Performance-Aware Reconfiguration of Software Systems (PARSY)

Moreno Marzolla[1] Raffaela Mirandola[2]

[1] Università di Bologna, Dip. di Scienze dell'Informazione
marzolla@cs.unibo.it
http://www.moreno.marzolla.name/

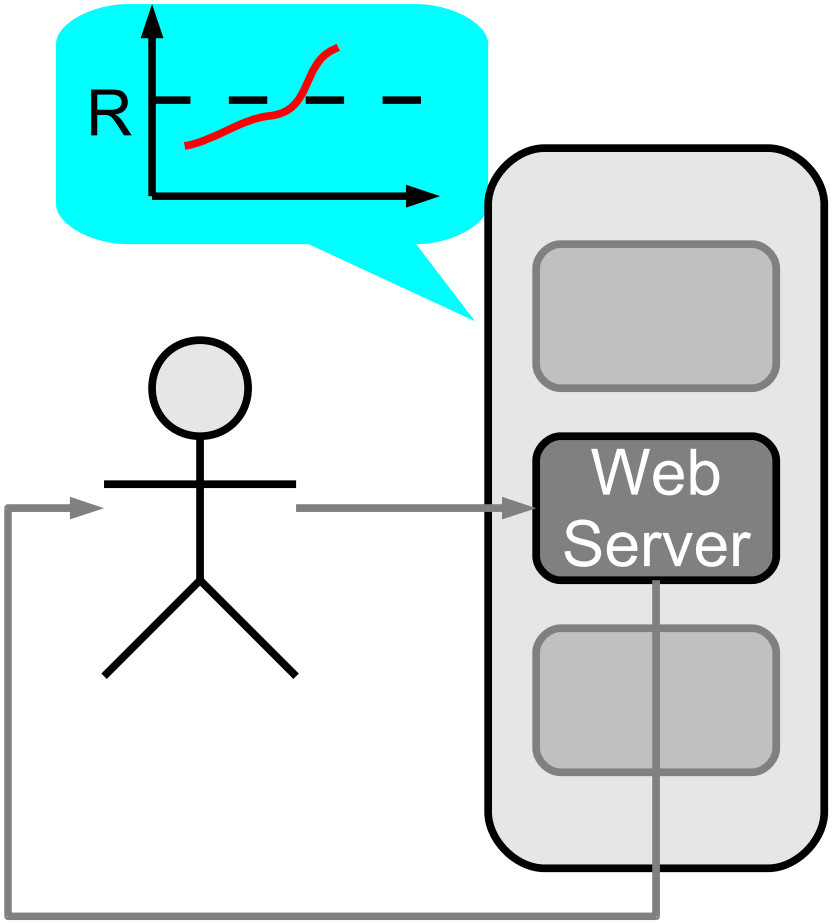
[2] Politecnico di Milano, Dip. di Elettronica e Informazione
mirandola@elet.polimi.it

Problem formulation

- We consider a component-based system which must be enhanced in order to provide an appropriate QoS level
  - Specifically, the system response time R should not exceed a pre-defined threshold R_{max}: $R < R_{max}$
- ...under variable workload
  - The number N of users accessing the system varies over time

