in f ra

SVE: Distributed Video Processing at Facebook Scale

Qi Huang

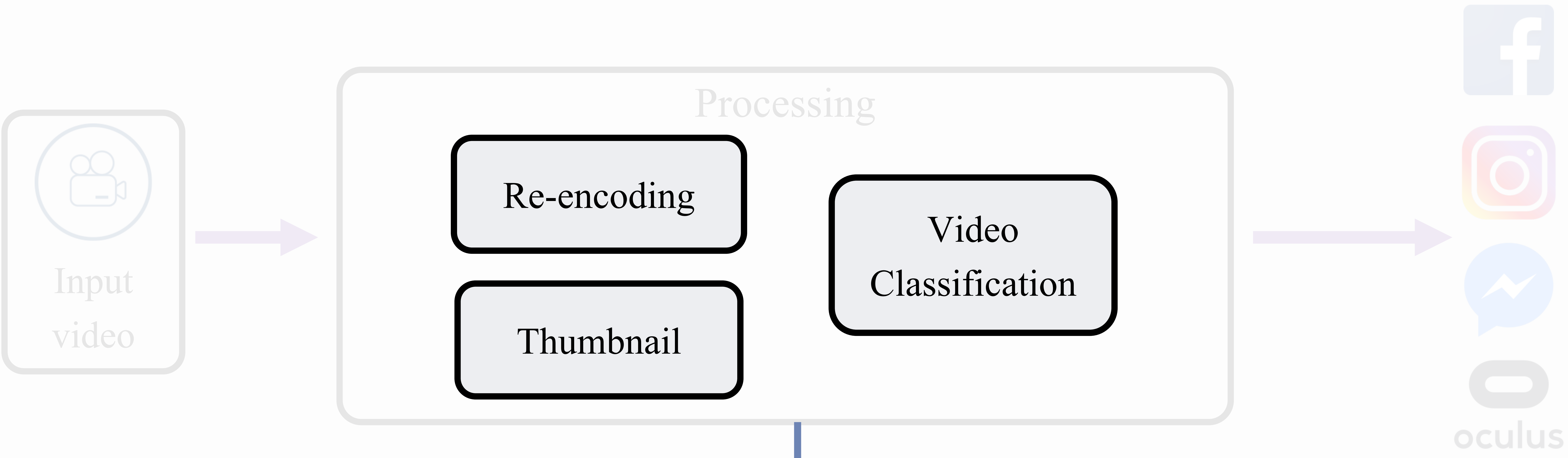
Petchean Ang, Peter Knowles, Tomasz Nykiel, Iaroslav Tverdokhlib, Amit Yajurvedi, Paul Dapolito IV, Xifan Yan, Maxim Bykov, Chuen Liang, Mohit Talwar, Abhishek Mathur, Sachin Kulkarni, Matthew Burke, Wyatt Lloyd

