FITCH: Supporting Adaptive Replicated Services in the Cloud

Vinicius Cogo
André Nogueira
João Sousa
Alysson Bessani
Marcelo Pasin
Hans P. Reiser

1 University of Lisbon, Portugal
2 University of Neuchatel, Switzerland
3 University of Passau, Germany
What is an adaptation?
What is an adaptation?

I need a new CPU and more RAM
What is an adaptation?

upgrade
We adapted the computer to our needs
What is a service adaptation?
What is a service adaptation?

3 machines providing a service
What is a service adaptation?

The service becomes overloaded
What is a service adaptation?

Add a new replica (without shutting down the service)
What is a service adaptation?

The service adapted to clients demand
Motivation

Dynamism of cloud computing

Dynamic adaptation of replicated services
Motivation

Dynamism of cloud computing

- “Infinite” elasticity
  - Create new instances
  - Destroy instances
- Different VM types
- Monitoring tools
- Pay-as-you-go model

Dynamic adaptation of replicated services
Motivation

Dynamism of cloud computing

- “Infinite” elasticity
  - Create new instances
  - Destroy instances
- Different VM types
- Monitoring tools
- Pay-as-you-go model

Dynamic adaptation of replicated services

- Group membership
  - Insert new replicas
  - Remove replicas
- State transfer
- Adaptation heuristics
Motivation

Dynamism of cloud computing

Dynamic adaptation of replicated services

Our goal is to **support dynamic adaptation of replicated services in cloud environments**

**Contributions:**

1. FITCH (Fault-and-Intrusion Tolerant Computing Hardpan)
2. An experimental evaluation (2 use cases + 3 experiments)
## 2. Adaptations of replicated services

<table>
<thead>
<tr>
<th>Adaptation solution</th>
<th>Performance</th>
<th>Economic</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>Increasing the number of service instances</td>
<td>✓</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Reducing the number of service instances</td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Upgrading the resources of replicas</td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Downgrading the resources of replicas</td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Moving replicated services</td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Moving server instances</td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Moving replicas</td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Replacing faulty replicated services</td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Software replacement</td>
<td></td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Software update</td>
<td></td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Replacing old service instances</td>
<td></td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

Each adaptation has a reason and a solution
## 2. Adaptations of replicated services

<table>
<thead>
<tr>
<th>Adaptation solution</th>
<th>Performance</th>
<th>Economic</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>Increasing the number of service instances</td>
<td>×</td>
<td></td>
<td>×</td>
</tr>
<tr>
<td>Reducing the number of service instances</td>
<td></td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>Upgrading the resources of replicas</td>
<td>×</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Downgrading the resources of replicas</td>
<td></td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>Moving replicas to different cloud providers</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>Moving service instances close to clients</td>
<td>×</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Moving replicas away from attackers</td>
<td>×</td>
<td></td>
<td>×</td>
</tr>
<tr>
<td>Replacing faulty replicas</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>Software replacement</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>Software update</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>Replacing old service instances</td>
<td>×</td>
<td></td>
<td>×</td>
</tr>
</tbody>
</table>
## 2. Adaptations of replicated services

<table>
<thead>
<tr>
<th>Adaptation solution</th>
<th>Performance</th>
<th>Economic</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>Increasing the number of service instances</td>
<td>×</td>
<td></td>
<td>×</td>
</tr>
<tr>
<td>Reducing the number of service instances</td>
<td></td>
<td></td>
<td>×</td>
</tr>
<tr>
<td>Upgrading the resources of replicas</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Downgrading the resources of replicas</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Moving replicated services to another location</td>
<td>×</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Moving service instances to another location</td>
<td>×</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Moving replicated service instances to another location</td>
<td>×</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Replacing faulty service instances</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>Software replacement of faulty service instances</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>Software update of service instances</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>Replacing old service instances</td>
<td>×</td>
<td></td>
<td>×</td>
</tr>
</tbody>
</table>

But how can we perform all these adaptations in a cloud environment?
3. The FITCH architecture

Architectural overview
3. The FITCH architecture

Architectural overview
3. The FITCH architecture

Architectural overview

Adaptation Heuristics

Service Clients

Replicated Service

Host

Server

Server

Server

Cloud IaaS

Cloud Resource Manager
3. The FITCH architecture

System and threat models

- Hybrid distributed system model

2 sub-systems

- Control plane (trusted)
  - Synchronous system model (bounded comp. and comm.)
  - Can be subject only to accidental faults (fail-stop)

