A Computing Platform for Video Crowdprocessing Using Deep Learning

Zongqing Lu, Kevin Chan, and Thomas La Porta

北京大學
PEKING UNIVERSITY

PennState

A few years ago

Now

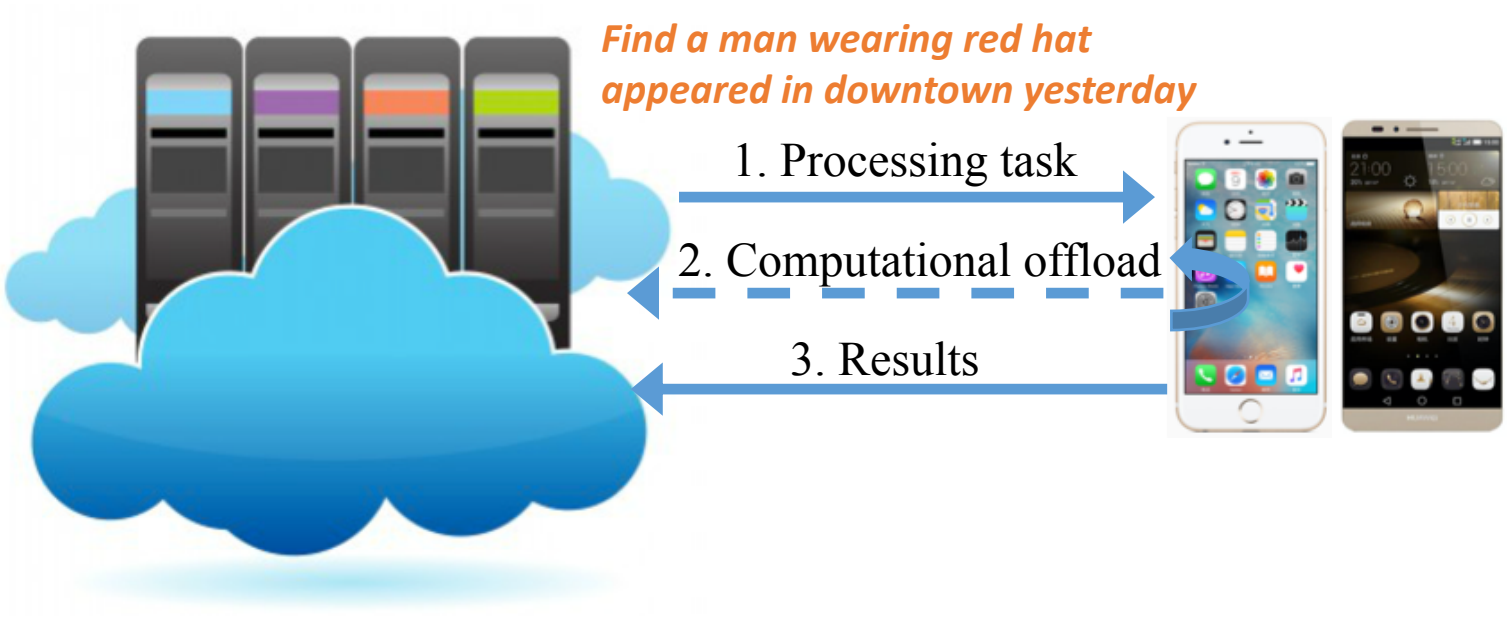
- Widespread adoption of smartphones
- Explosion of videos
- Videos stored on networked mobile devices

- Videos are a rich source of **information**
- The information could be gathered for **a variety of purposes**
- Videos can be exploited for **on-demand information retrieval**

Challenges

- A huge collection of videos
  - Impossible to collect and process them at some centralized entity, on-demand
- Video processing is computationally demanding (object detection, activity recognition, etc)
  - *Deep learning frameworks, e.g., caffe, TensorFlow*
- Resource-constrained mobile devices
  - Limited computational capability
  - Battery, network bandwidth, and data usage

