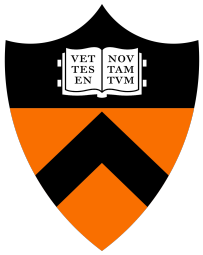
Making Systems Faster: Distributed Video Processing



COS 418: Distributed Systems
Lecture 21

Wyatt Lloyd

[Grey slides from Qi Huang's SOSP 2017 Talk]

Distributed Video Processing Outline

- Motivation for video processing
  - (How streaming video works)
- Legacy design
- SVE design
- Why SVE is faster than legacy

SVE: Distributed Video Processing at Facebook Scale

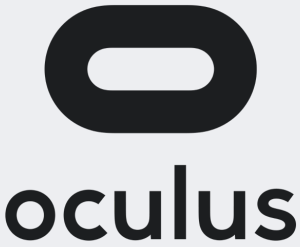
Qi Huang

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

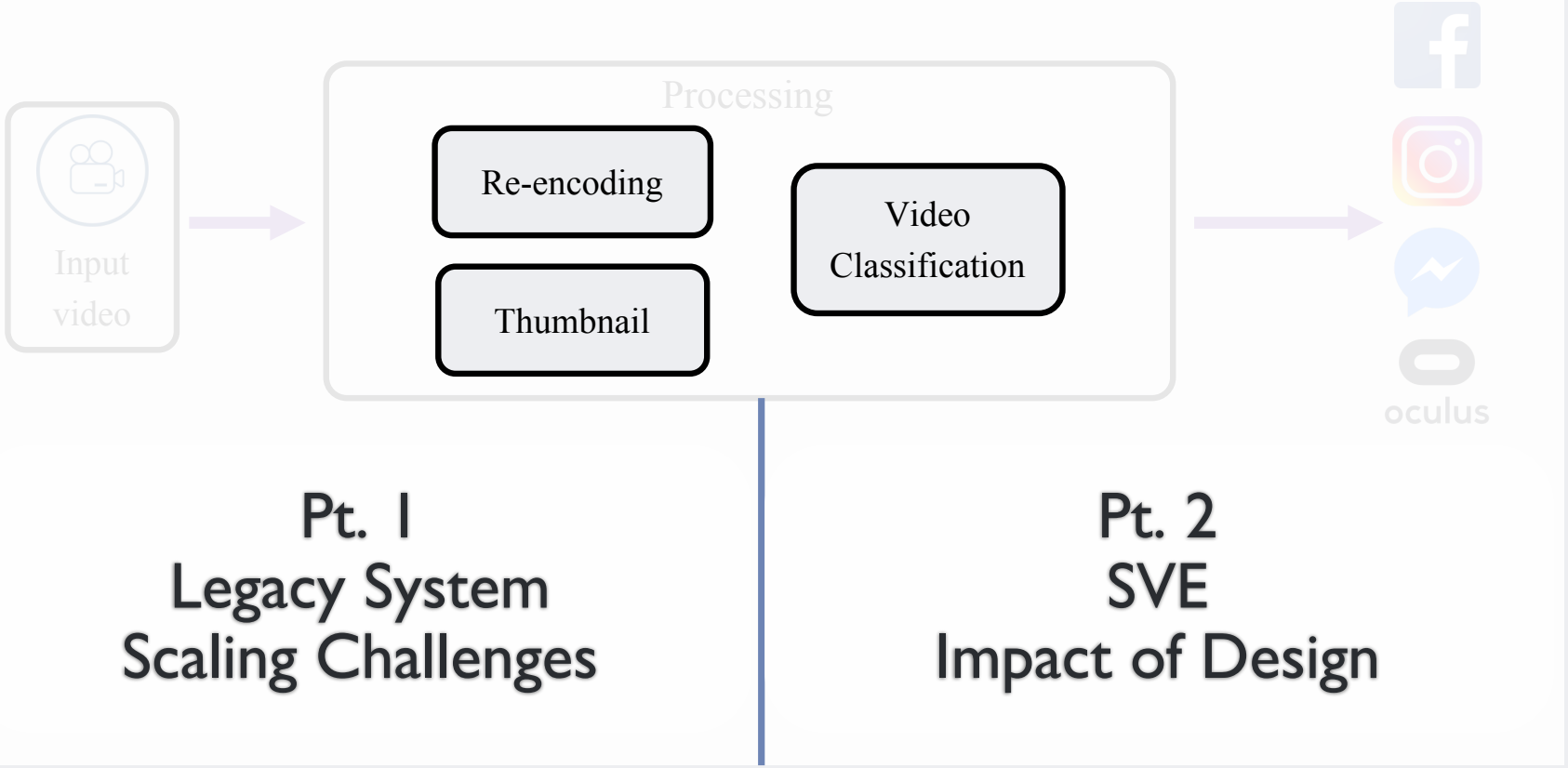
Video is growing across Facebook



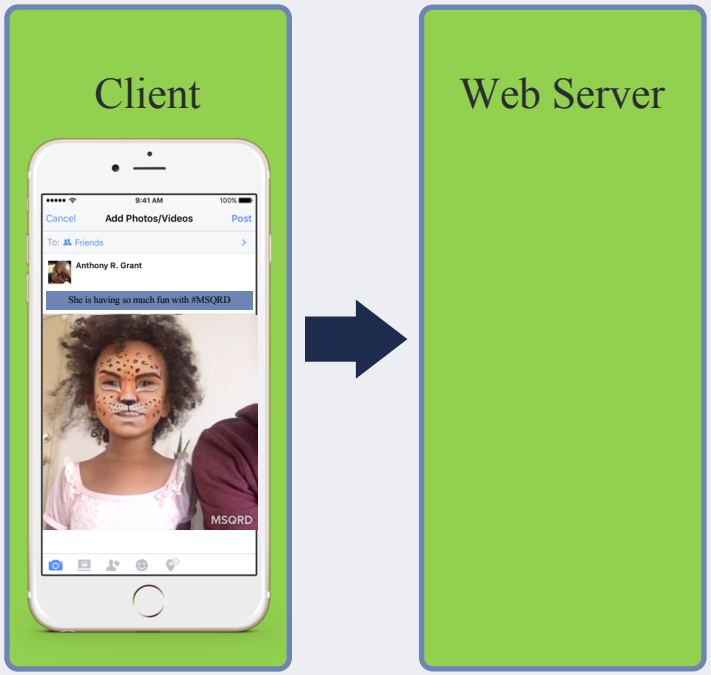



- FB: **500M** users watch **100M hours** video daily (Mar. 16)
- Instagram: **250M** daily active users for stories (Jun. 17)
- All: **many tens of millions** of daily uploads, **3X** NYE spike

