TinyEdge: Enabling Rapid Edge System Customization for IoT Applications

Wenzhao Zhang, Yuxuan Zhang, Hongchang Fan, Yi Gao, Wei Dong, Jinfeng Wang

Zhejiang University
Alibaba-Zhejiang University Joint Institute of Frontier Technologies
**Edge Systems and Applications are Increasing**

- **Application**
  - AR/VR
  - Video Surveillance
  - Autonomous Vehicle

- **System**
  - AWS Greengrass
  - Azure IoT Edge
  - Baidu BIE
  - Tecent ICEP
  - Paradrop
  - CORD
  - Alibaba LinkEdge
  - K3s
  - OneEdge

Year:
- 2016
- 2017
- 2018
- 2019
- 2020
Why Do We Need Rapid Customization?

• Similar Functionalities with Different Requirements!
  • Scenario 1: Industrial Internet
    - Slow Network
    - Traditional DB
    - Low-end Device
    - Fast Network
    - Lightweight DB
    - High-end Device

• Scenario 2: Education Experiment
  - Old Solution
  - Old Device
  - New Solution
  - New Device
Conventional Customization Steps

• **Target Edge System**

  Option 1: Develop from the ground up
  
  Option 2: Redevelop from existing modules
Conventional Customization Steps

• Target Edge System

Option 1: Develop from the ground up  ❌
Option 2: Redevelop from existing modules
Conventional Customization Steps

- **Target Edge System**

  ![Diagram of Edge System Components]

  - IoT Sensor → Data Base → Visualization

  **Prototype Design**

  - 1.5GHz, 4GB, $55
  - 2.0GHz, 8GB, $399

  **System Integration**

  - Power Supply
  - Sensors, Protocols
  - LoRa, 5G

  **Option 1:** Develop from the ground up
  **Option 2:** Redevelop from existing modules
Conventional Customization Steps

- **Target Edge System**

  ![Diagram of edge system components]

  - IoT Sensor
  - Data Base
  - Visualization

  Option 1: Develop from the ground up
  Option 2: Redevelop from existing modules

  ![Prototype Design]

  - 1.5GHz 4GB $55
  - 2.0GHz 8GB $399

  ![System Integration]

  - Sensors
  - Protocols

  ![APP Development]
Issues of Existing Work

• **Rapid Customization is HARD!**
  • Hard to redevelop from existing modules

  ![Diagram](image)

  - Complex Configurations
  - Exhaustive Development

• Hard to select hardware and software with better trade-off

  ![Diagram](image)

  - User Requirements
  - Software Modules
  - Hardware Devices
TinyEdge Vision

• **Easy Customization**
  • Composing an edge system with simple steps

  - Click to select software modules
  - Provide high-level configuration
  - Use out-of-the-box deployment tool to set up
  - Write few lines of code to specify application logic

• **Being aware of system performance before deployment**

  - Profile module condition under different situations
  - Latency Models
  - Workload Models
  - Hardware and software selection guidance
TinyEdge Vision

- **Easy Customization**
  - Composing an edge system with simple steps

  - Click to select software modules
  - Provide high-level configuration
  - Use out-of-the-box deployment tool to set up
  - Write few lines of code to specify application logic

- **Being aware of system performance before deployment**

  - Profile module condition under different situations
  - Latency Models
  - Workload Models
  - Hardware and software selection guidance
TinyEdge Customization Service - Configuration

• Complex Configuration Items Within a Module
  • Recall the example of the IoT application

Module-specific
  • DB name
  • DB port
  • License key
  • Auth enable
  …..

Docker-specific
  • workdir
  • volume
  • CPU quota
  • network limit
  …..

TinyEdge Configuration
  Basic
    • DB name
    • DB port
    • workdir
    …..
  Advanced
    • Auth enable
    • CPU quota
    • network limit
    …..

