

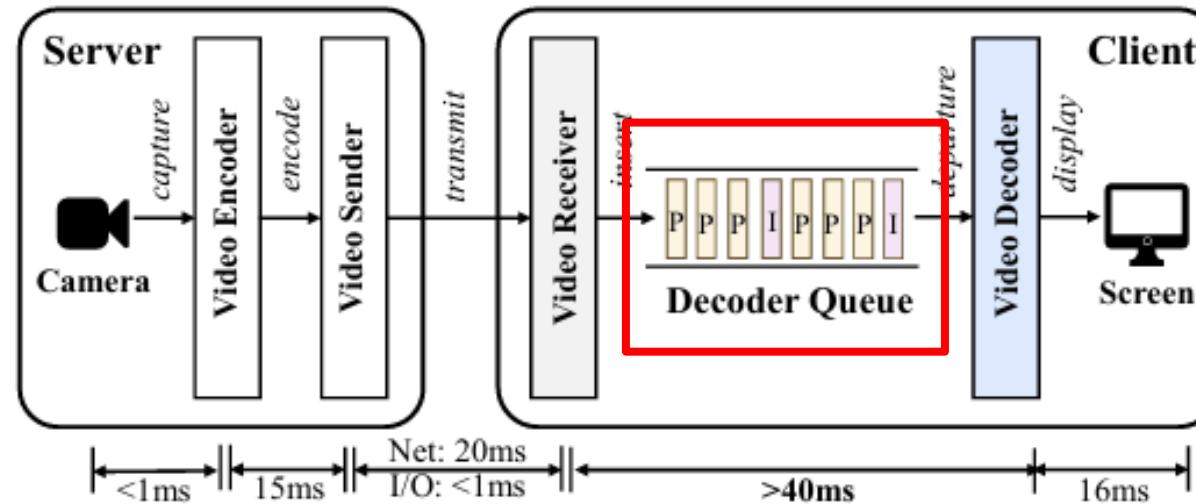
A wide-angle photograph of a natural landscape. In the foreground, there's a body of water with a small, dark island or peninsula extending into it. The middle ground shows green, hilly terrain. In the background, a range of mountains is visible, with several peaks reaching high into a sky filled with large, billowing clouds. The lighting suggests either sunrise or sunset, with warm colors like orange and yellow highlighting parts of the mountains and clouds.

2024/07/15 Research Report

RTCL Lab Meeting  
Presenter: Chaehoon Park

# About Paper

## ■ Problem Definition



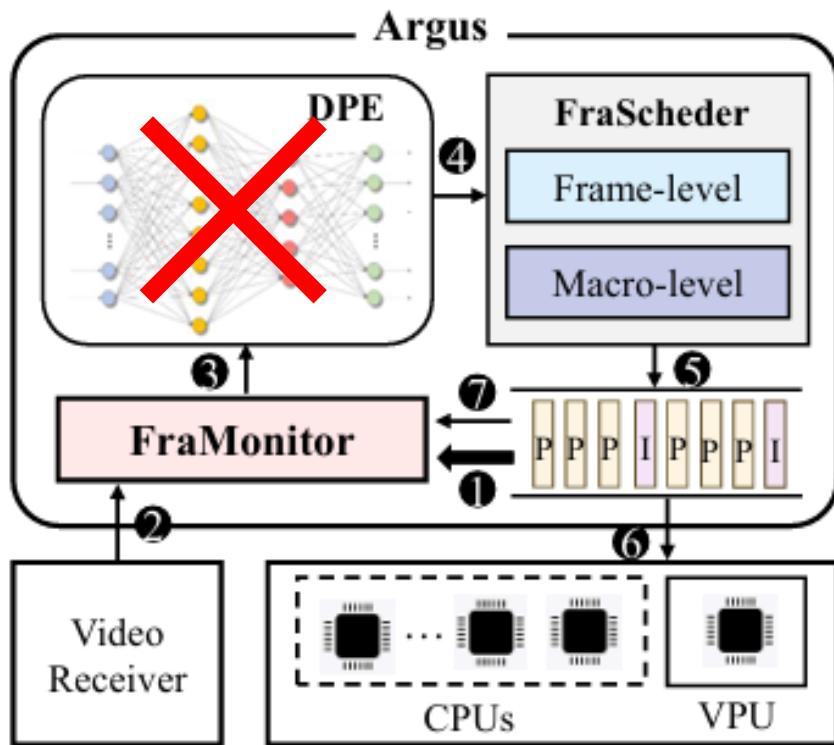
The encoding delay is light since videos are processed on commercial servers.