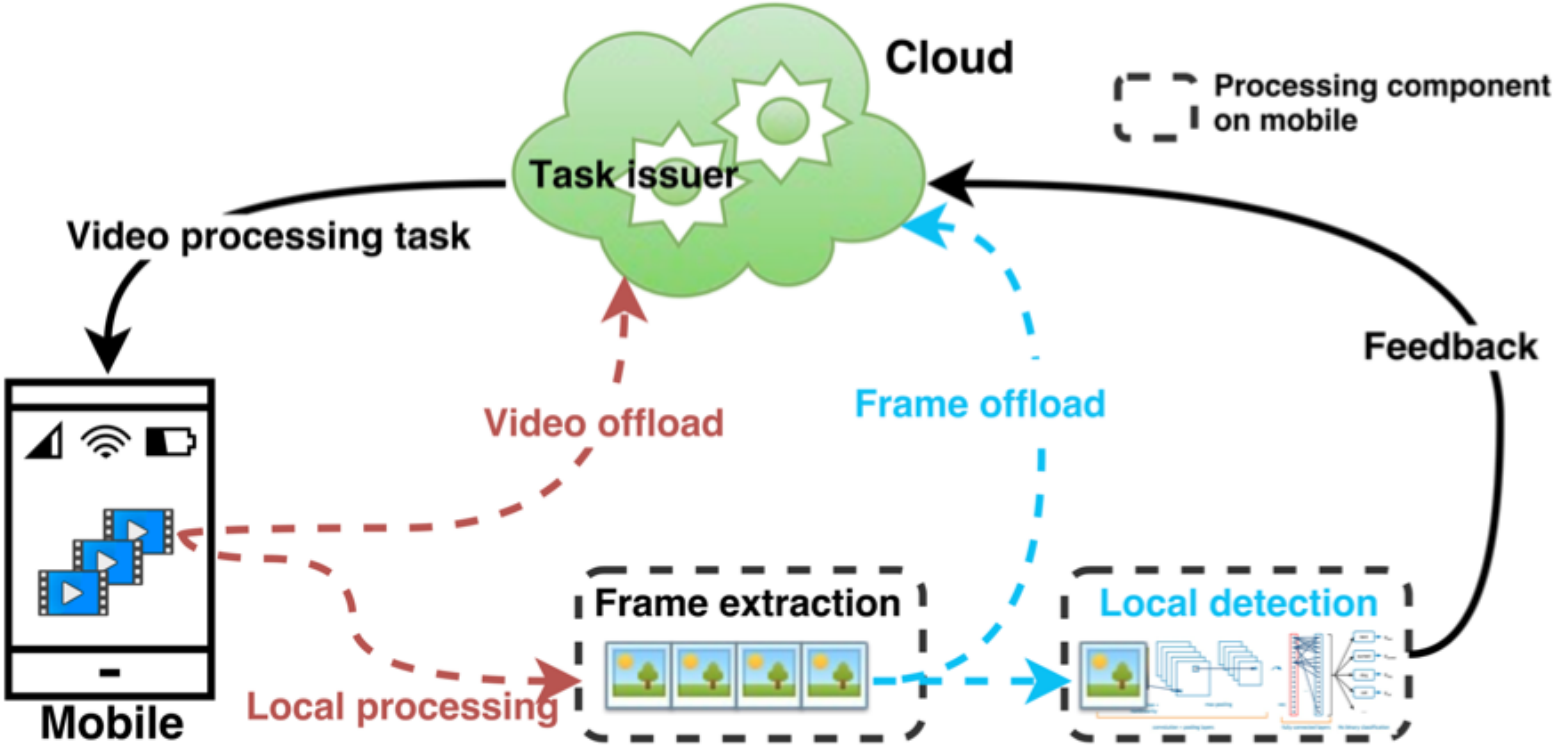
Video Crowdprocessing



- Processing tasks are issued from cloud to participants
- Participants process locally stored videos with assistant of cloud
- Participants return results

CrowdVision

- A distributed, energy-efficient computing platform for video *crowdprocessing* using deep learning

