SVE:Distributed Video Processing at Facebook Scale

By OUEDRAOGO W FABRICE GHISLAIN

Master 1 CCS University Of Rennes 1

Table Of Content

I. Introduction

II. Video At Facebook

III. How Facebook used to process video

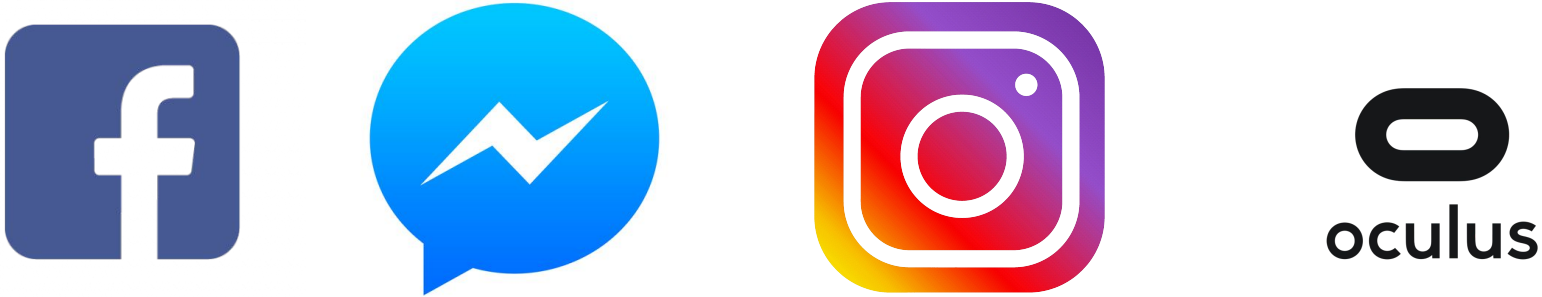
IV. Introducing Streaming Video Engine (SVE)

V. Related Work

VI. Conclusion


News Feed
Status
Photo
Check In
Latin
Tap into a world of fresh new audio. Listen to our newest playlist. Back to Fall Beats.

VIDEO AT FACEBOOK



FB:**500M** users watch **100M hours** video daily (Mar. 16)

Instagram:**250M** daily active users for stories (Jun. 17)

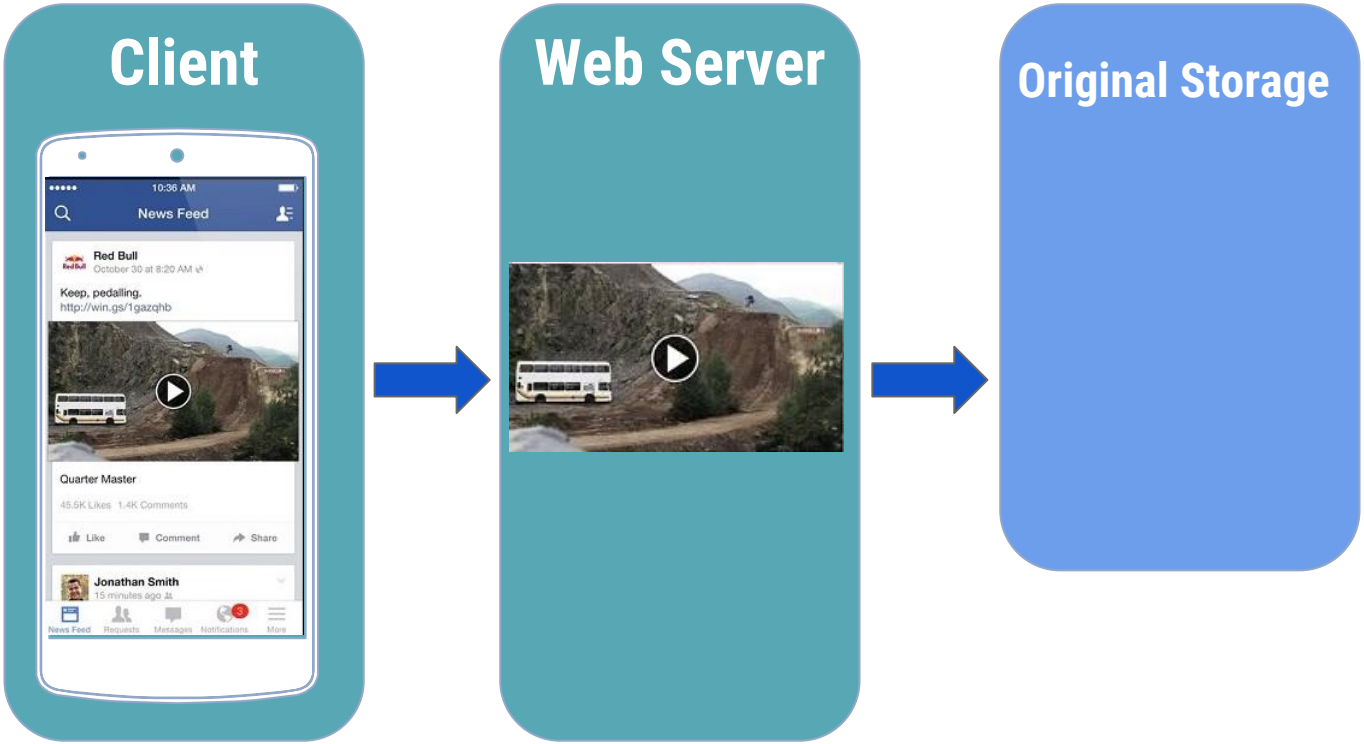
All:**many tens of millions** of daily uploads, **3X** NYE spike

