Video Analytics at the Edge: Fun with Apache Edgent, OpenCV and a Raspberry Pi

Dale LaBossiere, Will Marshall, Jerome Chailloux

*Apache Edgent* is currently undergoing Incubation at the Apache Software Foundation.
Introduction

• This presentation:
  • Introduction to datastreaming
  • How Edgent fits in to the datastreaming IoT universe
  • Code examples & Video Analytics demo
  • Wrap up & questions
Datastreaming in a nutshell

Consider a connected sensor:
- Might run “forever”
- We need a way of consuming the data stream
- Depending on the values, we might want to perform an action.
Datastreaming in a nutshell

- Reading from a number of remote sensors
- Perform a moving average
- Perform an action if the temperature goes above a certain value:

![Graph showing temperature over time with a trigger point]
Typical Data Flow

• Streaming data is usually pushed to a cluster or warehouse.
• Analysis of the data is done in the cluster.
Typical Data Flow

• Streaming data is usually pushed to a cluster or warehouse.
• Analysis of the data is done in the cluster.
Typical Data Flow

• Streaming data is usually pushed to a cluster or warehouse.
• Analysis of the data is done in the cluster.

• High network costs are prohibitive: lost greenbacks $$
Typical Data Flow

• Streaming data is usually pushed to a cluster or warehouse.
• Analysis of the data is done in the cluster.

{msg : “do something”}

• Return messages incur round trip latency
Takeaway

• Send less information – only send “interesting” data items.
• Perform streaming analysis on the device, within reason.
Apache Edgent Community

• A **community** for accelerating Edge Analytics
  • Open Source, incubating at Apache Software Foundation
    • [https://edgent.apache.org/](https://edgent.apache.org/)
    • [https://github.com/apache/incubator-edgent](https://github.com/apache/incubator-edgent)

• A programming SDK with functional flow API for streaming analytics
  • Initial support for Java 7,8 & Android,
  • Goal is to support multiple languages with priorities driven by the community

• A modular, lightweight, embeddable, and extensible runtime
Can create a data processing Topology
Features – Stream Processing API

- Functions / Lambdas, supports fluent style, per-tuple processing, interfaces
- Supplier<T>, Predicate<T>, Function<T,U>, and Consumer<T> are used to define data operations.
- Topology – one or more dataflow processing graphs
  - Construct source streams via: poll(), generate(), source(), events(), collection() of(), strings()
- TStream<T> - a logical sequence of tuples T
  - Transform via: filter(), modify(), map(), flatMap(), peek(), sink(), union(), split(), last(), join()
  - Plumbing transforms: isolate, pressureReliever, valve, gate, barrier, blockingDelay, concurrent, parallel
- TWindow<T,K> - partitioned numeric or time bounded collection; sliding/continuous and batch/tumbling aggregations
Features – Connectors

• Connectors provide a means of communicating with external systems:
  • IoT hub
  • MQTT, Kafka
  • JDBC, File,
  • HTTP, WebSocket client
  • Command
  • SerialDevice, SerialPort
  • PubSub TStreams
  • Contributions welcome!
Features – Analytics

• Filters: Range, deadBand, deadTime

• Common aggregations for windows
  • MIN, MAX, SUM, MEAN, STDDEV, SLOPE
  • Single and multiple variable aggregations on JsonObject and non-JsonObject tuples

• Again, contributions welcome!
public static void main(String[] args) {
    DirectProvider provider = new DirectProvider();
    Topology top = provider.newTopology();
    IotDevice iotDevice = IotpDevice.quickstart(top, "raspberry-pi-0003");
    // https://quickstart.internetofthings.ibmcloud.com/#/device/raspberry-pi-0003/sensor/

    TStream<Double> readings = top.poll(new TempSensor(), 1, TimeUnit.SECONDS);
    TStream<JsonObject> events = readings.map(JsonFunctions.valueOfNumber("temp"));
    iotDevice.events(events, "readingEvents", QoS.FIRE_AND_FORGET);
    events.print();

    provider.submit(top);
}

- Read from a temperature sensor every second
- Convert to JSON
- Send data to MQTT / Watson IoT Platform and print to screen
Sample Application – Connect via WIoT

```java
public static void main(String[] args) {
    DirectProvider provider = new DirectProvider();
    Topology top = provider.newTopology();
    IotDevice iotDevice = IotpDevice.quickstart(top, "raspberry-pi-0003");
    // https://quickstart.internetofthings.ibmcloud.com/#/device/raspberry-pi-0003/sensor/
    TStream<Double> readings = top.poll(new TempSensor(), 1, TimeUnit.SECONDS);
    TStream<JsonObject> events = readings.map(JsonFunctions.valueOfNumber("temp"));
    iotDevice.events(events, "readingEvents", QoS.FIRE_AND_FORGET);
    events.print();
    provider.submit(top);
}
```

