The CLASS project has received funding from the European Union's Horizon 2020 research and innovation programme under the grant agreement No 780622
CLASS V2X Application: Collision Avoidance

Data Knowledge Base

Trajectory computation

Time-stamped object records: car / pedestrian / bicycle

City smart cameras: Object tracking & anonymization

Filtering & Collision Detection

Collision Warnings

Car position, Speed & direction

Smart Cars with ADAS
CLASS Analytics Fabric - Overview

Data Analytics Platform
- **Programming Models**
- OpenWhisk/EdgeWhisk
  - **Invoker**
- COMPSs + DataClay
  - **Connectors**
- CaaS API
  - Rotterdam
- PPM
  - Linux
- Kubernetes
- Docker

Compute Continuum

Events:
- Network (MQTT/..)
- Invocation: REST / CLI
- Sensor Data
- User Input
- Timer/Calendar

Actions / Workloads
- Map/Reduce
- pCEP
- COMPSs
- DNN

Apache OpenWhisk

COMPSs + DataClay

COMPSs
Serverless Example - OpenWhisk

1. Write a code function

```javascript
/** * Hello world as an OpenWhisk action. */
function main(params) {
    var name = params.name || 'World';
    return {payload: 'Hello, ' + name + '!'};
}
```

2. Deploy the code function as a serverless function (“action”):

```bash
$ wsk action create hello hello.js
```

3. The serverless function can be invoked directly:

```bash
$ wsk action invoke hello -blocking -result
{
    "payload": "Hello, World!"
}
```

4. Create an event trigger:

```bash
$ wsk trigger create myTrigger
```

[the event can be triggered via REST API by e.g., network message feed, sensor data, timers, etc]

5. Bind the serverless function to the event using a rule:

```bash
$ wsk rule create rule myRule myTrigger hello
```

6. Every time the event is triggered, the function will compute using parameters from event payload – and the result will show up in OW logs – Auto-scaling!

```json
{
    "payload": "Hello, OpenWhisk!"
}
```
Serverless Foundation in CLASS Analytics - Discussion

• Advantages:
  • Event-driven + direct invocation
  • Auto-scaling of function instances
  • Isolation when using shared infrastructure (e.g., cloud)
  • Lightweight - fit edge & cloud
  • Polyglot - use different programming languages and frameworks in the same application
  • Rapid prototyping

• Issues:
  • Stateless-ness
    • Mitigation – state retrieved from parameters and dataClay (CLASS data backbone)
  • Initialization / finalization overhead
    • Mitigation - warm start, async execution
  • Lack of scheduling / scale control
    • Mitigation – [manual] tuning
Low-latency Map/Reduce in Lithops

- A Python library for parallel computation on various backends incl. serverless
- `map` applies `func` to every element of `dataset`, yielding a new dataset:
  
  ```python
  map(func, dataset)
  ```
- `reduce` applies `func` to all elements of `dataset` with an accumulator, yielding a single result:
  
  ```python
  reduce(func, dataset)
  ```

More parallelism → lower total latency

- Faster data preparation
- Faster worker launch (less I/O)
- Leverage warm start better
  - Session connection in-mem
- Persist `map func` in worker image
- Faster result/finish protocol
  - Object storage → MQ
- Chunking elements to mitigate worker re-init cost

Speedup: ~4000 msec → 350 msec
X11.5 times!

Each element in the dataset is dispatched as a serverless worker function invocation

https://lithops-cloud.github.io/
Edge Analytics Agent

- **Federation** allows propagating software assets from one **Leader OW** to multiple **Follower OW-s**
  - Leader and Followers can be **any** OW instances
- Each Follower OW is a **programmable agent** of the Leader OW
- **Eventual consistency** of propagated assets (but typically very quick)
- Used for scenarios of deploying/updating software
  - Deploy/update software at all street nodes, or all cars
  - Update neural network at cars
Federation Example – CLASS Detector

- Detector is a long-running stateful docker service for edge nodes
- Federation is used to propagate an updated project manifest to edge nodes
- A periodical OW action in the edge detects the updated manifest and handles image pull and service update
CLASS Analytics Fabric - Summary

- Event-driven serverless infrastructure
  - Polyglot
  - Autoscaling
  - Simple + productive
  - Mitigations for CLASS: statelessness and init/fin overhead

- Leverage parallelism to reduce latency for independent computations
  - Trajectory Prediction and Collision Detection in Lithops
  - Concurrent computation in COMPSs

- Span cloud and edge via federation
Backup
CLASS Goals & Architecture

- Provide the **right level of abstraction** to facilitate the development of distributed big-data systems
- Increase **data throughput speed-up and analytics accuracy** by efficiently combining data-in-motion and data-at-rest analysis, while providing the required **real-time capabilities**
- Apply current open **initiatives, standards** and **industrial practices** with special interest on those adopted in the smart city
Project Structure

- **WP1** (Smart City Use Case)
- **WP5** (Data Analytics)
- **WP2** (Computation Distribution)
- **WP4** (Cloud Software Components)
- **WP3** (Edge Software Components)
- **WP2 to WP5** (Low Level Resource Managers)

**Compute Continuum**

- **WP1** Smart City Use-case
- **WP2** Software architecture
- **WP3** Edge Computing
- **WP4** Cloud Computing
- **WP5** Data Analytics Platform

*Diagram showing the relationships between the work packages.*
CLASS Smart City Use-Case – Intelligent Traffic Management

- 2021+ Connected Vehicles - Cooperative, Predictive, Sub-second
- Connected cars V2V, V2C, V2I
- **Collision avoidance** computations at the edge+cloud using FaaS (safety)

CLASS: https://class-project.eu/
Collision Avoidance Test Scenarios (from CLASS)

MASA – Modena Automotive Smart Area

- Cyclist jumps to street behind fence. NO CITY CAMERAS.
- Child crossing behind parked truck.
- Car emerging from side street behind tree corner.
CLASS Project
https://class-project.eu/

- Big-Data Analytics Across Cloud and Edge
- Edge = Car, Street cell, Smart Camera
- 2 main use-cases:
  - **Obstacle detection** – extend car awareness beyond sensors – around corners, beyond parked cars, etc
  - **Safety area vs. warning area**
  - Air pollution simulation crossed with real-time traffic data
Smart Prototype Car

Highly-connected cars: V2I connectivity tailored to the specific use-case and application scenario

- Multiple cameras @4K resolution
- LRR
- MRR
- Ultrasound sensors for short range detection

Advanced module for Vehicle-To-Cloud (V2C)
V2I communication through a 4G and 5G connectivity
Edge-side: Real-time road user detection

- Multiple infrastructure cameras detect road users and obstacles
  - vehicles, buses, trucks, pedestrians, bicycles, etc.
- A consistent representation of RU’s is sent to L3/L4 vehicles (V2I) in real-time

Camera-to-car < 100ms!

- V2I sensing system for harsh urban environment
- Redundant and robust perception
- Safer obstacle detection

To cloud infrastructure
Obstacle Detection At The Car

- **Cameras** recognize & determine the hazard level of the obstacle/pedestrian.

- In case of danger a warning is sent to the **vehicles** placed in the limited area directly into the car.

- Connected vehicles communicate the hazard to **other traffic participants**, avoiding accidents.
MASA Computing/Communication Infrastructure