A Context Model and Policies Management Framework for Reconfigurable-by-design Distributed Applications

Cloud Forward 2016, Madrid, Spain

Panagiotis Gouvas, Eleni Fotopoulou, Anastasios Zafeiropoulos, Constantinos Vassilakis

UBITECH Ltd., R&D Department

contact: azafeiropoulos@ubitech.eu
Evolving applications requirements

• Application requirements are changing
  • horizontal scalability;
  • elasticity;
  • adaptability;
  • resiliency;
  • fault-tolerance characteristics.

• The design of reactive systems that are able to adapt based on their operational environment conditions is required.
Software engineering approaches

• Design of reactive systems that are able to adapt based on their operational environment conditions.

• Include more “context-awareness” into applications and services.

• Independently orchestratable software components.

• Development and operations teams have to work closely together.
Network Softwarization and Programmable Infrastructure

• Novel virtualisation technologies.

• Packaging of software components in virtual machines (VMs), containers or unikernels.

• Network softwarization: providing network functionalities via software
  • Network Function Virtualization (NFV): service chains consisted of virtual network functions;
  • Software Defined Networking (SDN): network control directly programmable and underlying infrastructure abstracted for applications and network services.
Need for alignment

• Novel software engineering approaches.
• Solutions for optimally deploying and managing applications.
ARCADIA in a nutshell...

DISTRIBUTED APPS DEVELOPMENT ENVIRONMENT

POLICY-AWARE ORCHESTRATION
Optimal app deployment using meta-heuristic algorithms. Enable real-time policy enforcement. ETSI NFV compliant.

MULTI DATACENTER EXECUTION MIDDLEWARE

COMPONENT & APPLICATION DISCOVERY
Example: A 4-tier application network

A demand for App. execution containers, links and VFs represented as flow graph

Each element has to be accompanied by a tuple describing resource capacity requirements e.g. for T1 (CPU: a, Mem: y, I/O storage: z, Storage: i)

ARCADIA
Distributed Application

Example: A 4-tier application network

Horizontal scaling
Software Development in ARCADIA

Annotations are a form of metadata that provide data about a program that is not part of the program itself.

@ArcadiaMetric
(name="averageProcessingTime",
description = "URL hashing algorithm performance",
unitofmeasurement = "msec",
valuetype = ValueType.SingleValue,
maxvalue = "6000",
minvalue = "1",
higherisbetter = false)
Web-based Development Environment

• ARCADIA IDE plug-in is integrated with the latest version of Eclipse Che that is the browser-based, cloud version of the classic Eclipse.

• Through the plug-in, developers can:
  • manage their previously generated API keys;
  • have a pre-compile validation of the ARCADIA annotations and developed microservices;
  • submit their code to the platform.
Component and Service Graph Repositories and Composer

- Provide access to components developed within the Web-based IDE.
- View components details (configuration details, chainable parameters, metrics)
- Provide access to set of available and running service graphs.
- View set of components per graph.
- Manage service graphs (deploy, undeploy, monitor)
**Policies Enforcement Mechanisms**

- Policies enforcement during deployment (optimization engine) and runtime (rule based management system).

- During runtime:
  - Design and apply policies for runtime management of service graphs.
  - Context-aware execution of components and graphs taking into account conditions in the deployed ecosystem.
  - Assure QoS and QoE levels to end users.
  - Prioritize services/applications provision on behalf of the Service Providers.
Policies Scope Overview

• A policy may be associated with a service graph and applied during runtime.

• Set of actions for:
  • Component lifecycle management
  • Manage component configuration parameters/metrics
  • Activate/deactivate virtual functions (e.g. scaling)
  • Manage allocated IaaS resources
  • Trigger alerts

• Conflict resolution based on specification of priorities.
Drools is a Business Rules Management System

• A Production Rule is a two-part structure: the engine matches facts and data against Production Rules to infer conclusions which result in actions.

• Facts are asserted into the Working Memory where they may then be modified or retracted.

• The process of matching the new or existing facts against Production Rules is called pattern matching, which is performed by the inference engine.

• Actions execute in response to changes in data - data driven approach to reasoning.

• The Agenda manages the execution order of conflicting rules.
Why Drools?