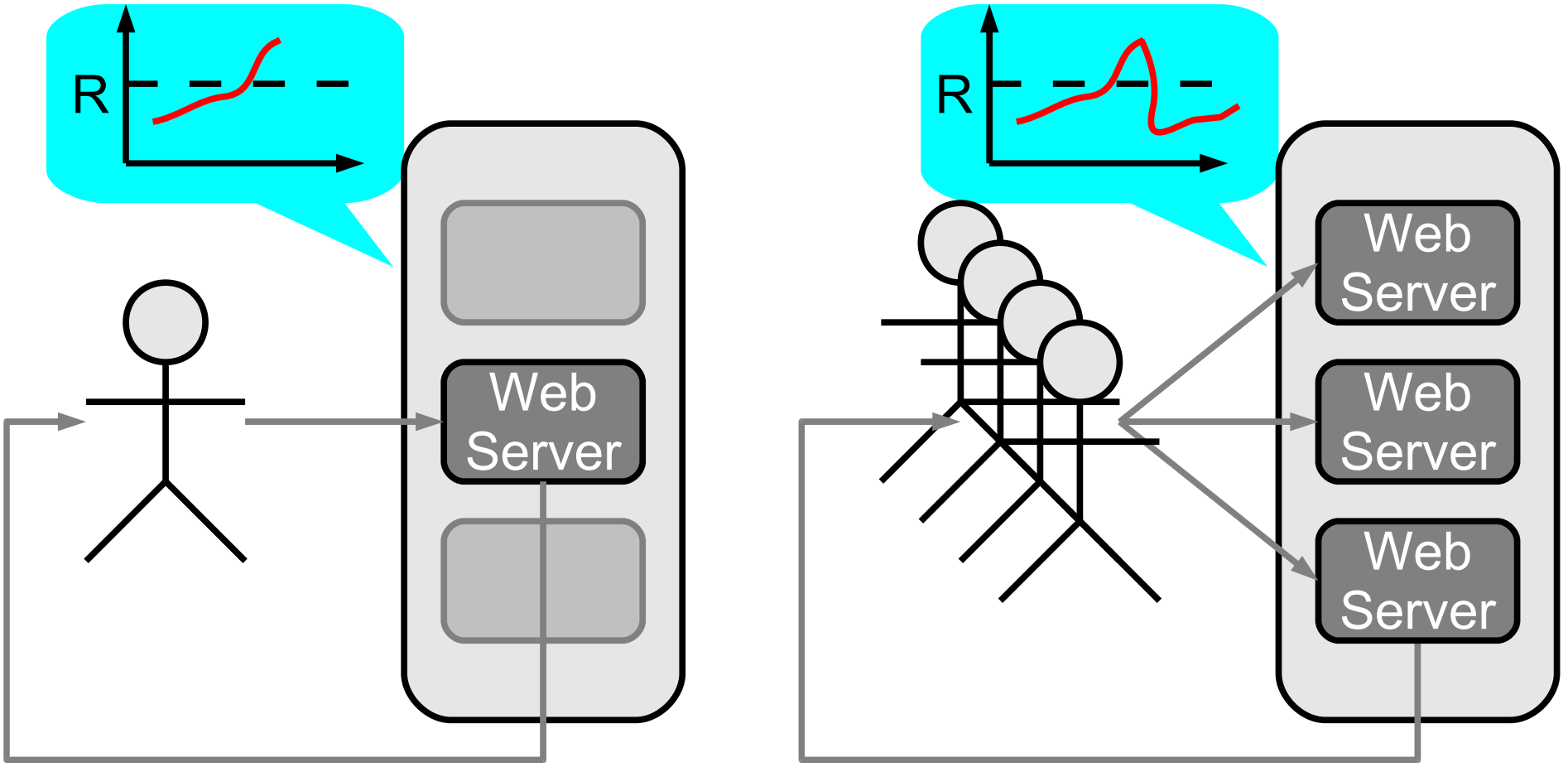
'a-la-Cloud solution

- Instantiate new components/services when necessary, to meet the increased workload



'a-la-Cloud solution

- Instantiate new components/services when necessary, to meet the increased workload