- A provider is the container in which applications are run.
- Creating a provider is the first step in writing an Edgent application.
public static void main(String[] args) {
    DirectProvider provider = new DirectProvider();
    Topology top = provider.newTopology();
    IotDevice iotDevice = IotDevice.quickstart(top, "raspberry-pi-0003");
    // https://quickstart.internetofthings.ibmcloud.com/#/device/raspberry-pi-0003/sensor/

    TStream<Double> readings = top.poll(new TempSensor(), 1, TimeUnit.SECONDS);
    TStream<JsonObject> events = readings.map(JsonFunctions.valueOfNumber("temp"));
    iotDevice.events(events, "readingEvents", QoS.FIRE_ANDFORGET);
    events.print();

    provider.submit(top);
}

• A Topology represents your application. It keeps track of data operations & flow.
public static void main(String[] args) {
    DirectProvider provider = new DirectProvider();
    Topology top = provider.newTopology();
    IotDevice iotDevice = IotDevice.quickstart(top, "raspberry-pi-0003");
    // https://quickstart.internetofthings.ibmcloud.com/#/device/raspberry-pi-0003/sensor/
    TStream<Double> readings = top.poll(new TempSensor(), 1, TimeUnit.SECONDS);
    TStream<JsonObject> events = readings.map(JsonFunctions.valueOfNumber("temp"));
    iotDevice.events(events, "readingEvents", QoS.FIRE_AND_FORGET);
    events.print();
    provider.submit(top);
}

- An IotDevice represents a connection to a backend. This can be MQTT, Kafka, a websocket or anything.
- This is a quickstart device connection, meaning it sends data to a public MQTT server. Good for demos.
- Edgent is extensible, if you need to create your own device type, you implement the device interface.
public static void main(String[] args) {
    DirectProvider provider = new DirectProvider();
    Topology top = provider.newTopology();
    IotDevice iotDevice = IotpDevice.quickstart(top, "raspberry-pi-0003");
    // https://quickstart.internetofthings.ibmcloud.com/#/device/raspberry-pi-0003/sensor/

    TStream<Double> readings = top.poll(new TempSensor(), 1, TimeUnit.SECONDS);
    TStream<JsonObject> events = readings.map(JsonFunctions.valueOfNumber("temp"));
    iotDevice.events(events, "readingEvents", QoS.FIRE_AND_FORGET);
    events.print();

    provider.submit(top);
}

- We poll a simulated temperature sensor every second.
- TempSensor is a Supplier<T> – calling .get() produces data.
public static void main(String[] args) {
    DirectProvider provider = new DirectProvider();
    Topology top = provider.newTopology();
    IotDevice iotDevice = IotpDevice.quickstart(top, "raspberry-pi-0003");
    // https://quickstart.internetofthings.ibmcloud.com/#/device/raspberry-pi-0003/sensor/

    TStream<Double> readings = top.poll(new TempSensor(), 1, TimeUnit.SECONDS);
    TStream<JsonObject> events = readings.map(JsonFunctions.valueOfNumber("temp"));
    iotDevice.events(events, "readingEvents", QoS.FIRE_AND_FORGET);
    events.print();

    provider.submit(top);
}

- **TStream** is the primary abstraction in Edgent. It represents an unbounded sequence of data items (tuples).
- The tuples that flow through a **TStream** can be of any Java type.
```java
public static void main(String[] args) {
    DirectProvider provider = new DirectProvider();
    Topology top = provider.newTopology();
    IotDevice iotDevice = IotpDevice.quickstart(top, "raspberry-pi-0003");
    // https://quickstart.internetofthings.ibmcloud.com/#/device/raspberry-pi-0003/sensor/

    TStream<Double> readings = top.poll(new TempSensor(), 1, TimeUnit.SECONDS);
    TStream<JsonObject> events = readings.map(JsonFunctions.valueOfNumber("temp"));
    iotDevice.events(events, "readingEvents", QoS.FIRE_AND_FORGET);
    events.print();

    provider.submit(top);
}
```

