

媒体云转码的演进： MapReduce、DASH 与稳定婚姻

Alan Zhuang

cheedoong@acm.org

2014 年 3 月 18 日

Tencent 腾讯



Outline

1 背景

2 从 Cloud Transcoder 到 TranscX

- 前腾讯研究院 Cloud Transcoder
- 架平流媒体 TranscX

3 DASH 与稳定婚姻

- DASH
- 稳定匹配

4 Future Vision

- 和一些新技术的结合
- Acknowledgment

Outline

1 背景

2 从 Cloud Transcoder 到 TranscX

- 前腾讯研究院 Cloud Transcoder
- 架平流媒体 TranscX

3 DASH 与稳定婚姻

- DASH
- 稳定匹配

4 Future Vision

- 和一些新技术的结合
- Acknowledgment

互动

- Q: 谁拥有以下设备之一?

互动

- Q: 谁拥有以下设备之一?
 - 小米 3 联通版、红米 Note

互动

- Q: 谁拥有以下设备之一?
 - 小米 3 联通版、红米 Note
 - 华为荣耀 X1、荣耀 3X

互动

- Q: 谁拥有以下设备之一?
 - 小米 3 联通版、红米 Note
 - 华为荣耀 X1、荣耀 3X
 - 三星 Galaxy S4

互动

- Q: 谁拥有以下设备之一?
 - 小米 3 联通版、红米 Note
 - 华为荣耀 X1、荣耀 3X
 - 三星 Galaxy S4
 - Google/LG Nexus 5

互动

■ Q: 谁拥有以下设备之一?

- 小米 3 联通版、红米 Note
- 华为荣耀 X1、荣耀 3X
- 三星 Galaxy S4
- Google/LG Nexus 5
- LG G2

互动

- Q: 谁拥有以下设备之一?
 - 小米 3 联通版、红米 Note
 - 华为荣耀 X1、荣耀 3X
 - 三星 Galaxy S4
 - Google/LG Nexus 5
 - LG G2
 - 酷派大神 F1/8297、9976A

互动

- Q: 谁拥有以下设备之一?
 - 小米 3 联通版、红米 Note
 - 华为荣耀 X1、荣耀 3X
 - 三星 Galaxy S4
 - Google/LG Nexus 5
 - LG G2
 - 酷派大神 F1/8297、9976A

- 举过手的，恭喜大家！

原因请继续听...

多屏时代的挑战

■ 多种平台



多屏时代的挑战

■ 多种平台



多屏时代的挑战

■ 多种平台



多屏时代的挑战

■ 多种平台



多屏时代的挑战

■ 多种平台



多屏时代的挑战

■ 多种平台



■ 多种屏幕大小



多屏时代的挑战

■ 多种平台



■ 多种屏幕大小



多屏时代的挑战

■ 多种平台



■ 多种屏幕大小



多屏时代的挑战

■ 多种平台



■ 多种屏幕大小



Tencent 腾讯

多屏时代的挑战

■ 多种平台



■ 多种屏幕大小



Tencent 腾讯

多屏时代的挑战

■ 多种码率



多屏时代的挑战

■ 多种码率



多屏时代的挑战

■ 多种码率



多屏时代的挑战

■ 多种码率



多屏时代的挑战

■ 多种码率



多屏时代的挑战

■ 多种码率



■ 多种解码能力

联发科芯片	MT6572	MT6582	MT6588	MT6592
Display	960×540P	1280×720P	1920×1280P	1920×1280P
H.264 Decode	720P@30fps	1080P@30fps	1080P@30fps	1080P@30fps
HEVC Decode	N/A	N/A	720P@30fps	720P@30fps

多屏时代的挑战

■ 不同封装容器支持

mp4, mkv, avi, flv, wmv, rmvb, webm, mpeg-ts...

多屏时代的挑战

■ 不同封装容器支持

mp4, mkv, avi, flv, wmv, rmvb, webm, mpeg-ts...

多屏时代的挑战

■ 不同封装容器支持

mp4, mkv, avi, flv, wmv, rmvb, webm, mpeg-ts...

■ 不同编码标准支持

多屏时代的挑战

■ 不同封装容器支持

mp4, mkv, avi, flv, wmv, rmvb, webm, mpeg-ts...

■ 不同编码标准支持

多屏时代的挑战

■ 不同封装容器支持

mp4, mkv, avi, flv, wmv, rmvb, webm, mpeg-ts...

■ 不同编码标准支持

H.264(AVC), H.265(HEVC), VC-1, AVS, VP8/9, RealVideo...

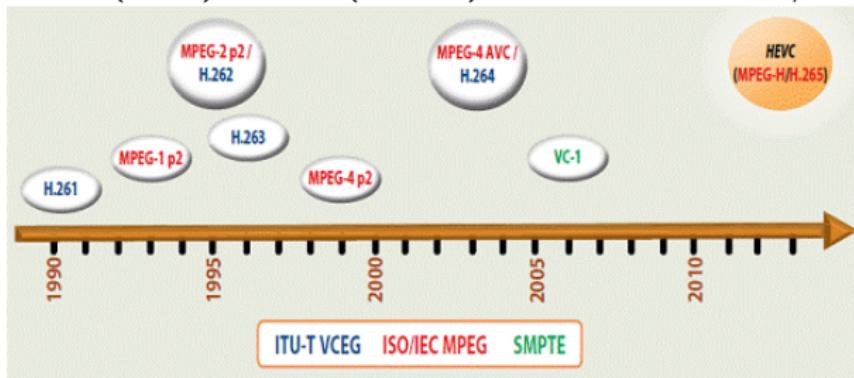
多屏时代的挑战

■ 不同封装容器支持

mp4, mkv, avi, flv, wmv, rmvb, webm, mpeg-ts...

■ 不同编码标准支持

H.264(AVC), H.265(HEVC), VC-1, AVS, VP8/9, RealVideo...



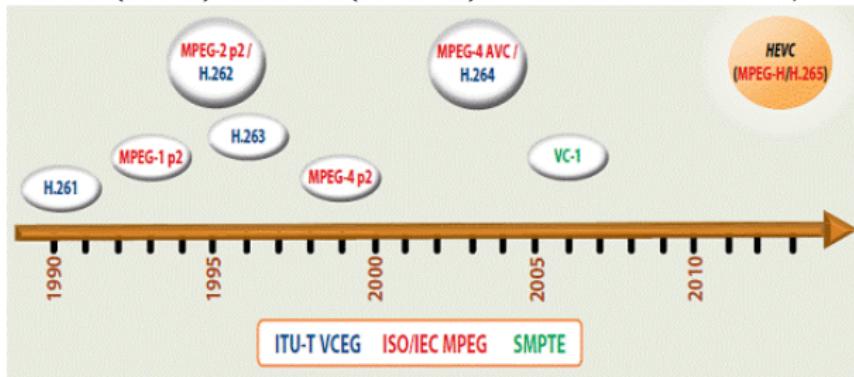
多屏时代的挑战

■ 不同封装容器支持

mp4, mkv, avi, flv, wmv, rmvb, webm, mpeg-ts...

■ 不同编码标准支持

H.264(AVC), H.265(HEVC), VC-1, AVS, VP8/9, RealVideo...



多屏时代的挑战

■ 巨头角力



多屏时代的挑战

■ 巨头角力



多屏时代的挑战

■ 巨头角力



幸运的是，

幸运的是，

绝大多数设备都支持：

幸运的是，

绝大多数设备都支持：

- 编码标准

幸运的是，

绝大多数设备都支持：

- 编码标准

幸运的是，

绝大多数设备都支持：

- 编码标准

H.264/AVC (ISO/IEC 14496-10; ITU-T H.264; MPEG-4 Part 10).