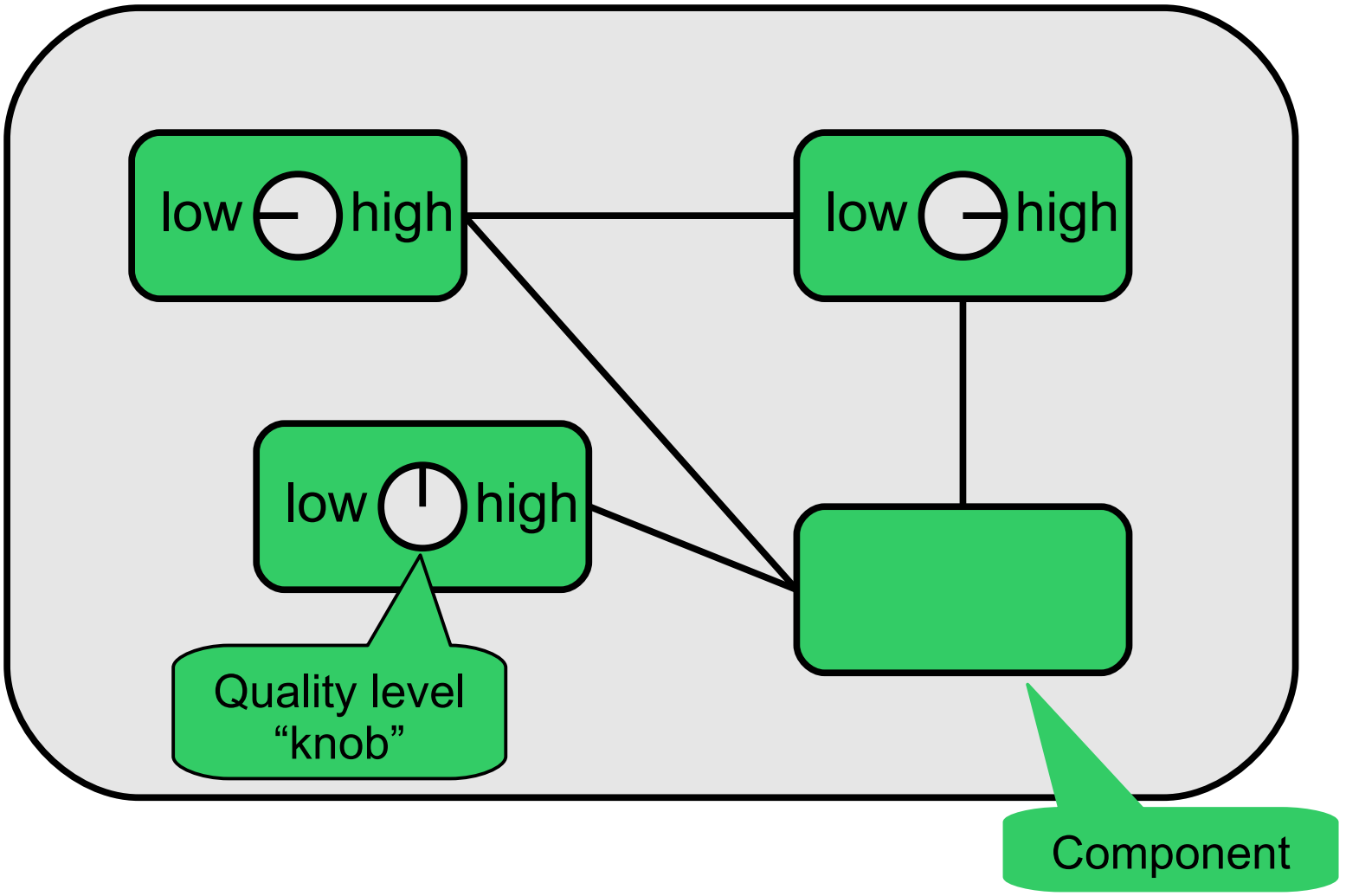
'a-la-Cloud solution—issues

- There can be situations where additional resources are not available, and thus the previous approach can not be applied
- Furthermore, bringing up a new service instance can incur significant delays
  - Virtual machine allocation;
  - Virtual machine configuration;
  - Service configuration;
  - Service startup;
  - ...

PARSY assumptions

- Component-based system
- Some of the components can be tuned to provide a "degraded" service with improved response time
  - Example: serve a highly compressed product image from an overloaded E-commerce Web site
  - Example: execute a faster (but less precise) query for products of interest on the abovementioned overloaded E-commerce site
- Note: of course, only non-critical components can in general be degraded

