Encoding, Fast and Slow:
Low-Latency Video Processing Using Thousands of Tiny Threads

Sadjad Fouladi¹, Riad S. Wahby¹, Brennan Shacklett¹,
Karthikeyan Vasuki Balasubramaniam², William Zeng¹,
Rahul Bhalerao², Anirudh Sivaraman³, George Porter², Keith Winstein¹

¹Stanford University, ²UC San Diego, ³MIT

https://ex.camera
Outline

• Vision & Goals
  • mu: Supercomputing as a Service
  • Fine-grained Parallel Video Encoding
  • Evaluation
• Conclusion & Future Work
The challenges

- Low-latency video processing would need **thousands of threads, running in parallel**, with **instant startup**.

- However, **the finer-grained the parallelism, the worse the compression efficiency**.
Enter ExCamera

- We made two contributions:
  - Framework to run **5,000-way parallel jobs** with IPC on a commercial “cloud function” service.
  - Purely functional video codec for **massive fine-grained parallelism**.
- We call the whole system **ExCamera**.
Outline

• Vision & Goals

• mu: Supercomputing as a Service

• Fine-grained Parallel Video Encoding

• Evaluation

• Conclusion & Future Work
Where to find thousands of threads?

- IaaS services provide virtual machines (e.g. EC2, Azure, GCE):
  - Thousands of threads
  - Arbitrary Linux executables

👎 Minute-scale startup time (OS has to boot up, ...)

👎 High minimum cost (60 mins EC2, 10 mins GCE)

3,600 threads on EC2 for one second ➔ >$20
Cloud function services have (as yet) unrealized power

- AWS Lambda, Google Cloud Functions
- Intended for event handlers and Web microservices, *but*...
- Features:
  - ✔ Thousands of threads
  - ✔ Arbitrary Linux executables
  - ✔ Sub-second startup
  - ✔ Sub-second billing → 3,600 threads for one second → 10¢
**mu**, supercomputing as a service

- We built *mu*, a library for designing and deploying general-purpose parallel computations on a commercial “cloud function” service.

- The system starts up thousands of threads in seconds and manages inter-thread communication.

- *mu* is open-source software: [https://github.com/excamera/mu](https://github.com/excamera/mu)
Outline

• Vision & Goals
• mu: Supercomputing as a Service
• Fine-grained Parallel Video Encoding
• Evaluation
• Conclusion & Future Work
Now we have the threads, but...

• With the existing encoders, the finer-grained the parallelism, the worse the compression efficiency.
Video Codec

• A piece of software or hardware that compresses and decompresses digital video.
How video compression works

• Exploit the temporal redundancy in adjacent images.

• Store the first image on its entirety: a **key frame**.

• For other images, only store a "diff" with the previous images: an **interframe**.

**In a 4K video @15Mbps, a key frame is ~1 MB, but an interframe is ~25 KB.**
Existing video codecs only expose a simple interface

\[
\text{encode}([\Box, \Box, \ldots, \Box]) \rightarrow \text{keyframe + interframe}[2:n]
\]

\[
\text{decode}(\text{keyframe + interframe}[2:n]) \rightarrow [\Box, \Box, \ldots, \Box]
\]
Traditional parallel video encoding is limited

\[
\text{serial} \downarrow \\
\text{encode}(i[1:200]) \rightarrow \text{keyframe}_1 + \text{interframe}[2:200]
\]

\[
\text{parallel} \downarrow \\
\begin{align*}
\text{[thread 01]} & \quad \text{encode}(i[1:10]) \rightarrow \text{kf}_1 + \text{if}[2:10] \\
\text{[thread 02]} & \quad \text{encode}(i[11:20]) \rightarrow \text{kf}_1 + \text{if}[12:20] \\
\text{[thread 03]} & \quad \text{encode}(i[21:30]) \rightarrow \text{kf}_2 + \text{if}[22:30] \\
\vdots & \\
\text{[thread 20]} & \quad \text{encode}(i[191:200]) \rightarrow \text{kf}_{191} + \text{if}[192:200]
\end{align*}
\]

finer-grained parallelism \Rightarrow more key frames \Rightarrow worse compression efficiency
We need a way to start encoding mid-stream

- Start encoding mid-stream needs access to intermediate computations.
- Traditional video codecs *do not* expose this information.
- We formulated this internal information and we made it explicit: the “state”.
The decoder is an automaton
What we built: a video codec in explicit state-passing style

- VP8 decoder with no inner state:
  \[ \text{decode}(\text{state}, \text{frame}) \rightarrow (\text{state}', \text{image}) \]

- VP8 encoder: resume from specified state
  \[ \text{encode}(\text{state}, \text{image}) \rightarrow \text{interframe} \]

- Adapt a frame to a different source state
  \[ \text{rebase}(\text{state}, \text{image}, \text{interframe}) \rightarrow \text{interframe}' \]
Putting it all together: ExCamera

- Divide the video into tiny chunks:
  - [Parallel] **encode** tiny independent chunks.
  - [Serial] **rebase** the chunks together and remove extra keyframes.
1. [Parallel] Download a tiny chunk of raw video

<table>
<thead>
<tr>
<th>thread 1</th>
<th>thread 2</th>
<th>thread 3</th>
<th>thread 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2 3 6</td>
<td>7 11 12</td>
<td>13 17 18</td>
<td>19 23 24</td>
</tr>
</tbody>
</table>
2. [Parallel] vpxenc → keyframe, interframe[2:n]

Google's VP8 encoder

\text{encode}(\text{img}[1:n]) \rightarrow \text{keyframe} + \text{interframe}[2:n]
3. [Parallel] \( \text{decode} \rightarrow \text{state} \rightarrow \text{next thread} \)

Our explicit-state style decoder

\[ \text{decode}(\text{state}, \text{frame}) \rightarrow (\text{state}', \text{image}) \]
4. [Parallel] last thread’s state $\rightarrow$ encode

Our explicit-state style encoder

$$\text{encode}(\text{state, image}) \rightarrow \text{interframe}$$
5. [Serial] *last thread’s state* $\rightarrow$ *rebase* $\rightarrow$ *state* $\rightarrow$ *next thread*

Adapt a frame to a different source state

$\text{rebase}(\text{state, image, interframe}) \rightarrow \text{interframe'}$
5. [Serial] *last thread’s state* $\rightarrow$ *rebase* $\rightarrow$ *state* $\rightarrow$ *next thread*

Adapt a frame to a different source state

**rebase***(state, image, interframe) $\rightarrow$ interframe’*
6. [Parallel] Upload finished video
<table>
<thead>
<tr>
<th>Format</th>
<th>Encoder Type</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>14.8-minute 4K Video</td>
<td>vpxenc Single-Threaded</td>
<td>453 mins</td>
</tr>
<tr>
<td></td>
<td>vpxenc Multi-Threaded</td>
<td>149 mins</td>
</tr>
<tr>
<td></td>
<td>YouTube (H.264)</td>
<td>37 mins</td>
</tr>
<tr>
<td></td>
<td>ExCamera[6, 16]</td>
<td>2.6 mins</td>
</tr>
</tbody>
</table>
Takeaways

• Low-latency video processing

• Two major contributions:
  
  • Framework to run 5,000-way parallel jobs with IPC on a commercial “cloud function” service.
  
  • Purely functional video codec for massive fine-grained parallelism.
  
  • 56× faster than existing encoder, for <$6.

https://ex.camera | excamera@cs.stanford.edu