- 封装容器

幸运的是，

绝大多数设备都支持：

- 编码标准

H.264/AVC (ISO/IEC 14496-10; ITU-T H.264; MPEG-4 Part 10).

- 封装容器

幸运的是，

绝大多数设备都支持：

- 编码标准

H.264/AVC (ISO/IEC 14496-10; ITU-T H.264; MPEG-4 Part 10).

- 封装容器

MP4 (ISO/IEC 14496-14; MPEG-4 Part 14).

幸运的是，

绝大多数设备都支持：

- 编码标准

H.264/AVC (ISO/IEC 14496-10; ITU-T H.264; MPEG-4 Part 10).

- 封装容器

MP4 (ISO/IEC 14496-14; MPEG-4 Part 14).

所以，我们需要把用户/编辑上传的各种不同封装、不同编码的，一般码率比较高的源视频转成若干种适合不同设备的不同码率的H.264 编码、MP4 封装的视频。

但是，

但是，

- 媒体转码是件极其消耗计算资源的工作
尤其视频编码，对于目前常见的支持 SSE4 指令集的
x86/x64 CPU 的机器：

但是，

- 媒体转码是件极其消耗计算资源的工作
尤其视频编码，对于目前常见的支持 SSE4 指令集的
x86/x64 CPU 的机器：
 - 编码 H.264 视频需要耗费播放时长的 1/3 到 2/3

但是，

- 媒体转码是件极其消耗计算资源的工作
尤其视频编码，对于目前常见的支持 SSE4 指令集的
x86/x64 CPU 的机器：
 - 编码 H.264 视频需要耗费播放时长的 1/3 到 2/3
 - 编码 H.265 视频需要耗费播放时长的 30+ 倍

但是，

- 媒体转码是件极其消耗计算资源的工作
尤其视频编码，对于目前常见的支持 SSE4 指令集的
x86/x64 CPU 的机器：
 - 编码 H.264 视频需要耗费播放时长的 1/3 到 2/3
 - 编码 H.265 视频需要耗费播放时长的 30+ 倍
 - 单个 CPU 核通常最多可跑 1-2 个编码任务

但是，

- 媒体转码是件极其消耗计算资源的工作
尤其视频编码，对于目前常见的支持 SSE4 指令集的 x86/x64 CPU 的机器：
 - 编码 H.264 视频需要耗费播放时长的 1/3 到 2/3
 - 编码 H.265 视频需要耗费播放时长的 30+ 倍
 - 单个 CPU 核通常最多可跑 1-2 个编码任务
- 媒体文件大，再加上多码率副本，极其消耗存储

但是，

- 媒体转码是件极其消耗计算资源的工作
尤其视频编码，对于目前常见的支持 SSE4 指令集的 x86/x64 CPU 的机器：
 - 编码 H.264 视频需要耗费播放时长的 1/3 到 2/3
 - 编码 H.265 视频需要耗费播放时长的 30+ 倍
 - 单个 CPU 核通常最多可跑 1-2 个编码任务
- 媒体文件大，再加上多码率副本，极其消耗存储
- 潜在的带宽消耗

以往的解决：并行与分布式

以往的解决：并行与分布式

Criteria:

以往的解决：并行与分布式

Criteria:

- 单机内

以往的解决：并行与分布式

Criteria:

- 单机内
 - 功能划分、数据局部性
 - 宏块组粒度的并行

以往的解决：并行与分布式

Criteria:

- 单机内

- 功能划分、数据局部性
宏块组粒度的并行
- 内存访问、CPU 核心/Cache 拓扑结构和转码格式
帧级或 GOP (图像组) 级并行
对 NUMA 机器特别友好

以往的解决：并行与分布式

Criteria:

- 单机内

- 功能划分、数据局部性
宏块组粒度的并行
- 内存访问、CPU 核心/Cache 拓扑结构和转码格式
帧级或 GOP (图像组) 级并行
对 NUMA 机器特别友好

- 分布式转码

以往的解决：并行与分布式

Criteria:

- 单机内

- 功能划分、数据局部性
宏块组粒度的并行
- 内存访问、CPU 核心/Cache 拓扑结构和转码格式
帧级或 GOP (图像组) 级并行
对 NUMA 机器特别友好

- 分布式转码

- 存储

以往的解决：并行与分布式

Criteria:

- 单机内

- 功能划分、数据局部性
 - 宏块组粒度的并行
- 内存访问、CPU 核心/Cache 拓扑结构和转码格式
 - 帧级或 GOP (图像组) 级并行
 - 对 NUMA 机器特别友好

- 分布式转码

- 存储
- 路由

以往的解决：并行与分布式

Criteria:

- 单机内

- 功能划分、数据局部性
 - 宏块组粒度的并行
- 内存访问、CPU 核心/Cache 拓扑结构和转码格式
 - 帧级或 GOP (图像组) 级并行
 - 对 NUMA 机器特别友好

- 分布式转码

- 存储
- 路由
- 资源分配、任务调度

Outline

1 背景

2 从 Cloud Transcoder 到 TranscX

- 前腾讯研究院 Cloud Transcoder
- 架平流媒体 TranscX

3 DASH 与稳定婚姻

- DASH
- 稳定匹配

4 Future Vision

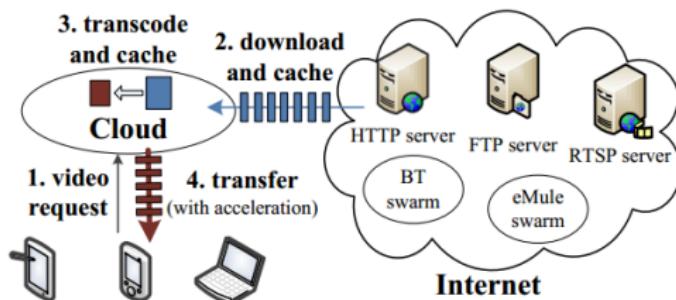
- 和一些新技术的结合
- Acknowledgment

前腾讯研究院 Cloud Transcoder

**Done 2011-. Gale Huang et al. Cloud transcoder:
bridging the format and resolution gap between internet
videos and mobile devices. ACM NOSSDAV 2012.**

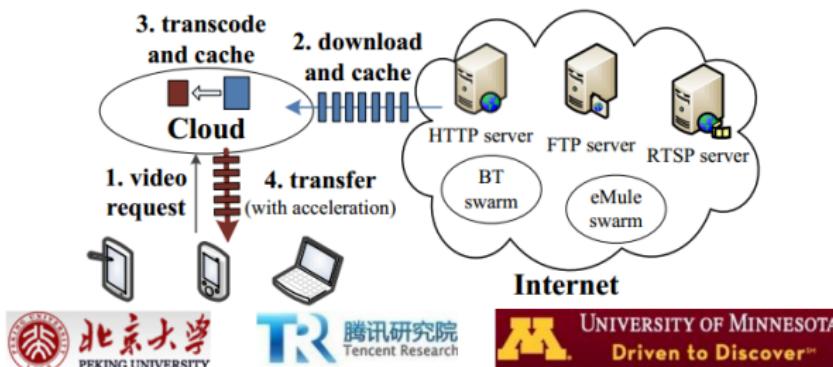
前腾讯研究院 Cloud Transcoder

Done 2011-. Gale Huang et al. Cloud transcoder: bridging the format and resolution gap between internet videos and mobile devices. ACM NOSSDAV 2012.

