Towards Scalable Edge-Native Applications

Junjue Wang
Carnegie Mellon University
junjuew@cs.cmu.edu

Ziqiang Feng
Carnegie Mellon University
zf@cs.cmu.edu

Shilpa George
Carnegie Mellon University
shilpag@cs.cmu.edu

Roger Iyengar
Carnegie Mellon University
raiyenga@cs.cmu.edu

Padmanabhan Pillai
Intel Labs
padmanabhan.s.pillai@intel.com

Mahadev Satyanarayanan
Carnegie Mellon University
satya@cs.cmu.edu

About the Authors

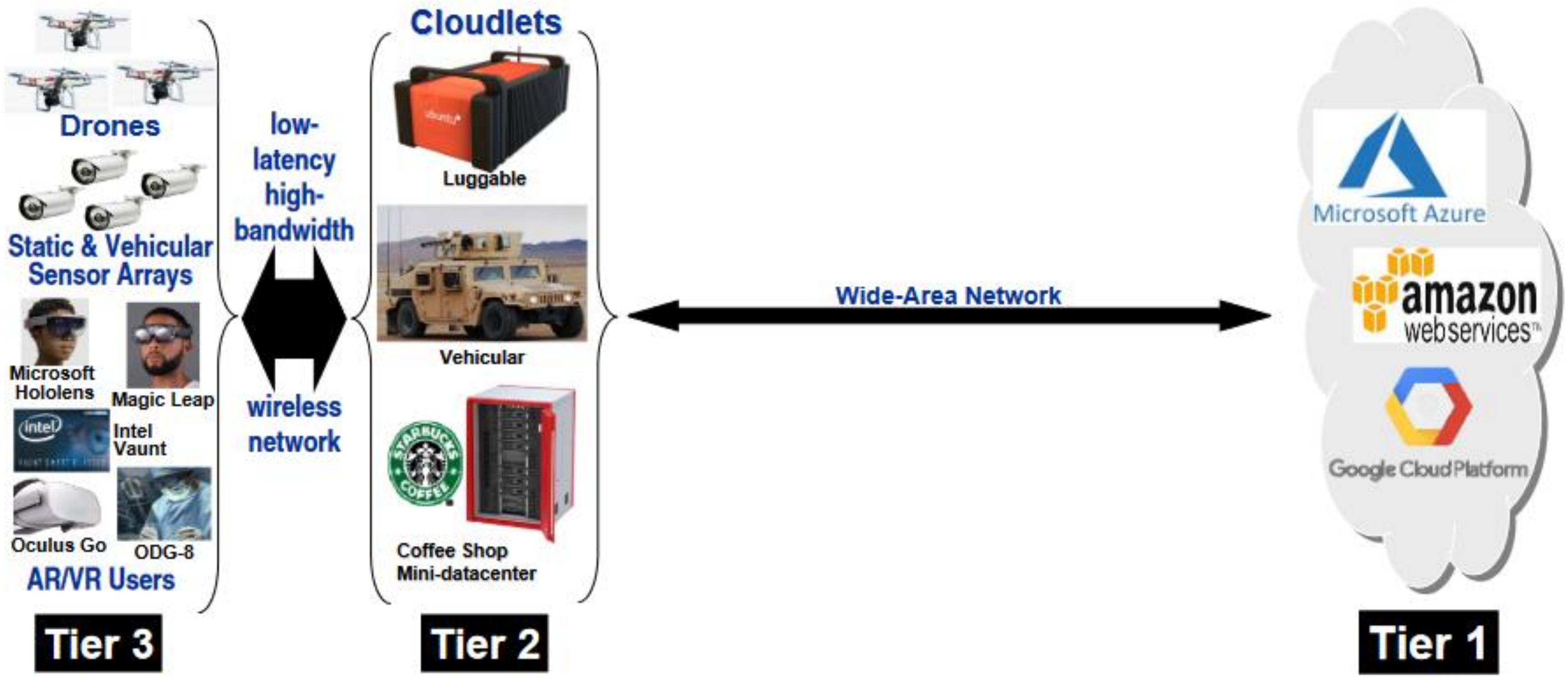
- Junjue Wang (Received PhD from Carnegie Mellon University)
  junjuew@cs.cmu.edu, https://junjuew.github.io/
- Ziqiang Feng (PhD candidate, Carnegie Mellon University)
  zf@cs.cmu.edu, https://fzqneo.github.io/
- Shilpa George (PhD candidate, Carnegie Mellon University)
  shilpag@cs.cmu.edu, https://a4anna.github.io/
- Roger Iyengar (PhD candidate, Carnegie Mellon University)
  raiyenga@cs.cmu.edu, https://rogeriyengar.com/
- Padmanabhan Pillai (Senior Research Engineer, Intel Labs)
  padmanabhan.s.pillai@intel.com,
  https://www.andrew.cmu.edu/user/pspillai/
- Mahadev Satyanarayanan (Carnegie Group Professor of Computer Science, Carnegie Mellon University)
  satya@cs.cmu.edu, https://www.cs.cmu.edu/~satya/

Overview

- **Background**
  - Edge Native
  - Scalable Gabriel
- Optimizations
  - Workload Reduction
    - Adaptive Sampling
    - IMU (Inertial measurement unit)-based Passive Phase Suppression
  - Resource Allocation
- Evaluation
  - Workload Reduction
  - Resource Allocation
  - Latency with both optimizations

Edge Native

- Unlike cloud ("Tier 1"), compute resources limited at the edge ("Tier 2")