How Facebook Used to Process Videos ?

The Monolithic Encoding Script

Upload Video File to Web Server

Preserve Original for Reliability



Process After Upload Completes

Client

Web Server

Original Storage

Procesing

Encoding with Different Bitrates

Client

Web Server

Original Storage

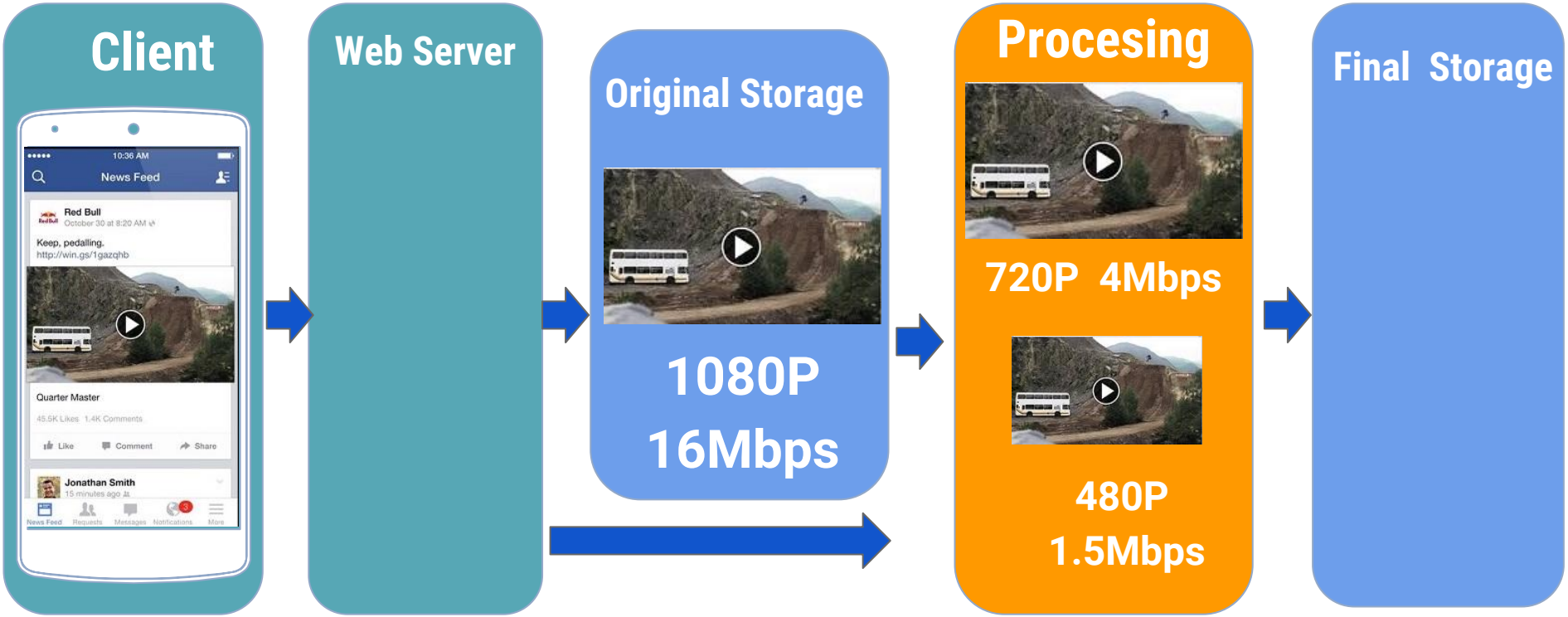
