


# Urban-Scaled Traffic Management Using FCD

**Ece Yılmaz & Murat Tulgaç**

*ISSD Bilisim Elektronik*

F4Fp-SME-1

November 2020

- Experiment Description
- Project Results
- Business Impact
- Feedback

# Overview

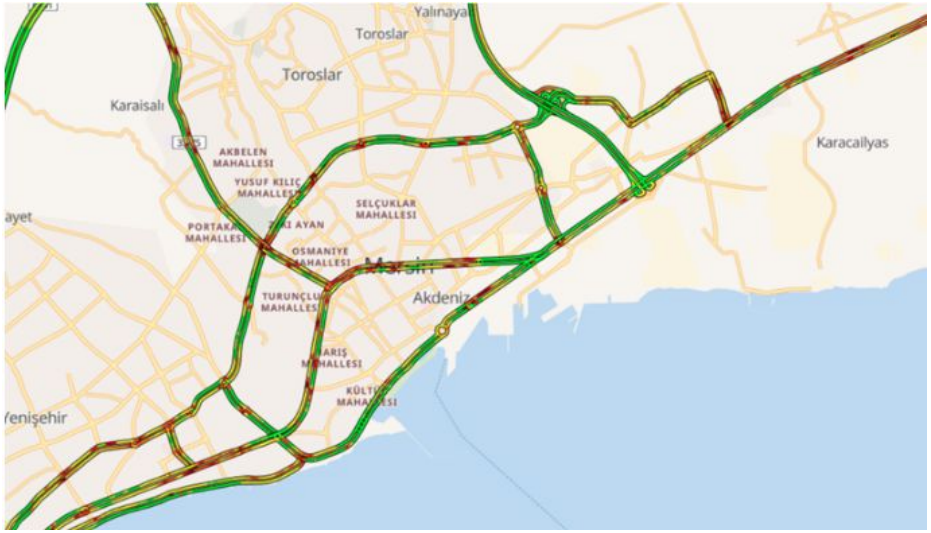
# Experiment Description

# Who we are

- ISSD was founded in Ankara, Turkey in 2009
- **One ultimate goal:** providing intelligent solutions to traffic and transportation

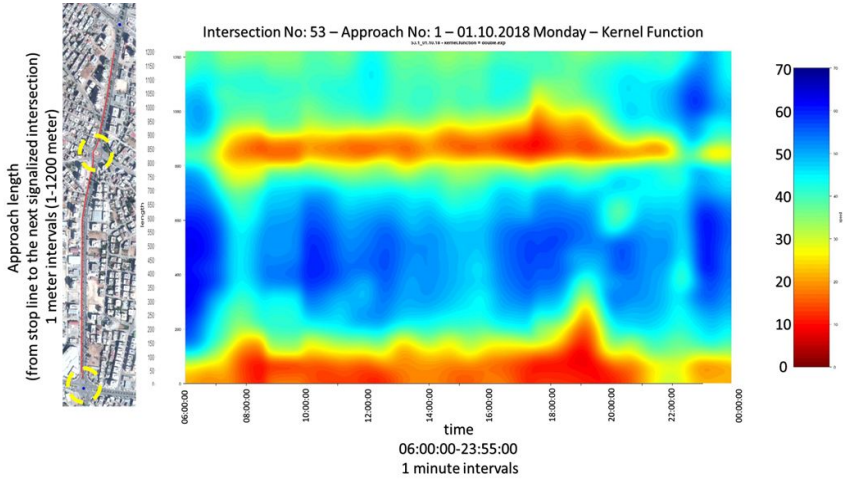
# What is FCD (Floating Car Data)

- Gathered through GPS modules in vehicles
- Provides information about time and coordinates of the probe vehicles to obtain travel time
- Our version directly provides travel time for road segments

