# SVE: Distributed Video Processing at Facebook Scale

Qi Huang

Petchean Ang, Peter Knowles, Tomasz Nykiel, Iaroslav Tverdokhlib, Amit Yajurvedi, Paul Dapolito IV, Xifan Yan, Maxim Bykov, Chuen Liang, Mohit Talwar, Abhishek Mathur, Sachin Kulkarni, Matthew Burke, Wyatt Lloyd
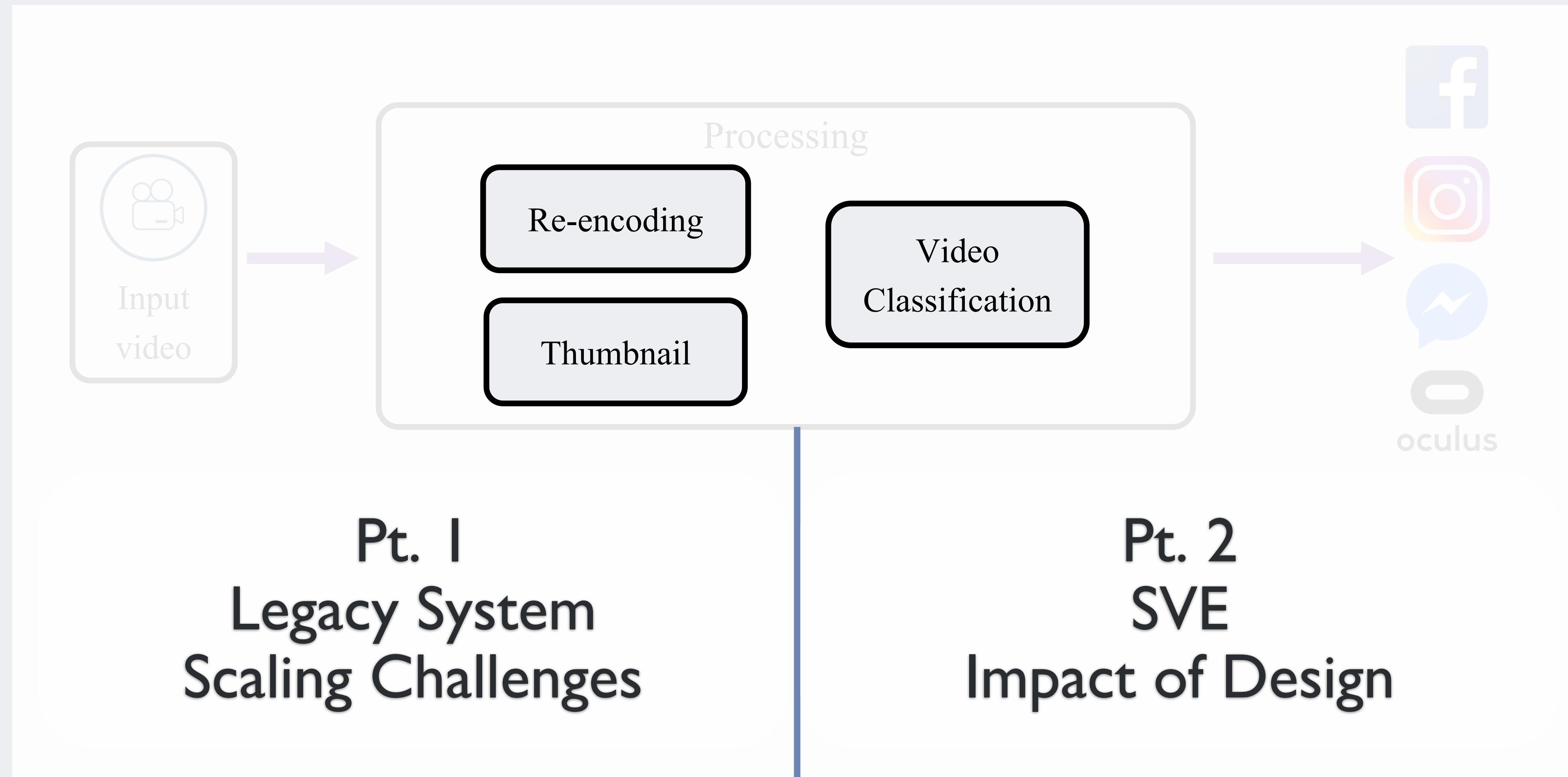
Facebook, University of Southern California, Cornell, Princeton

Video is growing across Facebook
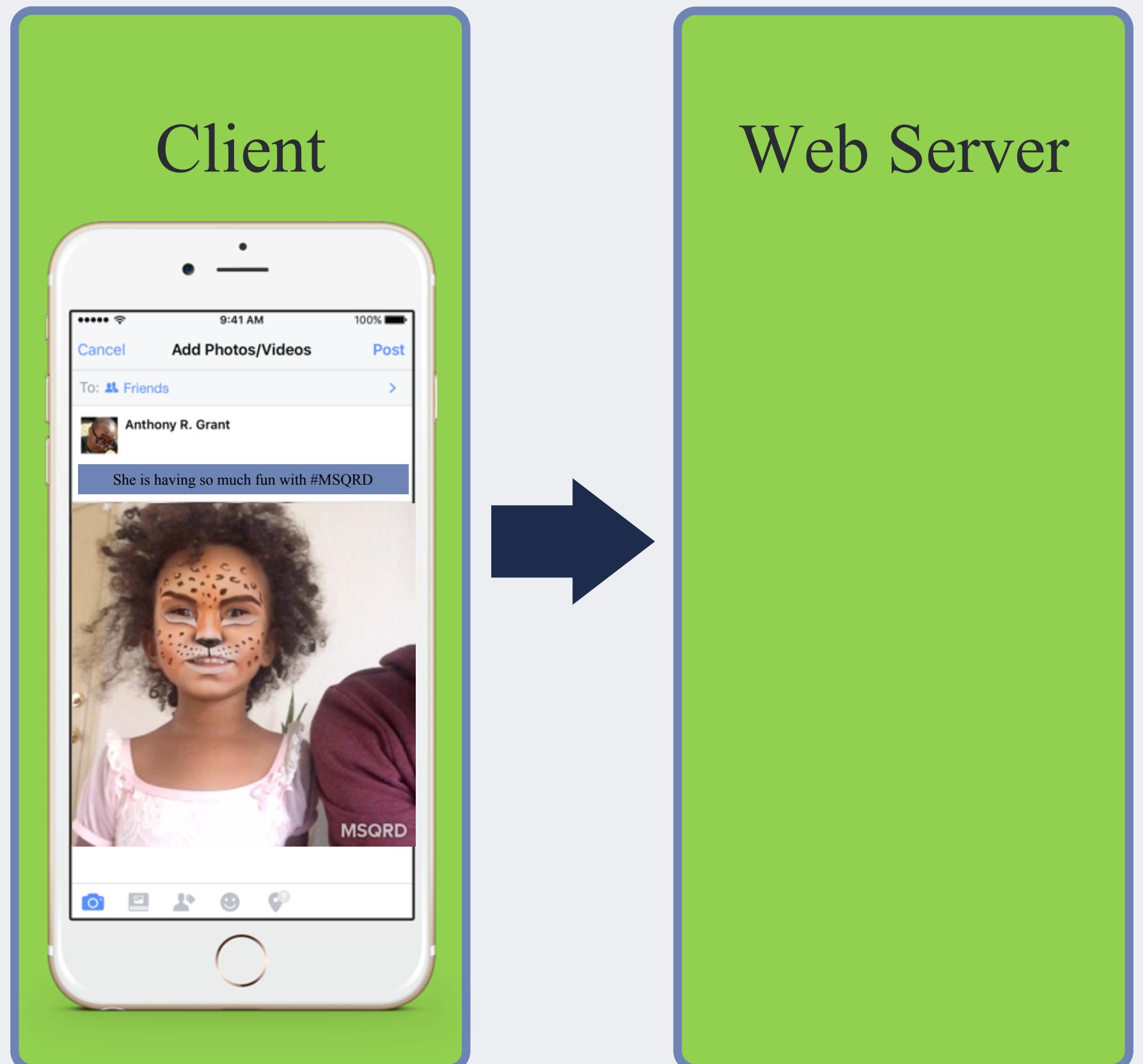
- FB: **500M** users watch **100M hours** video daily (Mar. 16)
- Instagram: **250M** daily active users for stories (Jun. 17)
- All: **many tens of millions** of daily uploads, **3X** NYE spike
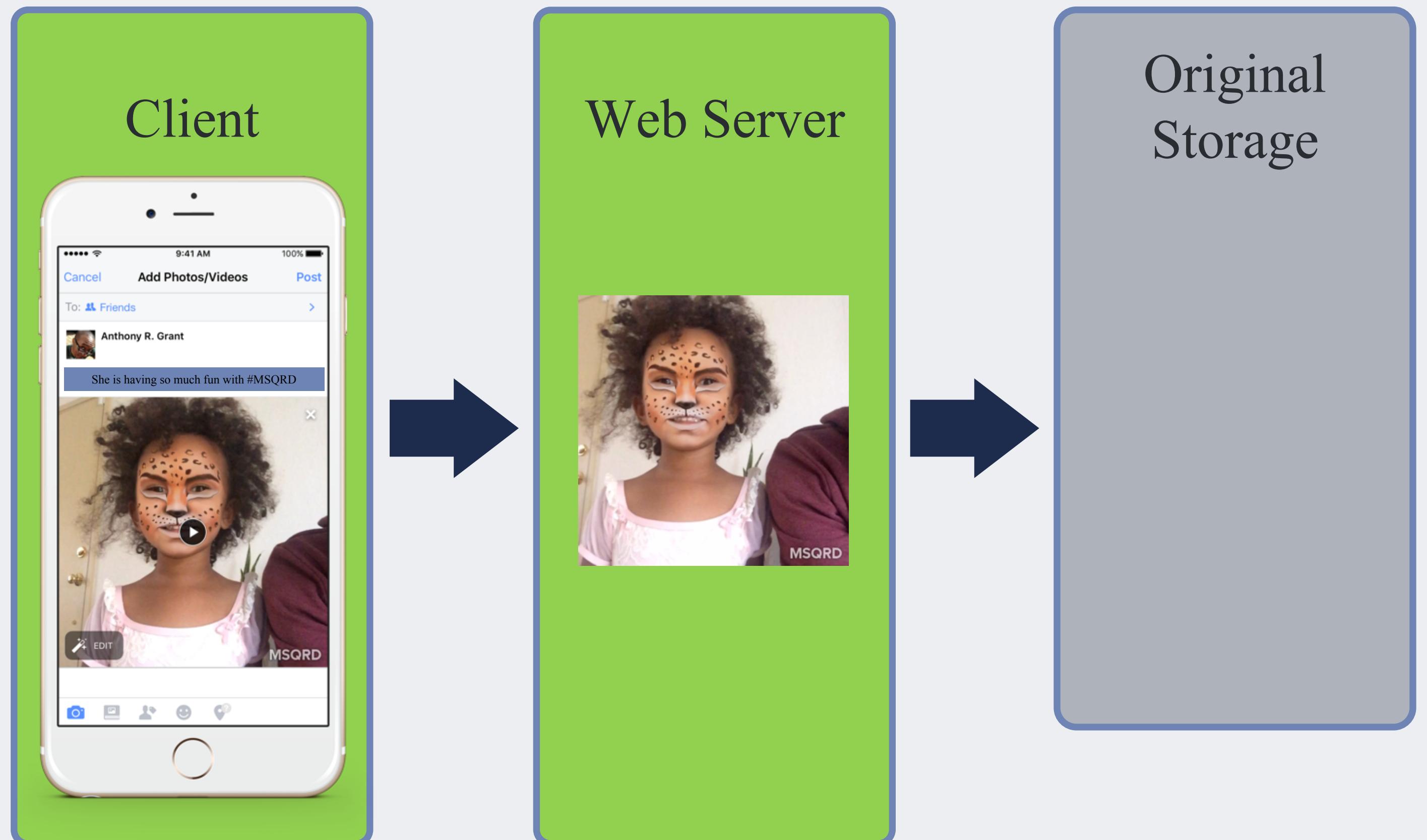
Processing is diverse and demanding

Input video → Processing (Re-encoding, Thumbnail, Video Classification) → Facebook, Instagram, Messenger, oculus

