Modèles de performance et émulation pour le dimensionnement autonome d'applications distribuées à base de composants

Introducing queuing network-based performance awareness in autonomic systems

(ICAS 2010)

Ahmed Harbaoui, Nabila Salmi

Bruno Dillenseger and **Jean-Marc Vincent**

Orange Labs, France Telecoms

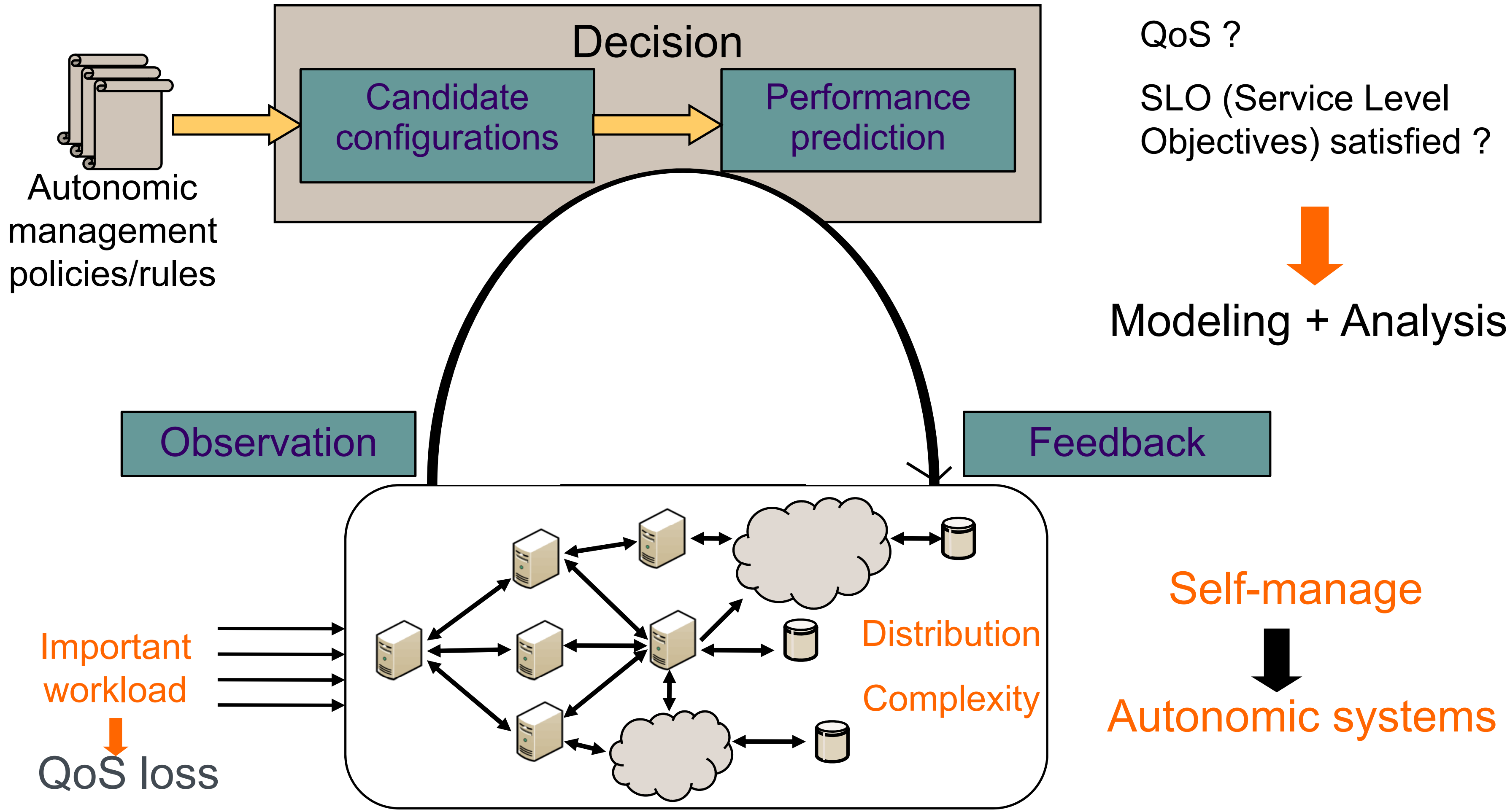
LIG/INRIA-MESCAL Team

Grenoble, France

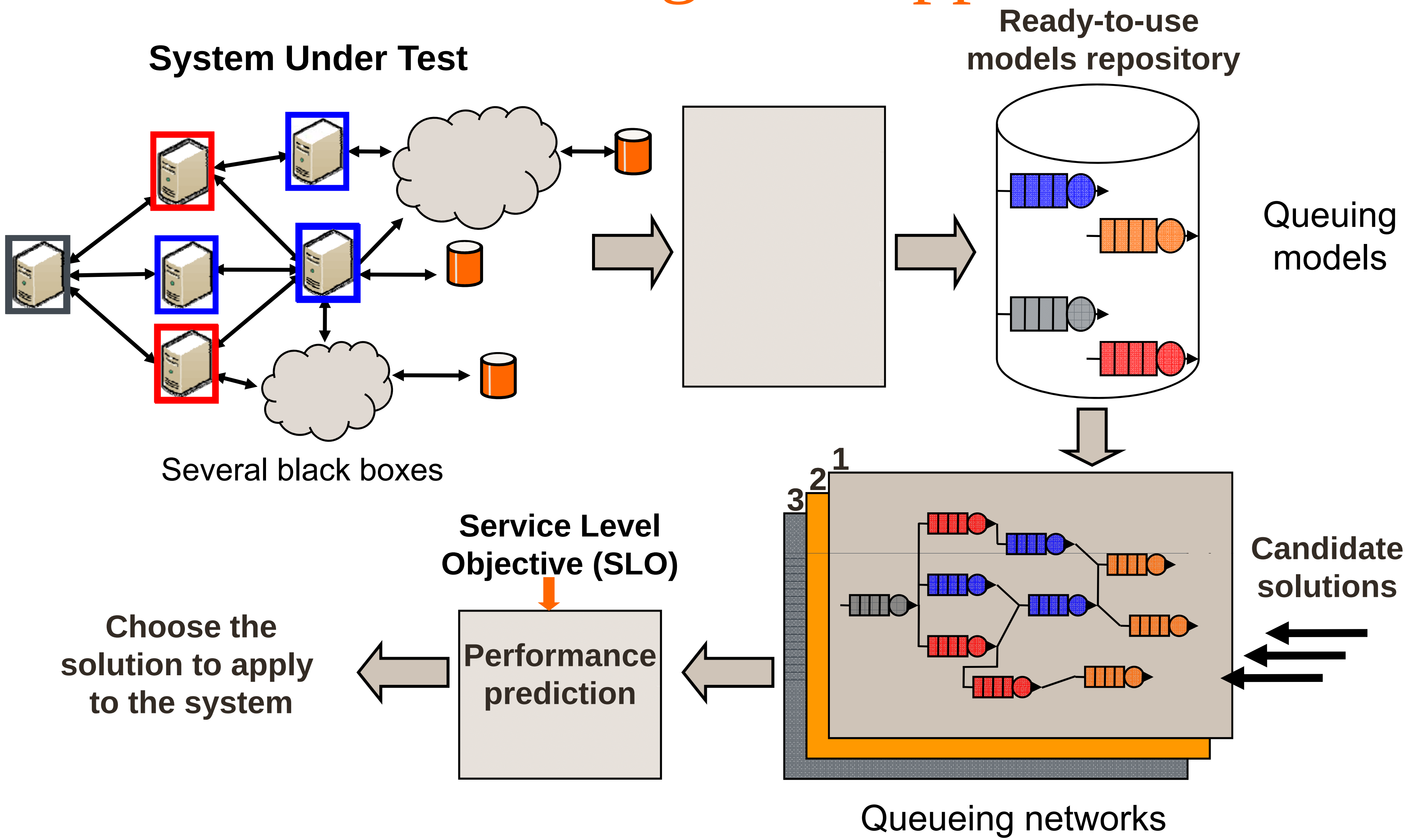
Outline

1. Towards autonomic management
2. Modeling black boxes
3. Automatic black box model identification
4. Experimental results
5. Conclusion & future work

