IoT at Scale: a Bike Story

What is IoT

The Internet of things (IoT) is a system of interrelated computing devices, mechanical and digital machines, objects, animals or people that are provided with unique identifiers (UIDs) and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction

PBSC
URBAN SOLUTIONS

PBSC Urban Solutions - Scale

- System runs 25 cities over four continents
- Thousands of stations
- Tens of thousand bikes and docking points
- 80 million trips in 2019

PBSC
URBAN SOLUTIONS

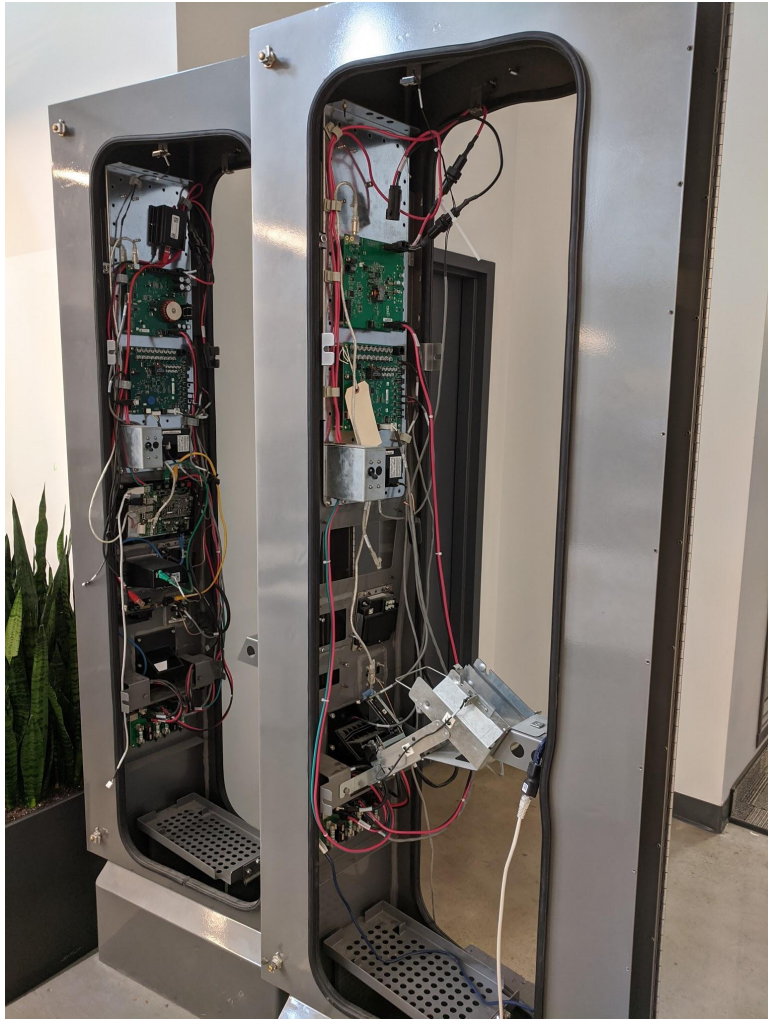
Data Streaming


METROLINX

Stations composed of many electronic components

Constant flow of data coming from the stations

- At periodic interval (sensors)
- Event-based (bike docking and undocking).



PBSC
URBAN SOLUTIONS

Reactive systems

- Asynchronous message-passing
- Non-blocking
- Backpressure

In practice: Reactive Streams, actor systems

PBSC
URBAN SOLUTIONS

Infrastructure

Stations send periodically sensor data about battery, connections, hardware components, etc.

