A DSL-based Approach for Elasticity Testing of Cloud Systems

Michel Albonico
UTFPr-*Science without borders*
Brazil

Amine Benellalam
INRIA-Mines Nantes
France

Jean-Marie Mottu, Gerson Sunyé
INRIA-University of Nantes
France

AtlanMod team
EMN — Lina — Inria
http://www.emn.fr/z-info/atlanmod/
atlanmod-contact@mines-nantes.fr

Outline

1. Elasticity Testing
2. Motivation
3. DSL-based Approach
4. Preliminary Results
5. Conclusion and Future Work

Elasticity Testing

Cloud computing elasticity:

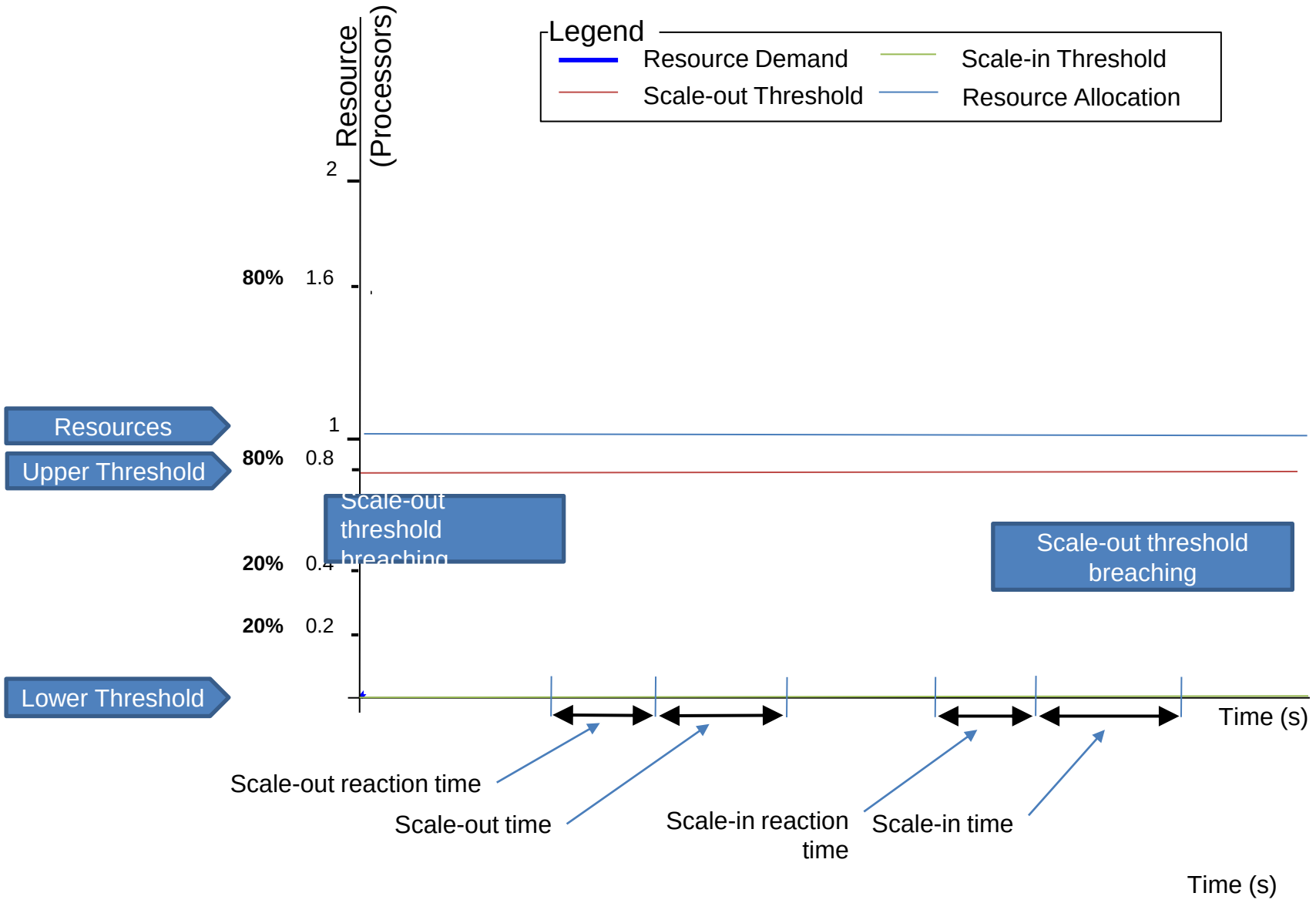
The ability of a cloud infrastructure/system to modify its resource configuration according to demand.