1080P
16Mbps

Procesing

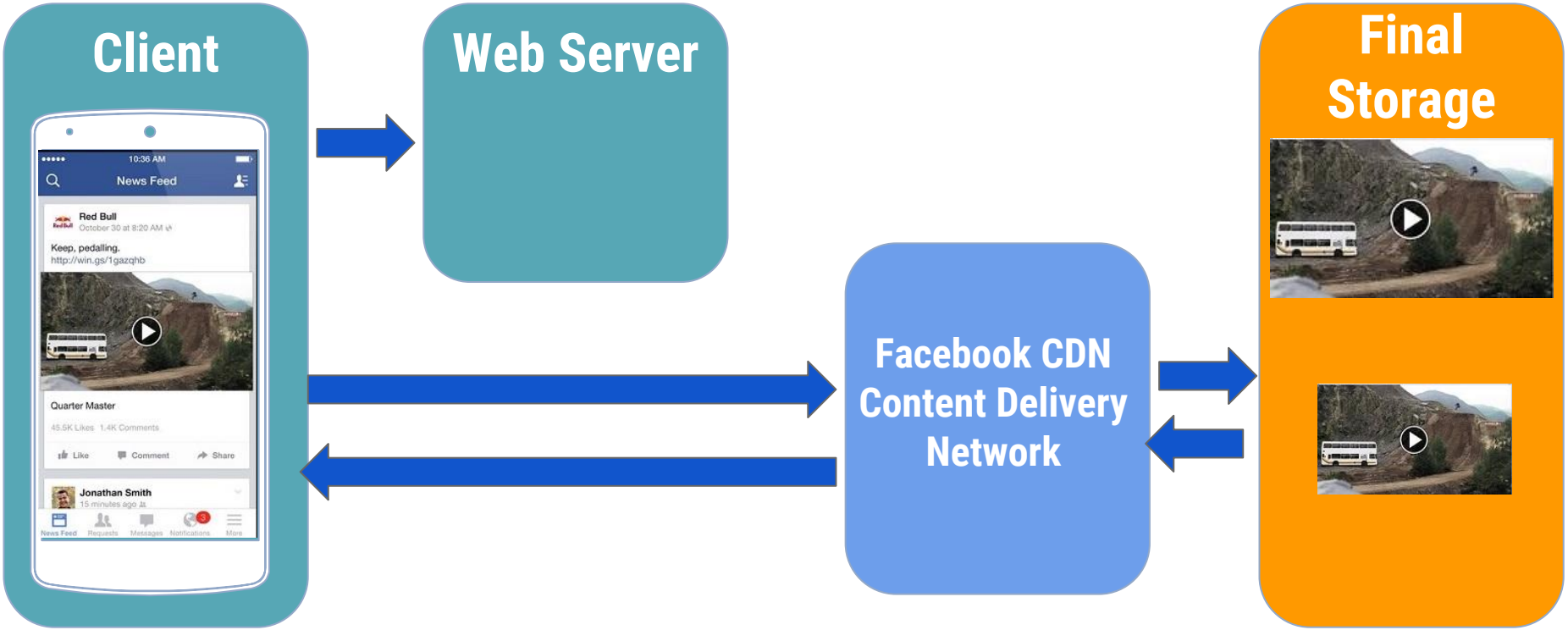
720P 4Mbps

480P
1.5Mbps

Store Encodings Before Sharing

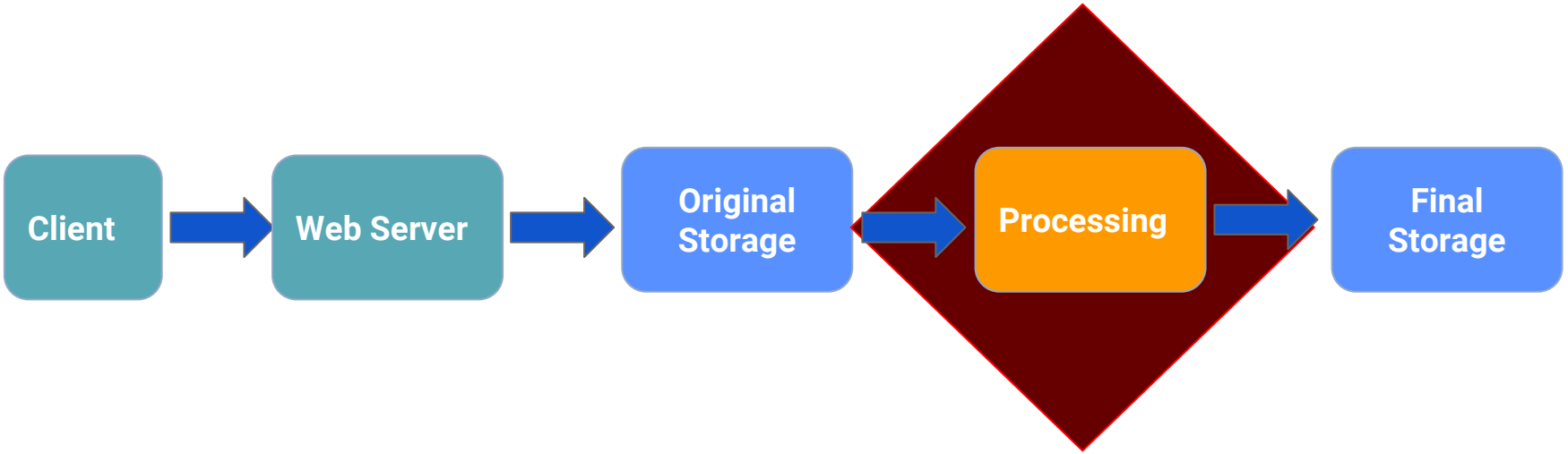


Encoding with Different Bitrates



The MES worked when videos were nascent but did not handle the requirements of low latency as we scaled

