Constraints Specification for Virtual Resource Orchestration in Cloud IaaS

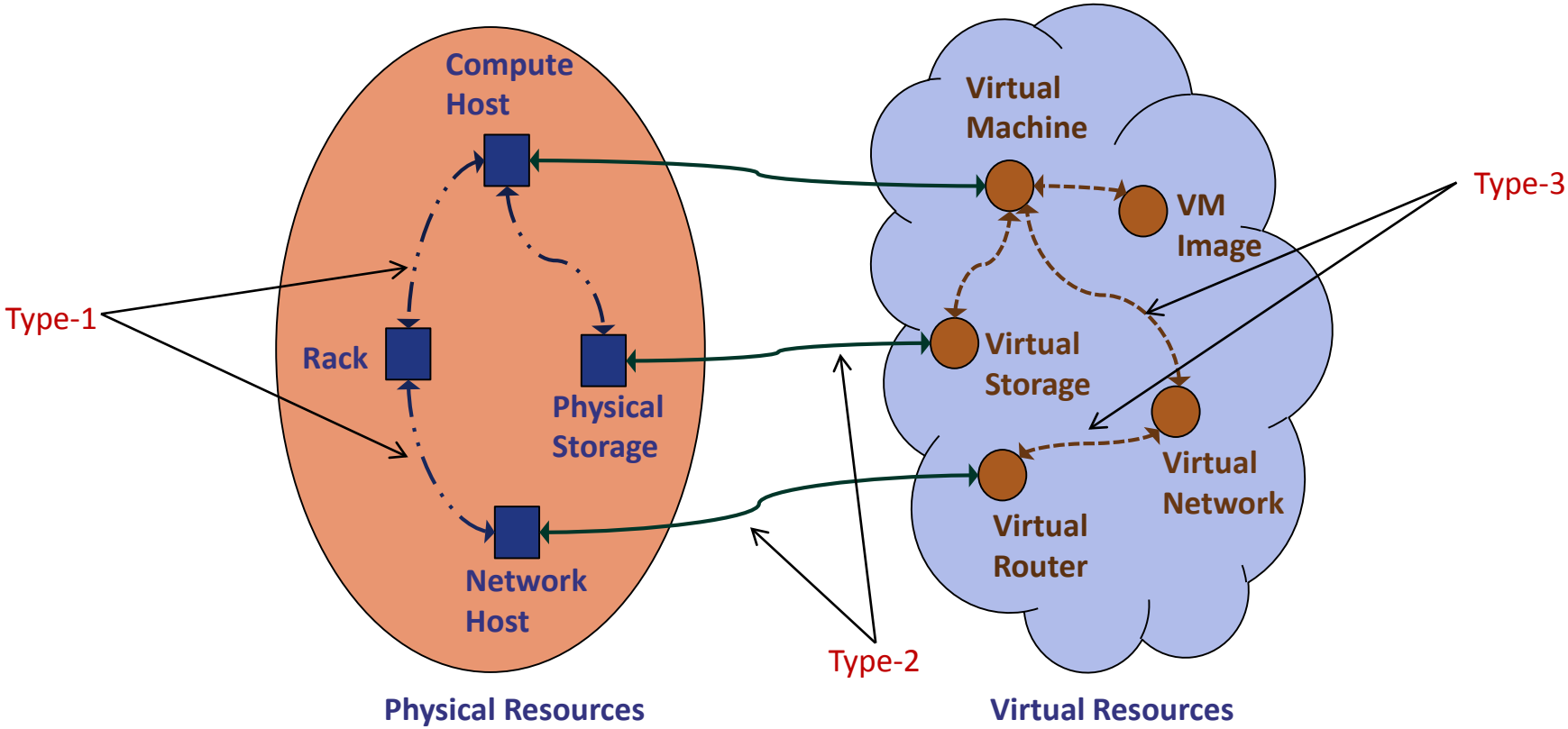
Khalid Bijon, Ram Krishnan and Ravi Sandhu
University of Texas at San Antonio

5th ACM Conference on Data and Applications Security and Privacy (CODASPY 2015)

Outline

- Introduction
- Motivation
- Goal
- Methodology
- Enforcement (in Cloud IaaS)
- Implementation (in OpenStack)
- Conclusion

Introduction



- Three Different Mapping Types
- Shared Responsibility
- Only Consider Type-3 Mappings
- Complex Management Process

Motivation

- **Inefficient and Tedious Management Plane**
  - Manual Identification
  - User Centric (unnecessary indirection)

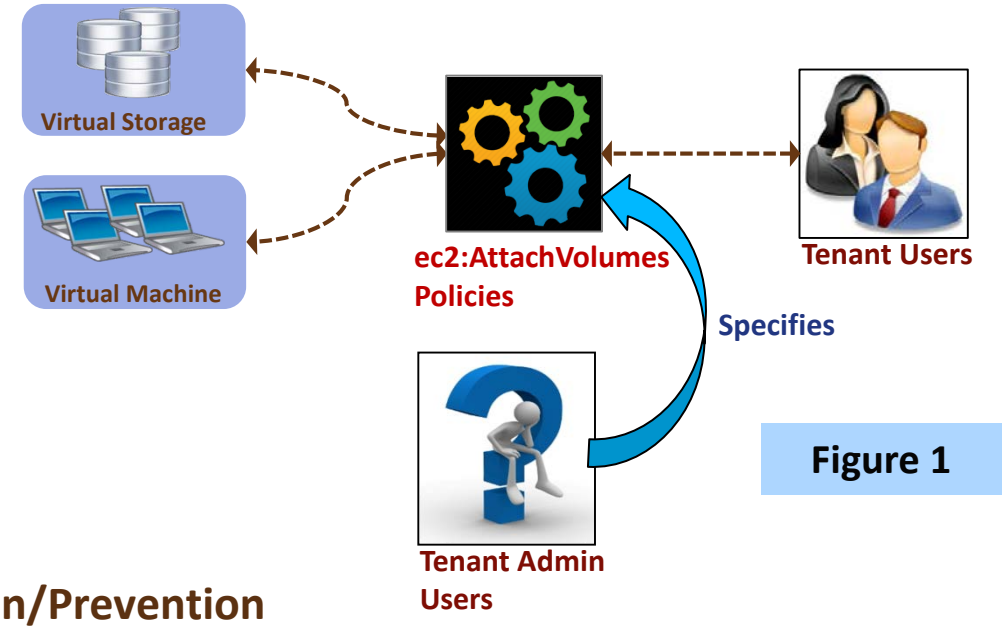


Figure 1

- **No Direct Misconfiguration Detection/Prevention**
  - Elevate Security Vulnerability