For instance: *When the CPU usage is higher than 60%, add a new resource (e.g., a Virtual Machine).*

Elasticity testing:

Testing cloud-based systems throughout elasticity.

Threshold-based Elasticity



Motivation

Tester's efforts:

- Set up cloud-based system;
  - Re-configure through elasticity.

Design and implement test methods, and coordinate their executions.

Motivation

Difficulties:

- Many setups: deployment of cloud-based system, elasticity, cloud system driving, and coordination of test execution;
- Each setup allows many parameters;
- Each cloud provider has its own user interface, and language.

DSL-based Approach

Previous work[1]:

- Set up cloud provider;
- Set up deployment of cloud-based system and its dependencies.

What we did not cover:

- Elasticity setup;
  - Elasticity driving;
  - Elasticity test execution.

```
software_bundles {
   software   httpd  : pkg 'apache2' '2';
   software   phpapp  : src './app/',
      dest '/var/www/app/';
   source   apachecfg  : src 'httpd.conf',
      dest '/etc/apache/httpd.conf';
   bundle   wsrv  :
      app phpapp, dep (httpd, php, mysq),
      src (apachecfg, createdb, addserver),
      provScript (createdb, addserver);
```

```
resources EC2 {
  image   iU704i386:
    imageId 'ami-1234',
    os 'Ubuntu' '7.04' 'i386';
  zone   EUWest :
     'eu-west-1a', 'eu-west-1b';
  instance   websrv :
    image iU704i386, machineType m3.large, zone EUWest,
    portConfig '80' = '0.0.0.0', bundle wsrv;}
```

[1] A. Thiery, T. Cerqueus, C. Thorpe, G. Sunye, and J. Murphy, “A DSL for Deployment and Testing in the Cloud,” in *2014 IEEE Seventh International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, 2014, pp. 376–382.

DSL-based Approach

Our approach:

Based on previous work:

Threshold-based elasticity setup;

Cloud-based system driving through elasticity[2];

Elasticity state based test execution[3].

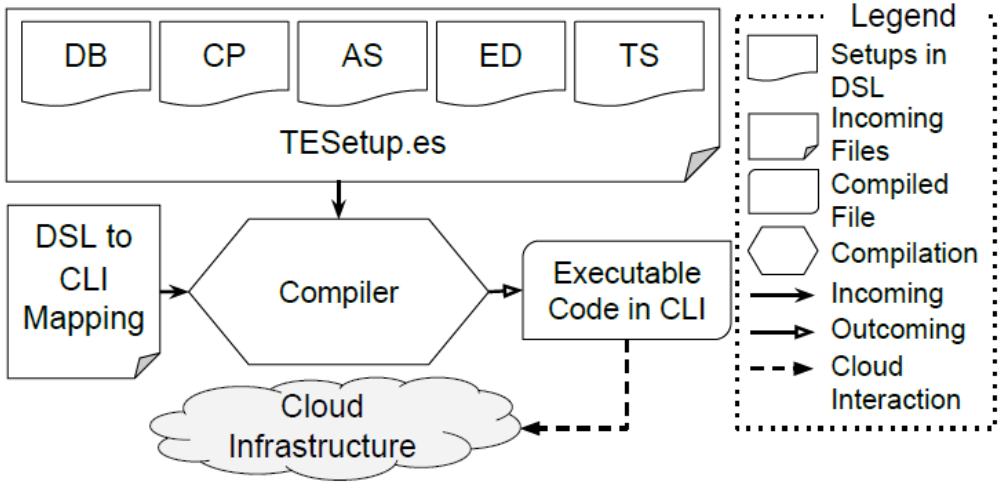
```
elasticity {
   resourcePool   wsrv_pool  : minSize 1,
       maxSize 10, cloudResource websrv;
   policy   wsrv_policy  : resourcePool slaves,
       coolDown 60000, reactionTime 60000,
       scalingAdjustment 1, adjustmentType Add;
   alarm   highCPU  : resourceType CPU,
       statistics Maximum, threshold 60,
       comparatorOperator '>', policy wsrv_policy;}
```

```
driving {
   drive   wsrv_drive  :
      pool wsrv_pool,  workType Read,
      states set(scaling-out, ready,
           scaling-in, scaling-out, ready);}
```

```
tests {
  test   t1 :  script 'java -jar ./test.jar test.test1';
  test   t2  :   script 'java -jar ./test.jar test.test2';
  suite   s1  : states scaling-out,
     driving wsrv_drive,
      test_method (t1,t2), in parallel; }
```

