Experimental Analysis on Autonomic Strategies for Cloud Elasticity

SIMON DUPONT, JONATHAN LEJEUNE, FREDERICO ALVARES THOMAS LEDOUX

Summarized by: Pranav Veldurthy
Dept. Of Computer Science
INTRODUCTION

- Cloud Elasticity is defined as “The degree to which a system is able to adapt to workload changes by provisioning and deprovisioning resources in an autonomic manner, such that at each point in time the available resources match current the demand as closely as possible”.

- Autonomic Cloud Elasticity is mainly used to scale the infrastructure resources in the IaaS layer.

- Limitations:
  - Resources are limited
  - Initiation time
  - Partial usage waste
  - Energy Consumption

- Software at the SaaS layer can take part in the elasticity process to overcome these limitations.

- Software Elasticity acts as an extra elasticity capability that goes beyond the infrastructure elasticity when resource are scarce.

- Autonomic approach to manage cloud elasticity in a cross-layered manner is implemented.

- Experimental analysis on the elastic tactics that can drive the approach through OpenStack is applied.
INFRASTRUCTURE ELASTICITY

- Infrastructure Elasticity (IaaS): The ability to scale infrastructure resources on demand. (Ex: VMs).
  - Infrastructure Horizontal Scaling ($HS_{infra}$): adjusts VM’s pool size by adding or removing instances
    - $SO_{infra}$
    - $SL_{infra}$
  - Infrastructure Horizontal Scaling ($VS_{infra}$): resizes existing VM’s by increasing or decreasing resources
    - $SU_{infra}$
    - $Sd_{infra}$
LIMITATIONS

- Resources are Limited:
  - IaaS provider restricts resources instances allocated for each user to manage datacenters
  - Failure will decrease resource availability
  - Amount of used resources based on the predefined budget

- Resource Initiation time is significant
  - Delay in resource initiation time preventing just-in-time provisioning
  - Larger the VM, larger the time needed to configure, deploy and launch it.

- Pricing is per instance-hour
  - Pay as you go billing model
  - SaaS providers might end up paying up more than they consume.
Software Elasticity – The capability of a software to adapt itself to meet demand changes and/or infrastructure resources limitations.

- Software Horizontal Scaling ($HS_{\text{soft}}$) : Accomplished on SaaS layer
  - $SO_{\text{soft}}$
  - $SI_{\text{soft}}$

- Software Vertical Scaling ($VS_{\text{soft}}$) : Inspired from Vertical Scaling to extend software elasticity.
  - $SU_{\text{soft}}$
  - $SD_{\text{soft}}$
Benefits of IaaS and SaaS elasticities complementarily

- Alleviate the use of infrastructure resources – The resources cannot be scaled up infinitely and the maximum resource utilization can be attained.
- Improve responsiveness in scaling – High dynamic environment systems require modularity and flexibility but should also provide rapid reconfiguration capabilities.
- Improve expression capability of elasticity – Cloud administrators have the right to create more sophisticated scaling dimensions and APIs.
Application behaves in an autonomous manner by choosing appropriate $\text{Off}_{\text{comp}}$ for advertising component with the current workload, runtime context and some constraints on the advertising component in accordance with the current workload, the runtime context and some constraints on the advertisement quality, the performance and the energy compliance.
ELASTIC STRATEGIES FOR AUTONOMIC CLOUD SERVICES

- **PaaS Autoscaling Service**
  - Manages elasticity of resources (Eg: Software component, VMs, Physical Infrastructure)
  - Cloud Administrator (CA) manages to monitor and act on both IaaS and SaaS resources
- **Autonomic Cloud Model overview**
  - Implements MAPE – K loop reference model
  - Interacts with managed system through sensors and actuators
ELASTIC STRATEGIES FOR AUTONOMIC CLOUD SERVICES

- **Resource Model**: managed system  
  - n-tier web applications comprises of VMs accommodating software components.  
  - Illustration of a two-tier application containing two VMs.  
  - Left represents physical architecture and right represents logical architecture.

- **Monitor**: Events  
  - Managed system has a metrics through sensors making it possible to capture their values at runtime.  
  - Basic Event, denotes pertinent event for one metric.  
  - Complex Events, composite events which involve one or few metrics with different time scales.

- **Execute**: predefined Actions  
  - Resources are operated through actuators offering reconfiguration capability to the system  
  - Basic Action, cloud elastic scaling actions  
  - Complex Actions, configuration of basic actions executable in a self-contained reconfiguration plan.
ELASTIC STRATEGIES FOR AUTONOMIC CLOUD SERVICES

- Reconfiguration Decisions, forms the bridge between the M and K from the MAPE – K loop.
  - Tactics Filter - Event-Action Mapping
    - The link between event patterns and predefined patterns is done through Tactics.
    - Automatic filtration is done during runtime based on the triggered events.

- Constraint Filter – Context
  - Inputs the eligible tactics and run-time constraints to identify unsuitable tactics and outputs only the applicable tactics.

- Preferences Filter – Strategy
  - Several factors are taken into consideration when choosing the eligible tactic after an event occurs at runtime such as, Cost, QoS, QoE, Elasticity Time and Responsiveness.
EXPERIMENTS

- Application configuration
  - Cloud based application in a SaaS-fashion.
  - Consists two tiers – load balancing tier (lbt) and business tier (bt)
  - Lbt is software elastic and offers several levels
- Infrastructure configuration
  - Grid’5000, 7 physical machines connected by 20Gbit/s ethernet switch
  - Each machine has 2.8 GHz Xeon Processor, 15GB RAM running Linux 2.6.
- Autonomic Manager
  - Monitors average response time of received and processed requests in 15 seconds.
  - Generates two event patterns
    - High Response time – Occurs when monitored values is higher than an upper threshold values. Reflects under-provisioning leading to SLO violation
    - Low Response time – Occurs when monitored values is lower than a lower threshold values. Reflects over-provisioning leading to useless operational cost
EXPERIMENTS

- **Experimental Evaluation**
  - At this instances, the upper and lower thresholds are fixed at 400ms and 20ms and evaluation is done using three experiments
    - Experiment 1: $SO_{infra}$ and $SI_{infra}$, where $SO_{infra}$ adds one virtual machine
    - Experiment 2: $SU_{soft}$ and $SD_{soft}$, where action degrades one level $Off_{comp}$ for each running component instance/virtual machine in tier
    - Experiment 3: $SU_{soft}$ and $Si_{infra}$, where $SU_{soft}$ is a composite action.
  - The experiments are conducted in three phases
    - **First Phase**: The workload increases during 10 minutes with a medium speed-up and then decreases during 3 minutes up to starting point
    - **Second Phase**: The workload increases during 10 minutes with a half speed-up of the first phase and then decreases during 3 minutes up to starting point
    - **Third Phase**: Three peak loads are injected at regular interval characterizing a high speed-up for short period of time.
  - Based on these three phases, the following metrics are collected:
    - Infrastructure’s Size
    - Software Offering
    - Number of Failed Requests
    - Average Response Time
EXPERIMENTS
CONCLUSION

- Asserted that SaaS can be a part of cloud elasticity
- Autonomic approach to cloud elasticity in across layered manner.
- Presented approach which mainly relied on dynamic selection of elasticity tactics.
- Currently working on the design of Domain Specific Language (DSL) for cloud elasticity.
- Future work includes to extend the experiments in numerous tactics involving four elasticity with more complex architectures and various types of workloads.