Manual Detection

Credit: www.iconarchive.com
www.consulting.ky
www.acm.icpc.org

Goal

- **Easily Manageable Type 3 Mapping**
  - High-level Policy
  - Configure Diverse Requirements

Tenants

- **Establish Direct Relations**
  - Keep Users Out of Loop

- **Automatically Prevent Misconfiguration**

Credit: www.bartley.hants.sch.uk

Methodology

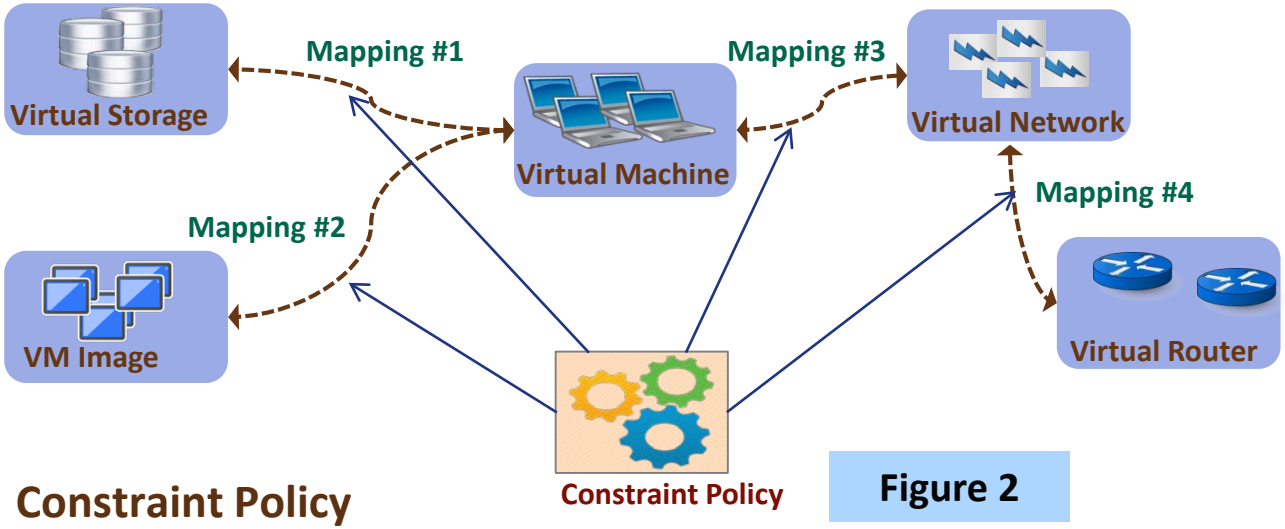


Figure 2

- **Constraint Policy**
  - For Each Type-3 Mappings