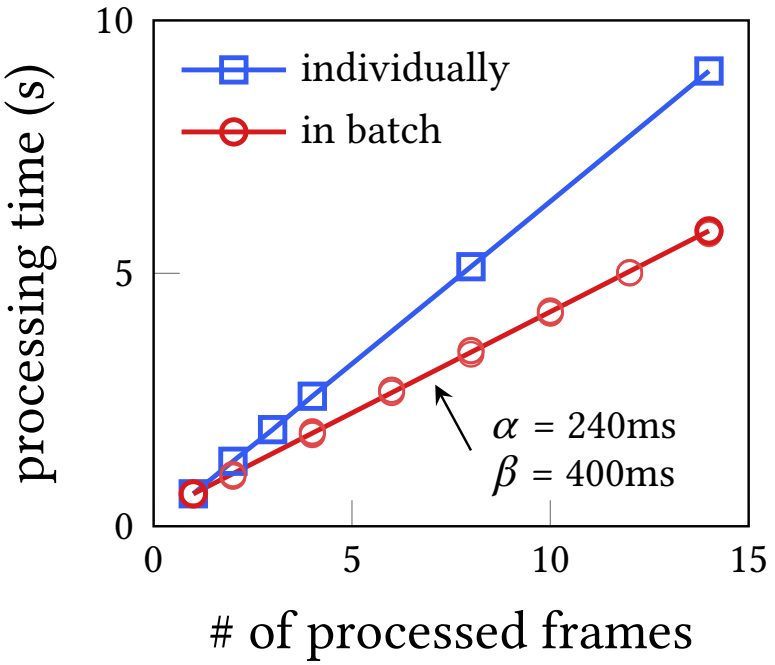


Detection

- Frame-based processing using *Caffe*

Galaxy S5	Power	CPU usage
Object detection (CPU)	2191mW	25%

Caffe on Galaxy S5



$t = \alpha + \beta x$

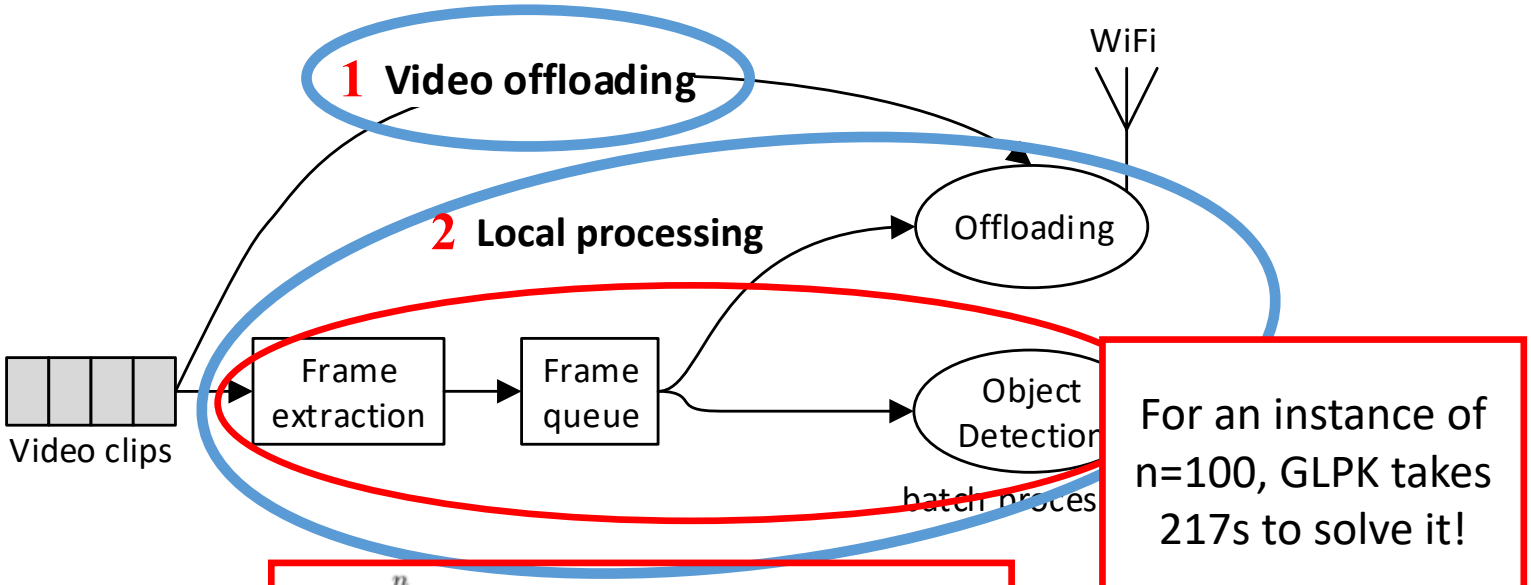
Offloading

- WiFi
  - Steady data rate
  - Optimize ***completion time*** (task issuer's concern) without/with ***energy constraint*** (user' concern)
- Cellular
  - Varying data rate
  - Tradeoff between ***completion time*** and ***energy*** with cellular ***data usage constraint***

We separate these two scenarios throughout the design of CrowdVision **for ease of presentation**

Processing under WiFi

- Optimizing completion time



For an instance of n=100, GLPK takes 217s to solve it!

ILP

$$\min \sum_{i=0}^{n} x_i$$

$$\text{s.t.} \sum_{i=1}^{n} y_i = n,$$

$$\sum_{i=0}^{k} x_i \geq \sum_{i=1}^{k+1} \gamma y_i, \qquad k = 0, \ldots, n-1$$

$$x_i + b_i \alpha \geq \alpha + \beta y_i, \qquad i = 1, \ldots, n$$

$$n(1 - b_i) \geq y_i \geq 0, \qquad i = 1, \ldots, n$$

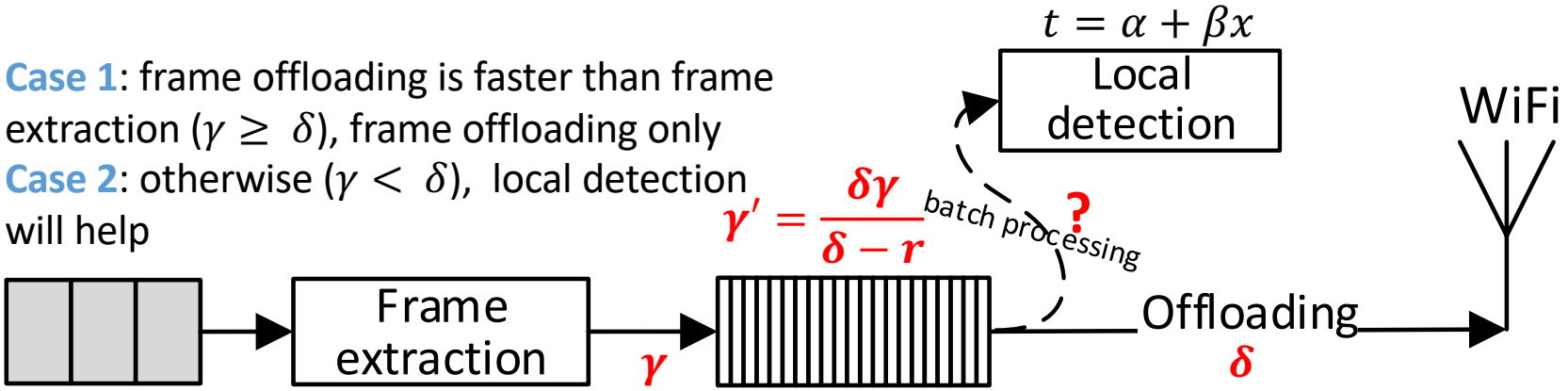
$$b_i = \{0, 1\}, \qquad i = 1, \ldots, n.$$

- *Split-Shift* algorithm
  - It determines both # of processing batches and # of frames for each batch

Split-Shift Algorithm

Case 1: frame offloading is faster than frame extraction ($\gamma \geq \delta$), frame offloading only