Pt. 1
Legacy System
Scaling Challenges

Pt. 2
SVE
Impact of Design

02
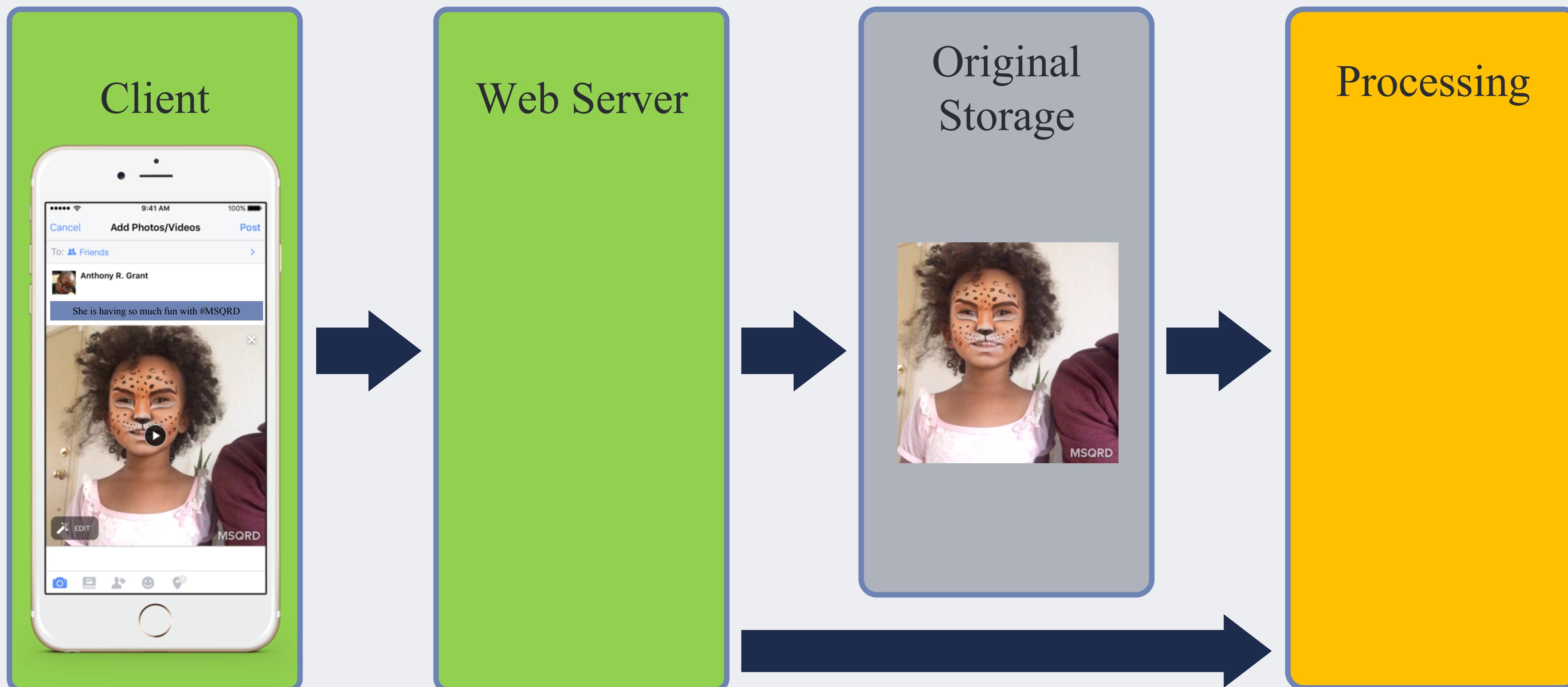
Legacy: upload video file to web server

Client

Web Server
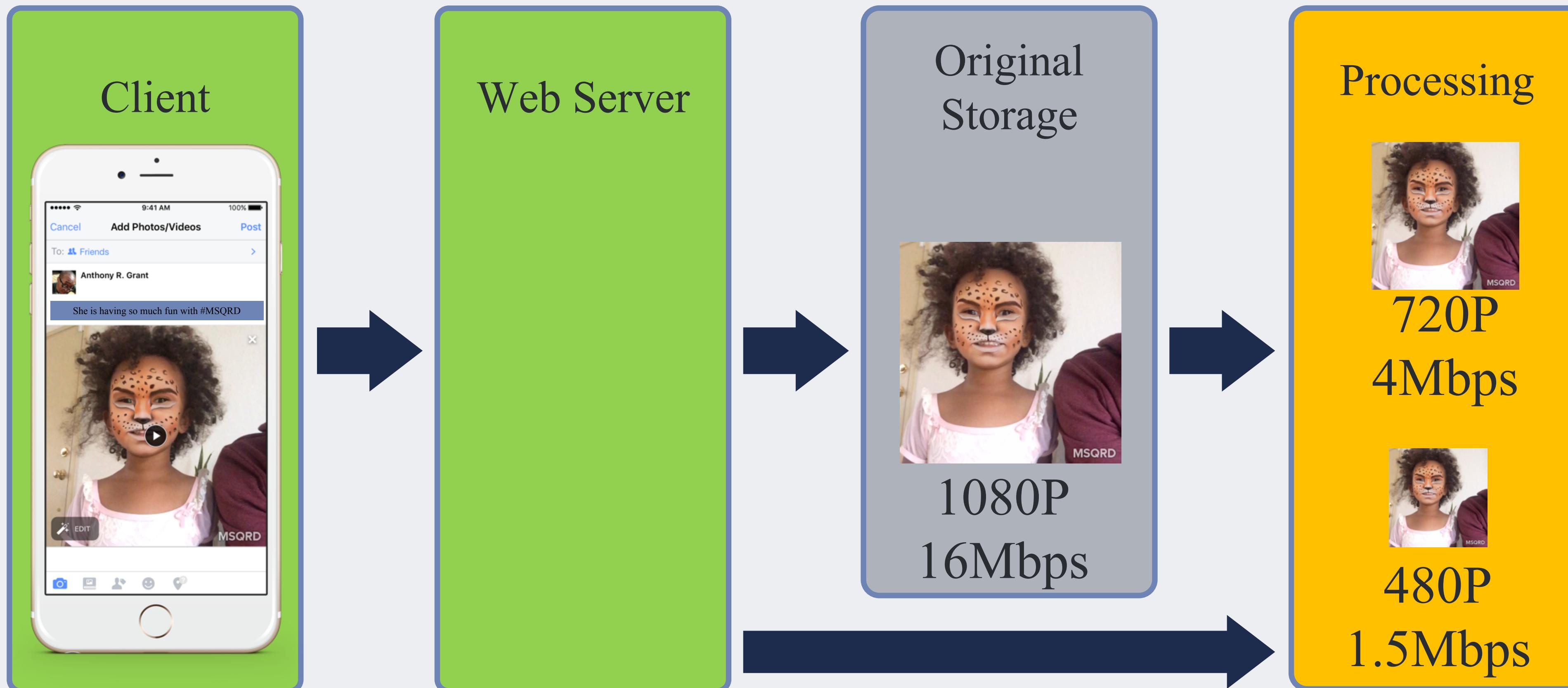
Legacy: preserve original for reliability

Client

Web Server

Original Storage
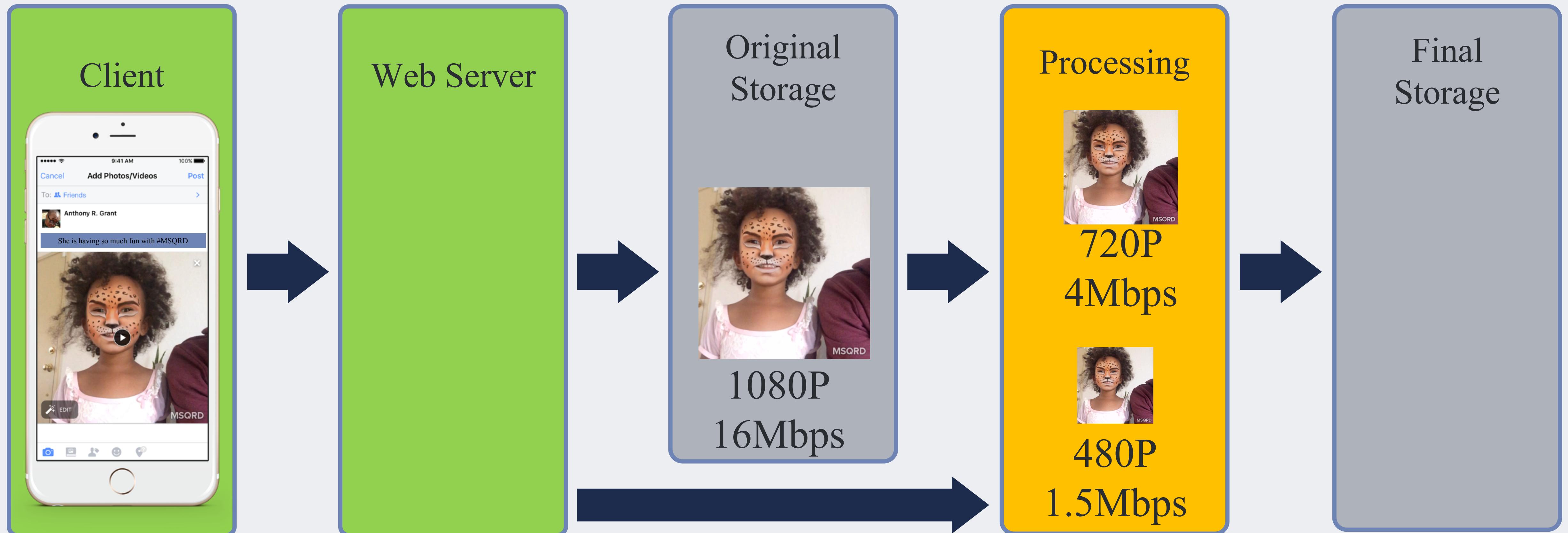
Legacy: process after upload completes

Client

Web Server

Original Storage

Processing
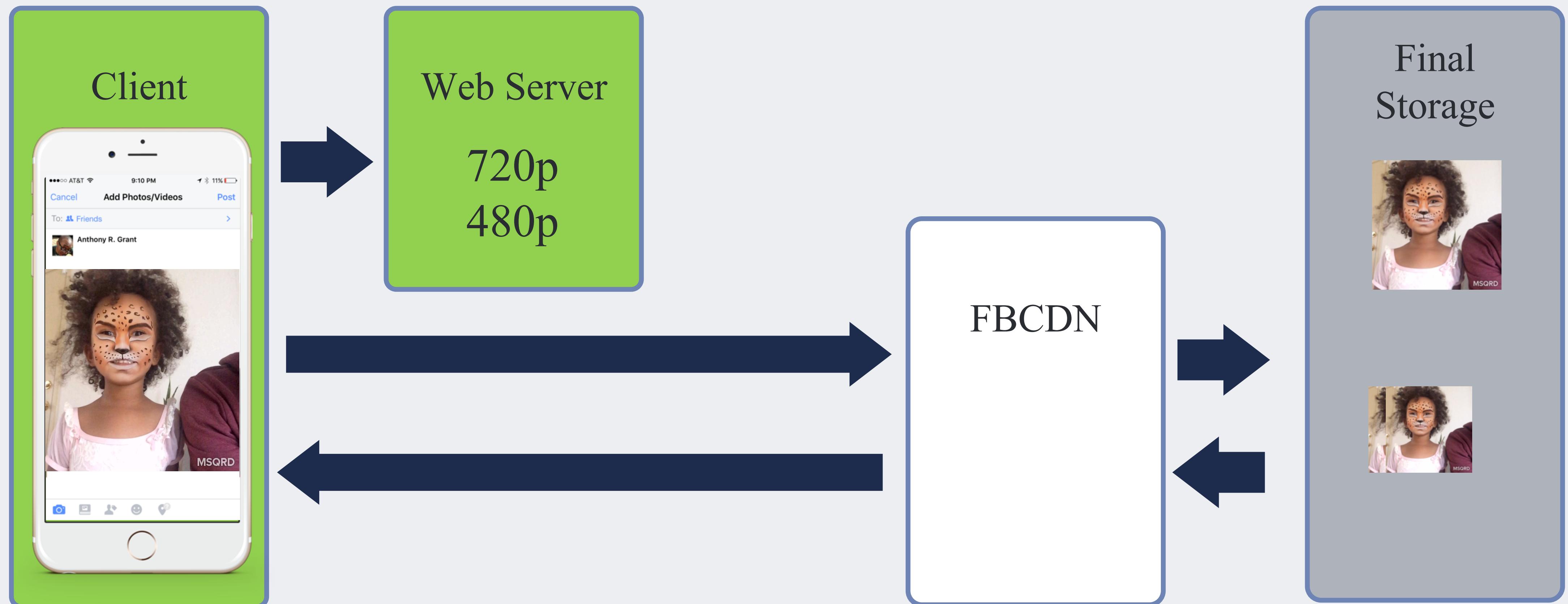
Legacy: encode w/ varying bitrates

Client → Web Server → Original Storage (1080P 16Mbps) → Processing (720P 4Mbps, 480P 1.5Mbps)

Legacy: store encodings before sharing

Sharing with adaptive streaming

Client

Web Server

720p
480p

FBCDN

Final Storage
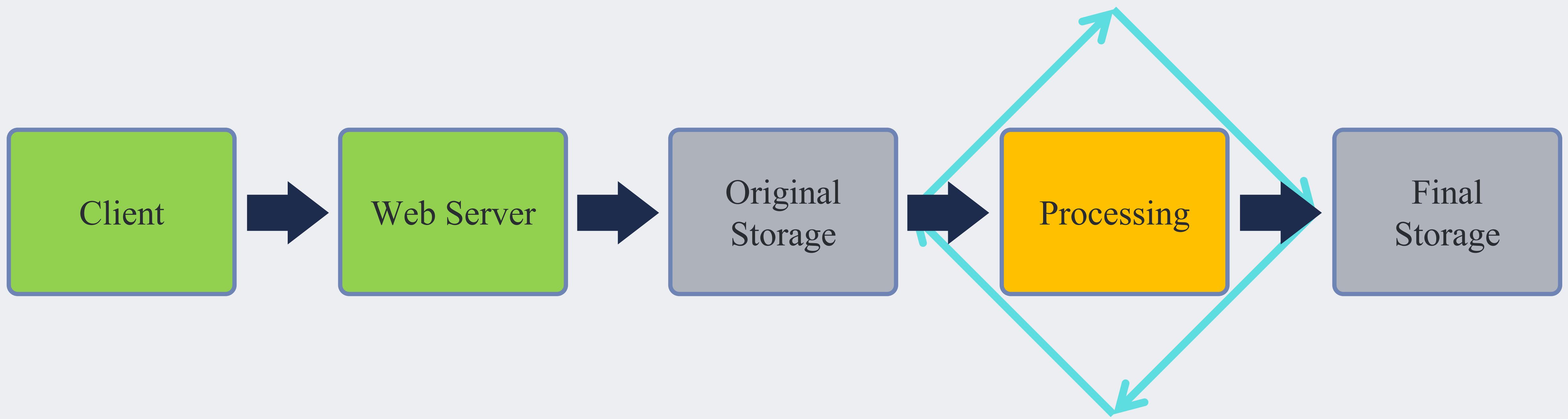
Focus: pre-sharing pipeline

Client → Web Server → Original Storage → Processing → Final Storage

All steps from when a user starts an upload until a video is ready to be shared

Serial pipeline leads to slow processing

Monolithic script slows development

"Let's experiment speech recognition, add a logic to extract audio and analysis"

"Change color coding at different time"

"We need to change the thumbnail generation logic for videos > x minutes to create scene-based scrubber preview"

Processing → Final Blob Storage

Client → Web Server

"Pass-through for small and well-formatted videos"