- **Satisfied By**
  - Individual Virtual Resources

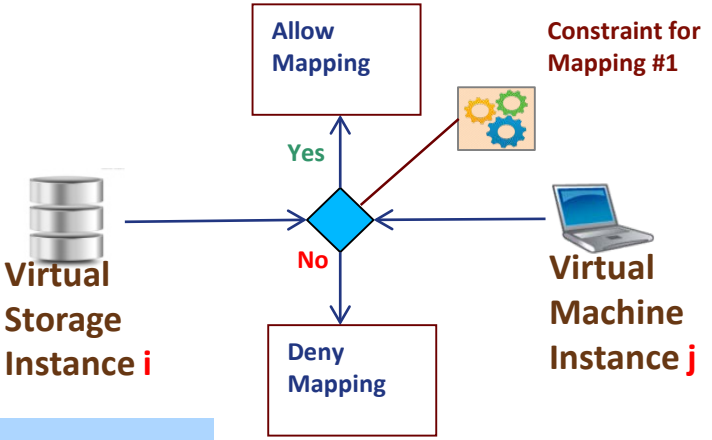


Figure 3

Credit: www.iconarchive.com

An Attribute Based Approach

- **Attribute Specifies Virtual Resource Properties**

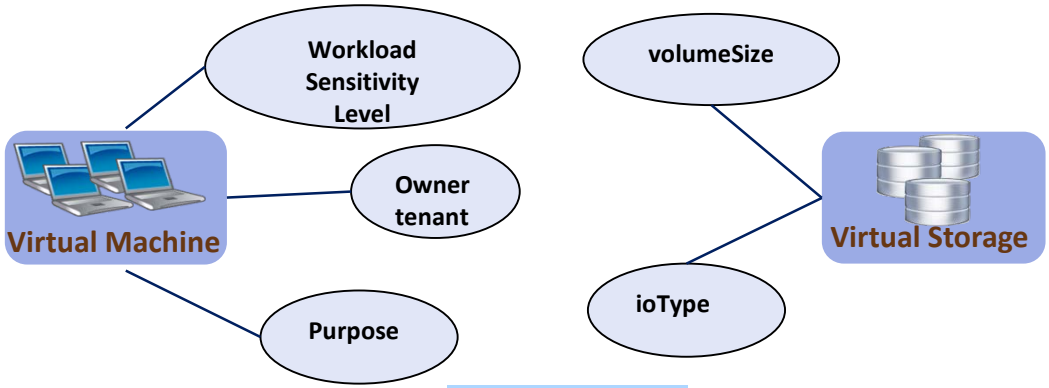


Figure 4

- **A name:*value* Pair**

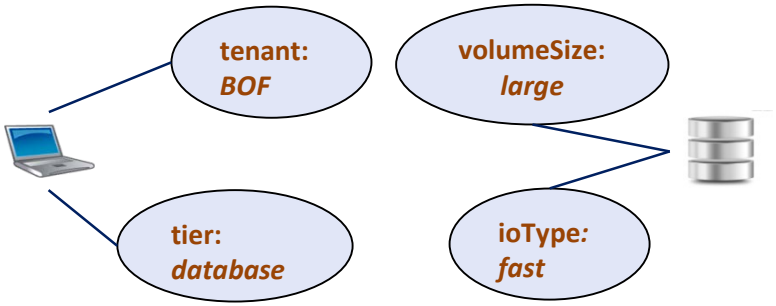


Figure 5

- **Designed as Functions**

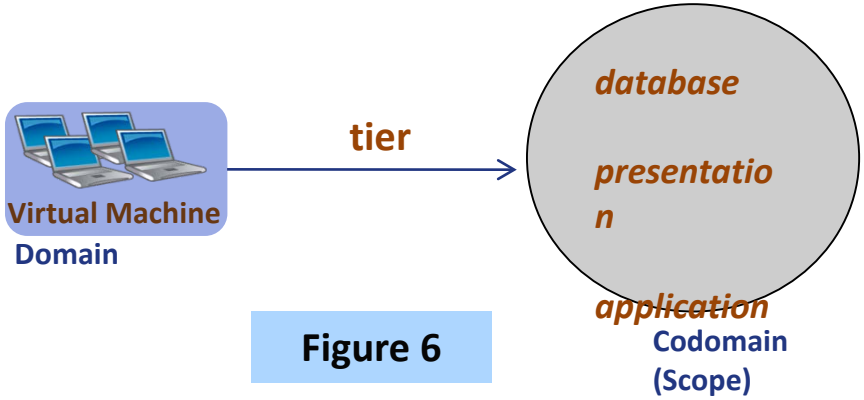
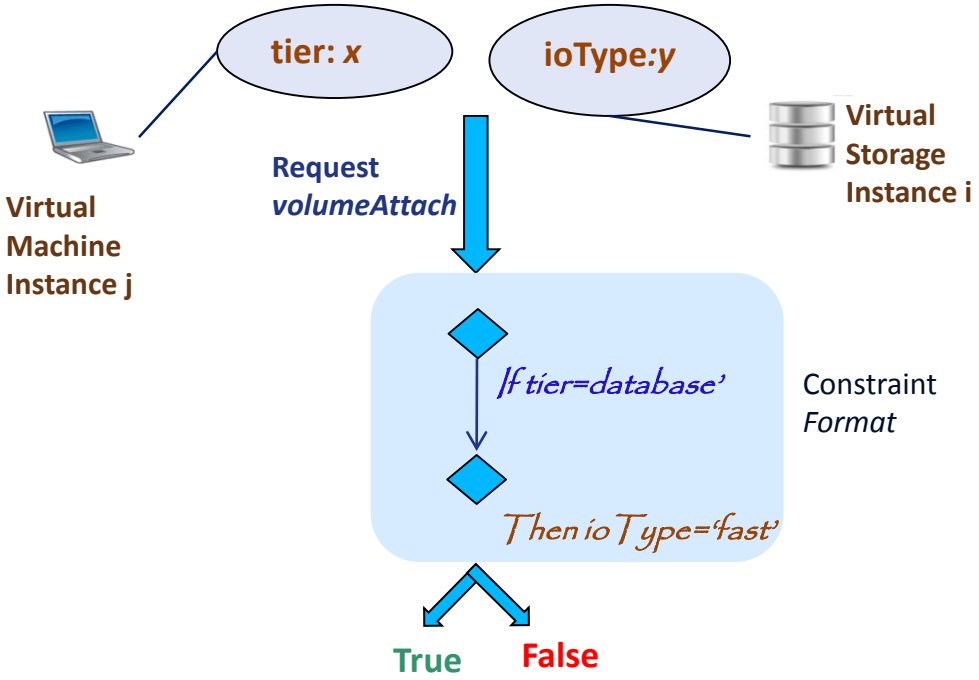