PARSY
System model



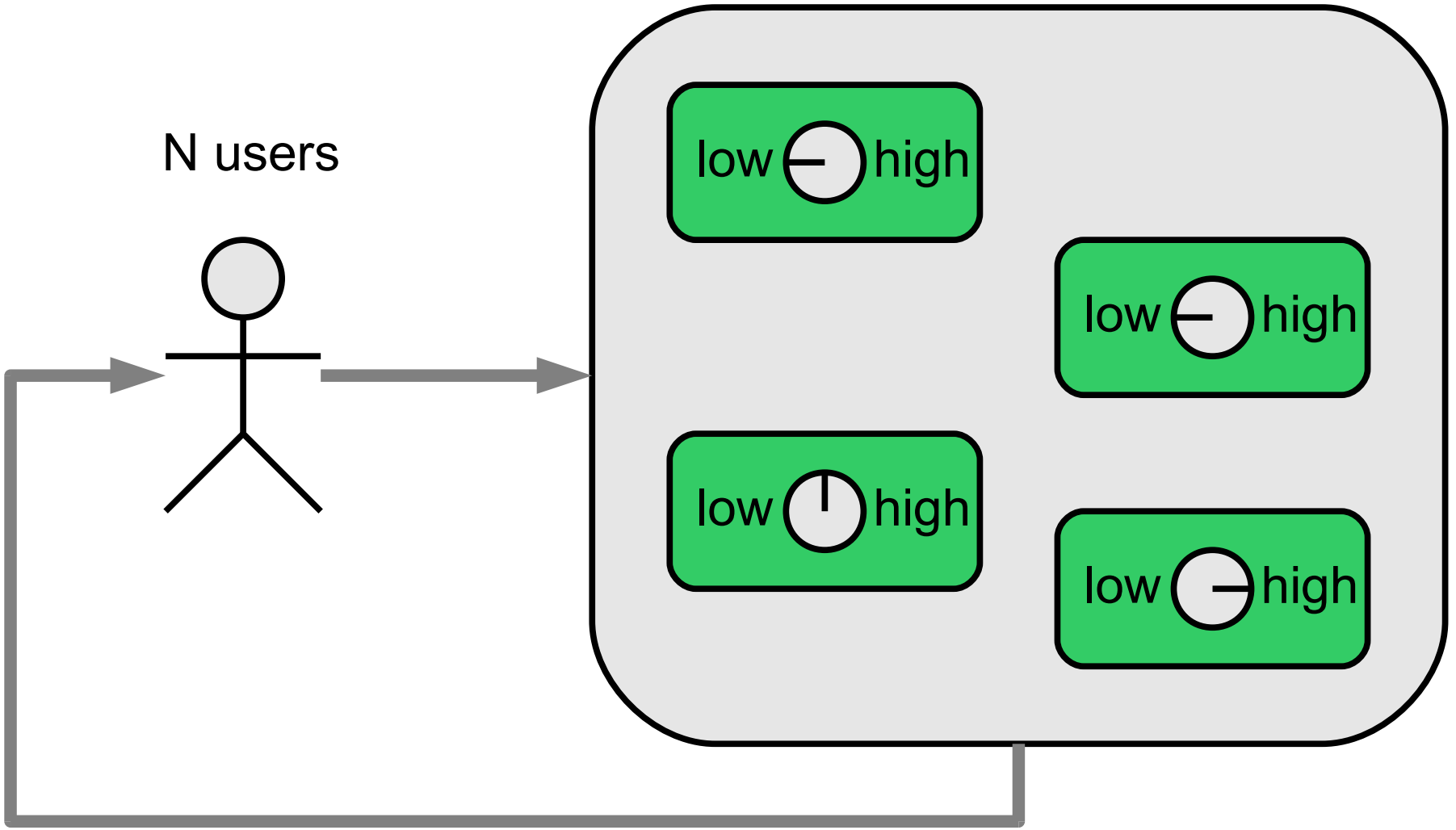
PARSY

- The system is assumed to be made of K components C_1, ... , C_K.
- Component C_k can be configured to provide service at different quality levels 1, ... , L_k (1=worst, L_k=best)
- Component C_k operating at level j requires average service demand D(k,j)
  - $D(k,1) < D(k,2) < ... < D(k,L_k)$
- Component C_k operating at quality level j is labeled with a positive utility value UT(k,j)
  - $UT(k,1) < UT(k,2) < ... < UT(k,L_k)$

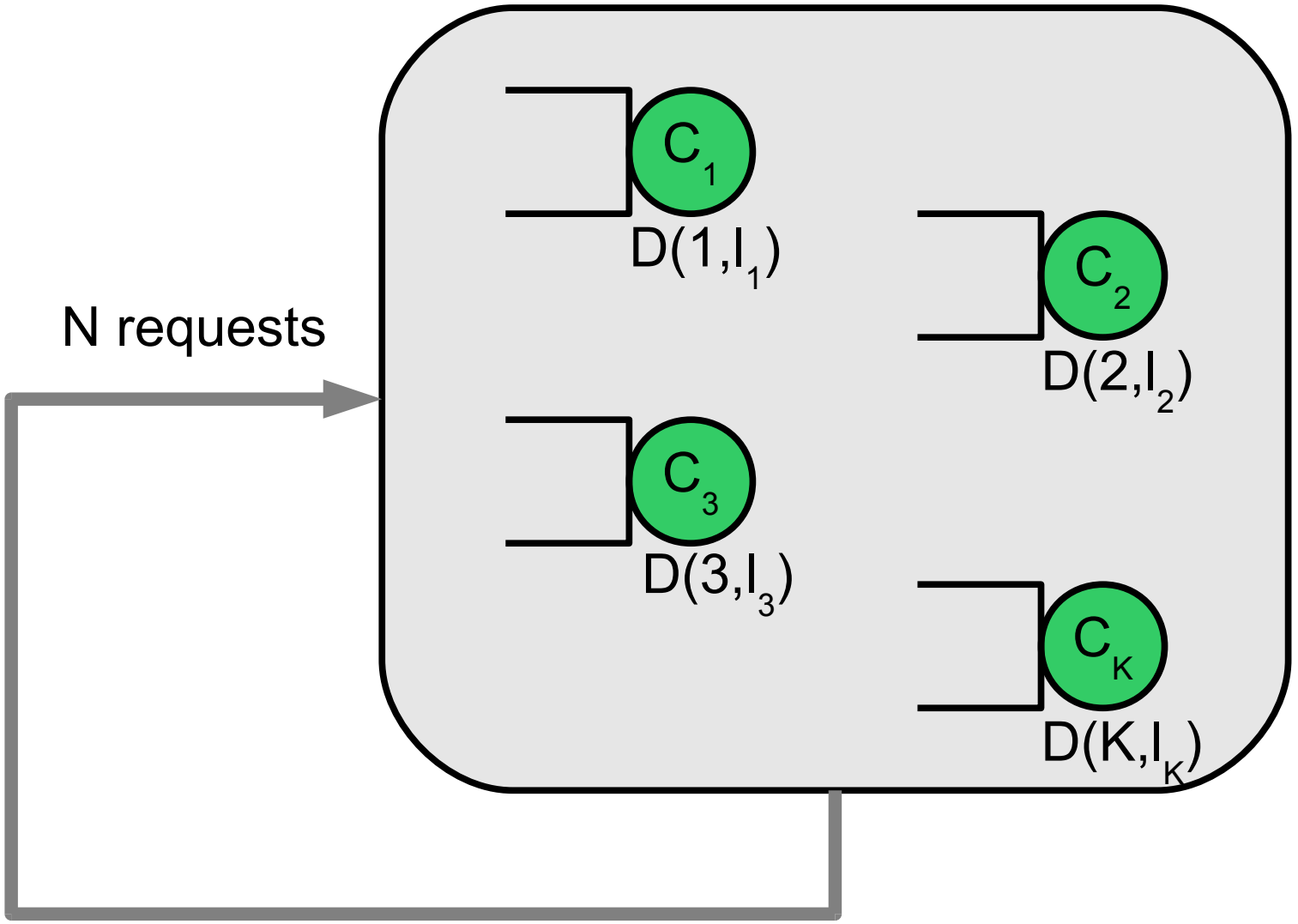
PARSY

- Goal:
  - Maximize: the total system utility
  - Subject to: *estimated response time R* < R_{max}
- We use a closed QN model to estimate the system response time wrt different configurations

