Programming Languages to Runtime Execution: A Continuum

Red Hat Research Day Europe
15/09/2022

Frédéric Le Mouël*, Patrik Fortier*, Arthur Navarro*, Julien Ponge*, Clément Escoffier*

* University of Lyon, INSA Lyon, Inria
° Red Hat
Lyon, France?

- 2h from Paris by train
- 1.4M inhabitants
- 2nd economical region in France
- Region, famous for:
  - Gastronomy
  - Metallurgy sector
  - Chemical sector
  - Healthcare, pharmaceutic sector
  - Electric, electronic sector
  - IT services sector
CITI Lab

- Topic: Telecommunications
- ~130 members (~40 professors), 8 teams
- Common Labs: Nokia Bell, Orange - Chair: SPIE ICS
- 12 platforms: EquipEx FIT/CorteXLab, EquipEx+ TIRREX
- 3 spinoff startups
- 9.6M€ turnover, 4.3M€ revenue generated in 2022
CorteXLab Platform
Scientific project « Reconnected Society »
Scientific project « Reconnected Society »

- Emeraude
- Radiocom
- Networks
- Embedded Systems
- Robots
- Chroma
- Dynamid
- Maracas
- Systems Modeling
- Complex Systems
- Large-scale Systems
- Mobility
- WSN/RAN
- M2M Networks
- Smart City
- Smart Building
- Drone Fleet
- Population
- Resources
- Governance
- Security & Privacy
- Wired
- Privatics
- Agora
- Phenix
- Collaborative Systems
- Emeraude
Scientific project « Reconnected Society »
Scientific project « Reconnected Society »

- Emeraude
- Radiocoms
- Maracas
- Networks
- Agora
- Privatics
- Wired
- Distributed Systems
- Phenix
- Chroma
- Dynamid

- Complex Systems Modelling
- Deep/Reinf Learning
- Quantum Comm
- Large-scale Systems
- Collaborative Systems
- Fog/Edge
- Zero-Energy IoT
- 6G
- M2M Networks
- WSN/RAN
- Smart City
- Smart Building
- Drone Fleet
- Mobility
- Population
- Resources
- Governance

INSA
Université de Lyon
Scientific project « Reconnected Society »
Frédéric Le Mouël

- Professor, head of CITI Lab
- Chair in Internet of Things
- Chair in Edge AI
- Topics: Distributed Systems, Edge/Fog Computing
- Industrial use cases: IT sector (SPIE ICS, Red Hat), Automotive sector (VALEO, Stellantis)
- Twitter: @flemouel
Smart Cars
Mobile Cloud Computing
Autonomous Robotics
Smart Cars
Dynamic Environments
Mobility
Scalability
Energy
Home Automation
Green Buildings
Smart Cities
Internet/Web of Things
Programming the Move!

I Adaptive in the Small

- Adaptive Virtual Machine

II Autonomous and Collaborative in the Large

- Long-life Software-Oriented Architectures
- Bio-inspired Algorithms
- Distributed Decision
- Context-aware DSL
Automatic Service Deployment

Moving with your services and ... what if you switch from a Kubernetes infrastructure to a Mesos one?
Automatic Service Deployment

- Proposal: a DSL to manage and automatise your deployment
- Service generation according to the targeted infrastructure
- Synchronization avoiding SPOF

<table>
<thead>
<tr>
<th>Abstraction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>@Service</td>
<td>Services are defined like function annotated. Dyninka generates a service with the same capabilities than the function. The annotation @Service is used to identify functions from which we generate services.</td>
</tr>
<tr>
<td>@Placed(val location: String)</td>
<td>User-defined local variable. The annotation @Placed(val location: String) is used to place variables. This technique is used for introducing local memory to normally stateless functions and services.</td>
</tr>
<tr>
<td>Abstracted synchronization</td>
<td>Generated services provides asynchronous constructs (Promise, Future) to perform dead-lock free communication.</td>
</tr>
<tr>
<td>Client side type safety</td>
<td>$T :=&gt; Future&lt;T&gt;$ Type safety performed on client side is respected when converting functions into asynchronous services.</td>
</tr>
</tbody>
</table>
Automatic Service Deployment

```kotlin
@Placed("get") val vertx: Vertx = Vertx.vertx()

@Placed("movingAvg") var counter: Int = 0

@Placed("movingAvg") var movingAvr: Float = 0F

@Service fun get(): Future<JsonObject>
{
    return WebClient.create(vertx)
        .get("sensor","/").expect(SC_OK)
        .send().map {
            it.bodyAsJsonObject()
        }
}

@Service fun Future<JsonObject>.movingAvg(): Future<JsonObject>
{
    val promise = Promise.promise<JsonObject>()
    this.onSuccess {
        val next = it.getFloat("Value")
        movingAvr = (next + counter * movingAvr) / (counter + 1)
        counter++
        val result = JsonObject().put("Average", movingAvr)
        promise.complete(result)
    }
    return promise.future()
}

fun main()
{
    get().filter().map().movingAvg()
}
```
Automatic Service Deployment

@Placed("get") val vertx: Vertx = Vertx.vertx()

@Placed("movingAvg") var counter: Int = 0

@Placed("movingAvg") var movingAvr: Float = 0F

@Service fun get(): Future<JsonObject>
{
    return WebClient.create(vertx)
        .get("sensor","/").expect(SC_OK)
        .send().map {
            it.bodyAsJsonObject()
        }
}

