Dependable and Resilient Cloud Computing

Vincenzo Piuri
Università degli Studi di Milano, Italy
vincenzo.piuri@unimi.it

Simon Fraser University
Vancouver, BC, Canada
25 April 2018

Based on joint work with: M. Albanese, S. Jajodia, R. Jhawar

Outline

- Motivation
- System overview
  - Vulnerability and failure characteristics of Cloud infrastructures
  - Security and fault tolerance of the mission
- Secure mission deployment and mission protection
- Fault tolerance of the mission
  - Fault tolerance as a service
  - Constraints-aware resource provisioning
  - Adaptive resource management

Motivation (1)

- Cloud computing is becoming increasingly popular
  - \+ Flexibility in obtaining and releasing computing resources
  - \+ Lower entry and usage costs
  - \+ Effective for applications with high scalability requirements
- Growing interest among users to leverage Cloud-based services to execute critical missions
- Exacerbate the need to ensure high security and availability of the system and the missions

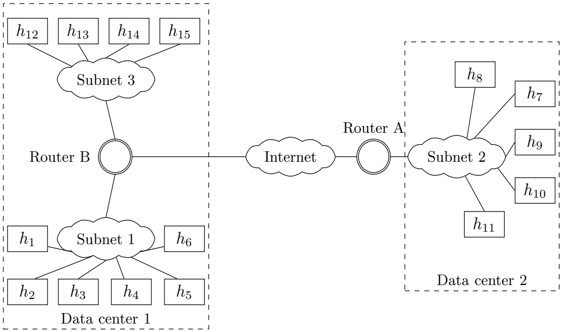
Motivation (2)

- Cloud computing infrastructure is highly complex
  - Vulnerable to various cyber-attacks and subject to failures
  - Outside the control scope of the user's organization
- Existing solutions individually focus on the security of the infrastructure and the mission
  - Do not take into account the interdependencies between them
- Fault tolerance methods are typically applied during development
  - Unfeasible to combine failure behavior and system architecture in the Cloud due to the abstraction layers

Motivation (3)

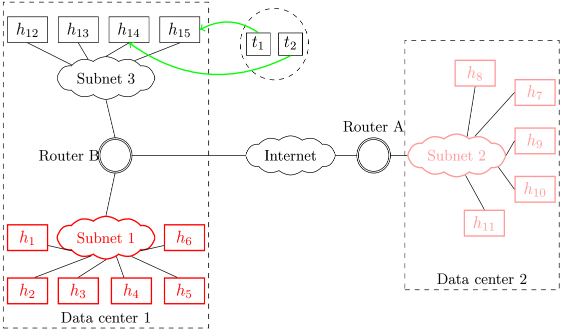
- User-centric approach to address the security and fault tolerance issues
  - Deploy missions so as to minimize their exposure to the vulnerabilities in the Cloud
  - Protect the hosts and network links used by the mission
  - Deliver fault tolerance as a service to the mission
  - Response to faults at run-time
  - Response to security attacks at run-time

System Overview – Cloud Infrastructure



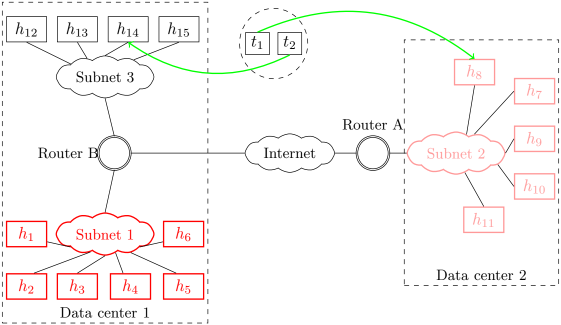
- Redundant switches, routers and links for fault tolerance
- Security tools (e.g., intrusion detection, firewalls)

System Overview – Cloud Infrastructure



- Hosts may be vulnerable to various cyber-attacks
  (e.g., Subnet 1: compromised, Subnet 2: vulnerable, Subnet 3: highly secure)

System Overview – Cloud Infrastructure

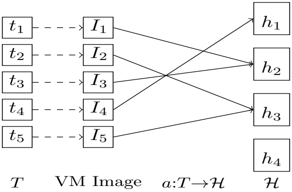


- Tasks may have vulnerability tolerance capability
  (e.g., Task 1 can handle buffer overflow attacks using memory management mechanisms)

Mission Deployment

Static Mission Deployment

- Each host $h \in \mathscr{H}$ is associated with a vulnerability value V_h
- $tol(t)$ provides an estimate of the maximum level of vulnerability the task can be exposed to
- Task allocation problem with two sub-problems
    - Map each task to an appropriate VM image in the repository
    - Allocate VMs on suitable physical hosts in the Cloud



