EdgeBatch: Towards AI-empowered Optimal Task Batching in Intelligent Edge Systems

Daniel (Yue) Zhang, Nathan Vance, Yang Zhang, Md Tahmid Rashid, Dong Wang
RTSS 2019
Artificial Intelligence (AI) is Changing Our Lives

- Recommender Systems
- Self-driving Cars
- Smart Robotics
- Natural Language Processing
AI and BigData in The Era of Internet of Things (IoT)

- Traditional BigData: born and stored at mega-scale datacenters
  - social media contents, business records, financial informatics, etc.

- The new BigData: generated from widespread end devices (IoT devices)
The Emergence of Edge Computing

- How we do process this massive amount of data?
  - **Cloud computing**: bandwidth, latency concerns
  - **Edge computing**: keeps data closer to where it is generated
Edge Intelligence – Edge Computing in the AI Era
Human-centric Edge Computing

*Where should the AI computations be performed?*

<table>
<thead>
<tr>
<th></th>
<th>Cooperative -ness</th>
<th>Latency</th>
<th>Bandwidth</th>
<th>Computing power</th>
<th>Power Supply</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Server</strong></td>
<td>Fully cooperative</td>
<td>High</td>
<td>High</td>
<td>Powerful</td>
<td>Power line</td>
</tr>
<tr>
<td><strong>Device</strong></td>
<td>Non-cooperative</td>
<td>Low</td>
<td>Low</td>
<td>Limited</td>
<td>Battery</td>
</tr>
</tbody>
</table>

Our Philosophy: Data born and live at the edge.
Why Privately-owned Devices?

- Devices are increasingly pervasive

<table>
<thead>
<tr>
<th>Year</th>
<th>World Population</th>
<th>Connected Devices</th>
<th>Connected Devices Per Person</th>
</tr>
</thead>
<tbody>
<tr>
<td>2003</td>
<td>6.3 Billion</td>
<td>500 Million</td>
<td>0.08</td>
</tr>
<tr>
<td>2010</td>
<td>6.8 Billion</td>
<td>12.5 Billion</td>
<td>1.84</td>
</tr>
<tr>
<td>2015</td>
<td>7.2 Billion</td>
<td>25 Billion</td>
<td>3.47</td>
</tr>
<tr>
<td>2020</td>
<td>7.6 Billion</td>
<td>50 Billion</td>
<td>6.58</td>
</tr>
</tbody>
</table>

- Devices are increasingly powerful
A Collaborative Edge Design

- Allow devices to be “socially interactive”
The Collaborative Edge Design

- **Edge devices are good at different tasks**
  - Vehicle: data collection, video analysis
  - Phone: image processing
  - ...

A crowd plate detection application example
Objectives

- **Objective 1 – Energy Minimization for Devices**
  - Battery is one of the most precious resource of devices
  - System design must be energy aware

- **Objective 2 – Quality of Service**
  - Delay, Deadline Hit Rate, Throughput
The Heterogeneity Challenge

- **Heterogenous Edge Devices:**
  - Heterogeneous hardware (w/ GPU, w/o GPU) and architecture (X86, ARM)
  - Heterogeneous execution environment (OS, AI framework)

- **Heterogenous Tasks:**
  - Sensing vs Computing
  - GPU intensive vs CPU intensive
  - GPU exclusive vs non-exclusive

How to make heterogeneous edge devices to collaboratively finish heterogeneous tasks?
We map the task mapping problem into a "supply chain" problem.

- Sensing data – Raw material
- Hardware components – Workers
- Edge nodes - Factories

![Supply Chain based Task Mapping Diagram]
Supply Chain Graph Modeling

- Supply Chain Graph models the possible supply chain path among devices.

- s1, s2 – virtual source nodes
- t – destination node
- ES – edge server
- A, B, C – end devices
Rational Edge Challenge

- What if the devices are non-cooperative?

I can do everything on my own!

I do not want to compute for you!