Figure 6

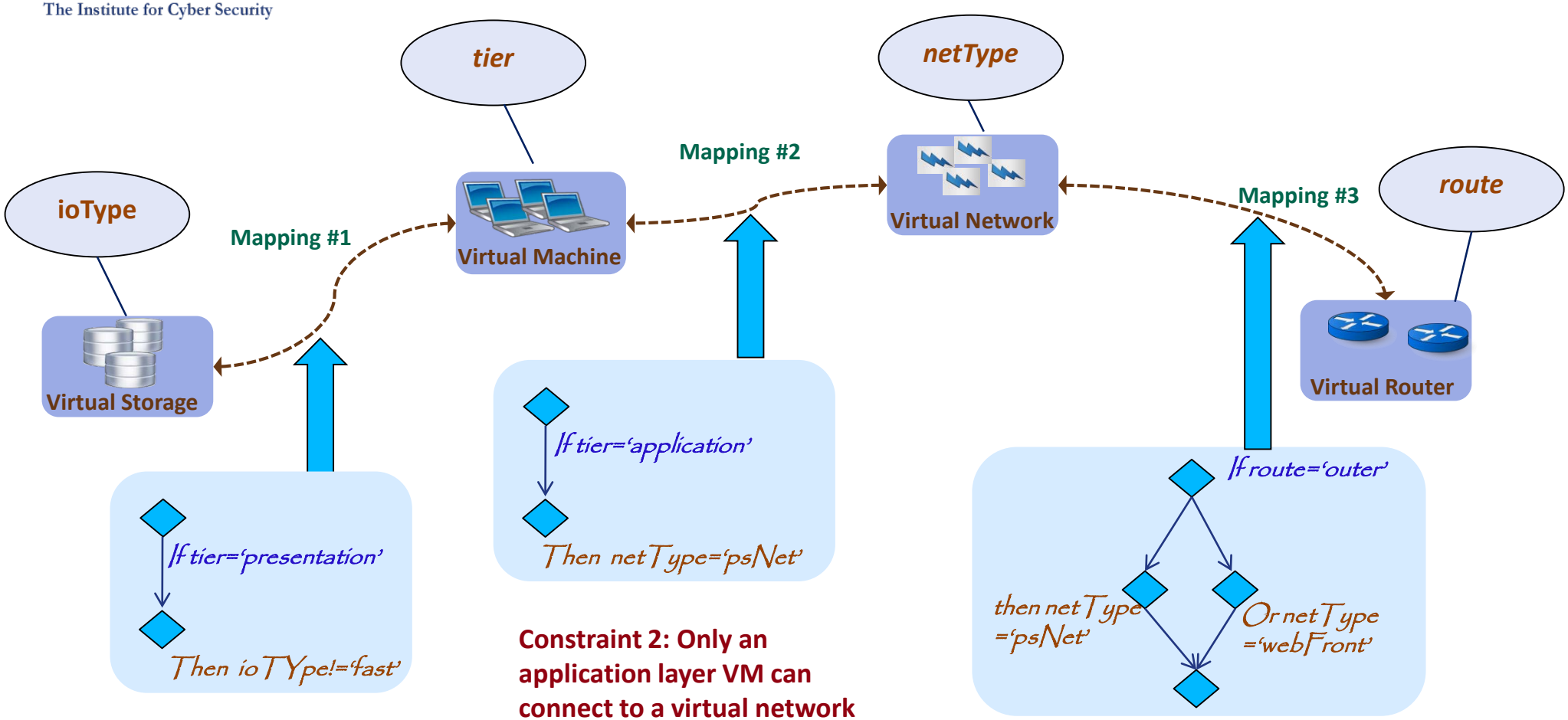
Constraint Policy

- **A Constraint**
  - Logical Formula
  - Compares Certain Attribute Values

- **Simple but Powerful**
  - Hadoop Cluster
  - 3-tier business application



UseCase (3-Tier System)



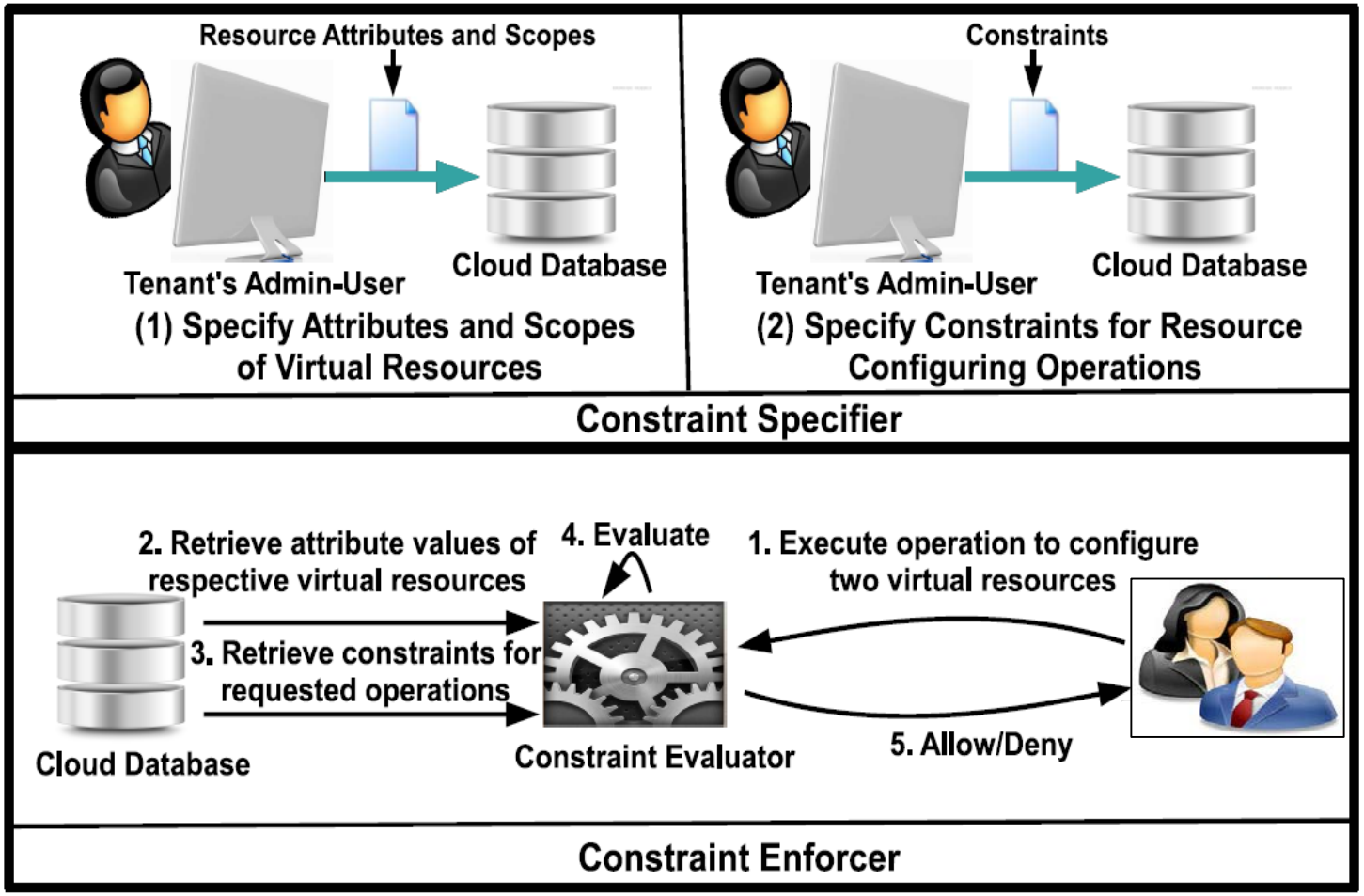
Constraint 1: If a VM is for presentation layer, attaching storage's ioType cannot be fast.

Constraint 2: Only an application layer VM can connect to a virtual network which is created for passing application layer data.

Constraint 3: If a router is for connecting to out-side internet, only presentation layer network or web-fornt network can connect to it.