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

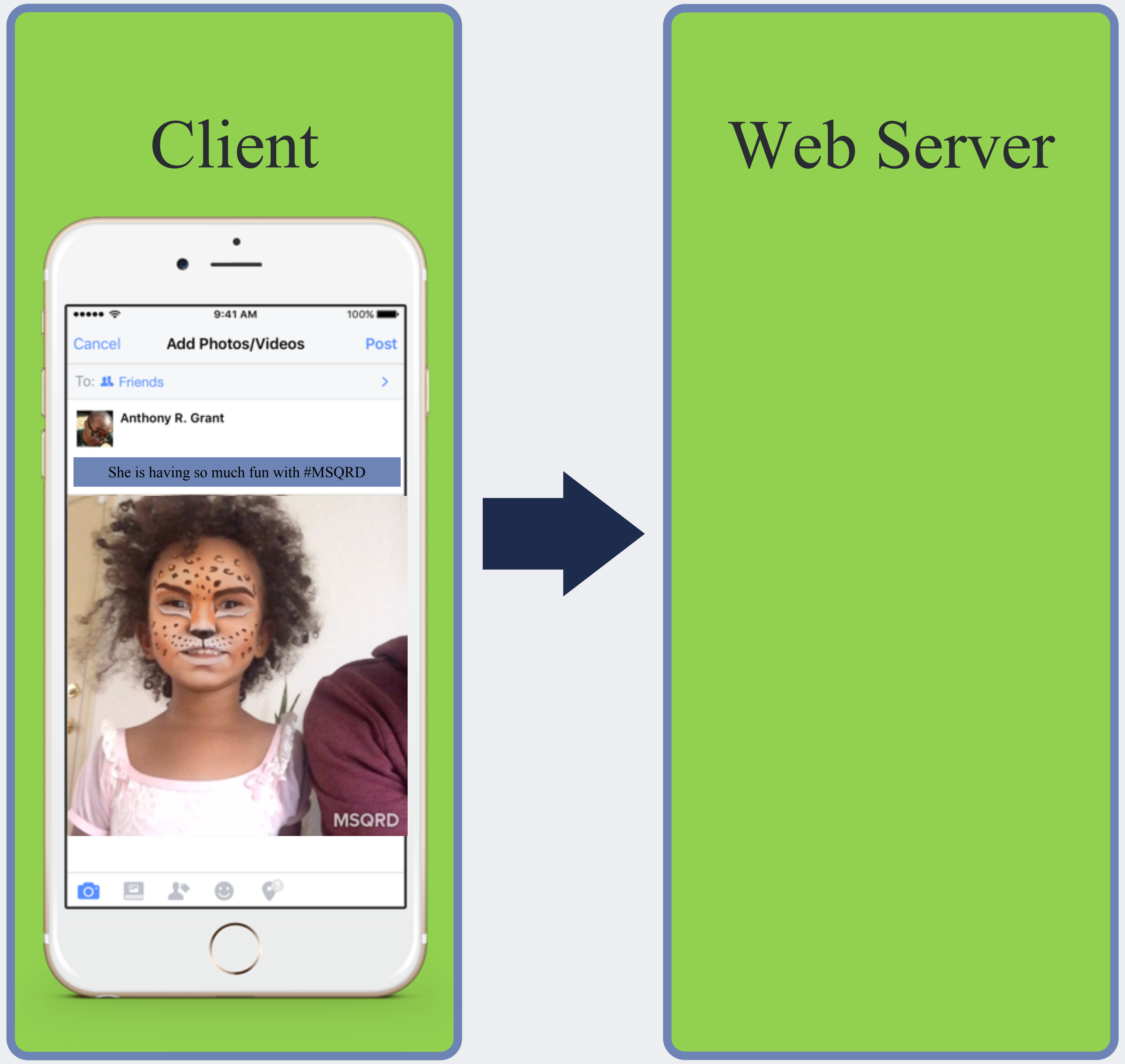
Video Classification

oculus

Pt. 1
Legacy System
Scaling Challenges

Pt. 2
SVE
Impact of Design

Legacy: upload video file to web server

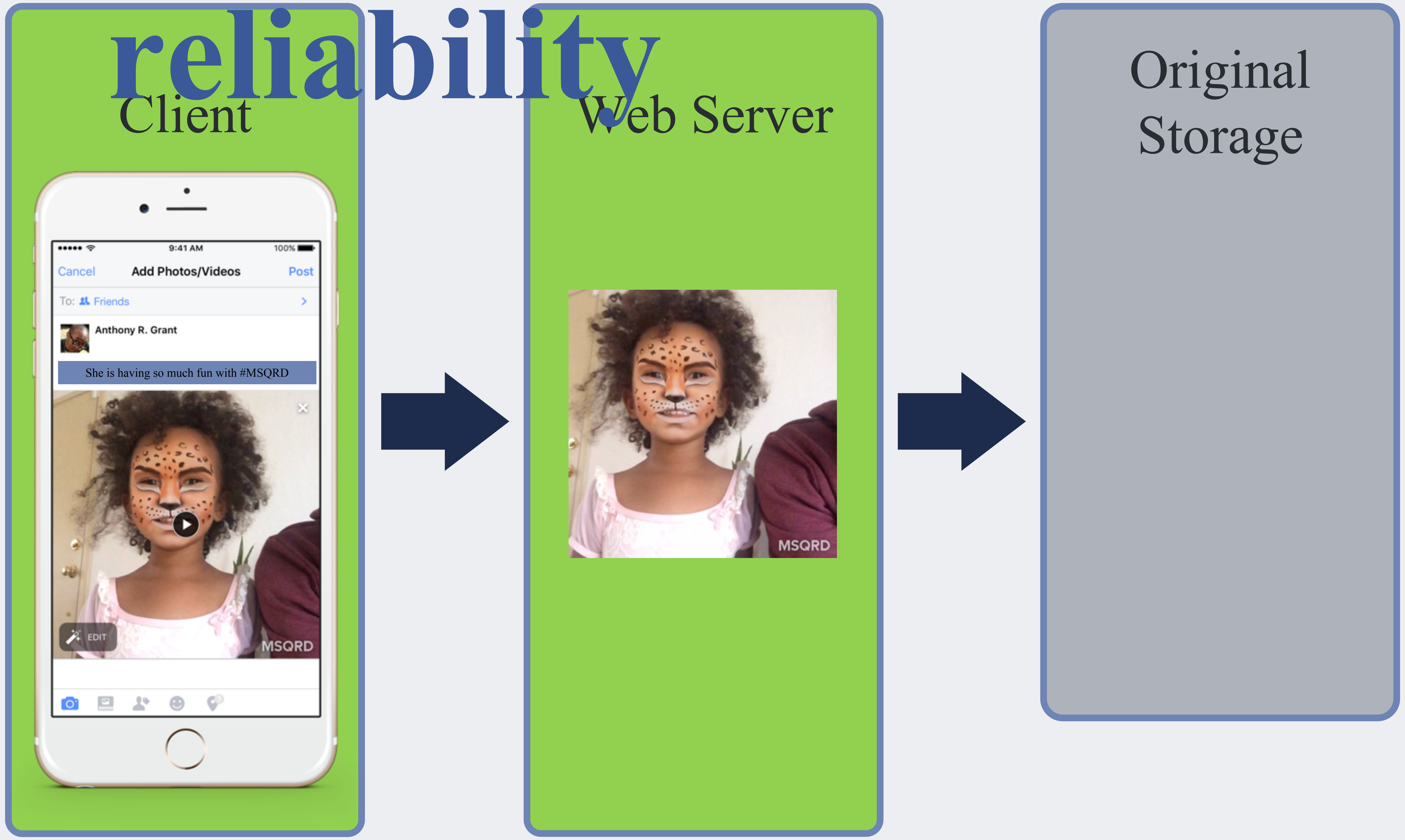


Client

Web Server

She is having so much fun with #MSQRD

Legacy: preserve original for reliability

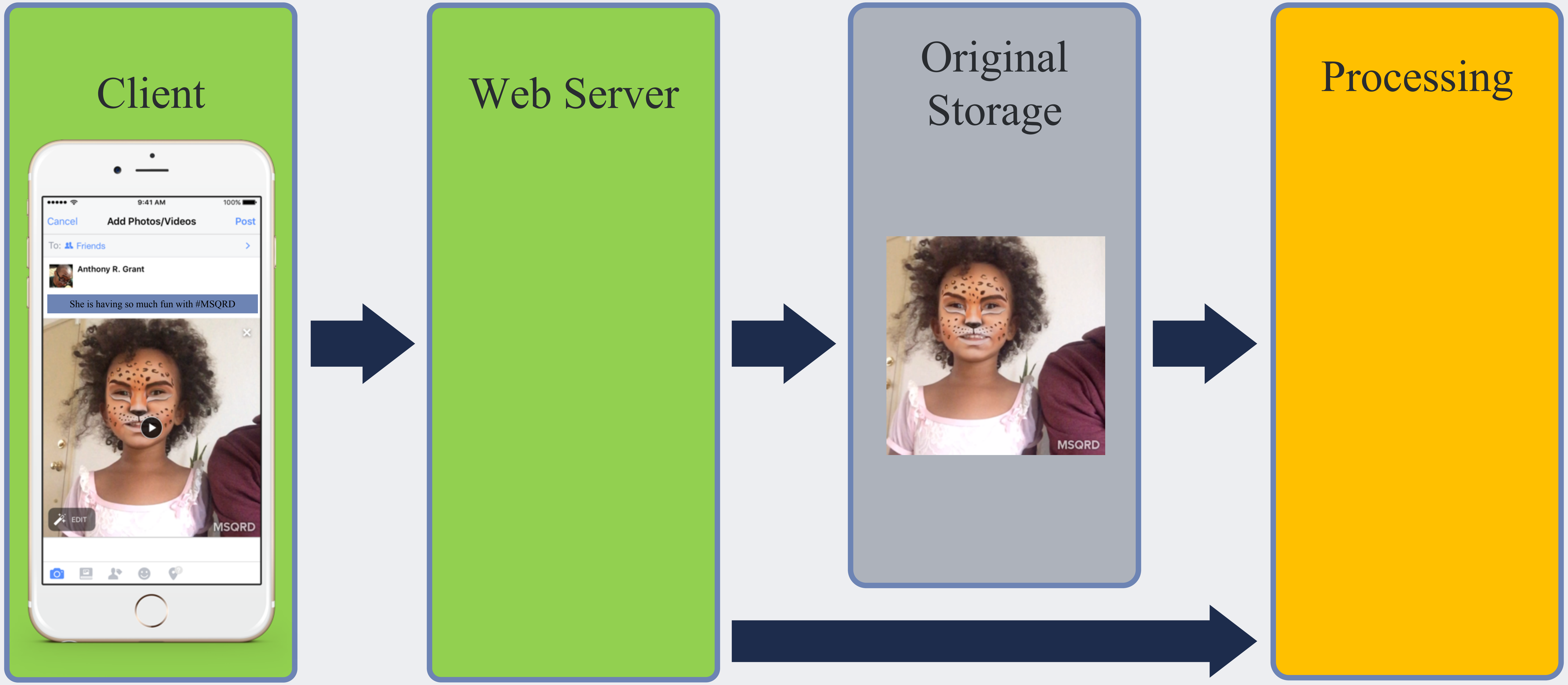


Client

Web Server

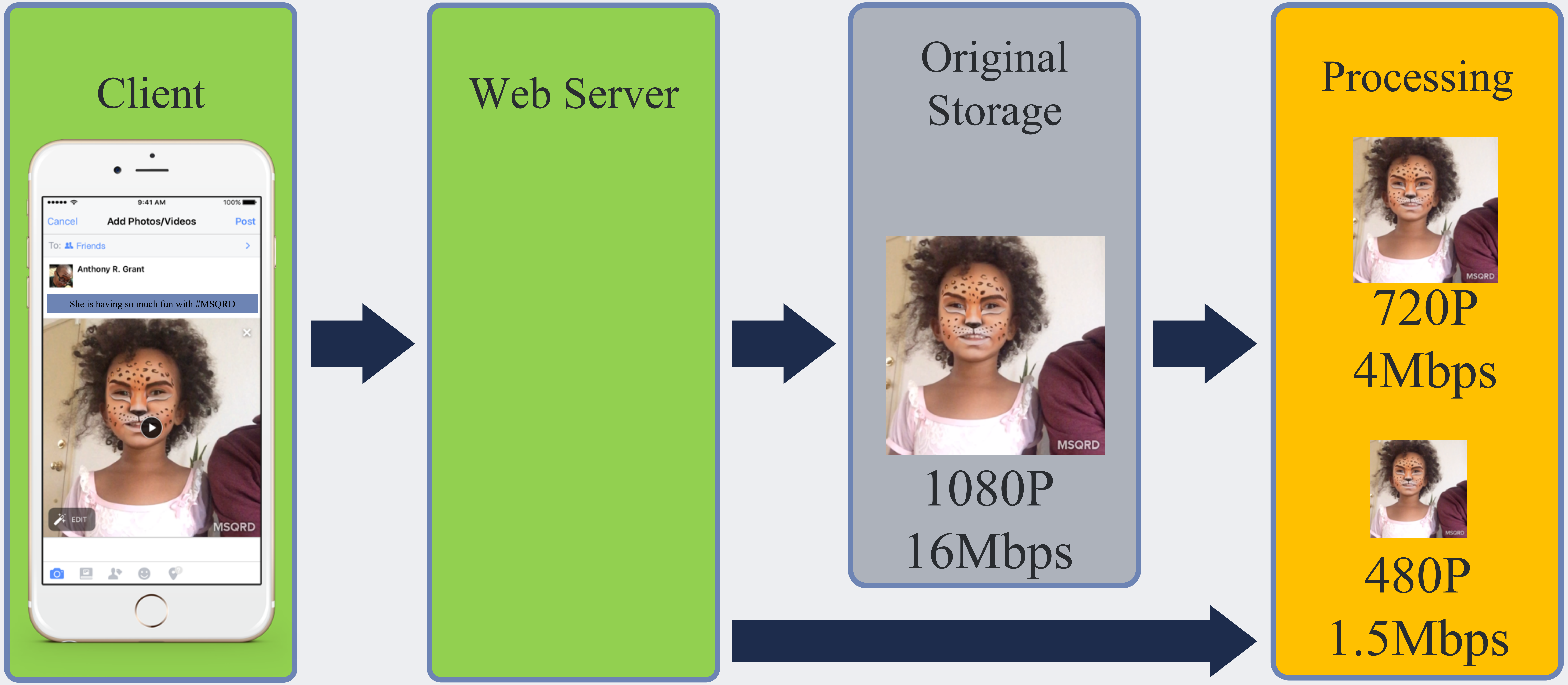
Original Storage

Legacy: process after upload completes



She is having so much fun with #MSQRD

Legacy: encode w/ varying bitrates



Client

Web Server

Original Storage

1080P
16Mbps

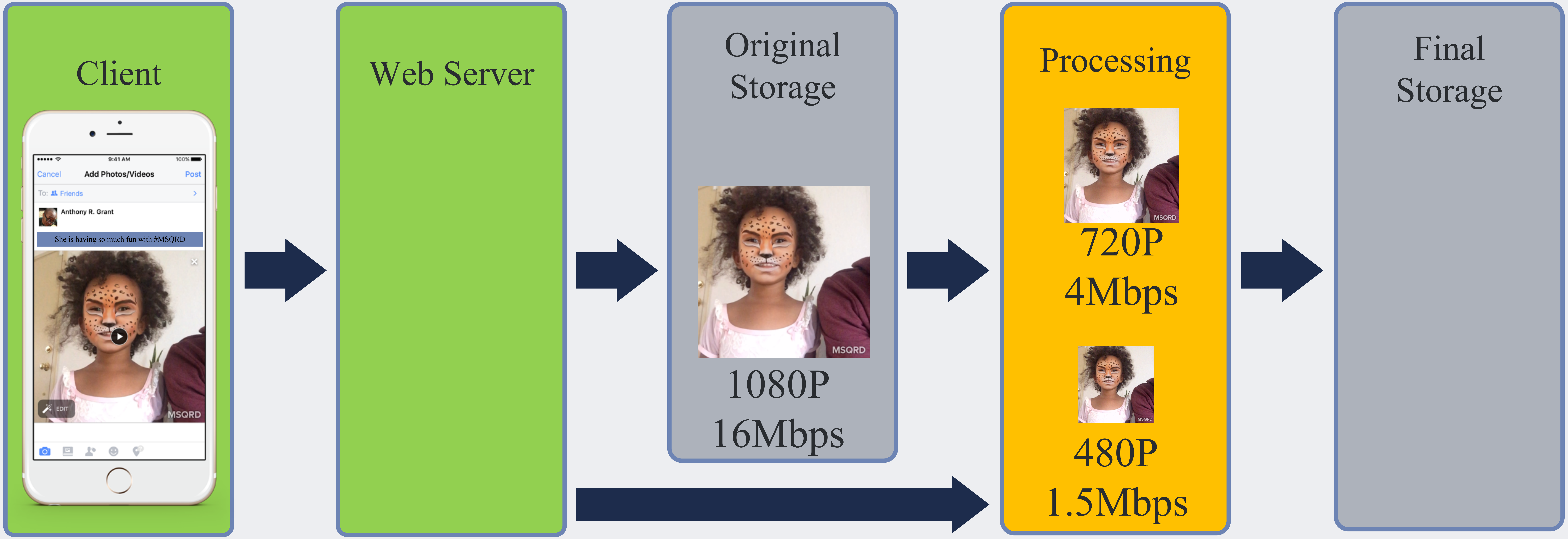
Processing

720P
4Mbps

480P
1.5Mbps

She is having so much fun with #MSQRD

Legacy: store encodings before sharing



Client

Web Server

Original Storage

1080P
16Mbps

Processing

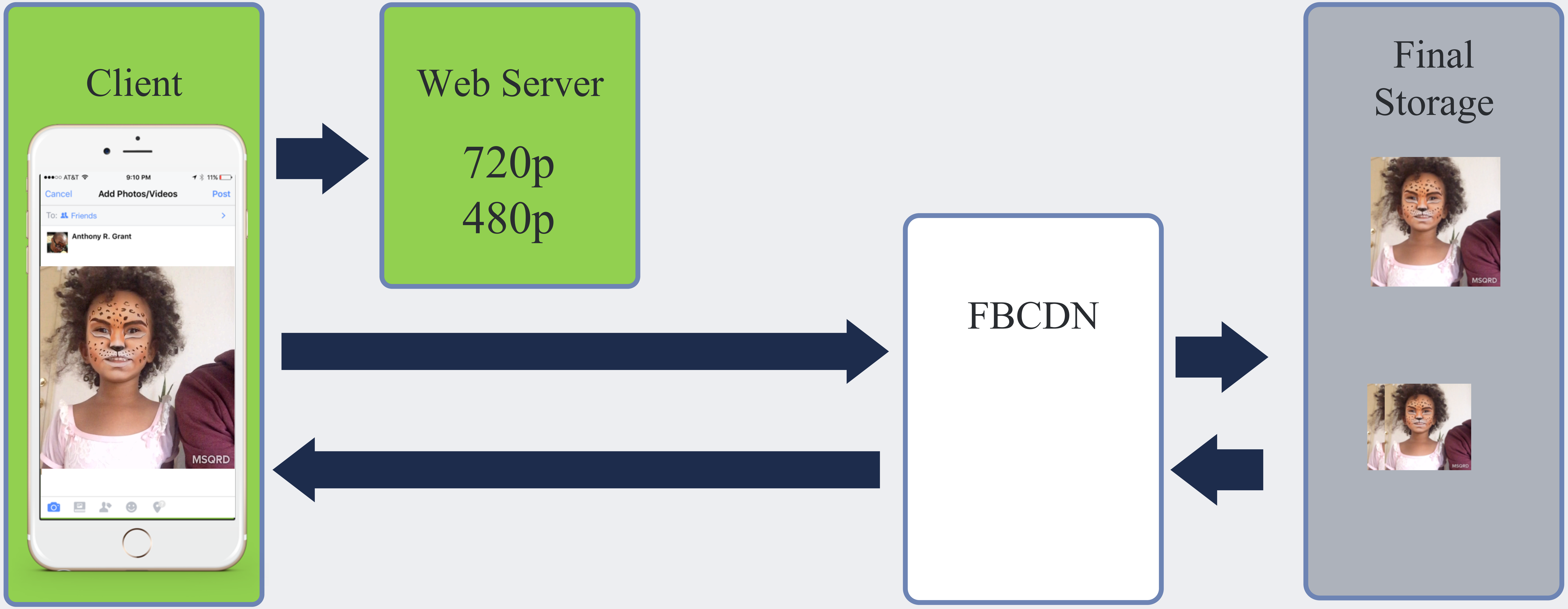
720P
4Mbps

480P
1.5Mbps

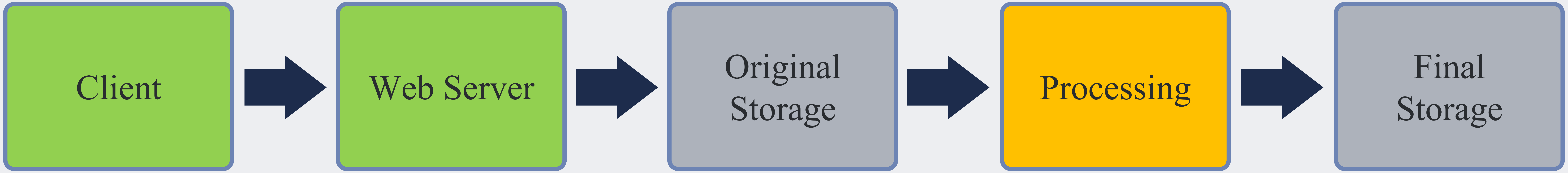
Final Storage

She is having so much fun with #MSQRD

Sharing with adaptive streaming

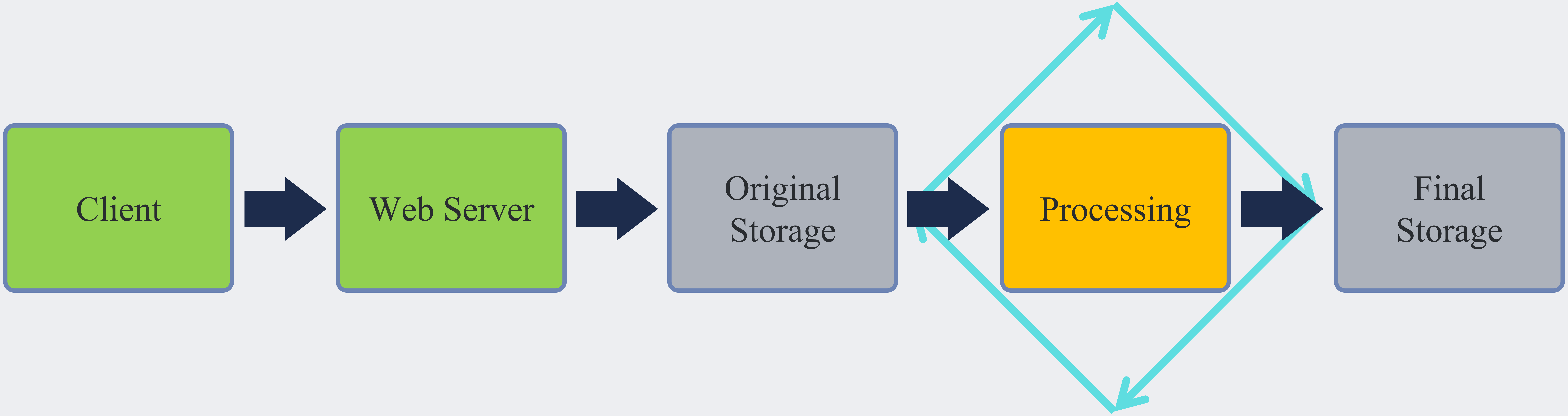


Focus: pre-sharing pipeline

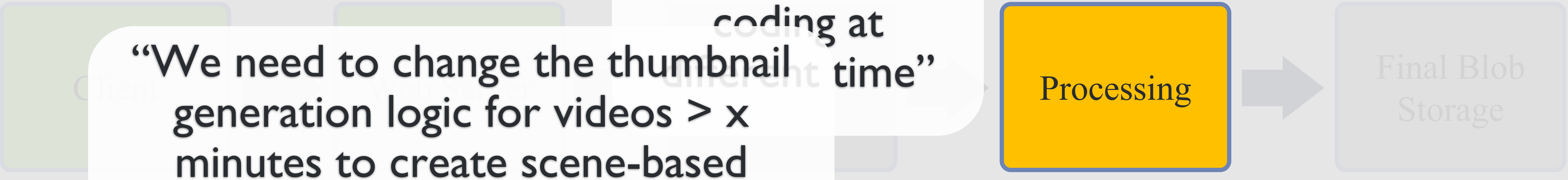


All steps from when a user starts an upload until a video is ready to be shared

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”

“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”