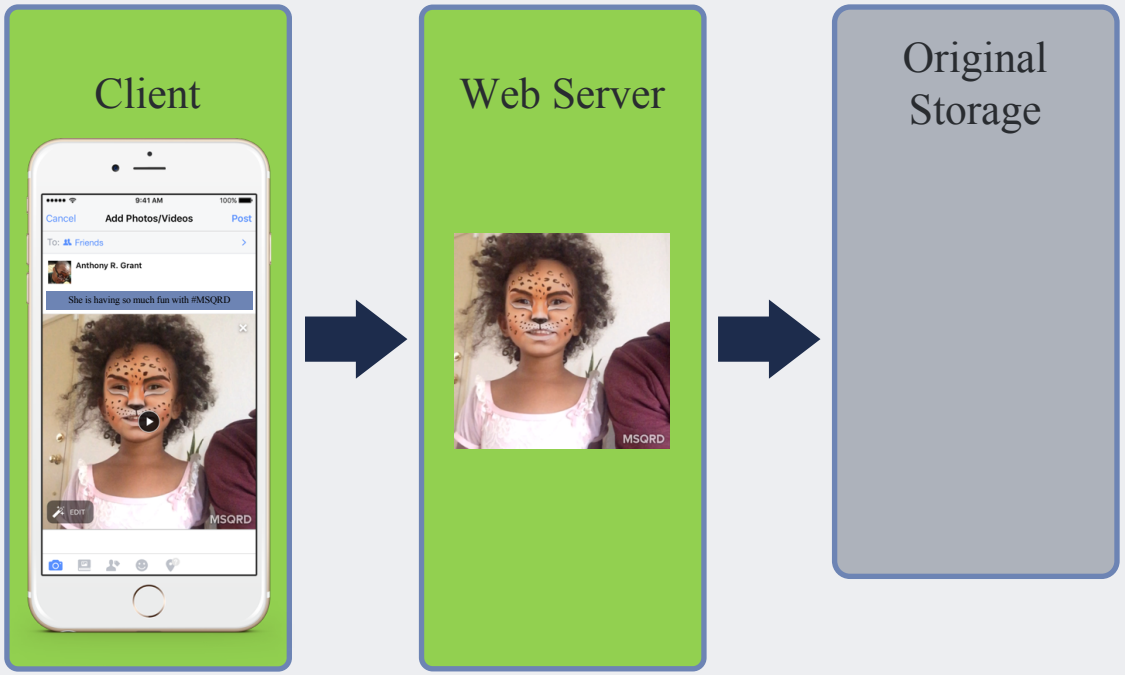
Processing is diverse and demanding



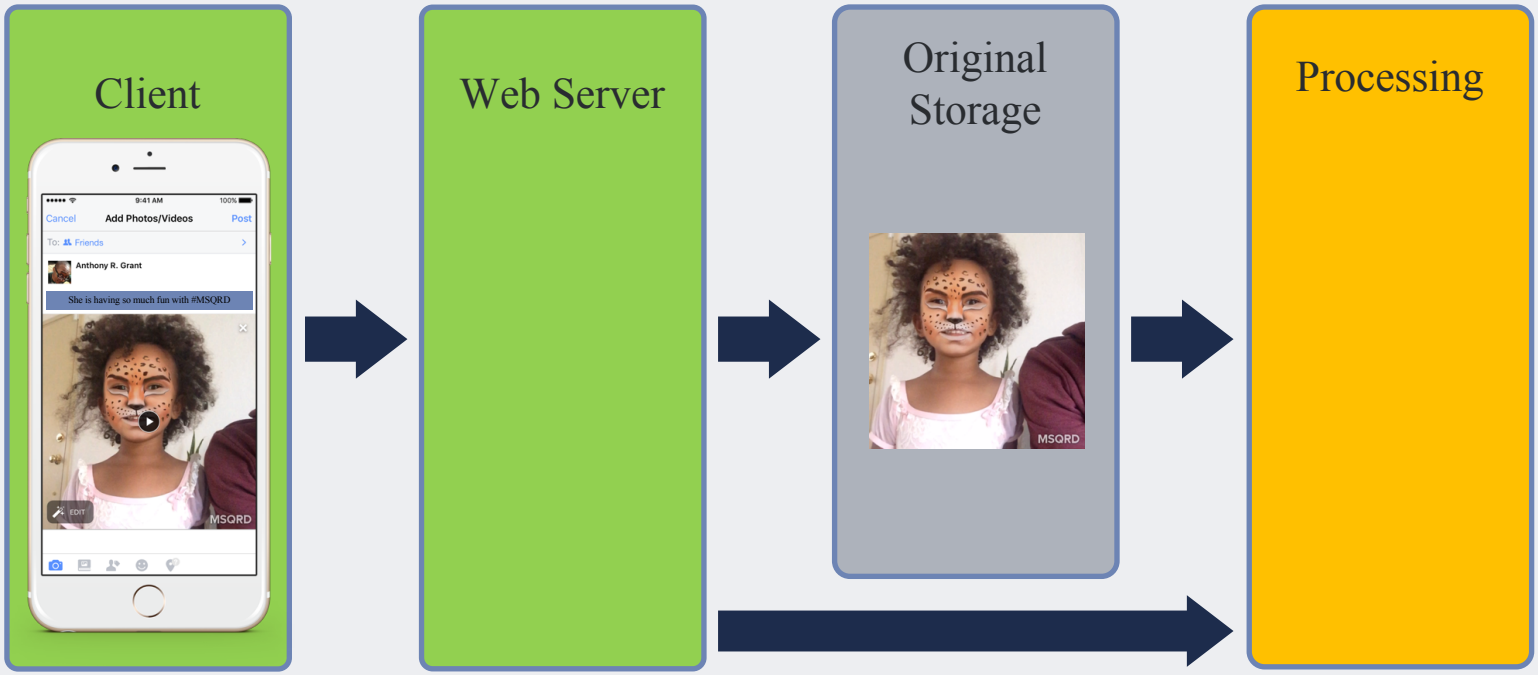
Legacy: upload video file to web server



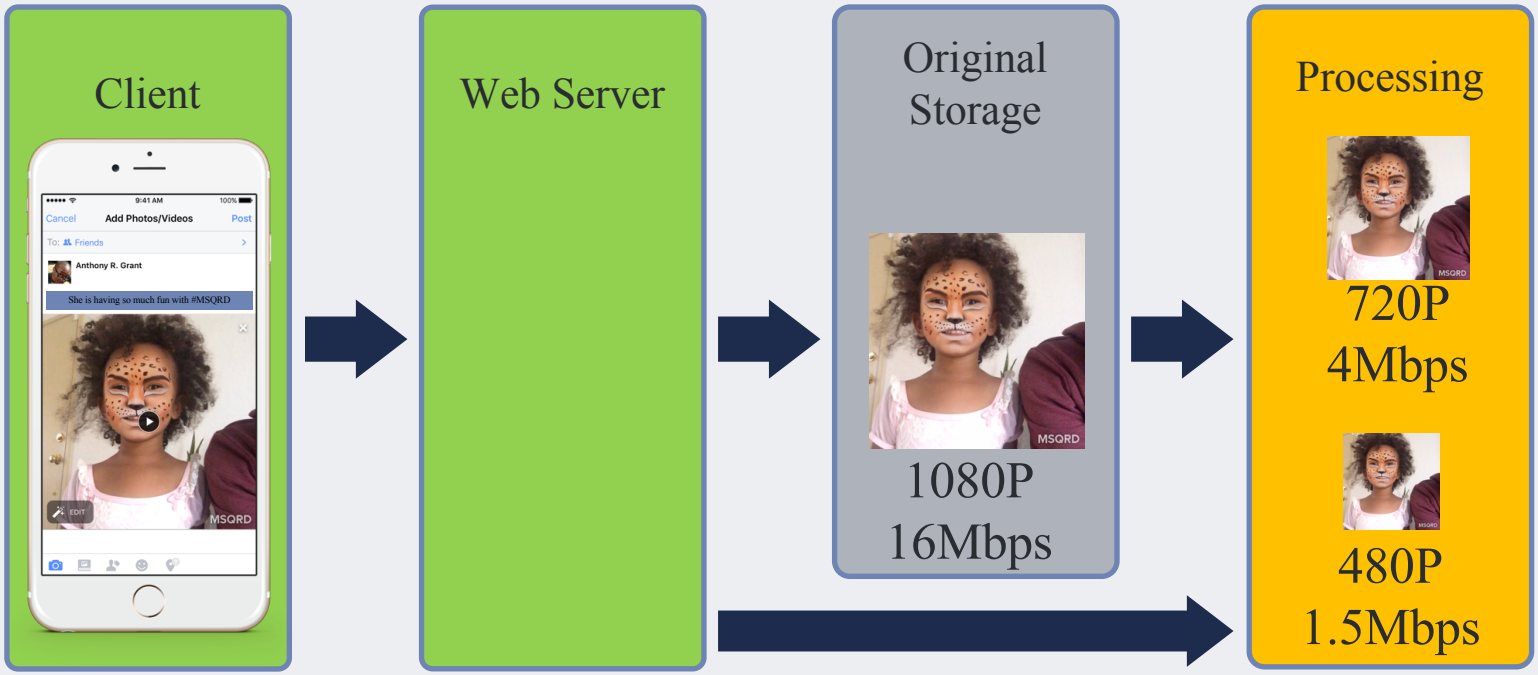
Legacy: preserve original for reliability



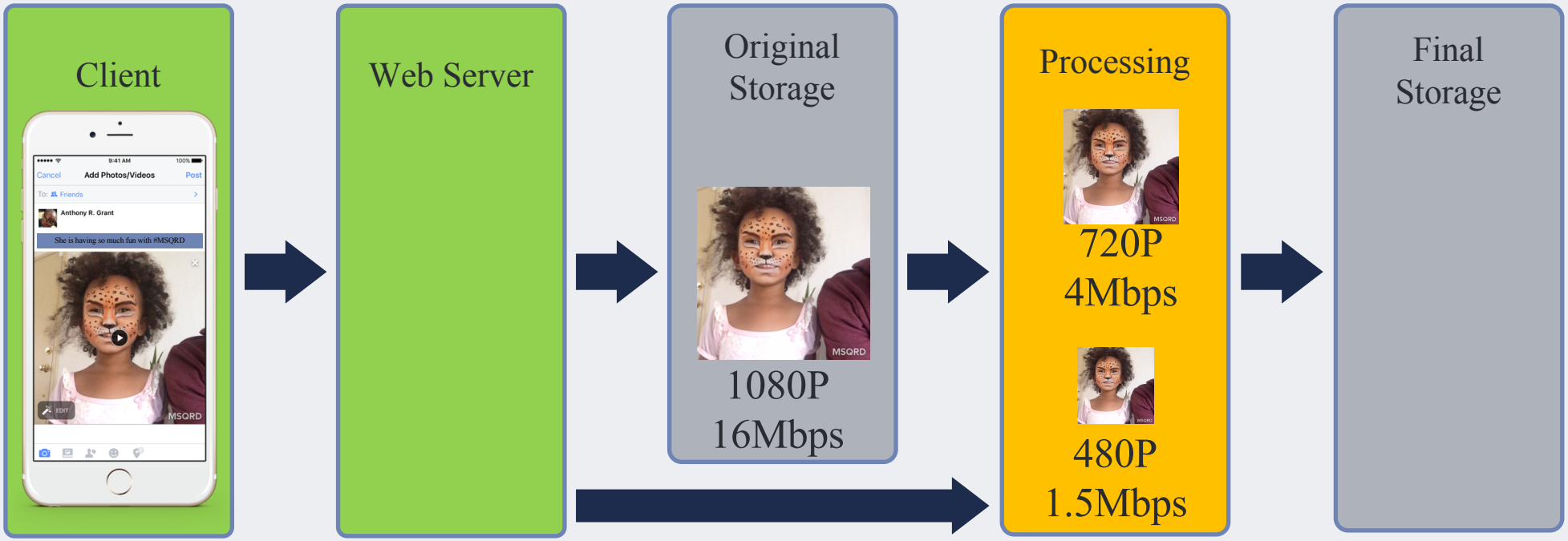
Legacy: process after upload completes



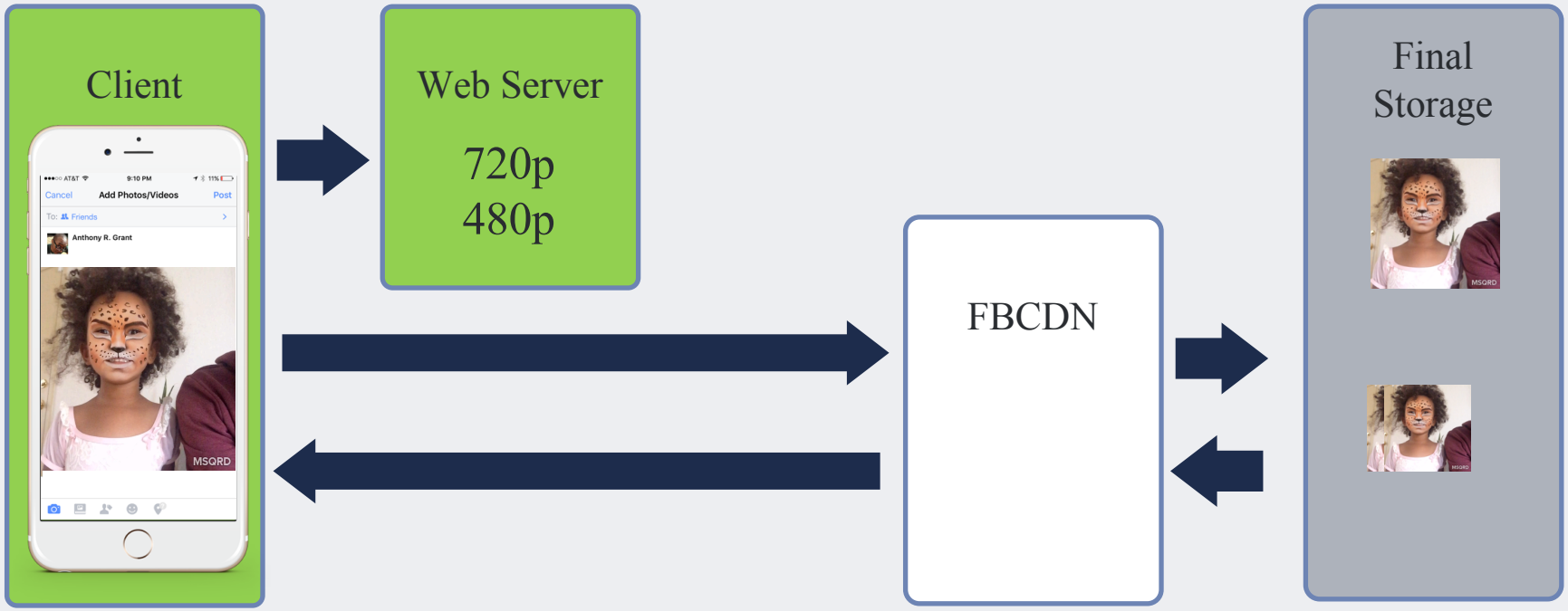
Legacy: encode w/ varying bitrates



Legacy: store encodings before sharing



Sharing with adaptive streaming

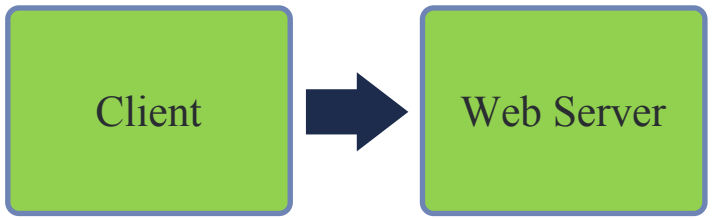


Focus: pre-sharing pipeline

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

How Long Does This Take? (Latency)

How Long Does This Take? (Latency)