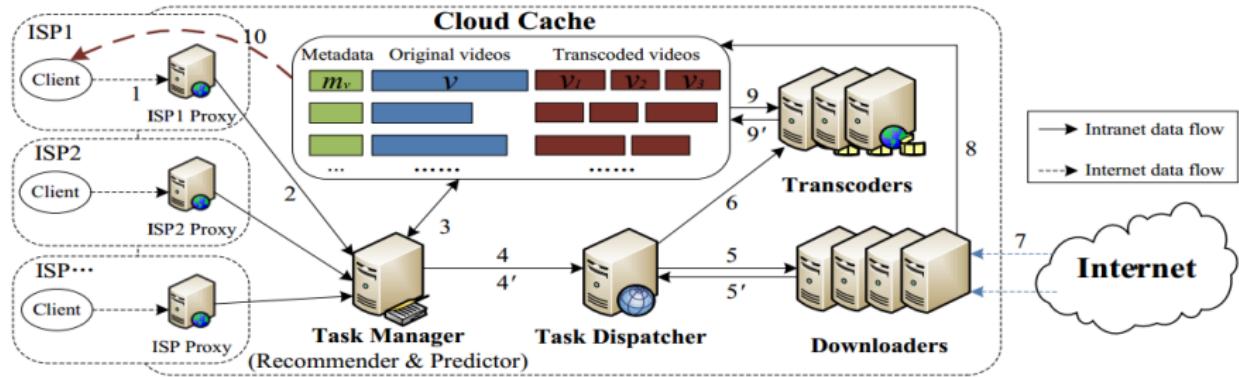


前腾讯研究院 Cloud Transcoder

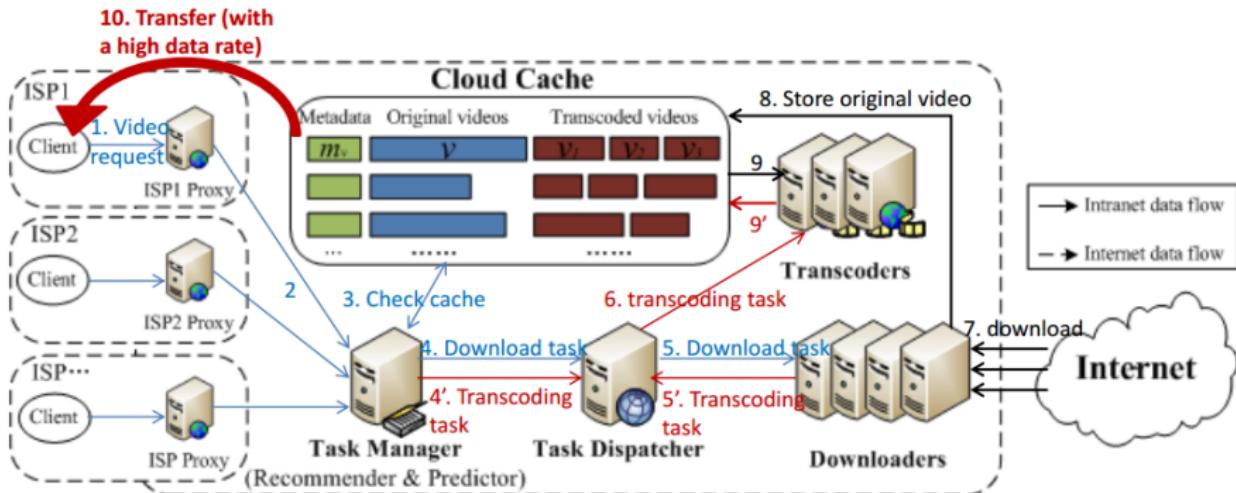
Done 2011-. Gale Huang et al. Cloud transcoder:
 bridging the format and resolution gap between internet
 videos and mobile devices. ACM NOSSDAV 2012.



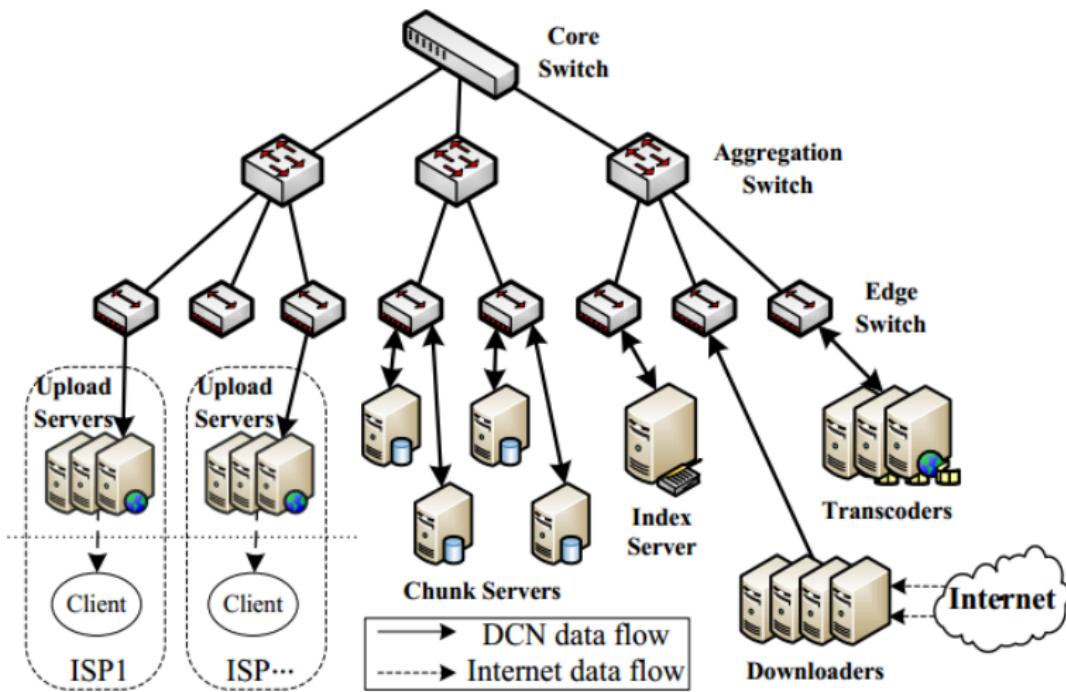
前腾讯研究院 Cloud Transcoder



前腾讯研究院 Cloud Transcoder



前腾讯研究院 Cloud Transcoder



评价

- 整体上是个设计优秀的系统

评价

- 整体上是个设计优秀的系统
- 但还存在一些问题：

评价

- 整体上是个设计优秀的系统
- 但还存在一些问题：
 - 下载、转码、任务分发、任务管理都需要专门的机器资源

评价

- 整体上是个设计优秀的系统
- 但还存在一些问题：
 - 下载、转码、任务分发、任务管理都需要专门的机器资源
 - 下载后，数据还需要再次传输到相应的转码机器

评价

- 整体上是个设计优秀的系统
- 但还存在一些问题：
 - 下载、转码、任务分发、任务管理都需要专门的机器资源
 - 下载后，数据还需要再次传输到相应的转码机器
 - 切片的开销

评价

- 整体上是个设计优秀的系统
- 但还存在一些问题：
 - 下载、转码、任务分发、任务管理都需要专门的机器资源
 - 下载后，数据还需要再次传输到相应的转码机器
 - 切片的开销
 - 划分的各模块，增加了部署复杂性，限制了可扩展性

评价

- 整体上是个设计优秀的系统
- 但还存在一些问题：
 - 下载、转码、任务分发、任务管理都需要专门的机器资源
 - 下载后，数据还需要再次传输到相应的转码机器
 - 切片的开销
 - 划分的各模块，增加了部署复杂性，限制了可扩展性
 - 存储部分依赖 NFS，不能支持海量的媒体资源