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

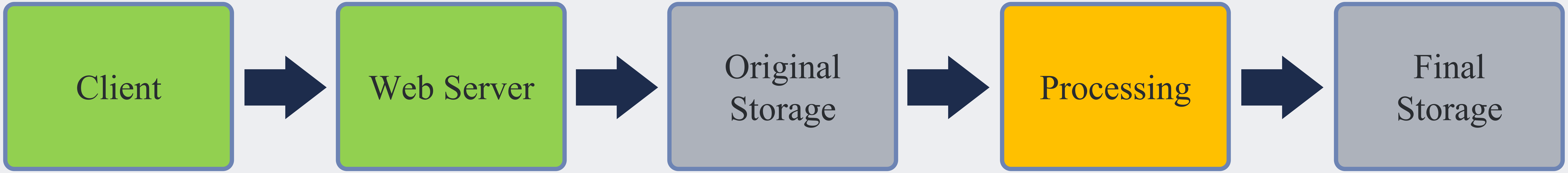
Our Streaming Video Engine (SVE) is speedy, flexible, and robust

Speedy: harness parallelism

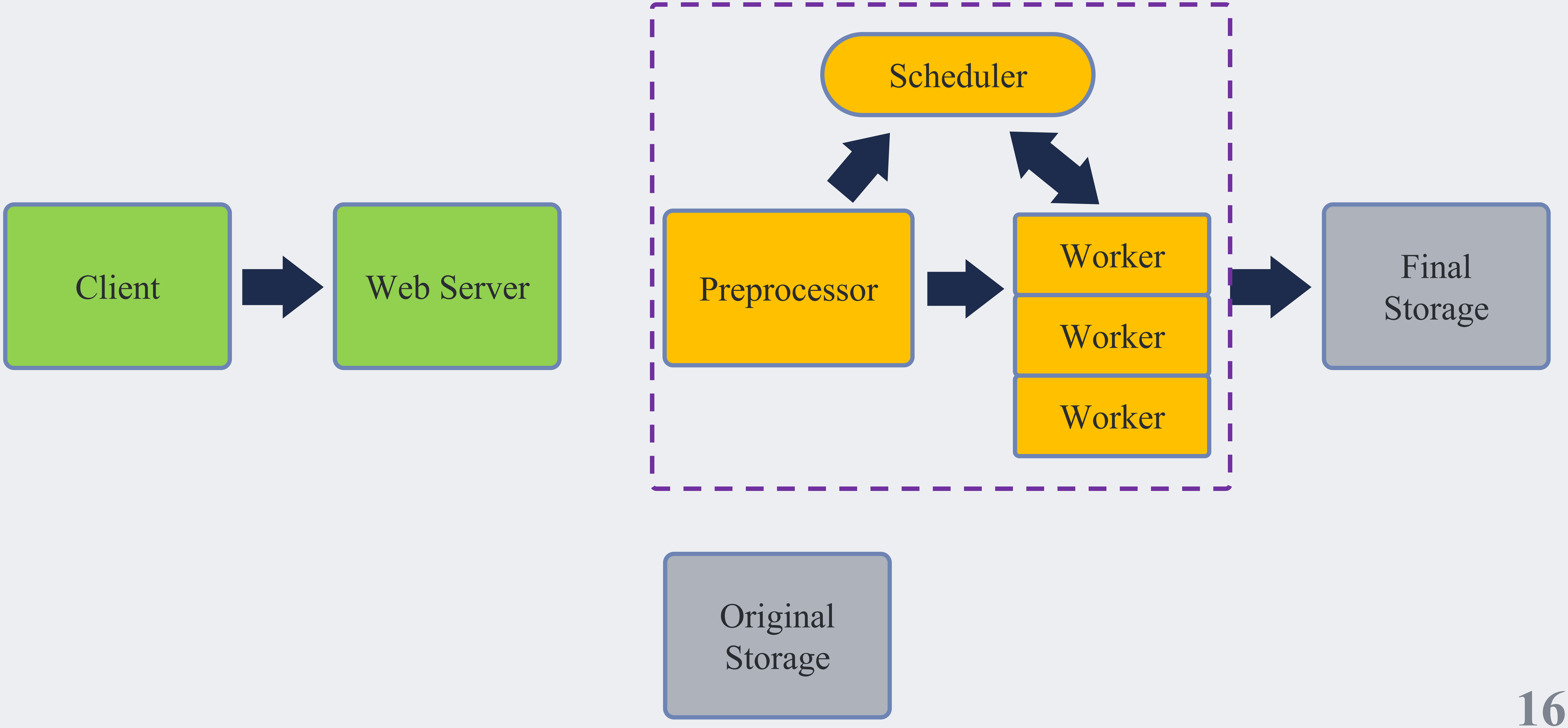
Users can share videos quickly

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

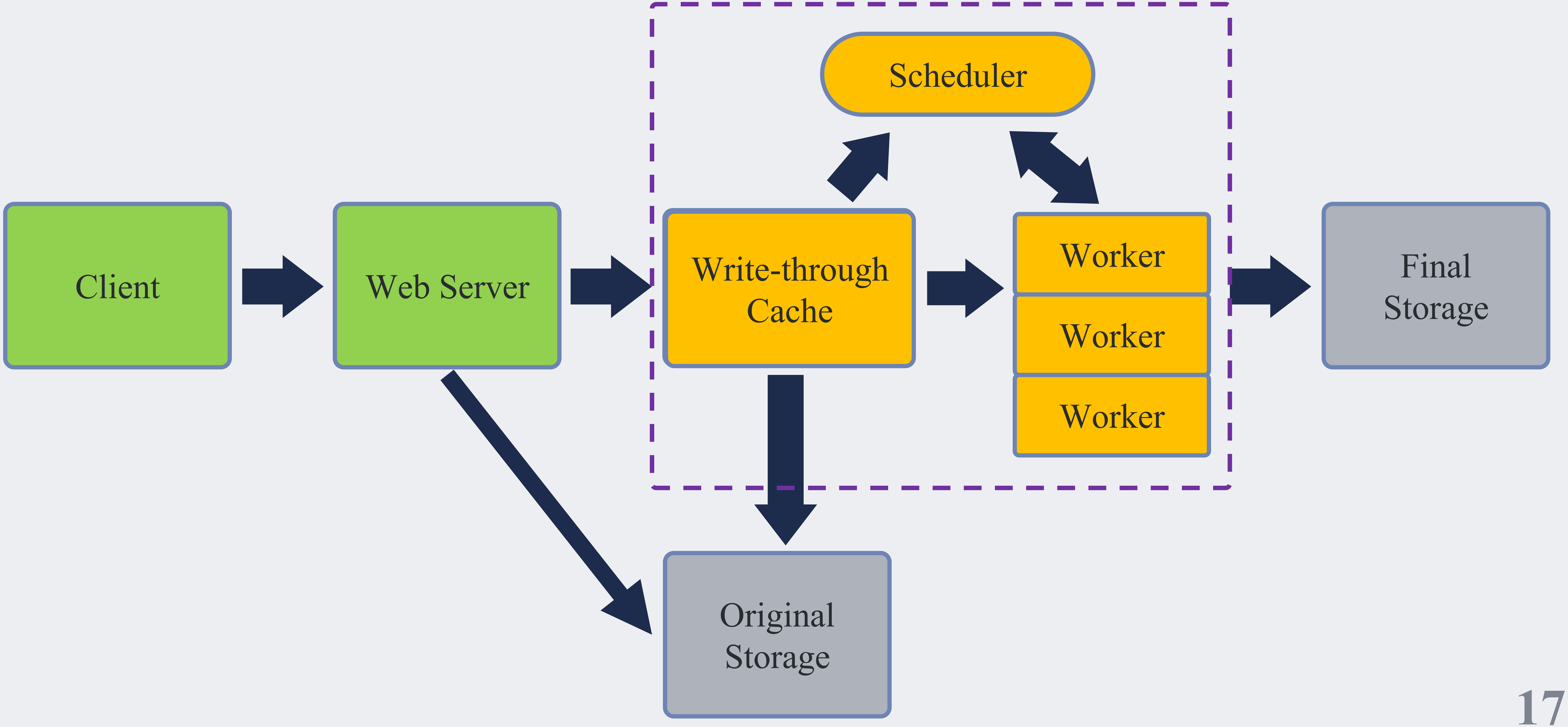
Architectural changes for parallelism



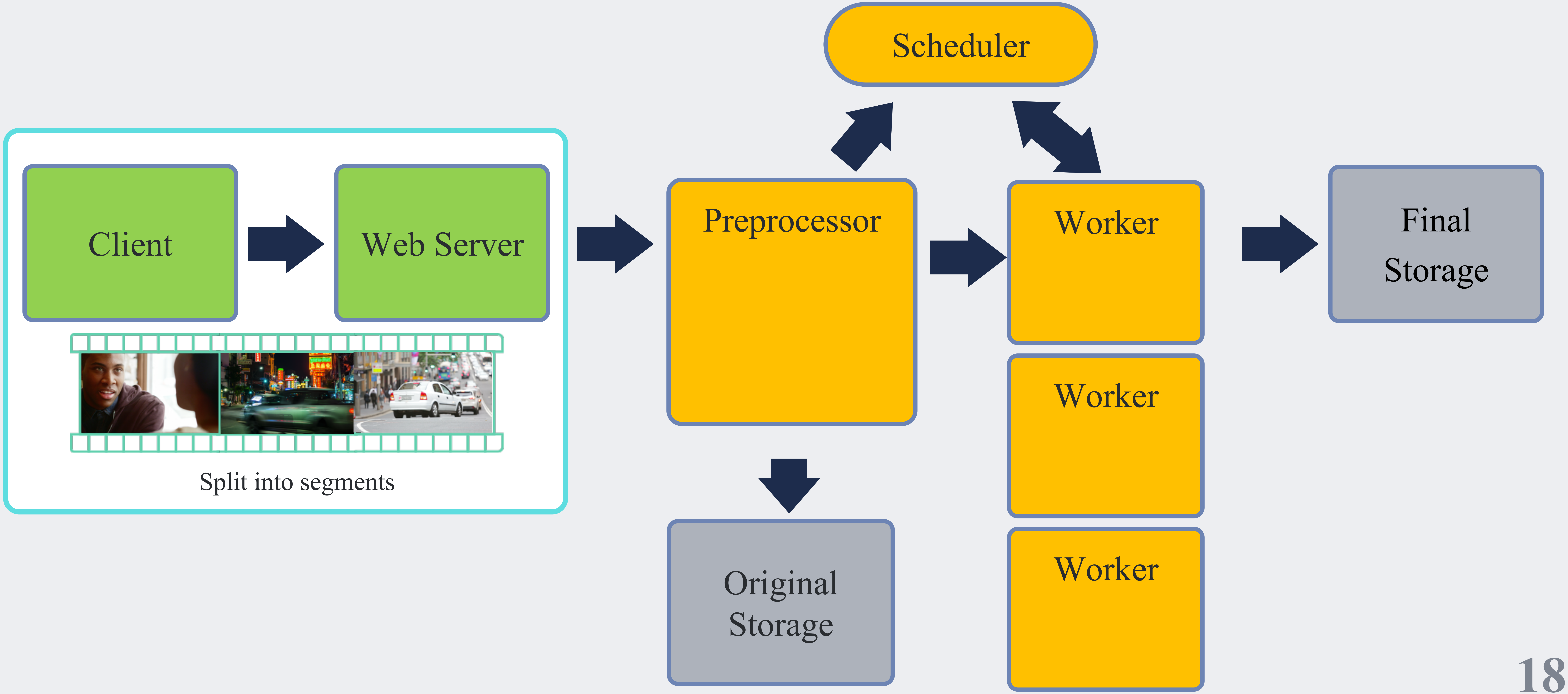
Architectural changes for parallelism



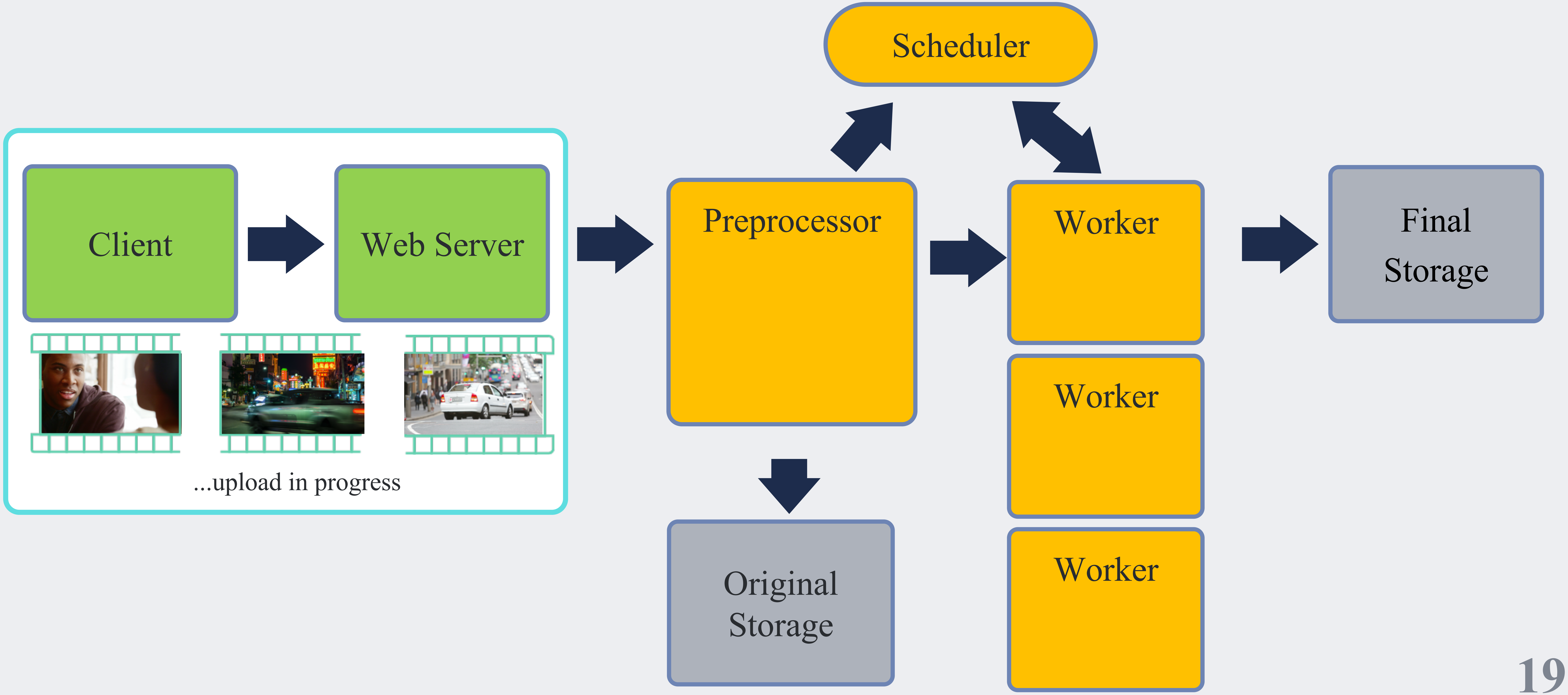
Overlap fault tolerance and processing



