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")

- Only 2 options:
  1. Reduce the amount of work given to edge servers
  2. Improve scheduling

- 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
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
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:

  \[ sr = \min_{sr} + \alpha \times (\max_{sr} - \min_{sr}) \times \text{cdf\_Gaussian}(t) \]

  • \( \max_{sr} \): constant
  • \( \min_{sr} \): minimum sampling rate that meets latency requirements
    • Depends on k frames in each sample (constant set empirically)
  • \( \alpha \): constant, determines how fast we return to active rate
  • \( \text{cdf\_Gaussian} \): probability user has finished by t
Adaptive Sampling

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

$$sr = \text{min}_sr + \alpha \times (\text{max}_sr - \text{min}_sr) \times \text{cdf}_\text{Gaussian}(t)$$

- $\text{max}_sr$: constant
- $\text{min}_sr$: minimum sampling rate that meets latency requirements
  - Depends on $k$ frames in each sample (constant set empirically)
- $\alpha$: constant, determines how fast we return to active rate
- $\text{cdf}_\text{Gaussian}$: probability user has finished by $t$
Adaptive sampling increases the sampling rate to the maximum during a passive phase.

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

(a) Percentage of Frames Sampled

<table>
<thead>
<tr>
<th>Trace</th>
<th>Sample Half Freq</th>
<th>Adaptive Sampling</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>50%</td>
<td>25%</td>
</tr>
<tr>
<td>2</td>
<td>50%</td>
<td>28%</td>
</tr>
<tr>
<td>3</td>
<td>50%</td>
<td>30%</td>
</tr>
<tr>
<td>4</td>
<td>50%</td>
<td>30%</td>
</tr>
<tr>
<td>5</td>
<td>50%</td>
<td>43%</td>
</tr>
</tbody>
</table>

Guidance Delay (frames±stddev)

<table>
<thead>
<tr>
<th>Sample Half Freq</th>
<th>Adaptive Sampling</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.6 ± 6.9</td>
<td>5.9 ± 8.2</td>
</tr>
</tbody>
</table>
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
IMU-based Passive Phase Suppression

Most of the suppressed frames are passive frames

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
Resource allocation

- \( a \): an application in \{FACE, LEGO, PING PONG, POOL, \ldots \},
- \( u_a \): utility of an application (from profile)
- \( r_a \): vector of resources for application
- \( \hat{r} \): vector of total resources
- \( c_a \): number of clients for application \( a \)
- \( k_a \): number of instances of application \( a \)
- \( \gamma \): maximum utility per application, trades off fairness and total utility

\[
\begin{align*}
\max_{\{k_a, r_a\}} & \quad \sum_a k_a \cdot u_a(r_a) \\
\text{s.t.} & \quad \sum_a k_a \cdot r_a \leq \hat{r} \\
& \quad 0 \leq r_a \quad \forall a \\
& \quad k_a \cdot u_a(r_a) \leq \gamma \cdot c_a \quad \forall a \\
& \quad k_a \in \mathbb{Z}
\end{align*}
\]
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
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
- 90\textsuperscript{th} percentile latency lower overall for Scalable Gabriel
- Scalable Gabriel able to prioritize high FPS for PING PONG and POOL with increasing number of clients
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)
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?