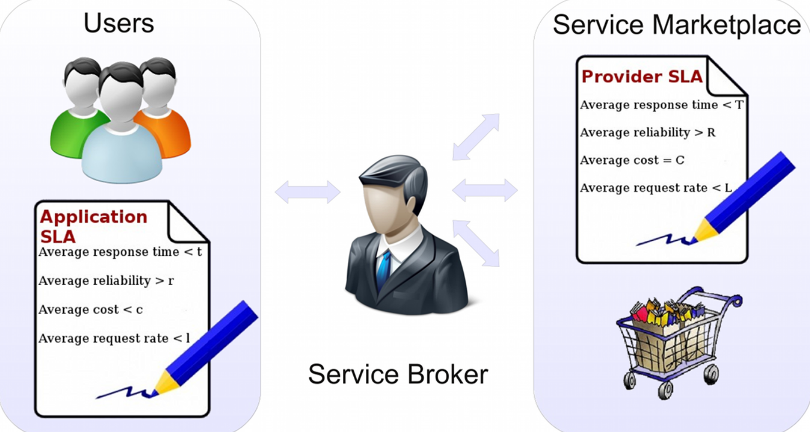
Improving SOA Applications Response Time with Service Overload Detection

Valeria Cardellini, Stefano Iannucci

{cardellini, iannucci}@ing.uniroma2.it

Department of Civil Engineering and Computer Science Engineering

University of Rome "Tor Vergata", Italy

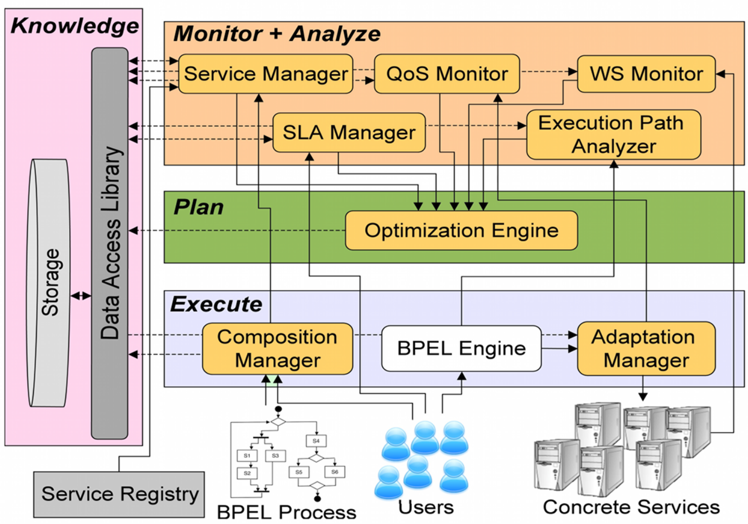


Introduction

- **SOA applications** are usually built by assembling third-party software services.
  - Each service can be characterized by a **Service Level Agreement (SLA)** stating its **Quality of Service (QoS)** properties.
  - SLA violations could occur, therefore a **monitoring** policy is needed to detect them.
- **MOSES [1]** (MOdel-based SElf adaptation of SOA systems): **QoS-driven runtime adaptation framework for service-oriented systems**
  - It acts as a Qos-enabled **service broker:**
    - It stipulates SLA with third-party services;
    - It stipulates SLA with users;
    - It provides a composite service with QoS guarantees.

In this poster we present a monitoring system capable of detecting **service state changes** through an online **adaptive Cumulative sum (Cusum) detector** implemented into **MOSES,** showing a 26% improvement of the SOA application response time.

MOSES Architecture



Adaptive Cumulative sum (Cusum) State Change Detector

Response time is estimated using an **EWMA filter:**

$$\mu_i = \alpha y_i + (1-\alpha)\mu_{i-1}$$

where y_i is the *i-th* collected response time sample

To detect state changes, Cusum [2] uses:

- two **accumulators g_i^+, g_i^-**
- a **threshold H^***

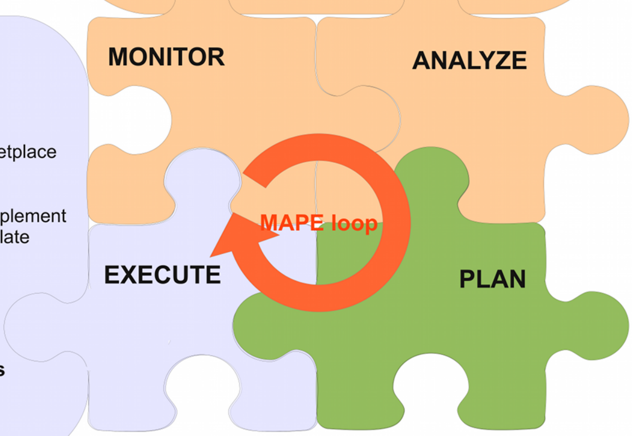
$$g_i^+ = max\{0, g_{i-1}^+ + y_i - (\mu_i + K^+)\}$$

$$g_i^- = max\{0, g_{i-1}^- + (\mu_i - K^-) - y_i\}$$

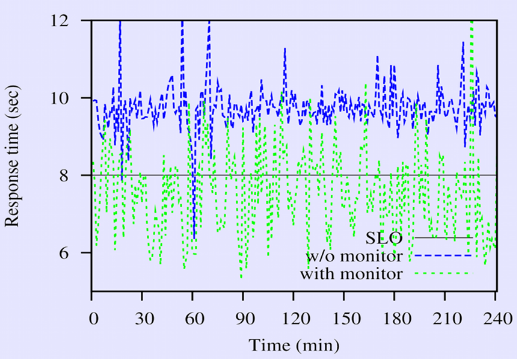
where K^+ (K^-) is the smallest shift we want to detect on the leading (trailing) edge.

Whenever g_i^+ or g_i^- exceeds H^* a new state is detected and a new optimal service selection strategy is computed using the predicted average response time:

$$\mu_i = \begin{cases} \mu_{i-1} + K + g_i^+/N^+ & \text{if } g_i^+ > H^* \\ \mu_{i-1} - K - g_i^-/N^- & \text{if } g_i^- > H^* \end{cases}$$



Experimental Results



Experimental setup

Node 1) Execute + Monitor

Node 2) Plan

Node 3) Analyze and Marketplace

Node 4) Client

Services in marketplace implement a M/D/m/PS queue to simulate CPU load

26% response time improvement using monitoring and analysis

REFERENCES

[1] V. Cardellini, E. Casalicchio, V. Grassi, S. Iannucci, F. Lo Presti, and R. Mirandola. MOSES: a framework for QoS driven runtime adaptation of service-oriented systems. *IEEE Trans. Softw. Eng.*, 2012. to appear

[2] S. Casolari, S. Tosi, and F. Lo Presti. An adaptive model for online detection of relevant state changes in internet-based systems. *Perform. Eval.*, 69(5), 2012.