Overlap upload and processing



Overlap upload and processing



Scheduler

Client

Web Server

...upload in progress

Preprocessor

Worker

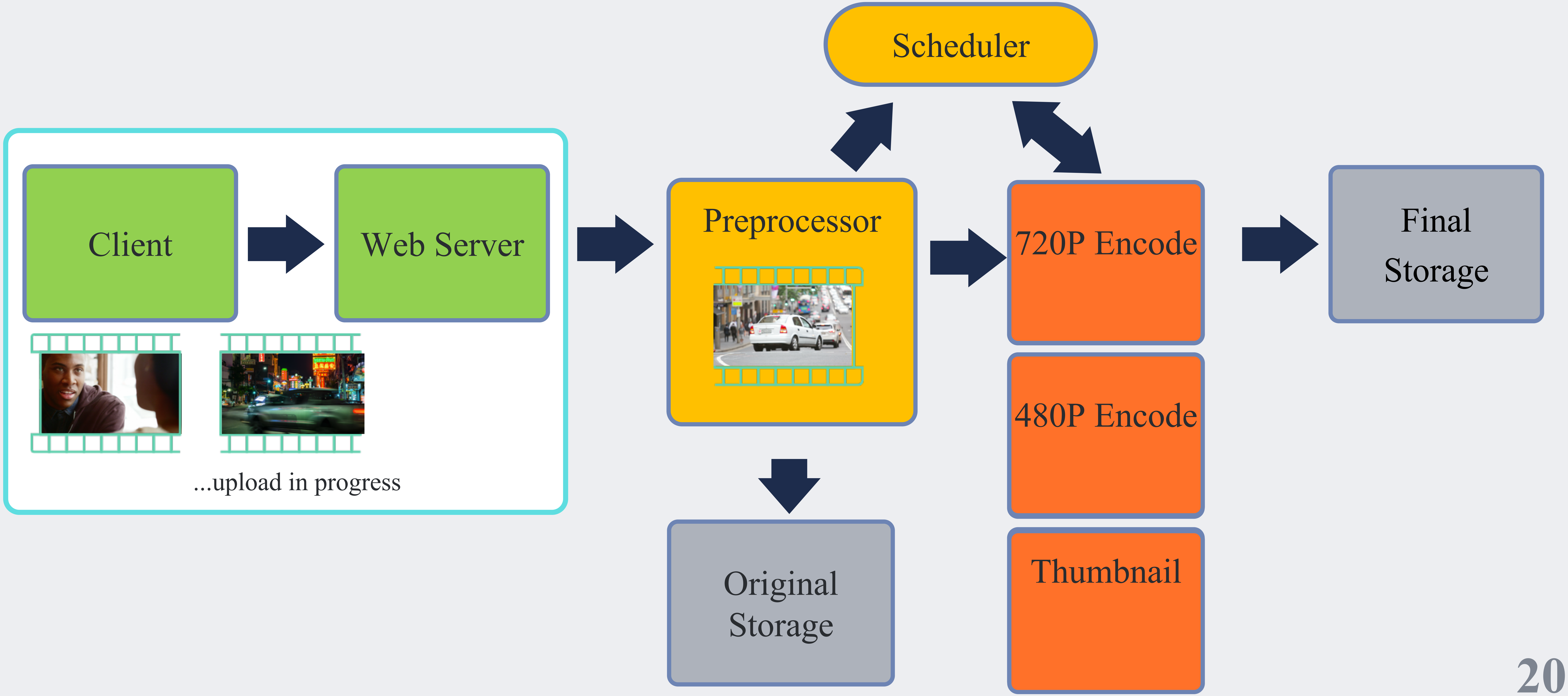
Worker

Worker

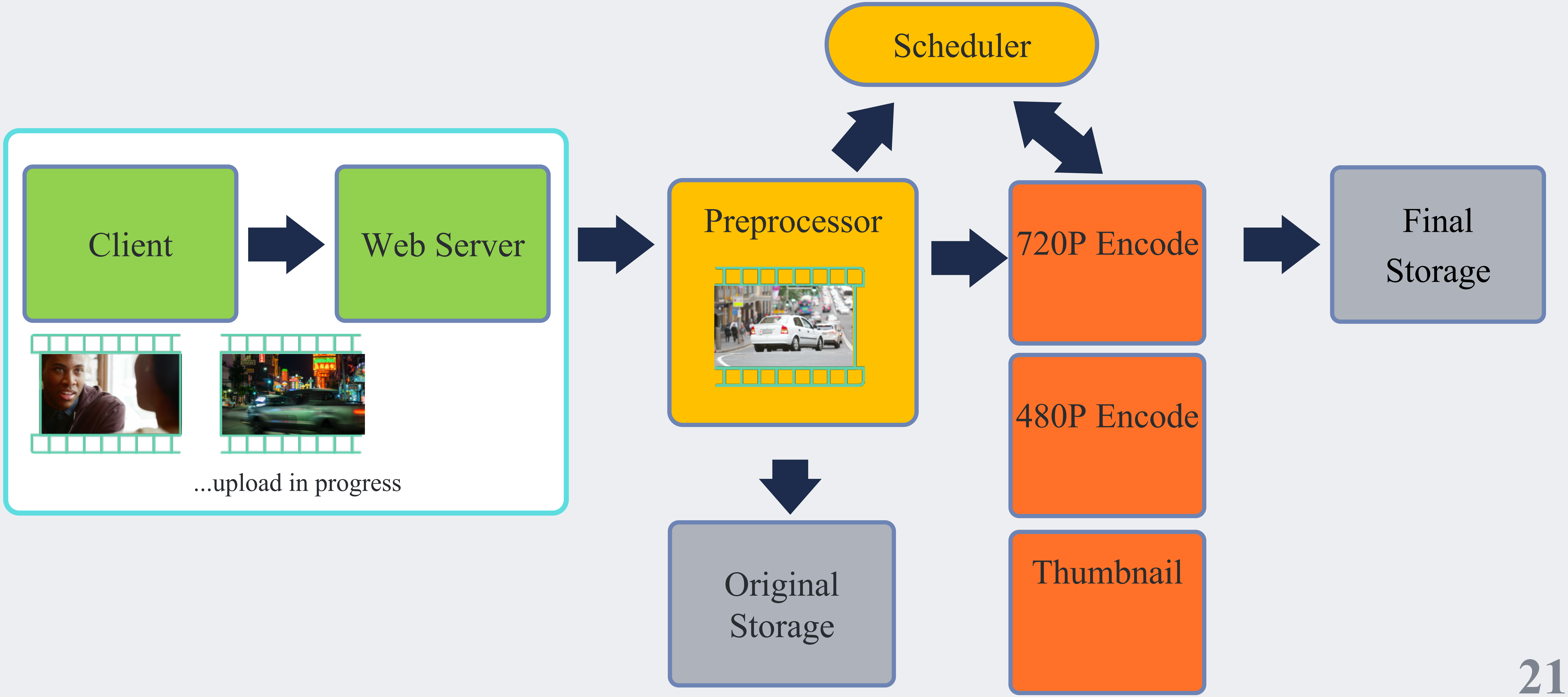
Final Storage

Original Storage

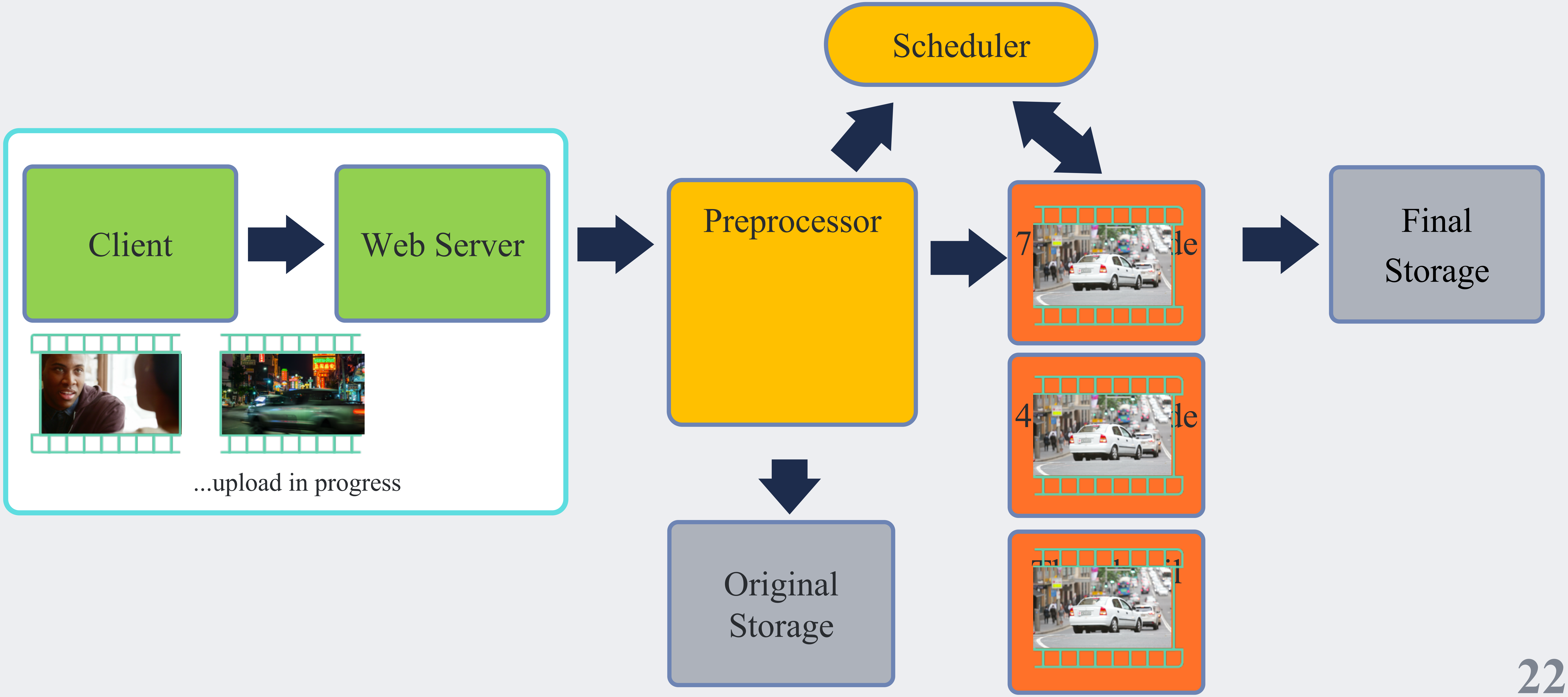
Parallel processing w/ many workers



Parallel processing w/ many workers



Parallel processing w/ many workers



Scheduler

Client

Web Server

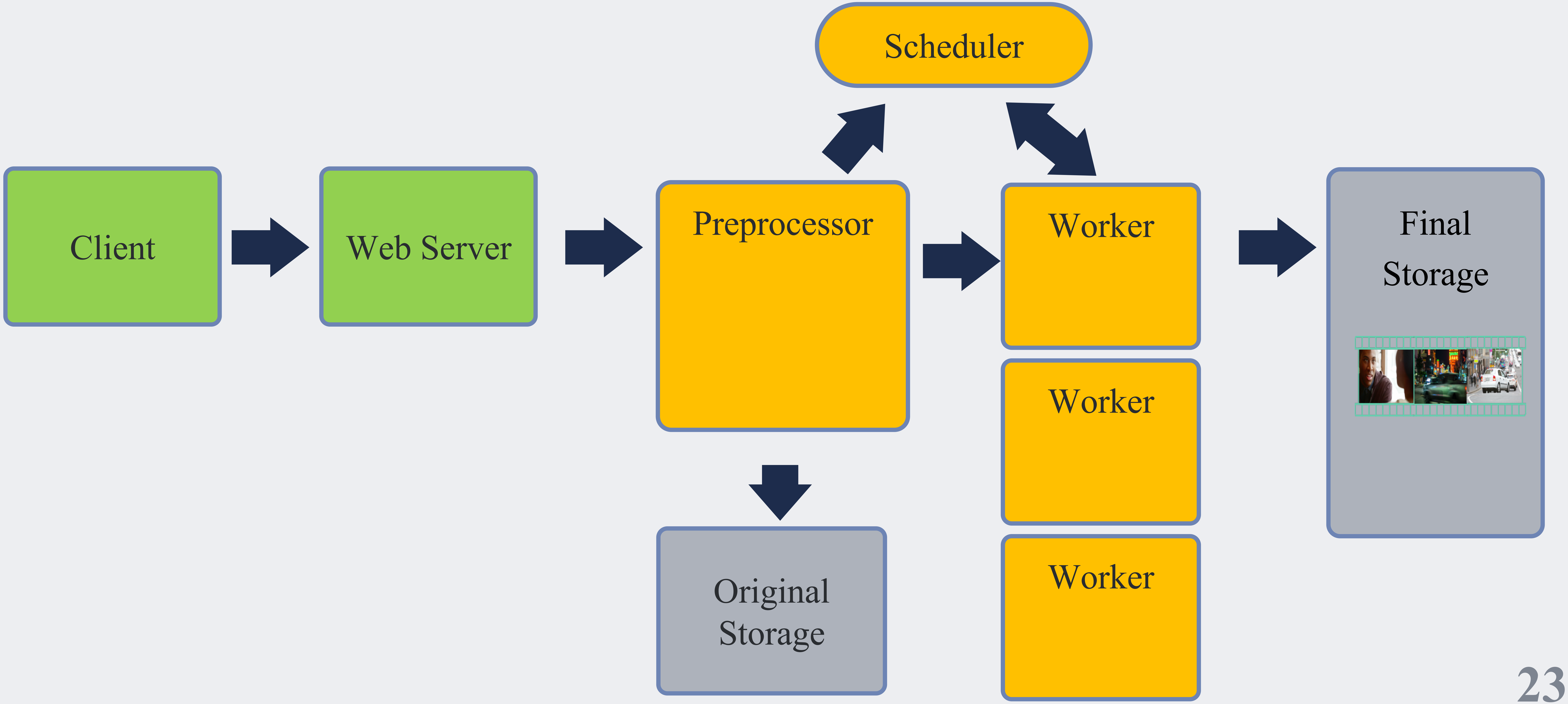
...upload in progress

Preprocessor

Original Storage

Final Storage