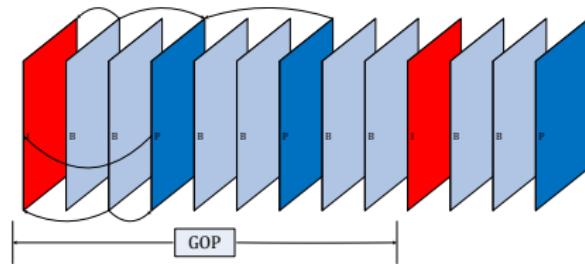
评价

- 整体上是个设计优秀的系统
- 但还存在一些问题：
 - 下载、转码、任务分发、任务管理都需要专门的机器资源
 - 下载后，数据还需要再次传输到相应的转码机器
 - 切片的开销
 - 划分的各模块，增加了部署复杂性，限制了可扩展性
 - 存储部分依赖 NFS，不能支持海量的媒体资源
 - 转码完成后，分发是个问题

架平流媒体 TranscX

TranscX:

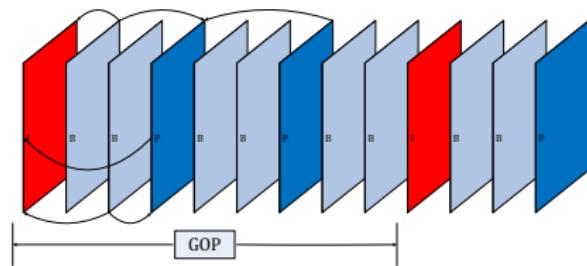
- Transcoding eXpress/eXperience/eXtreme...; transc(x)



架平流媒体 TranscX

TranscX:

- Transcoding eXpress/eXperience/eXtreme...; transc(x)



- GOP 级的并行，不需要真正“切割”

Media head parsing, $\langle \text{begin_time}, \text{end_time} \rangle$ or $\text{GOP_id} \rightarrow \langle \text{start_offset}, \text{size} \rangle$



Tencent 腾讯

■ B



架平流媒体 TranscX

■ B



■ P



■ I: IDR



Tencent 腾讯

■ B



■ B...



Tencent 腾讯

- Job/Map/Thread 三级精确控制, CPU & I/O limitation
避免影响存储集群的现网服务

- Job/Map/Thread 三级精确控制, CPU & I/O limitation
避免影响存储集群的现网服务
- 对 DASH 和实时在线直播的支持

- Job/Map/Thread 三级精确控制，CPU & I/O limitation

避免影响存储集群的现网服务

- 对 DASH 和实时在线直播的支持

- 计算向存储迁移：云计算的理想状态

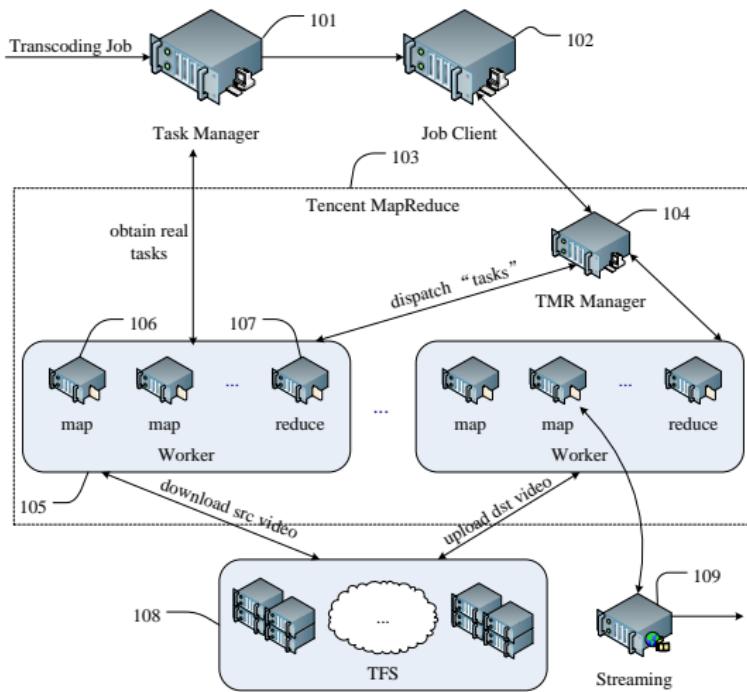
而且转码工作在存储机器上完成，避免了大部分的数据传输/拷贝，效果接近本地计算

- Job/Map/Thread 三级精确控制，CPU & I/O limitation
避免影响存储集群的现网服务
- 对 DASH 和实时在线直播的支持
- 计算向存储迁移：云计算的理想状态
而且转码工作在存储机器上完成，避免了大部分的数据传输/拷贝，效果接近本地计算
- 后来逐步支持了 WeChat、Qzone 和 Weishi

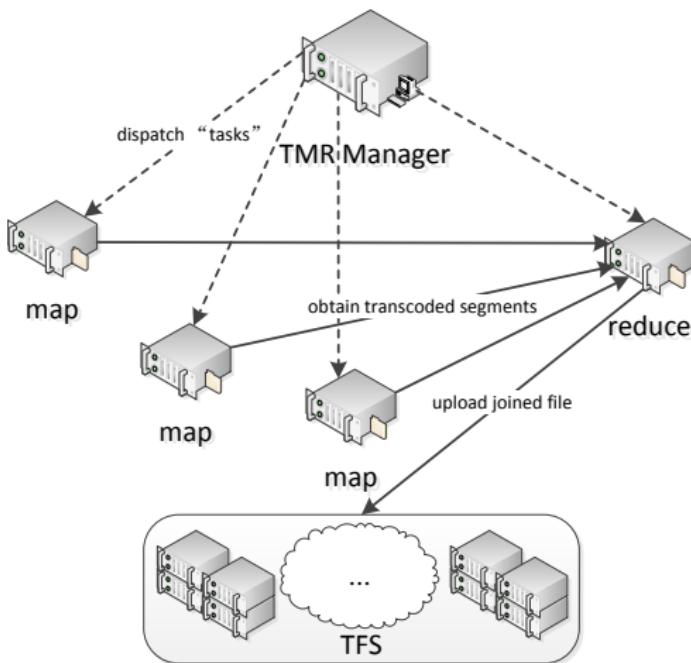
- Job/Map/Thread 三级精确控制，CPU & I/O limitation
避免影响存储集群的现网服务
- 对 DASH 和实时在线直播的支持
- 计算向存储迁移：云计算的理想状态
而且转码工作在存储机器上完成，避免了大部分的数据传输/拷贝，效果接近本地计算
- 后来逐步支持了 WeChat、Qzone 和 Weishi
- 用 MapReduce 框架统一了起来，减少了调度的复杂性，增强了可扩展性
通过 TFS/CFS 分布式文件系统和网络协议，可处理 PB 级别的媒体资源

- Job/Map/Thread 三级精确控制，CPU & I/O limitation
避免影响存储集群的现网服务
- 对 DASH 和实时在线直播的支持
- 计算向存储迁移：云计算的理想状态
而且转码工作在存储机器上完成，避免了大部分的数据传输/拷贝，效果接近本地计算
- 后来逐步支持了 WeChat、Qzone 和 Weishi
- 用 MapReduce 框架统一了起来，减少了调度的复杂性，增强了可扩展性
通过 TFS/CFS 分布式文件系统和网络协议，可处理 PB 级别的媒体资源
- 复用存储服务器的空间计算资源，减少碳排放（绿色计算）