• Redundant Configuration Items Across Modules
  • Recall the example of the IoT application

InfluxDB
  • host
  • DB name
  • DB port
  • User name
  • User passwd

Grafana
  • host
  • DB name
  • DB port
  • User name
  • User passwd

#InfluxDB.<host>
#InfluxDB.<dbn>
#InfluxDB.<dbp>
#InfluxDB.<usrn>
#InfluxDB.<usrp>

TinyEdge Configuration
  InfluxDB
    • host
    • DB name
    • DB port
    • User name
    • User passwd
  Grafana
    • host
    • DB name
    • DB port
    • User name
    • User passwd

#InfluxDB.<host>
#InfluxDB.<dbn>
#InfluxDB.<dbp>
#InfluxDB.<usrn>
#InfluxDB.<usrp>
TinyEdge Customization Service - DSL

- Grammar of TinyEdge DSL
  - Module function call
  - Message routing
    - Pub()/Sub()
  - Serverless function
    - Serverless()/get_results()

- Dynamic Topic Generation
  - Performance issue
    - # of Topics ↑ → Throughput ↓ Resources ↑
  - Generation rules
    - 1. Define 1 topic multi-sub of single pub in 1 APP
    - 2. Merge topics with the same sub in 1 SYS

```python
1 from tinyedge.modules import mqtt, influxdb, grafana
2 from tinyedge.utils import Serverless
3 4 MQTT_Connector = mqtt("emg")
5 data = MQTT_Connector.get_data("device_id")
6 topic_id_1 = MQTT_Connector.Pub(data)
7 func = Serverless(language=""name":"python", "version":"3.6"", package=[["numpy":"1.14"], path)
8 func.Sub(topic_id_1)
9 data = func.get_results(**args)
10 topic_id_2 = func.Pub(data)
11 Influx = influxdb("influxdb")
12 data = Influx.Sub(topic_id_2)
13 InfluxDB.insert(data)
14 Grafana = grafana("grafana")
15 data = Grafana.Sub(topic_id_1)
16 Grafana.visualize(data)
```
TinyEdge Performance Estimation Service

• **Module Profile**
  - Configuration: Basic/Advanced | User
  - Functionality: Read/Write | Built-in
  - Performance model: Path | User
  - Resource requirement: Memory/Storage | User

• **Estimation Models**
  - Workload model
    - workload vector + hardware specs.
  - Latency model
    - Connecting/Processing modules
      - Data I/O size + RTT + exec + wait
    - $M^\alpha / M^\beta / \mathcal{C}$ queuing model

```json
"config":{
  "basic": [{"port":8086}, {"dbname":"test"}],
  "adv": [{"admin":true}, {"adminport":8083}],
}
"func":{
  "read": read(), "write": write(),
}
"model":{
  "load": "/iflx_w", "latency": "/iflx_t",
}
"requirement":{
  "mem": "20MB", "vol": "260MB",
}
```
Implementation

• Overall Architecture

Cloud
- Customization, Estimation Backend
- HW Specs, Module Profile Database
- Container Registry

Edge
- TinyEdge Runtime
- Deployment Tools
- Customized System

Connector
MQTT, HTTP, Modbus, Bluetooth

Database
InfluxDB, MySQL, Redis, SQLite

System
Device Mgmt, Grafana, Kafka, Serverless

Data Analysis
Obj Recog, ALPR, Filter, Car Tracking, Vibration Detection
Evaluation

• **Baseline**
  - Azure IoT Edge (Industry), EdgeX (Open-source community)

• **Cases**
  - Data connection and visualization (IoT)
  - Intelligent data processing (EI)
  - A hybrid-analysis system (GIoTTO[1])

[1] IoT Expedition: A large-scale deployment of Internet of Things that is extensible, privacy-sensitive, and end-user-programmable. Online Available: https://iotexpedition.org/
Evaluation

- **Customization**
  - Overall time reduction
    - 1. Environment set-up
    - 2. Module configuration
    - 3. System deployment
  - Lines-of-code reduction

- **Performance Estimation**
  - Baseline models
    - Linear regression
    - SVM
    - Random forest
  - Data size: 1.5k/0.5k each case

- **Results**
  - **Customization**
    - Environment set-up: 44.15% reduction
    - Module configuration: 67.79% reduction
  - **Performance Estimation**
    - Linear regression: ε = 0.83%
    - SVM: ε = 15.47%
TinyEdge

A rapid customization approach for edge systems target at IoT applications:

- **Full-stack optimizations** → Faster customization
- **2 types of models** → Better performance awareness

Thanks for your attention!

Q & A