Parallel processing w/ many workers



Scheduler

Client

Web Server

Preprocessor

Worker

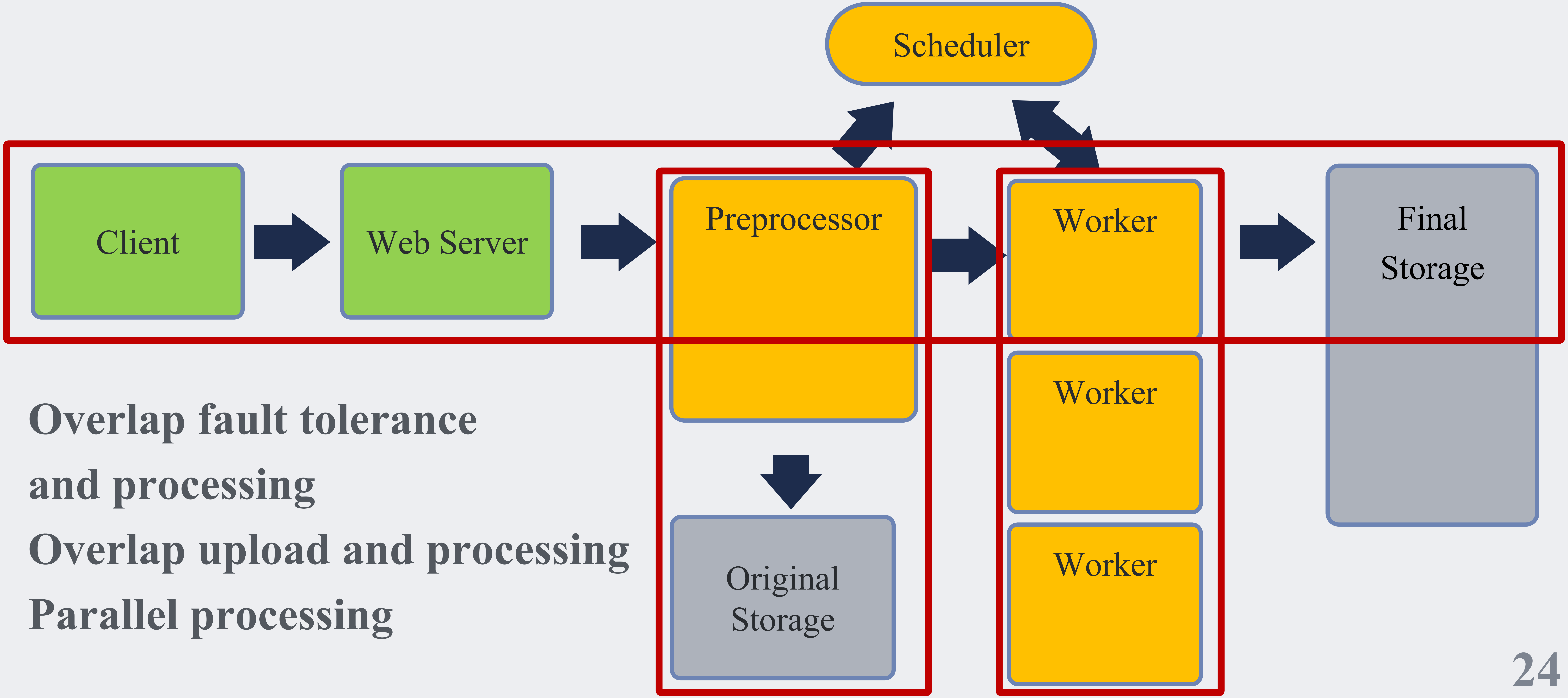
Worker

Worker

Final Storage

Original Storage

Three sources of parallelism

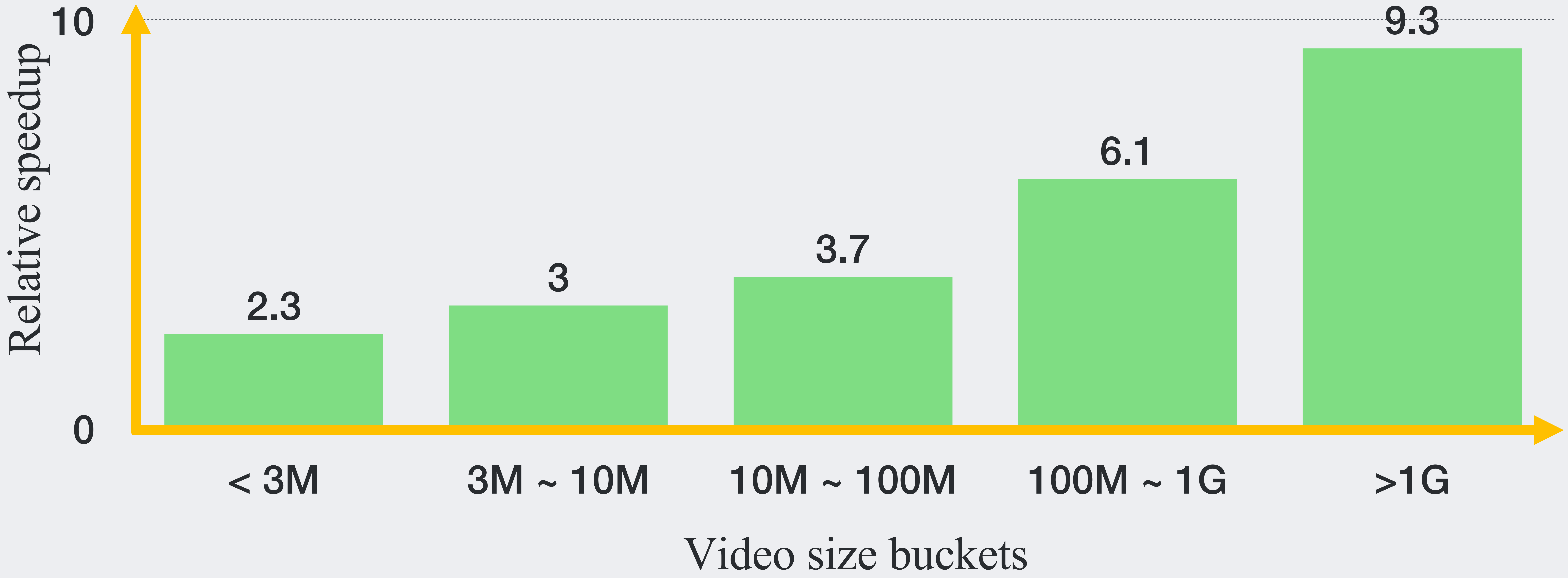


Overlap fault tolerance and processing

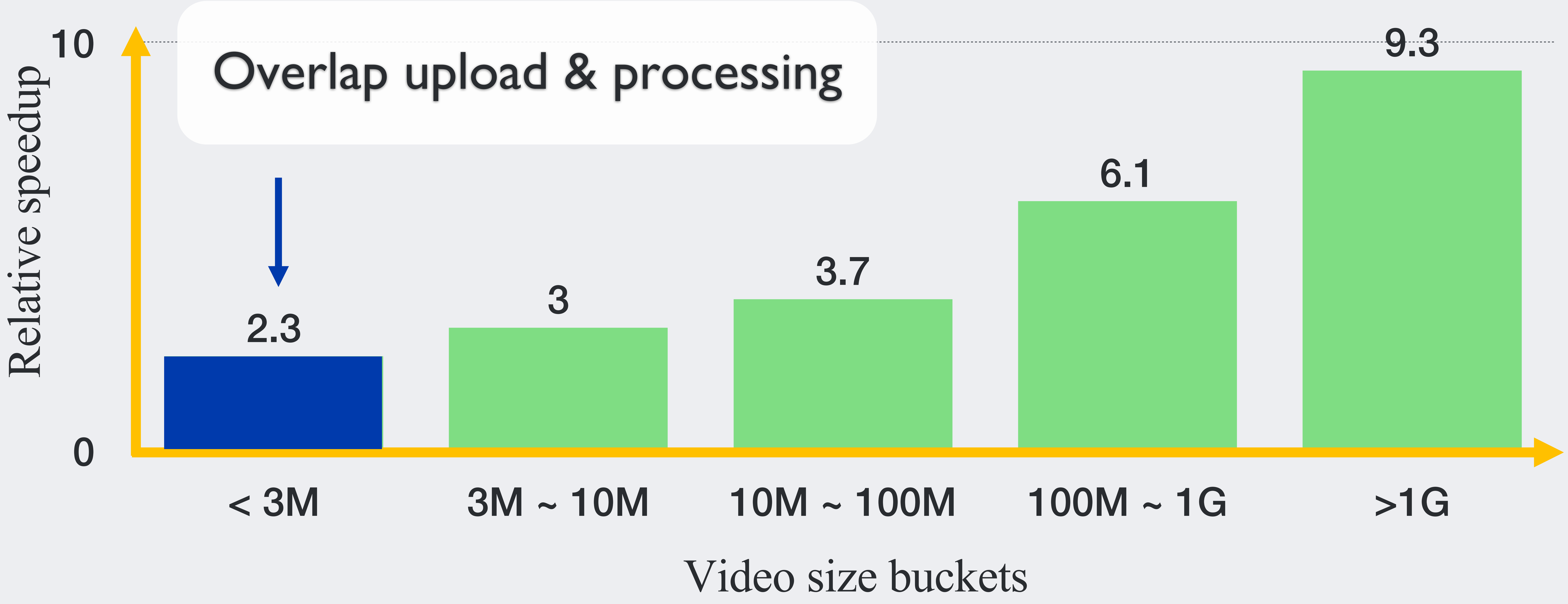
Overlap upload and processing

Parallel processing

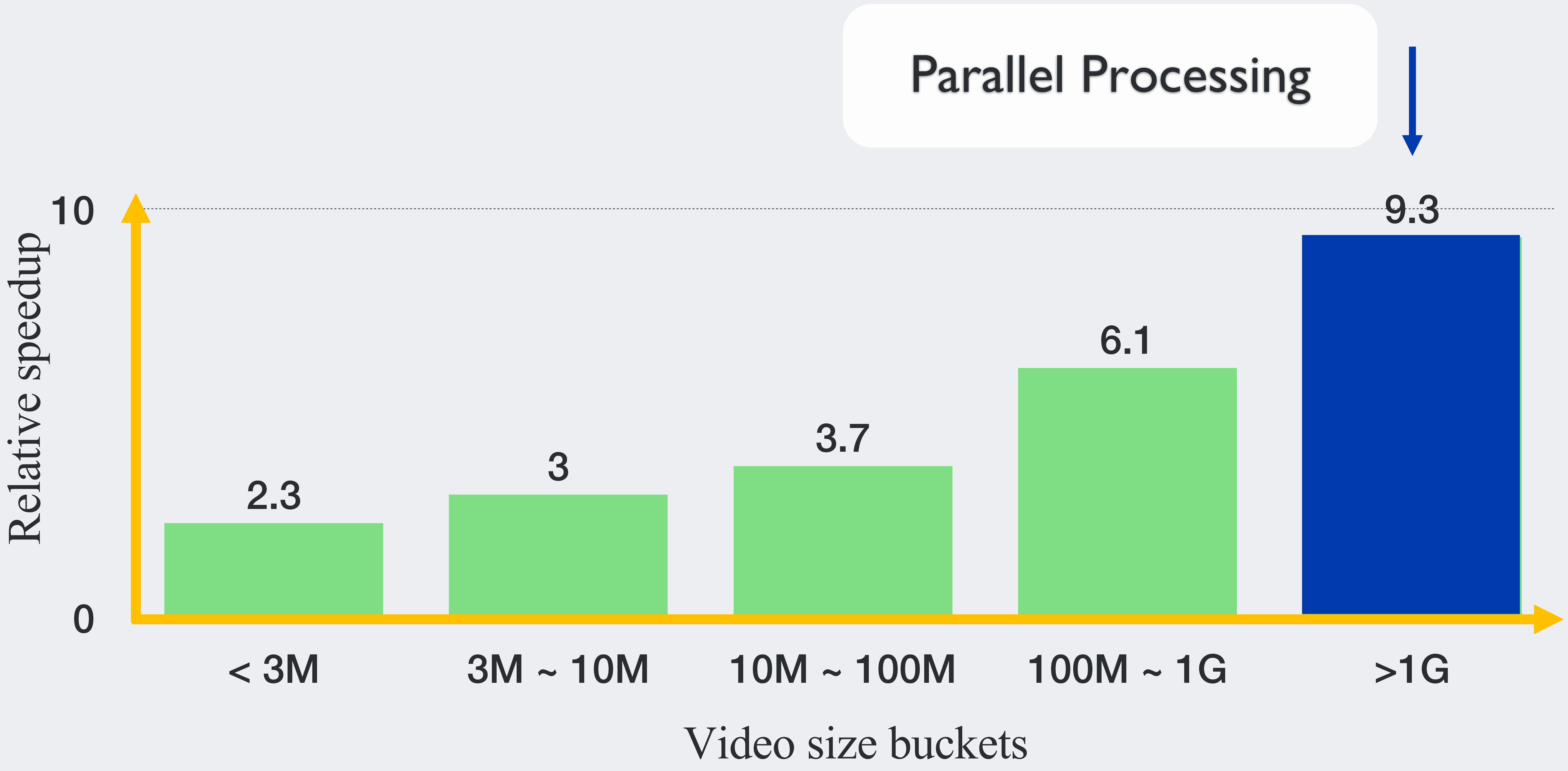
Results: 2.3x ~ 9.3x speedup



Results: 2.3x ~ 9.3x speedup



Results: 2.3x ~ 9.3x speedup



Challenges for video processing @ FB