- Data plane (untrusted)
  - Partially synchronous system model
  - Can be subject to Byzantine faults (arbitrary)
3. The FITCH architecture

Architectural components

![Diagram showing the FITCH architecture]

- Adaptation Heuristics
- Adaptation Manager
- Service Clients
- Replicated Service
- Service Gateway
- Cloud IaaS
- Cloud Resource Manager

- Untrusted domain
- Data plane
- Trusted domain
- Control plane
Service model

- FITCH supports diverse replicated services on untrusted domain

CFT services (stateless)

Load-balanced web server clusters
3. The FITCH architecture

Service model

- FITCH supports diverse replicated services on untrusted domain

CFT services (stateless)

- Load-balanced web server clusters

Consistent BFT Services (stateful)

- Paxos-based coordination and storage systems (Google)
3. The FITCH architecture

Service model

- FITCH supports diverse replicated services on untrusted domain

CFT services (stateless)

Many types of replicated services fit in this range

Consistent BFT Services (stateful)

Load-balanced web server clusters

Paxos-based coordination and storage systems (Google)
3. The FITCH architecture

**Service model**

- FITCH supports diverse replicated services on untrusted domain

CFT services (stateless)

Eventually consistent systems

Consistent BFT Services (stateful)

Load-balanced web server clusters

Dynamo (Amazon)

Paxos-based coordination and storage systems (Google)
3. The FITCH architecture

Service model

- FITCH supports diverse replicated services on untrusted domain
3. The FITCH architecture

Service adaptation

- 3 basic operations in service group membership
  - Add replica
  - Remove replica
  - Replace replica

- And what about FITCH unavailability?

  Are all FITCH components available?

  Yes
  - FITCH supports adaptations in the replicated service

  No
  - The replicated service still can be available
  - But adaptations operations are not
3. The FITCH architecture

Service adaptation algorithm

If adding:
1. Create a new VM R1
2. Add R1 to the service group
3. Remove a replica R2 from the service group
4. Destroy the VM of R2

If removing:

If replacing:

Adaptation Heuristics

Adaptation Manager

Service Gateway

Cloud Resource Manager
4. Implementation

2 use cases

• A CFT web service (stateless)
  o Tolerates only crash faults
  o Service implemented by WS-Test using Java and Apache Tomcat
  o Clients access through a LVS (Linux Virtual Server) load balancer
  o WS-Test benchmark

• A consistent BFT key-value store (stateful)
  o Tolerates Byzantine faults (arbitrary)
  o Service implemented with BFT-SMaRt (https://code.google.com/p/bft-smart/)
  o Clients access through a service lookup
  o YCSB benchmark
## 5. Experimental evaluation

### Experimental environment

<table>
<thead>
<tr>
<th>Component</th>
<th>Qty.</th>
<th>Software</th>
<th>Hardware</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adaptation Manager</td>
<td>1</td>
<td>Java</td>
<td>Dell PowerEdge 850</td>
</tr>
<tr>
<td>Client (stateless)</td>
<td>5</td>
<td>WS-Test</td>
<td>1 core, 2.8 GHz</td>
</tr>
<tr>
<td>Cloud RM</td>
<td>3</td>
<td>OpenNebula</td>
<td>2 GB RAM</td>
</tr>
<tr>
<td>Client (stateful)</td>
<td>1</td>
<td>YCSB</td>
<td>Dell PowerEdge R410</td>
</tr>
<tr>
<td>Service Gateway</td>
<td>1</td>
<td>LVS (stateless)</td>
<td>8 cores, 2.27 GHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Tomcat (stateful)</td>
<td></td>
</tr>
<tr>
<td>Physical Cloud Hosts</td>
<td>6</td>
<td>Xen</td>
<td>32 GB RAM</td>
</tr>
</tbody>
</table>
5. Experimental evaluation

3 experiments

• Proactive recovery
• Scale-out and scale-in
• Scale-up and scale-down
5. Experimental evaluation

1st experiment: Proactive recovery

- **Recover** the entire group of replicas **proactively** (one replica at a time)

- **Recover** the entire service repeatedly **as soon as possible** (worst scenario)
5. Experimental evaluation