架平流媒体 TranscX



架平流媒体 TranscX



Outline

1 背景

2 从 Cloud Transcoder 到 TranscX

- 前腾讯研究院 Cloud Transcoder
- 架平流媒体 TranscX

3 DASH 与稳定婚姻

- DASH
- 稳定匹配

4 Future Vision

- 和一些新技术的结合
- Acknowledgment

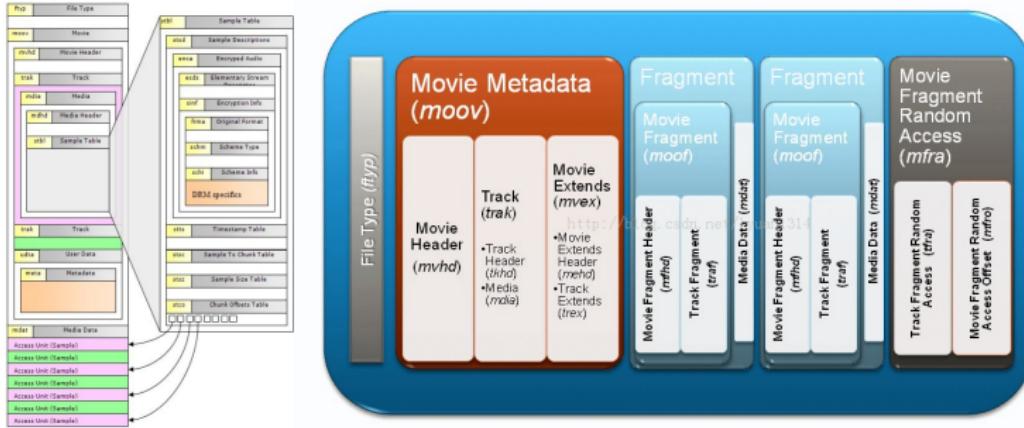
Why DASH?

DASH: Dynamic Adaptive Streaming over HTTP.

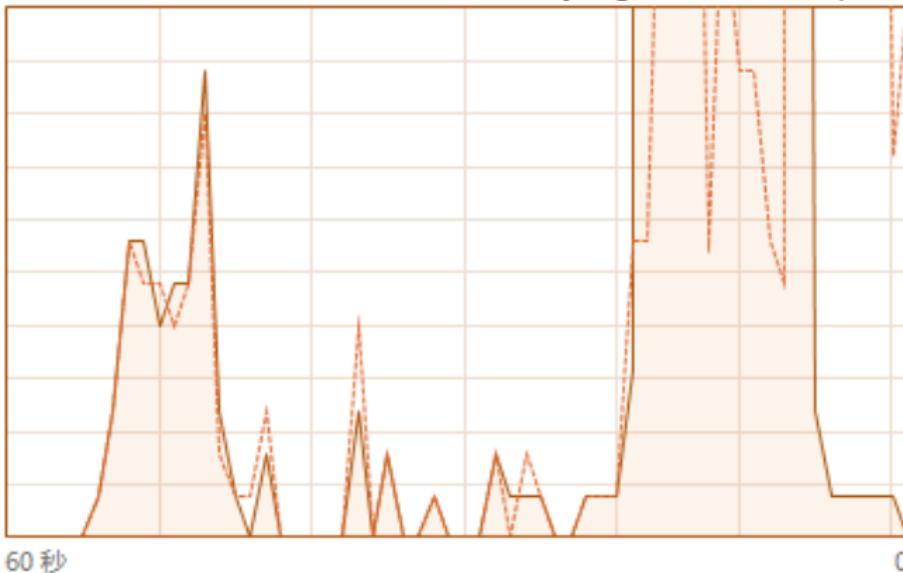
Why DASH?

DASH: Dynamic Adaptive Streaming over HTTP.

- 庞大的文件头，导致在线播放时较大的 initial/VCR delay



- 非平滑的码率间切换 due to varying download speed



DASH

DASH: Dynamic Adaptive Streaming over HTTP.

几种 DASH 标准：

DASH

DASH: Dynamic Adaptive Streaming over HTTP.

几种 DASH 标准：

- ## ■ Apple HLS (HTTP Live Streaming) 2009

DASH

DASH: Dynamic Adaptive Streaming over HTTP.

几种 DASH 标准：

- Apple HLS (HTTP Live Streaming) 2009
 - Microsoft HSS (HTTP Smooth Streaming) 2010

DASH

DASH: Dynamic Adaptive Streaming over HTTP.

几种 DASH 标准：

- Apple HLS (HTTP Live Streaming) 2009
 - Microsoft HSS (HTTP Smooth Streaming) 2010
 - Adobe HDS (HTTP Dynamic Streaming) 2010

DASH

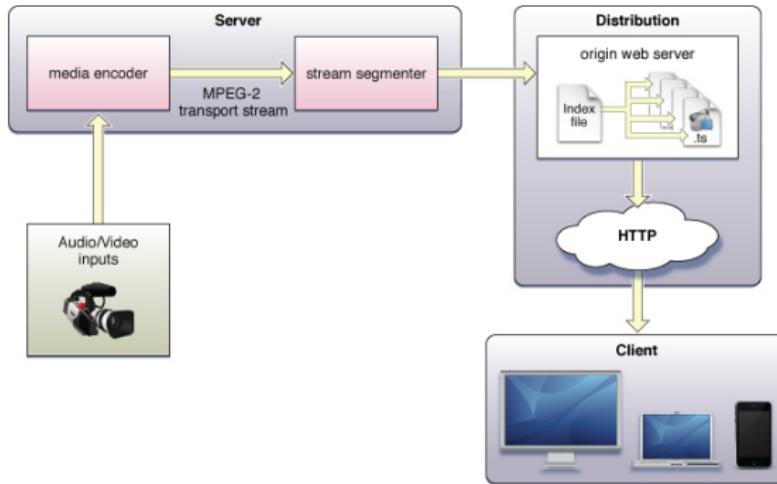
DASH: Dynamic Adaptive Streaming over HTTP.

几种 DASH 标准：

- Apple HLS (HTTP Live Streaming) 2009
 - Microsoft HSS (HTTP Smooth Streaming) 2010
 - Adobe HDS (HTTP Dynamic Streaming) 2010
 - MPEG-DASH (ISO/IEC 23009-1) 2012