With the development of communication techniques and I/O protocols (e.g., 5G/6G, WiFi-6), the frames can be delivered within tens of microseconds

→ Bottleneck is located at the decoding phase.

# About Paper

## Architecture of Argus



*“Argus: Real-Time HQ Video Decoding with CPU Coordinating on Consumer Devices”*

CPU resources are not fully used when the VPU is busy

Argus: new real-time HQ video decoding solution migrates some frames to the CPU for decoding when the decoder queue is congested.

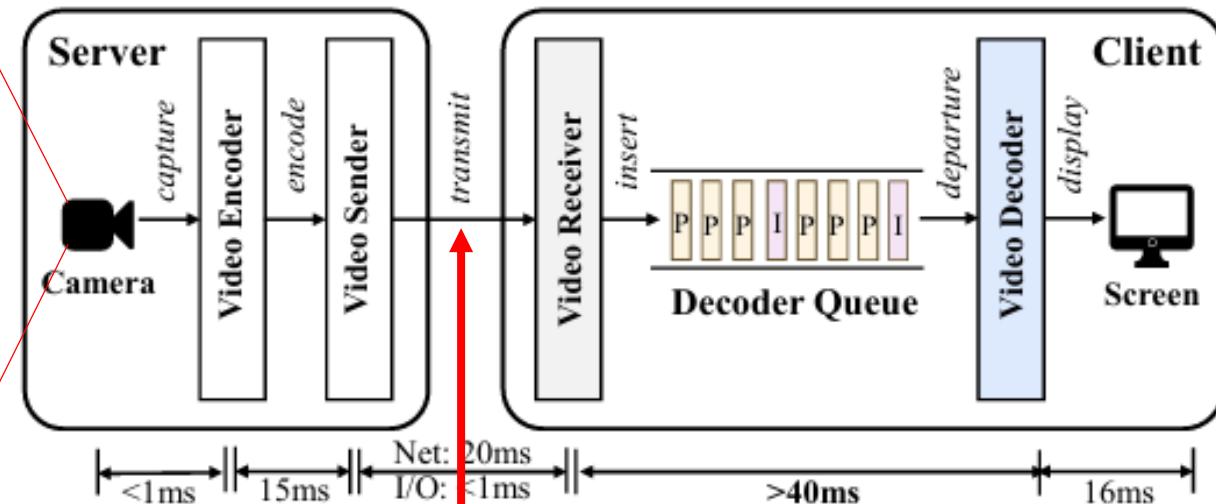
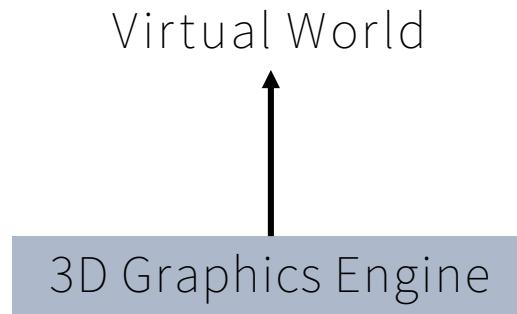
**DPE (Intelligent decoder pressure estimation model)**

: Identify the decoder queue busy state as early as possible  
→ Take “Queue length > Threshold” as busy condition