Speedy

Users can share videos quickly

2.3x ~ 9.3x speedup

Flexible

Thousands of engineers can write pipelines for tens of apps

Robust

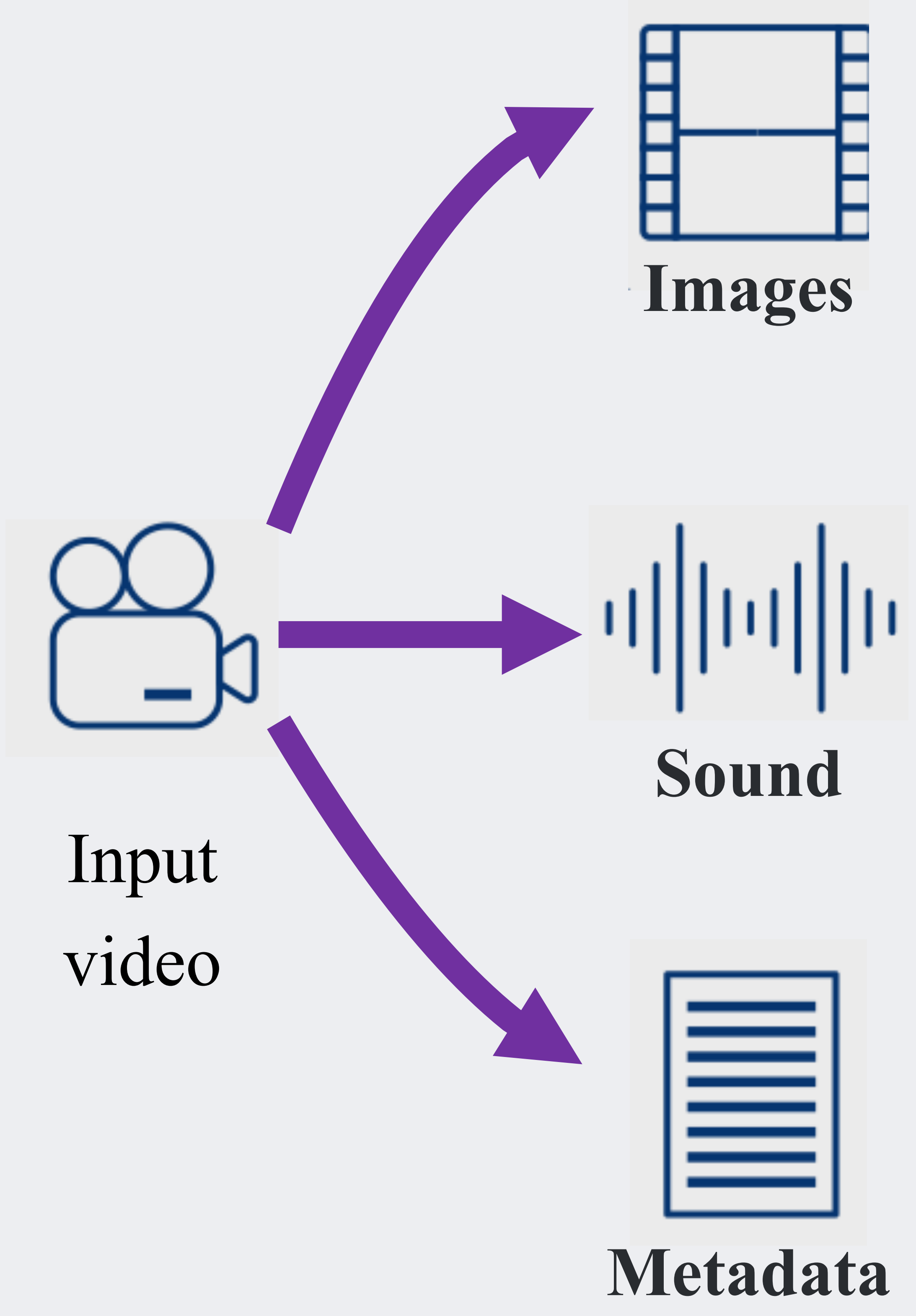
Handle faults and overload that is inevitable at scale

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

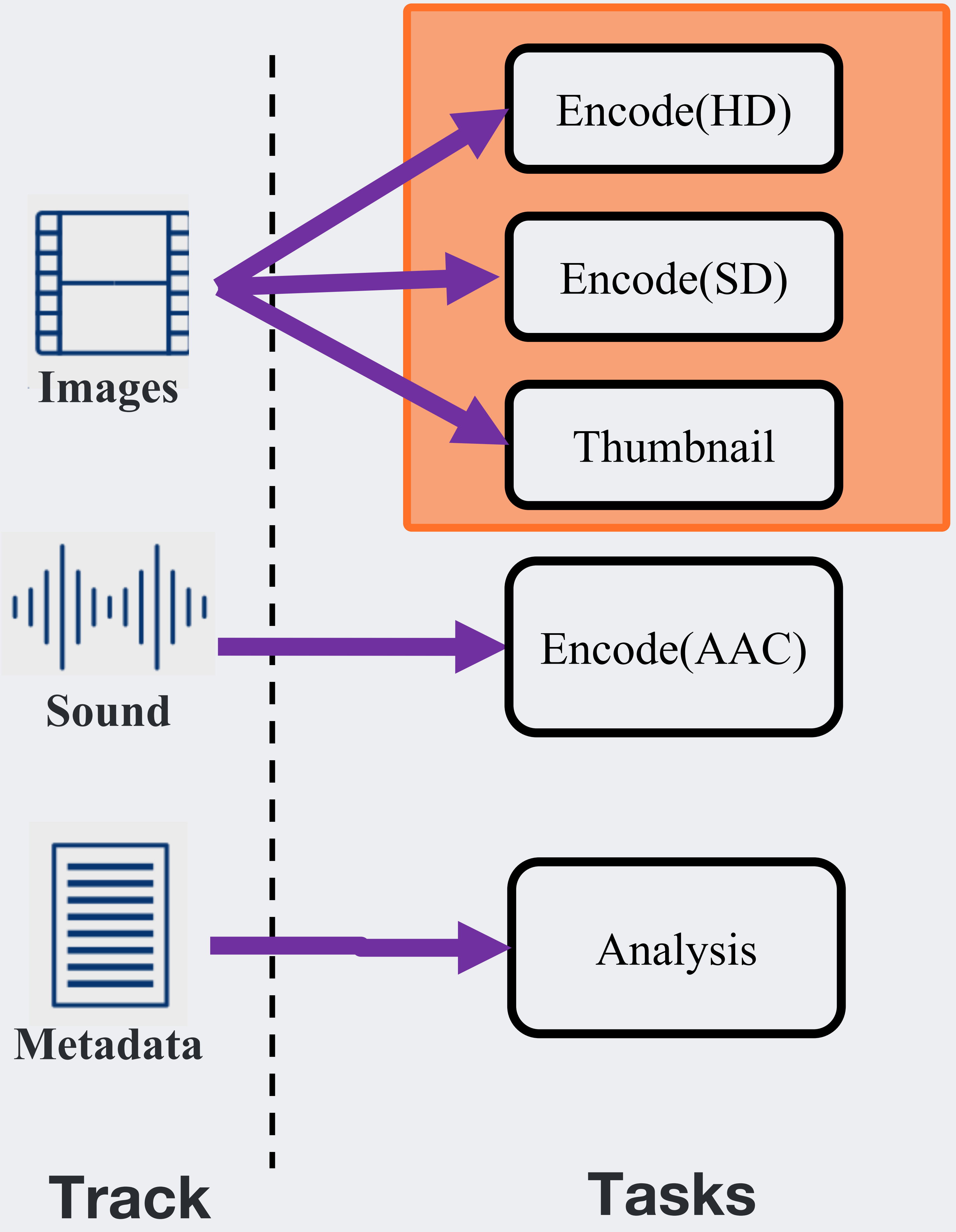
DAG on stream-of-tracks abstraction



Track

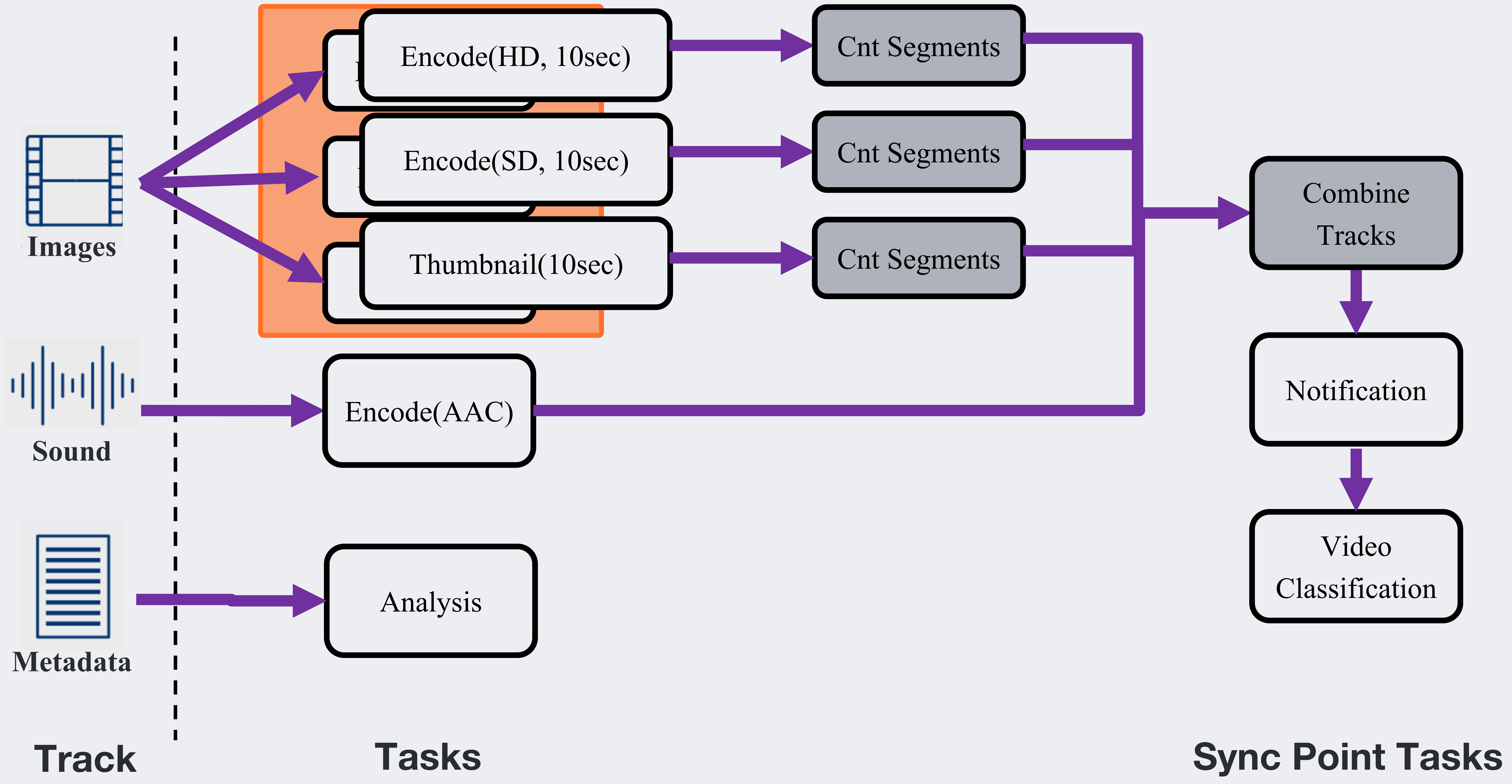
```
$pipeline = Pipeline.build()

$video_track=$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

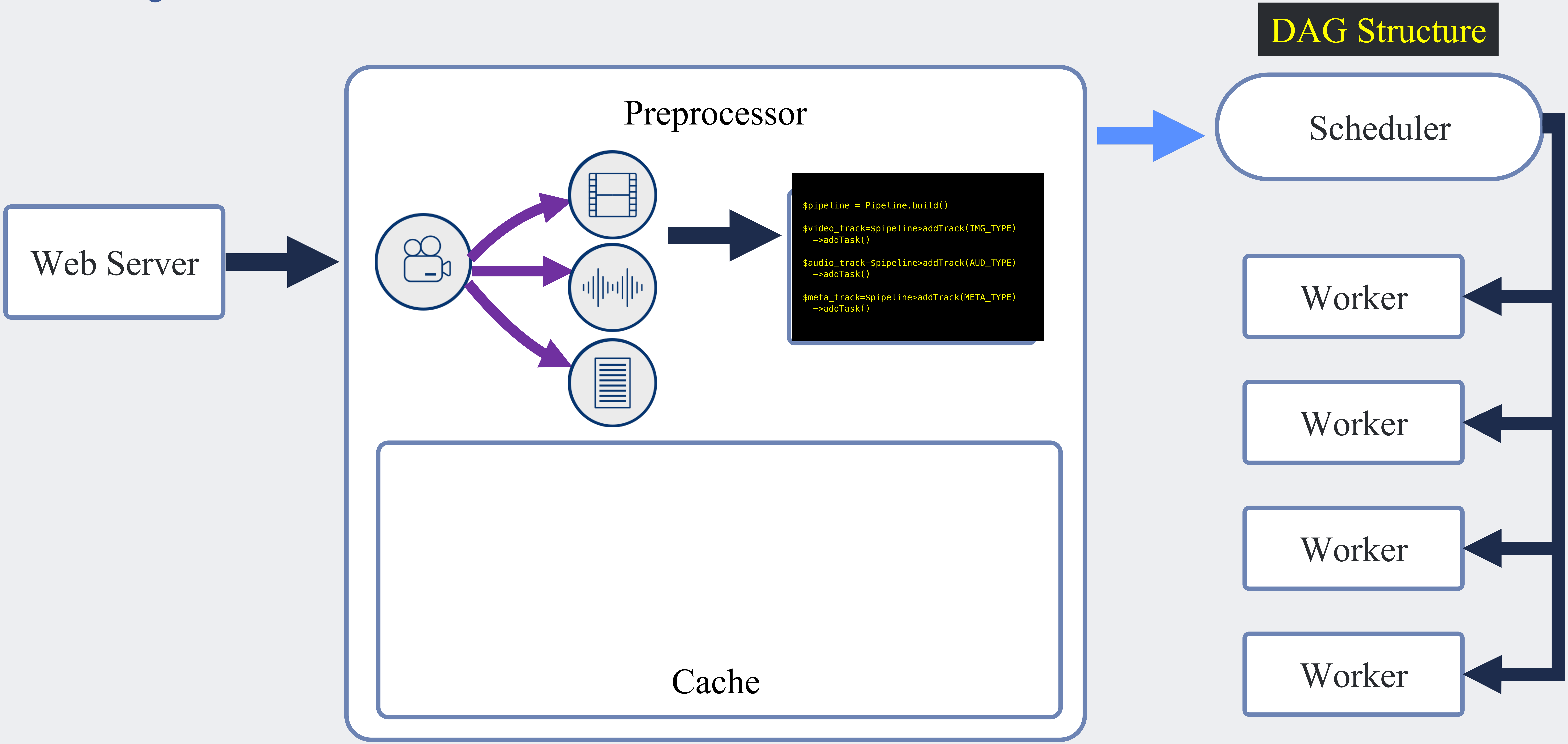