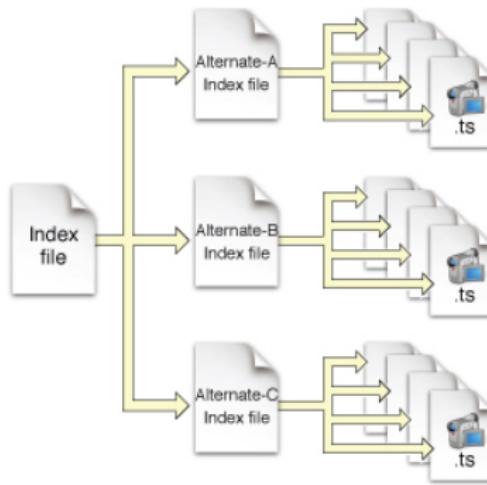
Apple HLS

■ Architecture



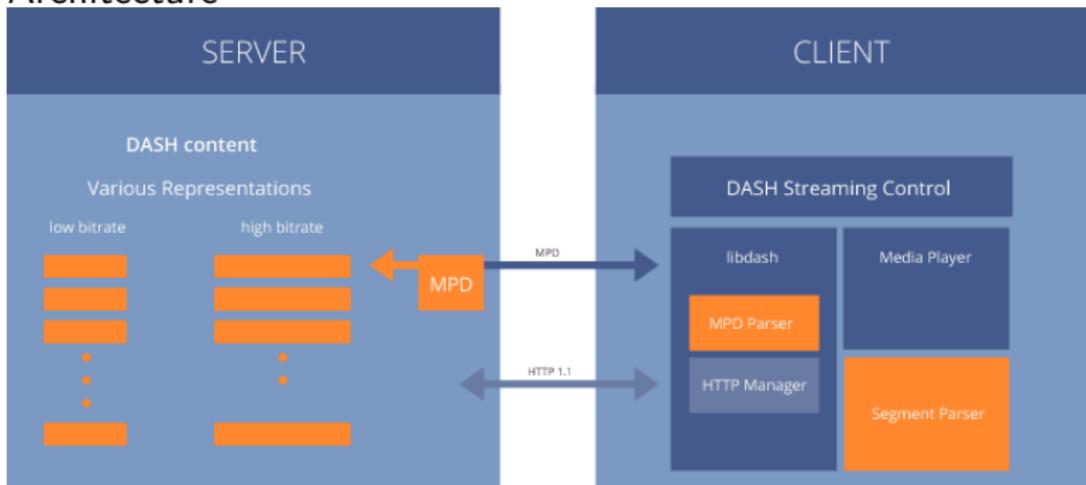
Apple HLS

■ Segment Indexing



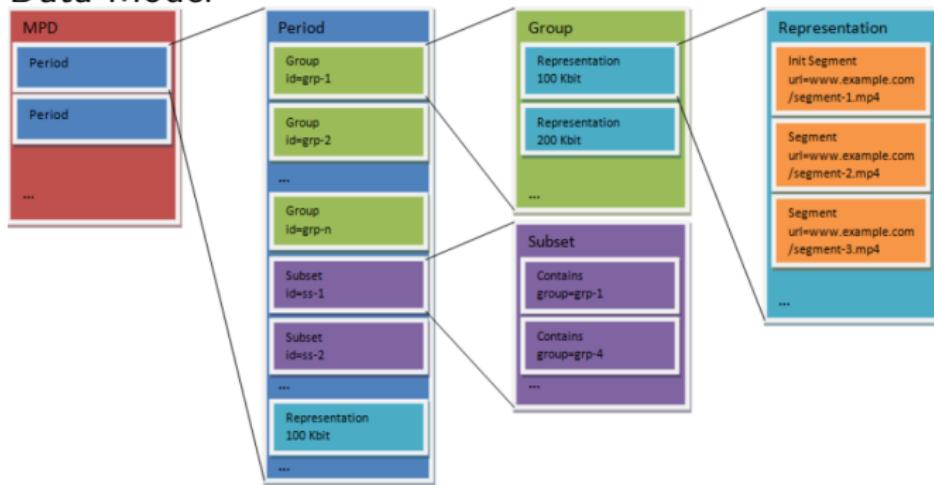
MPEG-DASH

■ Architecture



MPEG-DASH

■ Data Model



MPEG-DASH

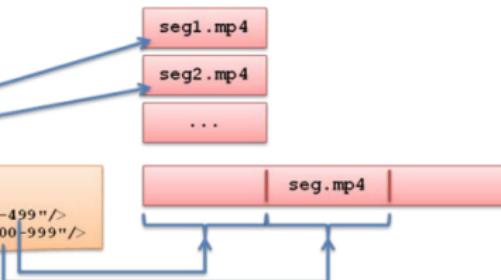
■ Segment Indexing

Segment Index in MPD only

```
<MPD>
...
<URL sourceURL="seg1.mp4"/>
<URL sourceURL="seg2.mp4"/>
</MPD>
```

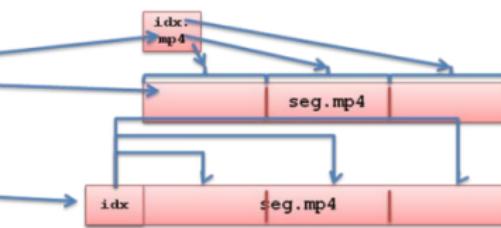


```
<MPD>
...
<URL sourceURL="seg.mp4" range="0-499"/>
<URL sourceURL="seg.mp4" range="500-999"/>
</MPD>
```



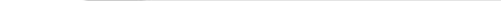
Segment Index in MPD + Segment

```
<MPD>
...
<Index sourceURL="idx.mp4"/>
<URL sourceURL="seg.mp4"/>
</MPD>
```



Segment Index in Segment only

```
<MPD>
...
<BaseURL>seg.mp4</BaseURL>
</MPD>
```



架平流媒体对 DASH 的使用

- 尽量只转封装，不转编码

架平流媒体对 DASH 的使用

- 尽量只转封装，不转编码
- 实时、按需转封装/转码

架平流媒体对 DASH 的使用

- 尽量只转封装，不转编码
- 实时、按需转封装/转码
- 使用/自写的多种协议，解决并发 I/O 流少的问题

架平流媒体对 DASH 的使用

- 尽量只转封装，不转编码
- 实时、按需转封装/转码
- 使用/自写的多种协议，解决并发 I/O 流少的问题

`tmpfs` 最简单、方便，对 Cache 管理要求不强的场合

架平流媒体对 DASH 的使用

- 尽量只转封装，不转编码
- 实时、按需转封装/转码
- 使用/自写的多种协议，解决并发 I/O 流少的问题

`tmpfs` 最简单、方便，对 Cache 管理要求不强的场合

`in_mem` in memory, 进程内的编解码 I/O

架平流媒体对 DASH 的使用

- 尽量只转封装，不转编码
- 实时、按需转封装/转码
- 使用/自写的多种协议，解决并发 I/O 流少的问题

`tmpfs` 最简单、方便，对 Cache 管理要求不强的场合

`in_mem` in memory, 进程内的编解码 I/O

`shm_mem` shared memory, 多进程的编解码 I/O，可自定义 Cache 管理、淘汰算法

架平流媒体对 DASH 的使用

- 尽量只转封装，不转编码
 - 实时、按需转封装/转码
 - 使用/自写的多种协议，解决并发 I/O 流少的问题

`tmpfs` 最简单、方便，对 Cache 管理要求不强的场合

`in_mem` in memory, 进程内的编解码 I/O

`shm_mem` shared memory, 多进程的编解码 I/O, 可自定义 Cache 管理、淘汰算法

AVIOContext FFMPEG built-in

架平流媒体对 DASH 的使用

- 尽量只转封装，不转编码
 - 实时、按需转封装/转码
 - 使用/自写的多种协议，解决并发 I/O 流少的问题

`tmpfs` 最简单、方便，对 Cache 管理要求不强的场合

`in_mem` in memory, 进程内的编解码 I/O

`shm_mem` shared memory, 多进程的编解码 I/O, 可自定义 Cache 管理、淘汰算法

AVIOContext FFMPEG built-in

`async_http` 适用直播，数据到达不可预期

架平流媒体对 DASH 的使用

- 尽量只转封装，不转编码
 - 实时、按需转封装/转码
 - 使用/自写的多种协议，解决并发 I/O 流少的问题

`tmpfs` 最简单、方便，对 Cache 管理要求不强的场合

`in_mem` in memory, 进程内的编解码 I/O

shm_mem shared memory, 多进程的编解码 I/O, 可自定义 Cache 管理、淘汰算法

`AVIOContext` FFmpeg built-in

`async_http` 适用直播，数据到达不可预期

- #### ■ FPGA HEVC Encoding (proposed)

