Dynamic and Scalable Deployment of Edge Internet-of-Things Analytics

蔡霈萱 Pei-Hsuan Tsai
Outline

- Introduction and Motivation
- Dynamic Deployment
- Edge Analytics
- System Overview
- Implementation and Demo Scenarios
- Evaluations
- Related Works
- Conclusion and Future Work
Introduction and Motivation
Motivation

- Internet of Things (IoT) grows rapidly
- Produce incredible amount of data
  - Overload the data centers and congest the networks seriously

<table>
<thead>
<tr>
<th>Category</th>
<th>2016</th>
<th>2017</th>
<th>2018</th>
<th>2020</th>
</tr>
</thead>
<tbody>
<tr>
<td>Consumer</td>
<td>3,963.0</td>
<td>5,244.3</td>
<td>7,036.3</td>
<td>12,863.0</td>
</tr>
<tr>
<td>Business: Cross-Industry</td>
<td>1,102.1</td>
<td>1,501.0</td>
<td>2,132.6</td>
<td>4,381.4</td>
</tr>
<tr>
<td>Business: Vertical-Specific</td>
<td>1,316.6</td>
<td>1,635.4</td>
<td>2,027.7</td>
<td>3,171.0</td>
</tr>
<tr>
<td>Grand Total</td>
<td>6,381.8</td>
<td>8,380.6</td>
<td>11,196.6</td>
<td>20,415.4</td>
</tr>
</tbody>
</table>

Source: Gartner (January 2017)
Limitations of Current Solution

Analyze and Compute

Huge Amount of Data

in Data center
Edge Analytics - Pre-processing

- Reduce latency
- Reduce network traffic
- Reduce the load of data centers
Fog Computing

- Fog computing leverages devices in data centers, edge networks, and end devices in simultaneously
Advantages: Fog >> Cloud

- Diverse kinds of resources
  - Computations, communications, storage, and sensors
- Utilize wasted resources
- Reduce network traffic
- Short response time
- Low cost
- Low carbon footprint
- ...

...
Shooter Tracking Usage Scenario

- Sound recognition
- Face recognition
- Path estimation
Dynamic Deployment
Dynamic Deployment Mechanism

- Frequently updating or replacing the applications
  - Container-based applications
- Managing lots of fog devices and applications
  - Orchestration tool
- Triggering another application when something happened
  - Event-driven mechanism
Virtualization Technology

- Virtualized modules
  - Dynamically placed on the fog devices
  - Migrated among the fog devices
  - Allocated the resource on-demand
  - More private

- Traditional virtual machine v.s. container
  - Xen, KVM
  - LXC, Docker
Traditional VM v.s Container

- **Container**
  - Share the same OS kernel, and use the namespaces to distinguish one from another

- **Traditional Virtual Machine**
  - Need large storage space and more computing power

[Diagram showing the comparison between VM and Container]

Container-based Applications

- Lightweight
  - Quick start
  - Easy to replace the configuration of the applications

<table>
<thead>
<tr>
<th></th>
<th>Virtual Machine</th>
<th>Container</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size</td>
<td>GB</td>
<td>MB</td>
</tr>
<tr>
<td>Startup</td>
<td>Minute</td>
<td>Second</td>
</tr>
</tbody>
</table>
Orchestration Tools

- **SaltStack**
  - Remote execution tool and configuration management system

- **OpenStack**
  - Used to manage virtual machines in data centers

- **Swarm**
  - Native clustering system for Docker

- **Kubernetes**
  - Automating deployment, scaling, and management of containerized applications
Kubernetes Architecture

- Each fog device hosts several containers, can be assembled into pod
- A service is a group of pods that are running on the cluster
Resource Monitoring

- Fog devices situation monitoring
  - Resource usage (CPU, Memory)
  - Containers status

Providing the important information for deployment strategy.
Event-driven Mechanism

- Allows developers to make the logical rules that automatically deploy another application when a specific event is triggered
  - Motion detected $\rightarrow$ Capture an image
Edge Analytics
Requirement of Edge IoT Analytics

- Location-based and sensor-based services
  - Tag the devices
- Raw sensor data are huge
  - Deep learning
- Resource-constrained fog computing devices
  - Distributed computing
Tag the Devices