Can we make device owners to form supply chain themselves?
A Game-theoretic Task Trading Protocol

- If device A finishes all tasks by itself, the reward is $1 (the reward depends on how long the task finishes)
- If device A and B jointly finishes the task, the reward will be $2 for each

➢ A and B will collaborate even though they are rational and competitors.
The Task Batching Problem

- Task batching fully utilizes GPU
  - Instead of processing 1 by 1
  - $\text{batch\_size} = X$

![Diagram showing ideal and actual task batching]

**Ideal**

<table>
<thead>
<tr>
<th>Tasks</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time Slots</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>6</td>
<td>8</td>
<td>10</td>
</tr>
</tbody>
</table>

**Actual (unpredictable arrival time)**

<table>
<thead>
<tr>
<th>Tasks</th>
<th>A</th>
<th>BCD</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time Slots</td>
<td>0</td>
<td>2</td>
<td>6</td>
<td>10</td>
</tr>
</tbody>
</table>
Batch Size Trade-offs

- **Small batch size:**
  - Lower waiting time
  - Longer processing time
  - Lower peak energy

- **Large batch size:**
  - Long waiting time
  - Lower processing time
  - High peak energy

How to set the batch size on each device?

Figure: Delay and Energy Trade-off of Batch Size
Decompose The Delay Cost

- **L1**: delay of tasks that were left from previous batch
- **L2**: processing time of current batch
- **L3**: holding time for image to arrive

\[ W_{m}^{(D)} = \mathcal{L}_1 + \mathcal{L}_2 + \mathcal{L}_3 \]

\[ W_{m}^{(E)} = \sum_{\Gamma_m^{c}} g(0) \cdot H_m + \sum_{\Gamma_m} g(|B_m|) \cdot \mathcal{L}_2 \]

\[ \text{arg min}_{B_1, B_2, \ldots, B_M} W_{m}^{(D)} + \lambda \cdot W_{m}^{(E)}, 1 \leq m \leq M \]

\[ \text{s.t., } |B_m| \leq \Theta \]
Solution – Bus Waiting Problem

Algorithm 1: Online Regret Minimization

1: Input: weight vector \( W = \{w_1, w_2, ..., w_\Theta\} \), learning parameter \( \eta \), current batch index \( m \)
2: Output: updated weight vector \( W' = \{w'_1, w'_2, ..., w'_\Theta\} \)
3: for all \( t \in [0, T] \) do
4:   if \( t \) is a control point then
5:     for all \( i \in [1, \Theta] \) do
6:       Normalize \( p_i = \frac{\eta \cdot w_i}{\sum_{i=1}^{\Theta} (\eta \cdot w_i)} \)
7:     end for
8:     Update \( R_B |_{m-1} = R_B |_{m-1} \cdot p_i \)
9:   for all \( i \in [1, \Theta] \) do
10:      \( w'_i = (w_i \cdot (1 + \eta \cdot R_i))^{\frac{\eta}{\eta-1}} \)
11:   end for
12: end if
13: end for
14: Return \( W' \)
### System Implementation

#### Specifications of Edge Devices

<table>
<thead>
<tr>
<th>Device Type</th>
<th>CPU</th>
<th>GPU</th>
<th>Memory</th>
<th>OS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pi3</td>
<td>1.2 GHz quad-core ARM Cortex-A53</td>
<td>N/A</td>
<td>1GB LPDDR2</td>
<td>Raspbian</td>
</tr>
<tr>
<td>TX2</td>
<td>2.0 GHz ARM quad-core Cortex-A57</td>
<td>256-core NVIDA Pascal</td>
<td>8GB LPDDR4</td>
<td>Linux (Ubuntu)</td>
</tr>
<tr>
<td>TK1</td>
<td>2.32 GHz ARM quad-core Cortex-A15</td>
<td>192-core NVIDA Kepler</td>
<td>2GB LPDDR3</td>
<td>Linux (Ubuntu)</td>
</tr>
<tr>
<td>PC</td>
<td>2.7 GHz i5-7500T quad-core</td>
<td>N/A</td>
<td>8GB LPDDR4</td>
<td>Windows 10</td>
</tr>
</tbody>
</table>

**Heterogenous Embedded Devices**

**Energy Module**
Evaluation Results

Our Solution

- **Achieved**: 42% decrease in end-to-end delay.

