Kappa–Serverless IoT

Per Persson
Ericsson Research
Lund, Sweden
Serverless IoT
Does it even make sense?

- IoT is about
  - capturing, digitally representing, and manipulating the physical world
  - devices forming the interface between the domains
  - long-running, stateful applications
- Things IoT should adopt from cloud and serverless
  - simplicity
  - resource sharing, multi-tenancy
  - fine-grained metrics
Serverless IoT

Does it even make sense?

- IoT is about
  - capturing, digitally representing, and manipulating the physical world
  - devices forming the interface between the domains
  - long-running, stateful applications
- Things IoT should adopt from cloud and serverless
  - simplicity
  - resource sharing, multi-tenancy
  - fine-grained metrics

<table>
<thead>
<tr>
<th>Serverless computing</th>
<th>Serverless IoT</th>
</tr>
</thead>
<tbody>
<tr>
<td>transparent provisioning</td>
<td>semantics</td>
</tr>
<tr>
<td>stateless</td>
<td>local state</td>
</tr>
<tr>
<td>short lived</td>
<td>long lived, but mostly sleeping</td>
</tr>
<tr>
<td>use your favourite language</td>
<td>restricted programming model</td>
</tr>
<tr>
<td>paradigm</td>
<td></td>
</tr>
</tbody>
</table>
IoT-programming with Calvin
Separating the what from the where and how
Runtimes and Actors

- **Capabilities**: sense.temp, math.fft
- **Attributes**: location="kitchen", owner="me"

---

**Runtime**
- Portable platform abstraction layer
  - ARM CortexM (constrained)
  - ARM CortexA and up to DC (base)
- Tracks capabilities and attributes
- Handles message passing and security
- Mesh of runtimes create **single machine illusion**

**Actor**
- Unit of isolation and execution
- Responds to events and incoming messages
- Atomic operations
- Local state only
- Moves to runtime to access its resources
- Common code across all runtimes
Applications

- **iip**: flow.Init
  - in: token
  - out: token

- **delay**: std.ClassicDelay
  - token: token

- **sense**: sensor.TriggeredTemperature
  - trigger: token
  - centigrade: token

- **collect**: flow.Collect
  - token: token

- **print**: io.Print
  - in: token
Applications

- trigger
  - MyTrigger
  - token

- sense
  - sensor.TriggeredTemperature
    - trigger
    - centigrade

- collect
  - flow.Collect
    - token
    - token

- print
  - io.Print
    - in
Finding Resources

<table>
<thead>
<tr>
<th>sense</th>
<th>capabilities:</th>
<th>attributes:</th>
</tr>
</thead>
<tbody>
<tr>
<td>sensor:TriggeredTemperature</td>
<td>sense.temp</td>
<td>country=&quot;us&quot;</td>
</tr>
<tr>
<td>trigger</td>
<td>attributes:</td>
<td>country=&quot;sweden&quot;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>capabilities:</th>
<th>sense:humidity</th>
</tr>
</thead>
<tbody>
<tr>
<td>attributes:</td>
<td>name=&quot;us&quot;</td>
</tr>
</tbody>
</table>
Finding Resources

apply sense : attr_match(country="us")

capabilities: sense.temp
attributes: country="us"
capabilities: sense.temp
attributes: country="sweden"
capabilities: sense.humidity
attributes: name="us"
apply sense : attr_match(country="us")

```
capabilities:
sense: temp
attributes:
country="us"
```
Finding Resources

apply `sense : attr_match(country="us")`

```
sense
  sensor.TriggedTemperature
  trigger  centigrade

capabilities:
sense.temp
attributes:
country="us"

capabilities:
sense.temp
attributes:
country="sweden"

capabilities:
sense.humidity
attributes:
name="us"
```
Replication and Scaling

capabilities:
sense.temp
attributes:
name="s1"
owner="me"
capabilities:
sense.temp
attributes:
name="s2"
owner="me"
capabilities:
sense.temp
attributes:
name="s3"
owner="me"
Replication and Scaling

apply sense : device_scaling() & attr_match(owner="me")

capabilities: sense.temp
attributes:
  name="s1"
  owner="me"

capabilities: sense.temp
attributes:
  name="s2"
  owner="me"

capabilities: sense.temp
attributes:
  name="s3"
  owner="me"
Replication and Scaling

apply `sense : device_scaling()` & `attr_match(owner="me")`

capabilities:
sense.temp

attributes:
name="s1"
owner="me"

capabilities:
sense.temp

attributes:
name="s2"
owner="me"

capabilities:
sense.temp

attributes:
name="s3"
owner="me"
Replication and Scaling

apply sense : device_scaling() & attr_match(owner="me")

capabilities:
sense.temp

attributes:
name="s1"
owner="me"

capabilities:
sense.temp

attributes:
name="s2"
owner="me"

capabilities:
sense.temp

attributes:
name="s3"
owner="me"
Replication and Scaling

apply sense : device_scaling() & attr_match(owner="me")

capabilities:
sense.temp
attributes:
name="s1"
owner="me"

capabilities:
sense.temp
attributes:
name="s2"
owner="me"

capabilities:
sense.temp
attributes:
name="s3"
owner="me"
Replication and Scaling

apply sense : device_scaling() & attr_match(owner="me")
apply sense : performance_scaling() & attr_match(owner="me")
Kappa–Serverless IoT
Architecture