Towards Autonomic management



Overview of our global approach



Automatic black box model identification

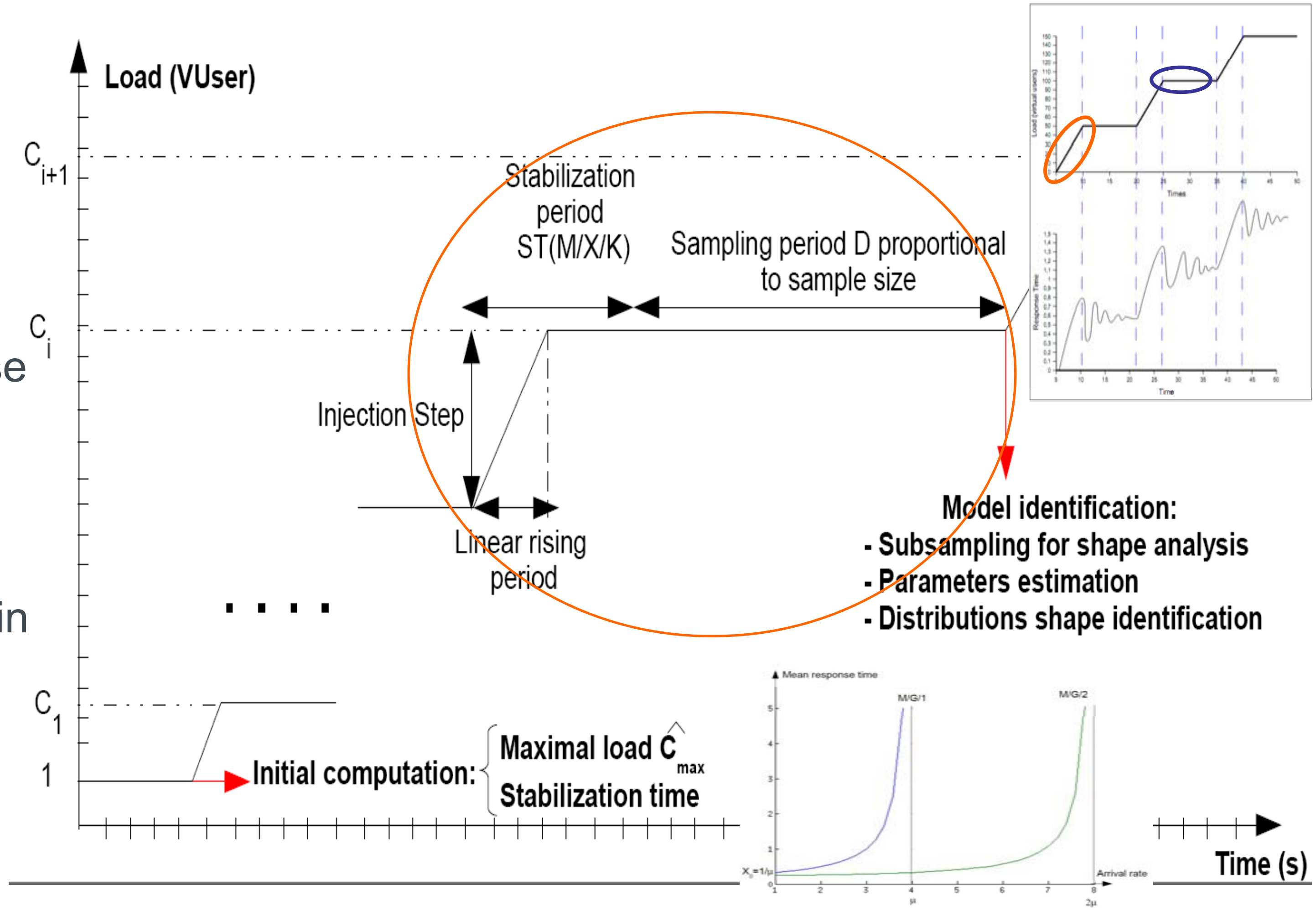
Goal: Capture the complete behavior of a black box and limit points

Principle

1. Several automatic load injection steps.

2. Start with a low injected load, increase load until black box saturation.

3. Collect measures in each step (response times, resources utilization), deduce a queueing model

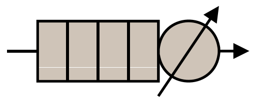


Outline

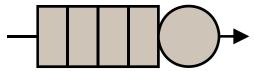
1. Towards autonomic management
2. Modeling black boxes
3. Automatic black box model identification
4. Experimental results
5. Conclusion & future work

Modeling :

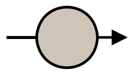
- Load-dependent black boxes : Queuing and service times depend on the load.



- Load-independent black boxes : The service time does not depend on the load.



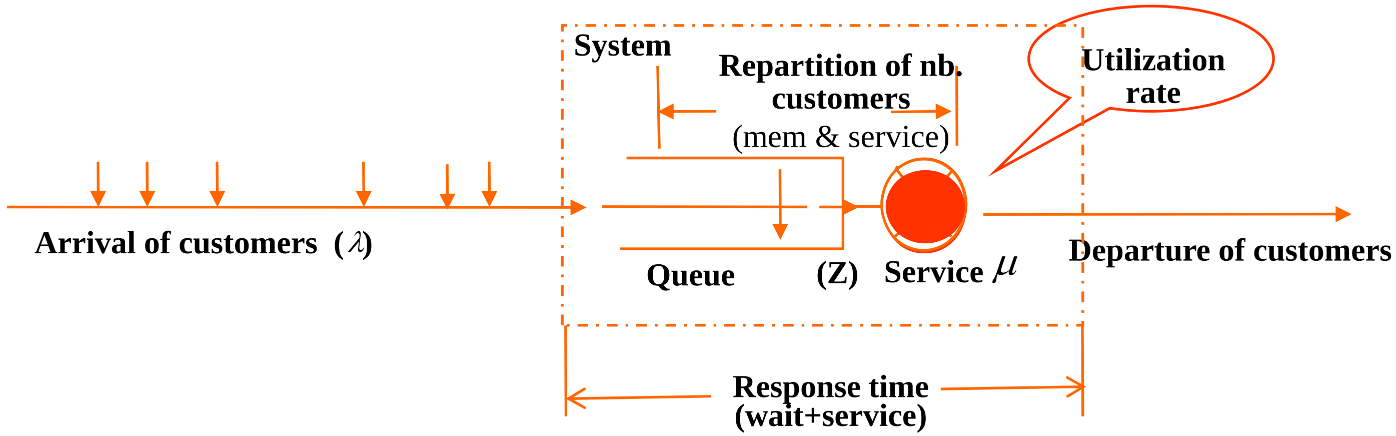
- Constant delay black boxes : Service time does not depend on the load and there is no queuing