Problems of MES



Problems of MES

MES

Modification

Challenges of Video Processing

Speed

Users can share videos quickly

Flexibility

Write pipelines for tens of # apps

Robustness

Handle faults and overload at scale

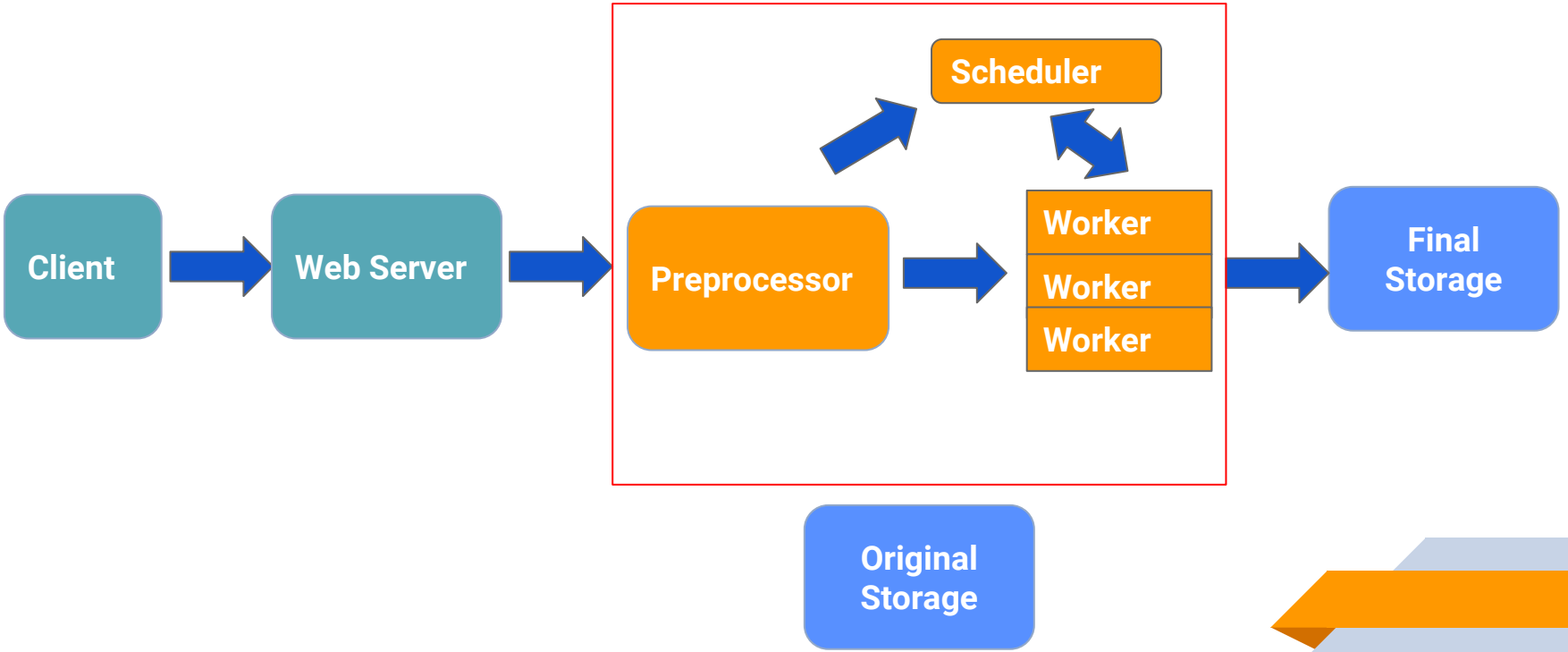
Introducing Streaming Video Engine (SVE)