1 MB Video ≈ 1 secs
8 Mbps link

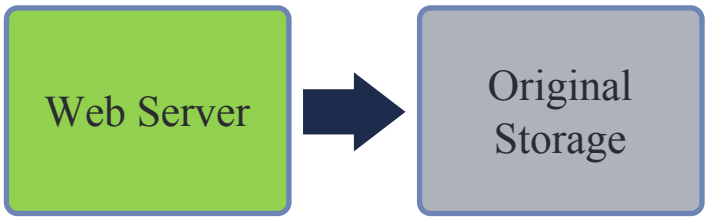
16 MB Video ≈ 16 secs
1 Mbps link

SVE paper stats:

Video Size	
≤1MB	10% of uploads over 10 seconds
3-10MB	50% of uploads over 10 seconds
300MB -1GB	50% of uploads over 9 minutes

How Long Does This Take? (Latency)

How Long Does This Take? (Latency)

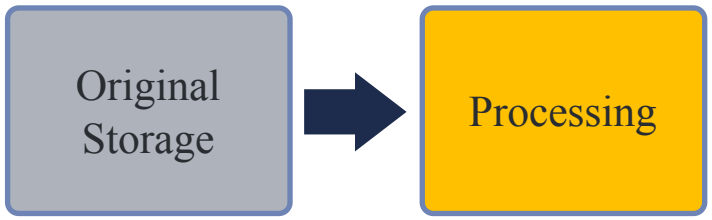


(pipelined with uploading)

SVE paper stats:

median	200 ms
90%	650 ms
99%	900 ms

How Long Does This Take? (Latency)



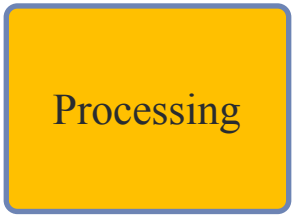
SVE paper stats:

10% of all video take ≥ 1.3 s

Proportional to video size:

Most videos over 100 MB take over 6 seconds

How Long Does This Take? (Latency)

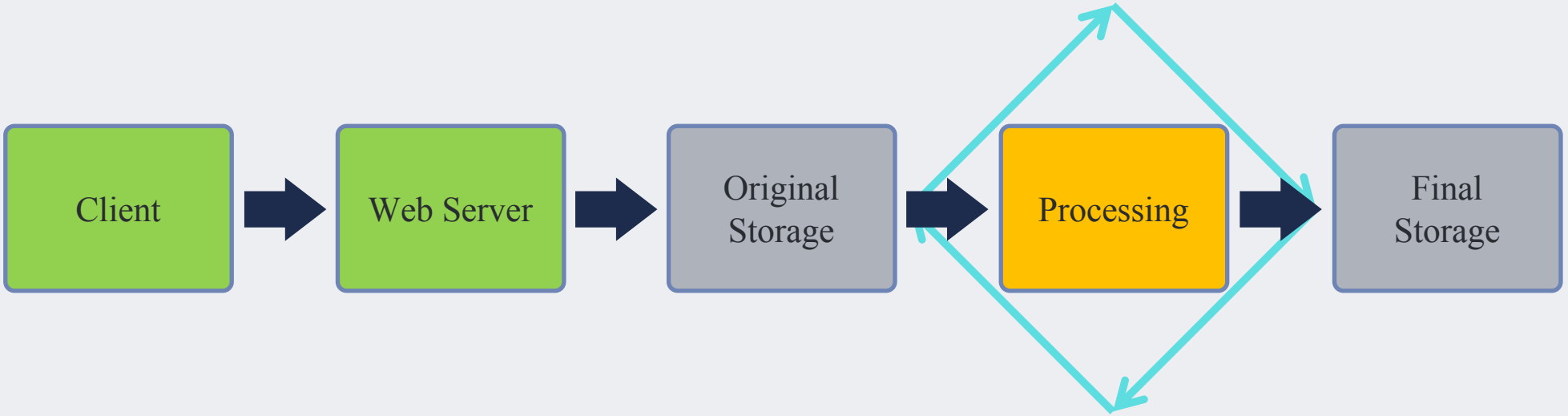


SVE paper stats:

Video Size

1-3MB	20% take over 10 seconds
100-300MB	50% take over 1 minute
>1GB	23% take over 10 minutes

Serial pipeline leads to slow processing



Let’s Make This Faster!

Chat me how!

Monolithic script slows development

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

"Change color coding at time"

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

Processing

Final Blob Storage

"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

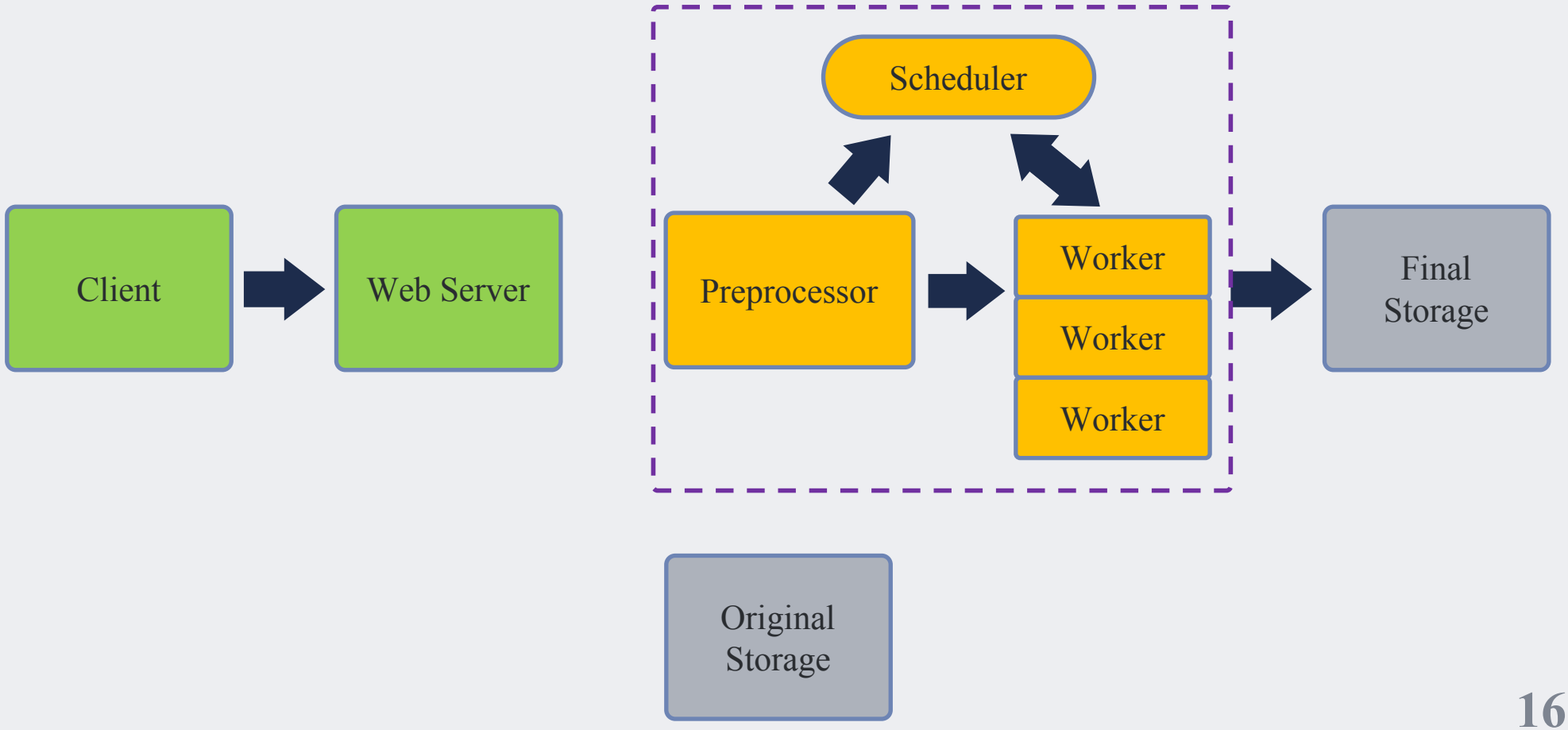
Speedy: harness parallelism

Users can share videos quickly

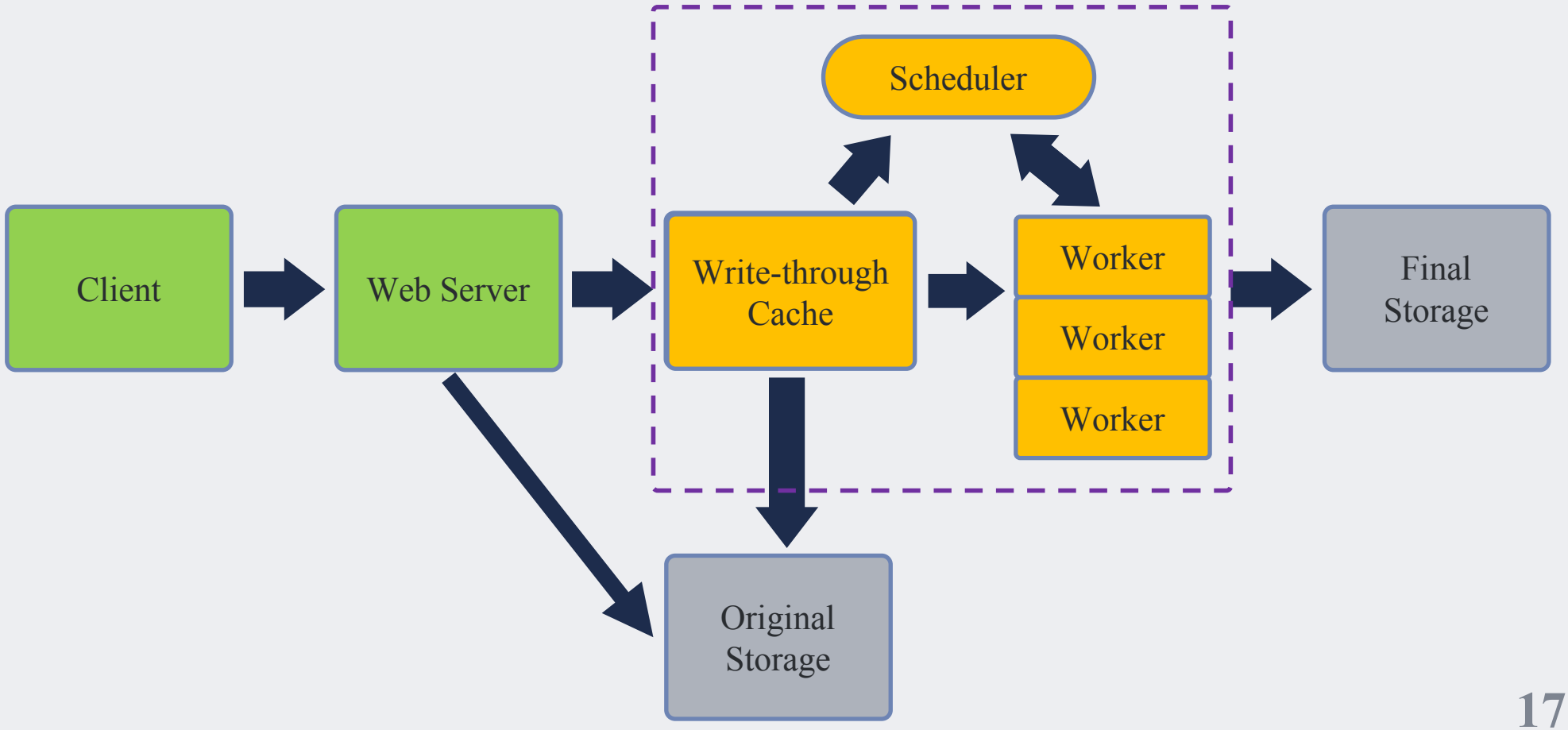
- Overlap fault tolerance and processing
- Overlap upload and processing
- Parallel processing