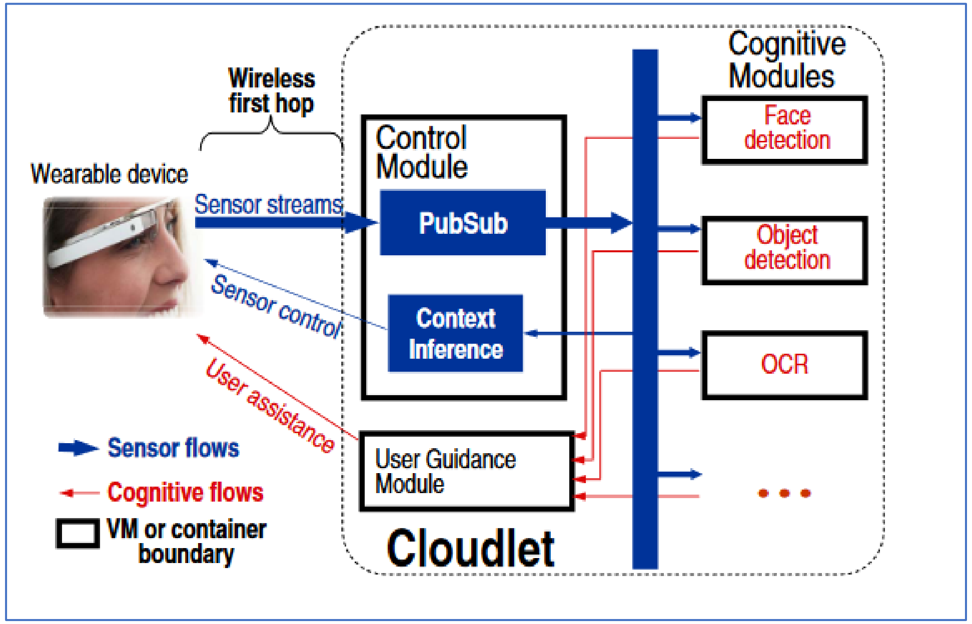


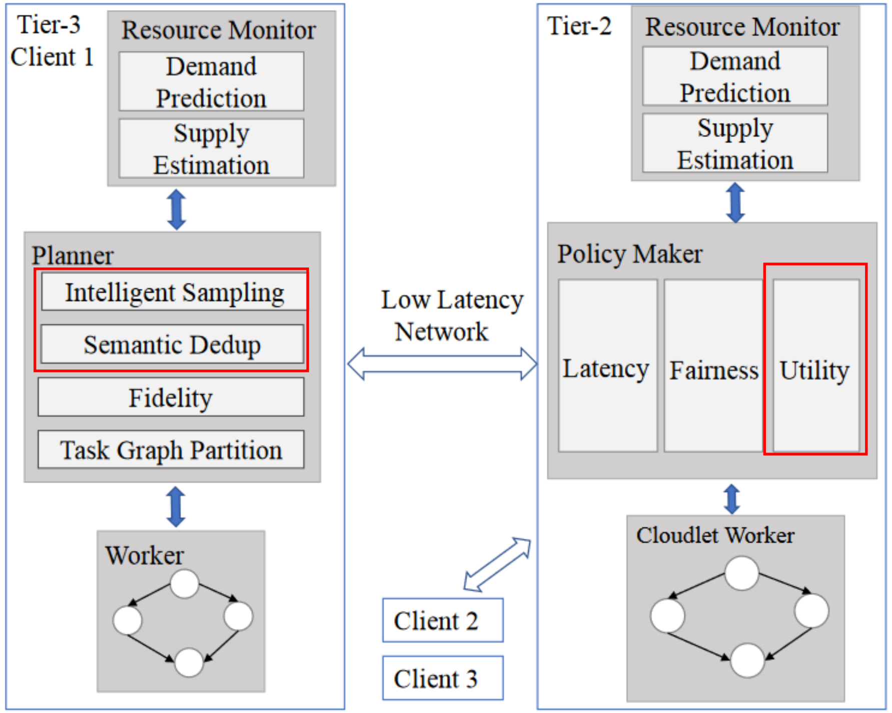
Edge Native

- Unlike cloud (“Tier 1”), compute resources limited at the edge (“Tier 2”)
  - Only 2 options to scale:
  1. Workload reduction: clients reduce the amount of data sent to edge servers
  2. Resource allocation: edge server favors important jobs
- Edge Native: application needs to support option 1
- Work reduction is application specific
- Focus on Wearable Cognitive Assistance:
  1. Large amount of data
  2. Latency requirement
  3. High compute requirement
     - Use GPUs on edge server for DNNs
  - Care about keeping latency (consistently) low

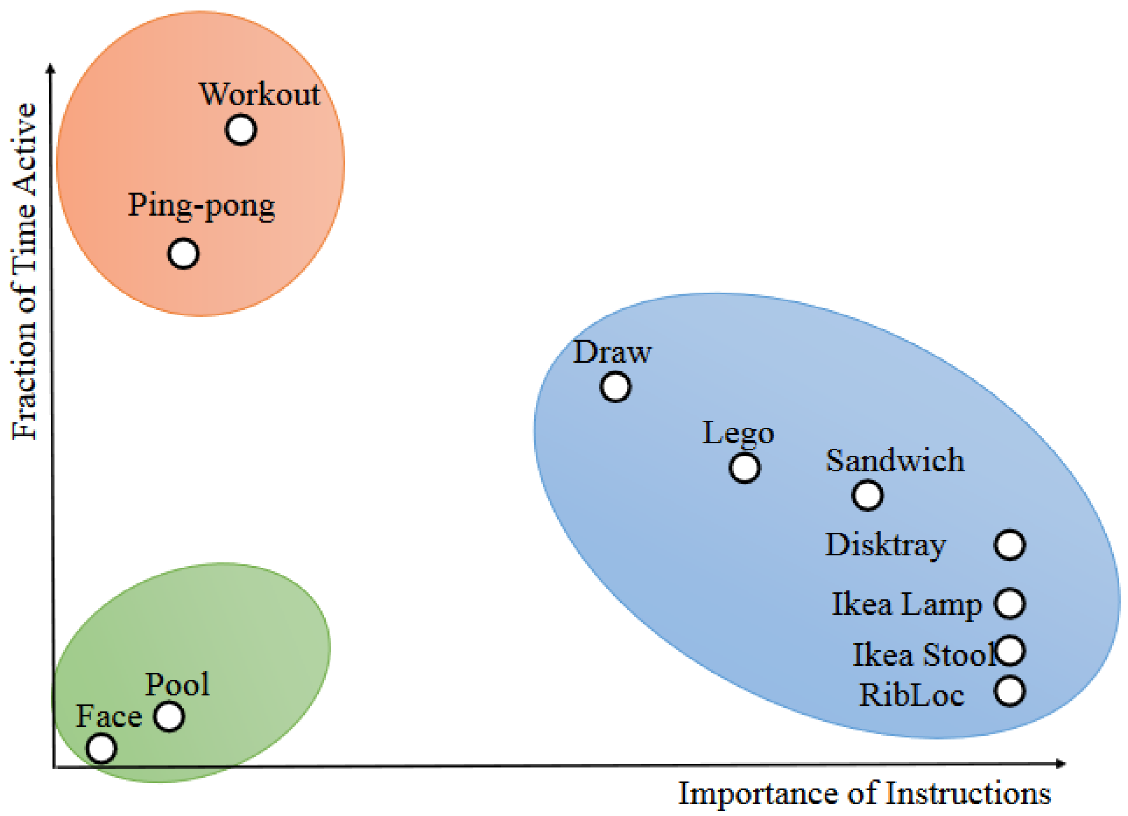
Scalable Gabriel

- Platform for Wearable Cognitive Assistance
- Gabriel: Single user
  - Client sends data to edge server
  - Edge server sends instructions to client
- Scalable Gabriel: Multi user
  - Resource monitors at client and server
  - Edge server Policy Maker module
    - Decides resource allocation
  - Client Planner module
    - Applies workload reduction