"We want to experiment AI-based encodings to spend 10x CPU for 30% compression improvement on popular videos"

11

Challenges for video processing @ FB

Speedy

Users can share videos quickly

Flexible

Thousands of engineers can write pipelines for tens of apps

Robust

Handle faults and overload that is inevitable at scale

12
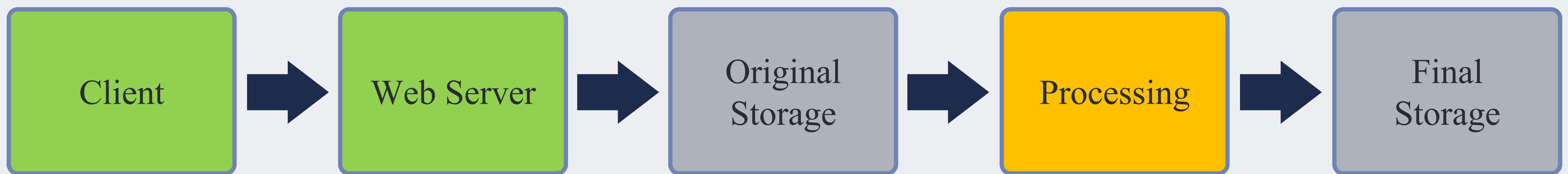
Our Streaming Video Engine (SVE)
# is speedy, flexible, and robust

Speedy: harness parallelism
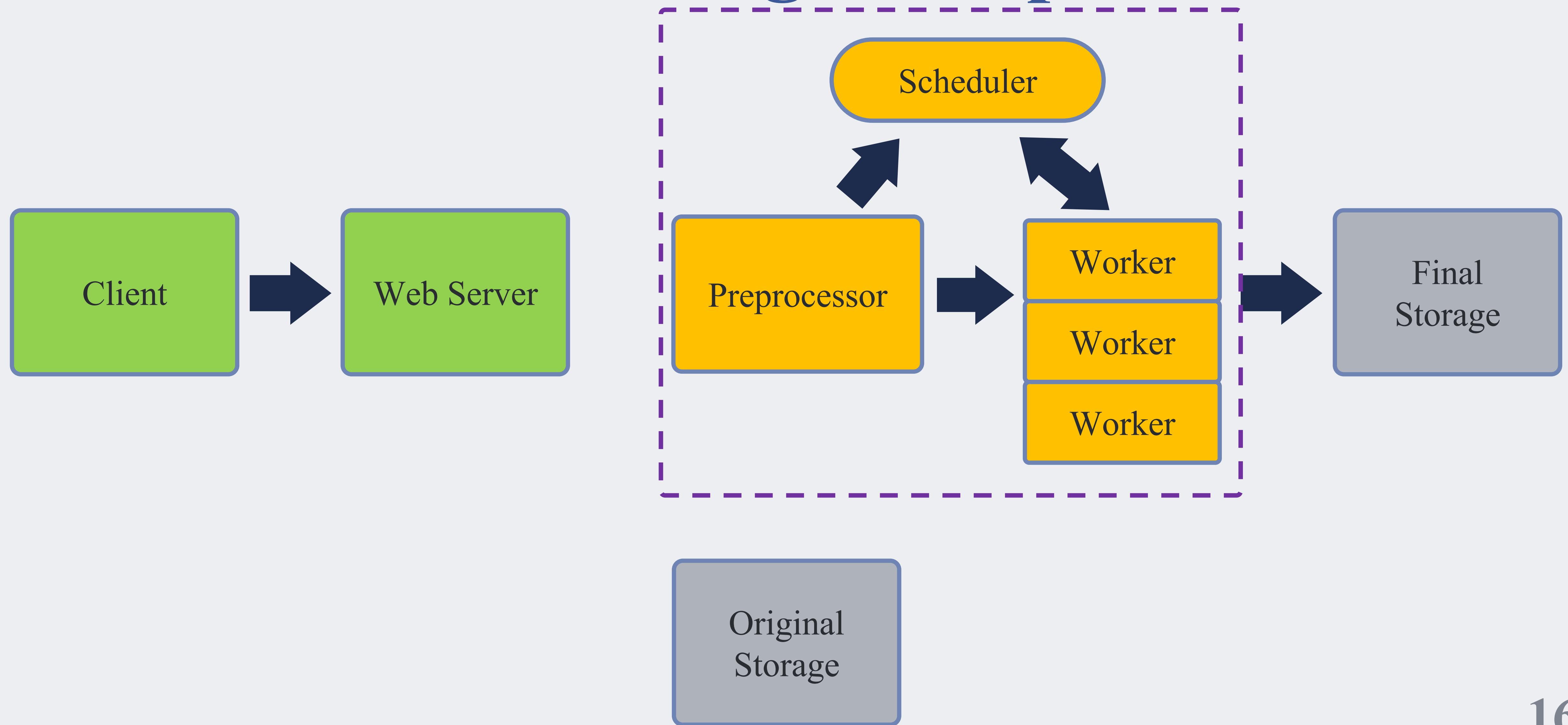
Users can share videos quickly

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

14
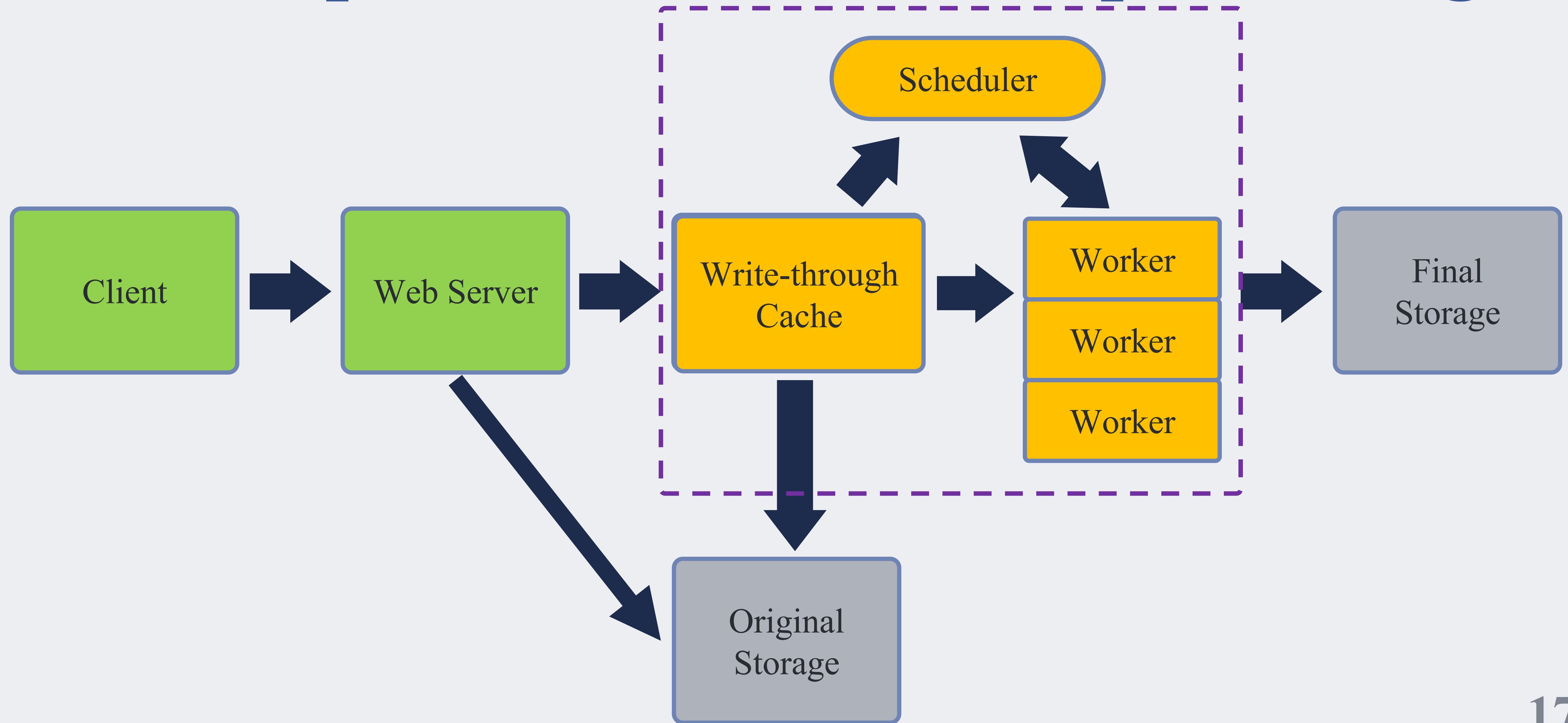
# Architectural changes for parallelism
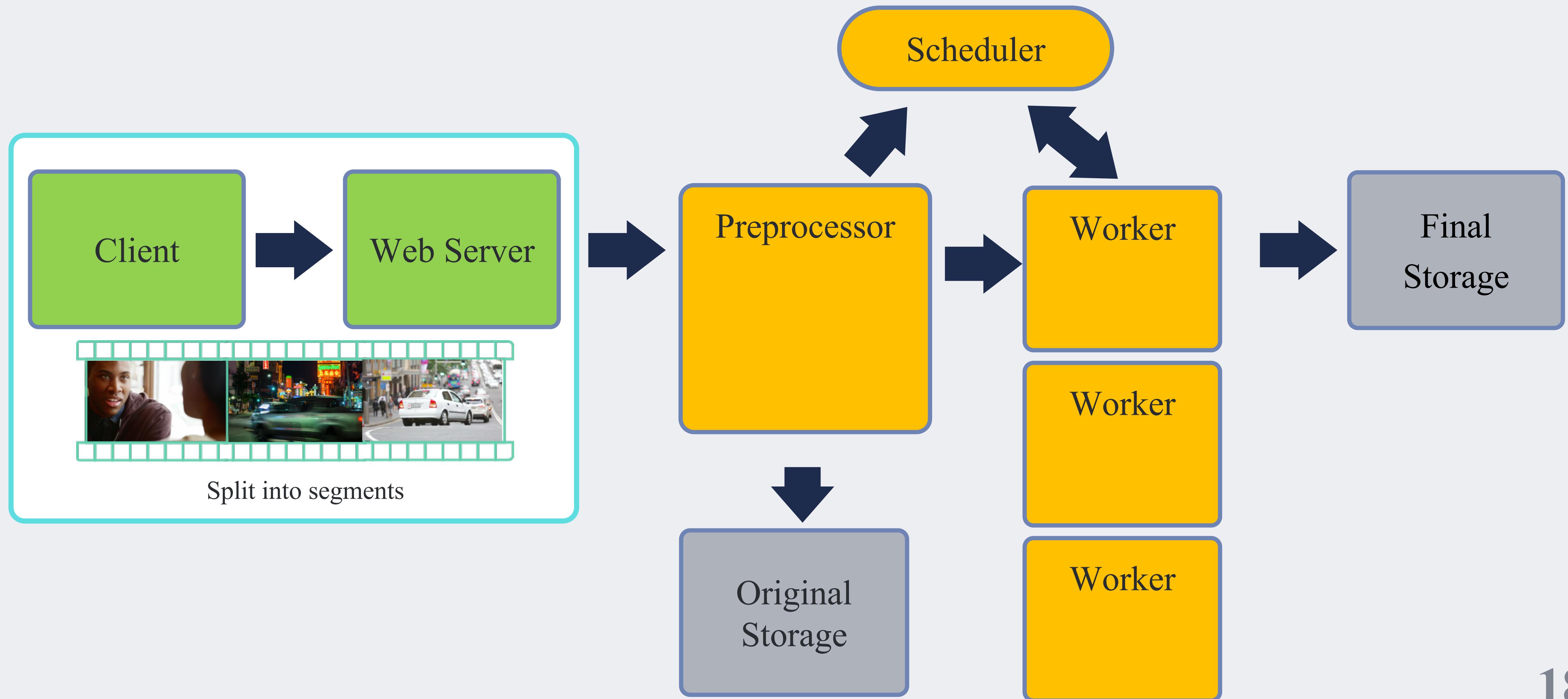
Architectural changes for parallelism
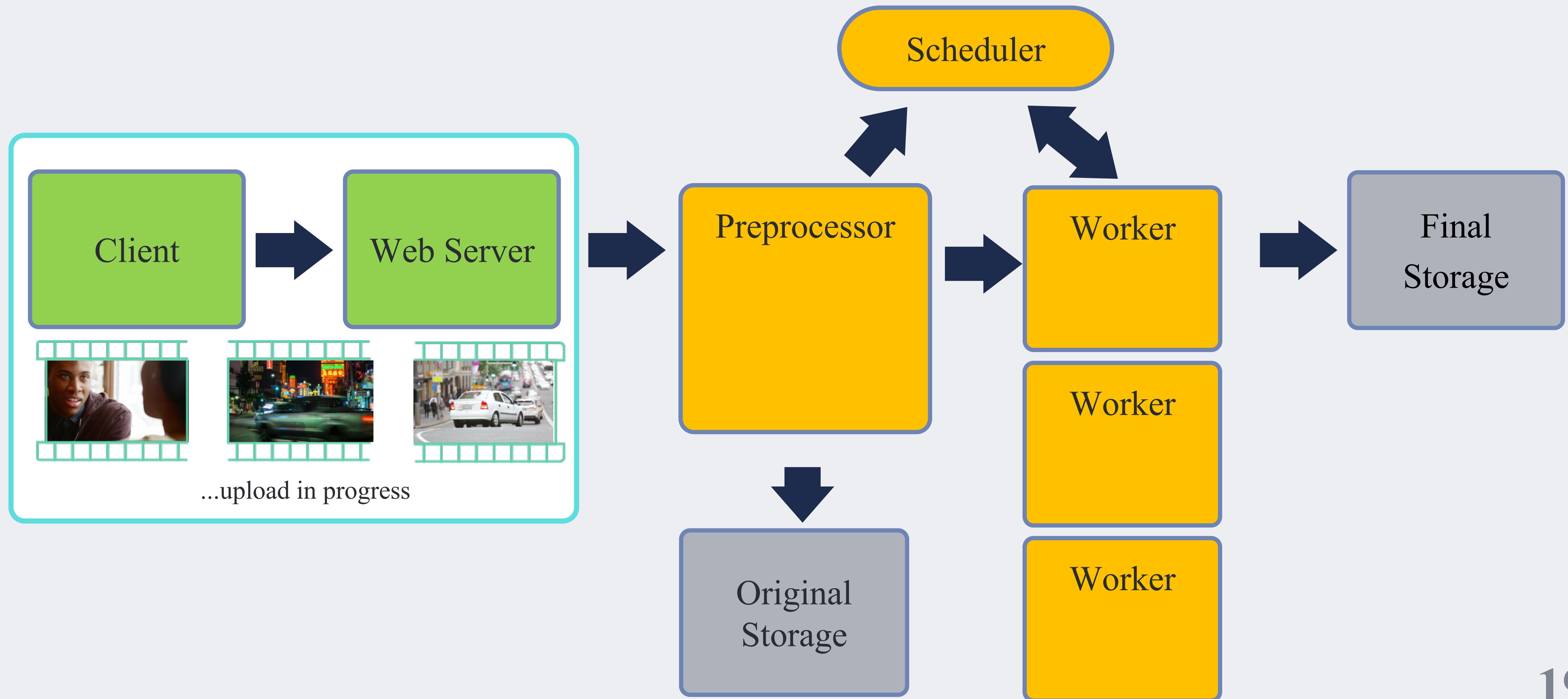
# Overlap fault tolerance and processing
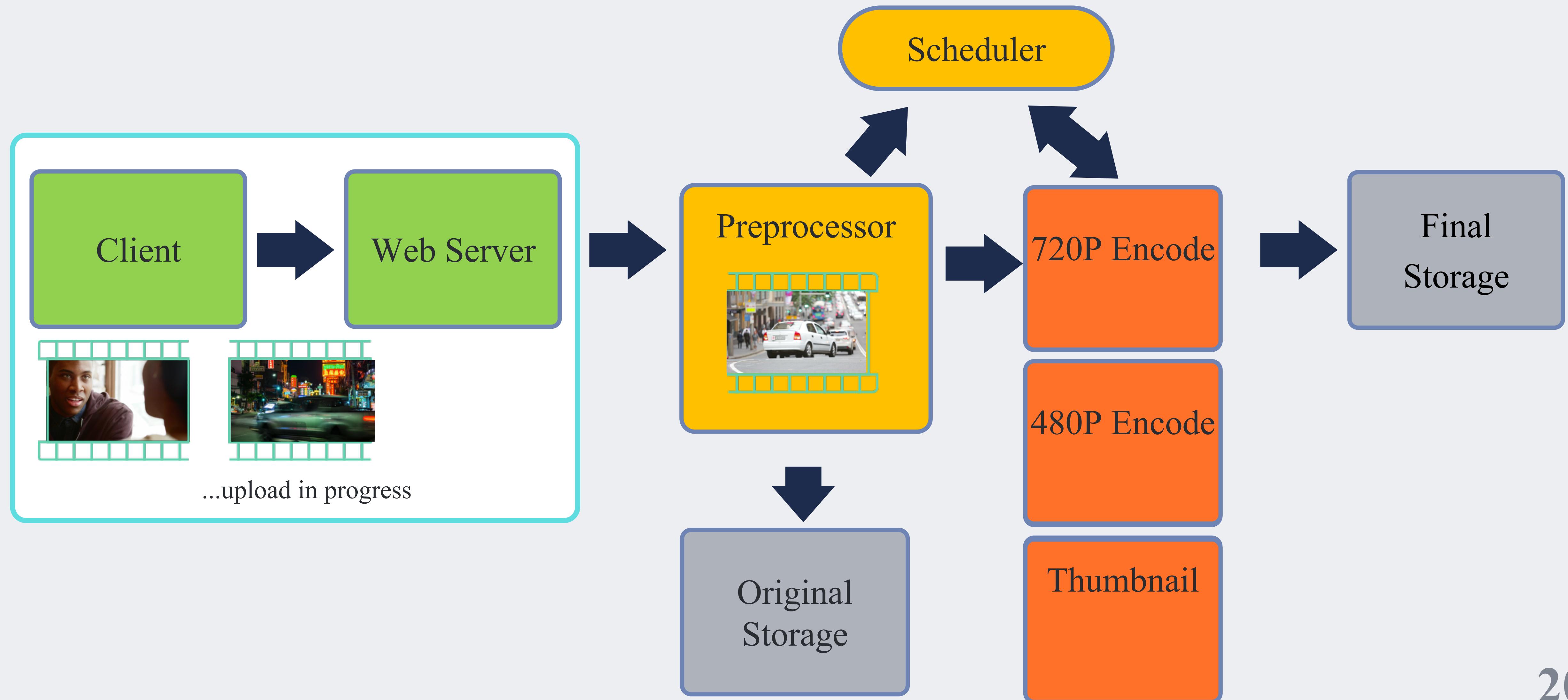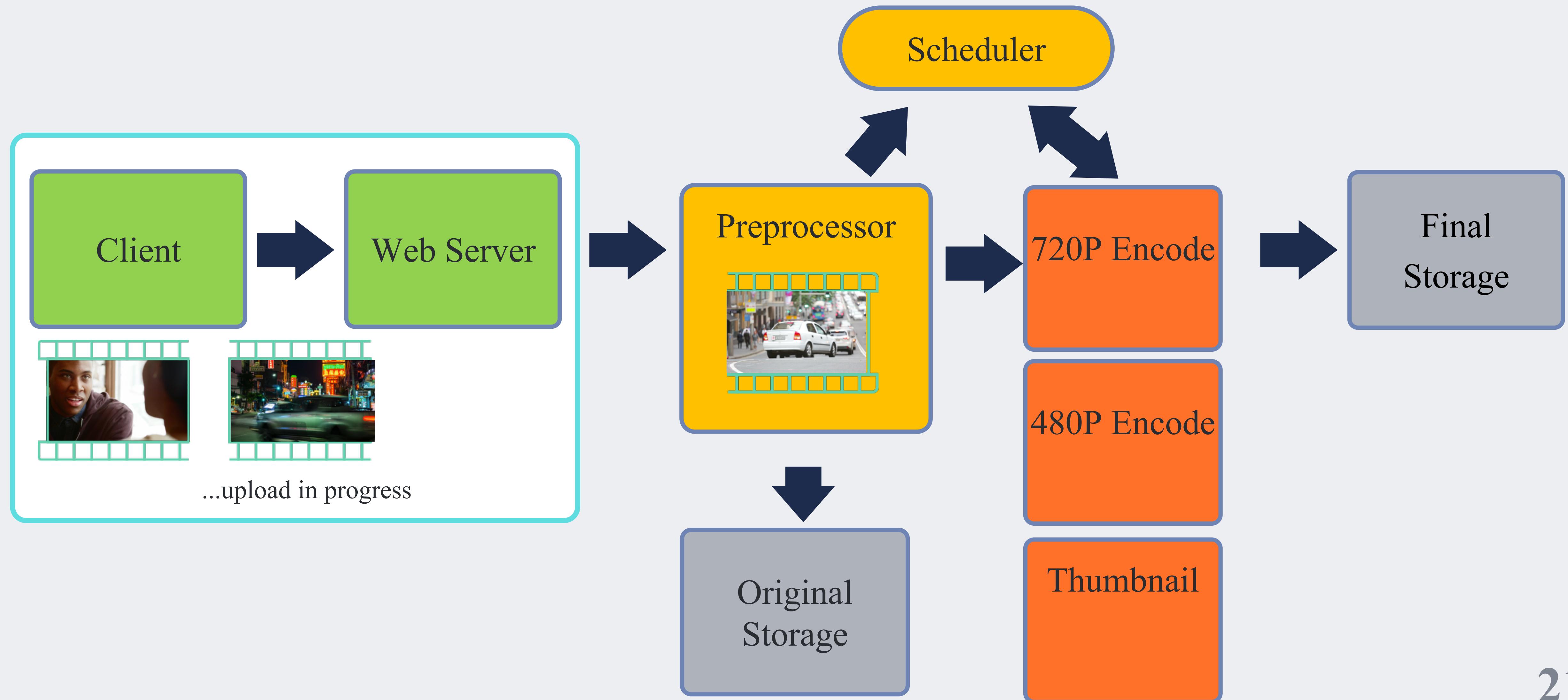
Overlap upload and processing