We define the type of each black box according to the test results

Queuing models

Queue with 1 server



λ = **arrival rate of customers** : mean *number of arrivals per time unit*

μ = **service** rate : mean *number of served customers per time unit*

Z = scheduling **policy** : *FIFO, PS, RR, random, …*

Queue model : **T / X / m / K / Z**

Interarrival time distribution

Service time distribution

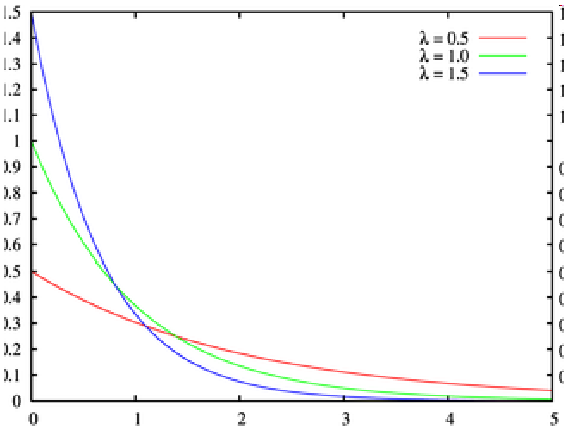
Queue Capacity

Number of servers

Queueing models

- <u>M/M/K model</u> : Inter-arrivals *Exponential(λ)*, Service *Exponential(μ)*, Infinite Capacity, K servers, *FIFO* → Analyzable by MVA algorithm

Exponential distribution (λ)



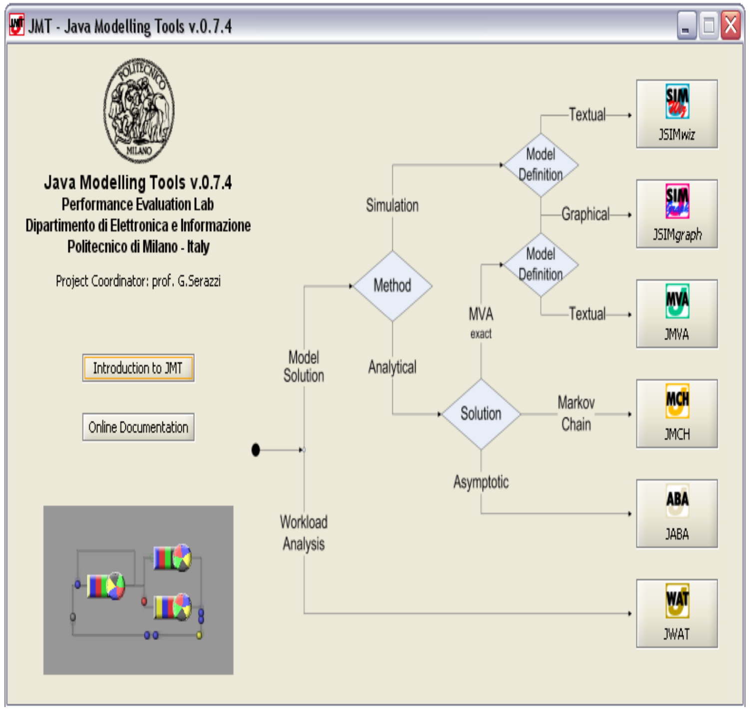
$$\text{Prob}\{T \le t\} = 1 - e^{-\mu t}, \quad t \ge 0$$

- <u>M/G/K model</u> : Inter-arrivals *Exponential(λ)*, *General* service, Infinite capacity, K servers, *FIFO* → Analyzable by R. Marie Algorithm

- <u>G/G/K model</u> : *General* Inter-arrivals, *General* service, Infinite capacity, K servers, *FIFO* →Simulable

Java Modelling Tool (JMT)

- Suite of tools developed by Politecnico di Milano, 2006 -2009



- Six Java applications:

1. JSIMGraph, JSIMWiz: QN models designer/simulator (graphical/ Wizard interface)

2. JMVA: Mean Value Analysis of BCMP compliant QN models

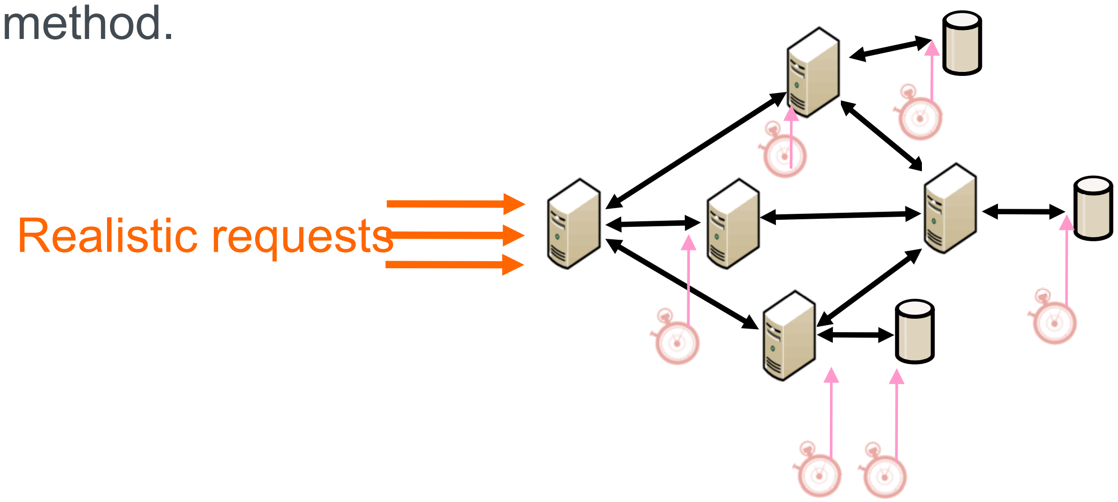
3. JABA: asymptotic analysis of QN models for identification of the bottlenecks

4. JWAT: Workload Analysis from log/ used data

5. JMCH : Markov chain (M/M/1, M/M/1/K models) simulator

Inter-arrival times distribution

- Collect arrival times of submitted requests
- Deduce the inter-arrival sample,
- Identify the shape of the inter-arrival sample →
  - use statistical tests against several distribution families: exponential, heavy-tail, etc (Kolmogorov-Smirnov test).
  - Keep distributions whose p-value > 0.1
  - Estimate distribution parameters with the Maximum likelihood estimator method.



Service time distribution

- Inject load requests with exponential inter-arrivals.
- Collect response times (R_k), inter-arrival times (t_k) and utilization of all resources (U).
- Inferring service times $(X_k)_{1 \leq k \leq n}$

$$\begin{cases} R_k = [R_{k-1} - t_k]^+ + X_k & \text{1 server} \\ R_k = [R_j - t_{j,k}]^+ + X_k & \text{Several servers} \end{cases}$$

- Identify the shape of the service time sample with Kolmogorov-Smirnov tests.

- Validation : Compare empirical measures with theoretical ones
  - mean response time,
  - mean waiting time bound

Number of servers (K)

- $Cmax = Cmax_0$, $K=1$
- Iterate until saturation
  - When reaching the maximal load Cmax, check the utilization of all black box resources.

    If , for all resource, $U < 1$

    $K=K+1;$

    $Cmax = Cmax_0 * K$

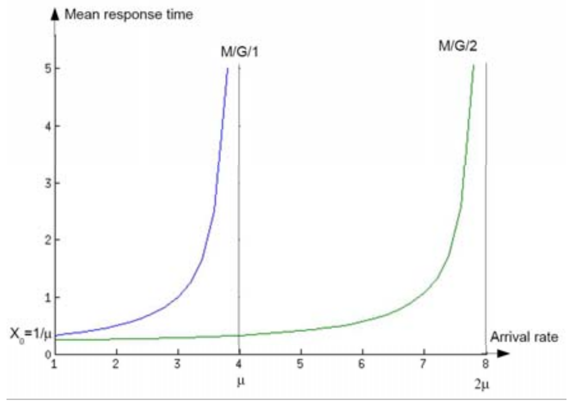
  - If, for a resource, $U \approx 1$

  Stop injection experiment,

  K = last identified value

Achieving self-regulated injection

1. Injection policy



- Initial maximal load Cmax = 1/ $\overline{R}$

  (R: response time measures sample)

- Injection step, Rising period
- Sampling period: Number of measures *n* >= $\left(\frac{100 z \sigma_n}{r\bar{m}}\right)^2$