Case 2: otherwise ($\gamma < \delta$), local detection will help



Assuming all frames are available at the beginning

1 $\delta(n - n_p^*) = \alpha + \beta n_p^*$

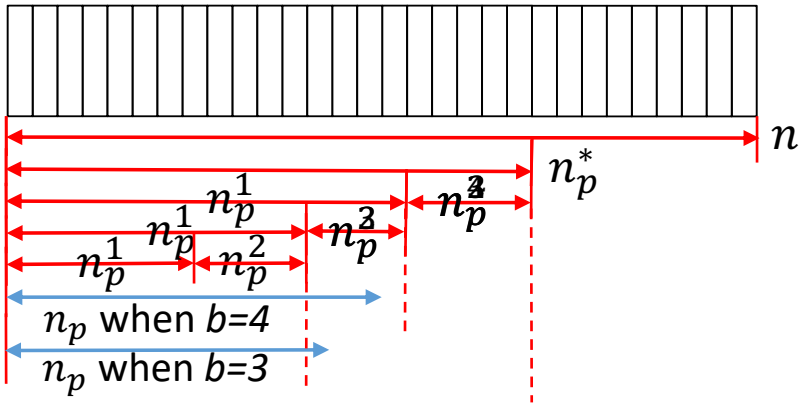
2 $n_p^* \gamma' > \alpha$

$n_p^1 \gamma' + \alpha + \beta n_p^1 \geq n_p^* \gamma'$

$n_p^1 \gamma' > \alpha$

3 ***split process***

$n_p^1 \gamma' <= \alpha$



$\delta(n - n_p) = n_p^1 \gamma' + \alpha b + \beta n_p$

$\sum_{i=1}^{b} n_p^i - n_p \geq n_p^b$

$b = b - 1$

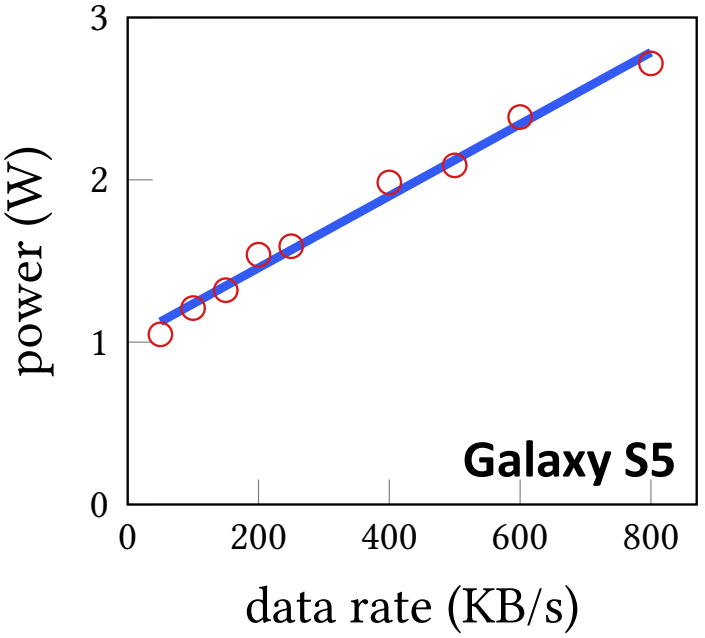
4 ***shift process***

Computational complexity $O(n)$

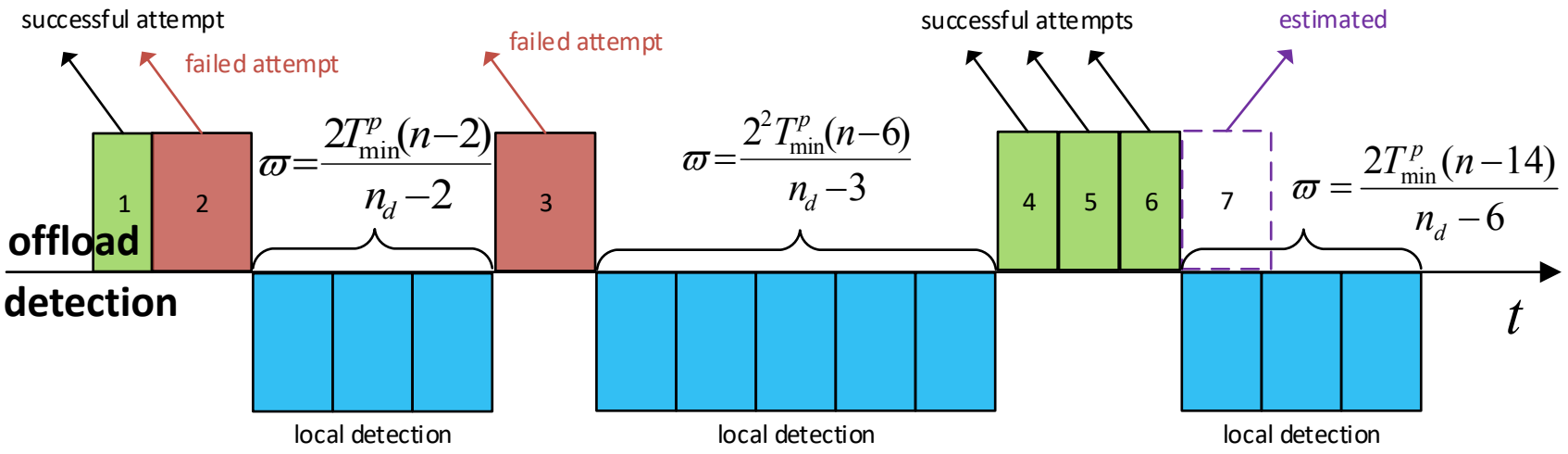
Processing under Cellular

- No video offloading
- ***Tradeoff*** between *completion time* and *energy* with *data usage constraints*

