# Proactive Cloud Management for Highly Heterogeneous Multi-Cloud Infrastructures

Alessandro Pellegrini
Pierangelo Di Sanzo
Dimiter R. Avresky

Sapienza, University of Rome
IRIANC

DPDNS 2016

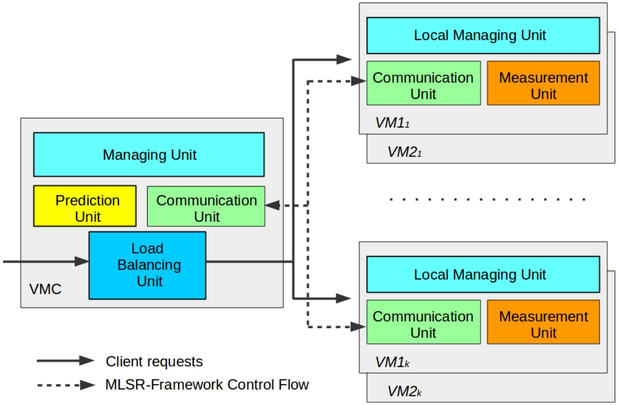
## Motivations

- The cloud computing paradigm can help to improve availability and performance of applications subject to the problem of software anomalies
- Prompt access to new resources (even geographically distributed)
  - To be used in case of crashes
  - To be used for load balance in case of overloads
- Managing a complex geographically-distributed cloud deploy could be a complex and time-consuming task
- Hybrid Clouds make the deploy more complex
  - *Cloud region* can be highly heterogeneous: number of VMs, allocated resources, ...

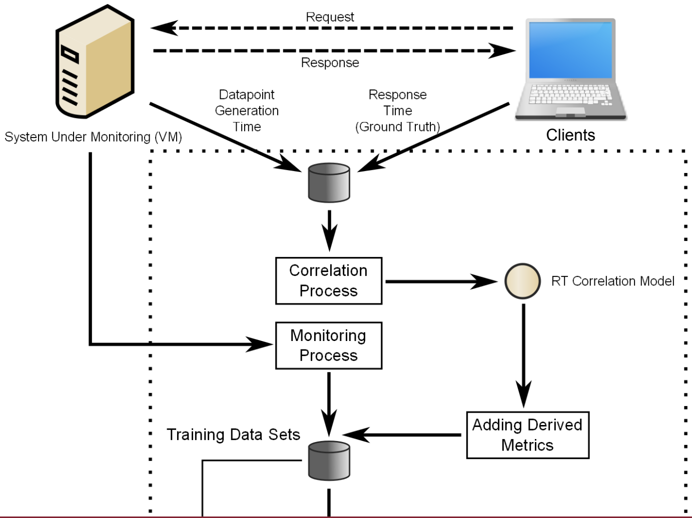
## Autonomic Cloud Manager (ACM) Framework

- It is an autonomic framework for supporting proactive management of applications deployed over multiple cloud regions
- It uses machine learning models
  - to predict failures of virtual machines
  - to proactively redirect the load to healthy machines/cloud regions
- It supports different policies to perform efficient proactive load balancing across cloud regions in order to mitigate the effect of software anomalies
- It is completely *agnostic* of the running application

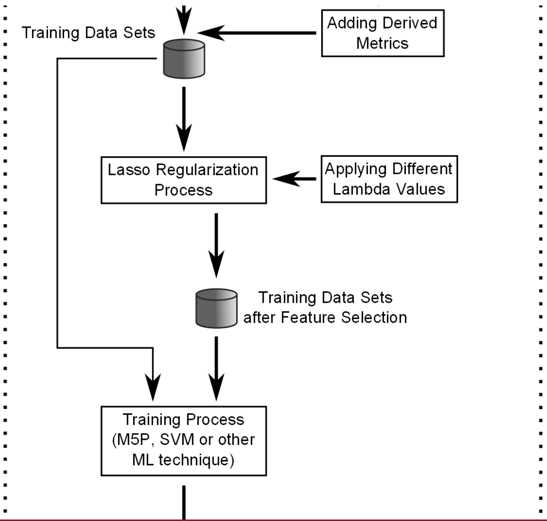
## Organization at one Cloud Region



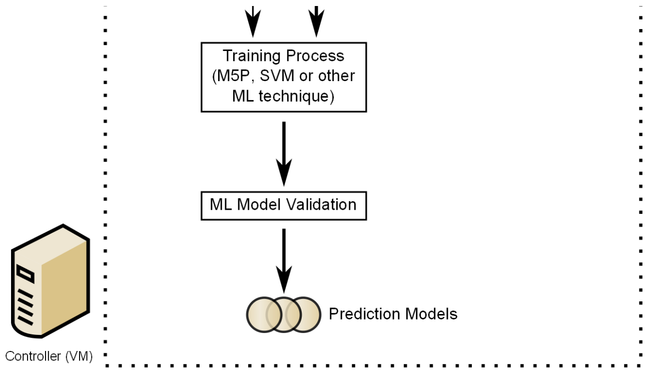
## Building Prediction Models (1)