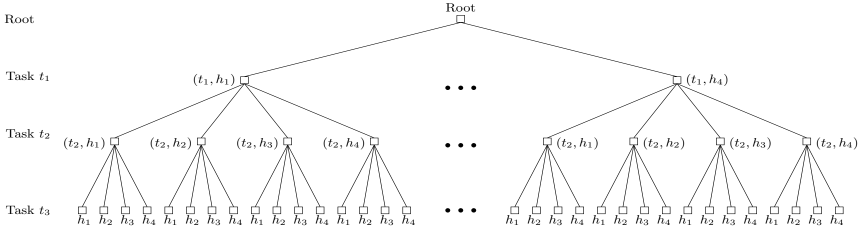
Selecting VM Images

- Challenge: Develop techniques to assess security of VM images at run-time and an automated security-driven search scheme to deploy mission tasks
- VM images encapsulate the entire software stack and determine the initial state of running VM instances
- Most Cloud IaaS require users to manually select VM images; in public Cloud services, VM images have critical vulnerabilities
- Objective: Select VM images that satisfy both functional requirements and security policy of mission tasks

Allocating VMs on Cloud Infrastructure (1)

- Challenge: Develop approximation algorithms to find suboptimal allocation solution in a time-efficient manner
- Objective is to minimize exposure of mission tasks to the vulnerabilities in the Cloud infrastructure
- Satisfy additional dependability constraints
  (e.g., host's capacity and task's vulnerability tolerance constraint)

Allocating VMs on Cloud Infrastructure (2)



- Possible solution: Use A^*-based state-space search approach
- State is a possible choice for allocating a task on a host (t_i, h_j)
- Root state is the initial state where no task is allocated
- Operation generates child states for a given state s
- Goal state is a state in which all the tasks have been allocated (leaf)
- Solution path is the path from root state to any goal state

Allocating VMs on Cloud Infrastructure (3)

- Objective is to find the solution path with minimum vulnerability value
- Cost function is the vulnerability measure of complete allocation $fvul(s) = gvul(s) + hvul(s)$
- $gvul(s)$ is the total minimum vulnerability due to task allocation from the root state to the current state s
- $hvul(s)$ is the lower-bound vulnerability estimate of the allocation from the current state to any goal state
  - $hvul(s)$ is computed using an admissible heuristic
  - Improves search performance while not compromising optimality

Dynamic Mission Deployment (1)

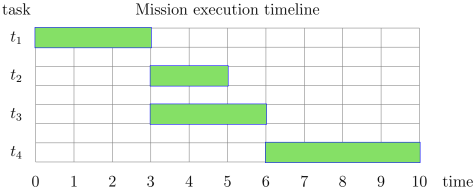
- Each task is associated with temporal constraints
  (e.g., a task may only run after another task)

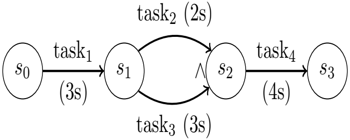
- Critical missions must complete within a certain amount of time

- Possible solution: Complex task scheduling solution that takes into account the capability of the VM while computing the solution

Dynamic Mission Deployment (2)

- Challenge: Schedule mission tasks on the hosts
  1. To minimize their exposure to the vulnerabilities in the network
  2. To ensure their deadlines are met



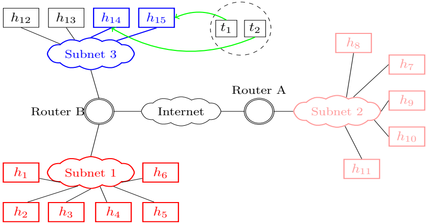


- Critical tasks (e.g., task t_3) must be placed on highly reliable host
- Adopt scheduling schemes such as greedy heuristics, genetic algorithms, tabu search, A* to solve the scheduling problem

Mission Protection

Static Network Hardening

- Challenge: Given a mission is deployed in the Cloud, protect the resources used by the mission tasks
- In the static version, all the hosts and network links are protected for the entire duration of mission execution



- Possible solution: Build on top of previous work for network hardening

Dynamic Mission Protection

- Challenge: At any point in time, find a cost-optimal time-varying strategy to harden the resources not yet used by the mission
- Dynamic protection minimizes the disruption that hardening strategy causes to legitimate users
- Possible solution: Efficient technique that analyzes huge streams of security threats at real-time
- For example, use of attack graphs to track where the attacker is going (the penetration path)

