Real-Time QoS Control for Service Orchestration

Joost Bosman
Hans van den Berg (TNO)
Rob van der Mei

Overview:
1. Motivating example
2. Background
3. Model & approach
Powder snow trip advisor
Example of Orchestrated Service - Offpiste skiing trip advisor (OTA)-

- OTA service is composed of a multitude of sub-services, offered by third parties
- OTA crosses multiple administrative domains

**Question:** how do we realize desired end-to-end QoS levels?
Service Level Agreements

Typical deal for client-server relation:

→ service domain provides guarantees about service-level parameters
← “client” accepts deal and price

**Example:**
- average RT < 3 seconds
- Prob \{RT > 5 seconds\} < 1%
- availability > 99%
- price per service invocation = $1

- Multiple classes: Gold – Silver – Bronze
- SLA monitoring
- SLA enforcement and SLA violation
- “Negotiable” versus “non-negotiable” QoS parameters
Typical question:
What combinations of SLA’s lead to desired QoS levels? And what’s the cheapest?
**Subcontractor View**

In a way this is a recursive structure

**Question**: QoS1 “+” QoS2 “+” QoS3 = ??

**QoS calculus**: response times, availability, throughput

Concept of “SLA negotiation space”
Workflows

• A composition schema specifies the “process” of the composite service
  – the “workflow” of the service

• Different clients, by interacting with the composite service, satisfy their specific needs (reach their goals)
  – a specific execution of the composition schema for a given client is an orchestration instance
Workflows

- Services are rarely used in isolation
- Usually, they form the building blocks for more complex applications

A workflow is a collection of tasks and their dependencies
Calculations with Workflows

Sequential

Probabilistic

p

1-p

Loop

Parallel

fork

join

Use results form Stochastic Activity Networks (SANs)

Mapping fork-, or- and prob- constructions

Availability/reliability: straightforward calculus

Response times: method by Choudhury and Houck
Choudhury and Houck

- Response time random variable $D$
- CDF: $F_D(t) = P(D \leq t)$
- Discretize RT distributions with stepsize $h$, index $k \geq 0$:
  \[
  q^D_k = \begin{cases} 
  F_D(h[k + \frac{1}{2}]) - F_D(h[k - \frac{1}{2}]) & k < T \\
  F_D(h[k - \frac{1}{2}]) & k \geq T
  \end{cases}
  \]
- Numerical convolutions:
  \[
  D_1 * D_2 \rightarrow q^{D_1*D_2}_k = \sum_{j=0}^{k} q^{D_1}_j q^{D_2}_{k-j}
  \]
Service Composition

Task A
Service A
  t: time  p: price
Service B
  t: time  p: $$$$

Task B
Service K
  t: time  p: $$$$$

Task 1
Service X
  t: time  p: $$$$$
Service Y
  t: time  p: $$(

Task 2

Task 3

The workflow’s QoS?

1st COST ACROSS Open Workshop
• **Types of web services:**
  - **atomic** Web Services - represent a **single activity** in a business process
  - **composite** Web Services – represent more complex business logic often encompassing multiple business steps
Web Services Composition Problem

- Linear workflow structure (for simplicity)
- Composition is a path from A to B
- Reference versus alternative compositions
- Static, dynamic or real time dynamic (different levels)
Service Composition

Composition at design time:
• Choices about the composition are based **upfront**
• **Inflexible:** Impossible to modify composition on-the-fly
• Simple

Composition at execution time:
• Choices about the composition are made on-the-fly
• Flexible
• Complicated
Orchestrator/Composite Service Provider (CSP)

The CSP
- knows the workflow (could be very complex)
- selects appropriate services
- makes appropriate Service Level Agreements (SLAs) with 3rd party providers and its clients
- has no impact to or control of third party domains
- knows the services’ response times

In most papers: once selected, the composition is fixed
Model Description

Fully dynamic runtime re-composition:

- Composition may be adapted during execution
- Use of elapsed time info
- For each task, there may be alternatives
cSLA (end-to-end)

- $\delta_p$ e2e RT deadline
- R reward for response time $\leq \delta_p$
- V penalty for response time $> \delta_p$
End-to-end 
Response Time Deadlines

- $\leq$ deadline $\rightarrow$ reward $R$

- $>\text{ deadline}$ $\rightarrow$ penalty $V$

$\delta_p$ e2e RT deadline

1st COST ACROSS Open Workshop
iSLA

- RT of service $S_{i,j}$ is represented by random variable $D_{i,j}$
- $f_{D_{i,j}}$ response time (RT) PDF of service $S_{i,j}$
- $c_{i,j}$ cost of execution
Dynamic Re-Composition Decisions

Decision: what service alternative to make at task i, based on the elapsed response time?

\[ \delta_p \] e2e RT deadline

Elapsed time

Response time budget \( B \) for \( i = 2 \)
Optimized Dynamic Decisions

- **Given:**
  - position in workflow $i$
  - Time until deadline violated $B$
  - Response time distributions $f_{D_{i,j}}$
  - Costs $c_{i,j}$, Reward $R$ and Penalty $V$

- **Problem:**
  - Optimize expected profit $E[R]$

- **Solution:**
  - Apply Dynamic Programming
  - Backward recursion (Bellman equation)
Dynamic program

- For each service $S_{i,j}$:
  - Stoch. variable: $D_{i,j}$
  - PDF: $F_{D_{i,j}}(t) = \Pr(D_{i,j} \leq t)$
  - CDF: $f_{D_{i,j}}(t) = \frac{d}{dt}F_{D_{i,j}}(t)$

- Remaining budget $B$, workflow position $X$

- Conditional profit expectation:
  - $E[R|B = b, X = N + 1] = \begin{cases} R & \text{if } b > 0 \\ -V & \text{otherwise} \end{cases}$
  - $E[R|B = b, X = i] = \max_j \left\{ -c_{i,j} + \int_0^b f_{D_{i,j}}(t)E[R|B = b - t, X = i + 1]dt \right\}$

- Expected profit: $E[R|B = \delta_p, X = 1]$
Approach:

- **Simple** solution
- Calculate lookup table **off-line**
- Apply lookup table **on-line** (no computing)
- Closed-loop control possible based on monitoring
Closed-Loop Control

- For each significant change
- Calculate/update empirical distribution(s)
- Apply backward recursion on empirical distributions
Closed-loop Control approach

1. Execute DP
2. Update response time info
3. Probe time expired?
   - No
     - Get DP reference & current info
     - Test distribution change
       - Significant change detected
         - Store DP reference info
       - No significant change detected
         - Update response time info
3. Yes
   - Send probe
4. Response time realizations $d_{n(i,j)}$
5. Execute DP
6. Calculate DP
7. Store DP reference info

1st COST ACROSS Open Workshop
Empirical distribution

- **RT stochastic variable** \( D \)
- **\( N \) Realizations** \( d_1, d_2, \ldots, d_N \)
- **Empirical distribution** 
  \[
  F_{\hat{D}}(t) = \frac{1}{N} \sum_{j=1}^{N} 1\{d_j \leq t\}
  \]
- **Discretized empirical distribution (histogram):**
  \[
  q_{\hat{D}}^k = \frac{1}{N} \sum_{j=1}^{N} 1\{h(k-0.5) \leq d_j < h(k+0.5)\}
  \]
Tracking change

- **Sliding window** $W$

\[
q_k^\tilde{D} = \frac{1}{W} \sum_{j=0}^{W-1} 1 \{ (k-0.5) \leq d_{j+t} < (k+0.5) \}
\]

- **Exponential smoothing** $\alpha$

\[
q_k^\tilde{D} = (1 - \alpha) q_{k-1}^\tilde{D} + \alpha 1 \{ (k-0.5) \leq d_t < (k+0.5) \}
\]

  - No bookkeeping of realization necessary!