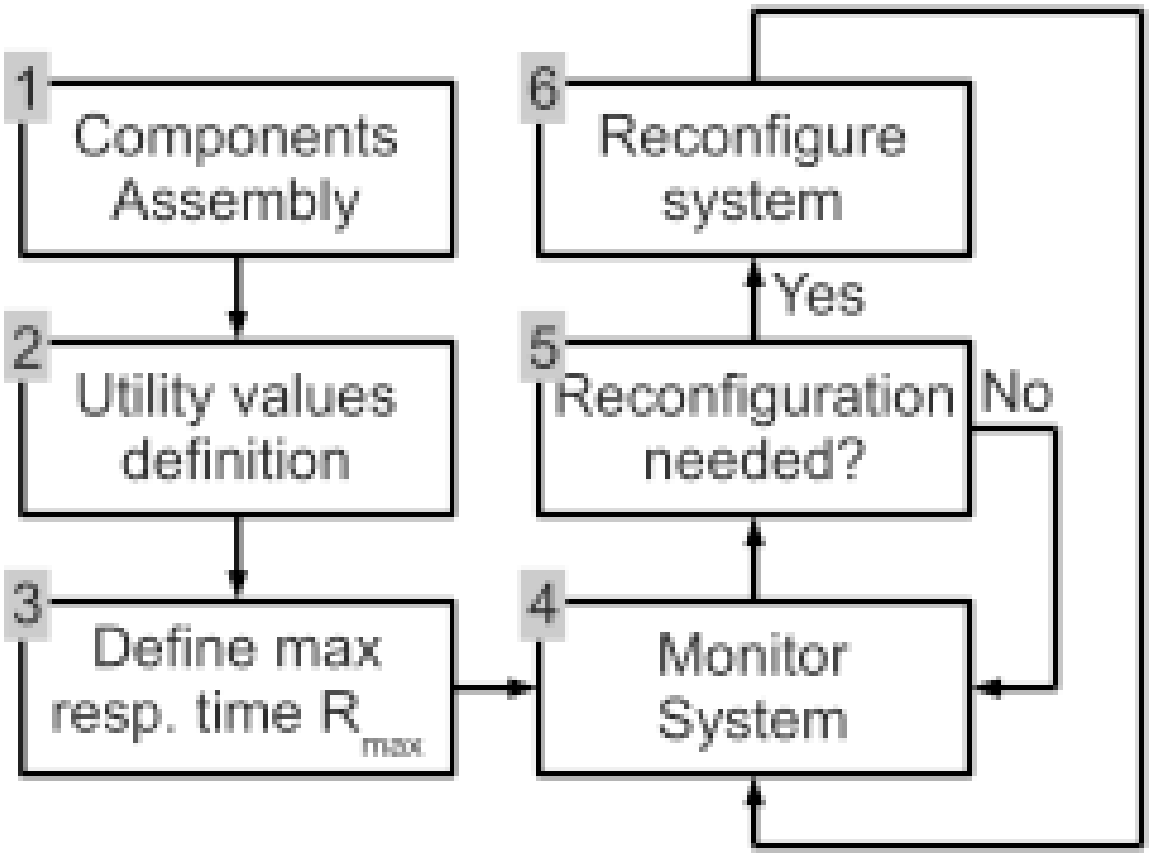
PARSY: system model



PARSY: Performance model



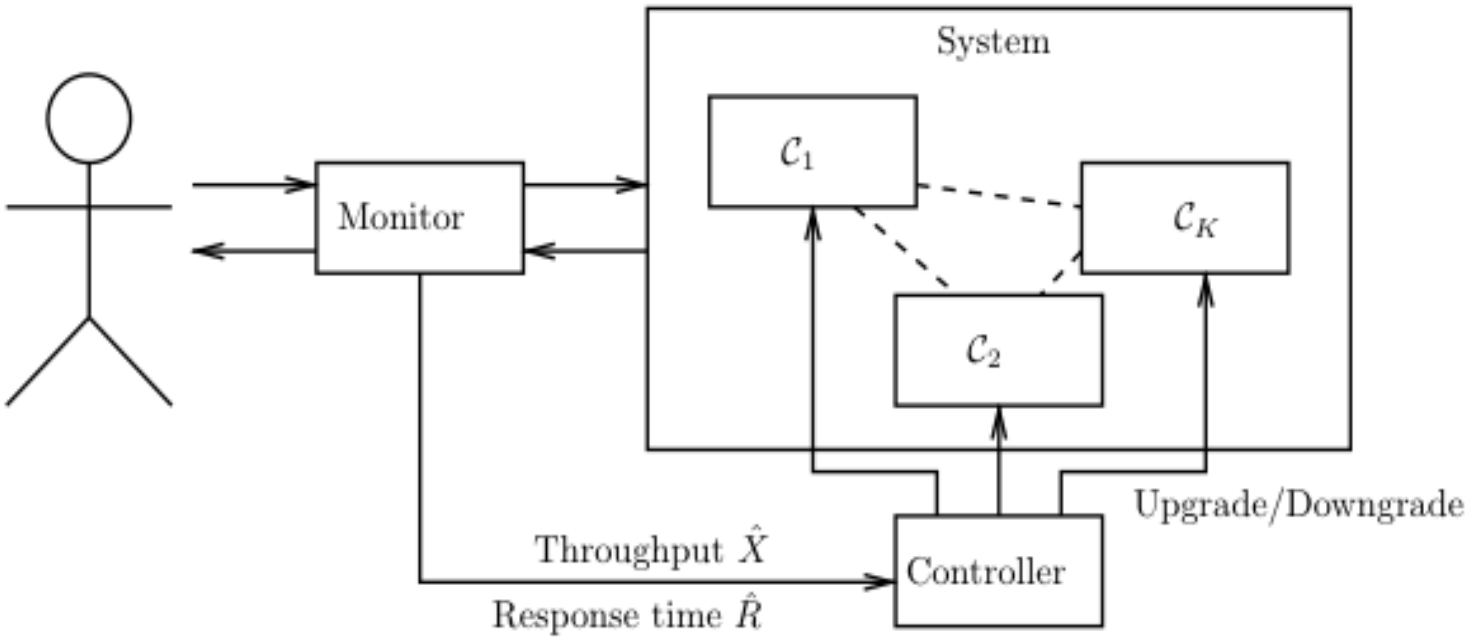
PARSY steps


1
Components Assembly
2
Utility values definition
3
Define max resp. time R_{max}
4
Monitor System
5
Reconfiguration needed?
Yes
No
6
Reconfigure system

PARSY: System monitoring

- The system is enhanced with a monitoring component, which continuously measures the average system response time.
- If the response time is above or below the threshold, a system reconfiguration can be triggered.
- It is important to ensure that reconfigurations are not too frequent, so that the system has enough time to settle down.

PARSY: System monitoring



- From Little's Law we can estimate N as

$$N = \hat{X}\,\hat{R}$$

System reconfiguration

- If $\hat{R} > R_{max}$, we identify components which can be degraded.
  - We keep degrading components until the estimated system response time, as computed using the QN model, becomes less than the threshold R_{max}.
- If $\hat{R} < R_{max}$, we identify components which can be upgraded.