Cloud

World of other Things
Architecture

- Web service with REST API

<table>
<thead>
<tr>
<th>Command</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>PUT /kappa</td>
<td>Deploy script in body, return reference</td>
</tr>
<tr>
<td>POST /kappa/ref</td>
<td>Send JSON data to kappa inport</td>
</tr>
<tr>
<td>GET /kappa/ref</td>
<td>Read JSON data from kappa outport</td>
</tr>
<tr>
<td>DELETE /kappa/ref</td>
<td>Delete a kappa</td>
</tr>
<tr>
<td>GET /kappa</td>
<td>List all kappas</td>
</tr>
</tbody>
</table>
Architecture

- Web service with REST API
- FaaS script defined as a component

<table>
<thead>
<tr>
<th>Command</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>PUT /kappa</td>
<td>Deploy script in body, return reference</td>
</tr>
<tr>
<td>POST /kappa/ref</td>
<td>Send JSON data to kappa inport</td>
</tr>
<tr>
<td>GET /kappa/ref</td>
<td>Read JSON data from kappa outport</td>
</tr>
<tr>
<td>DELETE /kappa/ref</td>
<td>Delete a kappa</td>
</tr>
<tr>
<td>GET /kappa</td>
<td>List all kappas</td>
</tr>
</tbody>
</table>
Architecture

- Web service with REST API
- FaaS script defined as a component
- Runtime with two special actors as bridge to API
  - kappa.Source
  - kappa.Sink

<table>
<thead>
<tr>
<th>Command</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>PUT /kappa</td>
<td>Deploy script in body, return reference</td>
</tr>
<tr>
<td>POST /kappa/ref</td>
<td>Send JSON data to kappa inport</td>
</tr>
<tr>
<td>GET /kappa/ref</td>
<td>Read JSON data from kappa outport</td>
</tr>
<tr>
<td>DELETE /kappa/ref</td>
<td>Delete a kappa</td>
</tr>
<tr>
<td>GET /kappa</td>
<td>List all kappas</td>
</tr>
</tbody>
</table>
Architecture

- Web service with REST API
- FaaS script defined as a component
- Runtime with two special actors as bridge to API
  - kappa.Source
  - kappa.Sink
- Wrap component in source, collect, and sink

<table>
<thead>
<tr>
<th>Command</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>PUT /kappa</td>
<td>Deploy script in body, return reference</td>
</tr>
<tr>
<td>POST /kappa/ref</td>
<td>Send JSON data to kappa inport</td>
</tr>
<tr>
<td>GET /kappa/ref</td>
<td>Read JSON data from kappa outport</td>
</tr>
<tr>
<td>DELETE /kappa/ref</td>
<td>Delete a kappa</td>
</tr>
<tr>
<td>GET /kappa</td>
<td>List all kappas</td>
</tr>
</tbody>
</table>
Architecture

- Web service with REST API
- FaaS script defined as a component
- Runtime with two special actors as bridge to API
  - kappa.Source
  - kappa.Sink
- Wrap component in source, collect, and sink
- Deploy, communicate with, and destroy kappa

<table>
<thead>
<tr>
<th>Command</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>PUT /kappa</td>
<td>Deploy script in body, return reference</td>
</tr>
<tr>
<td>POST /kappa/ref</td>
<td>Send JSON data to kappa inport</td>
</tr>
<tr>
<td>GET /kappa/ref</td>
<td>Read JSON data from kappa outport</td>
</tr>
<tr>
<td>DELETE /kappa/ref</td>
<td>Delete a kappa</td>
</tr>
<tr>
<td>GET /kappa</td>
<td>List all kappas</td>
</tr>
</tbody>
</table>
Example Use Cases
Example Use Cases

Introspection
Visualize running system, log-on-condition, etc.
Example Use Cases

Introspection
Visualize running system, log-on-condition, etc.

μService creation
Deploy local control loops, use API to set desired state
Example Use Cases

Introspection
Visualize running system, log-on-condition, etc.

μService creation
Deploy local control loops, use API to set desired state

Large area sampling
Statistically sound sampling of data without violating privacy (e.g. indoor temp)
Serverless IoT Revisited

If you can't be a first class cloud citizen, at least be a first class neighbour