Given events and sensor data, much higher write %

Specialized databases perform much better in that scenario

- Timeseries
- Column based

Popular alternative: dump the problem over the fence (to Kafka)

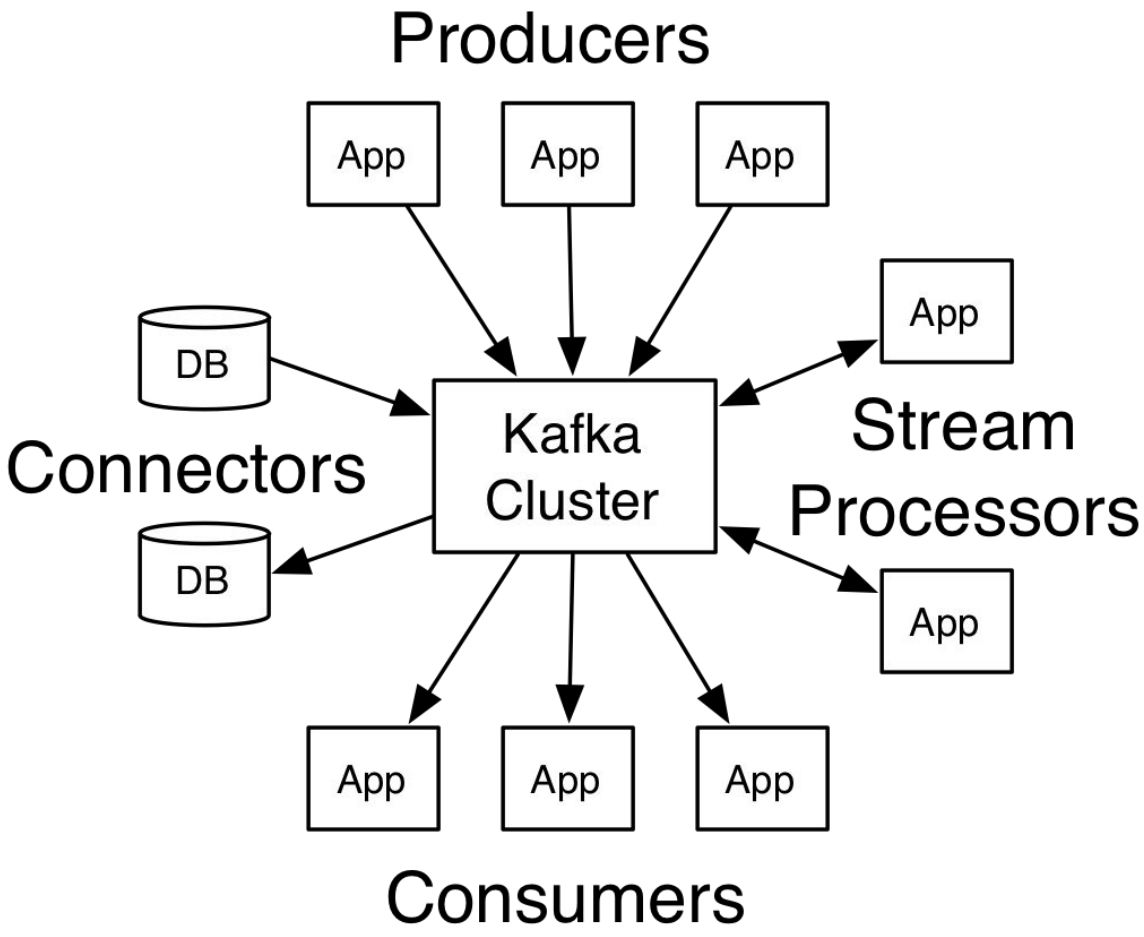
PBSC
URBAN SOLUTIONS

Infrastructure - Kafka