2. Estimation of the stabilization time

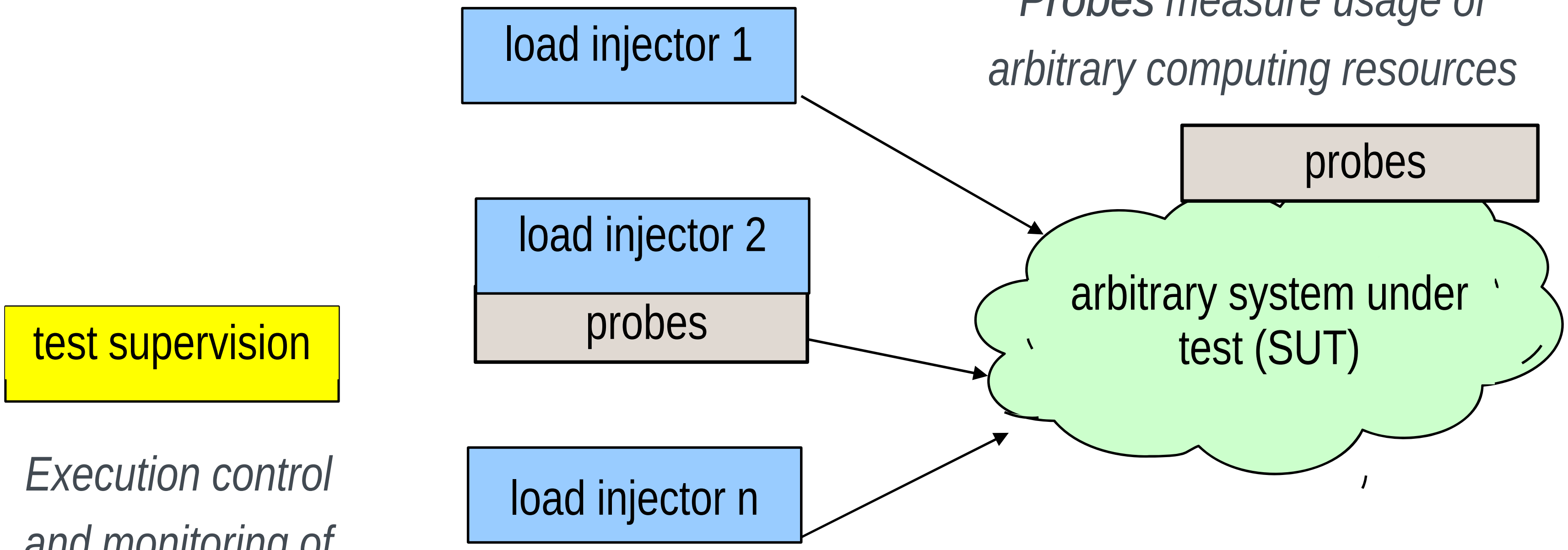
Stabilization time = convergence time of the queue Markov chain

(Restriction to Engset models)

Outline

1. Towards autonomic management
2. Modeling black boxes
3. Automatic black box model identification
4. Experimental results
5. Conclusion & future work

CLIF Load Injection Framework



Probes measure usage of arbitrary computing resources

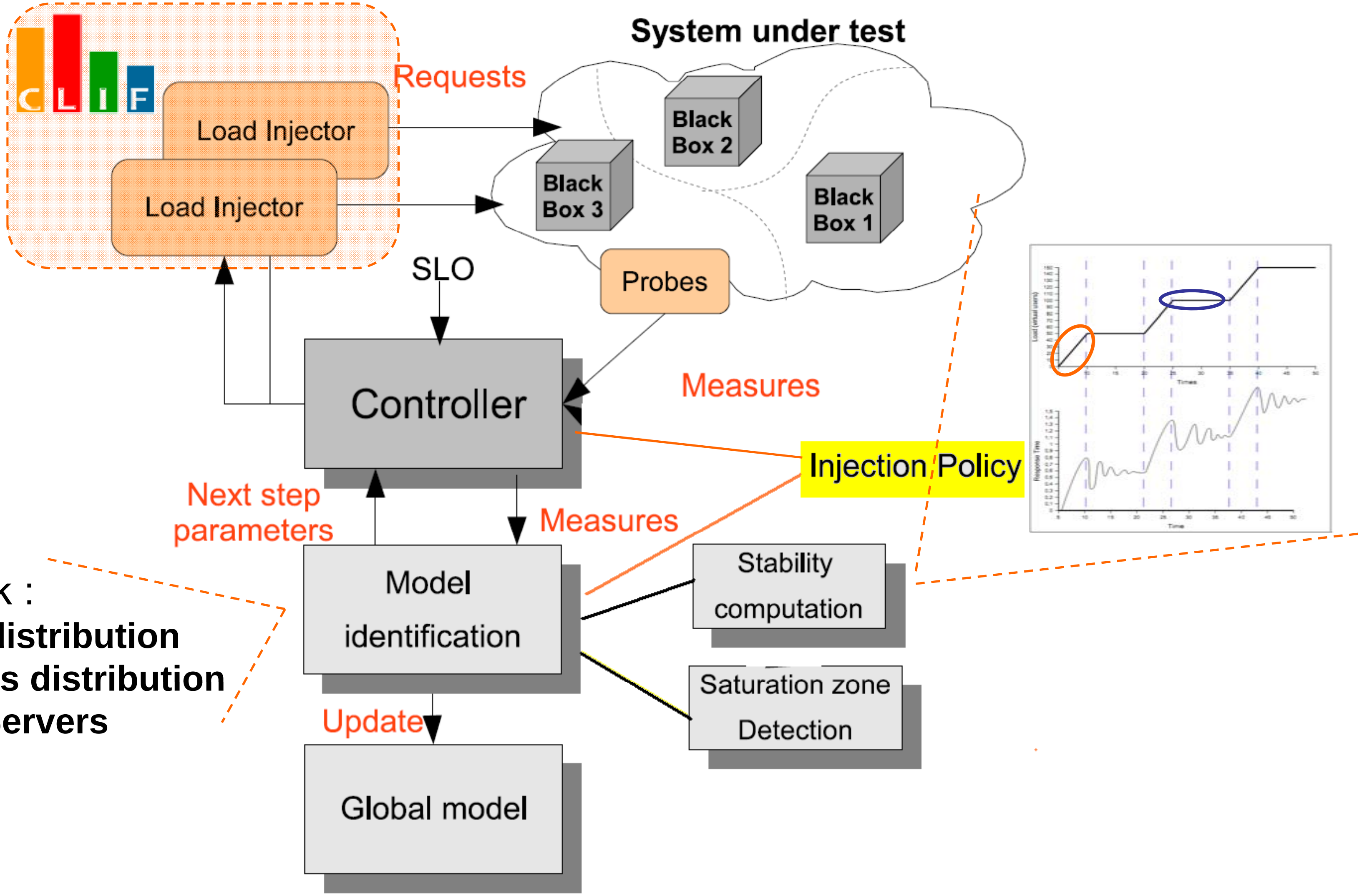
Execution control and monitoring of load injectors and resource probes.