Speedy: harness parallelism

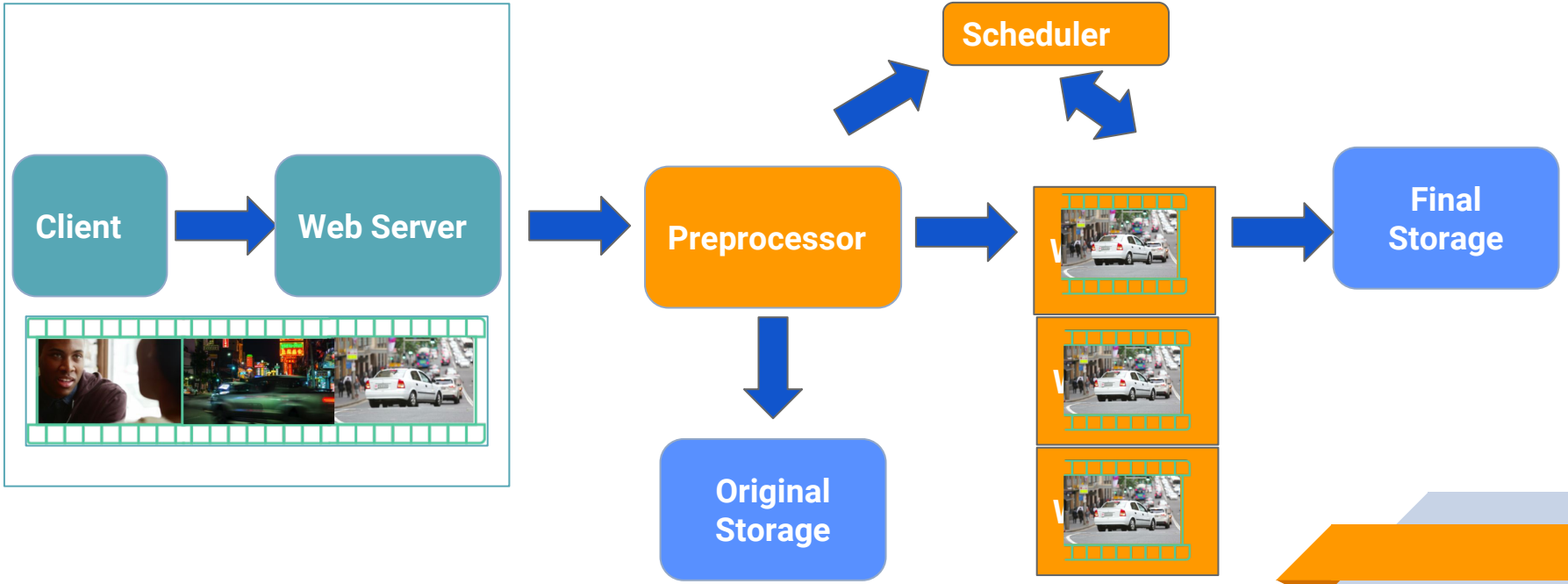
Users can share videos quickly

- Overlap fault tolerance and processing
- Overlap upload and processing
- Parallel processing

Architectural changes for parallelism



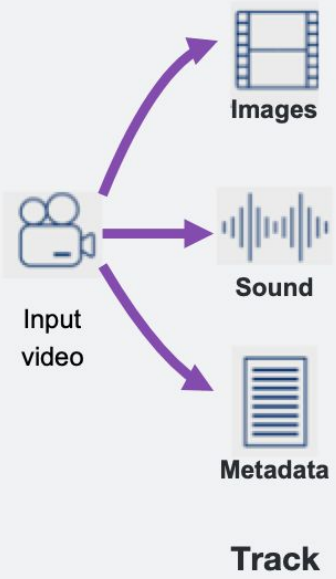
Overlap upload and processing



Flexible: Build DAG Framework

- DAG of computation on the stream-of-tracks abstraction