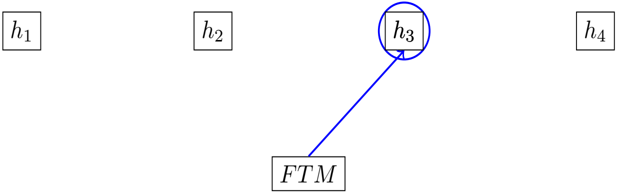
Fault Tolerance of the Mission

Fault Tolerance Support for Mission

- Realize the notion of Fault tolerance as a Service
- Fault tolerance mechanisms based on the virtualization technology (e.g., checkpointing virtual machine instances)
  + introduce fault tolerance in a transparent manner
  + offers high level of generality
  + Possible to change fault tolerance properties based on business needs
- Construct dependability mechanisms at runtime
  - Mission centric Service Level Agreement (SLA)

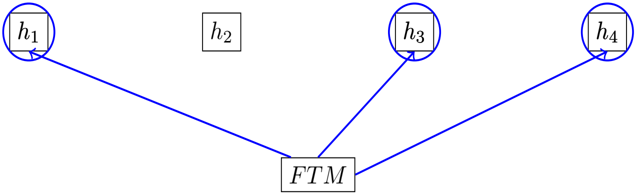
Fault Tolerance as a Service

- Build and deliver the service by orchestrating a set of micro-protocols
  - Realize fault tolerance techniques as independent, stand-alone, configurable modules (web services)
  - Operate at the level of virtual machine instances
- VM instance replication technique



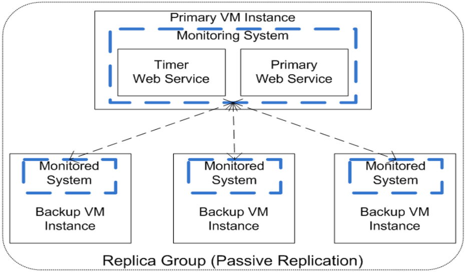
Fault Tolerance as a Service

- Build and deliver the service by orchestrating a set of micro-protocols
  - Realize fault tolerance techniques as independent, stand-alone, configurable modules (web services)
  - Operate at the level of virtual machine instances
- VM instance replication technique



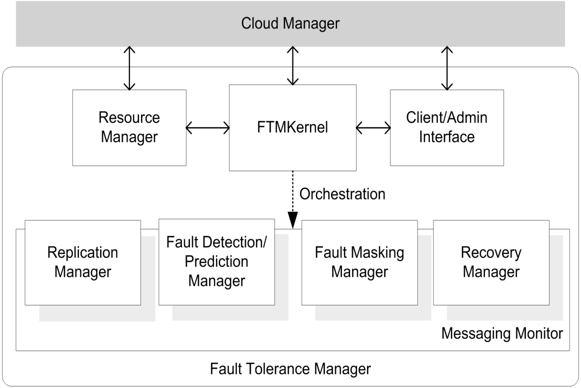
Fault Tolerance as a Service

- Build and deliver the service by orchestrating a set of micro-protocols
  - Realize fault tolerance techniques as independent, stand-alone, configurable modules (web services)
  - Operate at the level of virtual machine instances
- Failure detection using hearbeat test



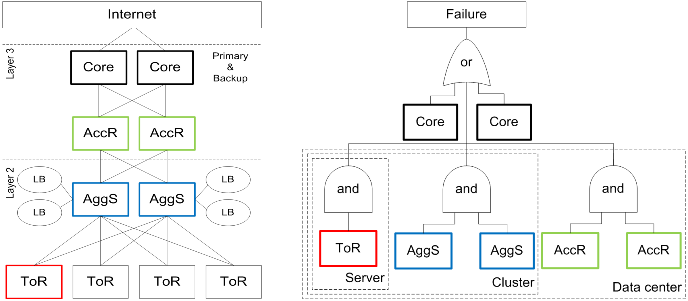
Fault Tolerance Manager

Fault Tolerance Manager – Framework Overview



Configuration of a Dependability Solution (1)

- Based on the affect of failures on mission's tasks
- Using Fault trees and Markov chains

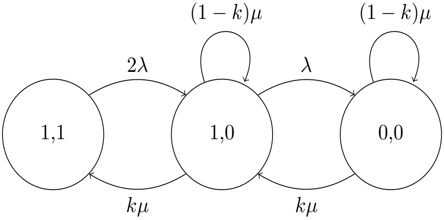


ToR–Top of Rack Switch   AggS–Aggregate Switch

AccR–Access Router   LB–Load Balancer

Configuration of a Dependability Solution (2)

- Analyze the properties of typical dependability mechanisms
- For example, semi-active replication
- Primary, Backup (λ – failure rate, μ – recovery rate, k – constant)



