Principles and Methods for Elastic Computing

CompArch Keynote - Lille, 1 July 2014

Schahram Dustdar

Distributed Systems Group
TU Vienna

dsg.tuwien.ac.at
Includes some joint work with Hong-Linh Truong, Muhammad Z.C. Candra, Georgiana Copil, Duc-Hung Le, Daniel Moldovan, Stefan Nastic, Mirela Riveri, Sanjin Sehic, Ognjen Scekic

NOTE: The content includes some ongoing work
Smart Evolution – People, Services, Things

Elastic Systems

Smart Energy Networks

Smart eGovernments & eAdministrations

Smart Transport Networks

Smart Homes

Smart Health Networks
Cloud Computing?

1. “Resources” provided as services

2. Illusion of infinite resources

3. Usage-based pricing model -> New and connected business models
Think Ecosystems: People, Services, Things

Diverse users with complex networked dependencies and intrinsic adaptive behavior – has:

1. **Robustness mechanisms**: achieving stability in the presence of disruption

2. **Measures of health**: diversity, population trends, other key indicators

Marine Ecosystem: http://www.xbordercurrents.co.uk/wildlife/marine-ecosystem-2
Approach

Elastic Computing

People

Software

Things
Computing Models

Connecting machines and people

Core principles:
- Human computation capabilities under elastic service units
- Programming human-based units together with software-based units
Elasticity ≠ Scaleability

Resource elasticity
Software / human-based computing elements, multiple clouds

Quality elasticity
Non-functional parameters e.g., performance, quality of data, service availability, human trust

Costs & Benefit elasticity
rewards, incentives
- Multi-dimensional Elasticity
- Service computing models
- Cloud provisioning models

Elasticity in computing – broad view

1. Demand elasticity
   Elastic demands from consumers

2. Output elasticity
   Multiple outputs with different price and quality

3. Input elasticity
   Elastic data inputs, e.g., deal with opportunistic data

4. Elastic pricing and quality models associated resources
Diverse types of elasticity requirements

- **Application user**: “If the cost is greater than 800 Euro, there should be a scale-in action for keeping costs in acceptable limits”

- **Software provider**: “Response time should be less than amount X varying with the number of users.”

- **Developer**: “The result from the data analytics algorithm must reach a certain data accuracy under a cost constraint. I don’t care about how many resources should be used for executing this code.”

- **Cloud provider**: “When availability is higher than 99% for a period of time, and the cost is the same as for availability 80%, the cost should increase with 10%.”
Internet of Things
and elasticity
Smart City Dubai
Pacific Controls

Command Control Center

- SMEs
- Dashboards
- User interfaces
- Reports
- Carbon footprint measurement
- Benchmarking
- Remote monitoring
- Engineers

Villas
- Fire
- Safety & security
- Energy
- HVAC
- CCTV
- Carbon footprint

Factories
- Fire
- Lift
- Safety & security
- Energy
- Chiller / HVAC
- Boiler
- CCTV
- Carbon footprint

Schools
- Fire
- Lift
- Safety & security
- Energy
- Chiller / HVAC
- Boiler
- CCTV
- Carbon footprint

Commercial & residential buildings
- Fire
- Lift
- Safety & security
- Energy
- Chiller / HVAC
- Boiler
- CCTV
- Carbon footprint

Utilities
- Sewage pumps
- Water treatment plants
- Irrigation

Hospitals
- Fire
- Lift
- Safety & security
- Energy
- Chiller / HVAC
- Boiler
- CCTV
- Carbon footprint
HVAC (Heating, Ventilation, Air Conditioning) Ecosystem
Water Ecosystem
Air Ecosystem
Some 50 billion devices and sensors exist for M2M applications. IoT and Cloud Computing enable smart services ecosystem and collaboration opportunities. Managed services include:

- Portfolio management
- Event management
- Analytics

Provisioning includes:

- Services
- SIM profile configuration
- Network configuration

Controls feature:

- Activation
- Deactivation
- Privacy
- Security

Transaction management involves:

- Visibility
- Billing
- Reporting

Ubiquitous Managed Services Solution Across Business Verticals

Numerous Forms Of Smart Services...
Human Command Control Center for Managed Services
Elasticity Engineering
Specifying and controlling elasticity

Basic primitives

- Elasticity directive primitives
  - Monitoring
  - Constraint
  - Strategy

  - Resource
    - Compute
    - People
    - Storage
    - Network
  - Quality
  - Cost
  - Resource/quality/cost
  - Scale in/out
  - Stop/wait/notify
  - Configure
  - Access

Data/Compute-intensive services

Workflow/Applications/services/Middleware/Systems

Software/Human-intensive services

Hybrid Mixed systems

Business/E-science

Domain-specific/Customized features

High Level Description of Elasticity Requirements

SYBL (Simple Yet Beautiful Language) for specifying elasticity requirements

SYBL-supported requirement levels

Cloud Service Level
Service Topology Level
Service Unit Level
Relationship Level
Programming/Code Level