```
$pipeline = Pipeline.build()

$video_track=$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

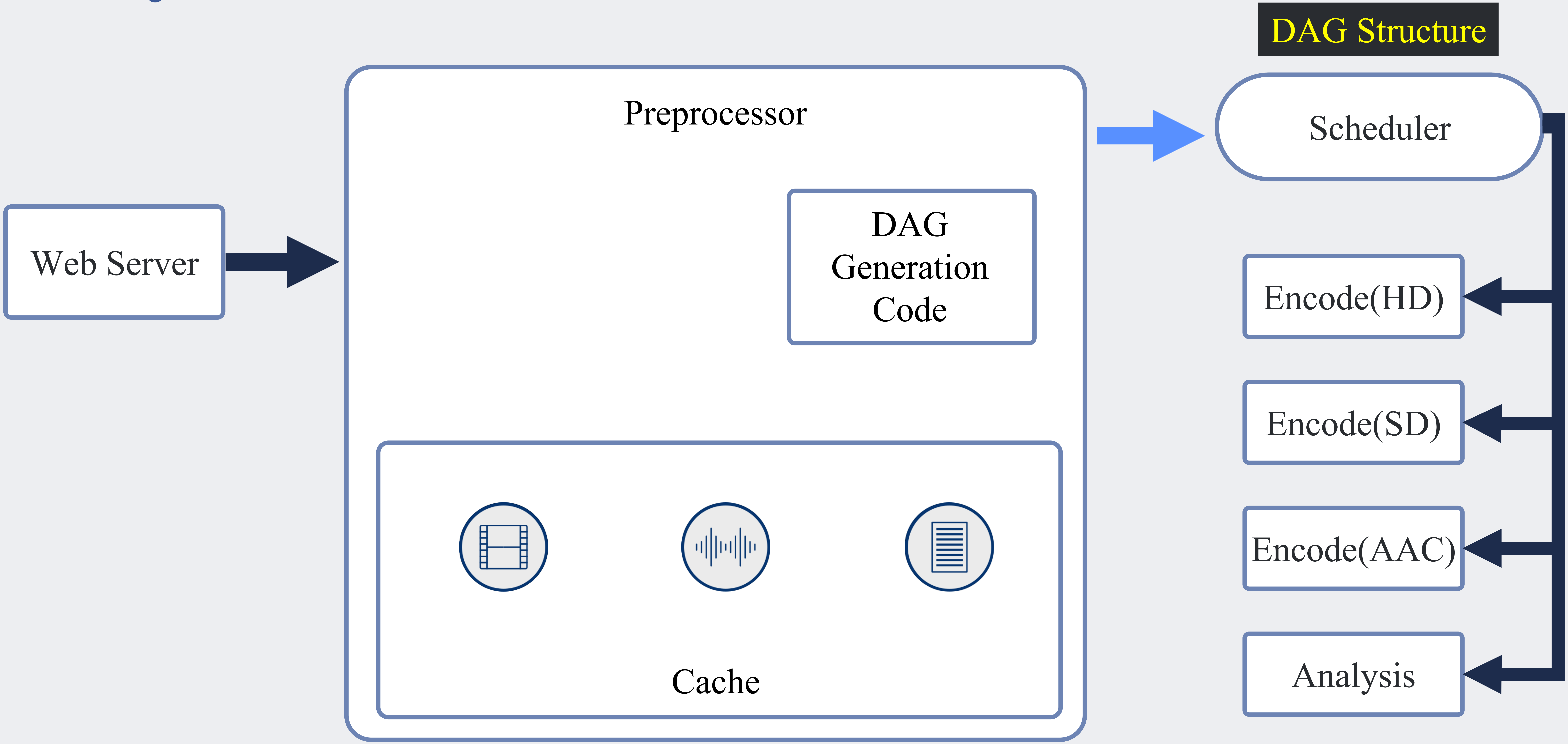


Dynamic DAG Generation

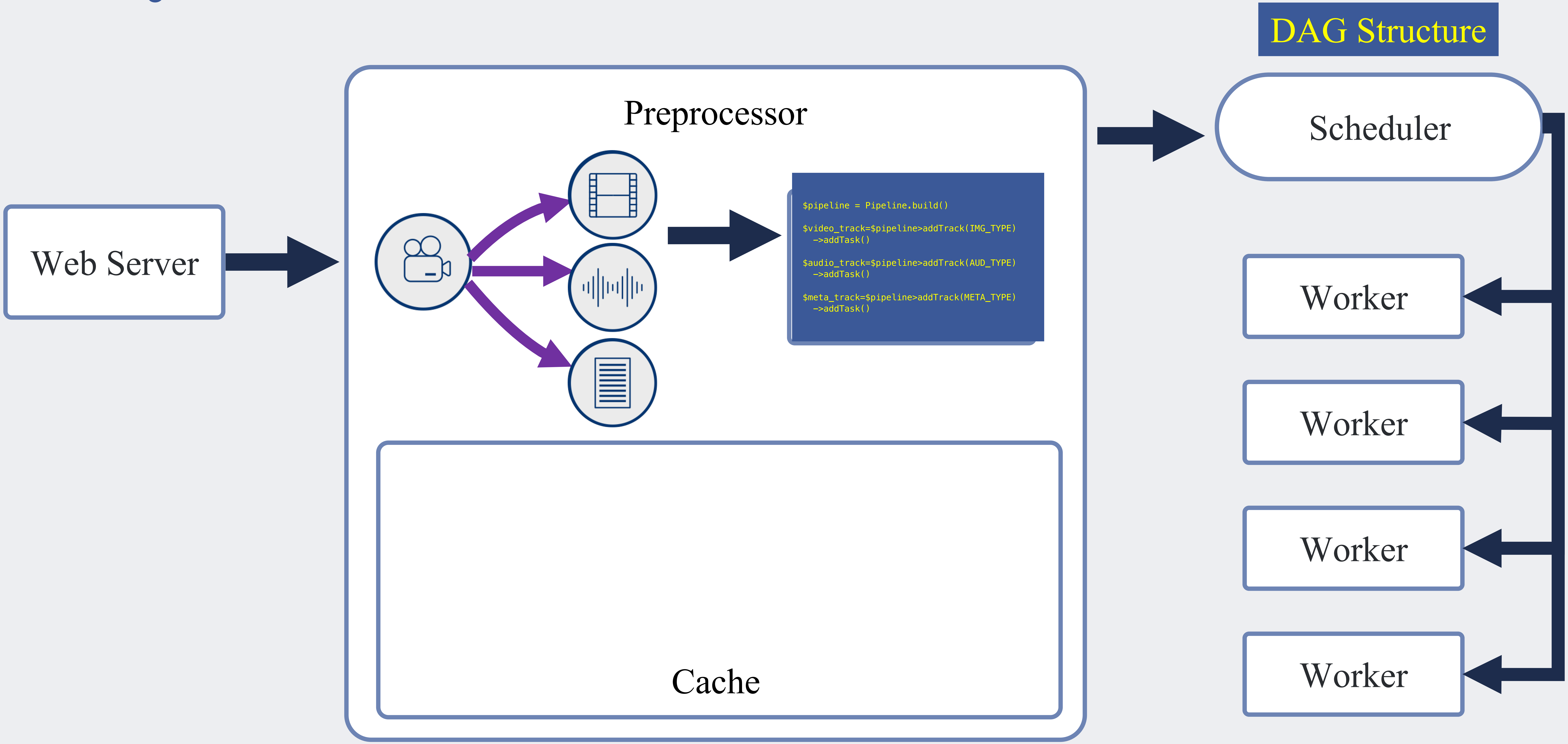


```
$pipeline = Pipeline.build()

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

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

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

Dynamic DAG Generation



Dynamic DAG Generation



```
$pipeline = Pipeline.build()

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

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

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

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 pipelines for tens of apps

One system for 15+ applications

Robust

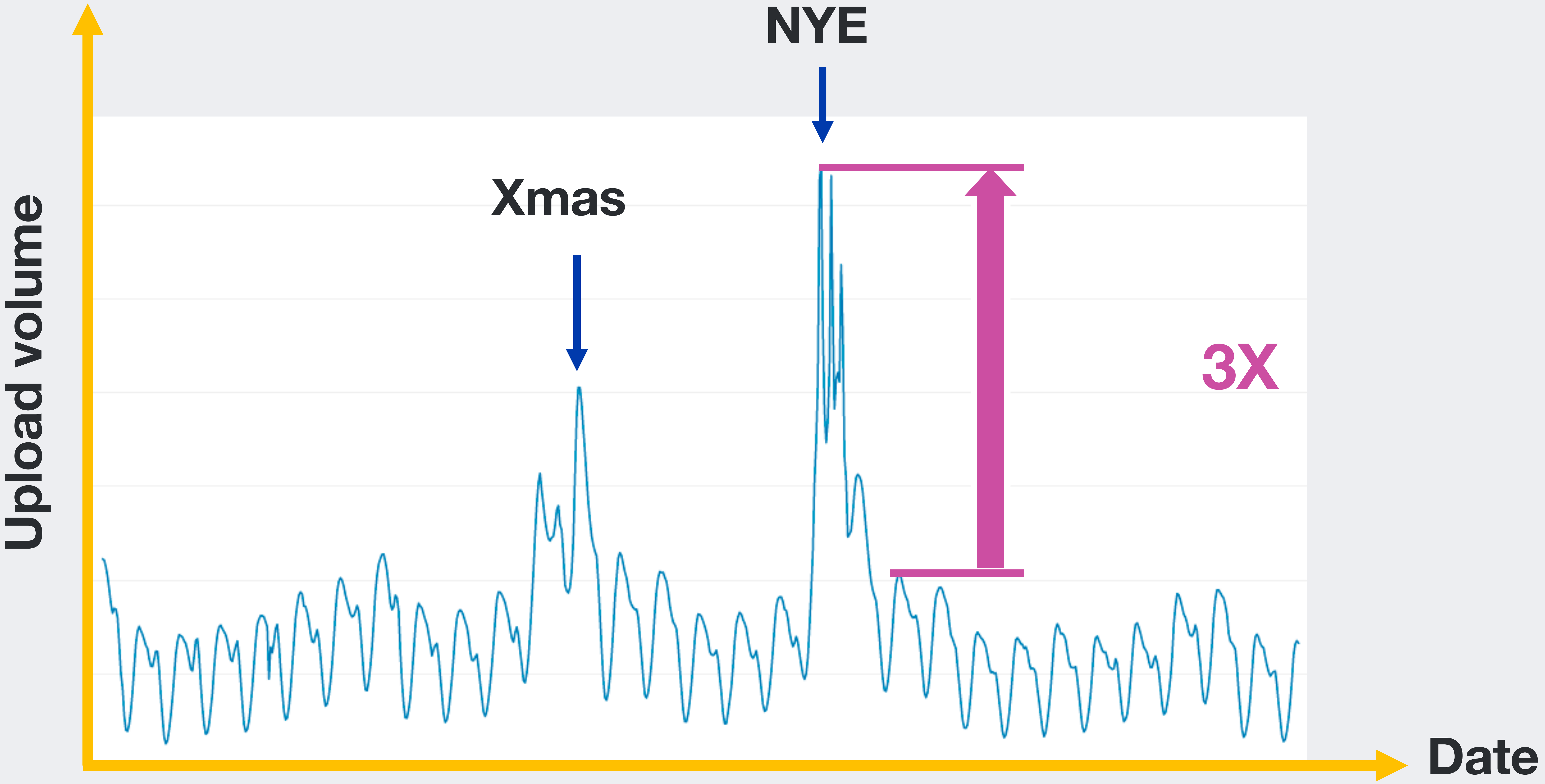
Handle faults and overload that is inevitable at scale