Matching and Comparison Process

- Represent fault tolerance properties of ft_sols using $p=(s,\hat{p},A)$
  - s denotes the ft_sol
  - $\hat{p}$ represents the high level abstract properties such as reliability and availability
  - A denotes the set of structural, functional and operational attributes
- Based on the mission's fault tolerance requirements
  - for each ft_sol $s \in S$ in the system, first shortlist $S' \subset S$ that satisfy abstract property requirements $\hat{v}_c(a) \preceq \hat{v}_i(a)$
  - for each ft_sol in S', compare $v_c(a) \preceq v_i(a)$ attribute values to obtain a set S'' of candidate ft_sols

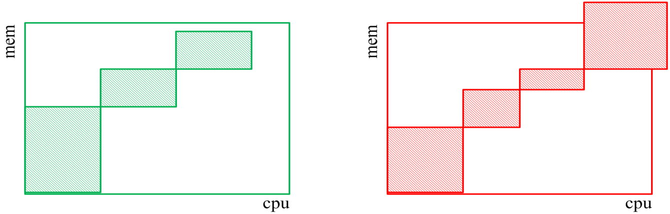
Replica Placement Constraints

- Location and performance requirements of replicas can be specified using constraints
  - Global constraints – Resource Capacity
  - Infrastructure oriented constraints – Forbid, Count
  - Application oriented constraints – Restrict, Distribute, Latency

Resource Capacity Constraint

- To avoid inconsistent system state
- Resources consumed by all VM instances on a single host cannot exceed a specified threshold of host's capacity

$$\forall h \in \mathscr{H},\ \ d \in \mathscr{D}, \quad \sum_{v \in \mathscr{V} | p(v) = h} v[d] \ \leq \ (h[d] * threshold[d])$$

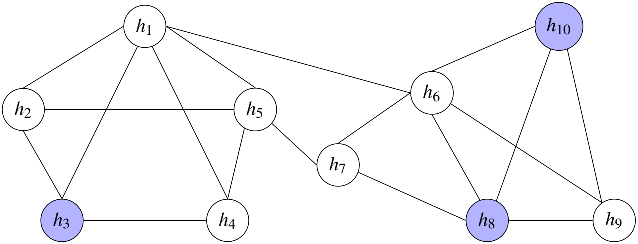


Forbid Constraint

- To dedicate hosts for system-level services (e.g., AC engine, Reference Monitor)
- Prevents VM instance v from being allocated on physical host h

$$\forall v \in \mathscr{V}, \ \ h \in \mathscr{H}, \quad (v,h) \in Forbid \implies p(v) \neq h$$

- $Forbid = \{(v,h_8),(v,h_3),(v,h_{10})\}$

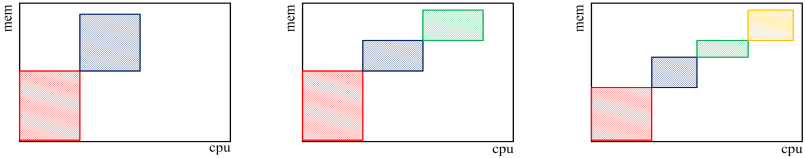


Count Constraint

- To avoid performance degradation due to co-hosted VM instances
- Limits the number of VM instances on a given host

$$\forall v \in \mathscr{V}, \quad h \in \mathscr{H}, \quad |\{v \in \mathscr{V} | p(v) = h\}| \leq count_h$$

- $Count_h \leq 3$

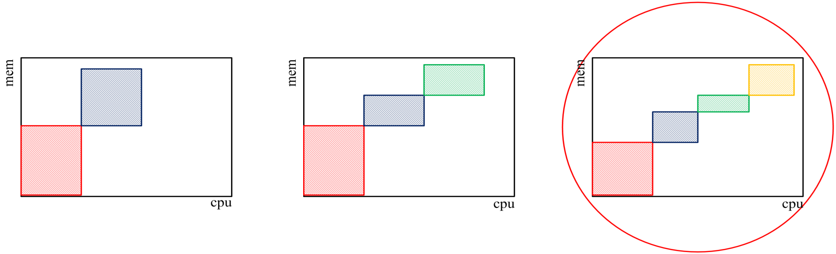


Count Constraint

- To avoid performance degradation due to co-hosted VM instances
- Limits the number of VM instances on a given host

$$\forall v \in \mathscr{V}, \quad h \in \mathscr{H}, \quad |\{v \in \mathscr{V} | p(v) = h\}| \leq count_h$$

- $Count_h \leq 3$

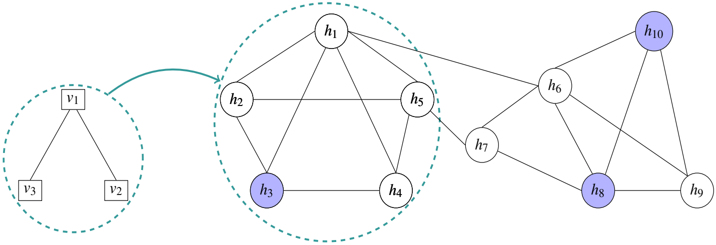


Restrict Constraint

- To support security and privacy policies and government enforced obligations
- Place VM instances only a specified group of physical hosts

$$\forall v_i \in \mathscr{V}, \quad H_j \in 2^{H_j}, \quad (v_i, H_j) \in Restr \implies p(v_i) \in H_j$$

