



HAL
open science

Poster: A Procedure for Testing Applications Through Cloud Computing Elasticity

Michel Albonico, Jean-Marie Mottu, Gerson Sunyé

► **To cite this version:**

Michel Albonico, Jean-Marie Mottu, Gerson Sunyé. Poster: A Procedure for Testing Applications Through Cloud Computing Elasticity. JDoc Day - ED STIM, Apr 2016, Nantes, France. 2016. hal-01317543

HAL Id: hal-01317543

<https://hal.science/hal-01317543v1>

Submitted on 18 May 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Copyright



A Procedure for Testing Applications Through Cloud Computing Elasticity

Director

SUNYÉ, Gerson
gerson.sunye@inria.fr

PhD Student

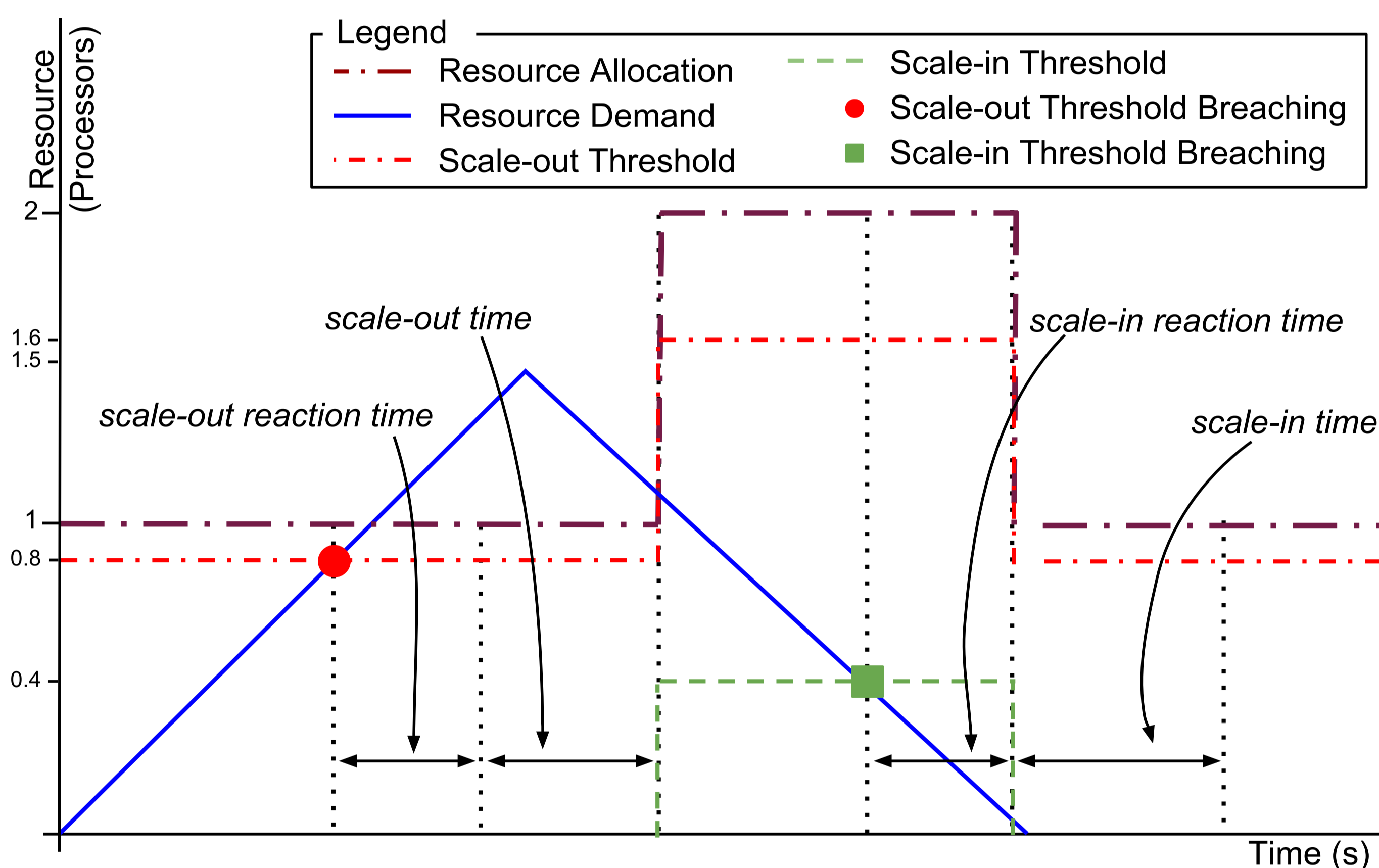
ALBONICO, Michel
michel.albonico@mines-nantes.fr