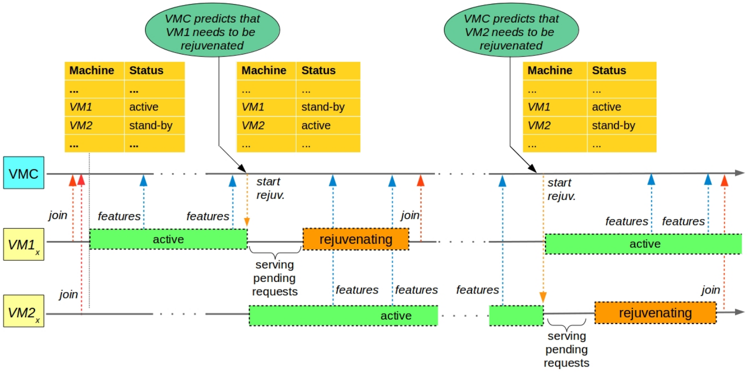
## Building Prediction Models (2)


Training Data Sets
Adding Derived Metrics
Lasso Regularization Process
Applying Different Lambda Values
Training Data Sets after Feature Selection
Training Process (M5P, SVM or other ML technique)

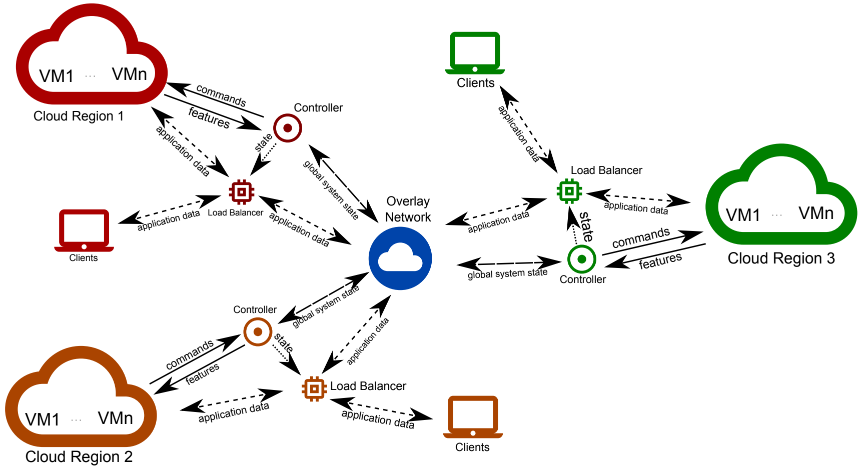
## Building Prediction Models (3)



## Rejuvenation Schema



## Distributed Organization



## Load Balancing in Highly Heterogeneous Environments

- Our policies aim at performing efficient proactive load balancing across cloud regions in order to avoid that different failure and rejuvenation rates in different regions lead to overloaded and underloaded region
- the goal of these policies is to ensure that all active VMs in all regions show the same Mean Time To Failure (MTTF) in front of the heterogeneity of regions in terms of number and computing power of VMs
- the MTTF of VMs hosted in a cloud region is estimated by the ML models
- The VMC of a region *i* periodically sends to the leader VMC the last average value of the Region Mean Time To Failure (RMTTF)

## Policy 1: Sensible Routing

- When the leader VMC receives $lastRMTTF_i$ at time $t$, the current RMTTF of the region $i$, say $RMTTF_i^t$, is (re-)calculated by using the following weighted average:

$$RMTTF_i^t = (1-\beta) \cdot RMTTF_i^{t-1} + \beta \cdot lastRMTTF_i, \qquad (1)$$

- $RMTTF_i^{t-1}$ is the previous value of RMTTF and $0 \leq \beta \leq 1$.
- Assuming to have $N$ cloud regions, the fraction $f_i$ of global incoming requests to be forwarded to cloud region $i$ is calculated as:

$$f_i = \frac{RMTTF_i^t}{\sum_{j=1}^{N} RMTTF_j^t}. \qquad (2)$$

## Policy 2: Available Resources Estimation

- This policy uses a single numeric parameter as an abstraction to quantify the amount of available resources in a region
- The *estimation* of the amount of available resources a in region $i$ is calculated as:

$$Q_i = RMTTF_i^t \cdot f_i \cdot \lambda \tag{3}$$

- $\lambda$ is the global incoming request rate, thus $f_i \cdot \lambda$ is the incoming request rate of region $i$.
- The fraction of requests to be forwarded to region $i$ is calculated as as:

$$f_i = \frac{Q_i}{\sum_{j=1}^{N} Q_j}. \tag{4}$$

## Policy 3: Exploration

- The third policy uses an exploration strategy, as it is inspired to the hill climbing search algorithm
- This policy calculates the Average RMTTF (ARMTTF) over all regions:

$$ARMTTF = \frac{\sum_{i=1}^{n} RMTTF_i^t}{N}. \tag{5}$$

- all regions for which $RMTTF_i^t > ARMTTF$ get their current value $f_i$ decreased, while the regions with $RMTTF_i^t < ARMTTF$ get their value $f_i$ increased.