Architectural changes for parallelism

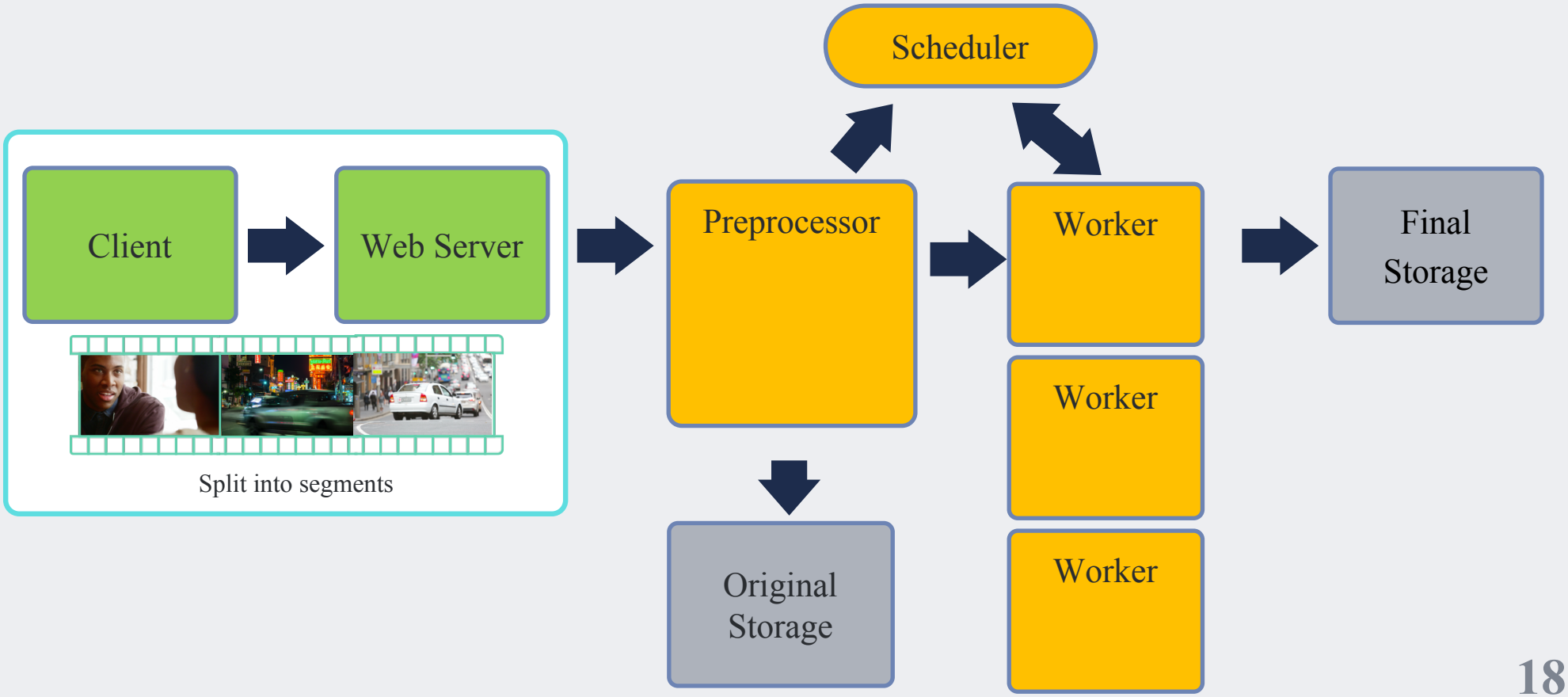
Architectural changes for parallelism



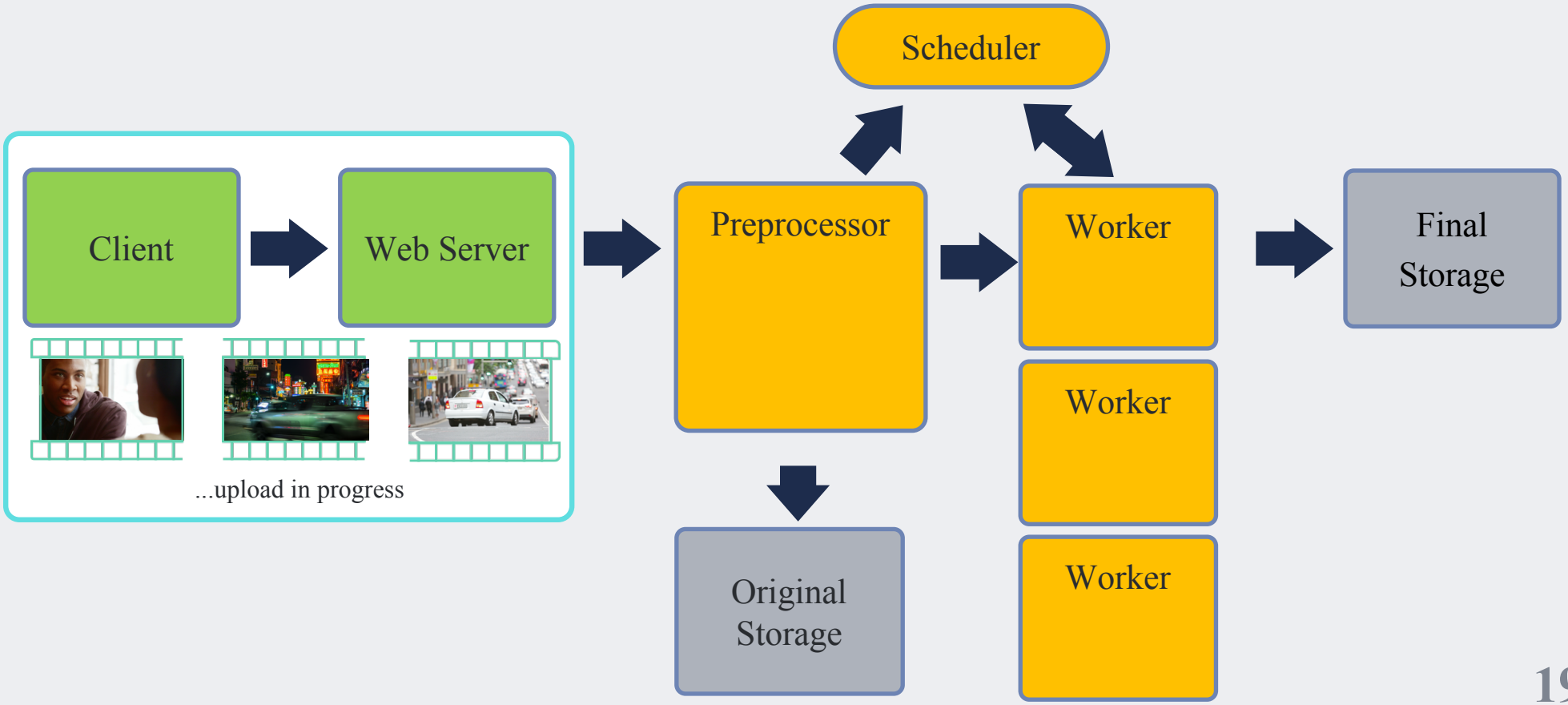
Overlap fault tolerance and processing



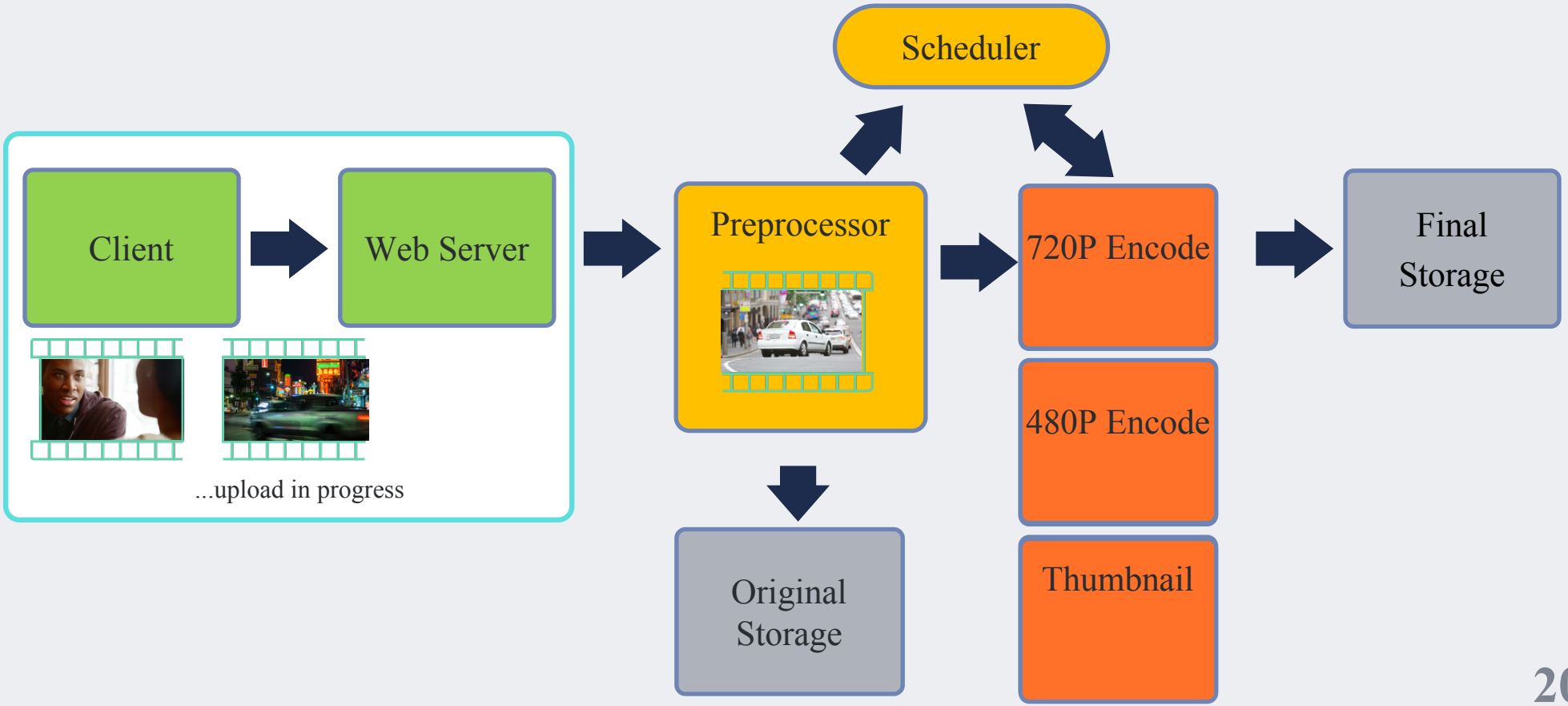
Overlap upload and processing



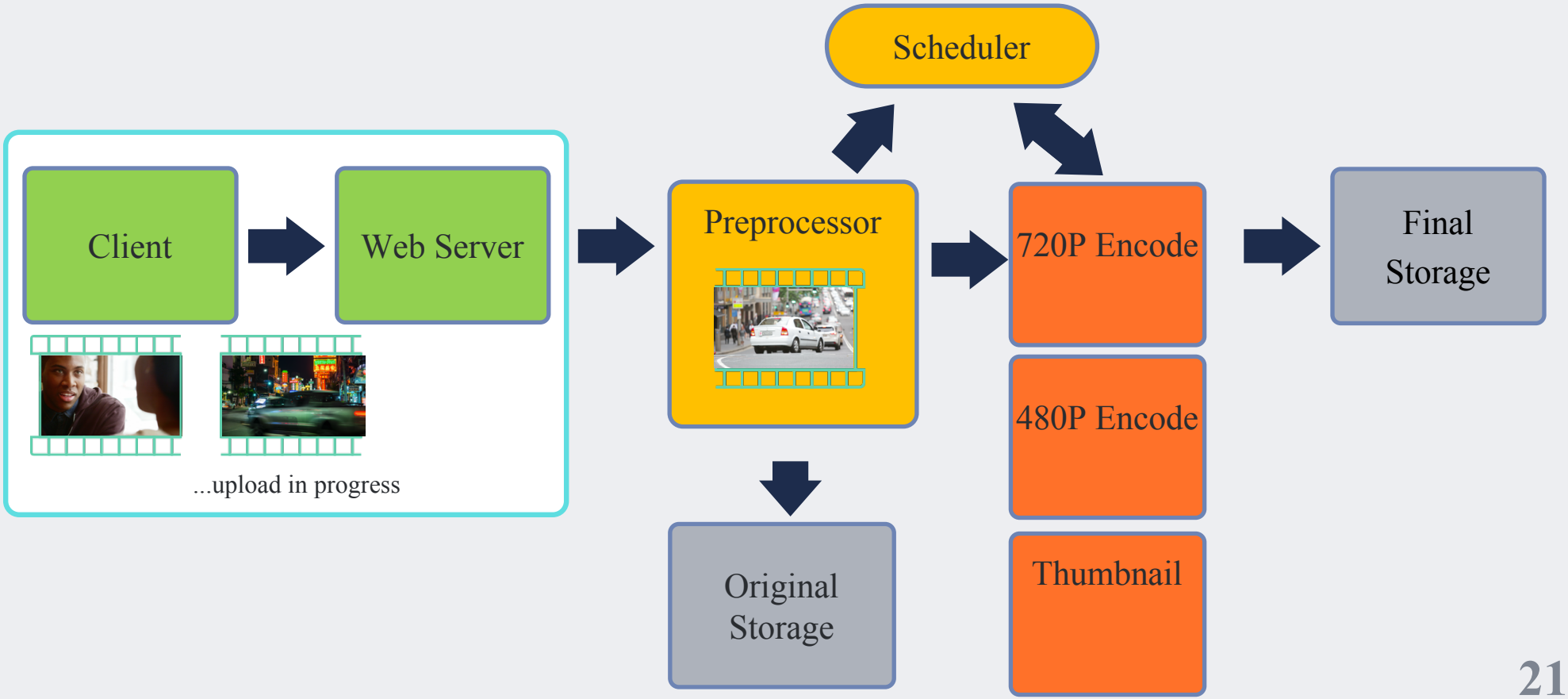