**FraScheduler (Frame-characteristics aware scheduler)**

: Offloads several frames to CPUs to alleviate the pressure

# Architecture

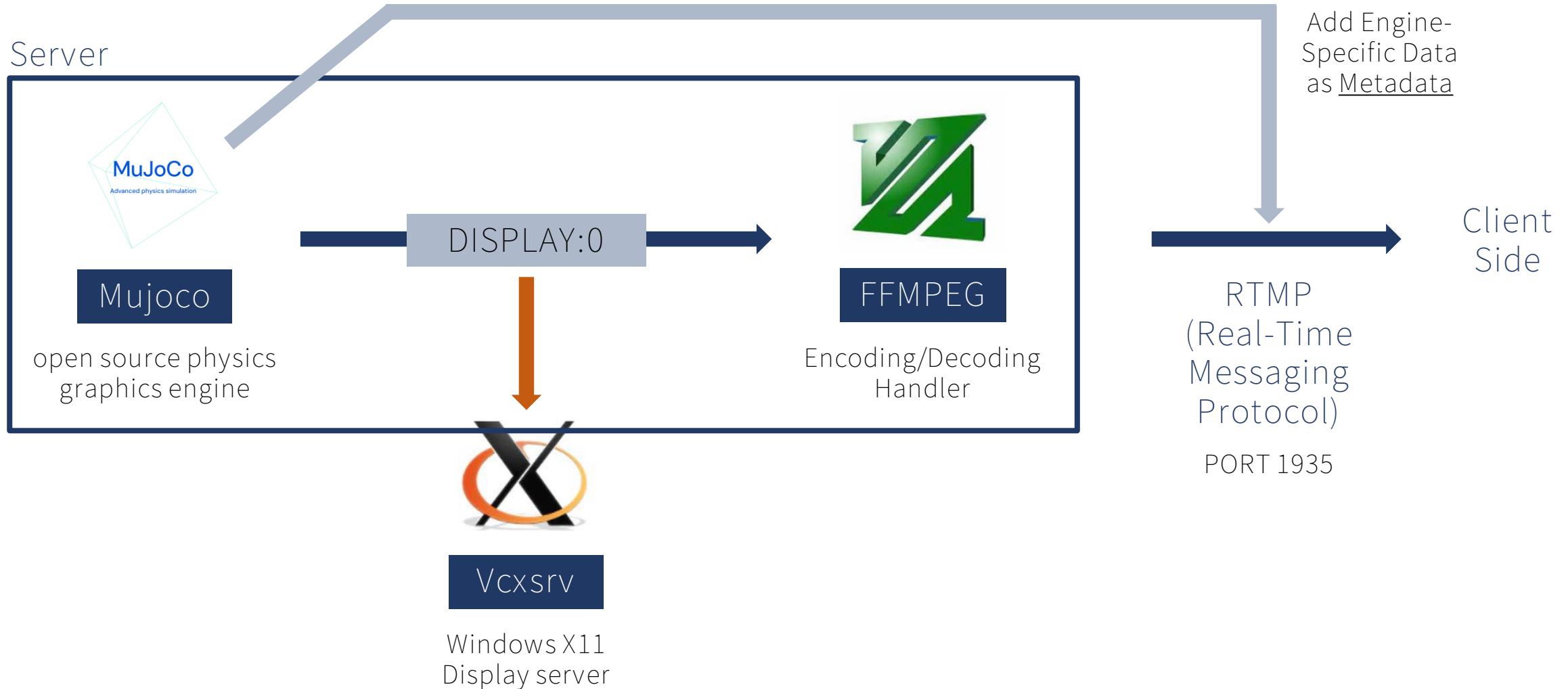


Offer Additional Information about frames (or macroblocks) that can help

- (1) Identifying decoder pressure
- (2) Judging which frames (or macroblocks) to schedule in parallel

# Architecture

## Server Side



# Architecture

## ■ Additional Information that 3D graphics engine can offer

### 1) Decoding Delay Estimation

~~Add DPE neural network input features~~

Serve as scheduling timing in FraScheder

(a) Frames with more complicated contents require more macroblock subdivisions in these details, leading to increased decoding computational complexity.