  - Virtual sliding window $W = \frac{1+\alpha}{1-\alpha}$
Statistical Test

• Modified Kolmogorov Smirnov test:
  
  \[ D_W := \sup_k |q_k - \tilde{q}_k| \]
  
  – Where \( k \) is the discretization index and \( D_W \) is the statistic
  
  – For the smoothing approach choose:
  
  \[ W = \frac{1 + \alpha}{1 - \alpha} \]
Probes

• Certain services are unattractive
  – May not invoked at all
• Probe
  – If service not visited for $t_p$ times then send a dummy request incurring probe cost
Test bed environment

- Test lab environment developed
- Based on JAVA + NIO
  - Event based, single threaded design
  - Real HTTP traffic
- Algorithms in Matlab using IP control interface
- Trigger mechanism to simulate “Sub service disasters”
- Currently running as multiple threads
- Future: Run on a multiple server test bed
Classical 'One thread per channel' model

- Not scalable
- Context switching overhead
Reactor IO model

- Selector handles IO in FIFO fashion

One Thread

- ServerSocketChannel
  - Incoming Request Handling
- SocketChannel
  - Response Handling
- SocketChannel
  - Dispatching Handling

Thread

Selector

Channel Channel Channel
Experiment design

Matlab (Algorithms and decision making)

Controller

Dispatcher & Registry

Switch

Load Generator

Service

Service

Service

Red - Command & Control (TCP)

Green - HTTP

Blue - JAVA interface
Test bed setup

Matlab

Orchestrator (composite service)
Dispatcher
Service registry + interface
Control interface

Load generator
Requestor
Control interface

Sub Service 1
Service interface
Control interface

Sub Service N
Service interface
Control interface

Algorithms run in Matlab
Real IP traffic

HTTP Connection
Experiment Control Connection

1st COST ACROSS Open Workshop
Demo

Swap services after 5000 requests

1st COST ACROSS Open Workshop
Results

- Swap service \((i,2)\) and \((i,3)\) at \(i = n_{swap}\) after 5000 requests
- Parameters:
  - \(W\) Window size
  - \(t_p\) Probe interval
  - \(\alpha\) Test significance level
  - \(n_{swap}\) Swap position (experiment parameter)
Window size $W$

$W = 25$  
$W = 50$  
$W = 100$

$n_{swap} = 1$

$n_{swap} = 3$

Swap services event

Recovery from swap event
Position in chain

Swap services event

\[ n_{\text{swap}} = 1 \quad n_{\text{swap}} = 2 \quad n_{\text{swap}} = 3 \quad n_{\text{swap}} = 4 \]
Probe interval $t_p$

 Swap services event

\[ t_p = 10 \quad t_p = 25 \quad t_p = 50 \quad t_p = 100 \]
Closed-loop Control

• **Challenges:**
  1. We do not prefer updating the policy after each realization.
  2. If a policy never selects a certain alternative we don’t observe changes

• **Solutions:**
  1. Apply statistical test to see whether an empirical distribution has changed significantly.
  2. If a service is not used for $t_p$ times send a probe request (and pay corresponding cost)
Tradeoffs

Window size for PDF updates:

• Big $W$: good estimate of distribution but slow response to changes
• Small $W$: poor estimate of distribution but quick response to changes

Frequency of probing for PDF updates:

• Small $t_p$: good updates of rarely used alternatives but higher cost
• Big $t_p$: poor updates of rarely used alternatives but lower cost

1st COST ACROSS Open Workshop
Possible Extensions

- Inclusion of **re-attempts** in case of excessive delays
- Combined stopping and alternative choices
- Take in account sample age (invocation time)
- Non-linear composition structures

### Conclusion:

- Dynamic workflow composition extremely powerful
- Many challenging open research questions