- Engineers write multiple tasks in a familiar language
- Dynamic DAG generation per video

Flexible: build DAG framework



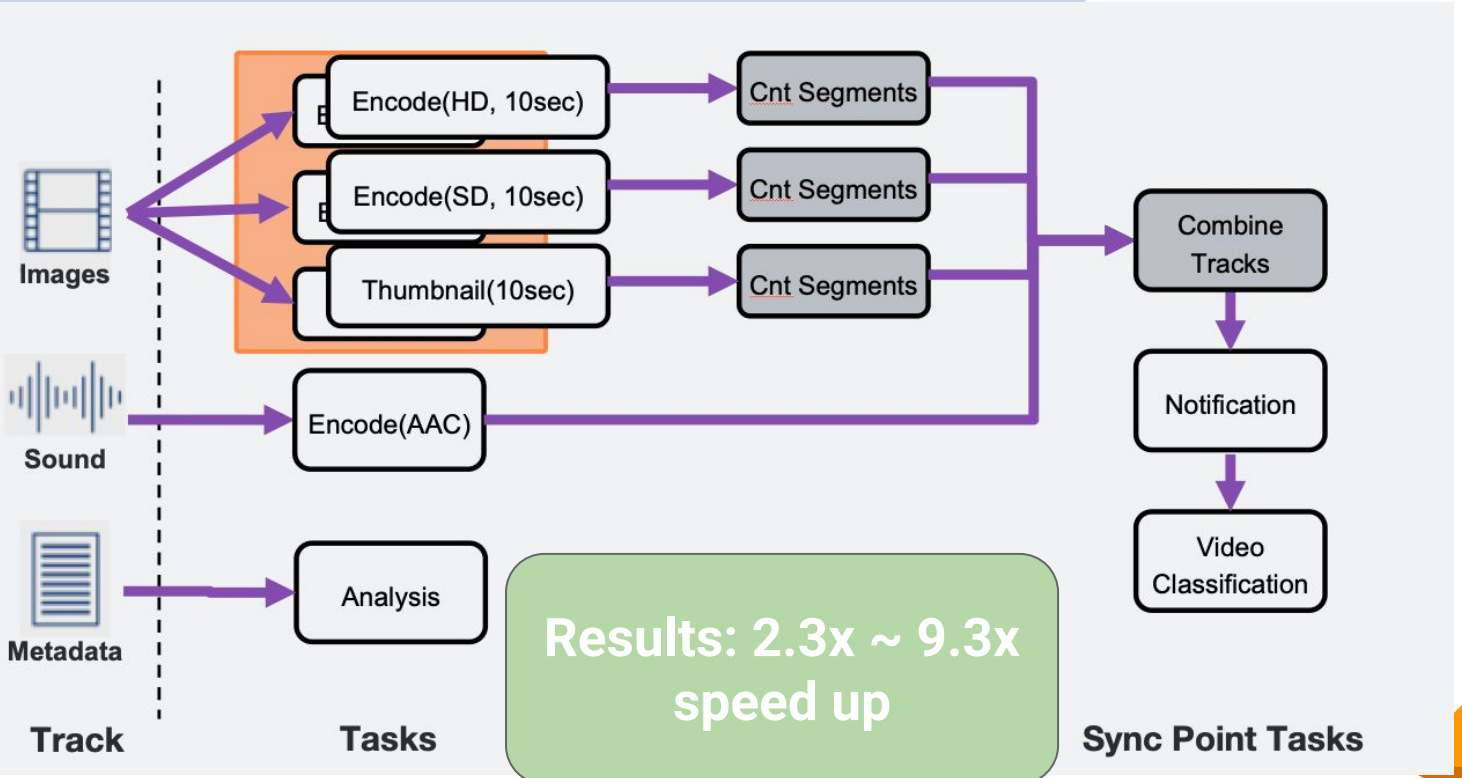

```
$pipeline = Pipeline.build()

$video_track=$pipeline>addTrack(IMG_TYPE)
  ->addTask()

$audio_track=$pipeline>addTrack(AUD_TYPE)
  ->addTask()

$meta_track=$pipeline>addTrack(META_TYPE)
  ->addTask()
```