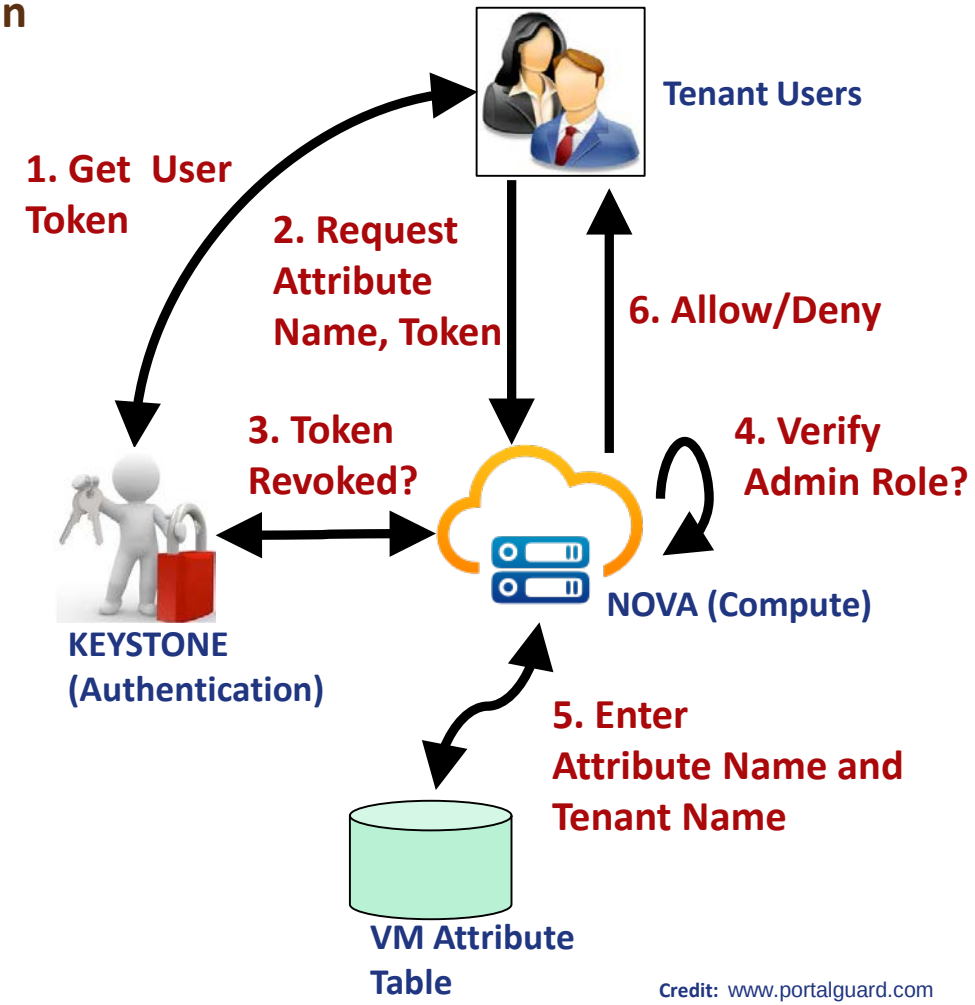
Enforcement

- **Two Components**
  - Specifier and Enforcer



Specifier

- Implemented in OpenStack
- Execution of "*attribute-creation*" operation
- Similarly,
  - Attribute-value specification
  - Constraint Specification
  - Attribute-value assignment



Credit: www.portalguard.com

Specifier (cont.)

- **API Specification**
  - **Rest API**

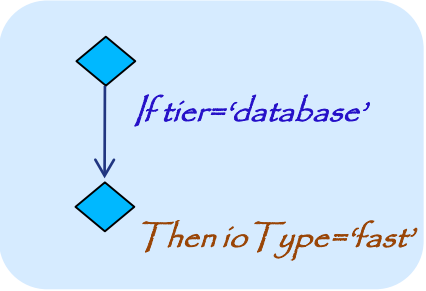
Attribute Name

Attribute Value

Attribute Value Assignment

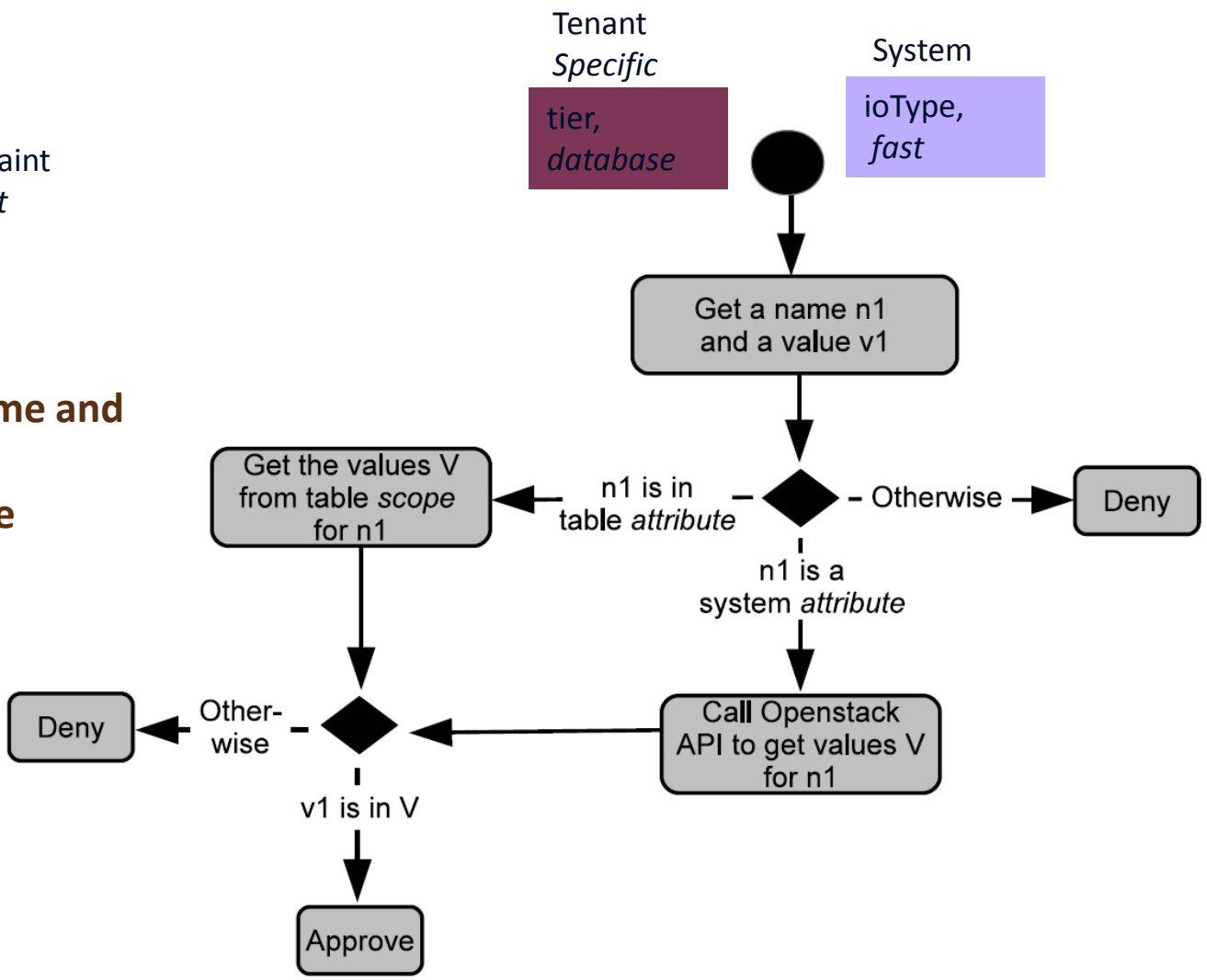
Name	URL	Type
att-create	*/v2/{tenant_id}/attributes*	POST
att-delete	*/v2/{tenant_id}/attributes/{id}*	DELETE
att-list	*/v2/{tenant_id}/attributes*	GET
att-value-set	*/v2/{tenant_id}/scopes*	POST
att-value-delete	*/v2/{tenant_id}/scopes/{id}*	DELETE
att-value-get	*/v2/{tenant_id}/scopes/*	GET
constraint-add	*/v2/{tenant_id}/constriants*	POST
constraint-delete	*/v2/{tenant_id}/constraints/{id}*	DELETE
constraint-get	*/v2/{tenant_id}/constraints*	GET
meta	*/v2/{tenant_id}/servers/{resource_id}/metadata*	POST