- $Restr = \{(v_1, \{h_1, \ldots, h_5\}), (v_2, \{h_1, \ldots, h_5\}), v_3, \{h_1, \ldots, h_5\})\}$

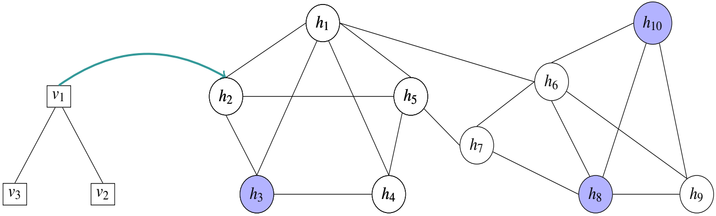


Distribute Constraint

- To avoid single points of failure among replicated applications
- Two VM instances are never located on the same physical host

$$\forall v_i, v_j \in \mathscr{V}, \;\; h \in \mathscr{H}, \quad (v_i, v_j) \in Distr \implies p(v_i) \neq p(v_j)$$

- $Distr = \{(v_1, v_2), (v_1, v_3)\}$

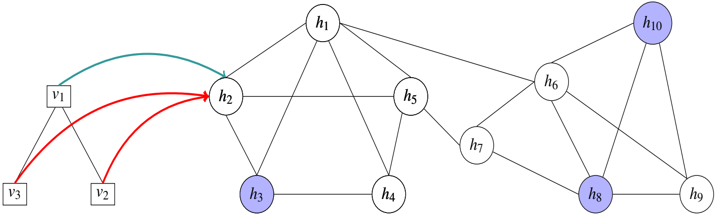


Distribute Constraint

- To avoid single points of failure among replicated applications
- Two VM instances are never located on the same physical host

$$\forall v_i, v_j \in \mathscr{V}, \;\; h \in \mathscr{H}, \quad (v_i, v_j) \in Distr \implies p(v_i) \neq p(v_j)$$

- $Distr = \{(v_1, v_2), (v_1, v_3)\}$

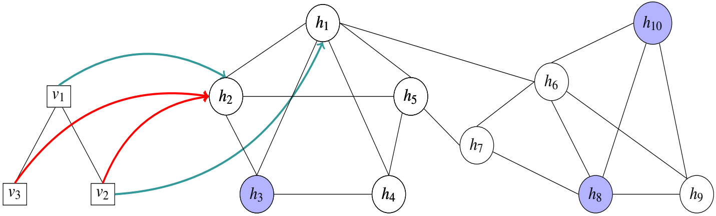


Distribute Constraint

- To avoid single points of failure among replicated applications
- Two VM instances are never located on the same physical host

$$\forall v_i, v_j \in \mathscr{V}, \; h \in \mathscr{H}, \quad (v_i, v_j) \in Distr \implies p(v_i) \neq p(v_j)$$

- $Distr = \{(v_1, v_2), (v_1, v_3)\}$

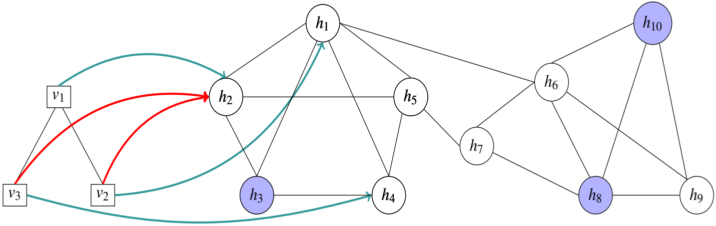


Distribute Constraint

- To avoid single points of failure among replicated applications
- Two VM instances are never located on the same physical host

$$\forall v_i, v_j \in \mathscr{V}, \;\; h \in \mathscr{H}, \quad (v_i, v_j) \in Distr \implies p(v_i) \neq p(v_j)$$

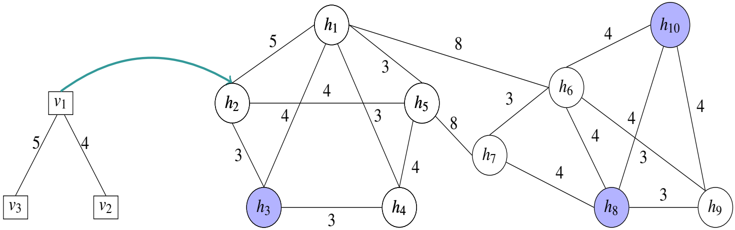
- $Distr = \{(v_1, v_2), (v_1, v_3)\}$



Latency Constraint

- To maintain performance of user's application
- Allocate VM instances such that network delay between them is less than a specified value