  - We keep upgrading components as long as the estimated system response time remains less than the threshold Rmax.

Degrade system

Algorithm 1 Degrade configuration

Require: ℓ current system configuration
Require: N number of requests in the system
Require: $D(k, j)$ service demand of component $\mathcal{C}_k$ operating at quality level j
Require: $\mathrm{UT}(k, j)$ utility of component $\mathcal{C}_k$ operating at quality level j
Ensure: ℓ^{new} is the new system configuration

$\ell^{\mathrm{new}} \leftarrow \ell$
$C \leftarrow \{k \mid \ell_k^{\mathrm{new}} > 1\}$ {Candidate set of components which can be degraded}
while $C \neq \emptyset$ **do**
  $B \leftarrow \arg\max_k \{D(k, \ell_k^{\mathrm{new}})/\mathrm{UT}(k, \ell_k^{\mathrm{new}}) \mid k \in C\}$
  $\ell_B^{\mathrm{new}} \leftarrow \ell_B^{\mathrm{new}} - 1$ {Degrade $\mathcal{C}_B$}
  Compute $R(N; \ell^{\mathrm{new}})$ using Algorithm 3 (MVA)
  if $R(N; \ell^{\mathrm{new}}) < R_{\max}$ **then**
    Break {Downgrade complete}
  else
    $C \leftarrow \{k \mid \ell_k^{\mathrm{new}} > 1\}$ {Recompute the candidate set}
Return ℓ^{new}