DSL-based Approach

Conceptual compilation workflow[1]:



DSL to CLI Mapping

```
provider AWS {
  instanceCommand 'aws ec2 run-instances' :
    '--image-id' imageId,   ...; }
```

Cloud Provider Setup

```
resources EC2 {
  image iU704i386 :
    imageId 'ami-1234',
    os 'Ubuntu' '7.04' 'i386';
  zone EUWest :
'eu-west-1a', 'eu-west-1b';
  instance websrv :
    image iU704i386, machineType m3.large, zone EUWest,
    portConfig '80' = '0.0.0.0', bundle wsrv;}
```

Executable Code in CLI

```
aws ec2 run-instance --image-id 'ami-1234' ...
```

[1] Currently, the steps are run manually

Preliminary Results

Cloud-based system case studies:

CS1: MongoDB deployed as a replica set.

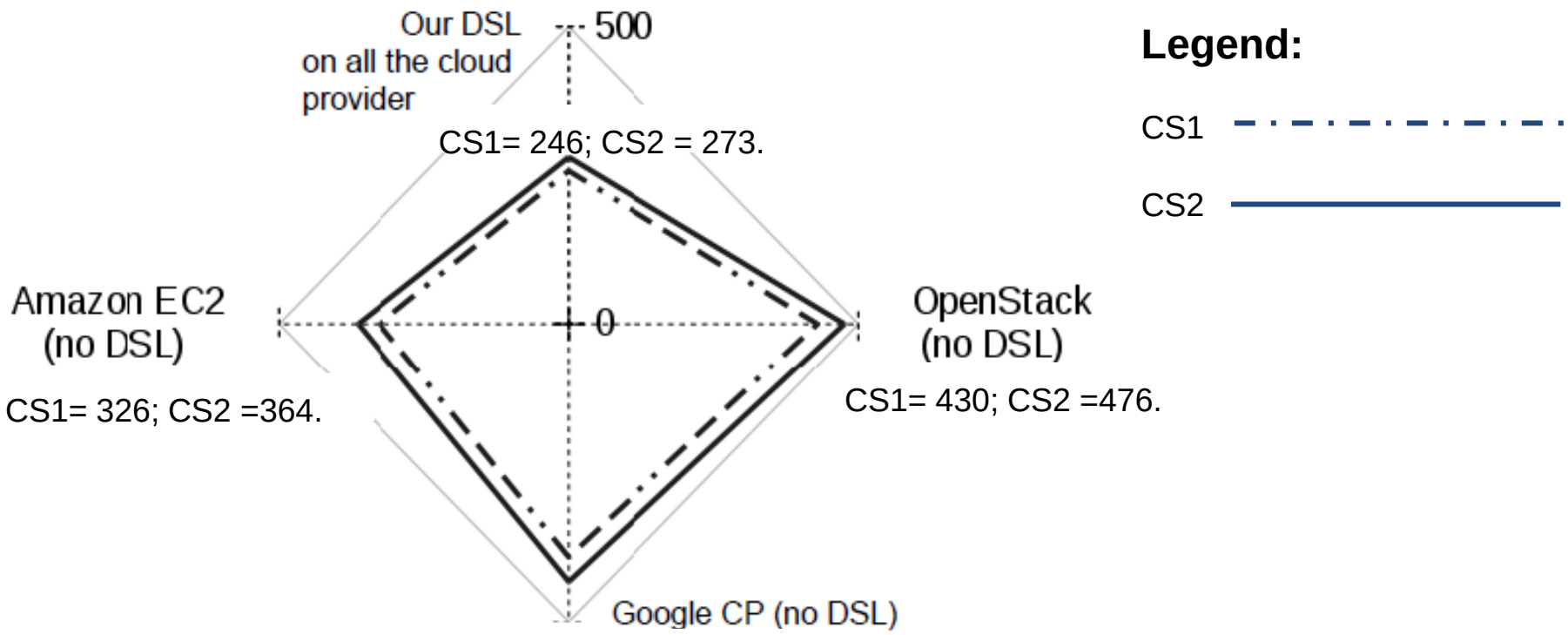
CS2: Distributed Web application composed by a centralized database server, a load balancer, and *n* Web servers.