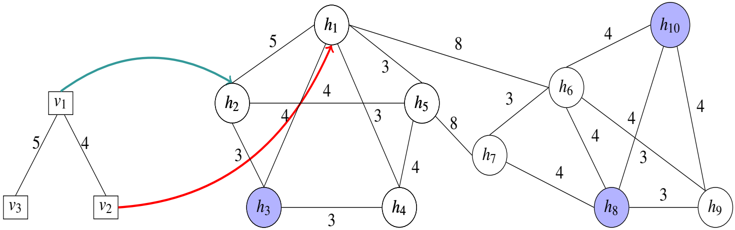
$$\forall v_i, v_j \in \mathcal{V} : \ (v_i, v_j, T_{max}) \in MaxLatency \implies latency(p(v_i), p(v_j)) \leq T_{max}$$



Latency Constraint

- To maintain performance of user's application
- Allocate VM instances such that network delay between them is less than a specified value

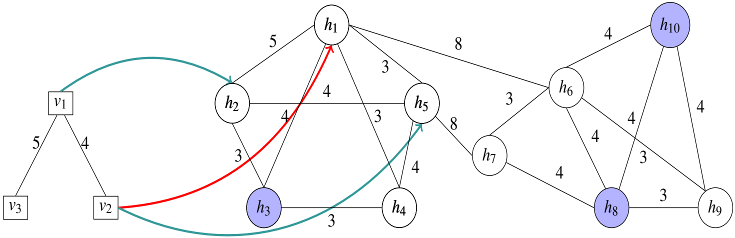
$$\forall v_i, v_j \in \mathscr{V}: \ (v_i, v_j, T_{max}) \in MaxLatency \implies latency(p(v_i), p(v_j)) \leq T_{max}$$



Latency Constraint

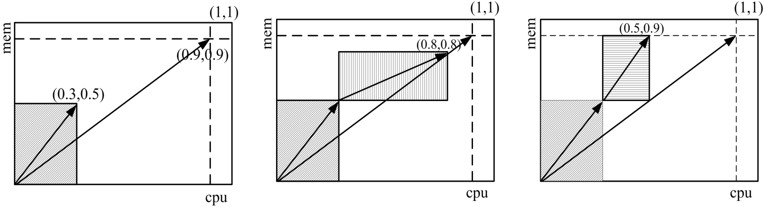
- To maintain performance of user's application
- Allocate VM instances such that network delay between them is less than a specified value

$$\forall v_i, v_j \in \mathscr{V} : \ (v_i, v_j, T_{max}) \in MaxLatency \implies latency(p(v_i), p(v_j)) \leq T_{max}$$



VM Provisioning (1)

- A two-stage, host-centric greedy heuristic
- Build a priority queue to select least-used cluster
- Consider hosts in the order of their identifiers
- Use vector dot-product method to allocate VM instances



- The vector dot-product values are 0.165 and 0.127 respectively

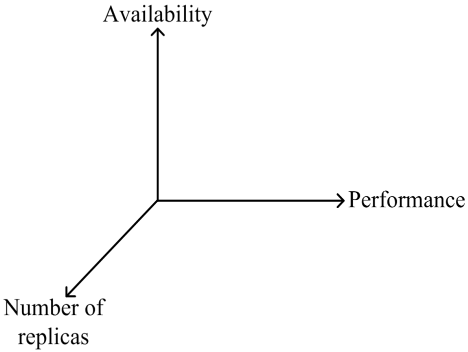
Fault Tolerance at Runtime (1)

- Fault tolerance policy of a mission may not be satisfied when the system's working status changes
- Static allocation schemes are computationally expensive

$\Longrightarrow$ Dynamically adapt the current allocation to the new working status of the Cloud by means of a heuristic

  - Online fault tolerance controller for the mission
  - Uses monitoring information (e.g., bandwidth availability, resource status) and virtualization technology constructs
  - Applies fewer actions to respond to the incidents (instead of computing an allocation from scratch)
  - Performs incremental allocation

Fault Tolerance at Runtime (2)