Validation Check



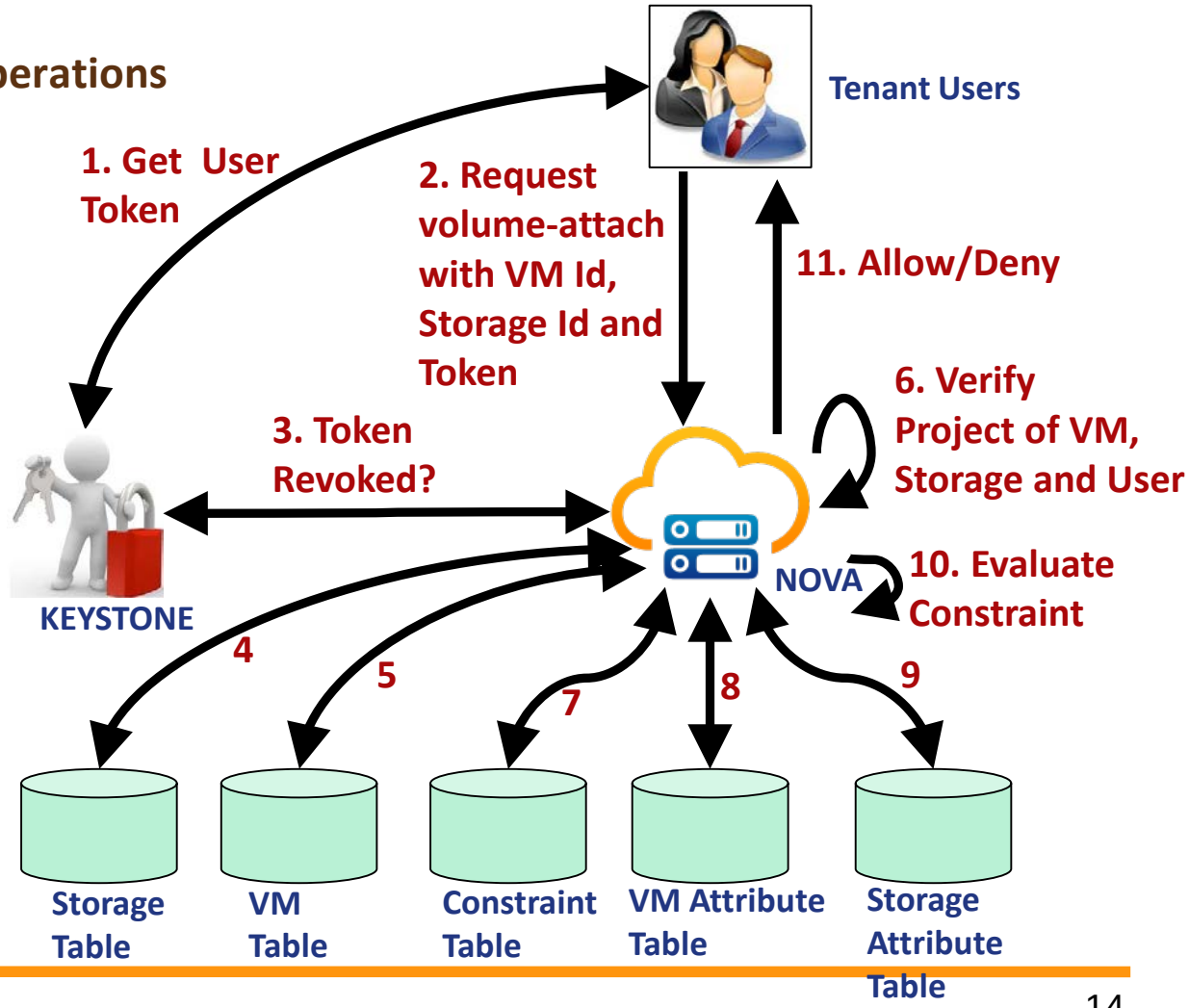
Constraint
Format

- **Validity of Attribute Name and Value**
- **Tenant-Specific Attribute**
- **System or Inter-Tenant Attribute**