Gabriel Applications



	Question	Example	Load-reduction Technique
1	How often are instructions given, compared to task duration?	Instructions for each step in IKEA lamp assembly are rare compared to the total task time, e.g., 6 instructions over a 10 minute task.	Enable adaptive sampling based on active and passive phases.
2	Is intermittent processing of input frames sufficient for giving instructions?	Recognizing a face in any one frame is sufficient for whispering the person's name.	Select and process key frames.
3	Will a user wait for system responses before proceeding?	A first-time user of a medical device will pause until an instruction is received.	Select and process key frames.
4	Does the user have a pre-defined workspace in the scene?	Lego pieces are assembled on the lego board. Information outside the board can be safely ignored.	Focus processing attention on the region of interest.
5	Does the vision processing involve identifying and locating objects?	Identifying a toy lettuce for a toy sandwich.	Use tracking as cheap approximation for detection.
6	Are the vision processing algorithms insensitive to image resolution?	Many image classification DNNs limit resolutions to the size of their input layers.	Downscale sampled frames on device before transmission.
7	Can the vision processing algorithm trade off accuracy and computation?	In image classification, MobileNet is computationally cheaper than ResNet, but less accurate.	Change computation fidelity based on resource utilization.
8	Can IMUs be used to identify the start and end of user activities?	User's head movements are of significantly higher magnitude when searching for a Lego block.	Enable IMU-based frame suppression.
9	Is the Tier-3 device powerful enough to run parts of the processing pipeline?	A Jetson TX2 can run MobileNet-based image recognition in real-time.	Partition the vision pipeline between Tier-3 and Tier-2.

Applications have different properties and requirements

Applications provide Policy Maker description of some of its properties/requirements for resource allocation

Overview

- Background
  - Edge Native
  - Scalable Gabriel
- **Optimizations**
  - Workload Reduction
    - Adaptive Sampling
    - IMU (Inertial measurement unit)-based Passive Phase Suppression
  - Resource Allocation
- Evaluation
  - Workload Reduction
  - Resource Allocation
  - Latency with both optimizations

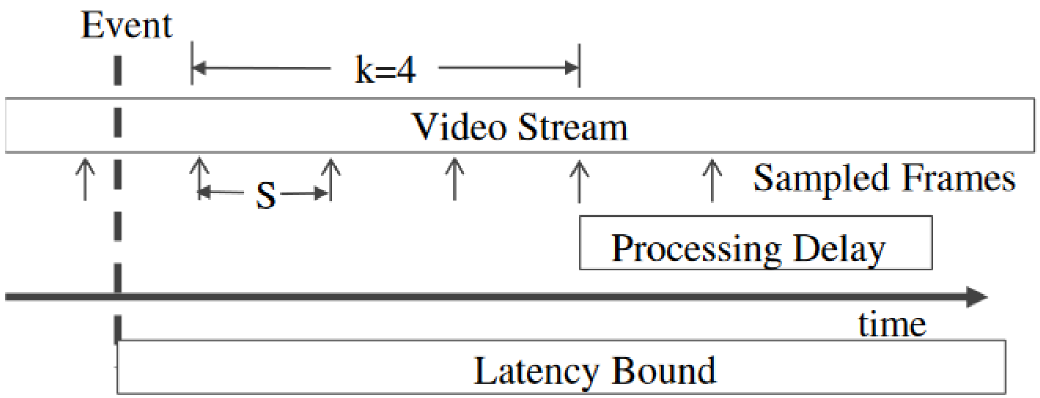
Adaptive Sampling

- Idea: Decrease sampling rate when user is acting on instruction
- Time to finish after instruction: Gaussian distribution from maximum likelihood estimation
  - Need data to find this
- At time t after sending an instruction, sampling rate (sr) is:

$$sr = min\_sr + \alpha * (max\_sr - min\_sr) * cdf\_Gaussian(t)$$

  - max_sr: constant
  - min_sr: minimum sampling rate that meets latency requirements
    - Depends on k frames in each sample (constant set empirically)
  - α: constant, determines how fast we return to active rate
  - cdf_Gaussian: probability user has finished by t

Adaptive Sampling

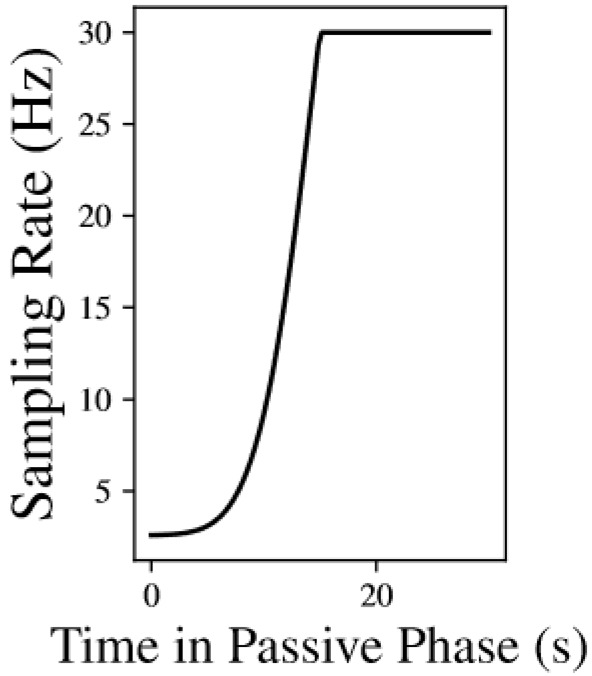


- At time t after sending an instruction, sampling rate (sr) is:

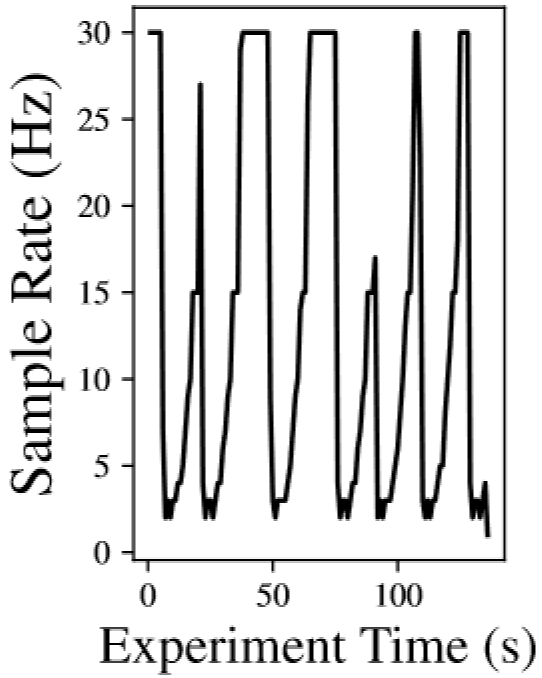
$$sr = min\_sr + \alpha * (max\_sr - min\_sr) * cdf\_Gaussian(t)$$

  - max_sr: constant
  - min_sr: minimum sampling rate that meets latency requirements
    - Depends on k frames in each sample (constant set empirically)
  - α: constant, determines how fast we return to active rate
  - cdf_Gaussian: probability user has finished by t

Adaptive Sampling



(a) Passive Sampling Rate



(b) Trace Sampling Rate

Adaptive sampling increases the sampling rate to the maximum during a passive phase

Trace	Sample Half Freq	Adaptive Sampling
1	50%	25%
2	50%	28%
3	50%	30%
4	50%	30%
5	50%	43%

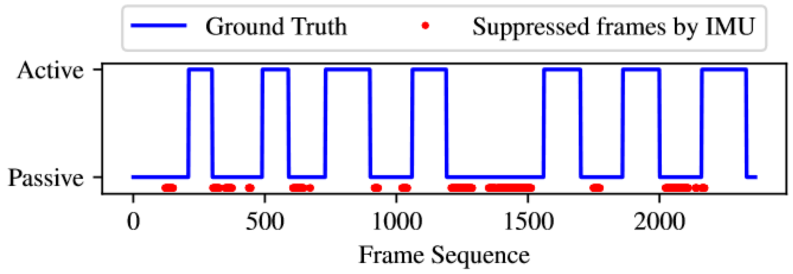
(a) Percentage of Frames Sampled

	Guidance Delay (frames±stddev)
Sample Half Freq	7.6 ± 6.9
Adaptive Sampling	5.9 ± 8.2

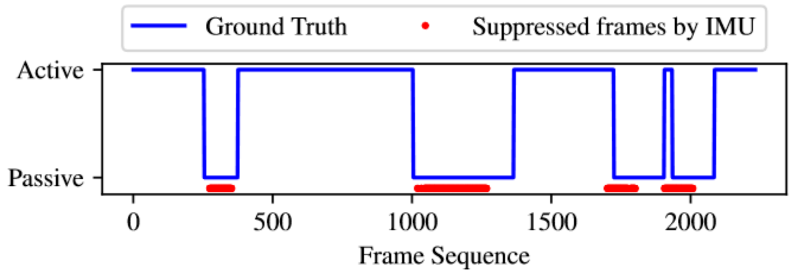
Adaptive sampling reduces latency and percentage of frames sampled on a trace of the LEGO application

IMU-based Passive Phase Suppression

- Idea: Don’t need to send frames to edge server when user is inactive
  - PING PONG: user not in a rally
  - LEGO: user looking for a piece
- 6 dimensions: 3 axes of rotation and 3 axes of acceleration
- SVM predicts active/passive state



(a) LEGO



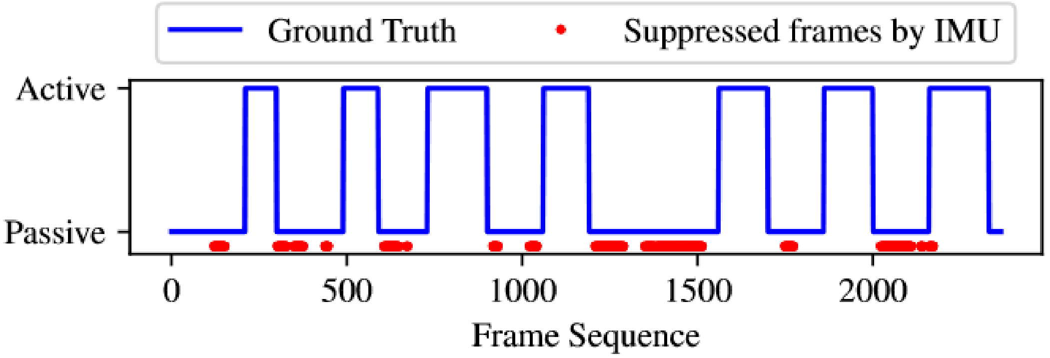
	Suppressed Passive Frames (%)	Max Delay of State Change Detection
Trace 1	17.9%	0
Trace 2	49.9%	0
Trace 3	27.1%	0
Trace 4	37.0%	0
Trace 5	34.1%	0

(a) LEGO

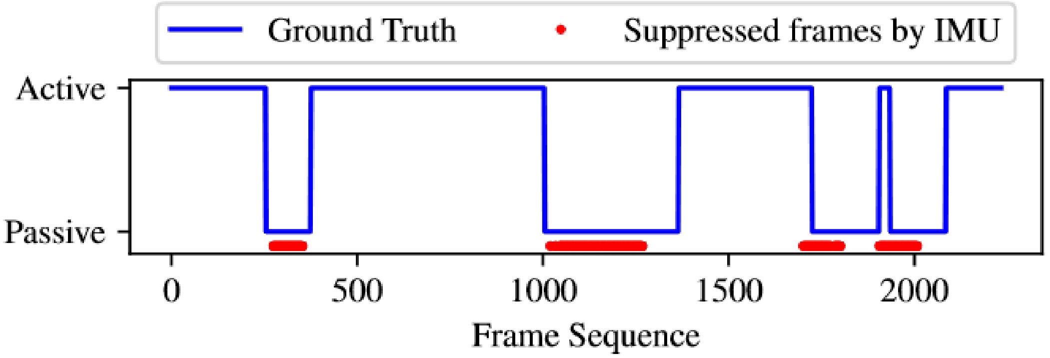
	Suppressed Passive Frames (%)	Loss of Active Frames (%)
Trace 1	21.5%	0.8%
Trace 2	30.0%	1.5%
Trace 3	26.2%	1.9%
Trace 4	29.8%	1.0%
Trace 5	38.4%	0.2%