Proactive recovery

Impact on a CFT web service (stateless)
5. Experimental evaluation

4 replicas
4 adaptations
6 min experiment time
90 s each adaptation time
5. Experimental evaluation

**Warm-up (30 s)**
(Apache Tomcat loading servlet + JIT)
Recoveries increase latency **20- to 30-fold**
(Load balancer reconfig. (~2 s) + warm-up (~7 s))
5. Experimental evaluation

Difference of 60%, concentrated in 7.6% of the time
5. Experimental evaluation

Proactive recovery

Impact on a BFT key-value store (stateful)
5. Experimental evaluation

4 replicas
4 adaptations
13 min experiment time
3 min each adaptation time

![Graph showing latency in ms over time with and without recovery.]
5. Experimental evaluation

Warm-up (45 s) (JIT)
5. Experimental evaluation

Recoveries increase latency **3- to 10-fold**
(Reconfiguration (w/ JIT) + state transfer (~35 s) + warm-up (~12 s))
5. Experimental evaluation

Service stops (PUT) during $3 \text{ s}$
(Leader election protocol)
5. Experimental evaluation

Difference of 12%, concentrated in 29% of the time
2nd experiment: Scale-out and scale-in

- Changes only the number of replicas
- Applied only in the CFT service
5. Experimental evaluation

Scale-out and scale-in

A CFT web service (stateless)
5. Experimental evaluation

2 replicas

6 adaptations

30 min experiment time

5 min each adaptation time

![Graph showing latency over time with adaptive replicas and adaptations]
5. Experimental evaluation

Group **size**

![Graph showing latency over time with scale-out and scale-in points]
5. Experimental evaluation

Latency (in ms)

[Graph showing latency over time with specific values at different time points]
Latency **spikes** when **inserting** replicas …
(Load balancer reconfig. + warm-up)
5. Experimental evaluation

... but **does not** when **removing**

(Load balancer protocol is simpler + no JIT)
5. Experimental evaluation

3rd experiment: Scale-up and scale-down

Scale-up (+$)

Scale-down (-$)

- Increase/Reduce the CPU and RAM for each replica
- Applied only in the BFT service

<table>
<thead>
<tr>
<th>VM</th>
<th>ECU</th>
<th>RAM</th>
<th>US$/h (EC2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small</td>
<td>1</td>
<td>2 GB</td>
<td>0.06</td>
</tr>
<tr>
<td>Medium</td>
<td>2</td>
<td>4 GB</td>
<td>0.12</td>
</tr>
<tr>
<td>Large</td>
<td>4</td>
<td>8 GB</td>
<td>0.24</td>
</tr>
</tbody>
</table>
5. Experimental evaluation

Scale-up and scale-down

A BFT key-value store (stateful)
5. Experimental evaluation

4 replicas

4 adaptations

2.2h experiment time

30 min each adaptation time
5. Experimental evaluation

Group types

![Diagram showing scale-up and scale-down points over time](image-url)
5. Experimental evaluation

Each scale-up or scale-down is comprised of 4 replacements
5. Experimental evaluation

**Latencies** (in ms)

![Graph showing latencies over time](image-url)
5. Experimental evaluation

**Throughput** (in ops/s)

![Graph showing throughput in ops/s over time with latency in ms]
There are still opportunities to improve replicated services performance, dependability and cost-efficiency with proper usage of cloud computing dynamism.

- We presented FITCH:
  - An infrastructure to support the dynamic adaptation of replicated services in cloud environments
  - 3 basic operations (add, remove and replace replicas)
  - 2 representative services
    - A CFT web service (stateless)
    - A consistent BFT key-value store (stateful)
  - 3 adaptation scenarios
    - Proactive recovery
    - Scale-out and scale-in
    - Scale-up and scale-down
Thank you!

vielmo@lasige.di.fc.ul.pt
http://lasige.di.fc.ul.pt/~vielmo

LaSISE: Large-Scale Informatics Systems Laboratory
Fundação para a Ciência e Tecnologia (FCT)
Multiannual Program
http://lasige.di.fc.ul.pt

CloudFIT: Fault-and-Intrusion Tolerance for Clouds
Fundação para a Ciência e Tecnologia (FCT)
PTDC/EIA-CCO/108299/20008
http://cloudfit.di.fc.ul.pt

TClouds: Trustworthy Clouds
EU’s 7th Framework Program
ICT-257243
http://tclouds.eu/