Co-Supervisor

MOTTU, Jean-Marie
jean-marie.mottu@inria.fr

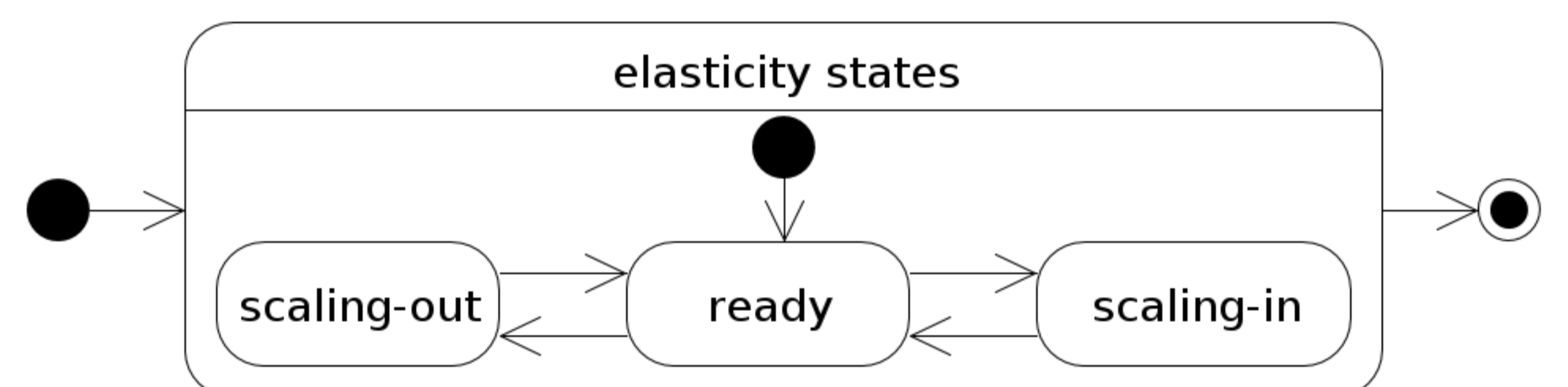
1 Elasticity

- It is the ability of a cloud infrastructure to vary its resources according to demand.
- The resource demand varies over time:
 - When it increases, it may breach the **scale-out threshold**, and remain higher for a while (**scale-out reaction time**). Then, a new resource is added, which takes some time (**scale-out time**);
 - When it decreases, it may breach the **scale-in threshold**, and remain lower for a while (**scale-in reaction time**). Then, the additional resource is removed (within the **scale-in time**).



2 Elasticity States

- A Web application starts by being exposed to the **ready state**: when the resource is steady;
- If a new resource starts being added, the application is exposed to the **scaling-out state**: period while the resource is being added;
- If a resource starts being released, the application is exposed to the **scaling-in state**: period while the resource is being released.