(b) PING PONG

IMU-based Passive Phase Suppression



(a) LEGO



(b) PING PONG

Most of the suppressed frames are passive frames

	Suppressed Passive Frames (%)	Max Delay of State Change Detection
Trace 1	17.9%	0
Trace 2	49.9%	0
Trace 3	27.1%	0
Trace 4	37.0%	0
Trace 5	34.1%	0

(a) LEGO

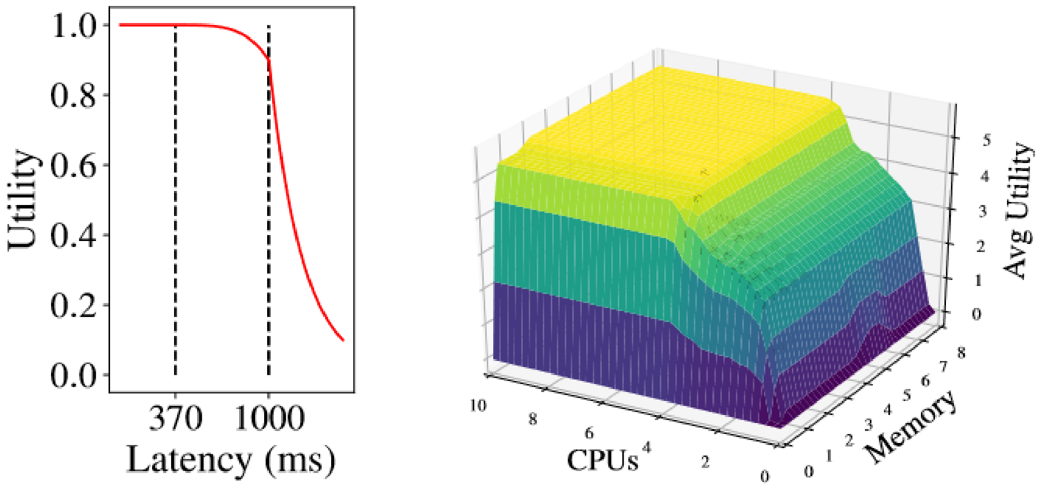
	Suppressed Passive Frames (%)	Loss of Active Frames (%)
Trace 1	21.5%	0.8%
Trace 2	30.0%	1.5%
Trace 3	26.2%	1.9%
Trace 4	29.8%	1.0%
Trace 5	38.4%	0.2%

(b) PING PONG

LEGO is unaffected and PING PONG loses 0-2% of active frames

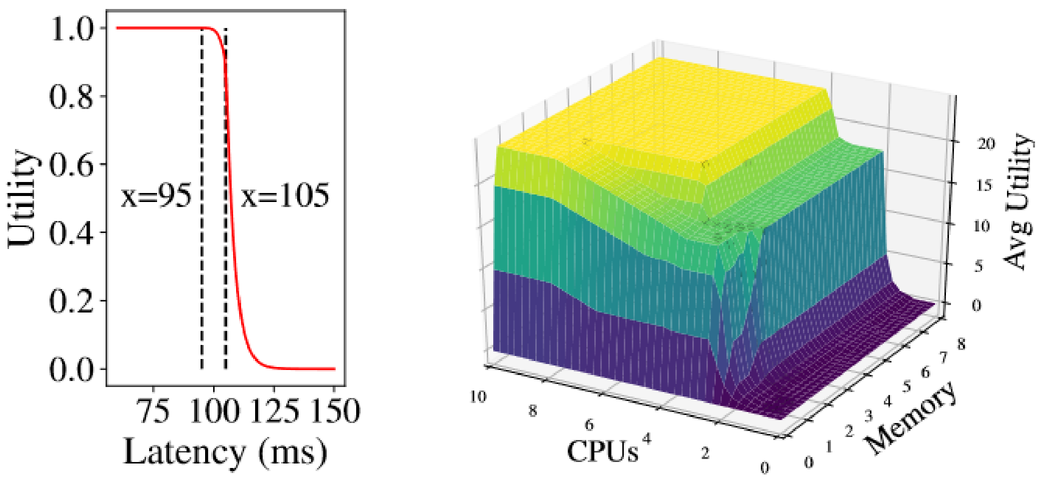
Resource allocation

- Idea: Maximize total utility (sum of utility for each application)
- Each application defines utility function in terms of system metrics (latency)
- Each frame has a utility in [0, 1]
- Profile application with different CPU and memory allocation
  - I think “Avg Utility” in Profiles has units utility per second



(a) Utility For FACE (b) Profile for FACE

Figure 10: FACE Application Utility and Profile



(a) Utility For POOL (b) Profile for POOL

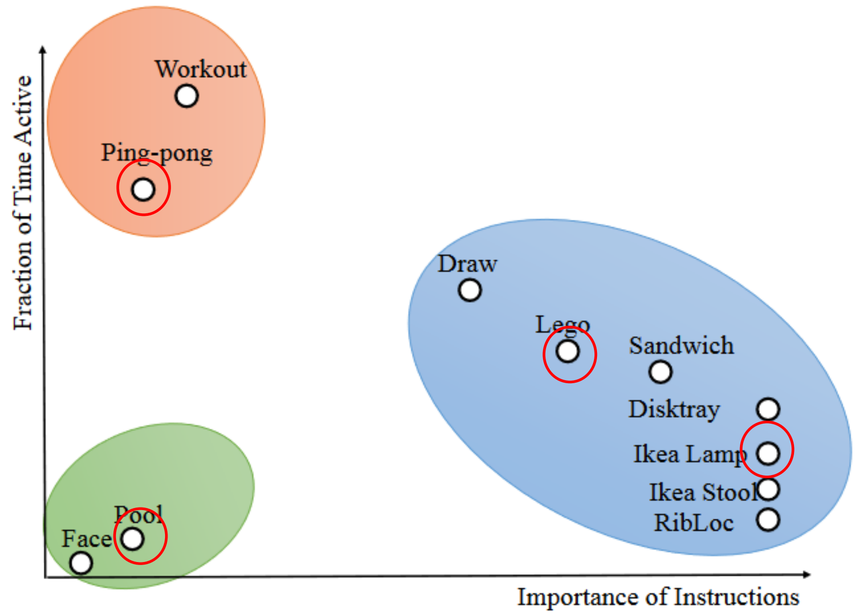
Figure 11: POOL Application Utility and Profile