Our solution achieves significantly lower delay and good energy efficiency.

```
<table>
<thead>
<tr>
<th>Scheme</th>
<th>TX2</th>
<th>TX1</th>
<th>TK1</th>
<th>Pi3</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td>EdgeBatch</td>
<td>0.802</td>
<td>0.781</td>
<td>0.783</td>
<td>0.633</td>
<td>8.470</td>
</tr>
<tr>
<td>BGTA + NB</td>
<td>0.893</td>
<td>0.875</td>
<td>0.879</td>
<td>0.675</td>
<td>9.344</td>
</tr>
<tr>
<td>BGTA + FS</td>
<td>0.874</td>
<td>0.865</td>
<td>0.855</td>
<td>0.675</td>
<td>9.238</td>
</tr>
<tr>
<td>BGTA + FP</td>
<td>0.852</td>
<td>0.866</td>
<td>0.847</td>
<td>0.675</td>
<td>9.180</td>
</tr>
<tr>
<td>BGTA + BW</td>
<td>0.887</td>
<td>0.858</td>
<td>0.840</td>
<td>0.675</td>
<td>9.221</td>
</tr>
<tr>
<td>BGTA + OL</td>
<td>0.824</td>
<td>0.827</td>
<td>0.829</td>
<td>0.675</td>
<td>9.011</td>
</tr>
<tr>
<td>TDA + NB</td>
<td>0.923</td>
<td>0.903</td>
<td>0.803</td>
<td>0.553</td>
<td>8.576</td>
</tr>
<tr>
<td>TDA + FS</td>
<td>0.885</td>
<td>0.864</td>
<td>0.755</td>
<td>0.553</td>
<td>8.326</td>
</tr>
<tr>
<td>TDA + FP</td>
<td>0.857</td>
<td>0.841</td>
<td>0.724</td>
<td>0.553</td>
<td>8.162</td>
</tr>
<tr>
<td>TDA + BW</td>
<td>0.862</td>
<td>0.852</td>
<td>0.748</td>
<td>0.553</td>
<td>8.242</td>
</tr>
<tr>
<td>TDA + OL</td>
<td>0.833</td>
<td>0.839</td>
<td>0.719</td>
<td>0.553</td>
<td>8.100</td>
</tr>
<tr>
<td>CoGTA + NB</td>
<td>0.906</td>
<td>0.880</td>
<td>0.843</td>
<td>0.682</td>
<td>9.350</td>
</tr>
<tr>
<td>CoGTA + FS</td>
<td>0.875</td>
<td>0.853</td>
<td>0.851</td>
<td>0.682</td>
<td>9.251</td>
</tr>
<tr>
<td>CoGTA + FP</td>
<td>0.837</td>
<td>0.832</td>
<td>0.819</td>
<td>0.682</td>
<td>9.068</td>
</tr>
<tr>
<td>CoGTA + BW</td>
<td>0.842</td>
<td>0.866</td>
<td>0.830</td>
<td>0.682</td>
<td>9.168</td>
</tr>
<tr>
<td>CoGTA + OL</td>
<td>0.822</td>
<td>0.817</td>
<td>0.828</td>
<td>0.682</td>
<td>9.026</td>
</tr>
</tbody>
</table>
```

“Overall” is the sum of normalized energy consumption of all edge nodes.
Conclusion

- Introduced the collaborative edge design for edge AI
- Present a optimal batching algorithm that identifies the ideal batch size with unpredictable task arrival time
- Introduced a supply chain model for peer-offloading tasks among heterogeneous devices
Social Sensing Lab @ ND
https://www3.nd.edu/~sslab/

Thanks!

Yue (Daniel) Zhang
University of Notre Dame