Overlap upload and processing
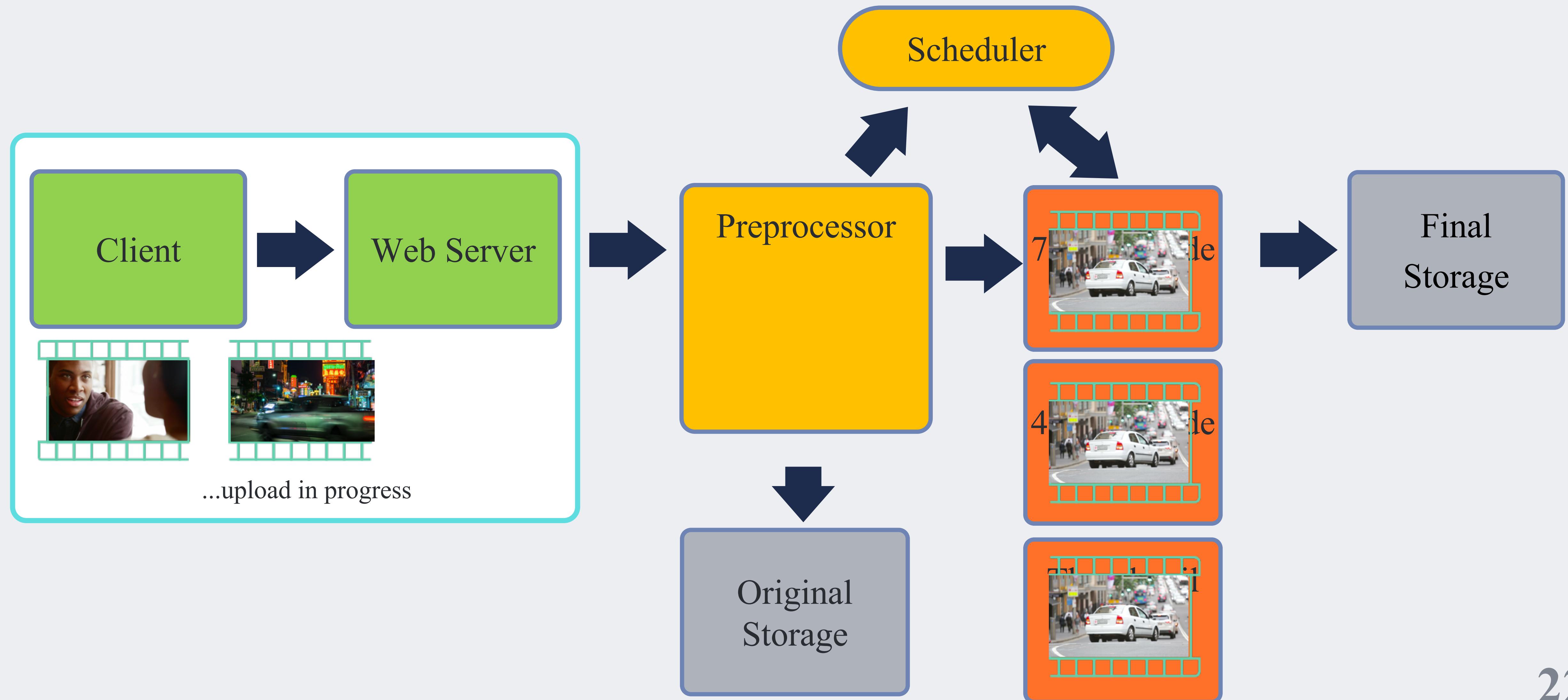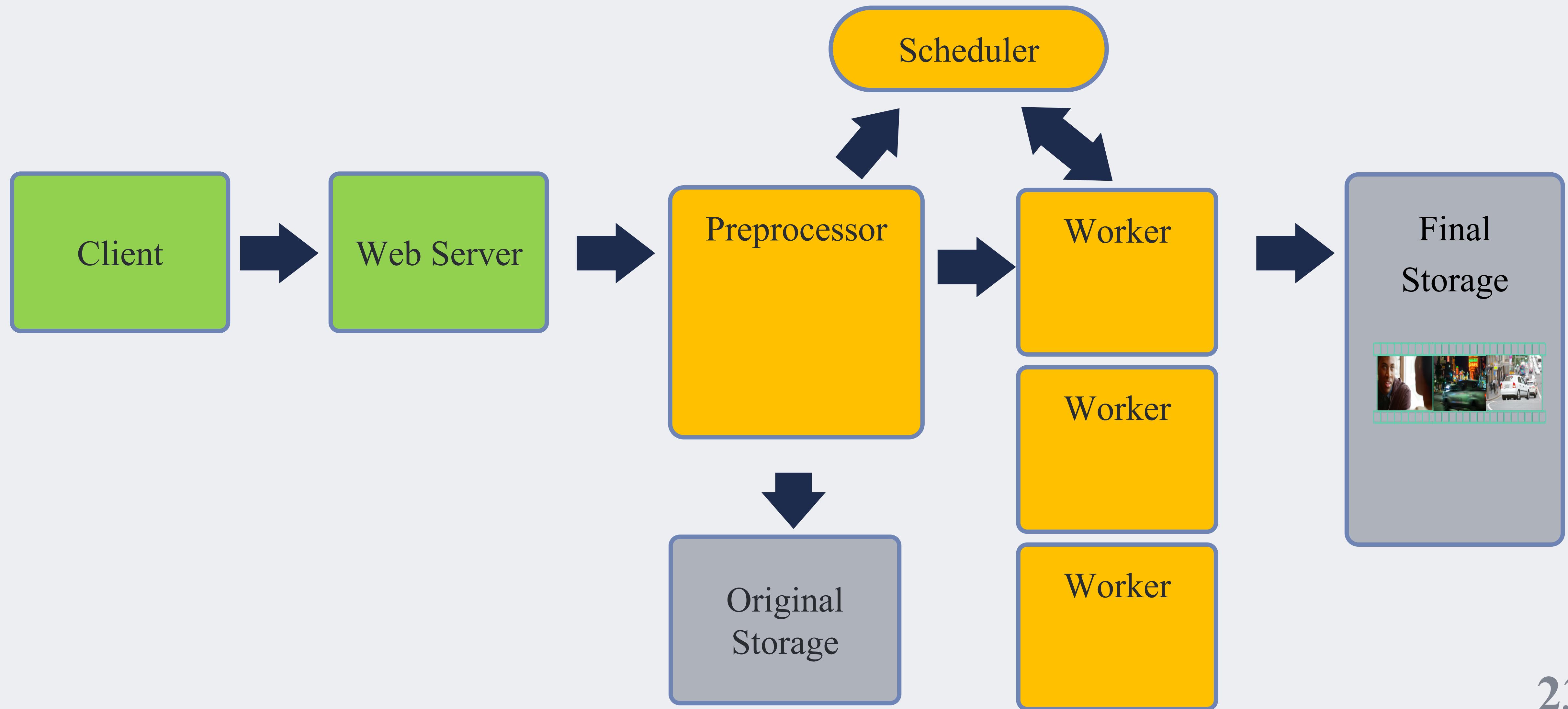
Parallel processing w/ many workers