Overlap upload and processing



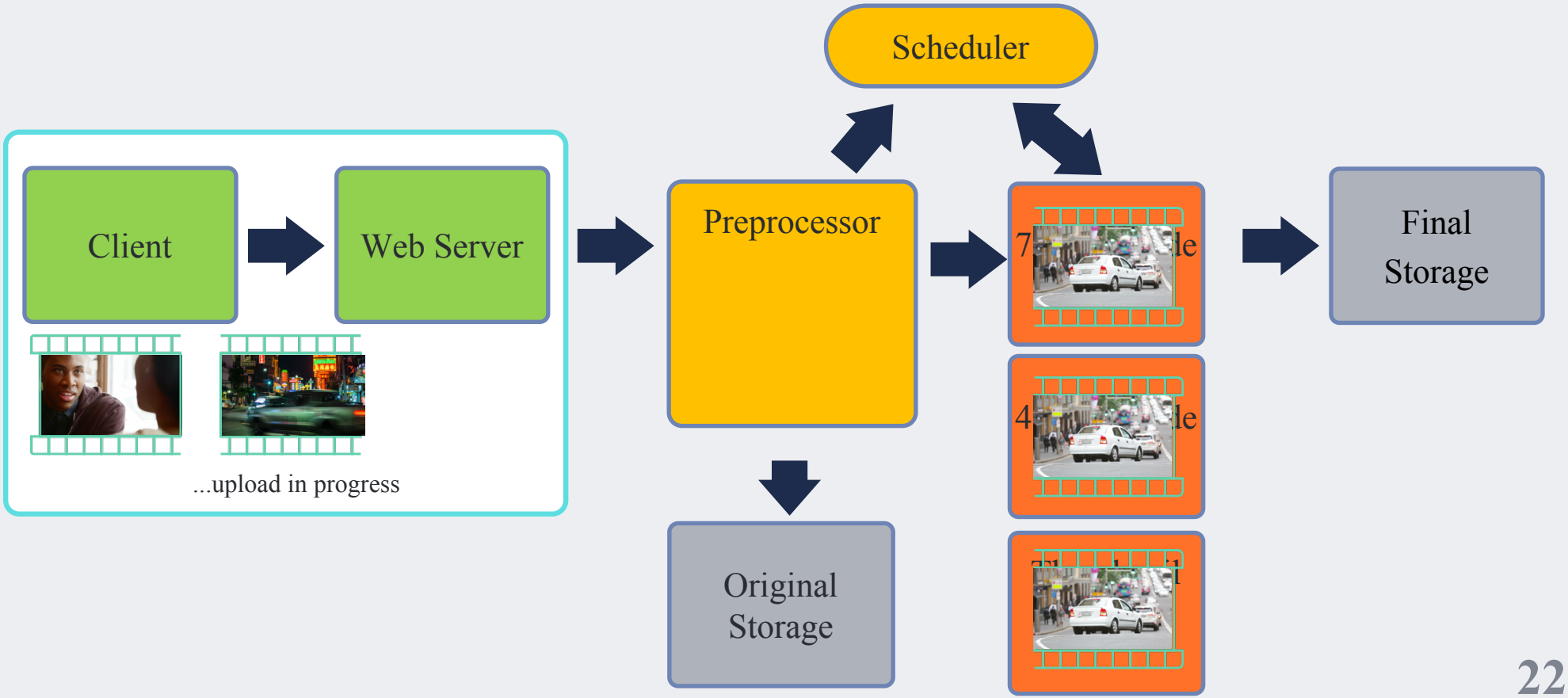
Parallel processing w/ many workers



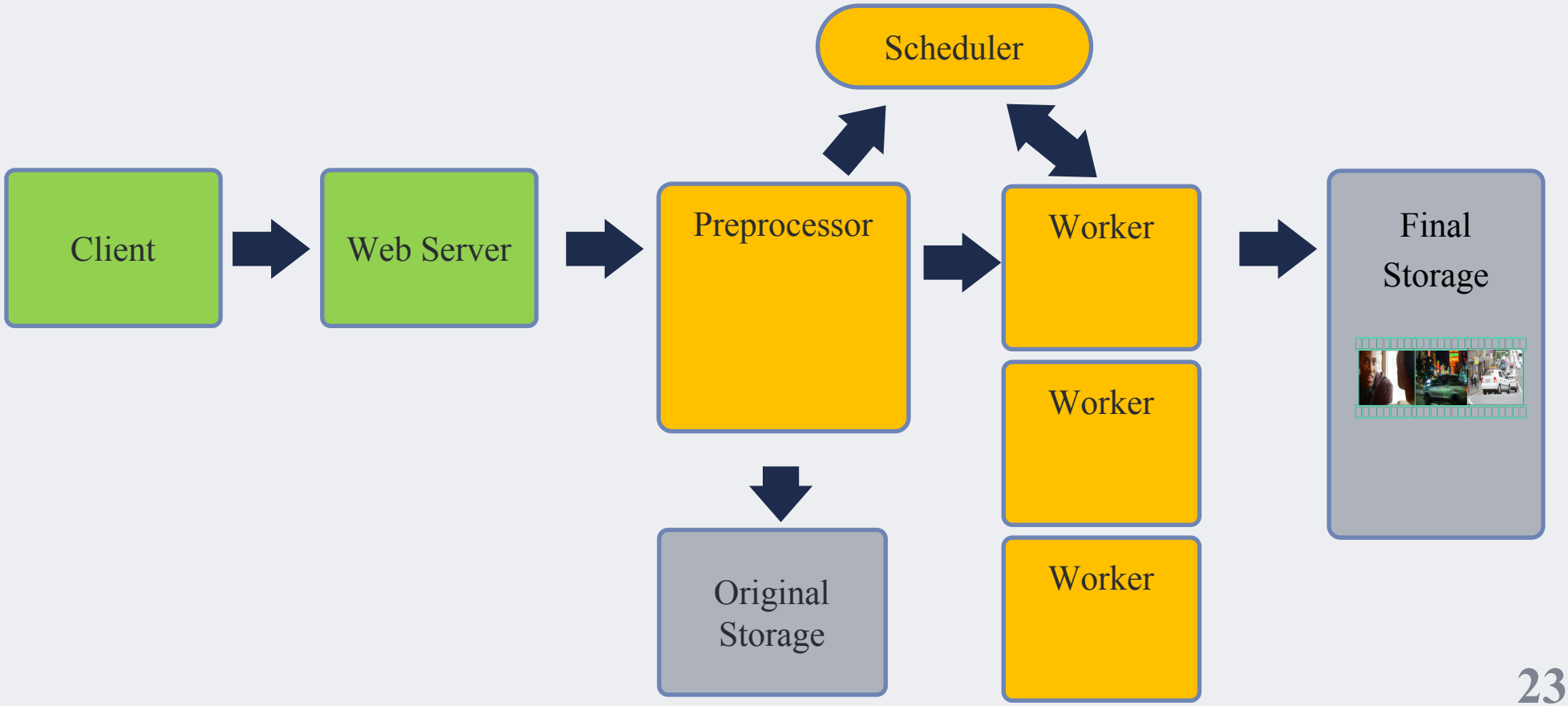
Parallel processing w/ many workers



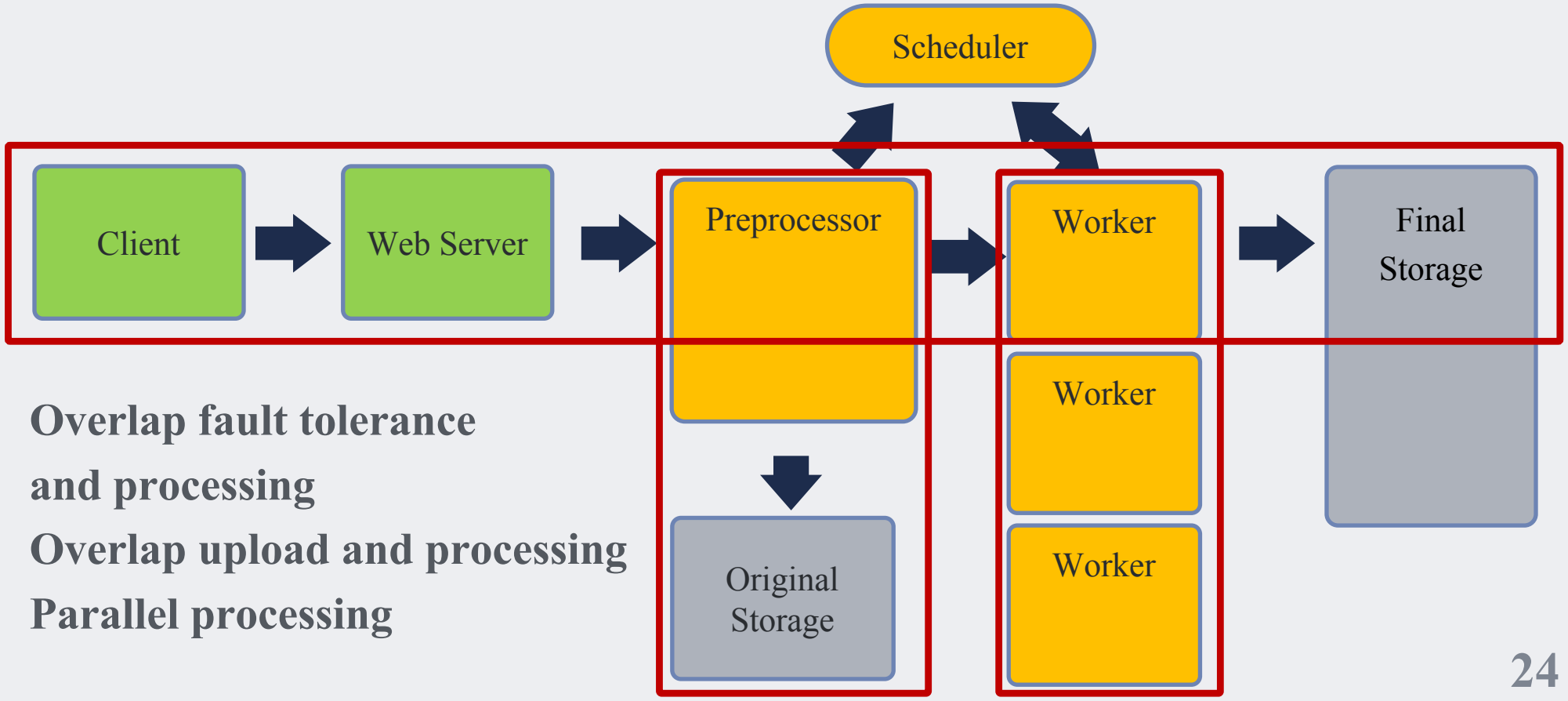
Parallel processing w/ many workers