- Number of distinct objects rendered on the screen → Not yet..
- Frame Complexity Value: ex)  $\Sigma ((\text{Object Texture Complexity}) \times (\text{Rendered Area Size}))$
- Object Texture Complexity: pre-calculated number of macroblocks consisting texture

(b) Frames with significant motion or changes require more motion vectors, thus increasing decoding time.

- Camera velocity & angular velocity (+ acceleration, etc.)
- Peripheral Object Transition Value: ex)  $\Sigma ((\text{Object Velocity}) \times (\text{Rendered Area Size}))$

# Architecture

## Client Side

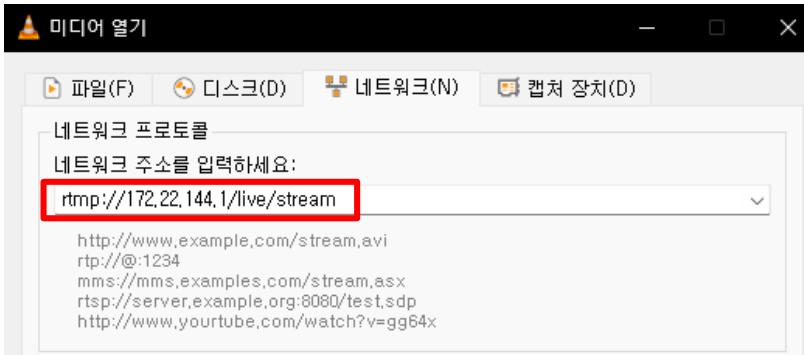


### VLC media player

VLC는 DVD, 오디오 CD, VCD 와 다양한 스트리밍 프로토콜뿐 아니라 대부분의 멀티미디어 파일을 재생할 수 있는 무료 오픈 소스 크로스 플랫폼 멀티미디어 재생기기자 프레임워크입니다.

다운로드 VLC

Version 3.0.21 • Windows 64bit • 40 MB  
42,984,319 다운로드 한정



http://www.example.com/stream.avi  
rtsp://@:1234  
mms://mms.examples.com/stream.asx  
rtsp://server.example.org:8080/test.sdp  
http://www.youtube.com/watch?v=gg64x



VLC

master vlc

extras/tools: ignore downloaded+patched config.guess  
Steve Lhomme authored 5 days ago

History Find file Code

e4ea36ba

Name	Last commit	Last update
autotools	gettext: update to 0.21	2 years ago
bin	meson: create vlc-cache-gen & plugins...	2 months ago
buildsystem	buildsystem: cargo-test: add cargo test...	3 weeks ago
compat	stdckdint: add compatibility header	4 months ago
contrib	contrib: rav1e: use vlc_build for the co...	6 hours ago
doc	doc: standalone: add documentation ar...	2 weeks ago
extras	extras/tools: ignore downloaded+patch...	6 hours ago
include	player: support vout_order in vlc_player...	1 week ago

Project information  
VLC: the ultimate media player  
vlc multimedia player + 1 more  
pipeline passed  
103,224 Commits  
2 Branches  
100 Tags  
README  
GNU GPLv2.1  
CHANGELOG  
GitLab Pages  
Created on  
January 09, 2019