- High availability and performance are competing attributes

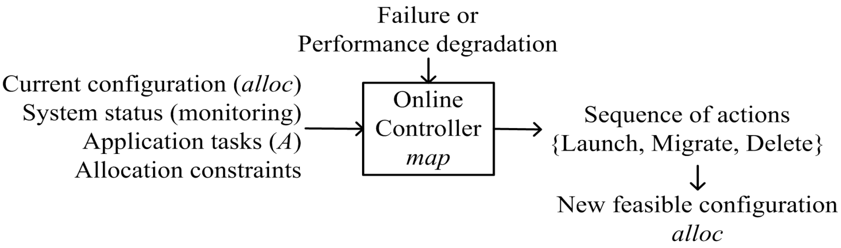
⟹ Balance availability and performance while generating a new configuration for a given mission

Adaptive Resource Management (1)

- Heuristics-based solution to minimize the performance and availability degradation of the mission due to the changes in the working status of the system
- Realized as an online fault tolerance controller that uses three activities to change the current allocation status of the mission
    - Launch(t,h): Create new replicas of a task – instantiate VM v, hosting task replica t, on the host h
    - Migrate(t,h_i,h_j): Change the current location of a task replica as a response to performance or availability degradation – move task t from host h_i to host h_j
    - Delete(t,h): Reduce the replication level of a task – remove task t from host h

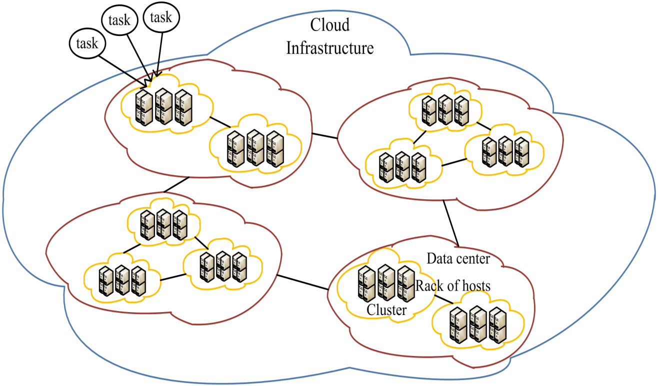
Adaptive Resource Management (2)

- A task can be allocated on host $h \in \mathscr{H}$ if the constraints (restriction, distribution, capacity and allowed latency) are satisfied
- Implement task allocation as the bin-packing problem (bins$\equiv$hosts and items$\equiv$VMs)
- The function $map : \mathscr{V} \to \mathscr{H}$ performs tentative search



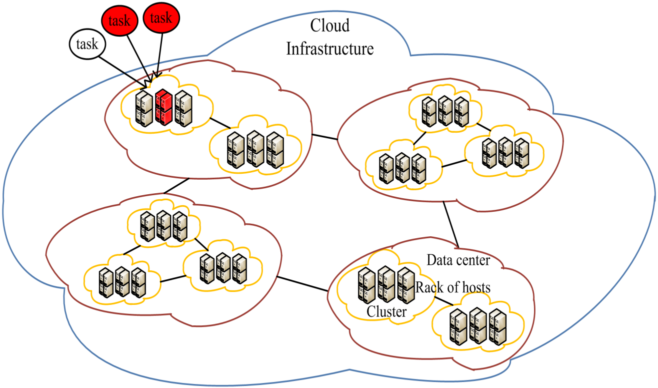
Adaptive Resource Management (3)

When **availability** of the mission is less than the desired one



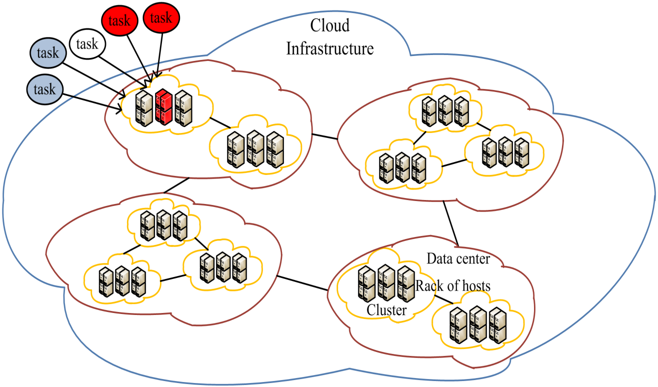
Adaptive Resource Management (3)

1 Identify task replica failures



Adaptive Resource Management (3)

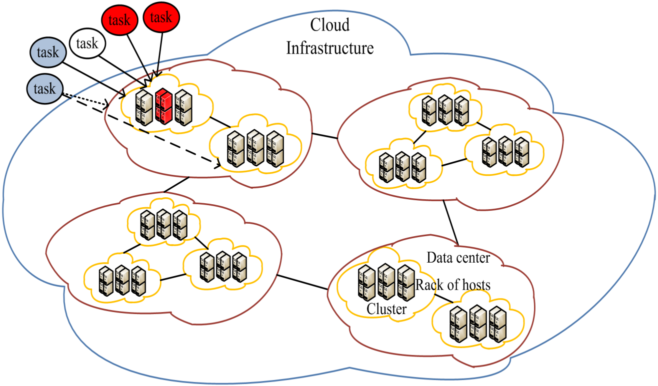
2 Launch new replicas at the same deployment level until current replication level is equal to original replication level and performance goals are satisfied