Parallel processing w/ many workers

Parallel processing w/ many workers

Parallel processing w/ many workers

Client → Web Server → Preprocessor → Worker → Final Storage

Scheduler

Preprocessor → Original Storage

Worker
Worker
Worker

23
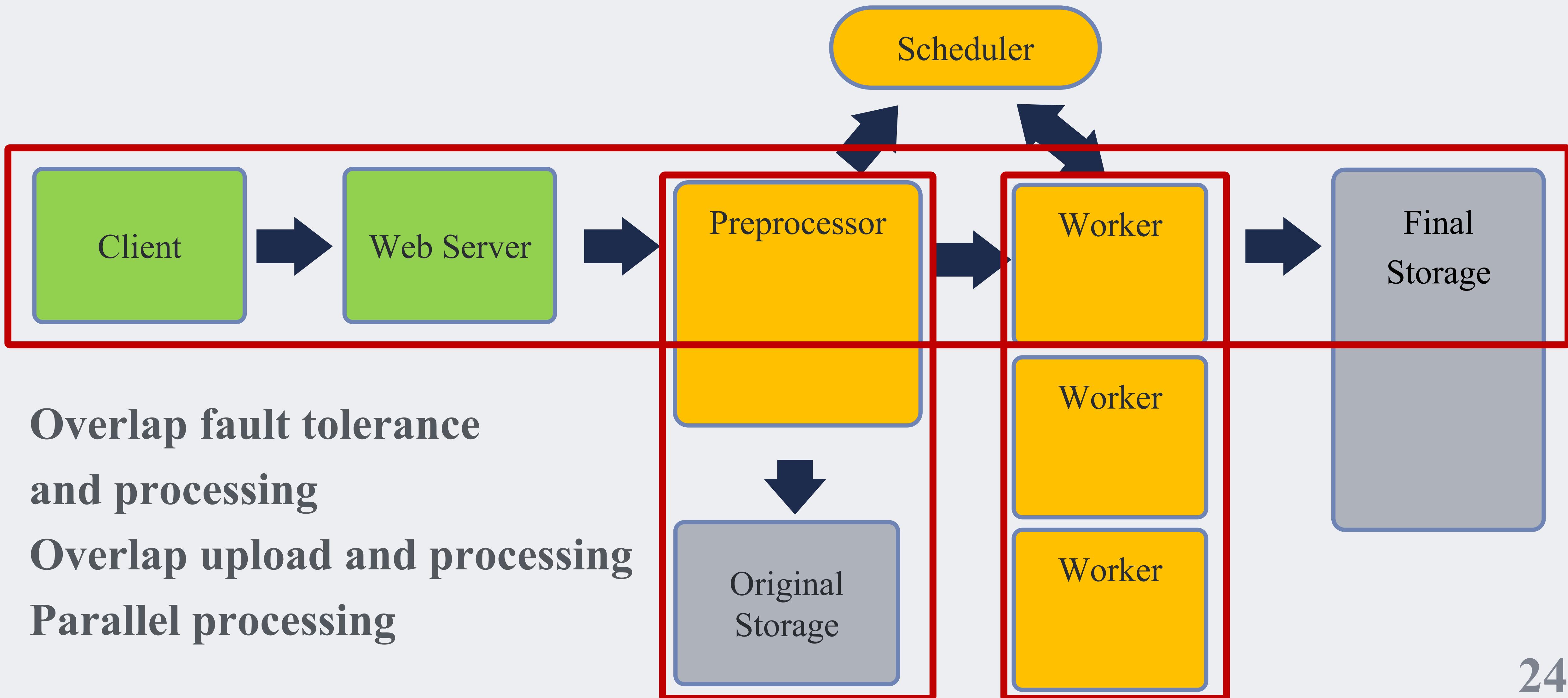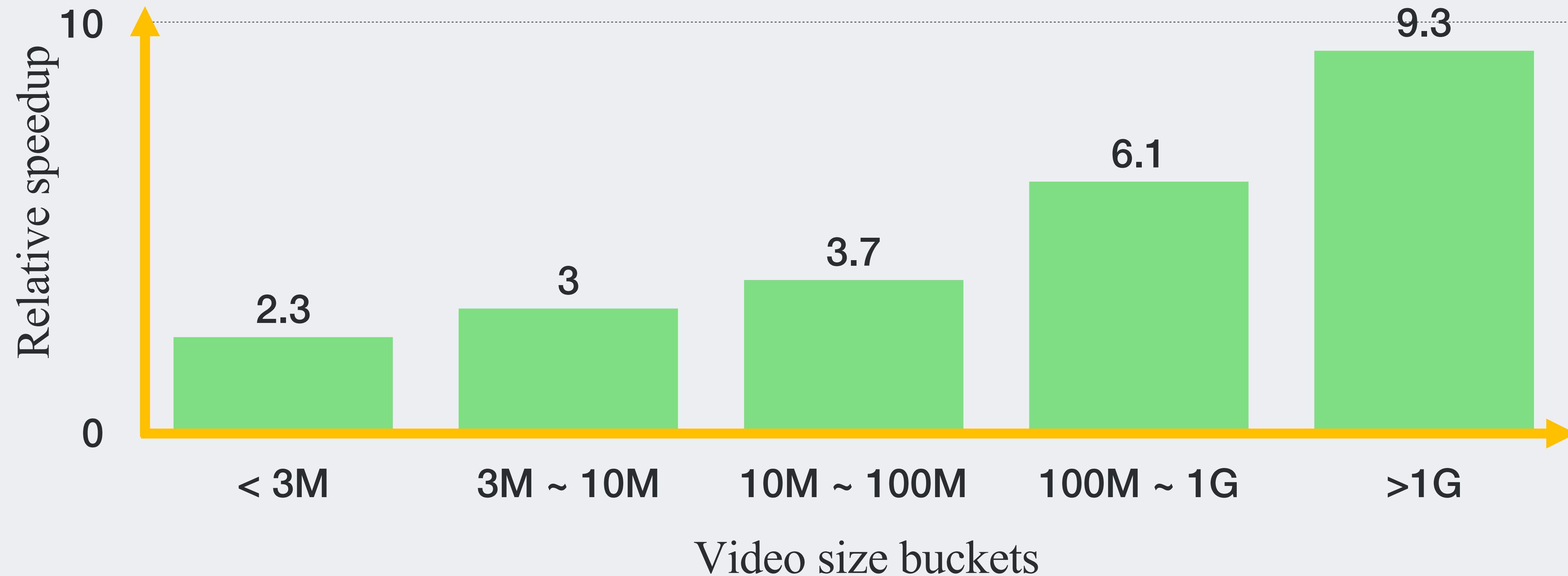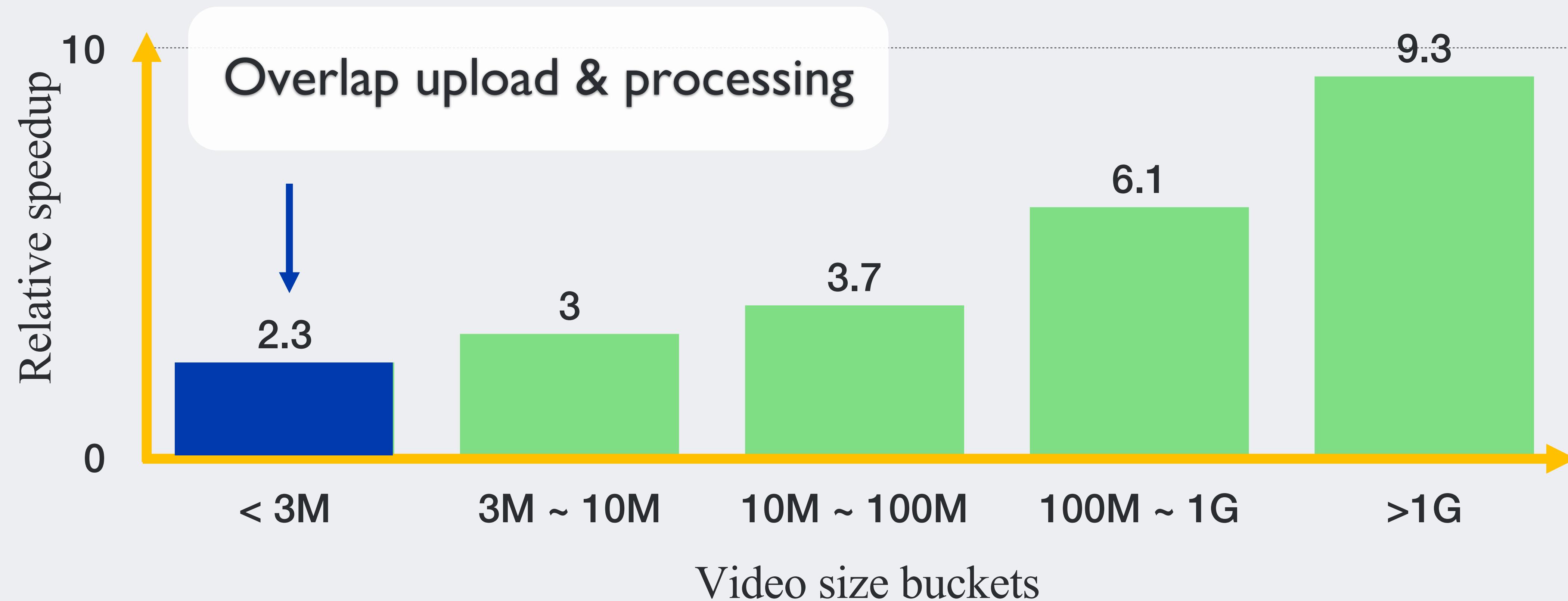
Three sources of parallelism
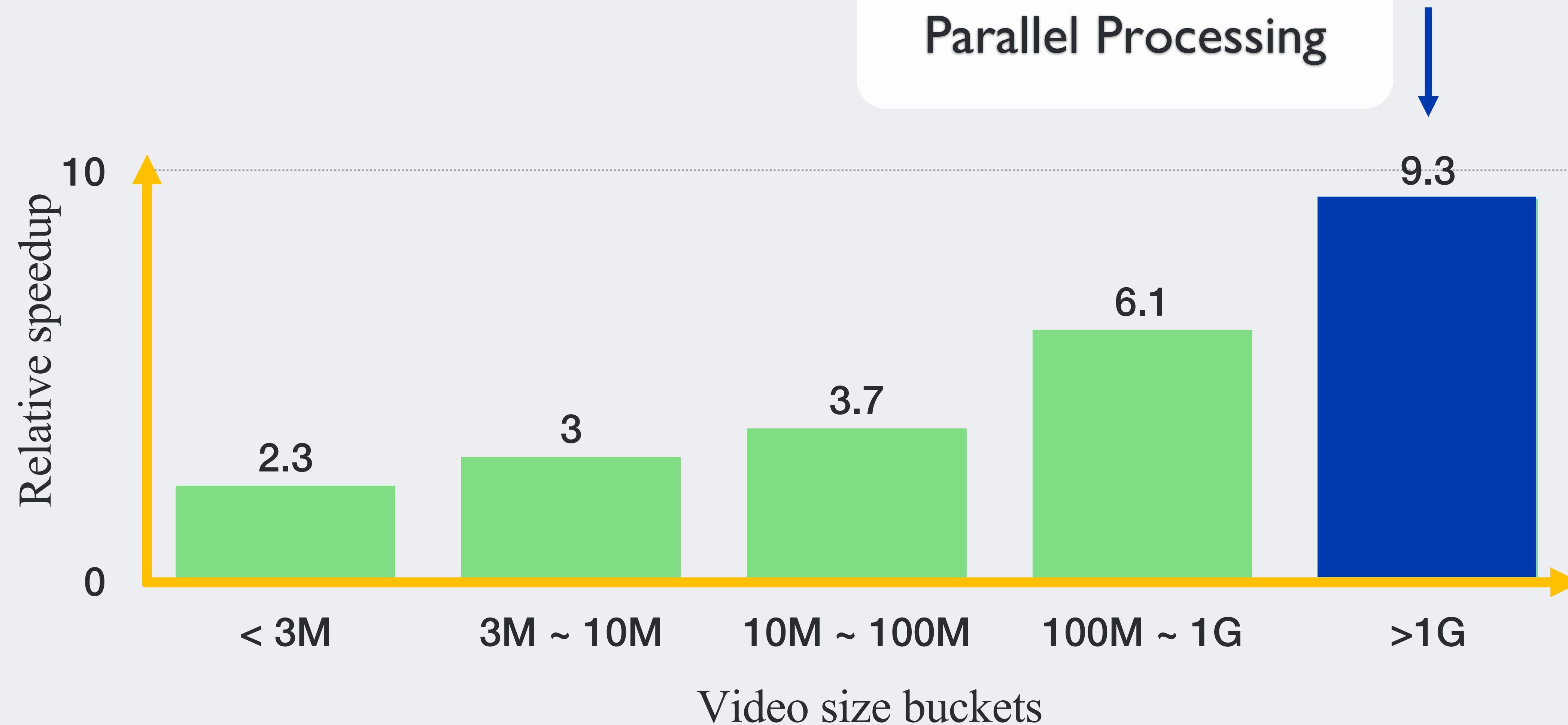
Results: 2.3x ~ 9.3x speedup