Location A
- Resource Usage: 79%
- Resource Usage: 29%
- Resource Usage: 5%

Location B
- Resource Usage: 11%
- Resource Usage: 36%
- Resource Usage: 44%
Pre-processing with Deep Learning

- Tensorflow
  - An open-source software library for Machine Intelligence
  - Data flow graphs
    - Nodes - mathematical operations
    - Edges - multidimensional data arrays (tensors)
Distributed Computing

- Collecting the resource from several heterogeneous fog devices
System Overview
Programming Model

Device

Container

Operator

TensorFlow

Kubernetes

Multi-operator Application
System Overview
Master

- User Interface
- Operator deployment algorithm
  - Decide deploying which operators on which minions
- Device manager
  - Collect crucial device status
- Deployment manager
  - Launch specific Docker images on chosen minions
- Registry
  - Images are stored in the registry at the server
Minions

- TensorFlow-enabled container
  - Docker containers including TensorFlow and its analytic libraries
- k8s agent
  - Monitor and report the status of minions and pod to device manager
- Local Registry
Algorithm - System Model Derivation

Requests
- IoT Analytics
- Required QoS
- Required Sensors
- Required Locations

System Models

Resource Capabilities of Fog Devices

Deployment Decisions

Implementations and Demo Scenarios
Fog Computing Platform

- Auto-deployment
  - Docker (container virtualization)
  - Kubernetes (deployment, resource monitoring)
  - Deployment Algorithm
- IoT edge analytics
  - TensorFlow
# User Interfaces

![User Interface Diagram]

## Lab Conditions

<table>
<thead>
<tr>
<th>Object Detection</th>
<th>Intrusion Detection</th>
<th>Sound Classification</th>
<th>Air Pollution Monitor</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Object: car)</td>
<td>(Sense value: 1)</td>
<td>(no value received)</td>
<td>(no value received)</td>
</tr>
<tr>
<td>![Car Image]</td>
<td>Nothing</td>
<td>Unknown</td>
<td>Unknown</td>
</tr>
<tr>
<td></td>
<td>Nothing</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Motion detected</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MQ5</th>
<th>MQ7</th>
<th>MQ131</th>
<th>MQ135</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Sensor value: 70)</td>
<td>(Sensor value: 281)</td>
<td>(Sensor value: 822)</td>
<td>(Sensor value: 323)</td>
</tr>
<tr>
<td>![MQ5 Sensor Chart]</td>
<td>![MQ7 Sensor Chart]</td>
<td>![MQ131 Sensor Chart]</td>
<td>![MQ135 Sensor Chart]</td>
</tr>
</tbody>
</table>
### Experiment Setup

- **Master**
  - i5 CPU PC installed with Kubernetes

- **Fog Devices**
  - 5 Intel PC (1.8 GHz 8-core i7 CPUs)
  - 5 Raspberry Pi (1.2 GHz 4-core ARM CPUs)

- **Bandwidth throttle: Wonder Shaper**
  - 8 Mbps (close to common WiFi bandwidth)
Experiment Setup

- Run applications using multiple threads to fully utilize the fog devices
- Add a queue between any two adjacent operators to increase the overall performance
- Run each experiment 5 times and present the average results
3 Applications

- Air quality monitor
- Sound classification
- Object detection
3 Applications

- **Air Quality Monitor**
- **Sound Classification**
- **Object Detection**
Scenario

Subscribe the event from broker, then determine it as an intrusion.

Deploy environment monitor app on every street light.

Ask Kubernetes to launch the surveillance service.

Capture an image, and publish it.

Publish the motion detection.

Subscribe the image, and then do the analyze.

detection app.

Deploy environment monitor app.

Deploy an object detection app.

Deploy an image capture app.