Parallel processing w/ many workers



Three sources of parallelism

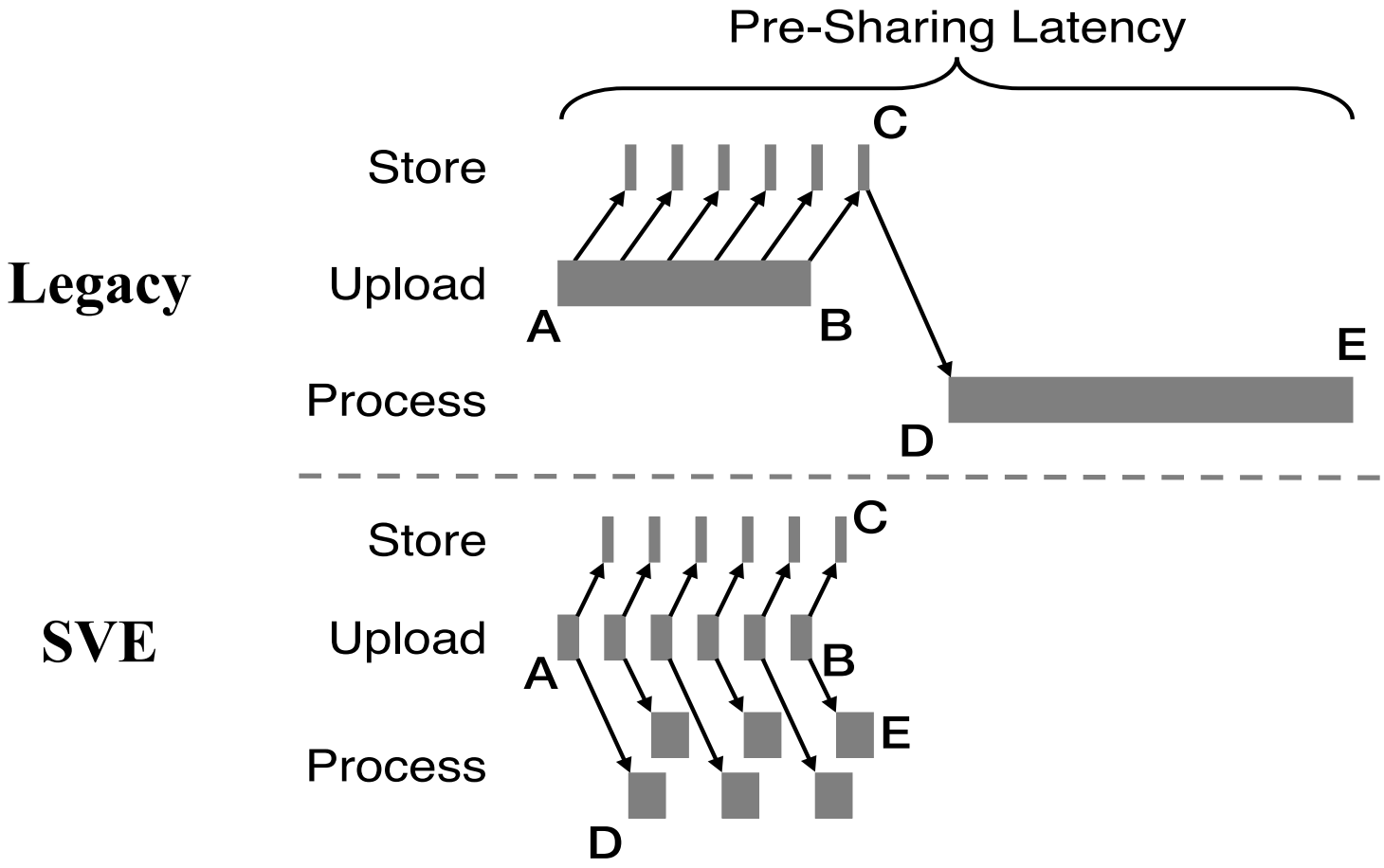


Overlap fault tolerance and processing

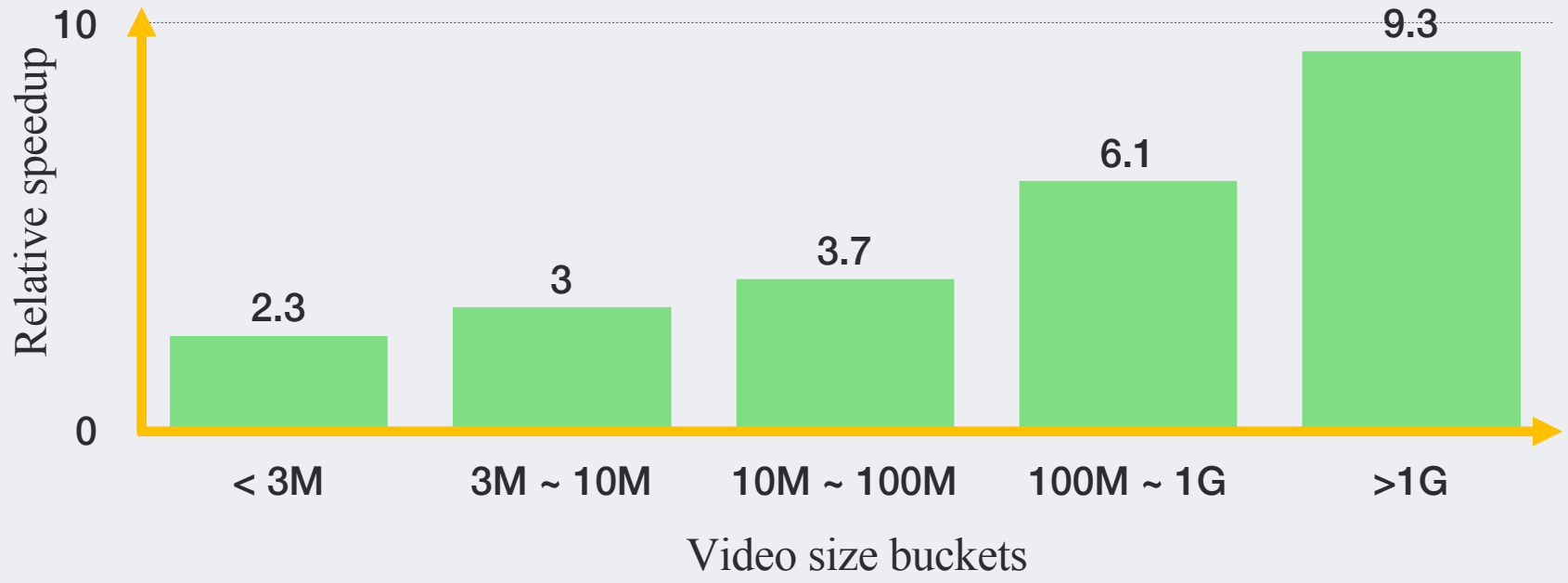
Overlap upload and processing

Parallel processing

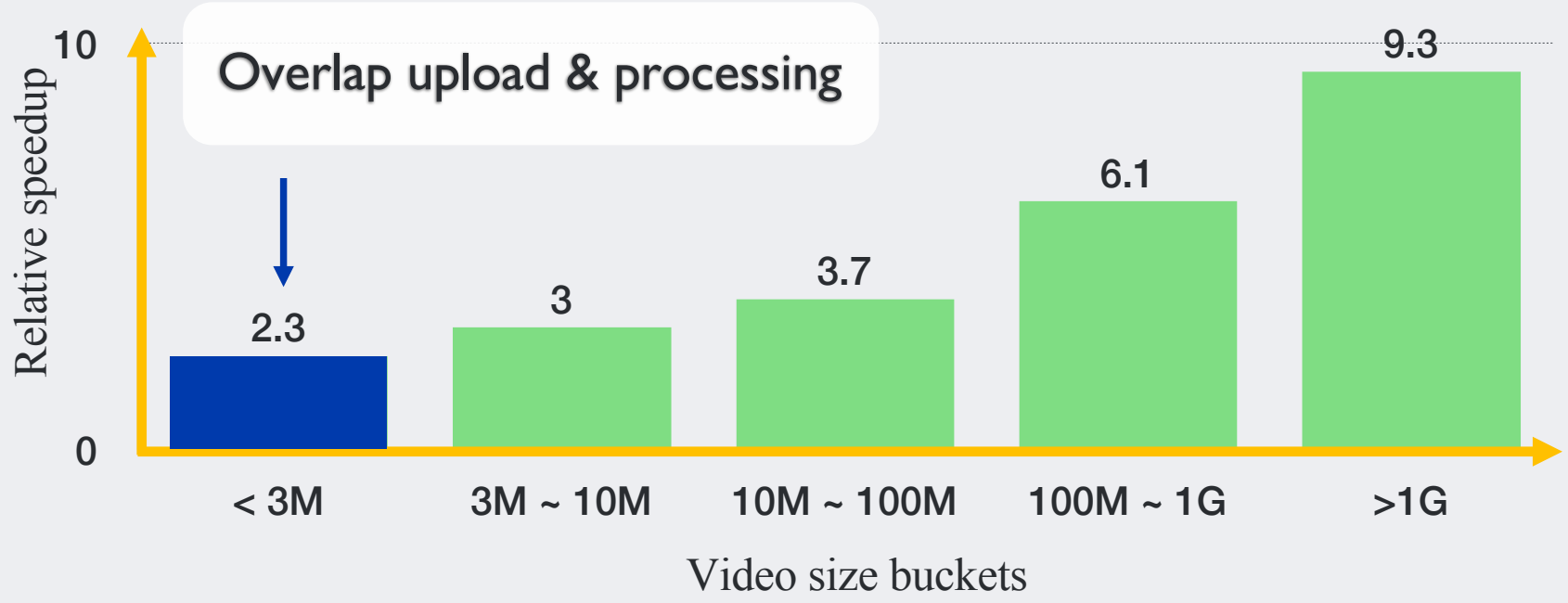
Let's Make This Faster!