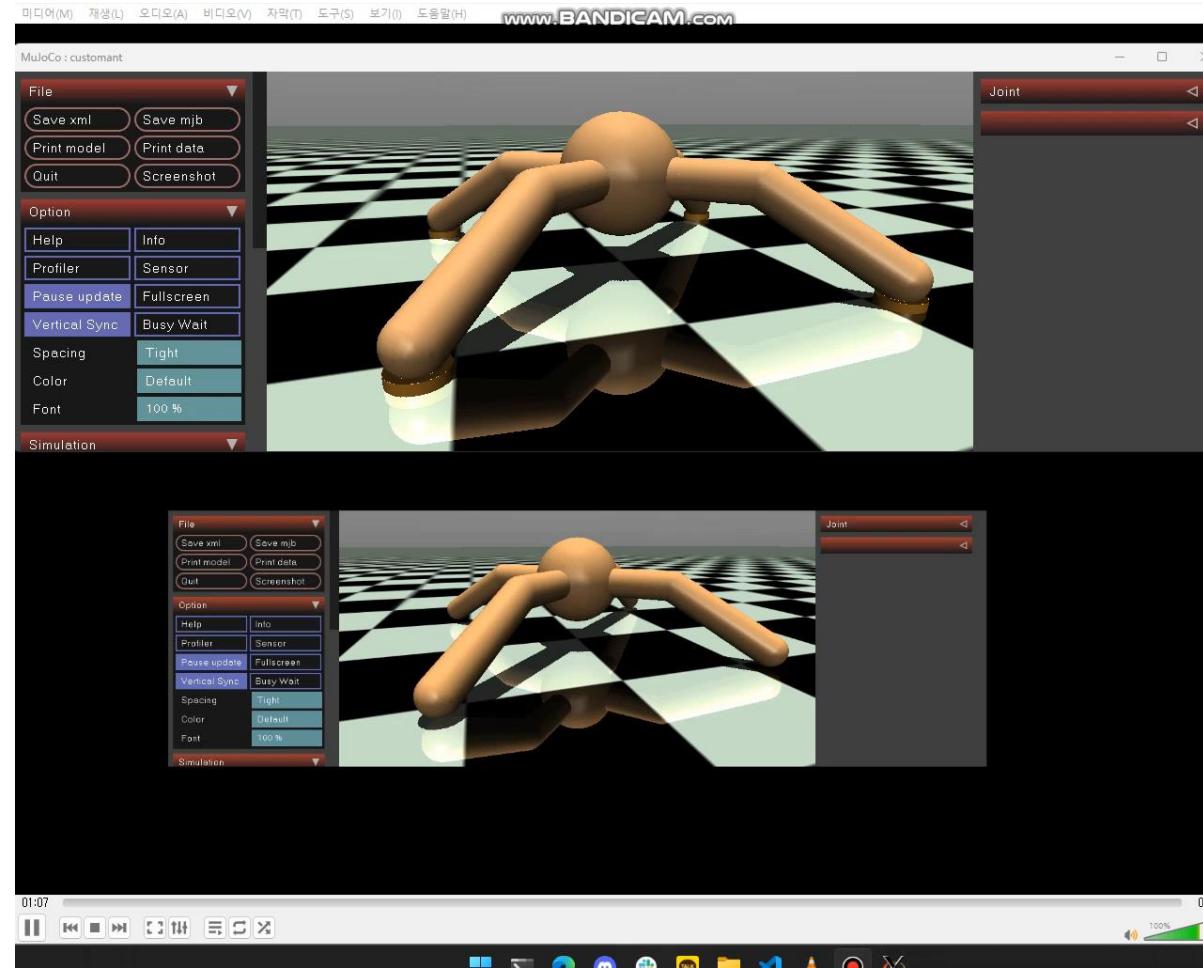
open source...

# Demo

## ■ Baseline Demo

Server Side

Client Side



5 sec delay...

# Demo

## ■ Baseline Demo

```
Stream #0:0: Video: rawvideo (BGR[0] / 0x524D414E) s, 29.97 fps, 1000k tbr, 1000k tbn
Stream mapping:
Stream #0:0 -> #0:0 (rawvideo (native) -> flv1 (flv))
Press [q] to stop, [?] for help
Output #0, flv, to 'rtmp://localhost/live/stream':
Metadata:
encoder      : Lavf61.5.101
Stream #0:0: Video: flv1 ([2][0][0][0] / 0x0002), yuv420p(progressive), 1920x1080, q=2-31, 3000 kb/s, 29.97 fps, 1k tbn
Metadata:
encoder      : Lavc61.10.100 flv
Side data:
cpb: bitrate max/min/avg: 0/0/3000000 buffer size: 0 vbv_delay: N/A
Frame= 7452 fps= 29 q=2.0 size= 44674KiB time=00:04:16.15 bitrate=1428.7kbits/s dup=0 drop=225 speed= 1x
```