#SYBL.CloudServiceLevel
Cons1: CONSTRAINT responseTime < 5 ms
Cons2: CONSTRAINT responseTime < 10 ms
WHEN nbOfUsers > 10000
Str1: STRATEGY CASE fulfilled(Cons1) OR
fulfilled(Cons2): minimize(cost)

#SYBL.ServiceUnitLevel
Str2: STRATEGY CASE ioCost < 3 Euro :
maximize( dataFreshness )

#SYBL.CodeRegionLevel
Cons4: CONSTRAINT dataAccuracy>90%
AND cost<4 Euro

High Level Description of Elasticity Requirements

Current SYBL implementation

in Java using Java annotations

@SYBLAnnotation(monitoring="",constraints="",strategies="")

in XML

<ProgrammingDirective><Constraints><Constraint
name=c1>...</Constraint></Constraints>...</ProgrammingDirective>

as TOSCA Policies

<tosca:ServiceTemplate name="PilotCloudService">  
tosca:Policy name="St1"
policyType="SYBLStrategy">  
St1:STRATEGY minimize(Cost) WHEN high(overallQuality)
</tosca:Policy>...

Other possibilities

C# Attributes

[ProgrammingAttribute(monitoring="",constraints="",strategies="")]  

Python Decorators

@ProgrammingDecorator(monitoring,constraints,strategies)
Mapping Services Structures to Elasticity Metrics
Multi-level Control Runtime: Generating Elasticity Control Plans

Algorithm 3 Constraint enforcement

Input: graph - Cloud Service Dependency Graph
Output: ActionPlan

1: violatedConstraints = findAllViolatedConstraints(model)
2: while size(violatedConstraints) > 0 do
3:     for each level in cloudServiceAbstractionLevel do
4:         violatedConstraintsL = selectConstraints(violatedConstraints, level)
5:         actionSet = evaluateLevelEnabledActions(graph, violatedConstraintsL)
6:         Action = findAction(actionSet) with max(constraints fulfilled - violated)
7:         Add action Action to ActionPlan
8:         estimatedActionEffect(Action))
9:     end for
10: end while
return ActionPlan

Cloud Providers/Tools must support higher and richer APIs for elasticity controls
Elasticity Model for Cloud Services


Elasticity Pathway functions: to characterize the elasticity behavior from a general/particular view

Elasticity space functions: to determine if a service unit/service is in the "elasticity behavior"
MELA -- Elasticity Monitoring as a Service

Multi-Level Elasticity Space

Service requirement

- COST $ \leq 0.0034$/client/h
- 2.5$ monthly subscription for each service client (sensor)

- Determined Elasticity Space Boundaries
  - Clients/h $> 148$
  - $300\text{ms} \leq \text{ResponseTime} \leq 1100 \text{ms}$
Multi-Level Elasticity Pathway

Service requirement

\[ \text{COST} \leq 0.0034\$/\text{client/h} \]

2.5\$ monthly subscription for each service client (sensor)
Specifying and controlling elasticity of human-based services

What if we need to invoke a human?

#predictive maintenance analyzing chiller measurement
#SYBL.ServiceUnitLevel
Mon1 MONITORING accuracy = Quality.Accuracy
Cons1 CONSTRAINT accuracy < 0.7
Str1 STRATEGY CASE Violated(Cons1):
Notify(Incident.DEFAULT, ServiceUnitType.HBS)
Human-based service elasticity

- Which types of human-based service instances can we provision?
- How to provision these instances?
- How to utilize these instances for different types of tasks?
- Can we program these human-based services together with software-based services?
- How to program incentive strategies for human services?
ICU/SCU for independent tasks

Consumer

Application / Workflow

ICU Clouds
- Task-based crowdsourcing platforms
- Collections of experts on SN
- Enterprise ICU pools

Provisioning Middleware

SCU

Human-Task Requests

Provisions

SCU

- ICU: individuals
- SCU: a collective of collaborative individuals

Problems
- Complex tasks
- Quality control
- Flexible quality requirements
Elastic SCU provisioning atop ICUs

SCU extension/reduction
- Task reassignment based on trust, cost, availability

Algorithms
- Ant Colony Optimization variants
- FCFS
- Greedy

Cloud APIs


Conclusions (1) – Engineering Elasticity

The evolution of underlying systems and the utilization of different types of resources under different models for elasticity requires:

- Complex, open hybrid service unit provisioning frameworks
- Different strategies for dealing with different types of tasks
- Quality issues for software, data, and people in an integrated manner
Conclusions (2) – Engineering Elasticity

- Service engineering analytics of elastic systems
  - Programming hybrid compute units for elastic processes
  - Elasticity specifications and reasoning techniques
  - Elasticity spaces analytics

- Application domains
  - “Social computer“ and smart cities (FP 7 FET Smart Cities and PC3L)
  - Computational science and engineering (FP 7 CELAR)
Thanks for your attention!

Prof. Dr. Schahram Dustdar
Distributed Systems Group
TU Wien

dsg.tuwien.ac.at