Results: 2.3x ~ 9.3x speedup

Results: 2.3x ~ 9.3x speedup

Parallel Processing

Relative speedup

10

9.3

6.1

3.7

3

2.3

0

< 3M   3M ~ 10M   10M ~ 100M   100M ~ 1G   >1G

Video size buckets

Challenges for video processing @ FB

Speedy

Users~~2.3x~~ sh~~9.3x~~ vi**2.3x - 9.3x speedup**uickly

Flexible

Thousands of engineers can write pipelines for tens of apps

Robust
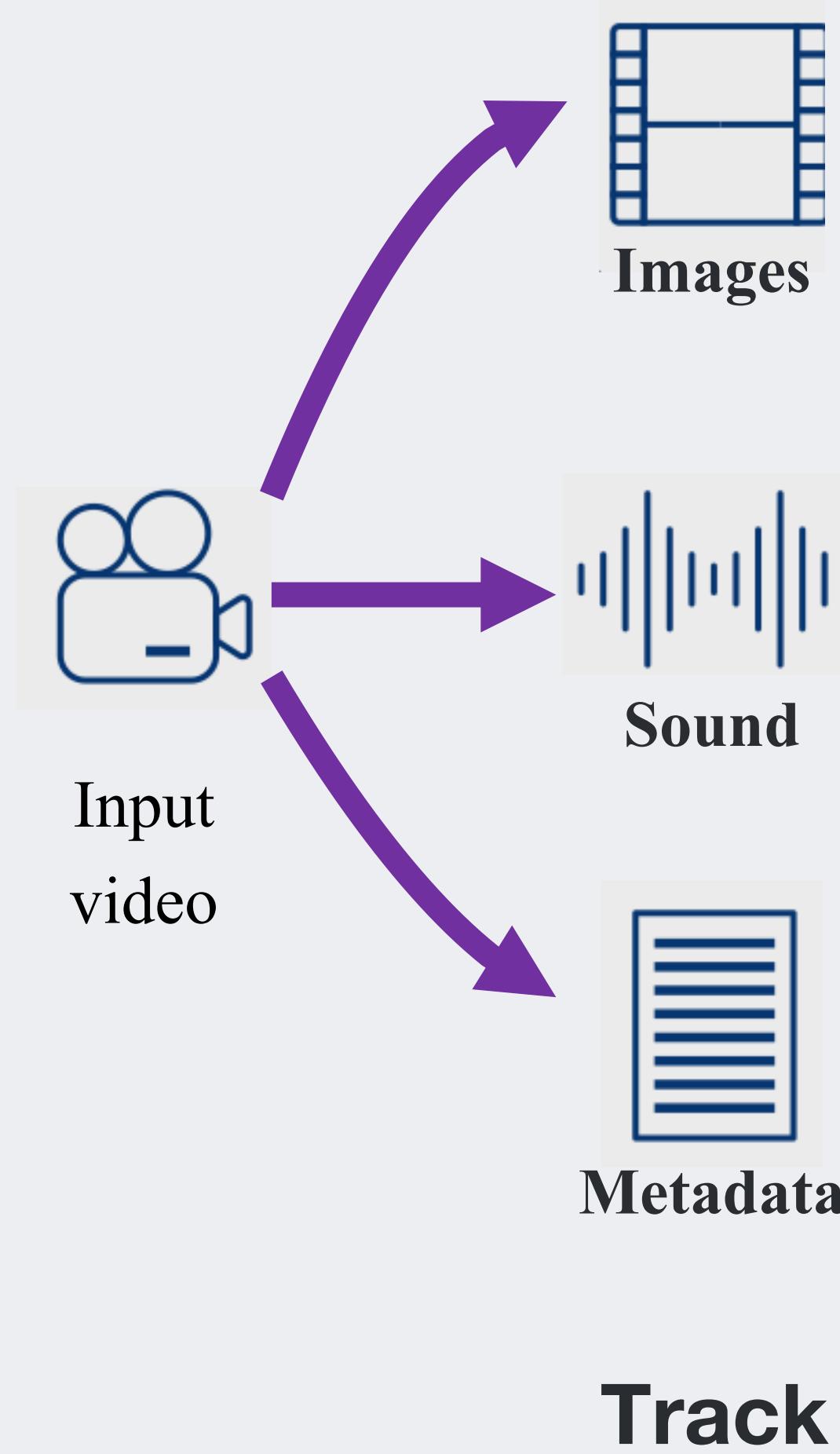
Handle faults and overload that is inevitable at scale

Flexible: build DAG framework

Thousands of engineers can write pipelines for tens of apps
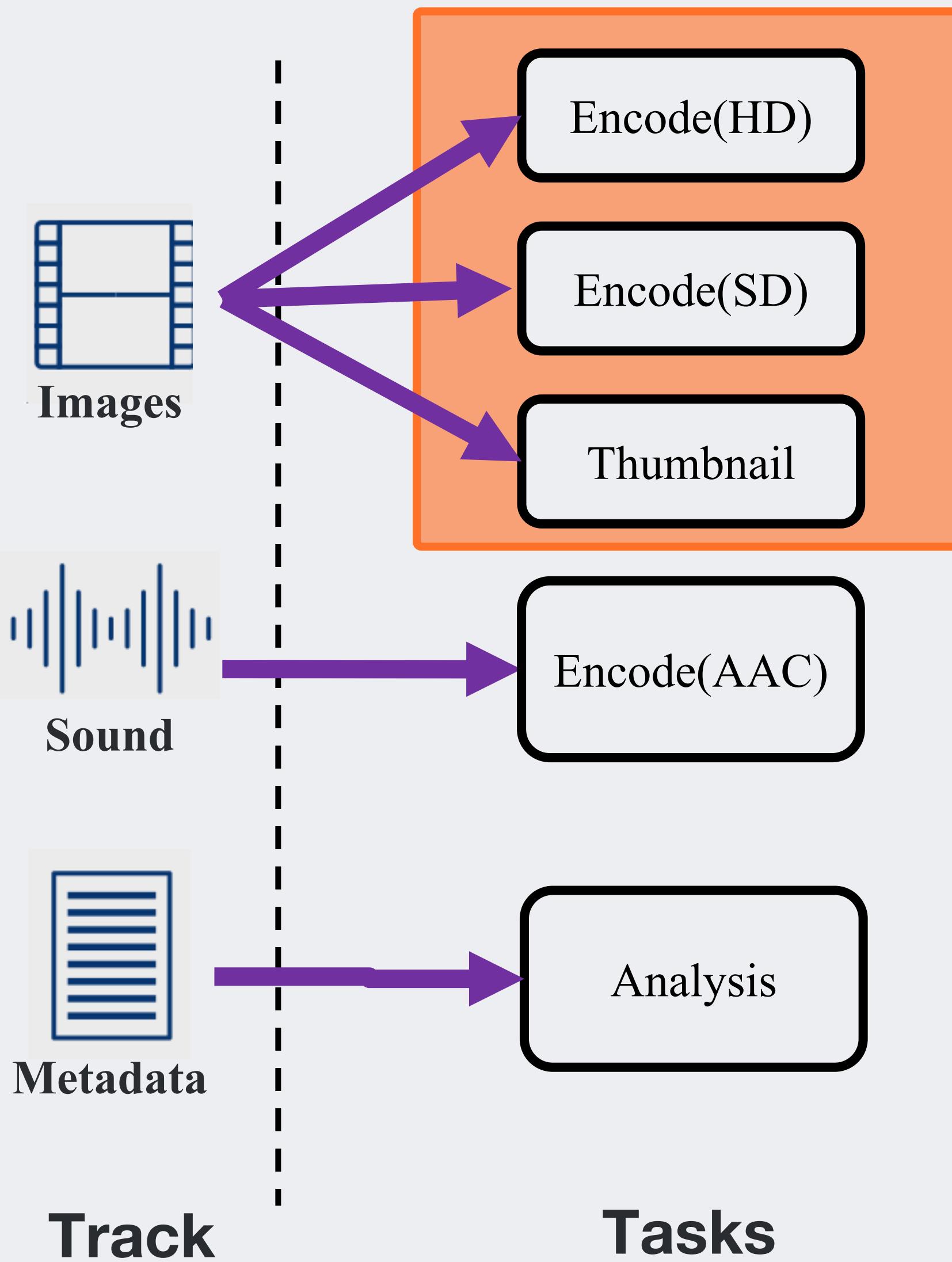
- DAG of computation on the stream-of-tracks abstraction
- Engineers write only sequential tasks in a familiar language
- Dynamic DAG generation per video

DAG on stream-of-tracks abstraction
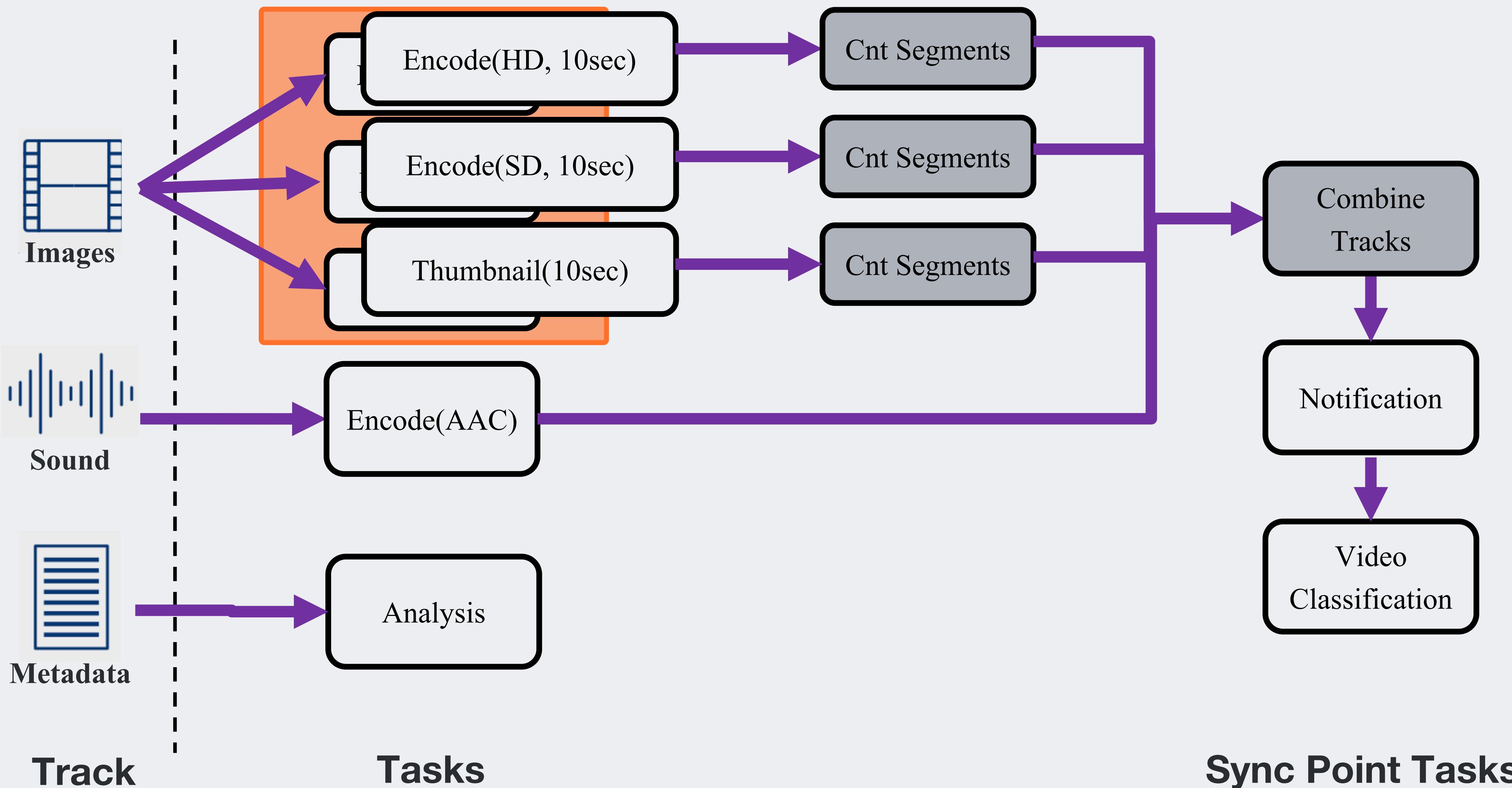
Input
video