Enforcer Implementation

- Implemented in OpenStack
- A Constraint Parser
- Invoked by Resource Mapping Operations (e.g., *volume-attach*)



Automated Constraint Construction

- Helps the tenants to find policy

- From Previous Configurations

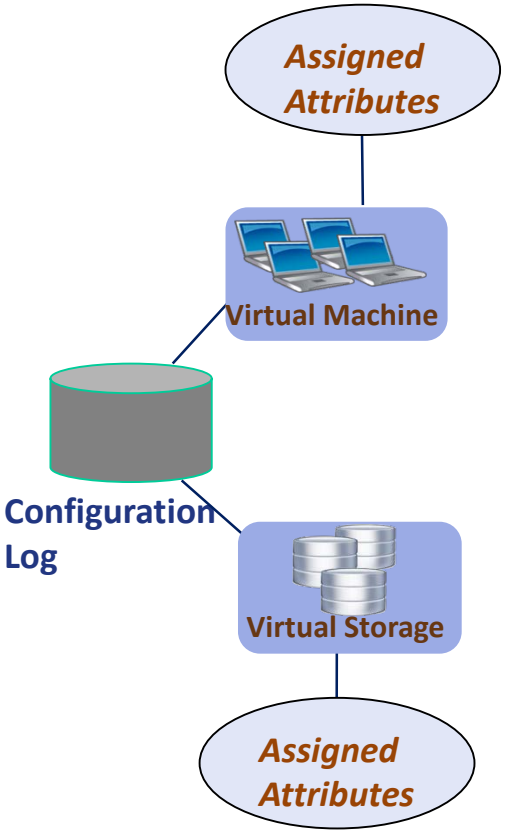


Figure 7

- Construct Relation between values of two attributes

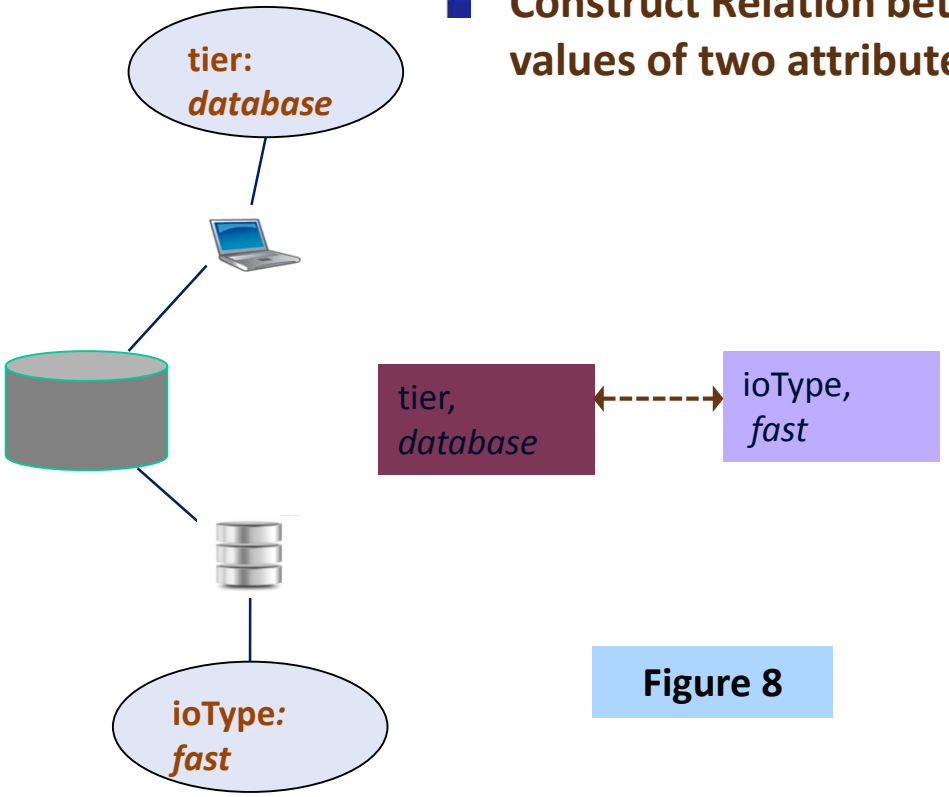
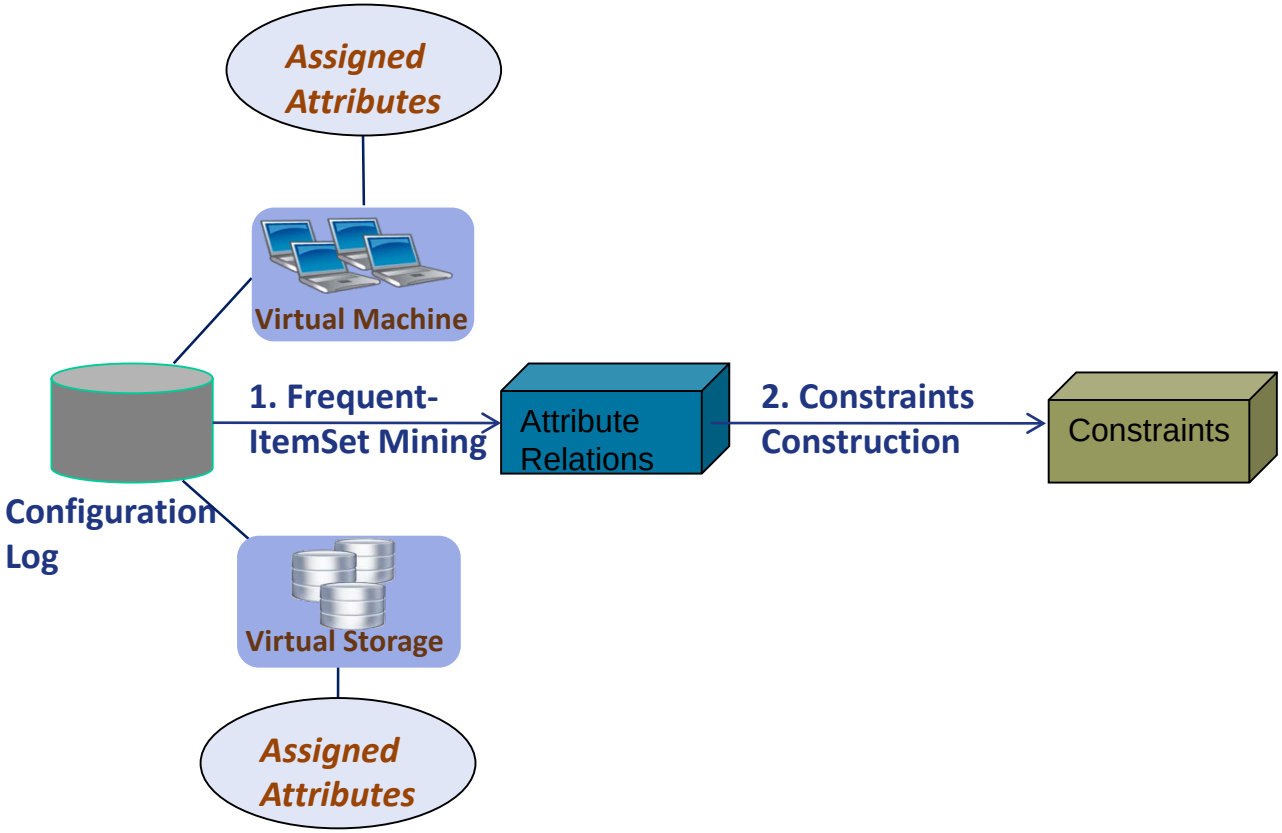