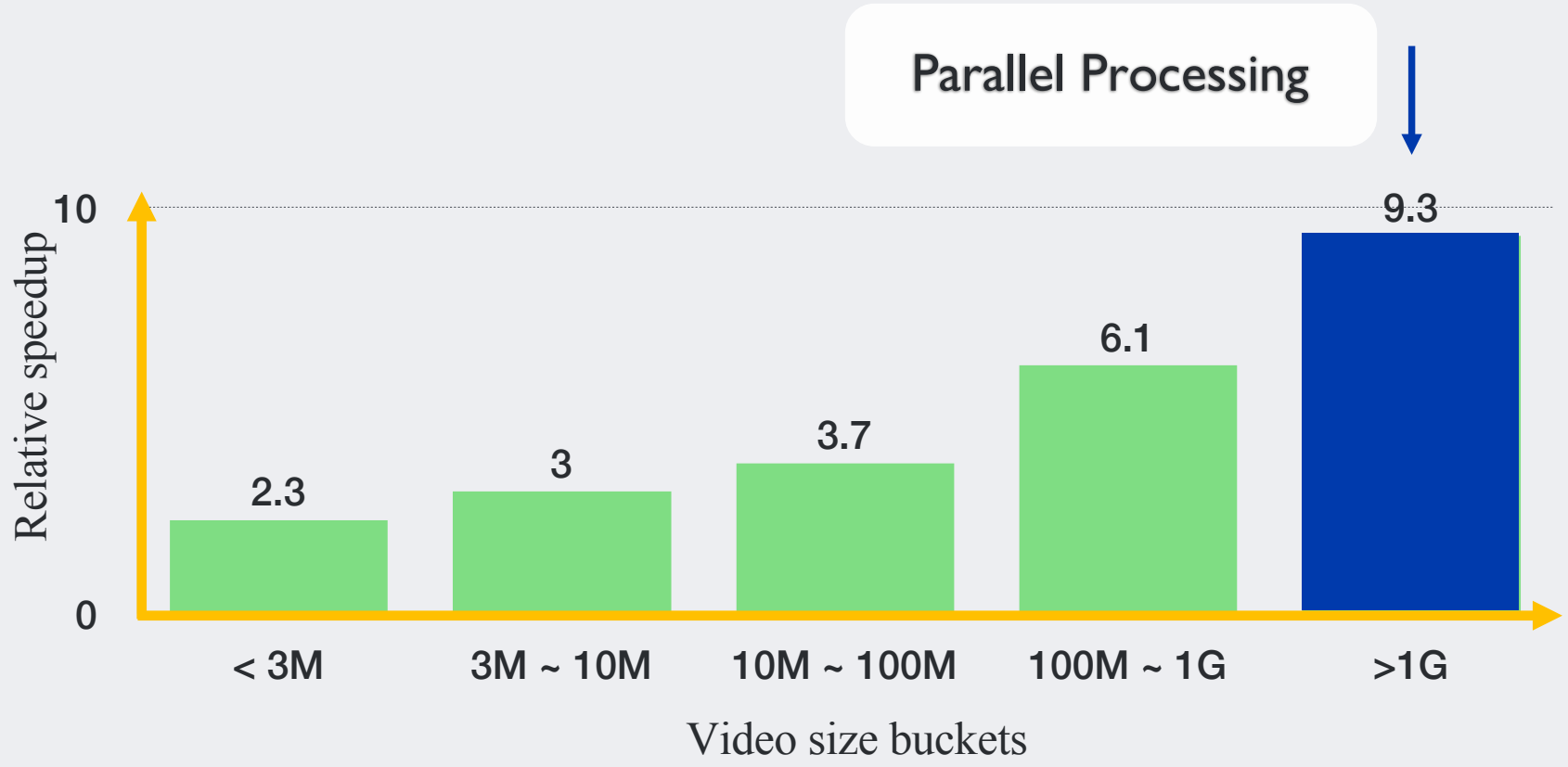
Results: 2.3x ~ 9.3x speedup



Results: 2.3x ~ 9.3x speedup



Results: 2.3x ~ 9.3x speedup

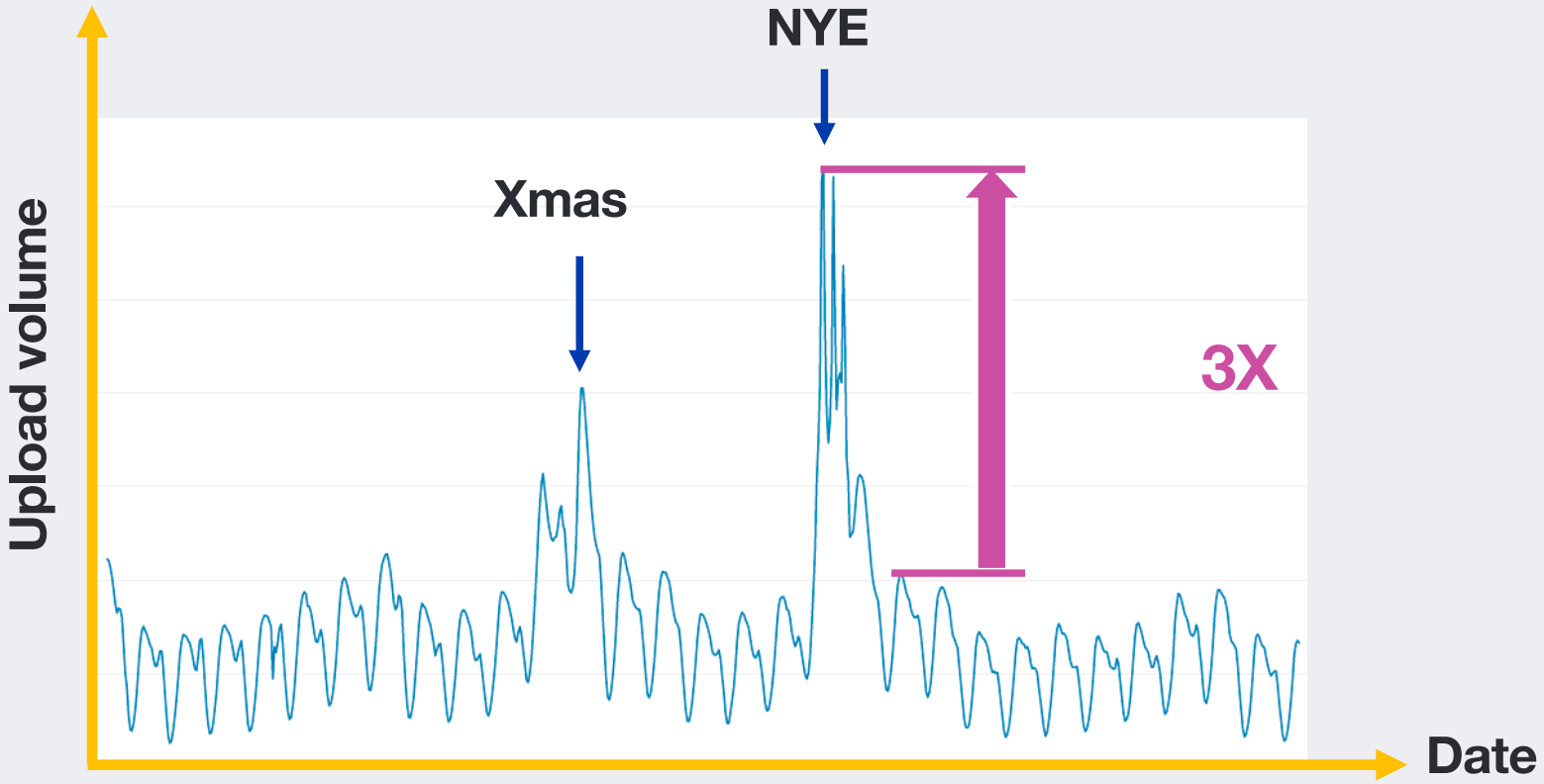


Robust: tolerate overload

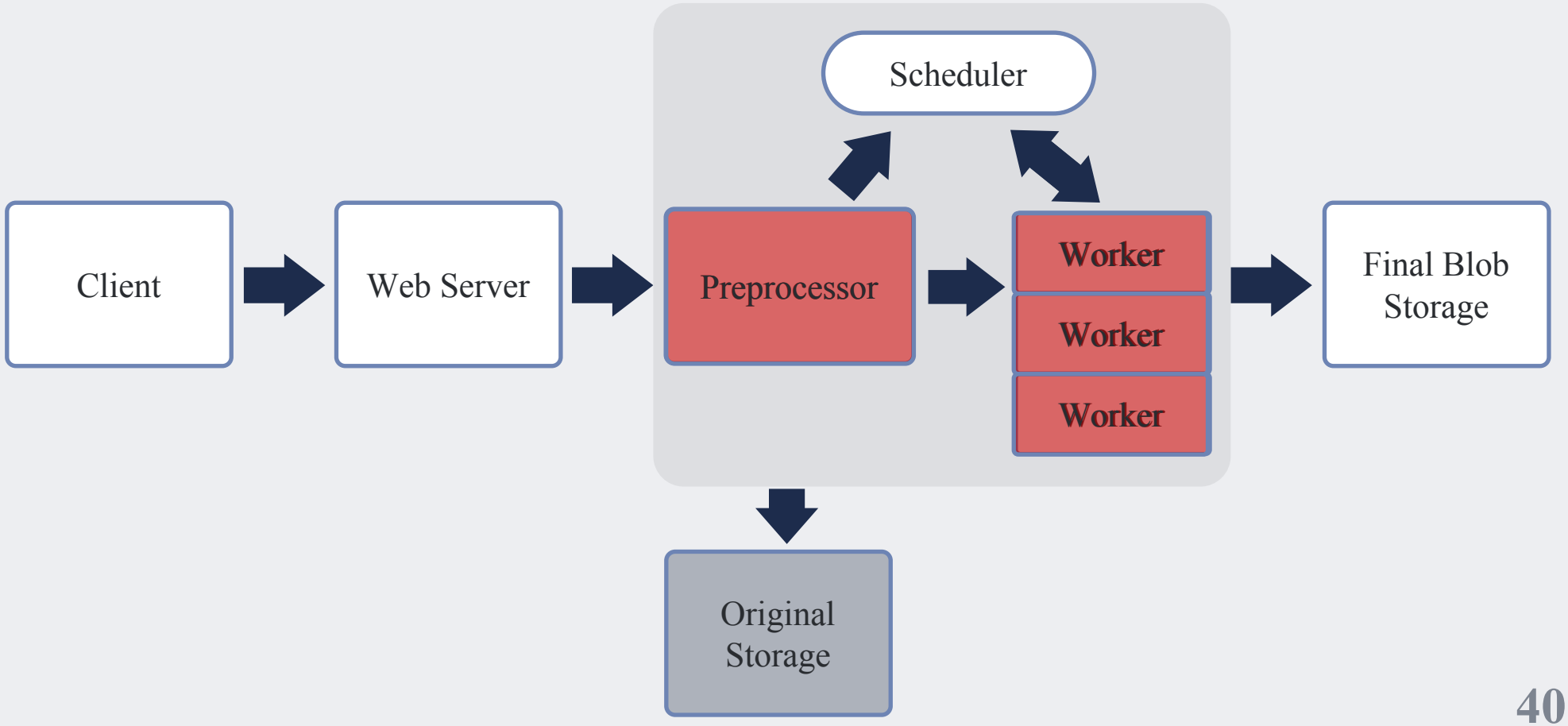
Handle faults and overload that is inevitable at scale