Publish and subscribe to streams of records

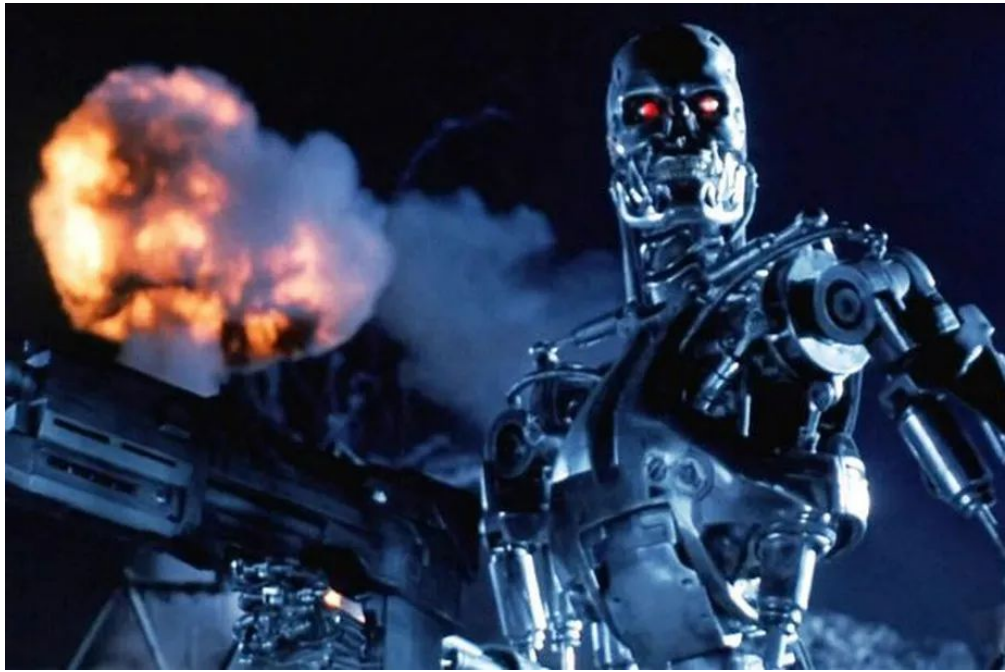
Store streams of records in a fault-tolerant durable way.

Process streams of records as they occur.


Producers
App
App
App
DB
Connectors
DB
Kafka
Cluster
App
Stream
Processors
App
App
App
App
Consumers

