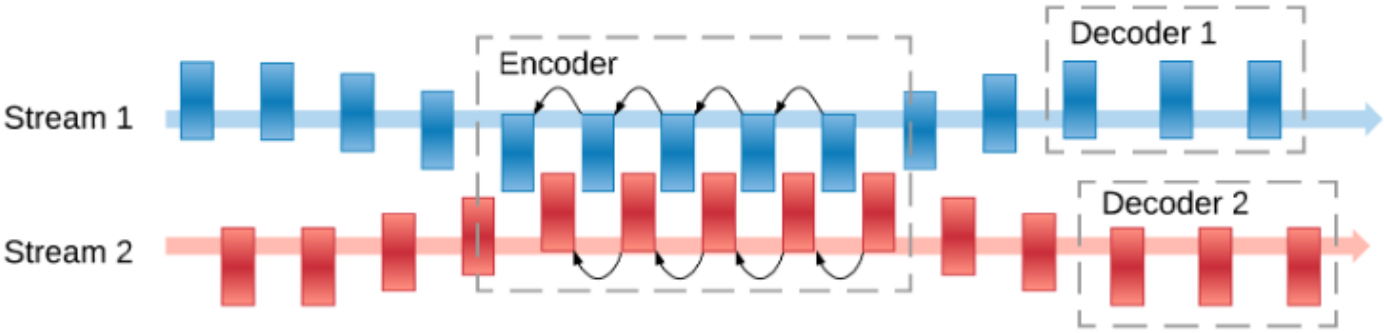
现在的商用GPU最多只支持两路视频同时压缩，而文中的方法需要4路，文中提出了多路复用的解决方法，如下图所示。

多路服用的思想:

- encoding session 1
  - left eye's upper half frame
  - right eye's upper half frame
- encoding session 2
  - left eye's bottom half frame
  - right eye's bottom half frame



(a) Encode two streams with two encoding sessions.



(b) Encode two streams with only one encoding session.

6 远程垂直同步渲染