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

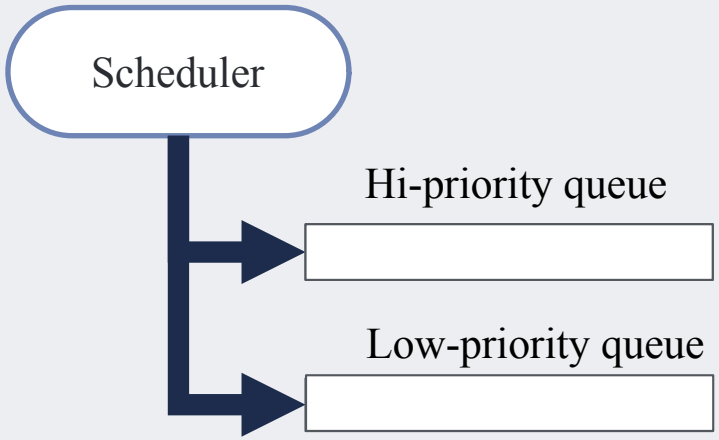
3X peak load during New Year Eve



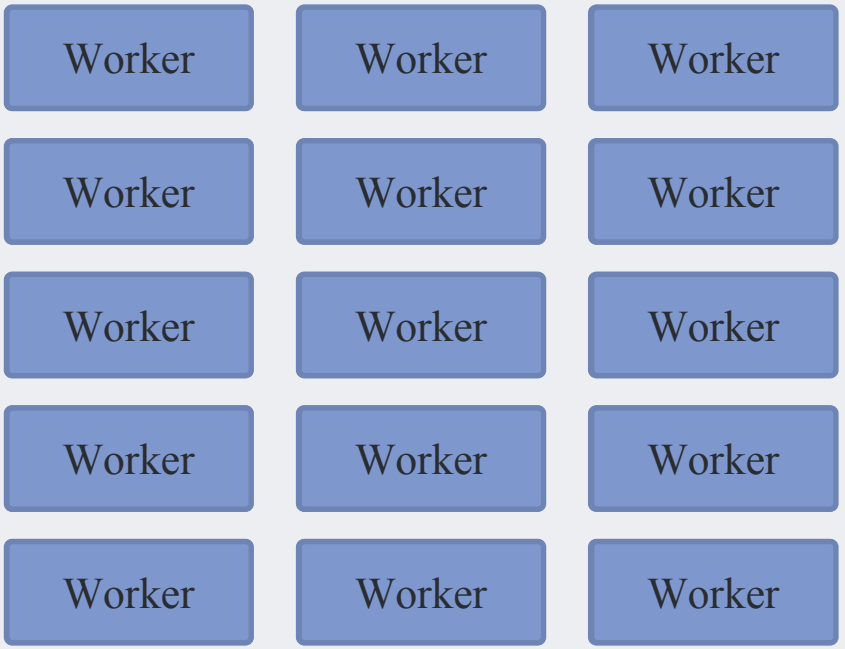
Prepare for overload



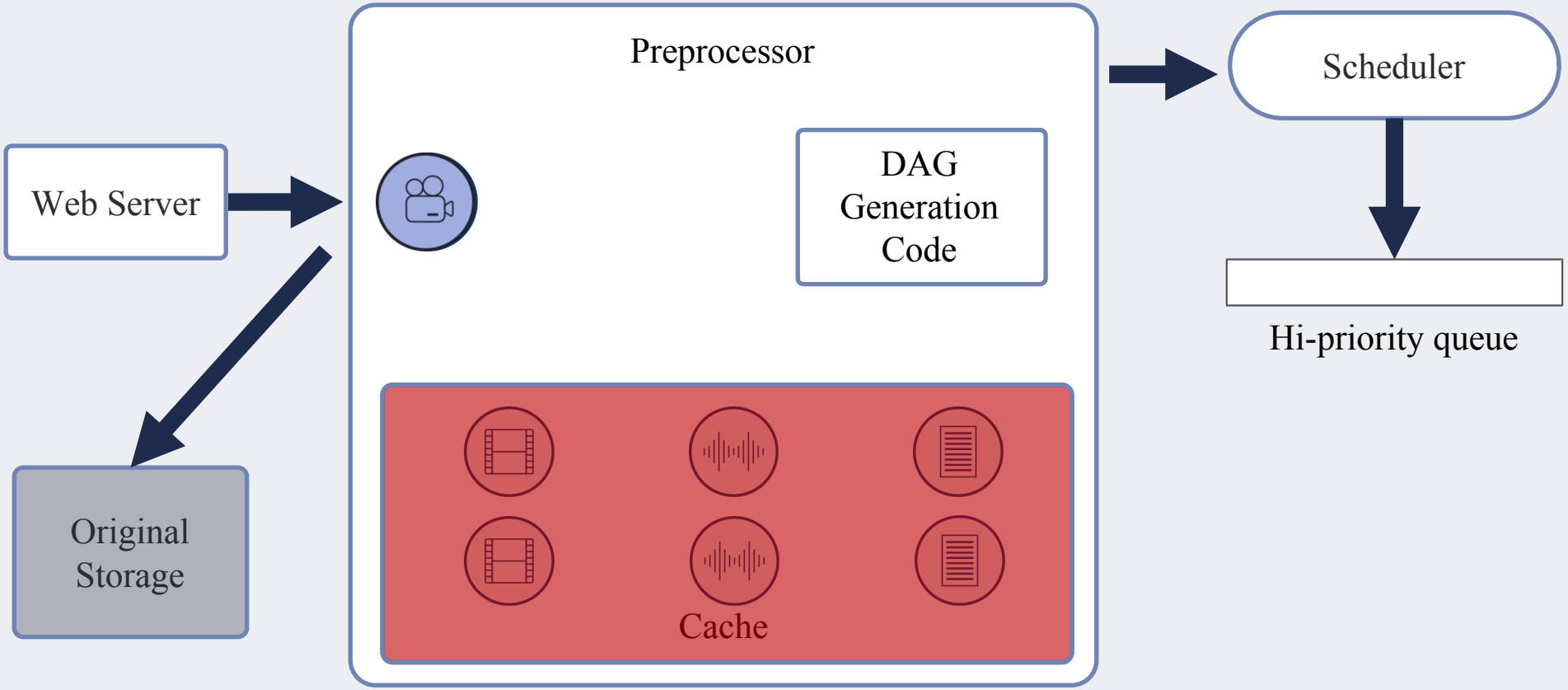
Use priority for worker overload



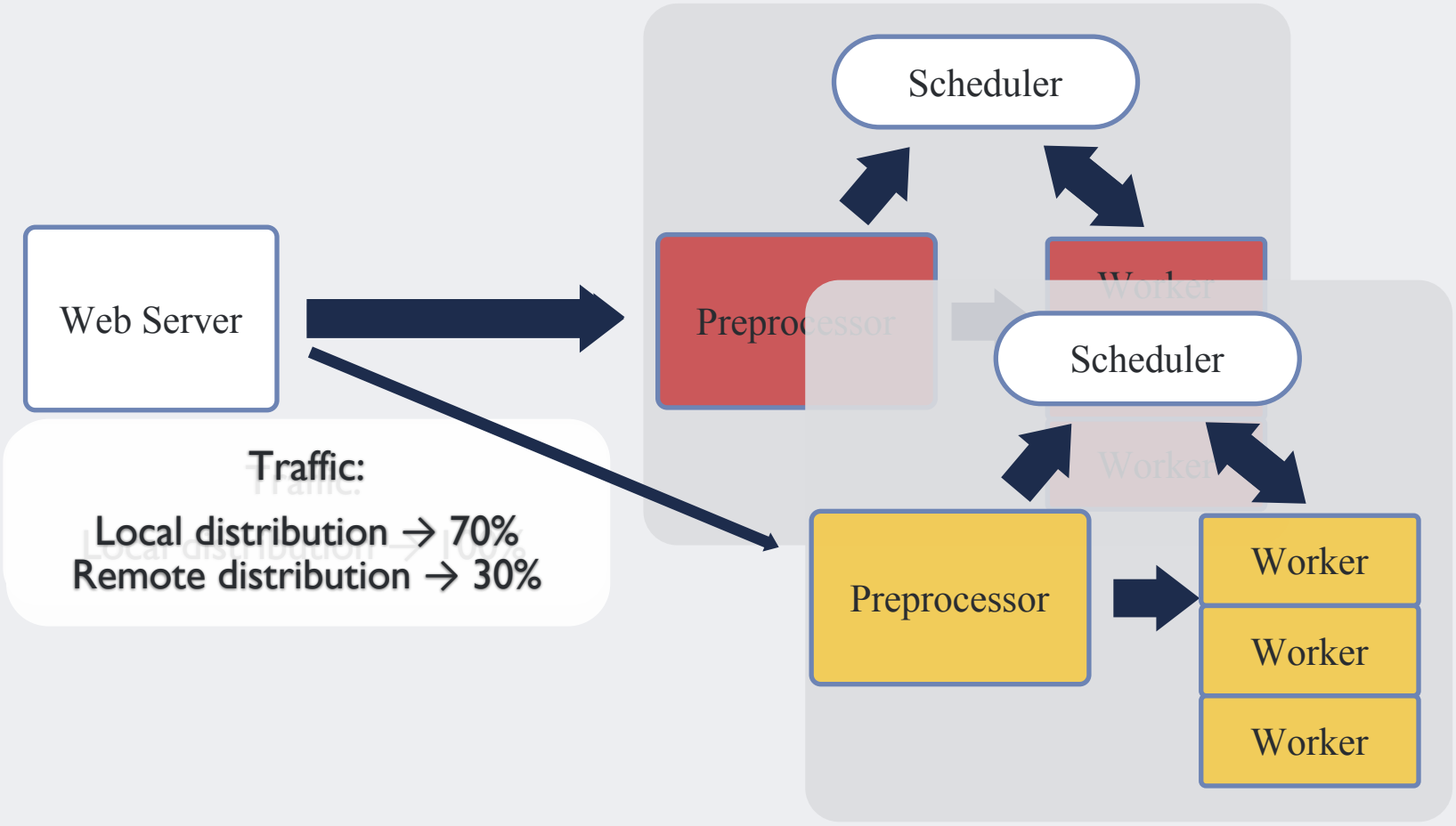
Only assign hi-pri tasks under overload



Defer full video processing



Regional redirection



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?

Summary

- Motivation for video processing
  - (How streaming video works)

- Legacy design – Serial processing was slow

- SVE design – Three sources of parallelism make SVE faster
  - Overlap upload and processing
  - Overlap fault tolerance and processing
  - Parallel processing