Policy Learning for Adaptive Allocation of Cloud Resources to Multi-tier Web Applications

Waheed Iqbal,[1] Matthew N. Dailey,[1] David Carrera[2]

[1]Asian Institute of Technology (AIT) Thailand, [2]Technical University of Catalonia (UPC) Spain.

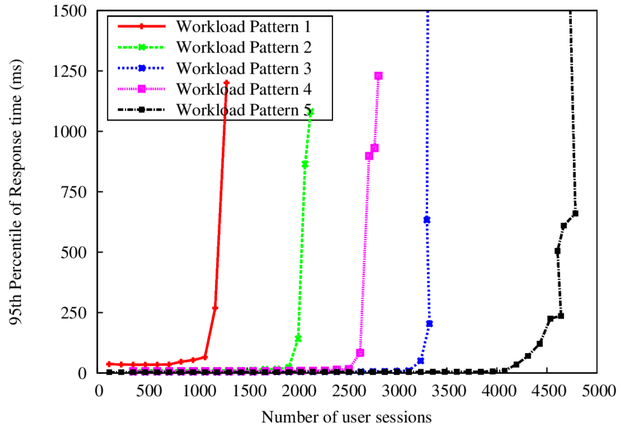
Motivation

Response time is an important quality attribute of a Web application.

Offering response time guarantees for **multi-tier Web applications** is challenging:

- a bottleneck tier affects all dependent tiers
- bottleneck locations depend on workload patterns
- the workload pattern is dynamic

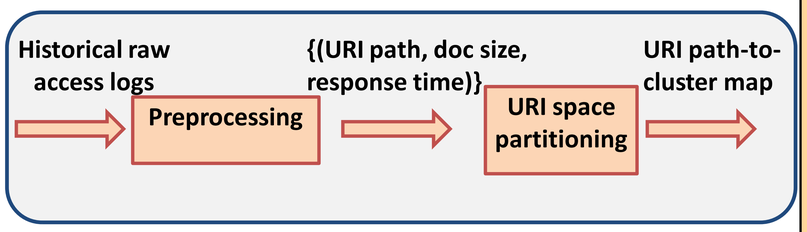
RUBiS benchmark application saturates at different workload levels under different workload patterns:



We propose a simple method to learn **workload specific** resource allocation **policies** to offer response time guarantees using **minimal resources**.

Workload Pattern Modeling

The workload pattern modeling method learns a clustering model:



For a specific time interval, the workload pattern is a vector containing the probability of each URI cluster.

Policy Learning

We use a greedy Q-learning method based on online observation.

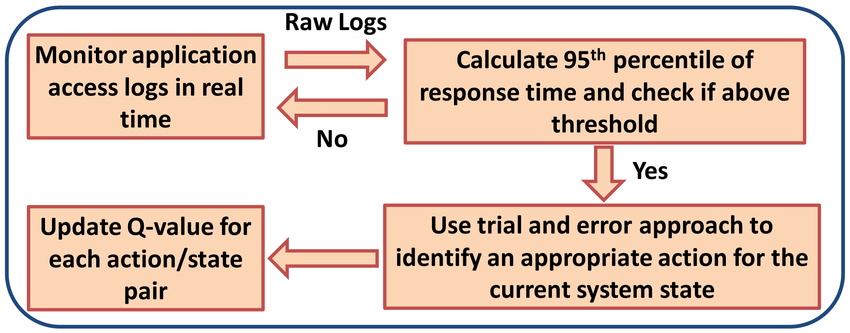
The **system state** consists of:
- the tier configuration
- the current workload pattern vector
- the current arrival rate
- the current response time

Possible **actions** for a two-tier application:
- scale up Web tier
- scale up database tier
- scale up both
- do nothing

The **environment** that the agent interacts with is a stochastic function mapping the current state and action to a new state.

The **reward function**:
- encourages the agent on success
- discourages the agent on failure to maintain the SLA
- discourages overprovisioning

Exploration phase block diagram:

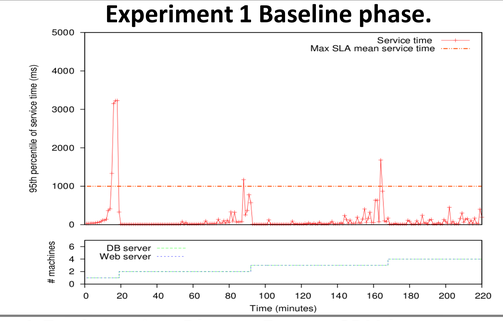


Experimental Evaluation

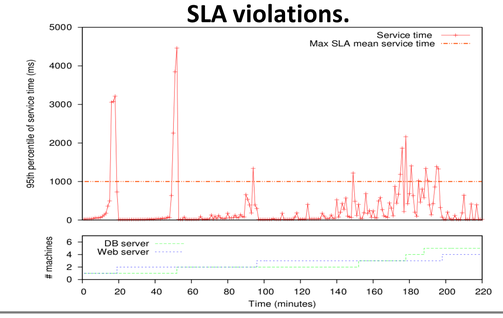
- EUCALYPTUS-based private testbed cloud
- Two-tier RUBiS benchmark Web application
- Two experiments executed in three phases:
  - **Exploration**: we initialize an empty policy and let the system learn in real time
  - **Exploitation**: the agent simply uses the previously-learned policy to automatically resolve bottlenecks
  - **Baseline**: scale up every replicable tier every time a bottleneck occurs

Exploitation Performs Better

Experiment 1 Baseline phase.



Experiment 1 Exploitation Phase. Fewer CPU hours, small increase in SLA violations.



Experimental results summary.

		Total allocated CPU hours	Percentage of requests violating SLA
Experiment 1	Exploitation	17.7	1.03
	Baseline	19.8	0.35
Experiment 2	Exploitation	24.7	1.75
	Baseline	27.0	0.233

Conclusion and Future Work

The proposed approach:

- enables us to learn **on-line** autoscaling policies for **multi-tier** Web applications
- helps us to offer **response time guarantees**
- **minimizes** resource allocation
- does not require any **prior knowledge** of the application
- minimizes the **overhead** needed to monitor, detect, and resolve bottlenecks

We are currently:

- **improving** the exploration phase
- integrating **scale-down** policy learning