Both are driven through specific sequences of elasticity states.

Preliminary Results

Effort on writing (total amount of words):

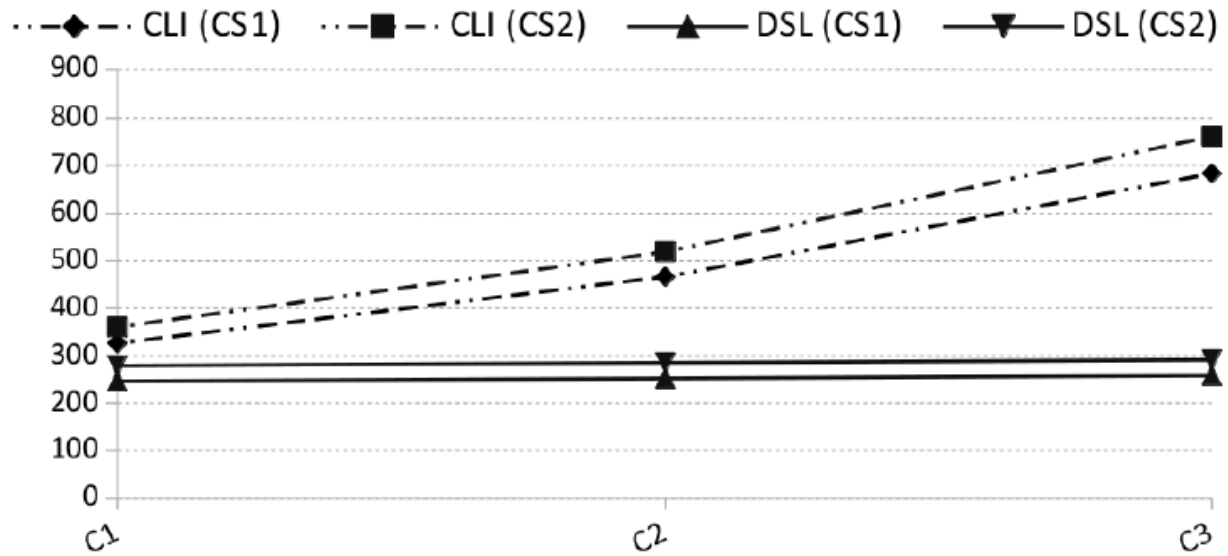
- Less effort using our DSL;
- From CS1 to CS2, the effort varies proportionally.



Preliminary Results

Increment in amount of words:

- New words to re-write an existing setup;
- Only a few words when using our DSL;
- Great increment (*2.1 times) when using native CLIs.



Conclusion and Future Work

Conclusion:

- Our DSL reduces the effort on writing elasticity testing setup.
- Multiple cloud providers with little code increment.

Future work:

- Language improvement;
- Automatic resource discovering;
- More exhaustive validation.

Thank you for your attention!