Track

Images

Sound

Metadata

```
$pipeline = Pipeline.build()

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

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

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

DAG on stream-of-tracks abstraction

Track   **Tasks**
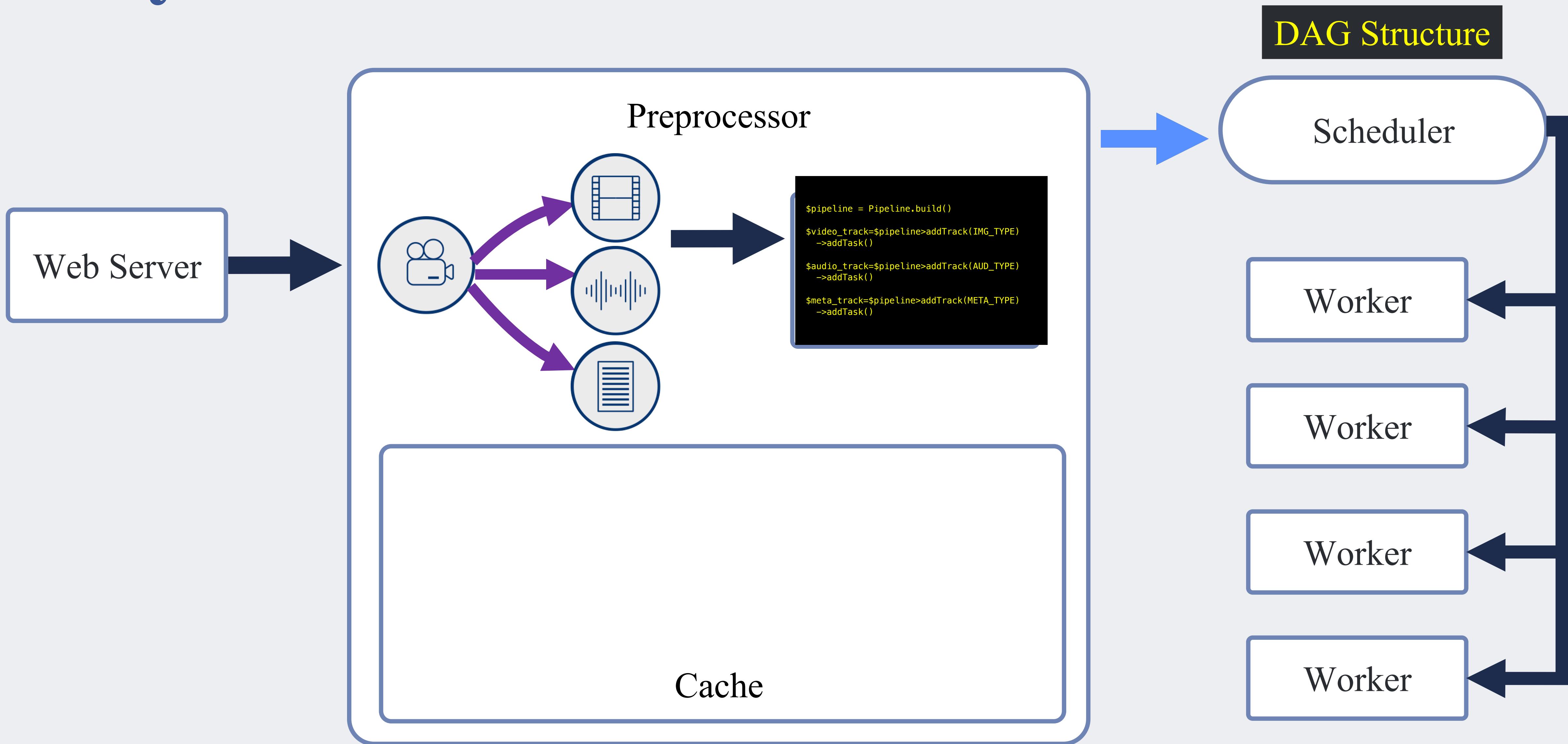
```
$pipeline = Pipeline.build()

$video_track=$pipeline>addTrack(IMG_TYPE)
    ->addTask(Encode(HD, 10s),
              Encode(SD, 10s), Thumb(10s))
$audio_track=$pipeline>addTrack(AUD_TYPE)
    ->addTask(Encode(AAC))

$meta_track=$pipeline>addTrack(META_TYPE)
    ->addTask(Analysis)
```
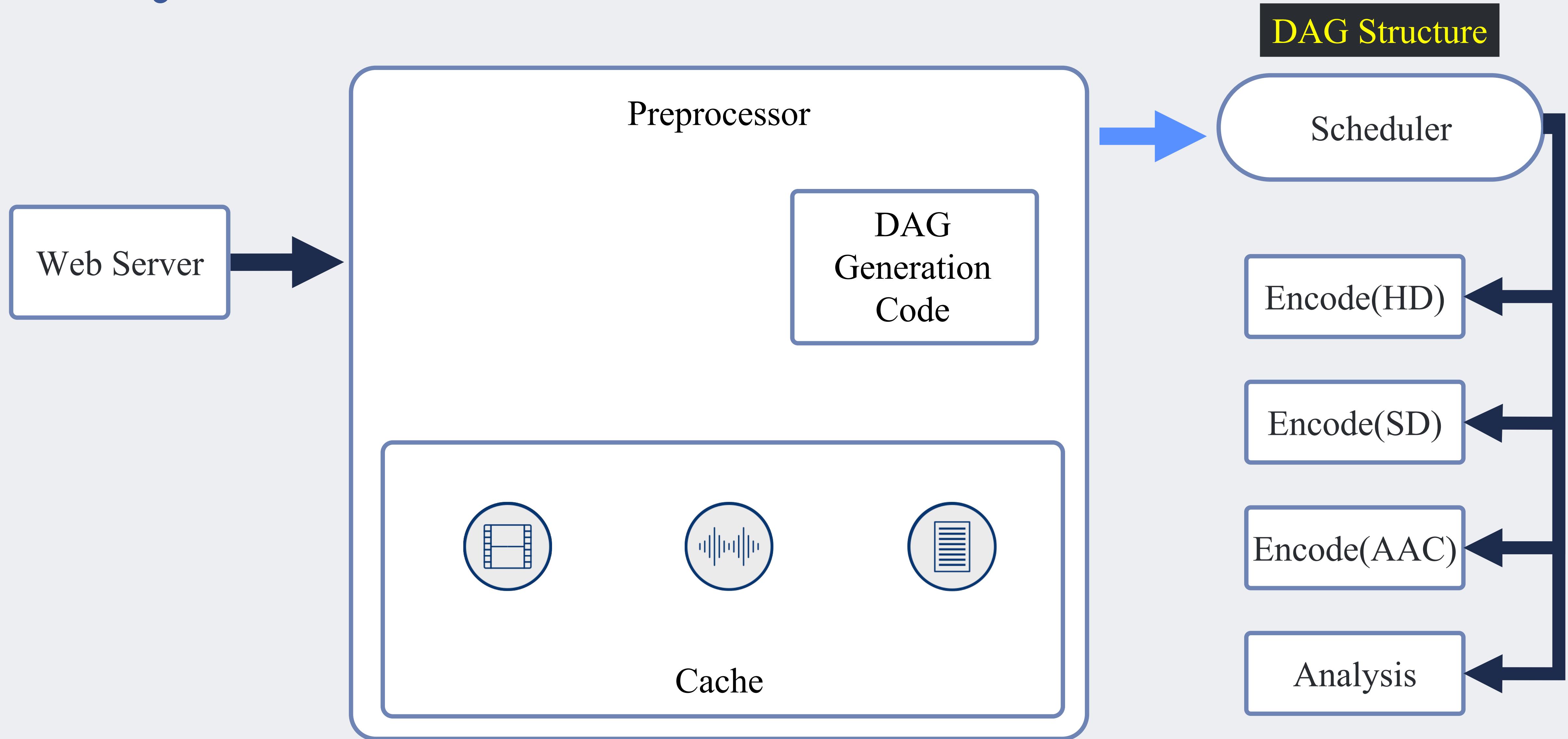
DAG on stream-of-tracks interface

Track   **Tasks**   **Sync Point Tasks**

32

Dynamic DAG Generation

Preprocessor

```
$pipeline = Pipeline.build()

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

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

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

Web Server

Cache

DAG Structure

Scheduler

Worker

Worker

Worker

Worker

33

Dynamic DAG Generation

Preprocessor

DAG Generation Code

Web Server

Cache

DAG Structure

Scheduler

Encode(HD)

Encode(SD)

Encode(AAC)