- Challenge: estimation of uplink rate and power
  - *Signal strength* to *data rate*
  - *Data rate* to *power*



Adaptive Algorithm



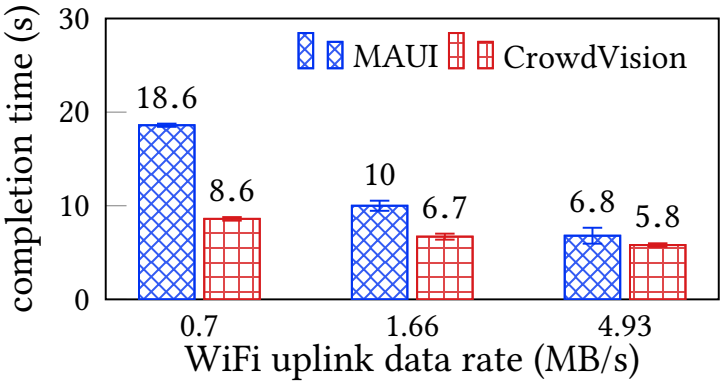
- Frames are continuously extracted.
- Frame offloading is attempted periodically.
- Switch to local detection when unsuccessful attempt occurs.
- Local detection processes the maximum number of frames within time period ω.

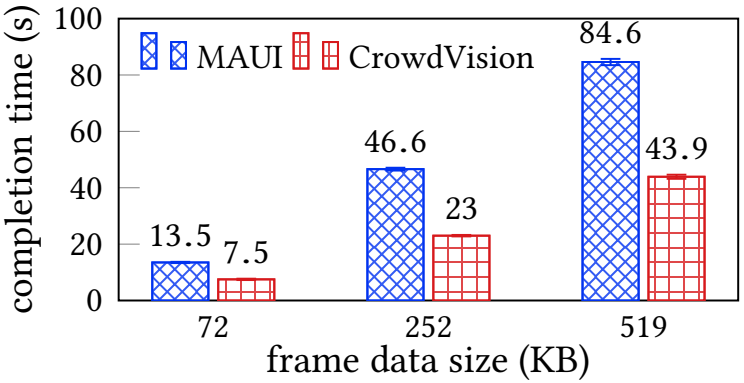
Evaluation

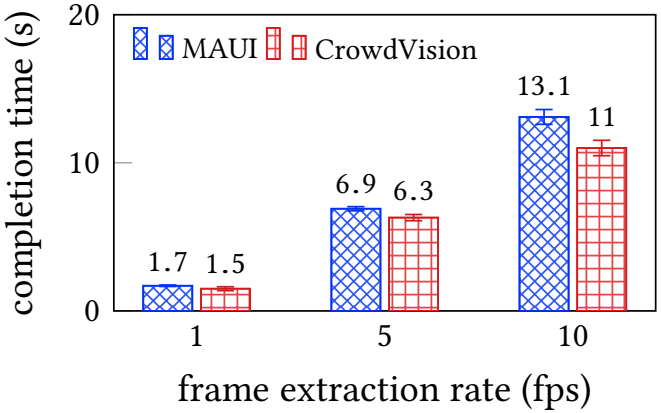
- Experimentation
  - Implemented on Android and Linux
  - Deployed on Galaxy S5 and a workstation with GeForce GTX TITAN X 12 GB GPUs
  - WiFi data rates: *0.7, 1.66, 4.93MB/s*
  - LTE: three different locations
  - Frame data sizes: *72, 152, 519KB (640x360, 1280x720, 1920x1080)*