Resource allocation

- a: an application in {FACE, LEGO, PING PONG, POOL, . . . },
- ua: utility of an application (from profile)
- ra: vector of resources for application
- r hat: vector of total resources
- ca: number of clients for application a
- ka: number of instances of application a
- γ: maximum utility per application, trades off fairness and total utility

$$
\begin{aligned}
\max_{\{k_a, \mathbf{r}_a\}} \quad & \sum_a k_a \cdot u_a(\mathbf{r}_a) \\
\text{s.t.} \quad & \sum_a k_a \cdot \mathbf{r}_a \preccurlyeq \hat{\mathbf{r}} \\
& 0 \preccurlyeq \mathbf{r}_a \quad \forall a \\
& k_a \cdot u_a(\mathbf{r}_a) \leq \gamma \cdot c_a \quad \forall a \\
& k_a \in \mathbb{Z}
\end{aligned}
$$

Evaluation



- 5 applications
  - FACE, PING PONG, LEGO, POOL, and IKEA
- Workload Reduction
  - 4 Nexus 6 mobile phone clients
  - PING PONG, LEGO, POOL
  - 2, 4, 6, and 8 cores on edge server
- Resource Allocation
  - 8 physical cores, 16GB memory for cloudlet resources
  - 15 to 40 clients
- Latency
  - 20 (4 clients per app), 30 (6 clients per app), and 40 (8 clients per app) clients
  - Pre-recorded video traces with random starting points

Evaluation: Workload Reduction

- Scalable Gabriel: Workload Reduction only
- Original Gabriel: Baseline
- Same number of active frames
- Original Gabriel receives more unnecessary passive frames

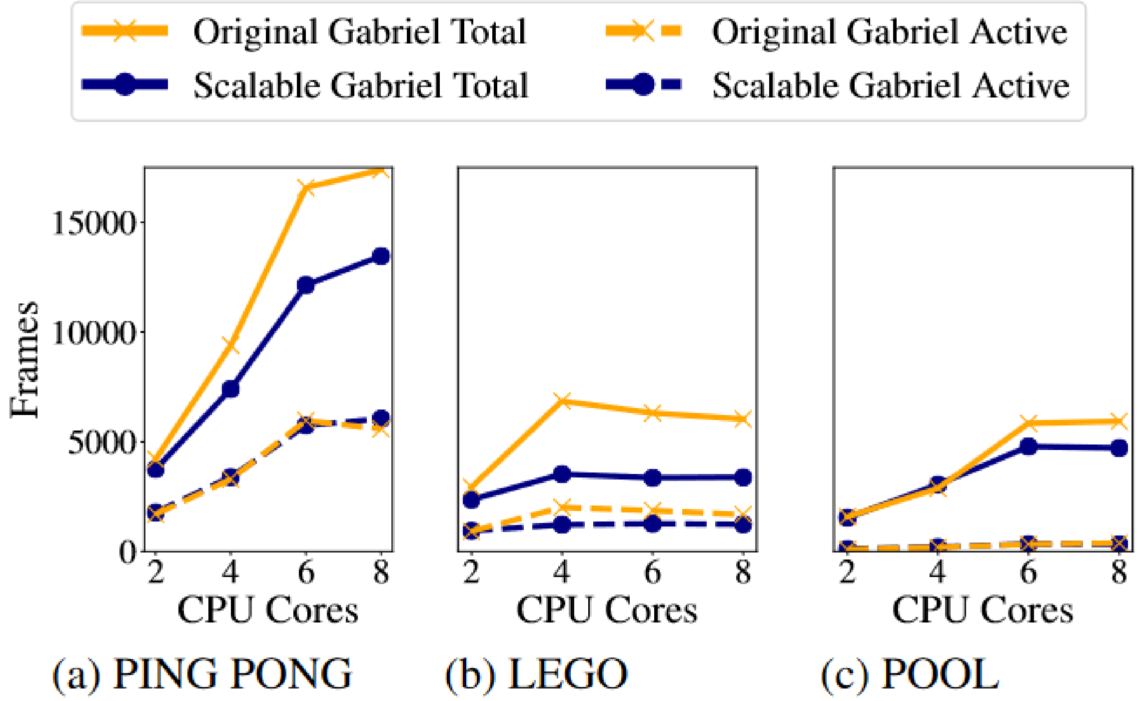
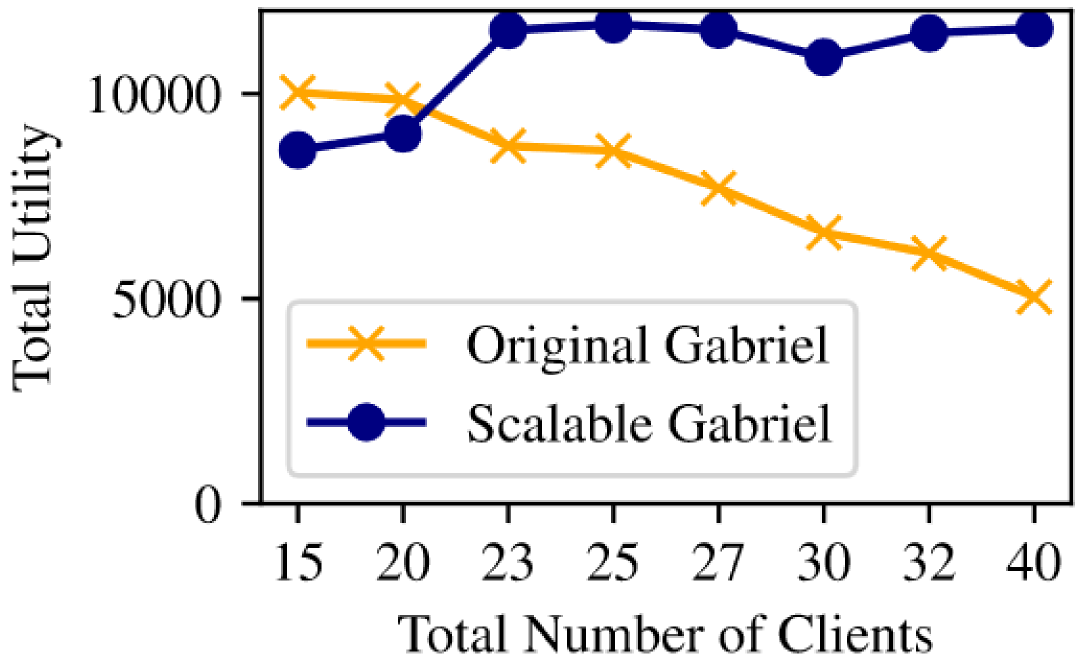