When the machines stop cooperating

- A single bug can lead to a DDoS
- Hardware failures can be… interesting

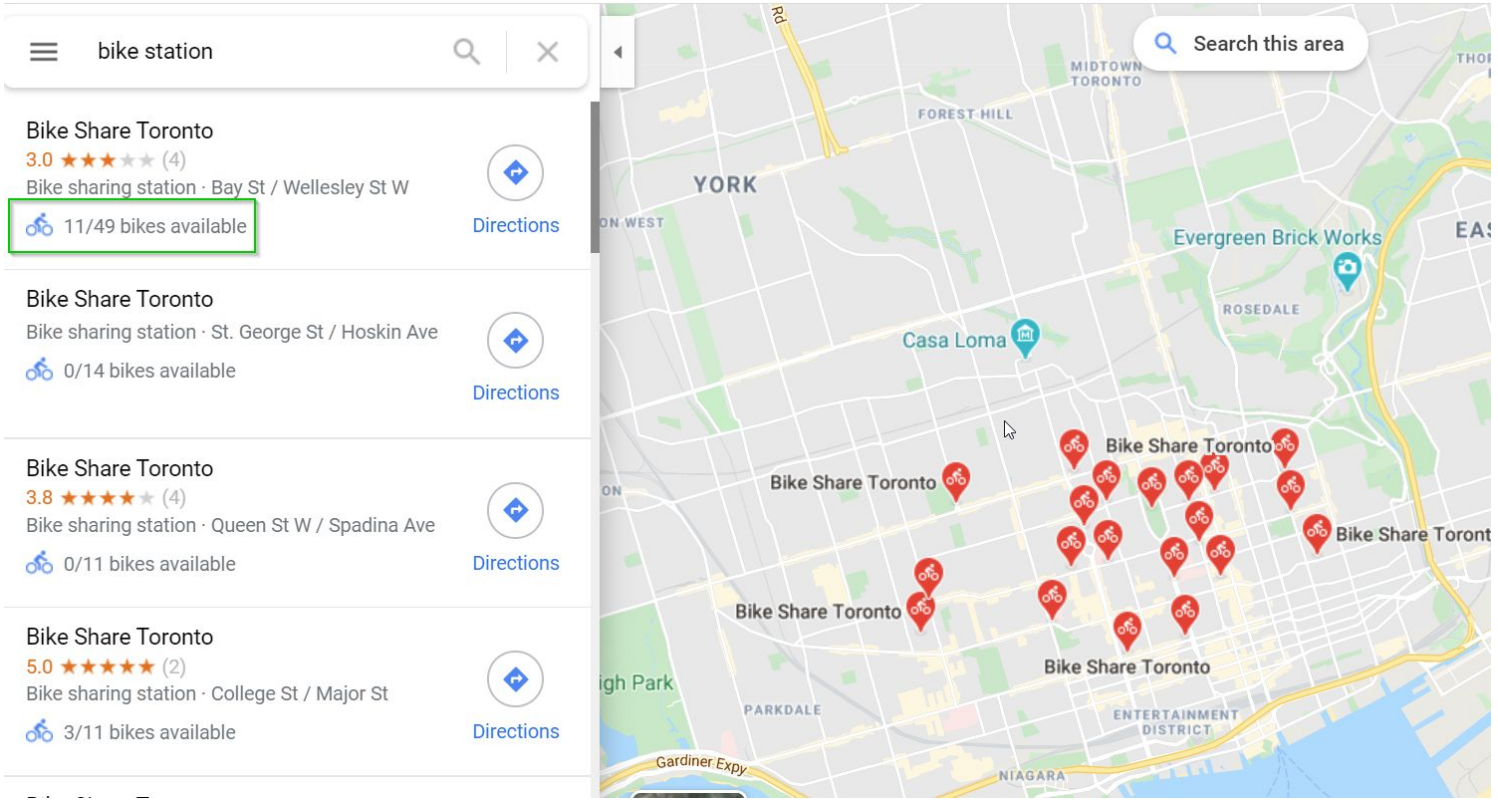


PBSC
URBAN SOLUTIONS

Coping with overflow of data

- Buffer it
- Drop it
- Block until consumer is done
- Throw it in a queue
- Apply backpressure

State Management



PBSC URBAN SOLUTIONS

Scenarios

Stations, bikes and docking points are all assets holding a state.

Bike at station 1, dock 12 -> rent by customer -> Returned to station 3, dock 15

Bike state changed.

Dock state changed.

Info needed for dashboards, rebalancing, ensure a customer cannot rent multiple bikes, etc.

Curveball: the station must keep functioning even if it loses connection.

Handling stateful assets

Stateless services are great for scaling, not so much for dealing with state

Possible solutions: In memory caches like Redis + asynchronous persistence to a DB.

Event-sourcing: the state of the assets is computed from

Actor based system like Akka.

PBSC
URBAN SOLUTIONS

Limitations of a CRUD based approach

Ordering of events

Concurrency is hard to get right (deadlocks and races)

SQL performance is not stellar for highly concurrent workloads

PBSC
URBAN SOLUTIONS

Event sourcing

End state computed from immutable records

Well suited for IoT as intrinsically event based.

Current state = E1 -> E2 -> E3 ...

PBSC
URBAN SOLUTIONS

16:30 Bike 12 rented from station A

16:35 Bike returned at station B

Bike marked as defective by customer in mobile app.

End state: Bike is inoperative at Station B.

Connectivity

Bi-directional communication

Commands must be sent to stations to unlock bikes

Operators taking bike for maintenance/rebalancing

Firmware updates

PBSC
URBAN SOLUTIONS

Possible solutions

- VPN
- Long-polling
- Websocket, combines well with actors

Non-HTTP is also an option -> MQTT

PBSC
URBAN SOLUTIONS

MQTT

Pub/Sub on top of TCP

Built for low-bandwidth and consumption battery

Benchmarks ranges from equivalent to HTTP to 10-50x better, depending on scenario

QoS options

Usually supported on IoT platforms

PBSC
URBAN SOLUTIONS

Thank you!