Object Detection

Image Capture

Environment Monitor

Node

Node

car

car

Alerter

MQTT Broker

Kubernetes
Evaluations
Dynamic Deployment
Container Overhead

- Setup: with and without Docker
  - Overhead caused by Docker virtualization on Raspberry Pi less than 5%
Efficiency of Deployment - Algorithm

- Update the system models online to iteratively derive the customized system models
- Run the experiments for 15 iterations, and update our system models after each iteration
- Generate a large number of requests, and execute our algorithm to deploy as many requests as possible

Efficiency of Deployment

- Deployed almost instantly in our platform
  - Operators take less than 20 seconds on average to be deployed

![Deployment Time Chart](chart.png)
Efficiency of Deployment

- When a large number of containers are simultaneously created on the same device, the I/O overhead is significantly increased.
Event-driven in Short Time

- Need 32.9 secs to finish the whole object detection scenario
- Only need 4.8 secs to trigger the new application

<table>
<thead>
<tr>
<th>Event</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>(scale box) PIR sensor send the value to MQTT</td>
<td>0</td>
</tr>
<tr>
<td>(monitor) Define it as a intrusion</td>
<td>0.0076</td>
</tr>
<tr>
<td>(monitor) Ask master deploy surveillance application</td>
<td>0.0108</td>
</tr>
<tr>
<td>(master) Start the yolo container (object detection)</td>
<td>4.3236</td>
</tr>
<tr>
<td>(master) Start the capture container (capture)</td>
<td>4.8292</td>
</tr>
<tr>
<td>(capture) Capture the image</td>
<td>4.9862</td>
</tr>
<tr>
<td>(capture) Publish the image to MQTT broker</td>
<td>7.7692</td>
</tr>
<tr>
<td>(yolo) Start to subscribe the image from broker</td>
<td>8.0892</td>
</tr>
<tr>
<td>(yolo) Receive the image and start to analyze</td>
<td>18.6576</td>
</tr>
<tr>
<td>(yolo) Finish the analyze</td>
<td>32.9148</td>
</tr>
<tr>
<td>(master) Get the detect result</td>
<td>32.9228</td>
</tr>
</tbody>
</table>
Edge Analytics
TensorFlow Achieves Low Collaboration Overhead

- Setup: run object detection without threading and container
  - Overhead adding one more device leads to only up to 10% overhead.
Benefits from Distributed Analytics

- Setup: run object recognizer on two fog devices (different cutting points)
- For heavy analytics applications (Object detection), distributed analytics results in large improvements
Benefits from Distributed Analytics

- Does not result in better performance when application’s analytics is quite simple (Air Pollution)
Different Service Quality Caused By Different Cutting Points

- Setup: 8 different cut points for object detection app.
  - Cutting into smaller operators with equal complexity results in the best performance

---

**Graphs:**
- **Left Graph:**
  - X-axis: Number of processed images/minute
  - Y-axis: CPU Usage (in percentages)
  - Bars for processed images and line graph for CPU usage,
    - Peaks occur around cut points 3 and 6.

- **Right Graph:**
  - X-axis: Cut Points
  - Y-axis: CPU Usage (%)
  - Line graphs for Device 1 and Device 2,
    - Device 1 shows a sharp drop after cut point 6.
    - Device 2 maintains a moderate level.
Different Service Quality Caused By Different Cutting Points

- When network resources is the bottleneck, we may not prefer equally-loaded splitting decisions

---
Related Works
# Related Works

<table>
<thead>
<tr>
<th></th>
<th>Support Heterogeneous Devices</th>
<th>Dynamic Deployment</th>
<th>Event-trigger</th>
<th>Pre-processing</th>
<th>Deep Learning</th>
<th>Distributed Computing</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Our Platform</strong></td>
<td>✅</td>
<td>✅</td>
<td>✅</td>
<td>✅</td>
<td>✅</td>
<td>✅</td>
</tr>
<tr>
<td><strong>AWS Greengrass</strong></td>
<td>✅</td>
<td></td>
<td>✅</td>
<td></td>
<td>✅</td>
<td></td>
</tr>
<tr>
<td><strong>IBM Watson</strong></td>
<td>✅</td>
<td>✅</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Azure IoT Suite</strong></td>
<td>✅</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>AT&amp;T IoT Platform</strong></td>
<td>✅</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Conclusion and Future Work
Demo Video
Conclusion

- Implementing a platform and programming model for IoT edge analytics
  - Dynamic Deployment → Docker, Kubernetes
  - Edge Analytics → TensorFlow
- Build a real testbed to evaluate and show the practicality and efficiency of my platform
  - Better performance of distributed analytics
  - Low overhead caused by Docker and TensorFlow
  - Tradeoff of different cut points
Future Work

- A complete eco-system for IoT
Q&A