Figure 12: Effects of Workload Reduction

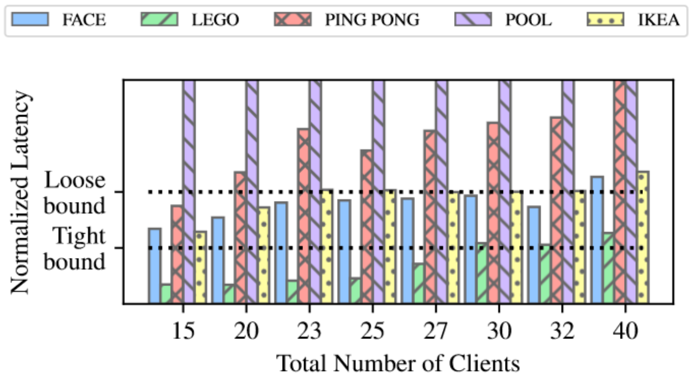
Evaluation: Resource Allocation

- Scalable Gabriel: Resource allocation only
- Original Gabriel: Baseline
- Utility of Scalable Gabriel not affected by increasing number of clients
    - Not clear why utility it starts off lower
- Original Gabriel drops to 40% of starting utility

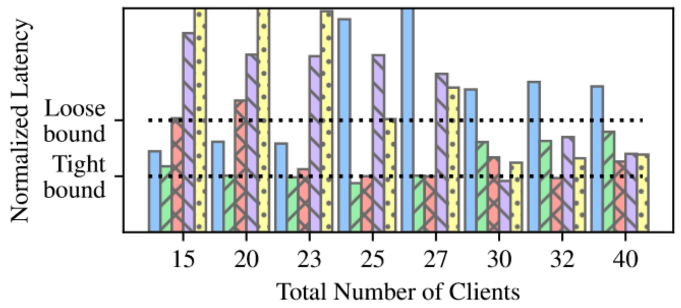


Evaluation: Resource Allocation

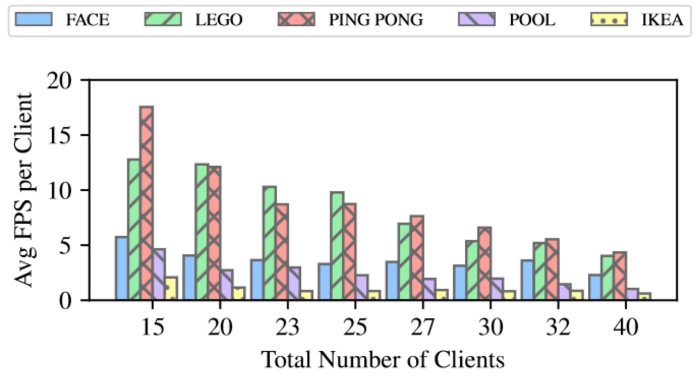
- Scalable Gabriel: Resource allocation only
- Original Gabriel: Baseline
- 90th percentile latency lower overall for Scalable Gabriel
- Scalable Gabriel able to prioritize high FPS for PING PONG and POOL with increasing number of clients