• **Declarative definition of the business rules**, that can be expressed using a domain specific language that platform users can validate and understand.

• The analysis of true statements are not done sequentially, allowing you to **analyze multiple conditions much faster**.

• Drools Fusion give more possibilities in terms or **dynamic creation and reloading of user-defined rules**.

• **Interoperability with Optaplanner** constraint satisfaction solver.
Policies Enforcement Framework

• Follow a continuous **match-resolve-act** approach.
  ◦ match phase: mapping of the set of applied rules which are satisfied based on the data streams coming from a set of monitoring probes,
  ◦ resolve phase: conflict resolution -if any- among the satisfied rules taking into account the pre-defined salience of each rule,
  ◦ act phase: provision of a set of suggested actions to the orchestration components.

• **Data monitoring and management** processes are supported through a set of active and passive monitoring probes. Data is transformed to facts.

• **Definition of rules per policy** is supported through the Policy Editor in a **per service graph basis**, based on the concepts represented in the Context model.
Policies Enforcement Framework

ARCADIA Context Model

Policies Editor

Production Memory (set of policy rules)

Inference Engine
- Pattern Matching
- Conflict Resolution

Working Memory (facts)

Analysis Engine (e.g. Spark, R)

Execution Manager

Deployment Manager
- Optimisation Framework

Database (e.g. MongoDB)

Data Aggregation and Filtering

Pub/Sub Framework

Physical Resources (Compute, Storage, Network)

Service Graph Network Metrics

Components Runtime Metrics (Resource Usage)

Graph and Component specific metrics
Indicative Policy

//Scale Horizontally Apache Web Server 100% (add one instance)
if average CPU usage is greater than 70% the last minute

rule "Scale Horizontally Apache Web Server 100%"
salience 10
When
$m : MonitoredComponent( name== "Apache_Web_Server" && metric== "avg_CPU_Usage" && value>=70 )
from accumulate(
  $m.value over window:time( 1m ), average( $m.value))
then
  horizontalScale($m,100%)
end

//Scale Horizontally Apache Web Server 30% (based on the total number of instances) if average response time is greater than 0.2 seconds the last minute

rule "Scale Horizontally Apache Web Server 30%"
salience 9
When
$m : MonitoredComponent( name== "Apache_Web_Server" && metric== "avg_response_time" && value>=0.2 )
from accumulate(
  $m.value over window:time( 1m ), average( $m.value))
then
  horizontalScale($m,30%)
End

19 October 2016
Cloud Forward 2016, Madrid, Spain
Deploying a distributed application to a cloud infrastructure requires assigning and instantiating an execution environment for each software component while illustrating the communication links among them as required.

- Assignment of resources to execution environments and communication links has to:
  - Fulfil requirements
  - Satisfy objectives
  - Avoid policy violations
- Initial Assignment happens on a request to deploy and operate a new application
- Partial Reassignments per application are triggered during operation on required scaling to cope with workload or on required migration to keep on satisfying requirements.
- Full or Partial Reassignments for one or more applications are triggered for operating distributed applications in order to keep satisfying or better satisfy objectives and avoid policy violations.
Implementation

• OptaPlanner is a **constraint satisfaction solver**. It optimizes business resource planning.

• OptaPlanner is a **lightweight, embeddable planning engine**. It enables normal Java™ programmers to solve optimization problems efficiently. Constraints apply on plain domain objects and can reuse existing code.

• OptaPlanner combines sophisticated **optimization heuristics and metaheuristics** with very efficient score calculation.

• OptaPlanner is **open source software** (100% pure Java™, runs on any JVM)
Next Steps

• Provide suggestions –through runtime policies- to Optimisation mechanisms.
  • Integration with OptaPlanner.

• Mechanisms for partial conflict resolution during design time.

• Data aggregation and filtering mechanisms extension.

• Interconnection with analysis toolkit.

• Performance evaluation of the Policies Enforcement Mechanisms.
Thank you!!!

contact: azafeiropoulos@ubitech.eu

When: Thursday 20 October 2016,
10:20-11:10,
Room: Ramon Gomez

http://www.arcadia-framework.eu/

https://twitter.com/eu_arcadia

https://www.linkedin.com/groups/6949809