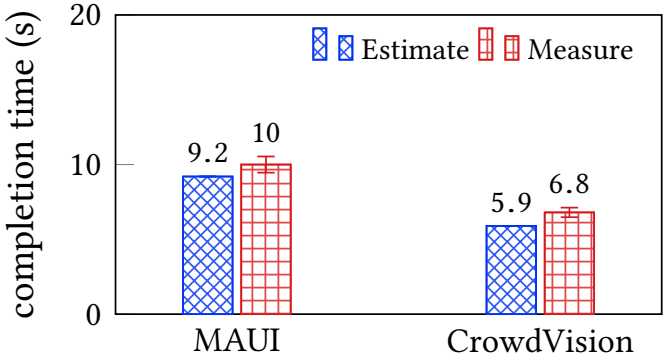
System Performance

- *Processing under WiFi*



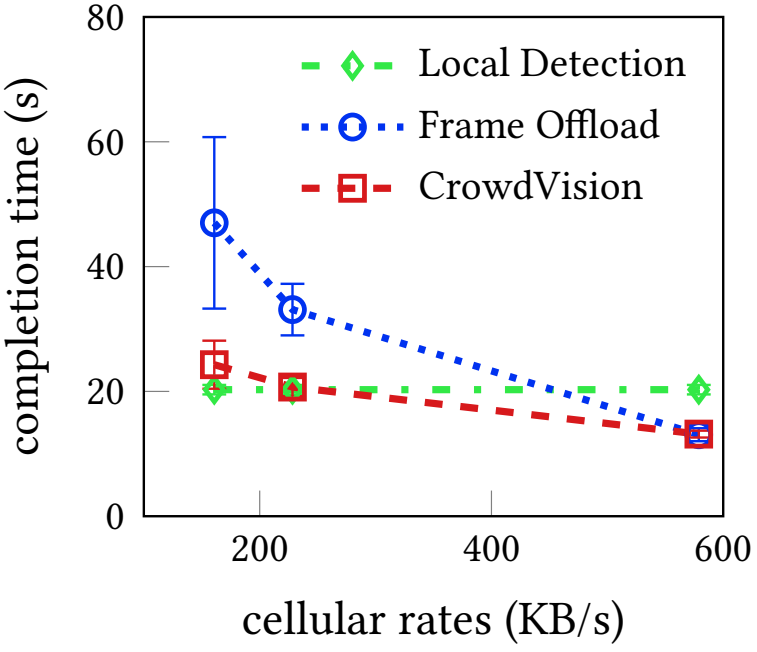


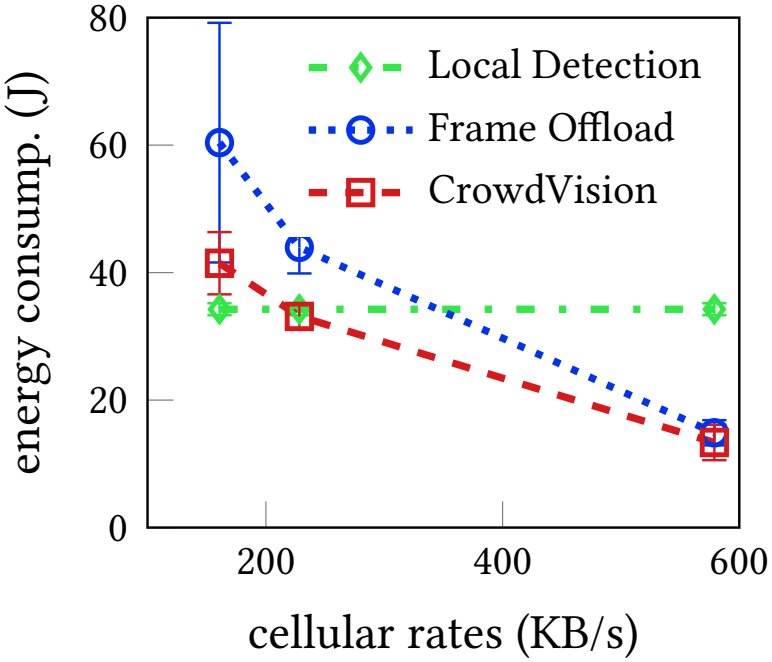




System Performance (cont'd)