## Policy 3: Exploration (2)

- It selects all the regions such that $RMTTF_i < ARMTTF$ (which we call the set of overloaded regions $OL = \{i : RMTTF_i < ARMTTF\}$)
- For each of these regions it computes the new value of the fraction $f_i$, say $f_i^{next}$ as:

$$f_i^{next} = \frac{RMTTF_i}{ARMTTF} \cdot f_i \cdot k \tag{6}$$

- $k$ is a constant scaling factor
- The equality $\sum_{i=1}^{n} f_i = 1$ must hold. To this end, it must be ensured that any portion taken out of some $f_i$ must be added to some $f_j$, $i \neq j$.

## Policy 3: Exploration (3)

- The policy computes the *total variation of the flow of overloaded regions*:

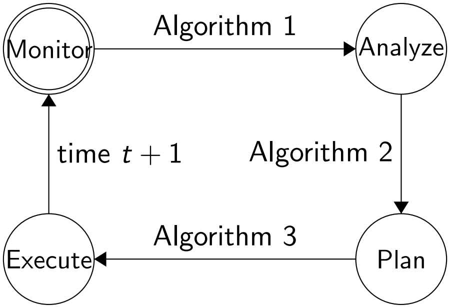
$$\Delta f^{<} = \sum_{i \in UL} (f_i^{next} - f_i) \tag{7}$$

- Then it selects all the regions such that $RMTTF_i > ARMTTF$ (which we call the set of underloaded regions $UL = \{i : RMTTF_i > ARMTTF\}$)
- For each of these regions it updates the workload fraction as:

$$f_i^{next} = \frac{\Delta f^{<}}{\sum_{i=1}^{N} RMTTF_i} \cdot f_i \cdot k \tag{8}$$

- $k$ is the same scaling factor.

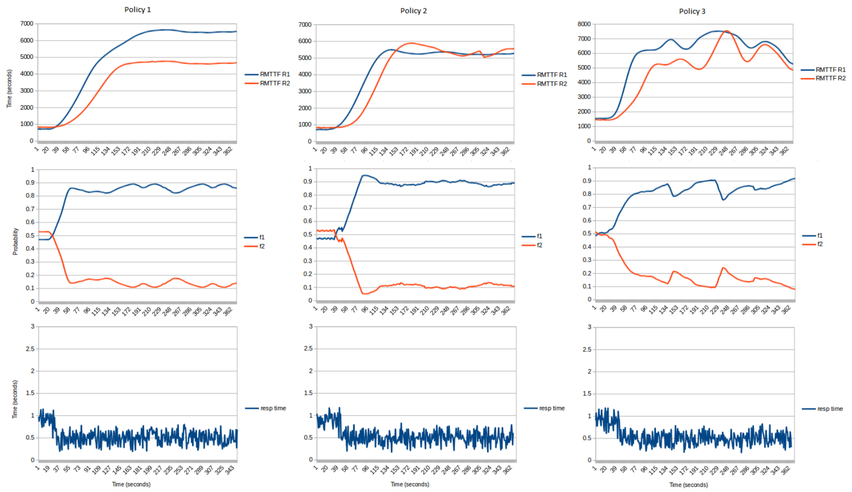
## The ACM Closed Control Loop



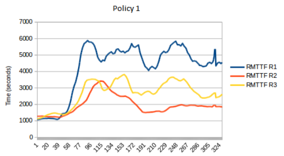
## Experimental Setup

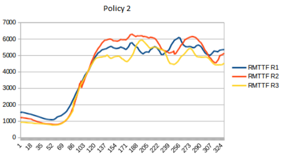
- Hybrid cloud architecture
  - Region 1, hosted in the Ireland Region of Amazon EC2
  - Region 2, hosted in the Frankfurt Region of Amazon EC2
  - Region 3, privately hosted in a 32-cores HP ProLiant located in Munich (Germany)
- The test-bed application was the TPC-W benchmark
- We used a Java implementation of TPC-W
- It relies on MySQL

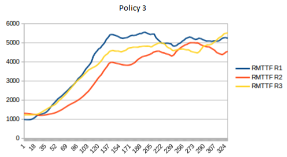
# Experimental Results

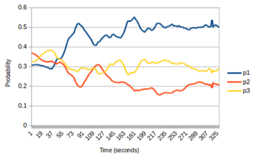


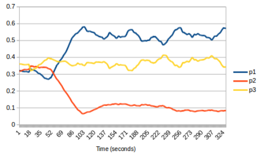
## Experimental Results

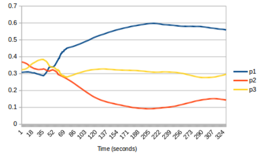












## Thanks for your attention

Questions?

pellegrini@dis.uniroma1.it
http://www.dis.uniroma1.it/∼pellegrini