- The `map` method takes a tuple and produces 0 or 1 tuple as output.
- The type can be different.
- Here, we convert each temperature reading to a `JsonObject` so we can send it over MQTT.
public static void main(String[] args) {
    DirectProvider provider = new DirectProvider();
    Topology top = provider.newTopology();
    IotDevice iotDevice = IotpDevice.quickstart(top, "raspberry-pi-0003");
    // https://quickstart.internetofthings.ibmcloud.com/#/device/raspberry-pi-0003/sensor/

    TStream<Double> readings = top.poll(new TempSensor(), 1, TimeUnit.SECONDS);
    TStream<JsonObject> events = readings.map(JsonFunctions.valueOfNumber("temp"));
    iotDevice.events(events, "readingEvents", QoS.FIRE_AND_FORGET);
    events.print();
}

- The `device.events()` method indicates that the tuples of a stream should be sent to the backend.
- Watson IoT, in this case
public static void main(String[] args) {
    DirectProvider provider = new DirectProvider();
    Topology top = provider.newTopology();
    IotDevice iotDevice = IotDevice.quickstart(top, "raspberry-pi-0003");
    // https://quickstart.internetofthings.ibmcloud.com/#/device/raspberry-pi-0003/sensor/

    TStream<Double> readings = top.poll(new TempSensor(), 1, TimeUnit.SECONDS);
    TStream<JsonObject> events = readings.map(JsonFunctions.valueOfNumber("temp"));
    iotDevice.events(events, "readingEvents", QoS.FIRE_AND_FORGET);
    events.print();

    provider.submit(top);
}

• Additionally, we print each temperature reading to stdout.
Sample Application – Connect via WIoT

```java
public static void main(String[] args) {
    DirectProvider provider = new DirectProvider();
    Topology top = provider.newTopology();
    IotDevice iotDevice = IotpDevice.quickstart(top, "raspberry-pi-0003");
    // https://quickstart.internetofthings.ibmcloud.com/#/device/raspberry-pi-0003/sensor/
    TStream<Double> readings = top.poll(new TempSensor(), 1, TimeUnit.SECONDS);
    TStream<JsonObject> events = readings.map(JsonFunctions.valueOfNumber("temp"));
    iotDevice.events(events, "readingEvents", QoS.FIRE_AND_FORGET);
    events.print();
    provider.submit(top);
}
```

- Invoking `provider.submit` on the topology object submits it for immediate execution.
Sample Application Demo
Use your device in an application created with IBM Bluemix.

Click here for more details.

Go to your Bluemix account

SIGN UP

LOG IN

Note: When you sign up for a trial you may have to wait up to 24 hours to receive your log-in information.

Create an app using the Internet of Things Starter from the Catalog

CREATE APP

Note: You will have to name your app and wait for a few minutes for it to start running.

When your app is running, select the app URL or type it into the browser to open the Node-RED...
Video Analytics at the Edge
Experiment: Video Analytics at the Edge

Diagram:

- Raw Video
- Face Detection
- Face Recognition
- Model
- Smart Camera
- IoT Device
- Face Events
- Control Commands
- IoT Hub
- Streaming Analytics Recognition
SMART CAMERA
RASPBERRY Pi

EDGE"NT OPENCV FACE DETECT

FACE EVENTS
COMMANDS

WATSON DATA PLATFORM
IBM BLUEMIX

WATSON IOT PLATFORM

STREAMING ANALYTIC SERVICE APPS

EVENT SUBSCRIBER

CONTROL APP
Smart Camera Demo & code show&tell
Summary

We covered

• Intro to Apache Edgent: what it is and why you want it
  • Smarter devices can reduce communication costs, reduce decision making latency, and provide autonomy while disconnected.

• Pushing video analytics to the edge:
  • a controllable Image/Face detection Smart Camera IoT device
  • Edgent, OpenCV and a Raspberry Pi
  • easily connected to a cloud enterprise IoT device hub, enabling integration with rich backend analytic consumers.
  • Get the code: URL TBD

• Conclusion: using Edgent can accelerate your development of smarter IoT devices and gateways

• Join the community. Contributors welcome!
  • https://edgent.apache.org/
  • Subscribe to dev@edgent.apache.org