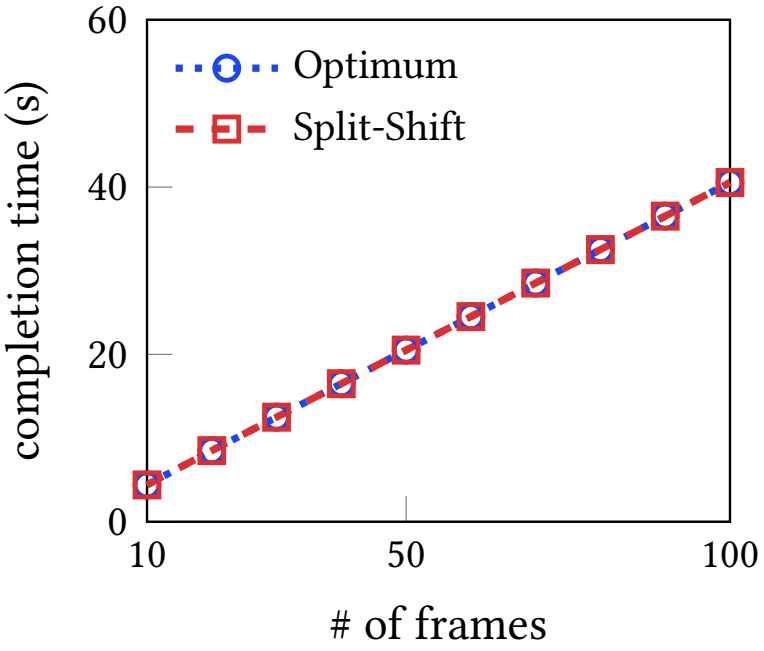
- *Processing under LTE*



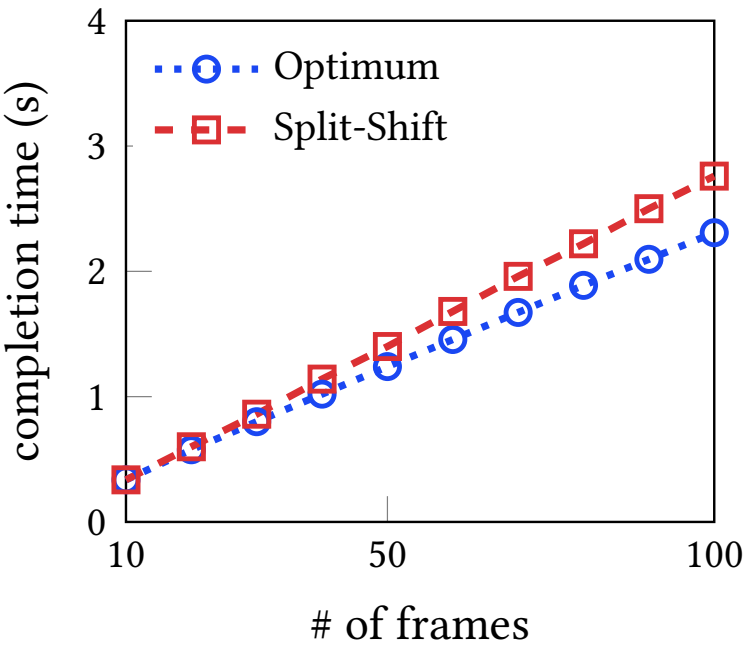


System Performance (cont'd)

➢ *Split-Shift Algorithm*



$\alpha = 240, \beta = 400, \gamma = 16$



$\alpha = 40, \beta = 20, \gamma = 16$

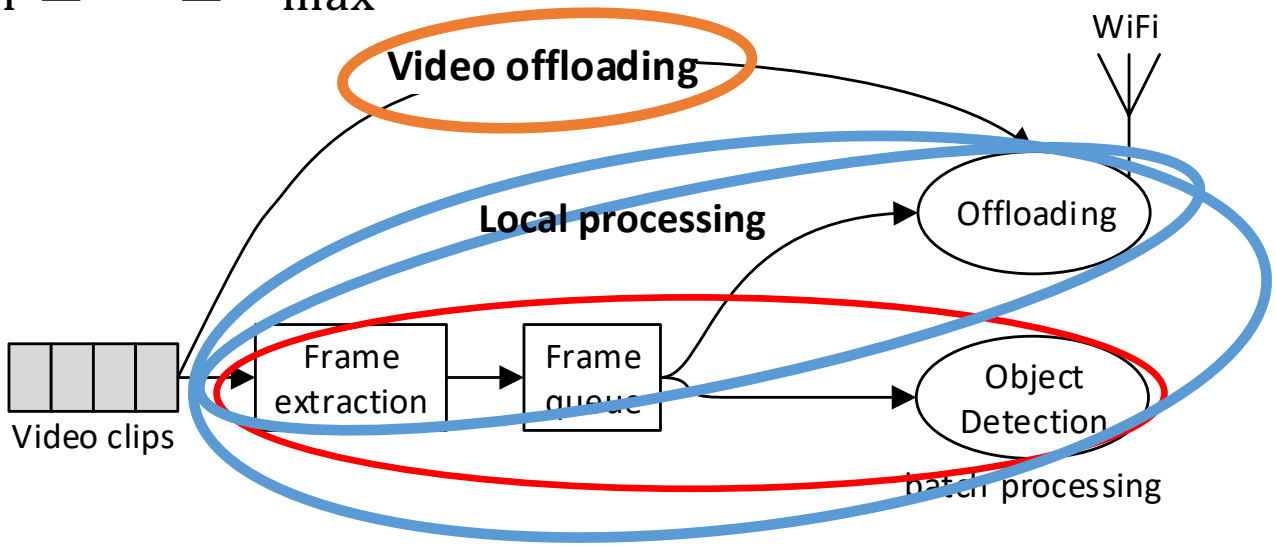
Thank You
Questions?

zongqing.lu@pku.edu.cn
https://z0ngqing.github.io

北京大学
PEKING UNIVERSITY

Processing under WiFi

- Optimizing completion time with energy constraint
  - $\mathbf{E}_{\min}$
  - Optimizing energy may not necessarily optimize the completion time
  - $\mathbf{E}_{\max} = E(\mathbf{T}_{\min})$
  - $\mathbf{E}_{\min} \leq E' \leq \mathbf{E}_{\max}$