Upgrade system

Algorithm 2 Upgrade configuration

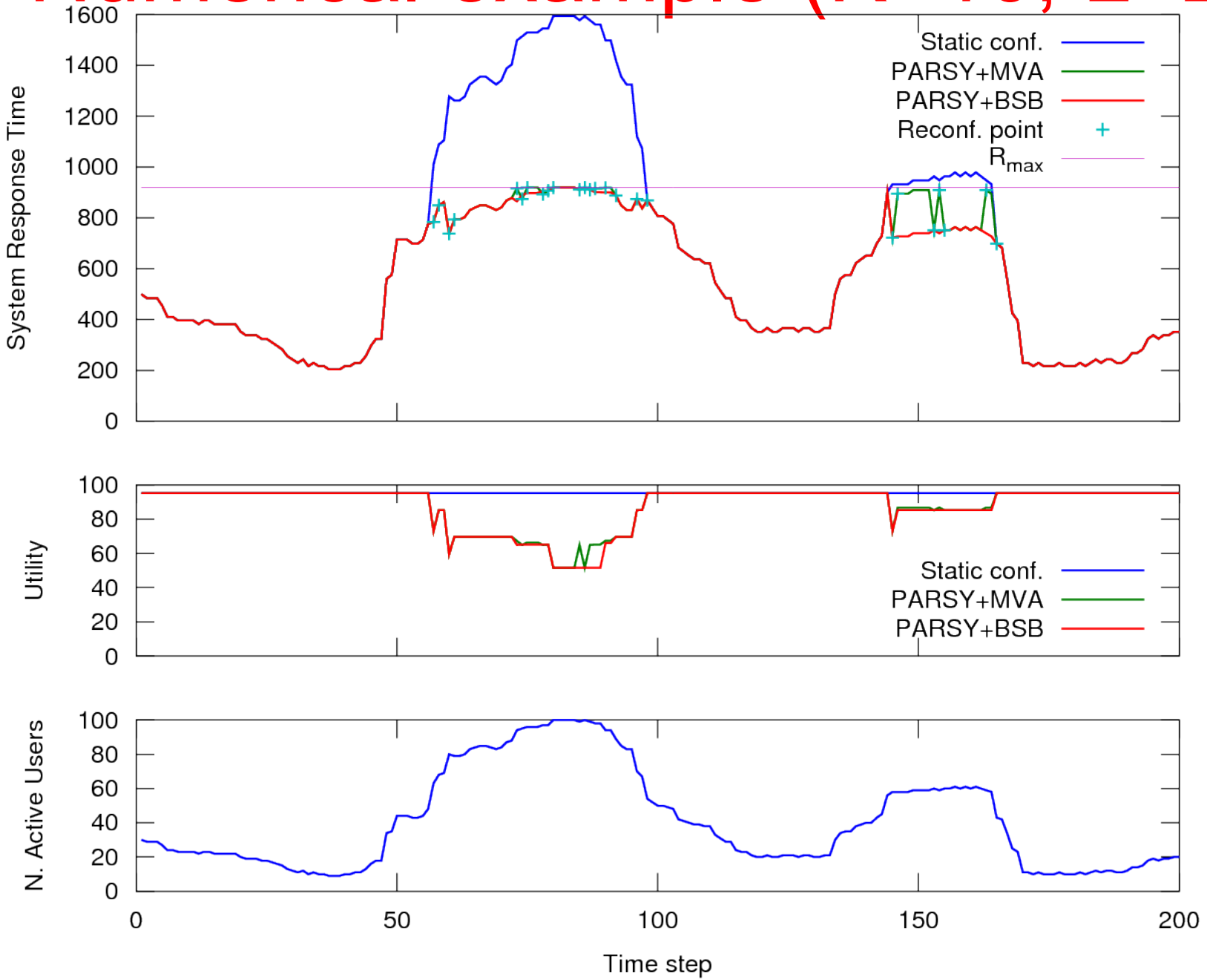
Require: $\boldsymbol{\ell}$ current system configuration
Require: N number of requests in the system
Require: $D(k, j)$ service demand of component $\mathcal{C}_k$ operating at quality level j
Require: $\mathrm{UT}(k, j)$ utility of component $\mathcal{C}_k$ operating at quality level j
Ensure: $\boldsymbol{\ell}^{\mathrm{new}}$ is the new system configuration

$\boldsymbol{\ell}^{\mathrm{new}} \leftarrow \boldsymbol{\ell}$
$C \leftarrow \{k \mid \ell_k^{\mathrm{new}} < L_k\}$ {Candidate set of components which can be upgraded}
while $C \neq \emptyset$ **do**
  $U \leftarrow \arg\min_k \{D(k, \ell_k^{\mathrm{new}} + 1)/\mathrm{UT}(k, \ell_k^{\mathrm{new}} + 1) \mid k \in C\}$
  $\ell_U^{\mathrm{new}} \leftarrow \ell_U^{\mathrm{new}} + 1$ {Try to upgrade $\mathcal{C}_U$}
  Compute $R(N; \boldsymbol{\ell}^{\mathrm{new}})$ using Algorithm 3 (MVA)
  if $R(N; \boldsymbol{\ell}^{\mathrm{new}}) > R_{\max}$ **then**
    $\ell_U^{\mathrm{new}} \leftarrow \ell_U^{\mathrm{new}} - 1$ {Rollback configuration for $\mathcal{C}_U$}
    $C \leftarrow C \setminus \{U\}$
  else
    $C \leftarrow \{k \mid \ell_k^{\mathrm{new}} < L_k\}$ {Recompute the candidate set}
Return $\boldsymbol{\ell}^{\mathrm{new}}$