## FFMPEG live streaming

Encoder: Lavf61.5.101 (planning to change into H.264)

Fps: 29

Q: 2.0 (Quality factor. Lower means better bitrate)

Average Bitrate: 1416.2kbits/s

Dup: 0

Drop: 225

Assume no temporal, spatial locality used while transmission

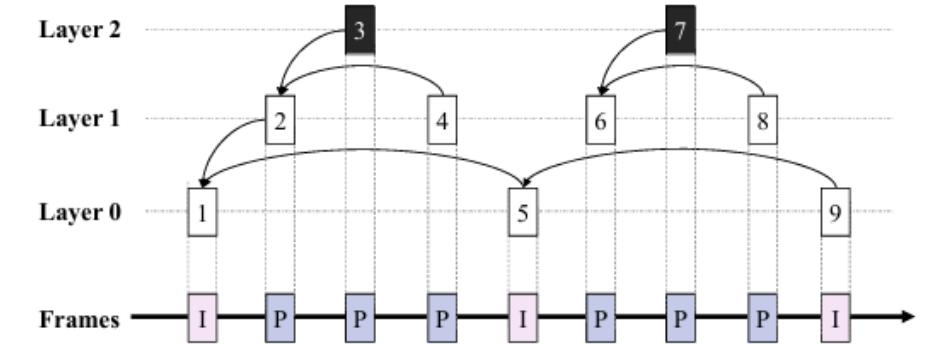
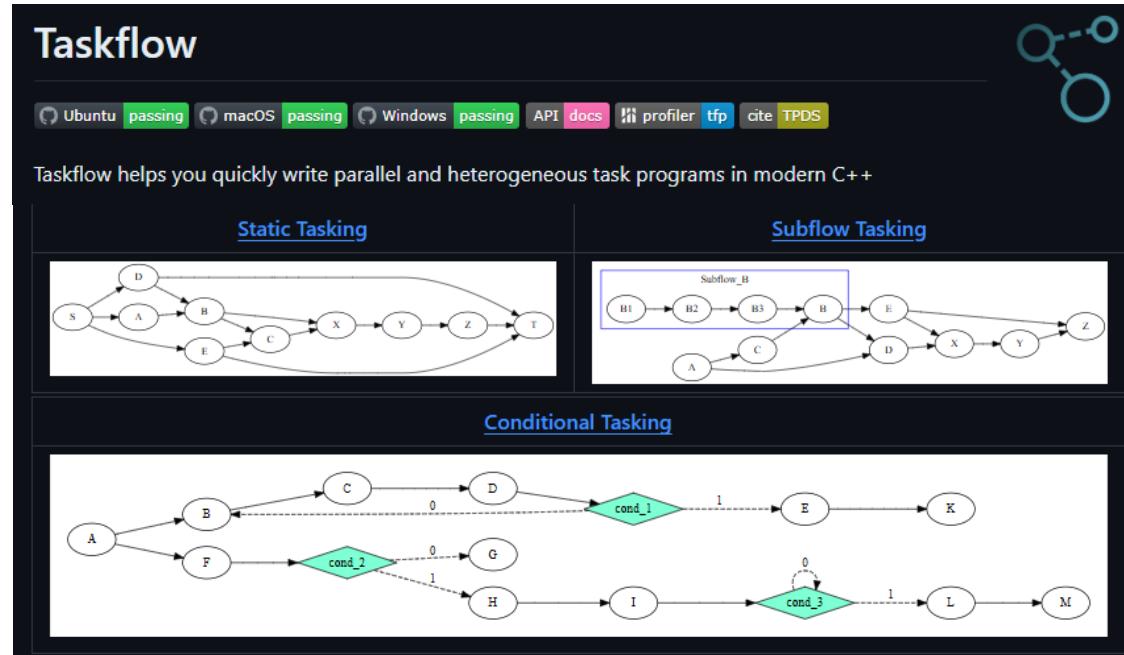
Assume 1920x1080 Image size  $\approx$  2MB