Motivation

DASH 和 Wechat, Weishi, Qzone 等社交 UGC 视频转码中存在的问题：

- 视频长度较短

一般仅一个到若干个 GOP，MapReduce 时仅需要一个 Mapper

Motivation

DASH 和 Wechat, Weishi, Qzone 等社交 UGC 视频转码中存在的问题：

- 视频长度较短

一般仅一个到若干个 GOP，MapReduce 时仅需要一个 Mapper

- 但 MapReduce 任务调度的延时较大

有时甚至超过播放时长。可通过 Linux Container, Docker 守护进程的实时 MapReduce 来缓解，但非根本解决之道

Motivation

DASH 和 Wechat, Weishi, Qzone 等社交 UGC 视频转码中存在的问题：

- 视频长度较短

一般仅一个到若干个 GOP，MapReduce 时仅需要一个 Mapper

- 但 MapReduce 任务调度的延时较大

有时甚至超过播放时长。可通过 Linux Container, Docker 守护进程的实时 MapReduce 来缓解，但非根本解决之道

- 大部分视频的未来访问少于 X 次

还有比较全码率转码 + 全量分发么？

大数据分析，得出结论：

- 对数千台后台服务器连续一周运行状态的测量：

大数据分析，得出结论：

- 对数千台后台服务器连续一周运行状态的测量：
 - 不仅 DC, OC 的多数服务器大部分时间 CPU 负载也较低甚至包括某些 HTTP 服务器

大数据分析，得出结论：

- 对数千台后台服务器连续一周运行状态的测量：
 - 不仅 DC, OC 的多数服务器大部分时间 CPU 负载也较低甚至包括某些 HTTP 服务器
 - 多数服务器的 CPU 负载基本不会出现突变CPU 负载随时间相对平稳，但不同地区的平均负载存在差异

大数据分析，得出结论：

- 对数千台后台服务器连续一周运行状态的测量：
 - 不仅 DC, OC 的多数服务器大部分时间 CPU 负载也较低甚至包括某些 HTTP 服务器
 - 多数服务器的 CPU 负载基本不会出现突变CPU 负载随时间相对平稳，但不同地区的平均负载存在差异
- 对应时间所有用户访问和调度情况的测量：

大数据分析，得出结论：

- 对数千台后台服务器连续一周运行状态的测量：
 - 不仅 DC, OC 的多数服务器大部分时间 CPU 负载也较低
甚至包括某些 HTTP 服务器
 - 多数服务器的 CPU 负载基本不会出现突变
CPU 负载随时间相对平稳，但不同地区的平均负载存在差异
- 对应时间所有用户访问和调度情况的测量：
 - 用户对 CDN 的 region 选择存在偏好
由于网络拓扑关系，用户去不同的 CDN 边缘节点下载的速度各不相同，而目前的调度算法，至少能得到一个次优解

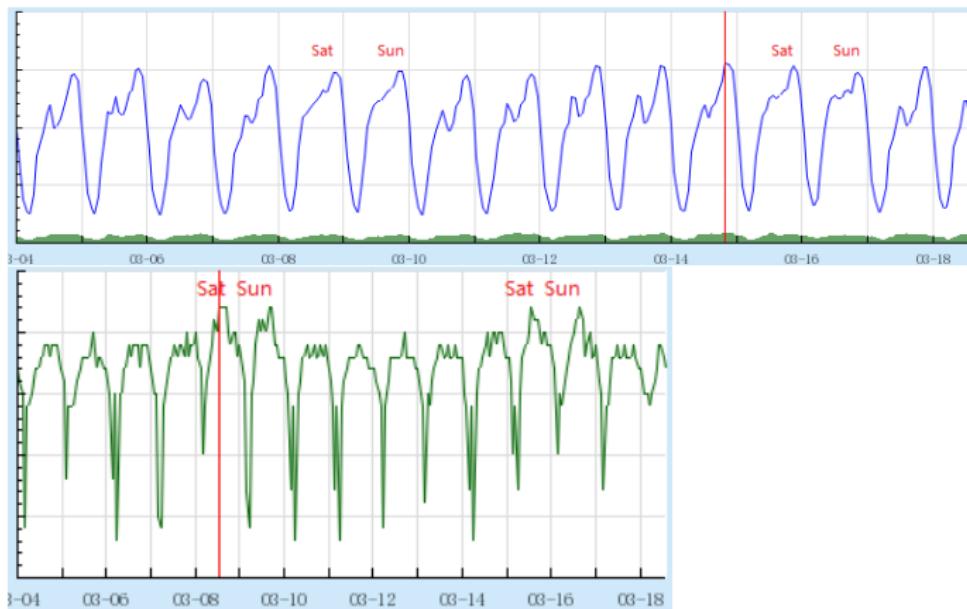
大数据分析，得出结论：

- 对数千台后台服务器连续一周运行状态的测量：
 - 不仅 DC, OC 的多数服务器大部分时间 CPU 负载也较低
甚至包括某些 HTTP 服务器
 - 多数服务器的 CPU 负载基本不会出现突变
CPU 负载随时间相对平稳，但不同地区的平均负载存在差异
- 对应时间所有用户访问和调度情况的测量：
 - 用户对 CDN 的 region 选择存在偏好
由于网络拓扑关系，用户去不同的 CDN 边缘节点下载的速度各不相同，而目前的调度算法，至少能得到一个次优解
 - 不同不同地区的用户对码率存在偏好
在前面次优解的前提下，不同地区得到平均服务质量有差异

Tencent 腾讯

稳定匹配

带宽和 CPU 使用率示例



启发

- CDN 的大部分服务器是否也可用来做转码？

启发

- CDN 的大部分服务器是否也可用来做转码？
- 是否可以按需转码，按需分发？
on-line, on-the-fly, on-demand?

启发

- CDN 的大部分服务器是否也可用来做转码？
- 是否可以按需转码，按需分发？
on-line, on-the-fly, on-demand?
- 当前 CDN 节点实在是没有所需码率片段时...
是否可以给一个不高于所需码率的最高码率副本 来替代？

优化问题

■ 用户重定向

用户当前请求应该被调度到哪个 CDN 节点来服务？资源应尽量在哪儿转码？服务器负载、带宽利用、用户体验，哪个更重要？

优化问题

■ 用户重定向

用户当前请求应该被调度到哪个 CDN 节点来服务？资源应尽量在哪儿转码？服务器负载、带宽利用、用户体验，哪个更重要？

■ 哪些资源的哪些片段最需要被转码

用户最需要的，能尽量让用户得到最大码率的，当前副本最少的？消耗服务器计算资源少的（节能/碳环保）？

优化问题

■ 用户重定向

用户当前请求应该被调度到哪个 CDN 节点来服务？资源应尽量在哪儿转码？服务器负载、带宽利用、用户体验，哪个更重要？

■ 哪些资源的哪些片段最需要被转码

用户最需要的，能尽量让用户得到最大码率的，当前副本最少的？消耗服务器计算资源少的（节能/碳环保）？

■ 转码后的片段应当怎样在 CDN 云中分发

怎样定义分发代价？怎样让代价最小？怎样尽可能不超出购买带宽量？

我们的启发式算法

- 启发式规则和稳定匹配（婚姻）结合

我们的启发式算法

- 启发式规则和稳定匹配（婚姻）结合
 - 启发式：用贪心规则给出 NP 难问题的近似最优解

我们的启发式算法

- 启发式规则和稳定匹配（婚姻）结合
 - 启发式：用贪心规则给出 NP 难问题的近似最优解
 - 稳定匹配：扩展的稳定婚姻问题

我们的启发式算法

- 启发式规则和稳定匹配（婚姻）结合
 - 启发式：用贪心规则给出 NP 难问题的近似最优解
 - 稳定匹配：扩展的稳定婚姻问题
 - 女追男 — 尽量利于女神
 - 女：用户；男：CDN 边缘节点

我们的启发式算法

- 启发式规则和稳定匹配（婚姻）结合
 - 启发式：用贪心规则给出 NP 难问题的近似最优解
 - 稳定匹配：扩展的稳定婚姻问题
 - 女追男 — 尽量利于女神
女：用户；男：CDN 边缘节点
 - 男方具有多容量