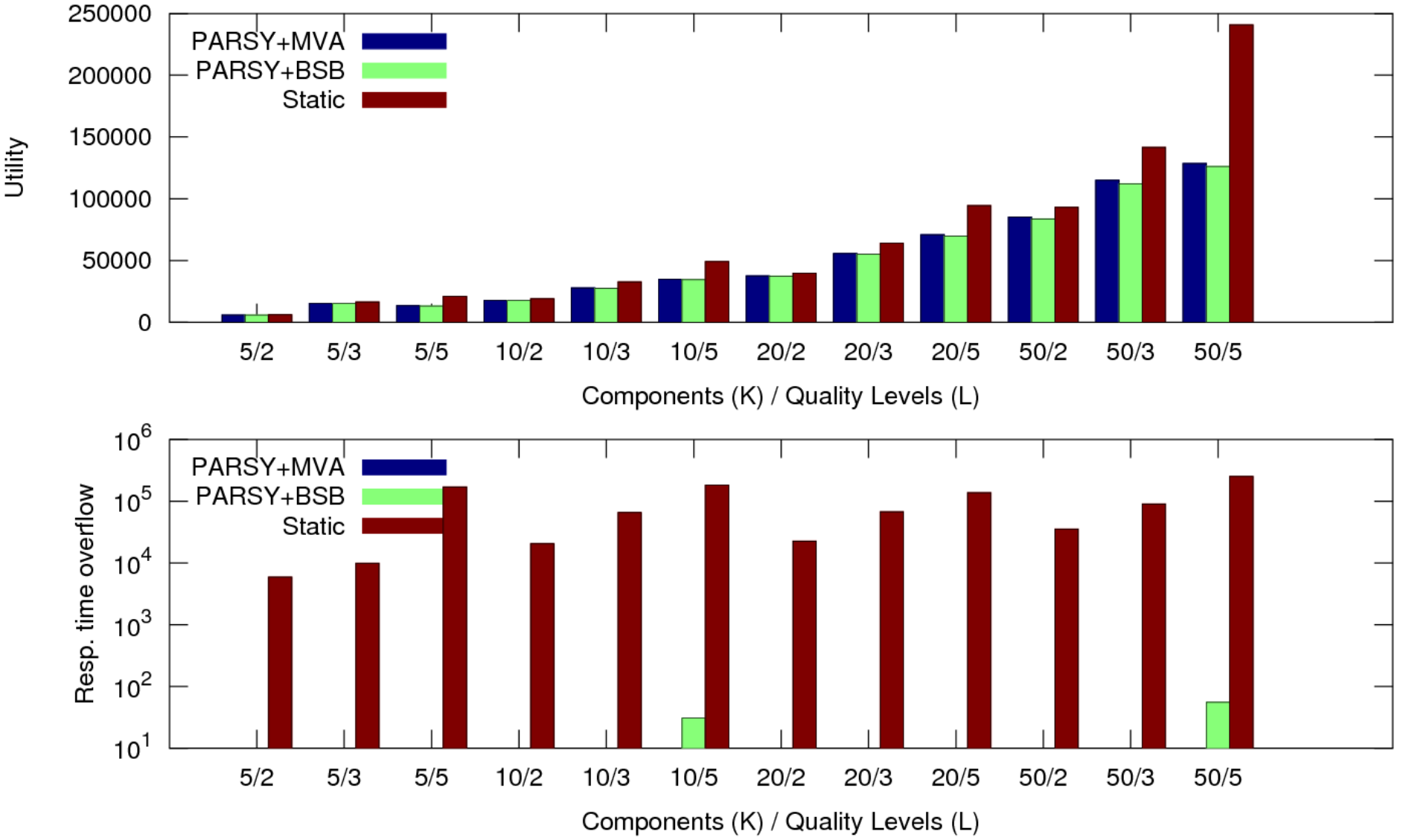
Numerical example

- We experiment with multiple combinations of K and L: we use K = 10, 20, 30, 50 and L = 2, 3, 5.
- Service demands D(k, j) and utilities UT(k, j) are randomly generated when each model is created.
- We evaluate each system for T = 200 time steps. The number of active users Nt at step t = 1,...,T is produced using a random walk model.

Numerical example (K=10, L=2)



Numerical example



Conclusions and future works

- PARSY seems effective in reducing the response time overflow; at the same time the total utility is kept a good fraction of the maximum possible value.
- In the considered test cases, PARSY+BSB produces only marginally worse reconfigurations than those produced by PARSY+MVA ←*That's a good news*
- TODO
  - What if D(k,j) are not known in advance?
  - What if D(k,j) vary over time?
  - What if there are multiple values for D(k,j) for the same k,j *at the same time*? ←*QN model with multiple job classes?*