@Service fun Future<JsonObject> .movingAvg(): Future<JsonObject>
{
    val promise = Promise.promise<JsonObject>()
    this.onSuccess {
        val next = it.getFloat("Value")
        movingAvr = (next + counter * movingAvr) / (counter + 1)
        counter++
        val result = JsonObject().put("Average", movingAvr)
        promise.complete(result)
    }
    return promise.future()
}

fun main()
{
    get().filter().map().movingAvg()
}
Automatic Service Deployment

@Placed("get") val vertx: Vertx = Vertx.vertx()

@Placed("movingAvg") var counter: Int = 0

@Placed("movingAvg") var movingAvr: Float = 0F

@Service fun get(): Future<JsonObject>
{
    return WebClient.create(vertx)
        .get("sensor","/").expect(SC_OK)
        .send().map {
            it.bodyAsJsonObject()
        }
}

@Service fun Future<JsonObject>.movingAvg(): Future<JsonObject>
{
    val promise = Promise.promise<JsonObject>()
    this.onSuccess {
        val next = it.getFloat("Value")
        movingAvr = (next + counter * movingAvr) / (counter + 1)
        counter++
        val result = JsonObject().put("Average", movingAvr)
        promise.complete(result)
    }
    return promise.future()
}

fun main()
{
    get().filter().map().movingAvg()
}
Automatic Service Deployment

```
@Placed("get") val vertex: Vertx = Vertx.vertex()

@Placed("movingAvg") var counter: Int = 0

@Placed("movingAvg") var movingAvr: Float = 0F

@Service fun get(): Future<JsonObject> {
    return WebClient.create(vertex)
        .get("sensor","/").expect(SC_OK)
        .send().map {
            it.bodyAsJsonObject()
        }
}

@Service fun Future<JsonObject>.movingAvg(): Future<JsonObject> {
    val promise = Promise.promise<JsonObject>()
    this.onSuccess {
        val next = it.getFloat("Value")
        movingAvr = (next + counter * movingAvr) / (counter + 1)
        counter++
        val result = JsonObject().put("Average", movingAvr)
        promise.complete(result)
    }
    return promise.future()
}

fun main()
    get().filter().map().movingAvg()
```
Automatic Service Deployment

@Place("get") val vertex: Vertx = Vertx.vertex()

@Place("movingAvg") var counter: Int = 0

@Place("movingAvg") var movingAvr: Float = 0F

@Service fun get(): Future<JsonObject> {
    return WebClient.create(vertex)
        .get("sensor","/").expect(SC_OK)
        .send().map {
            it.bodyAsJsonObject()
        }
}

@Service fun Future<JsonObject>.movingAvg(): Future<JsonObject> {
    val promise = Promise.promise<JsonObject>()
    this.onSuccess {
        val next = it.getFloat("Value")
        movingAvr = (next + counter * movingAvr) / (counter + 1)
        counter++
        val result = JsonObject().put("Average", movingAvr)
        promise.complete(result)
    }
    return promise.future()
}

fun main() {
    get().filter().map().movingAvg()
}
Automatic Service Deployment

```kotlin
@Placed("get") val vertex: Vertx = Vertx.vertex()

@Placed("movingAvg") var counter: Int = 0

@Placed("movingAvg") var movingAVr: Float = 0F

@Service fun get(): Future<JsonObject>
{
    return WebClient.create(vertex)
        .get("sensor","/").expect(SC_OK)
        .send().map {
            it.bodyAsJsonObject()
        }
}

 [...]
Automatic Service Deployment

(a) Dyninka

(b) OpenFaaS

1 worker
Automatic Service Deployment

**OpenFaas**
- min=0.2581, median=1.2174, max=12.8269, 3108 events, success=5.68%

**Dyninkka**
- min=0.6666, median=1.3379, max=2.5332, 2204 events, errors=0.0%

10 workers
On-going projects

- Automatic service deployment in dynamic environments
- Service re-orchestration
- Service virtualization in automotive applications
Future works?
Edge AI:  
Data Flow Analysis in Edge Infrastructures

Chair INSA Lyon - SPIE ICS

Frédéric Le Mouël (INSA Lyon)
AI & Data Flow
AI & Data Flow

- Sensing as a Network Service
- Network-traffic patterns in CDN
- Dynamic, Adaptive, Mobile, Multi-Tenant Slicing for E2E QoS
AI & Data Flow

• Sensing as a Network Service
• Network-traffic patterns in CDN
• Dynamic, Adaptive, Mobile, Multi-Tenant Slicing for E2E QoS
Embedded AI
Embedded AI

- Low carbon & energy-footprint AI - Learning under Resource Constraints
- Decision-Making under Uncertainty
- Intermittent Reinforcement Learning
Embedded AI

- Low carbon & energy-footprint
- AI - Learning under Resource Constraints
- Decision-Making under Uncertainty
- Intermittent Reinforcement Learning

Anticipating Devices
Smart Routers
Predictive Maintenance
Geo-distributed AI
Geo-distributed AI

- Service Handover in Edge/Fog Computing Infrastructures for ML applications
- Low-Energy Federated Learning
- Fault-tolerant CNN & Generative Adversarial Learning
Geo-distributed AI

- Service Handover in Edge/Fog Computing Infrastructures for ML applications
- Low-Energy Federated Learning
- Fault-tolerant CNN & Generative Adversarial Learning
Responsibility & Data Control
Responsibility & Data Control

- Trust & Privacy
- Sovereignty
- Embedded Ethics
Responsibility & Data Control

- Trust & Privacy
- Sovereignty
- Embedded Ethics
CITI Laboratory
http://www.citi-lab.fr
Head: Frédéric Le Mouël (frederic.le-mouel@insa-lyon.fr)
@flemouel