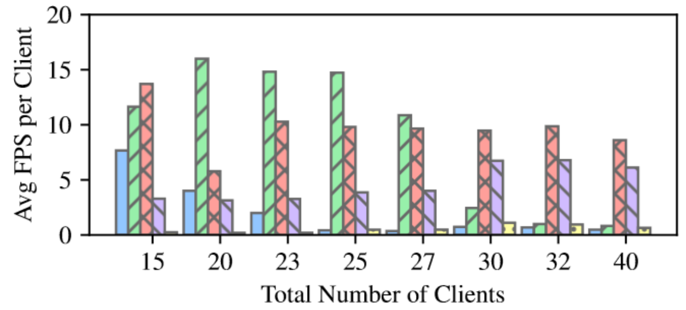
(a) Original Gabriel



(b) Scalable Gabriel



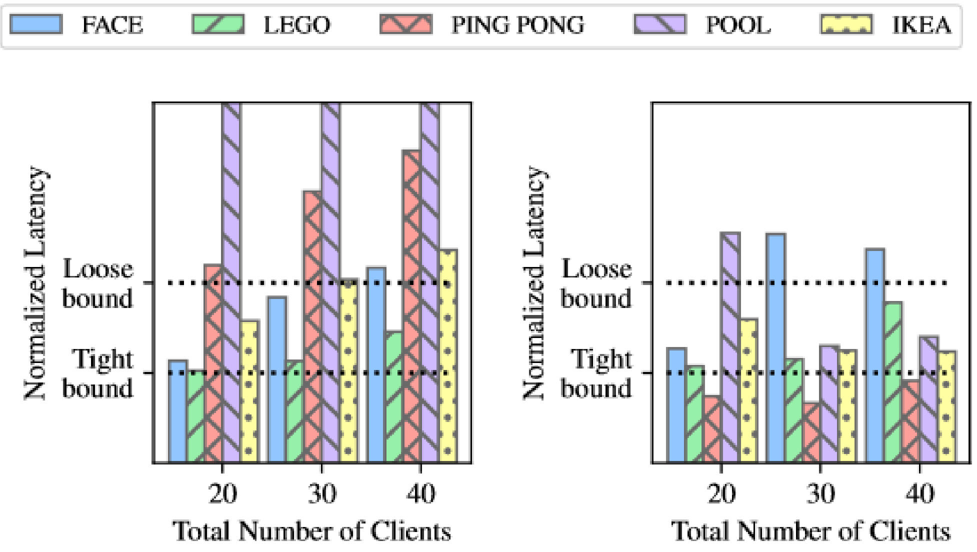
(a) Original Gabriel



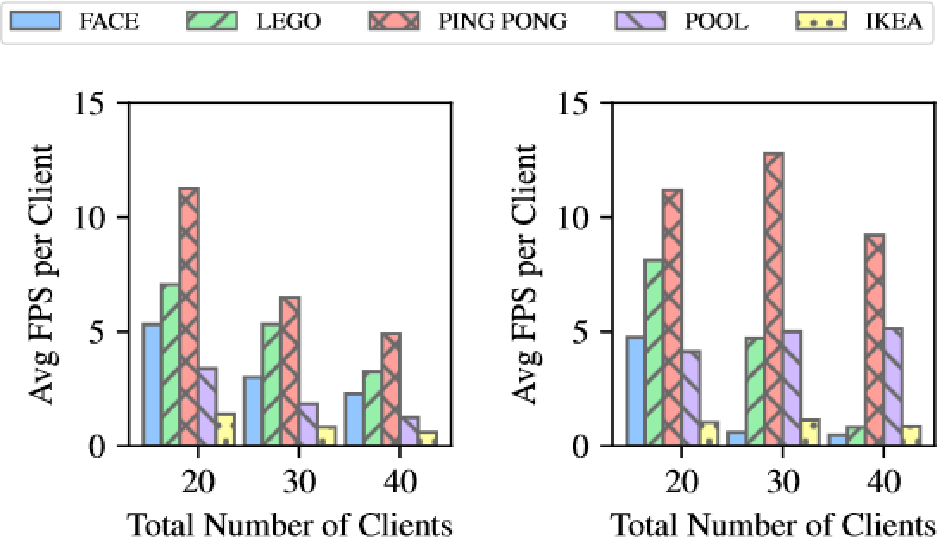
(b) Scalable Gabriel

Evaluation: Latency

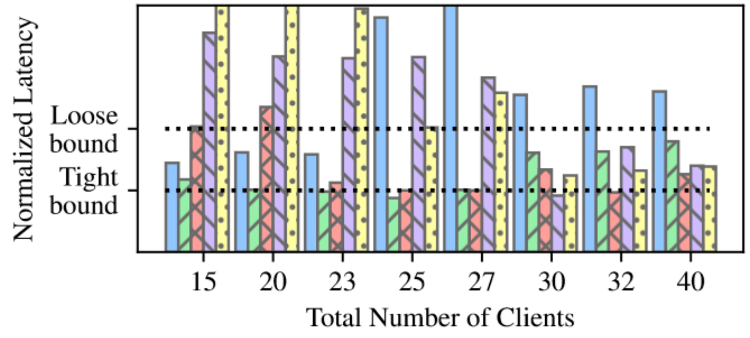
- Scalable Gabriel: Both Workload Reduction and Resource Allocation
- Original Gabriel: Baseline
- Using both workload and resource allocation better than just resource allocation (PING PONG 40 latency)



(a) Original Gabriel (b) Scalable Gabriel



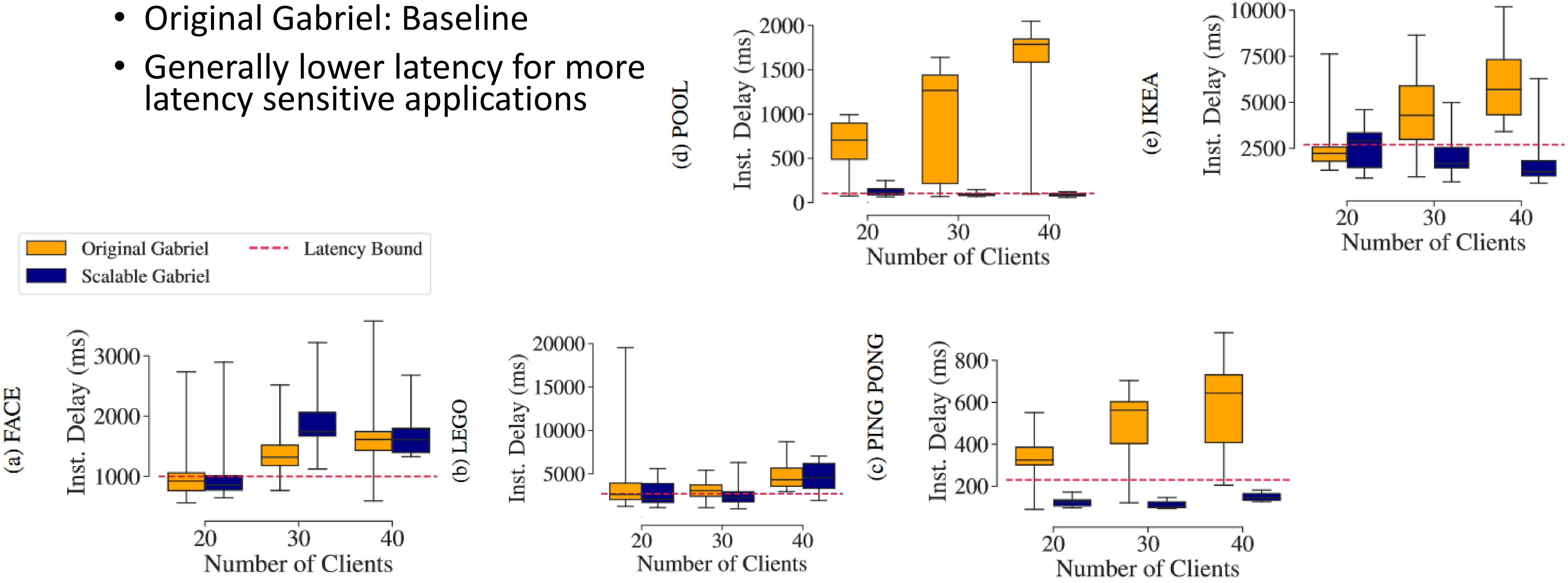
(a) Original Gabriel (b) Scalable Gabriel



(b) Scalable Gabriel (only resource allocation)

Evaluation: Latency

- Scalable Gabriel: Both Workload Reduction and Resource Allocation
- Original Gabriel: Baseline
- Generally lower latency for more latency sensitive applications



Positive/Negative Points

Positive

- Evaluated against baseline Gabriel using recorded video traces
- Strategy can be changed (can use different metric instead of total utility for resource allocation, fairness parameter gamma in utility)

Negative

- Relies on applications to provide a reasonable metric (ex. utility function)
- Not much evaluation of whether the loss of active frames in PING PONG affects results

Discussion

- Is there a simple way to relax the benevolent and cooperative assumption?
- How can we modify the system to prioritize more important applications?
- Information (e.g. the profile) needs to be sent to the cloudlet before running the application. Is this realistic?