Robust: tolerate overload

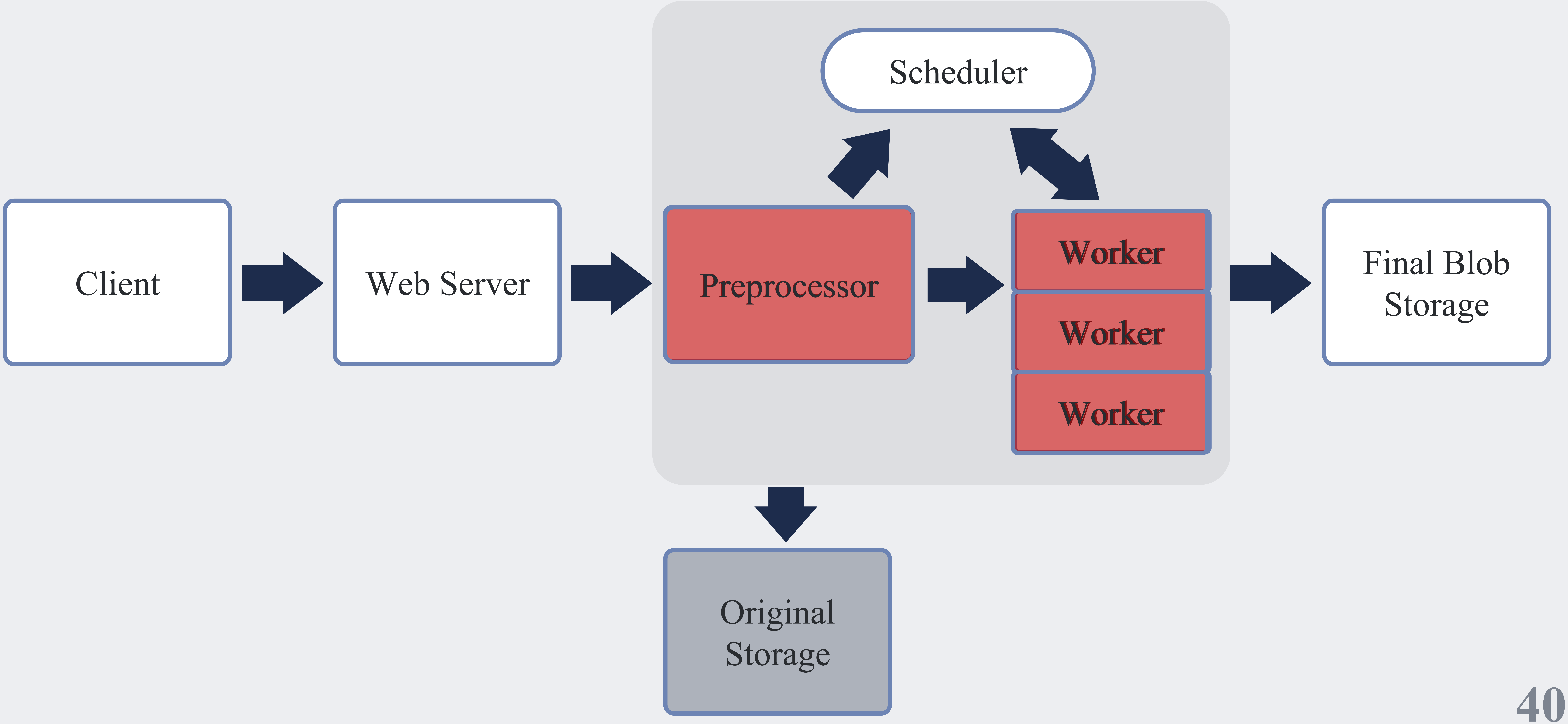
Handle faults and overload that is inevitable at scale

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

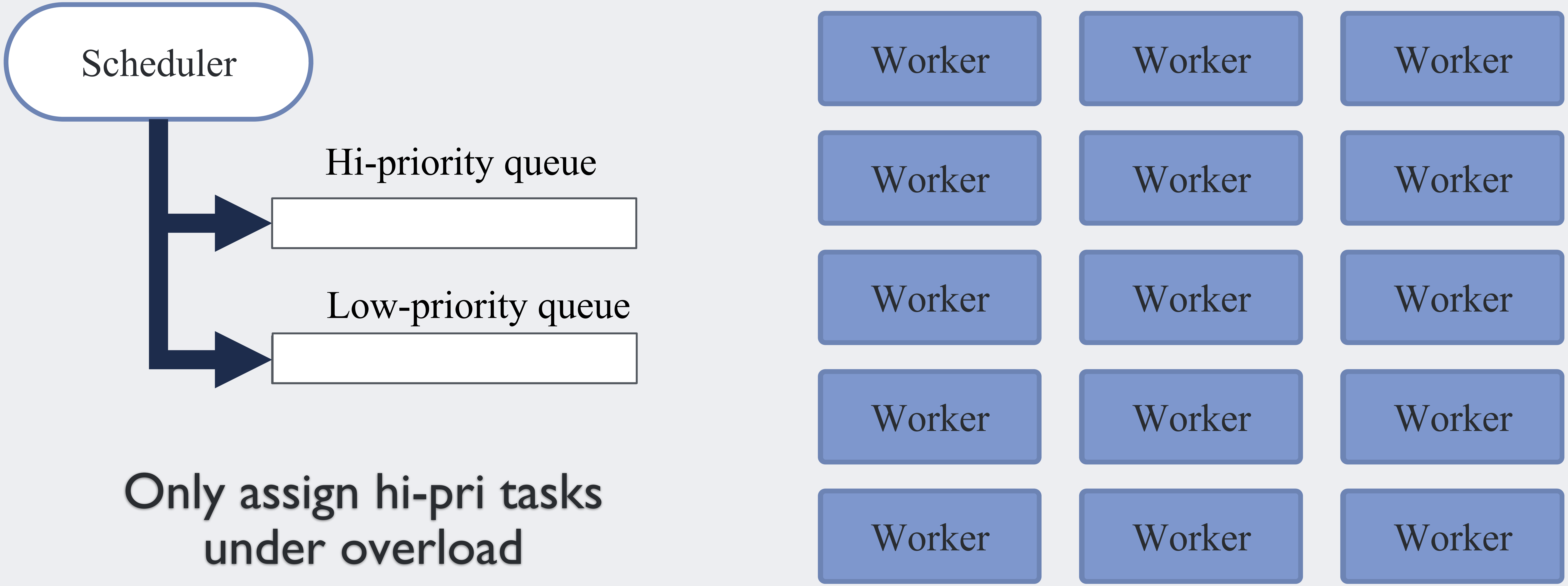
3X peak load during New Year Eve



Prepare for overload

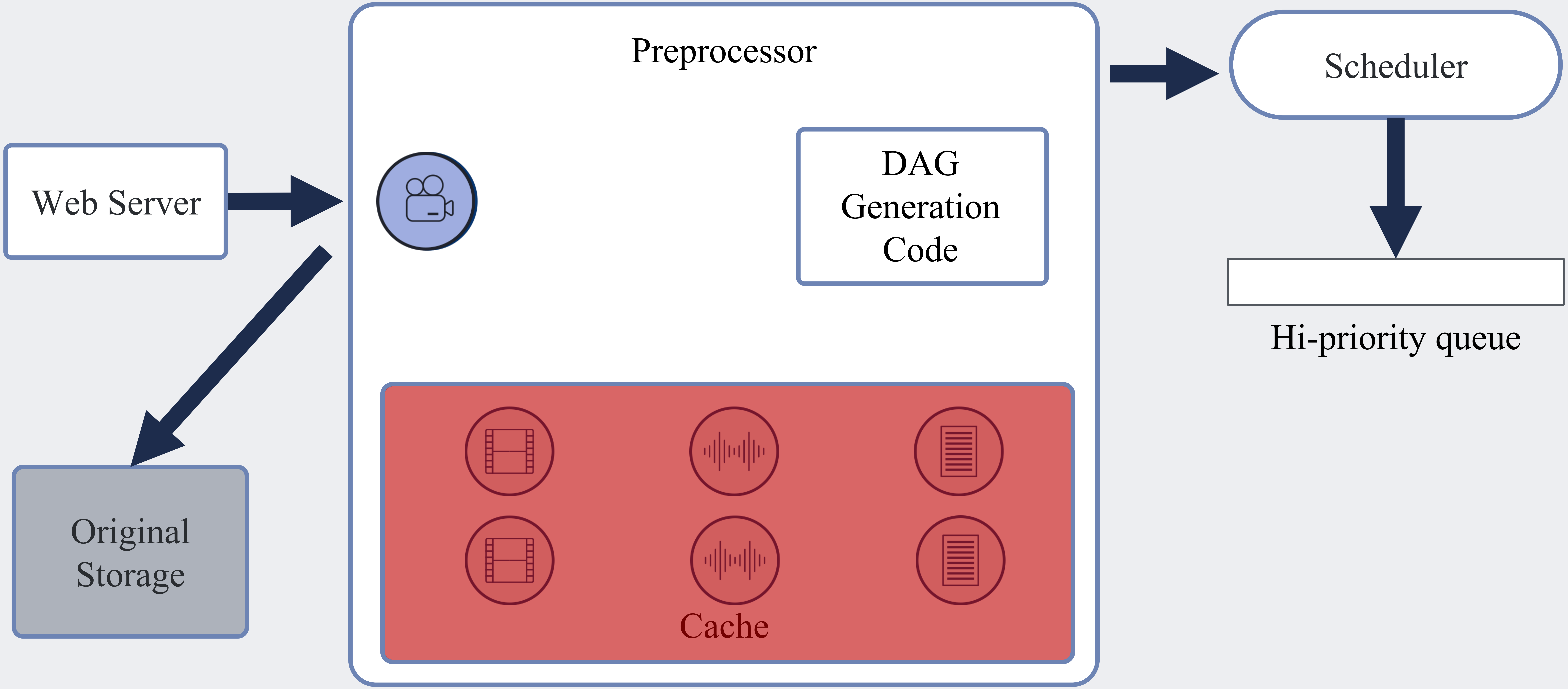


Use priority for worker overload

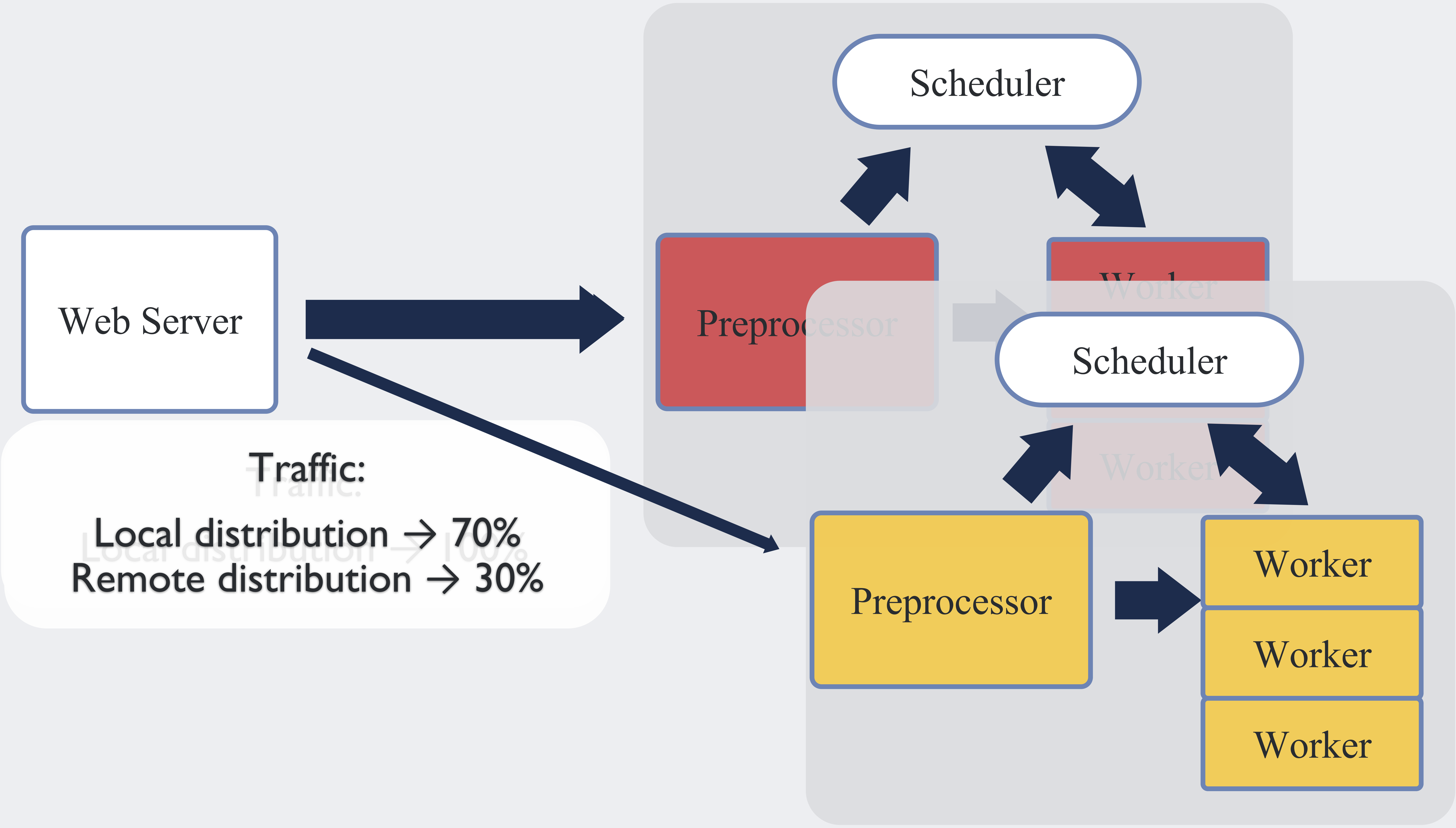


Only assign hi-pri tasks under overload

Defer full video processing



Regional redirection



Traffic:

Local distribution → 70%
Remote distribution → 30%

Challenges for video processing @ FB

Speedy

2.3x ~ 9.3x speedup

Flexible

One system for 15+ applications

Robust

Handle faults and overload that is inevitable at scale

Tolerate 3x traffic spike

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

Related work

- Batch processing
  - MapReduce, Dryad, Piccolo, CIEL, Spark, Naiad

SVE overlaps data ingestion and processing

- Stream processing
  - STREAM, Aurora, Spark streaming, JetStream, StreamScope

SVE offers dynamic DAG generation per input

- Video processing at scale
  - Netflix, ExCamera, Chess-VPS, VideoStorm

SVE support many production apps

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