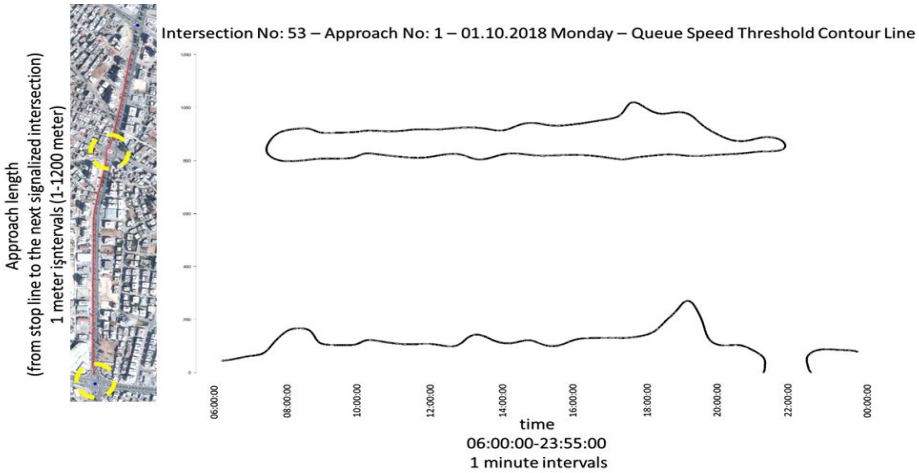


# What we do with FCD

## Speed Profile Calculation




## Queue Length Estimation



# The Experiment

## Motivation

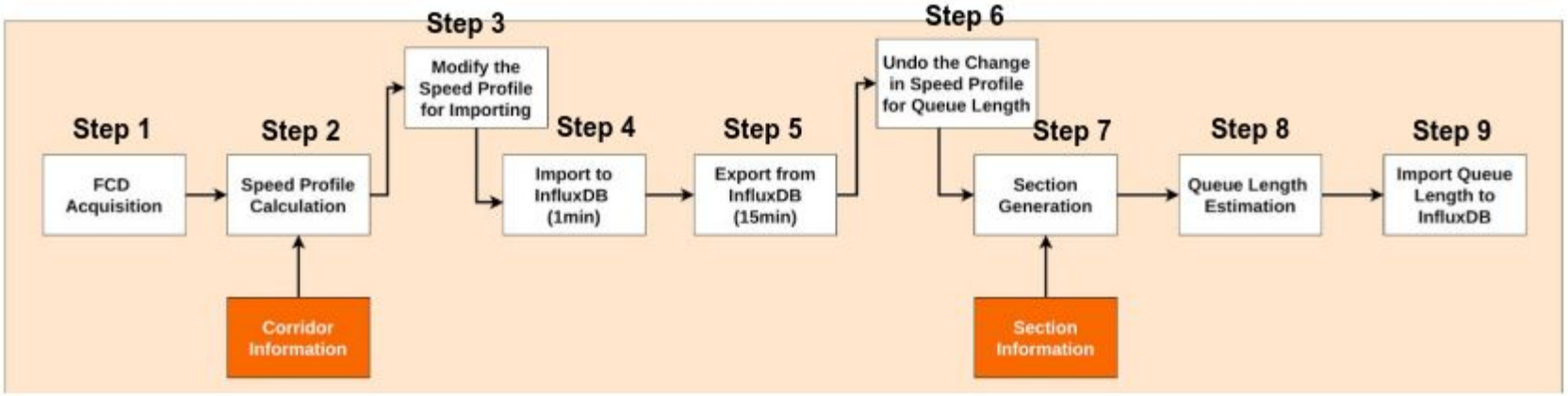
- **Goal:** Using FCD for Incident Detection and Junction Management
- **Requirement:** Real-time processing of FCD
- **Problem:** Lack of experience in the topic
- **Solution:** Fed4FIRE+

## Concept and Objectives

- Create a big data infrastructure using Fed4FIRE+ Testbed facilities
- Integrate the algorithms into this infrastructure