→ Transmission Delay for Extreme case:  
 $2\text{MB} / ((1416.2\text{kbits/s}) / (8\text{bits/1B})) = 11.57\text{sec}$

Maybe it reduces a lot when using locality and leads to smaller portion of transmission delay...  
(larger portion of decoding delay)

# What I am implementing...

## Scheduler

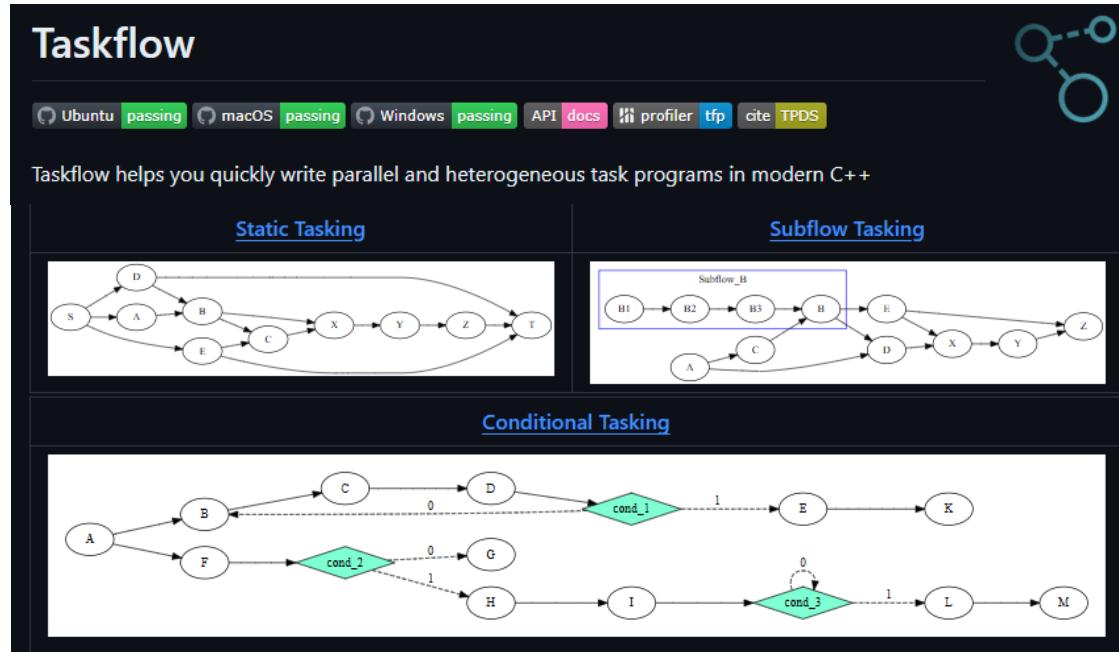


Determine frame dependency based on SVC(Scalable Video Coding) layer

supports heterogeneous tasking by harnessing the power of CPU-GPU collaborative computing

# What I am implementing...

## Scheduler



supports heterogeneous tasking by harnessing the power of CPU-GPU collaborative computing

```
#include <taskflow/taskflow.hpp> // Taskflow is header-only

int main(){

    tf::Executor executor;
    tf::Taskflow taskflow;

    auto [A, B, C, D] = taskflow.emplace( // create four tasks
        [] () { std::cout << "TaskA\n"; },
        [] () { std::cout << "TaskB\n"; },
        [] () { std::cout << "TaskC\n"; },
        [] () { std::cout << "TaskD\n"; }
    );

    A.precede(B, C); // A runs before B and C
    D.succeed(B, C); // D runs after B and C

    executor.run(taskflow).wait();

    return 0;
}
```

End of Presentation