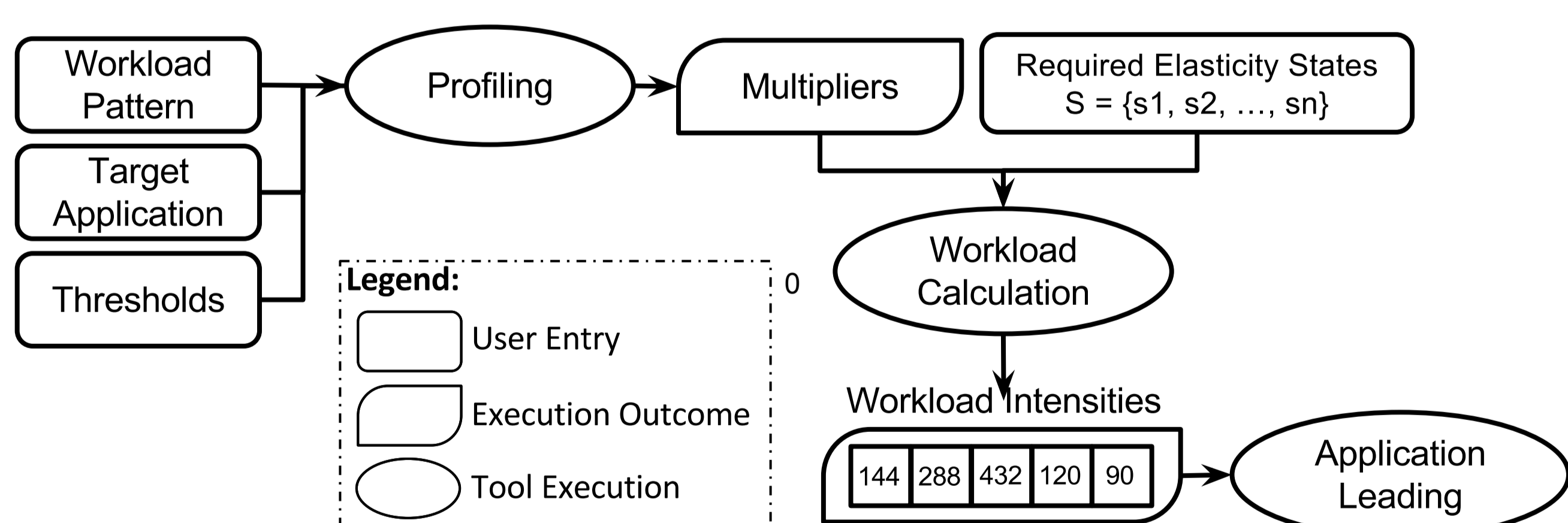


3 Research Problem

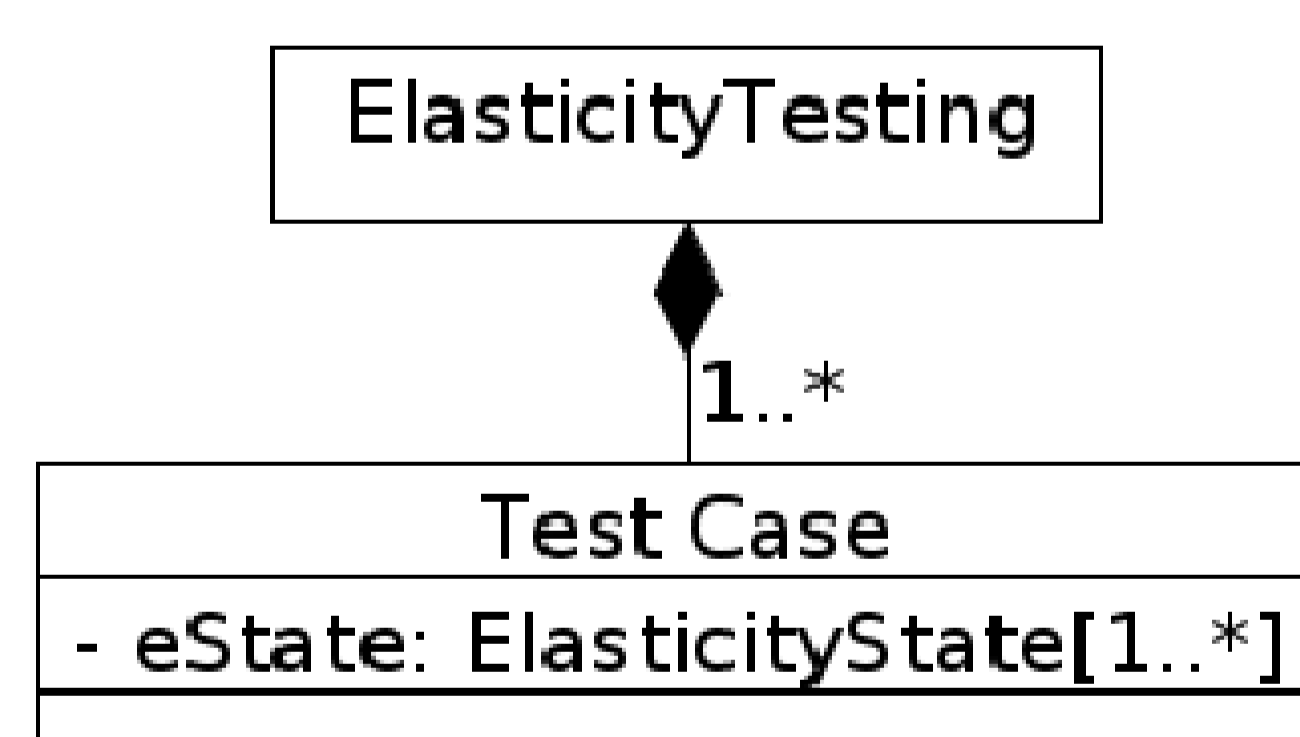
- Errors may happen at any elasticity state. Therefore, we should test applications at every elasticity state;
- Sometimes, we need a specific sequence of elasticity states, e.g., for regression testing, bug reproduction, etc.
- We propose an approach that controls the required sequence of elasticity states, and in parallel, tests the application dynamically, according to the current elasticity state.

4 Our Approach

a) Elasticity Control Workflow



b) Test Suite Metamodel



c) Test Algorithm

```

Data: Test Specification T, Execution Timeout eto
mon ← monitor();
i = 0;
tbegin ← current_time;
while tbegin - current_time < eto do
  foreach tc ∈ T.testCases do
    i = i + 1;
    csi ← mon.currentState();
    if tc.states.contains(csi) then
      Vcsi,tc ← run(tc);
    end
  end
end
end
    
```

5 Experiment Result

- We lead MongoDB through a sequence of elasticity states;
- We find non-functional errors at different elasticity states;
- We correctly assign the test verdicts to the elasticity states.

6 Conclusion and Future Work

- Our approach is able to control the required elasticity states, and test the application dynamically, in parallel.
- **Future work:**
 - Functional test cases, and generic cloud applications.