Flexible: build DAG framework



Results: 2.3x ~ 9.3x speed up

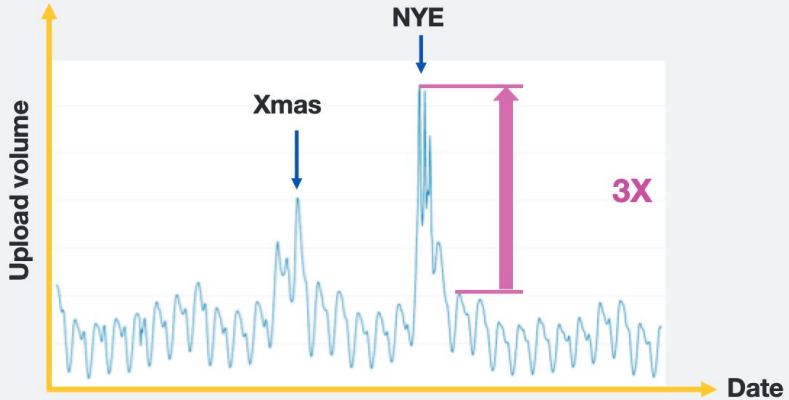
Robust: Tolerate Overload

Handle faults and overload that is inevitable at scale

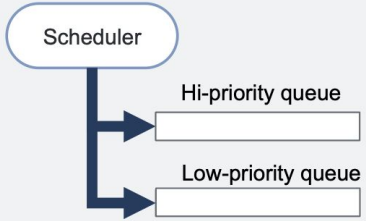
- Rely on priority to degrade non-latency-sensitive tasks
- Defer full video processing for some new uploads
- Load-shedding across global deployments

Handle Fault And Overload

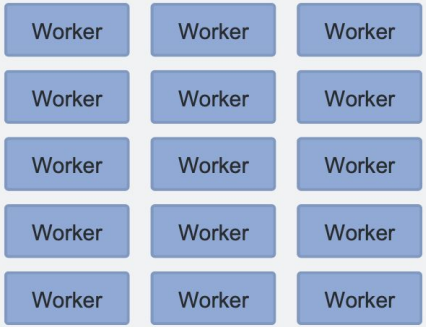
3X peak load during New Year Eve



Use priority for worker overload



Only assign hi-pri tasks under overload



Summary of SVE Potentialities

Speed

2.3x ~ 9.3x

Flexibility

One System for 15+ Apps

Robustness

Tolerate 3x traffic spike

Related Work

- Batch processing

  SVE overlaps data ingestion and processing

- Stream processing

  SVE offers dynamic DAG generation per input

Conclusion

Taken together, these improvements enable SVE to reduce the time between an upload complete and video share by 2.3X–9.3X over MES.

THANKS!

Any questions?