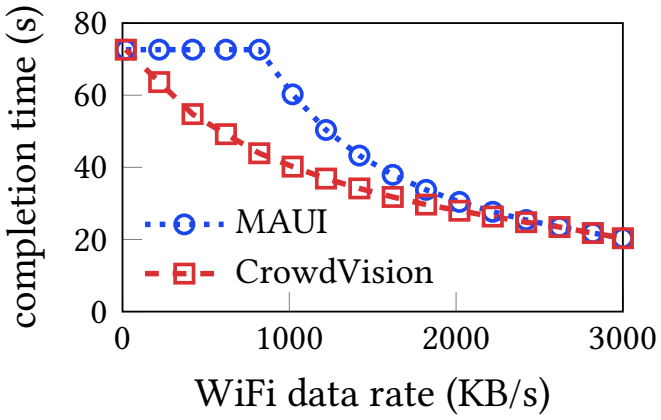


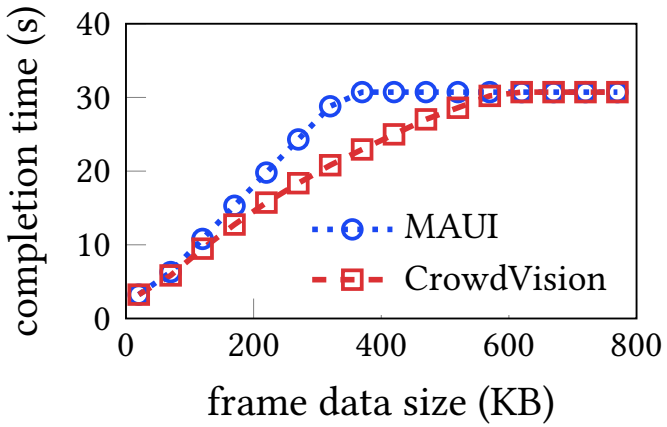
Optimization with Energy Constraint

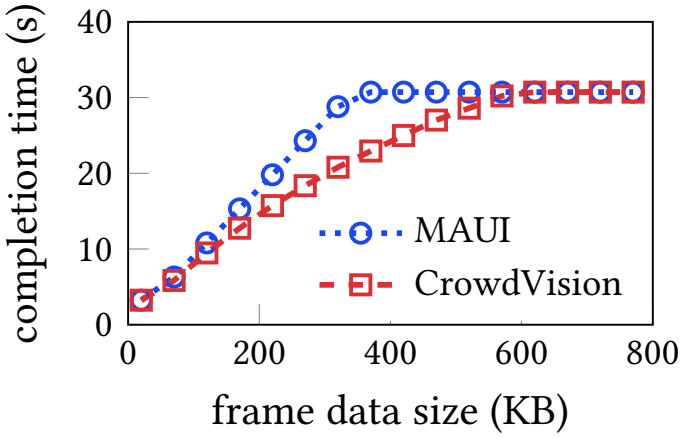
- Given $\mathbf{E}_{\min} \leq E' \leq \mathbf{E}_{\max}$, $T_{\min}(E')$**?**
  - Frame offloading ($\mathbf{E}_{\min}$) with extra energy
    - **Basic idea**: *using extra energy to perform detection so as to maximally reduce the completion time*
  - Local detection ($\mathbf{E}_{\min}$) with extra energy
    - Extra energy can be exploit to increase processing batches or offload frames
    - **Basic idea**: *efficiency* $\frac{\text{decrease of completion time}}{\text{energy cost}}$
  - Both cases are non-trivial

Performance

- *Processing under WiFi*

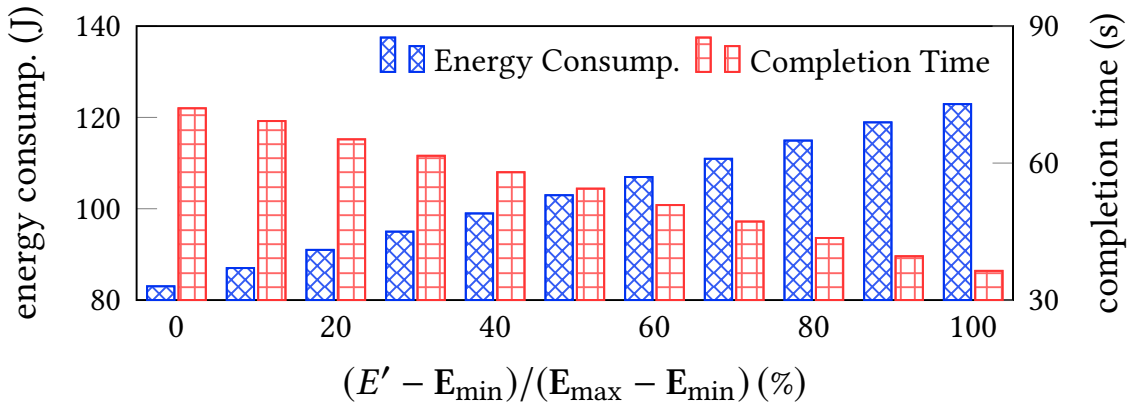


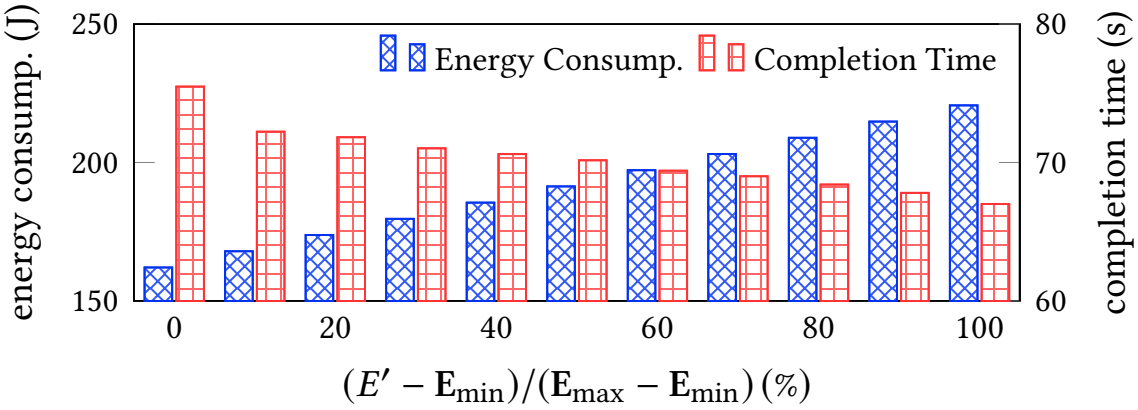




Performance

- *Processing under WiFi with energy constraints*





Performance

- *Processing under Cellular*