Load injectors :

- *send requests, wait for replies, measure response times*
- *according to a given scenario*
- *for example, emulating the load of a number of real users (through so-called virtual users)*

Architecture of FAMI

(Framework for Automatic Modelling Identification)



Model T/X/k :
- **Interarrival distribution**
- **Service times distribution**
- **Number of Servers**

Outline

1. Towards autonomic management
2. Modeling black boxes
3. Automatic black box model identification
4. Experimental results
5. Conclusion & future work

Experimental results

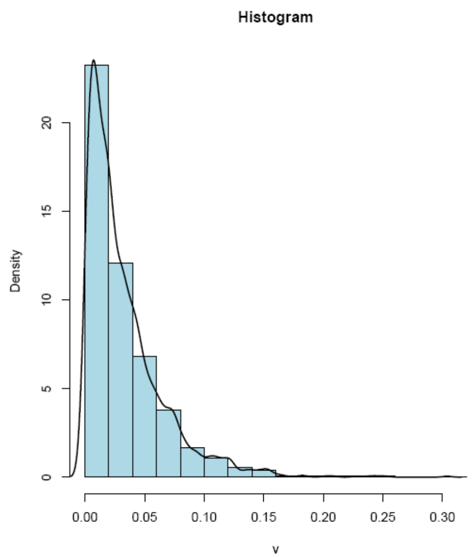
- Test-bed: Rubis Web-based application, workstation with 2 PIII 1.4Ghz, 1GO RAM.
- Injector Machine: workstation with quadri-processor Xeon 2Ghz, 2GO RAM.

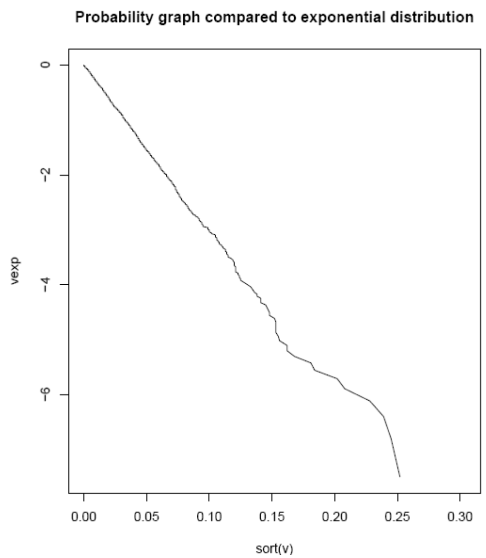
- 1st injection step
  - Average service time: $X_0 = 0.021$ s
  - Theoretical stabilization time = 0.043s
  - $Cmax_0 = 45.30$ Virtual users (requests)/s

- After 27 minutes of the experiment (12 steps), we reached
  - Number of virtual users = 120 vusers
  - Saturated resource : CPU → CPU load = 96%

Experimental results

Inter-arrival times distribution





CV^2=1.015 with a 95% confidence interval

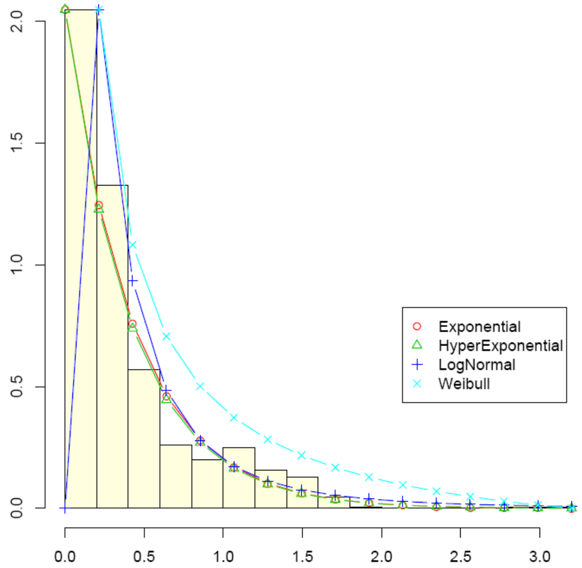
cv^2 → a possible fitness to an exponential distribution