# Project Results

# Project Steps



# Project Results

**Step 2**: Calculating meter-based speed profile

| discoveryTime | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | ... | 3566 | 3567 | 3568 | 3569 | 3570 | 3571 | 3572 | 3573 | 3574 | 3575 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2020-09-17 16:35:00 | 11.3 | 11.2 | 11.1 | 10.9 | 10.8 | 10.6 | 10.5 | 10.4 | 10.2 | 10.1 | ... | 48.7 | 48.7 | 48.7 | 48.7 | 48.7 | 48.7 | 48.7 | 48.7 | 48.7 | 48.7 |

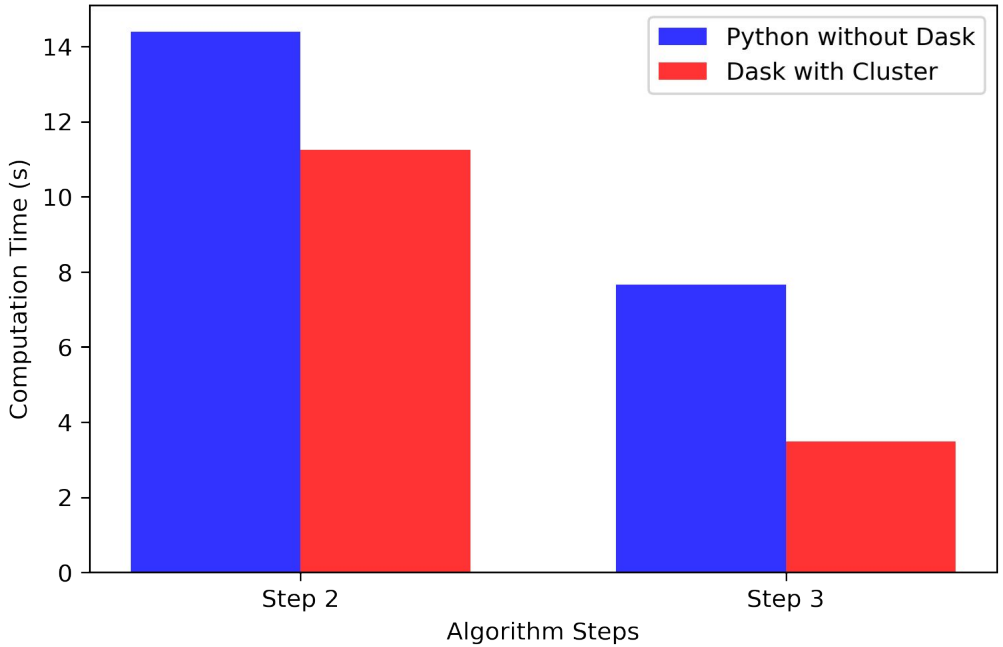
*The sample data of the Speed Matrix*

**Step 3:** Modifying the speed matrix for importing to InfluxDB

**Result:** Estimating the queue length for each section

# Project Results

Computation time of **Step 2** and **Step 3** for 10 corridors of the city in one minute with flowing data



# Business Impact

# Values perceived

- Primary goal was to gain experience on distributed computing and real-time data processing
- Contacted with Tengu about the tools can be used for our specific needs
- Used the recommended tools for the first time in this experiment

# Impact on our business

- Outputs of the experiment:
  - section-based speed profile (calculated in real-time)
  - section-based queue length information (calculated in real-time)

# Impact on our business

- Outputs are used in:
  - as a verification of our other traffic control products with before-after comparisons
  - more complex algorithms that can be a product

# Future plans with Fed4FIRE+

- Re-applying to the Stage 2 for more complex algorithms and tests with an improved infrastructure
  - Junction management with FCD
  - Using different tools for performance comparison
  - Finding the optimal infrastructure
- Using the networks that can be provided by FED4FIRE+