我们的启发式算法

- 启发式规则和稳定匹配（婚姻）结合
 - 启发式：用贪心规则给出 NP 难问题的近似最优解
 - 稳定匹配：扩展的稳定婚姻问题
 - 女追男 — 尽量利于女神
女：用户；男：CDN 边缘节点
 - 男方具有多容量
 - 稳定性与帕累托效率

算法应用

准备与预测框架

- 用户对所有 CDN 边缘节点的偏好

根据过去的测速数据来 Rank 用户对所有 CDN 边缘节点的偏好

算法应用

准备与预测框架

- 用户对所有 CDN 边缘节点的偏好

根据过去的测速数据来 Rank 用户对所有 CDN 边缘节点的偏好

- 每段视频未来一段时间的访问频率

根据 CDN 到用户的带宽和过去一段时间用户的访问频次预测未来一段时间所有有可能有访问的视频的访问频次

算法应用

准备与预测框架

- 用户对所有 CDN 边缘节点的偏好

根据过去的测速数据来 Rank 用户对所有 CDN 边缘节点的偏好

- 每段视频未来一段时间的访问频率

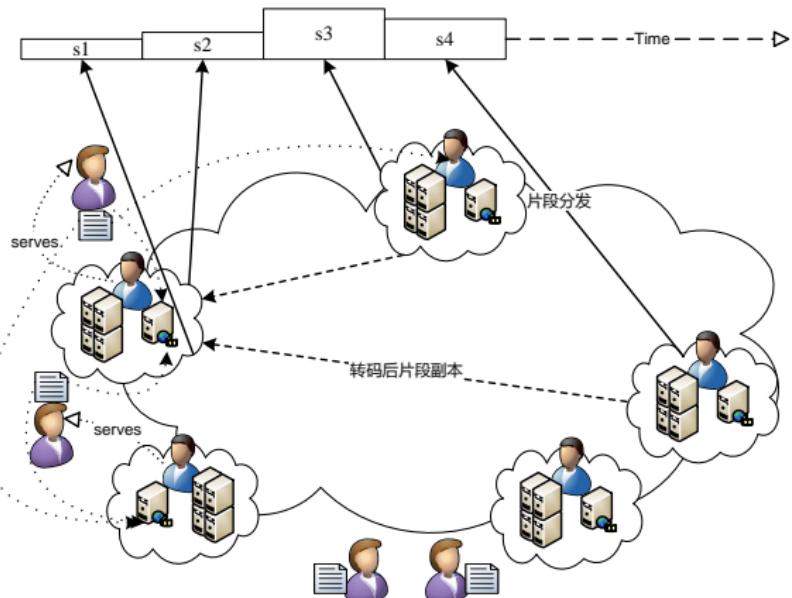
根据 CDN 到用户的带宽和过去一段时间用户的访问频次预测未来一段时间所有有可能有访问的视频的访问频次

- 未来一小段时间服务器的期望空间计算资源

根据分工作日/周末两种自回归模型预测未来一段时间每个 OC 节点服务器的期望 CPU 负载

稳定匹配

整体架构与“女追男”模型



至关重要：Satisfying Women



效果

- 44.8% 的用户会享受更高码率的版本
相比传统服务器负载均衡的算法

效果

- 44.8% 的用户会享受更高码率的版本
相比传统服务器负载均衡的算法
- 4.5 倍的用户享受到最高的可能码率
相比传统服务器负载均衡的算法

效果

- 44.8% 的用户会享受更高码率的版本
相比传统服务器负载均衡的算法
- 4.5 倍的用户享受到最高的可能码率
相比传统服务器负载均衡的算法
- 减少 42.2% 接受和当前带宽不匹配码率的可能
相比传统 FIFO（先来先服务）的算法

效果

- 44.8% 的用户会享受更高码率的版本
相比传统服务器负载均衡的算法
- 4.5 倍的用户享受到最高的可能码率
相比传统服务器负载均衡的算法
- 减少 42.2% 接受和当前带宽不匹配码率的可能
相比传统 FIFO（先来先服务）的算法
- 减少约 80% 的转码计算资源消耗
对于微视、腾讯视频这种 4 码率副本的场景。码率副本数越高，效果越明显。（Qzone 两码率副本场合，减少约 40%）

效果

- 44.8% 的用户会享受更高码率的版本
相比传统服务器负载均衡的算法
- 4.5 倍的用户享受到最高的可能码率
相比传统服务器负载均衡的算法
- 减少 42.2% 接受和当前带宽不匹配码率的可能
相比传统 FIFO（先来先服务）的算法
- 减少约 80% 的转码计算资源消耗
对于微视、腾讯视频这种 4 码率副本的场景。码率副本数越高，效果越明显。（Qzone 两码率副本场合，减少约 40%）
- 副本传输的带宽不随分段数显著增长

Publications



Publications



...Alan Zhuang... Joint Online Transcoding and Geo-distributed Delivery for Dynamic Adaptive Streaming. IEEE Transactions on Parallel and Distributed Systems (TPDS). 2014. (will appear)

Publications



...Alan Zhuang... Joint Online Transcoding and Geo-distributed Delivery for Dynamic Adaptive Streaming. IEEE Transactions on Parallel and Distributed Systems (TPDS). 2014. (will appear)



Publications



...Alan Zhuang... Joint Online Transcoding and Geo-distributed Delivery for Dynamic Adaptive Streaming. IEEE Transactions on Parallel and Distributed Systems (TPDS). 2014. (will appear)



Outline

1 背景

2 从 Cloud Transcoder 到 TranscX

- 前腾讯研究院 Cloud Transcoder
- 架平流媒体 TranscX

3 DASH 与稳定婚姻

- DASH
- 稳定匹配

4 Future Vision

- 和一些新技术的结合
- Acknowledgment

和一些新技术的结合

和一些新技术的结合

■ WebRTC

Client-end caching & delivery & broadcasting

和一些新技术的结合

和一些新技术的结合

- WebRTC
 - Client-end caching & delivery & broadcasting
- HTML5 WebSocket, HTTP 2.0
 - 会话保持, 多流传输

和一些新技术的结合

和一些新技术的结合

- WebRTC
 - Client-end caching & delivery & broadcasting
- HTML5 WebSocket, HTTP 2.0
 - 会话保持, 多流传输
- Social Could TV

和一些新技术的结合

和一些新技术的结合

- WebRTC
 - Client-end caching & delivery & broadcasting
- HTML5 WebSocket, HTTP 2.0
 - 会话保持, 多流传输
- Social Could TV
- SVC, MVC, HEVC, NC
 - 编码、多副本多版本机制、多视角、传输

Acknowledgment

Acknowledgment

本 slides 中部分内容源自与以下人员的协作/交流：

- 架平流媒体

Leon Ouyang, Devin Zeng, Guita Zhang, Kernel He, Chris Su

Acknowledgment

Acknowledgment

本 slides 中部分内容源自与以下人员的协作/交流：

- 架平流媒体

Leon Ouyang, Devin Zeng, Guita Zhang, Kernel He, Chris Su

- Tsinghua-Tencent Joint Lab.

Zhi Wang

Acknowledgment

Acknowledgment

本 slides 中部分内容源自与以下人员的协作/交流：

- 架平流媒体

Leon Ouyang, Devin Zeng, Guita Zhang, Kernel He, Chris Su

- Tsinghua-Tencent Joint Lab.

Zhi Wang

- SNG 社交平台部

Stone Huang

Acknowledgment

Acknowledgment

本 slides 中部分内容源自与以下人员的协作/交流：

- 架平流媒体

Leon Ouyang, Devin Zeng, Guita Zhang, Kernel He, Chris Su

- Tsinghua-Tencent Joint Lab.

Zhi Wang

- SNG 社交平台部

Stone Huang

- NTU

Young Wen

Tencent 腾讯