λ =30.34 req/s

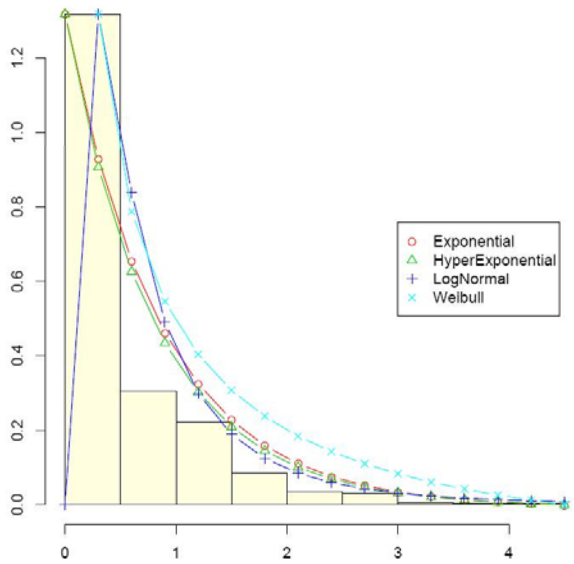
Kolmogorov-Smirnov test : p-value=0.59

Experimental results

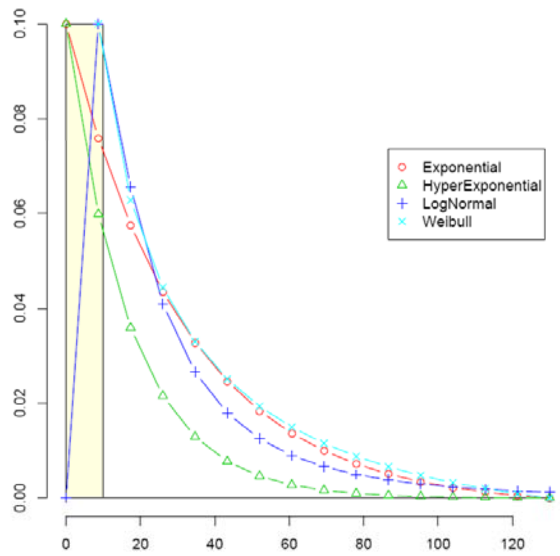
Service times distribution



80 Vusers



100 Vusers



120 Vusers

Identified distributions: Exponential (M), Lognormal (LN), Hyperexponential (Hr), Gamma (Γ), Weibull (Weib)

Experimental results

Identified Queue models

Load	Identified Model(s)	Parameters	Mean	CV^2
10	-	-	0.02	0.03
20	**M/LN/1**, $M/\Gamma/1$	$\lambda = 20,\ \mu = -3.80,\ \sigma = 0.20$	0.02	0.04
30	**M/LN/1**, $M/\Gamma/1$	$\lambda = 30,\ \mu = -3.71,\ \sigma = 0.30$	0.03	0.11
40	**M/LN/1**	$\lambda = 40,\ \mu = -3.52,\ \sigma = 0.51$	0.04	0.90
50	**M/LN/2**, $M/\Gamma/2$	$\lambda = 50,\ \mu = -3.44,\ \sigma = 0.51$	0.04	0.74
60	**M/LN/2**	$\lambda = 60,\ \mu = -3.00,\ \sigma = 0.89$	0.08	2.18
70	**M/M/2**, M/Hr(2)/2, M/LN/2, M/Weib/2	$\lambda = 70,\ \mu = 5.26$	0.19	1.96
80	**M/M/2**, M/Hr(2)/2, M/LN/2, M/Weib/2	$\lambda = 80,\ \mu = 2.48$	0.40	1.03
90	**M/M/2**, M/Hr(2)/2, M/LN/2, M/Weib/2	$\lambda = 90,\ \mu = 1.75$	0.57	1.46
100	**M/M/3**, M/Hr(2)/3, M/LN/3, M/Weib/3	$\lambda = 100,\ \mu = 1.73$	0.58	1.07
110	**M/M/3**, M/Hr(2)/3, M/LN/3, M/Weib/3	$\lambda = 110,\ \mu = 0.96$	1.04	23.90
120	M/M/3, **M/Hr(2)/3**, M/LN/3, M/Weib/3	$\lambda = 120,\ \mu_1 = 0.1321,\ \mu_1 = 2.5624$	9.72	0.74

Light load : M/LN/K model

Heavy load : M/M/K model

Conclusion

- Automatic performance modelling process of black boxes, based on a self regulated load injection testing and a theoretical approach.

- Benefit: performance prediction of an assembly of black boxes, useful when applying autonomic features.

- Development of a framework, FAMI, based on CLIF, which delivers a set of queue models for a range of workload.

- Difficulties:
  - Isolating a black box.
  - Configuration options necessary to test a black box (maximum connections, concurrent threads, ...).

Future work

- Improve the computation time of the stabilization time.
- Integrate, in the FAMI framework, a performance analysis/simulation tool.
- Instantiate the obtained FAMI framework to achieve self-sizing feature.

Thank you for your attention