Analysis

34

Dynamic DAG Generation

Web Server

Preprocessor

```
$pipeline = Pipeline.build()

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

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

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

Scheduler

Worker

Worker

Worker

Worker

Cache

35

One system for 15+ applications

- Generate **billions** of tasks per day
- Varying DAG size
  - 360 video has **thousands** of tasks per upload
  - Newsfeed post averages at **153** tasks per upload
  - Instagram averages at **22** tasks per upload
  - Messenger averages at **18** tasks per upload

Challenges for video processing @ FB

Speedy

2.3x ~ 9.3x speedup

Flexible

Thousands of ~~engineers can write applications~~ **One system for 15+ applications** for tens of apps
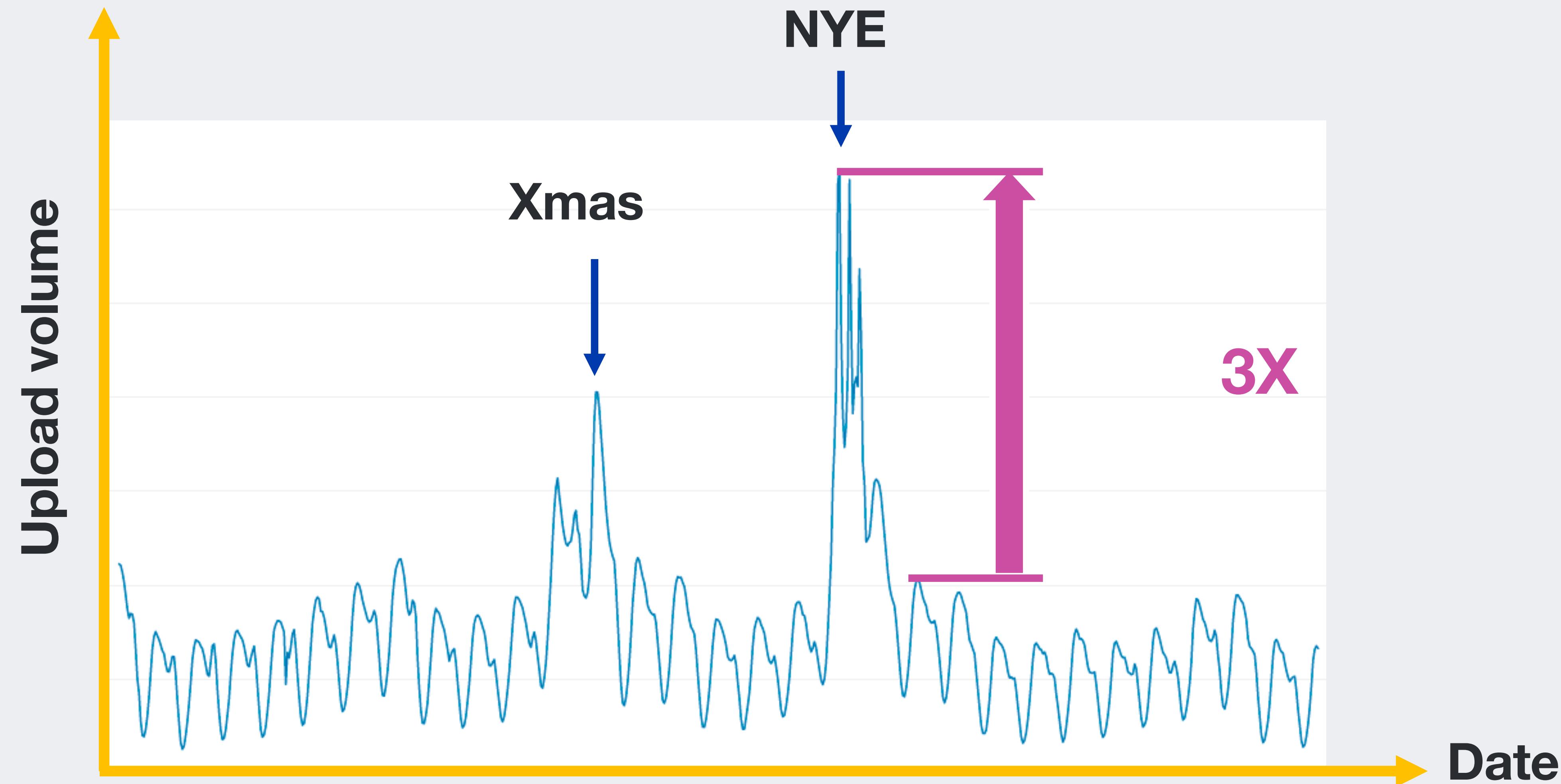
Robust

Handle faults and overload that is inevitable at scale

Robust: tolerate overload
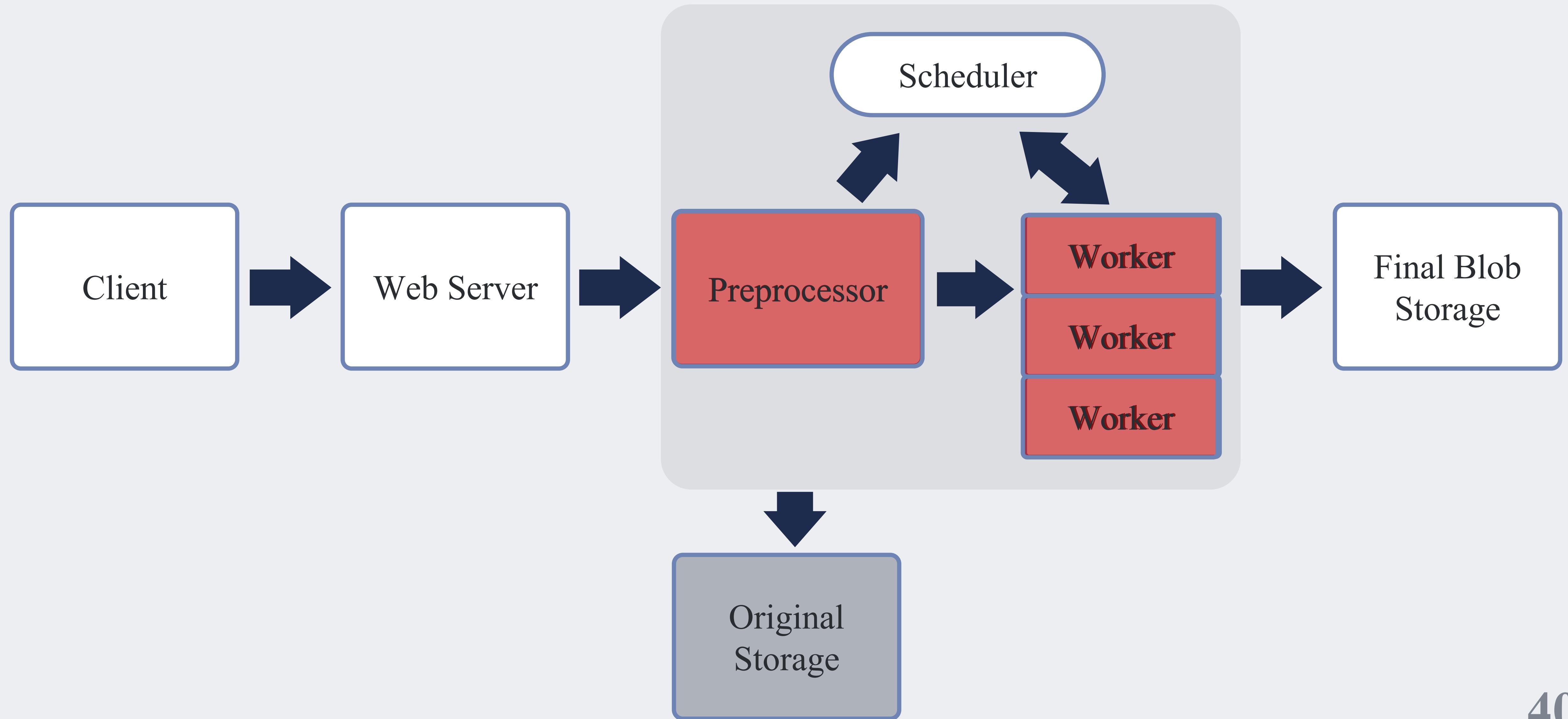
Handle faults and overload that is inevitable at scale

- Rely on priority to degrade non-latency-sensitive tasks
- Defer full video processing for some new uploads
- Load-shedding across global deployments
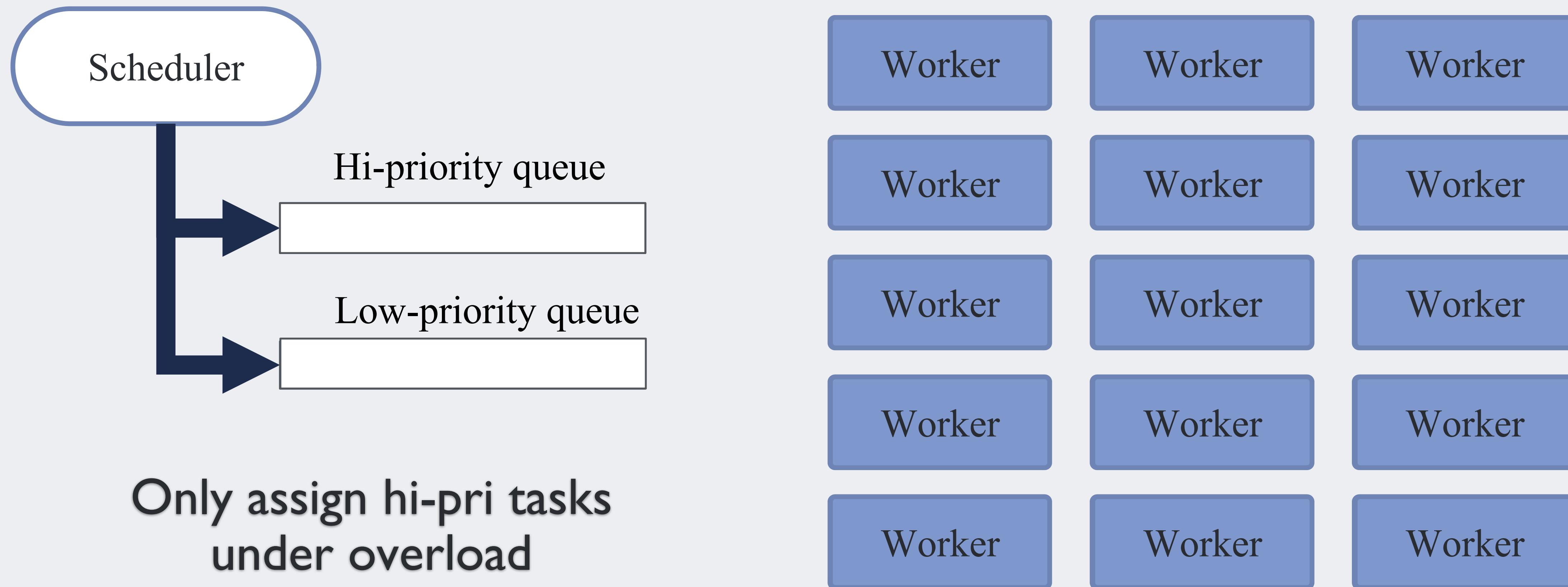
3X peak load during New Year Eve
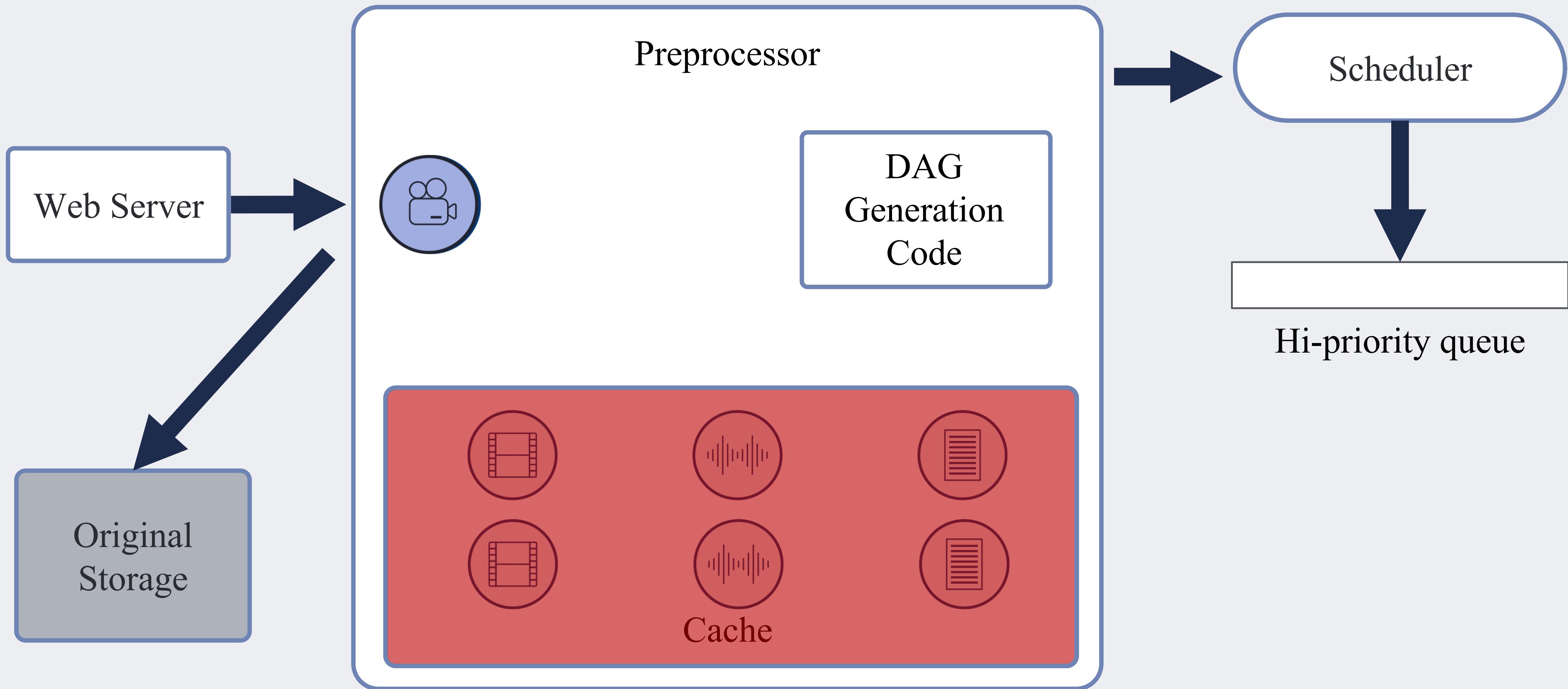
Prepare for overload
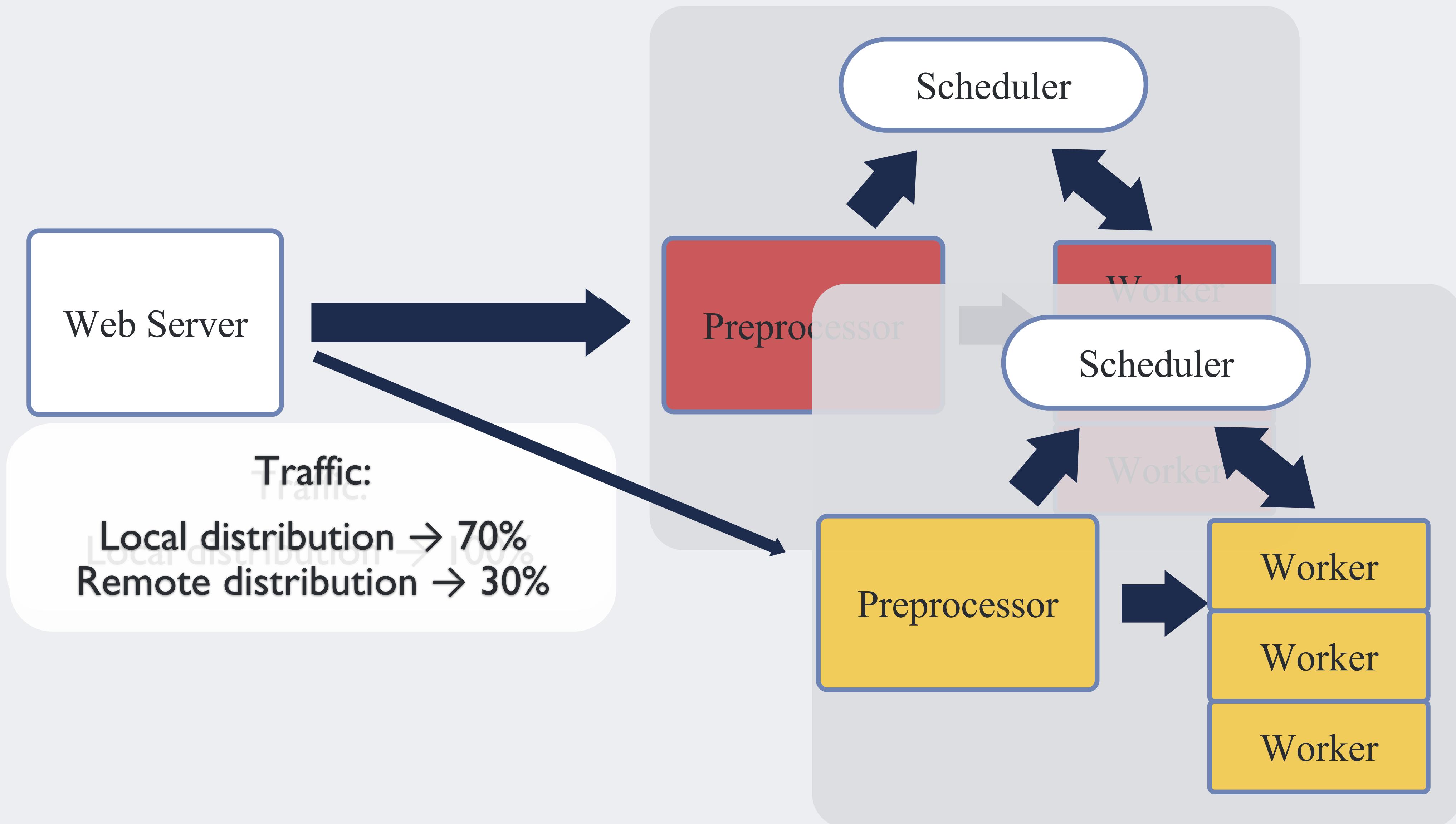
Use priority for worker overload

Scheduler

Hi-priority queue

Low-priority queue

Only assign hi-pri tasks
under overload

Worker	Worker	Worker
Worker	Worker	Worker
Worker	Worker	Worker
Worker	Worker	Worker
Worker	Worker	Worker

Defer full video processing

Regional redirection

Web Server

Scheduler

Preprocessor

Worker

Scheduler

Traffic:
Local distribution → 70%
Remote distribution → 30%

Preprocessor

Worker

Worker

Worker

43

Challenges for video processing @ FB

Speedy

2.3x ~ 9.3x speedup

Flexible

One system for 15+ applications

Robust

Handle faults **Tolerate 3x traffic spike** inevitable at scale

More details in paper

- Advanced DAG control
  - Task group: batch multiple tasks for schedule
  - Priority control: annotate latency-sensitive task
  - Optional task: okay to fail or skip
  - Customizable error handling: early termination
- Failure monitoring and recovery
- Overload scenario caused by Kraken and system bugs
- Lessons learned

Failures from Global Inconsistencies

Lesson Learned

```
preprocessor = null

if (segment.is_first_segment()) {
  preprocessor = get_preprocessor()
  storage_write(video_id, "preprocessor", preprocessor)
} else {
  preprocessor = storage_read(video_id, "preprocessor")
}

forward_segment(preprocessor, segment)
```

One preprocessor handles all segments of one video

Mapping from video to preprocessor determined when upload starts

Storage system is eventually consistent, what could go wrong?

Related work

- Batch processing

  SVE overlaps data ingestion and processing

- Stream processing

  SVE offers dynamic DAG generation per input

- Video processing at scale

  SVE support many production apps

46

Streaming Video Engine

- Deployed in production for 2 years

- Speedy to enable users to share videos quickly
  - Harness parallelism in upload, processing, and storage

- Flexible to support 15 app with tens of millions of uploads/day
  - Dynamic DAG generation on the stream-of-tracks abstraction

- Robust to tolerate faults and overload at scale
  - Prioritize processing and then shed load to other DCs or the future