# Feedback

# Hardware Properties

| Technologies | Nr of VMs | CPU per VM | Mem per VM | Disk per VM |
|---|---|---|---|---|
| Dask Kubernetes | 4 | 4 core | 16 GB | 1 TB |
| InfluxDB | 1 | 8 core | 16 GB | 2 TB |

- Unnecessary amount of RAM
- Low processing power in VMs
  - While the code running in Tengu took 8.47 seconds, it took 3.78 seconds in local

# Comparison

## Tengu

Client

**Scheduler:** tcp://my-dask-scheduler:8786
**Dashboard:** http://my-dask-scheduler:8787/status

Cluster

**Workers:** 3
**Cores:** 12
**Memory:** 50.46 GB

```
def square(x):
    return x ** 2
```

```
def neg(x):
    return -x
```

```
start = time.time()
A = c.map(square, range(10000))
B = c.map(neg, A)
total = c.submit(sum, B)
print(total.result())
end = time.time()
print(end-start)
```

```
-333283335000
8.472558736801147
```

## Local

Client

**Scheduler:** tcp://127.0.0.1:34475
**Dashboard:** http://127.0.0.1:8787/status

Cluster

**Workers:** 4
**Cores:** 12
**Memory:** 8.20 GB

```
def square(x):
    return x ** 2
```

```
def neg(x):
    return -x
```

```
start = time.time()
A = client.map(square, range(10000))
B = client.map(neg, A)
total = client.submit(sum, B)
print(total.result())
end = time.time()
print(end-start)
```

```
-333283335000
3.784132957458496
```

# Tools

- Kubernetes for cluster infrastructure
- Dask for distributed computing
- InfluxDB for storing outputs

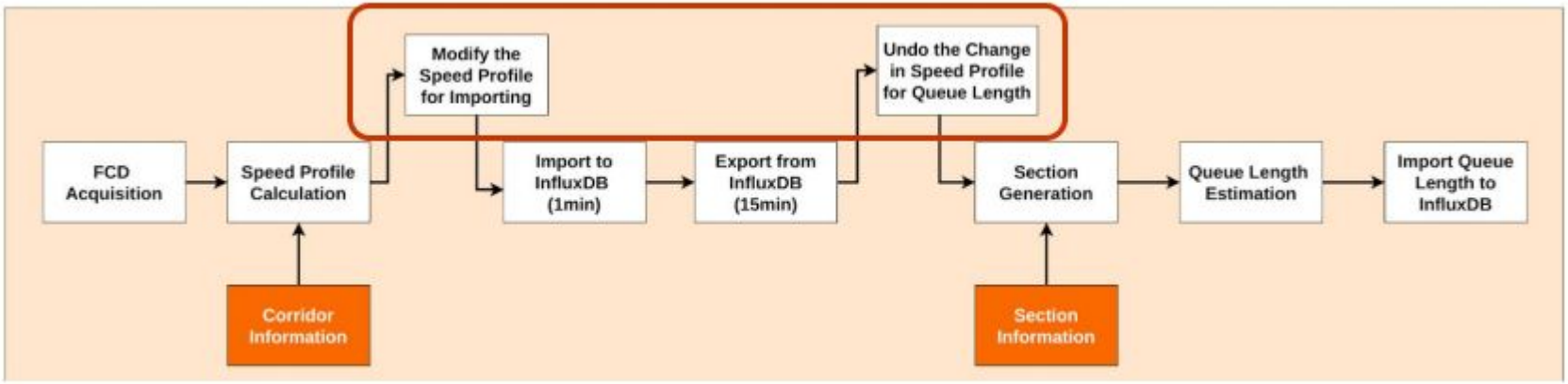
kubernetes


influxdb

# Tools

No need to use time-series database

- Caused altering data before importing and after exporting





Co-funded by the European Union


Co-funded by the Swiss Confederation

This project has received funding from the European Union's Horizon 2020 research and innovation programme, which is co-funded by the European Commission and the Swiss State Secretariat for Education, Research and Innovation, under grant agreement No 732638.

**WWW.FED4FIRE.EU**