Adaptive Resource Management (3)

3 If availability goals are still not satisfied

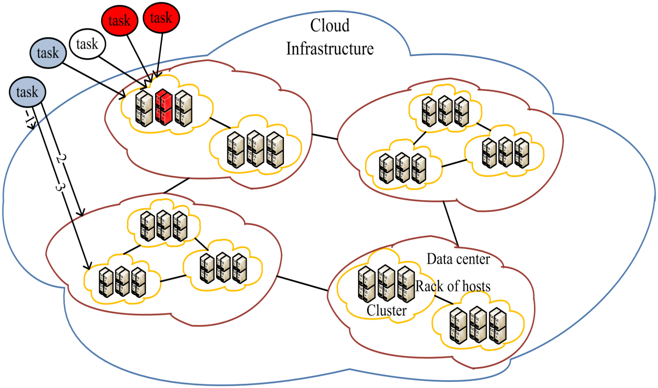
  3.1 Move the task replicas to higher deployment levels



Adaptive Resource Management (3)

3 If availability goals are still not satisfied

  3.2 If the performance conditions conflict, starting from higher deployment levels, move gradually to lower levels and launch replicas where availability and performance goals are fulfilled



Adaptive Resource Management (4)

When **performance** of the mission is less than the desired one

- Identify the tasks with affected response time
- Delete task replicas in the same deployment level without violating availability goals
- If expected performance is still lower than the desired one
  - Move task replicas to lower deployment level
  - If availability conditions conflict, traverse from lowest deployment level, move gradually to higher levels and decrease replication level until availability and performance goals are fulfilled

Conclusions

- Mission-centric techniques to improve the security and fault tolerance in Cloud computing
- Secure mission deployment techniques (allocation and scheduling)
- Static and dynamic mission protection by network hardening
- Provide complementary fault tolerance support to the mission as a service

Publications

Chapters in Books

- R. Jhawar, V. Piuri, “Dependability-oriented Resource Management Schemes For Cloud Computing Data Centers,” in Handbook on Data Centers, S.U. Khan, A.Y. Zomaya (eds.), Springer, 2015 (to appear)
- M. Albanese, S. Jajodia, R. Jhawar, V. Piuri, “Securing Mission-Centric Operations in the Cloud,” in Secure Cloud Computing, S. Jajodia, K. Kant, P. Samarati, V. Swarup, C. Wang (eds.), Springer, pp. 239-260, 2014

International Journals Articles

- R. Jhawar, V. Piuri, M. Santambrogio, “Fault Tolerance Management in Cloud Computing: A System-Level Perspective,” in IEEE Systems Journal, pp.288-297, June, 2013
- C. A. Ardagna, R. Jhawar, V. Piuri, “Dependability Certification of Services: A Model-Based Approach,” in Springer Computing Journal, pp.1-28, October 2013

Publications

International Conferences and Workshops

- R. Jhawar and V. Piuri, “Adaptive Resource Management for Balancing Availability and Performance in Cloud Computing,” in Proc. of 10th Int’l Conference on Security and Cryptography, Reykjavik, Iceland, July 29-31, 2013
- M. Albanese, S. Jajodia, R. Jhawar, V. Piuri, “Secure Mission Deployment in Vulnerable Networks,” IEEE Workshop on Reliability and Security Data Analysis, Budapest, Hungary, June 24-27, 2013
- R. Jhawar, V. Piuri, and P. Samarati, “Supporting Security Requirements for Resource Management in Cloud Computing,” in Proc. of the 15th IEEE Int’l Conference on Computational Science and Engineering, Paphos, Cyprus, December 5-7, 2012
- R. Jhawar, and V. Piuri, “Fault Tolerance Management in IaaS Clouds,” in Proc. of the 1st IEEE-AESS Conference in Europe about Space and Satellite Telecommunications, Rome, Italy, October 2-5, 2012

Publications

- C.A. Ardagna, E. Damiani, R. Jhawar, and V. Piuri, “A Model-Based Approach to Reliability Certification of Services,” in Proc. of the 6th IEEE Int’l Conference on Digital Ecosystem Technologies - Complex Environment Engineering, Campione d’Italia, Italy, June, 2012
- R. Jhawar, V. Piuri, and M. Santambrogio, “A Comprehensive Conceptual System-Level Approach to Fault Tolerance in Cloud Computing,” in 2012 IEEE Int’l Systems Conference, Vancouver, BC, Canada, March 19-22, 2012