Figure 8

Credit: www.iconarchive.com

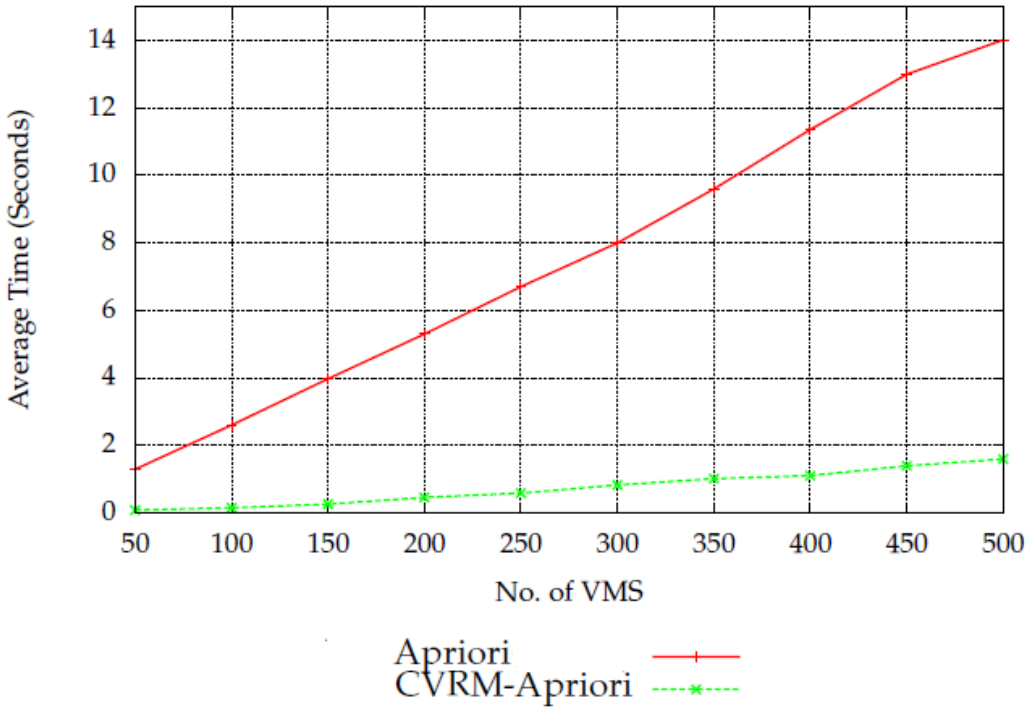
Approach

- **Frequent-ItemSet Mining**
  - **Apriori Algorithm**
  - **with customization for IaaS (CVRM-Apriori)**



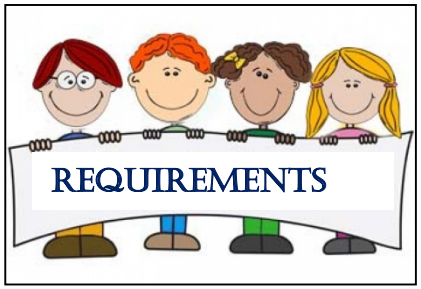
Evaluation

- Policy for VM-Network Connectivity Mapping
- From VM-Network Table (table *virtual_interfaces* in Nova, OpenStack)
- 10 Attributes each with 10 values
- 10 Virtual Networks
- At least three Networks per VM
- Mine relations between every two pair of attribute values



Conclusion

- A Constraint Specification Framework
- Easily manageable and generic



Tenants

- Can be applied for Misconfiguration Prevention
- Also, for detection (flag-generator)
- Automatic Generation of Constraints

- Flag Generator System
- Semantic meaning of mined Attribute Relation
- Improve mining (incorporate noise)

Credit: www.iconarchive.com

Language

$<\text{Quantifier}> := \forall(\text{vr1},\text{vr2}) \in \mathcal{R}_{<\text{Cls}>,<\text{Cls}>} \,.\, <\text{Stmt}>$

$<\text{Stmt}> := <\text{Stmt}> <\text{connector}><\text{Stmt}> \mid (<\text{rule}>)$

$<\text{rule}> := <\text{Token}> \rightarrow <\text{Token}>$

$<\text{Token}> := (<\text{Token}> <\text{connector}> <\text{Token}>)|(<\text{Term}>)$

$<\text{Term}> := <\text{Attribute}>(<\text{resource}>) <\text{comperator}> <\text{Scope}>$

$<\text{Attribute}> ::= <\text{letter}> \mid <\text{digit}> \mid <\text{Attribute}>$

$<\text{Scope}> ::= <\text{letter}> \mid <\text{digit}> <\text{Scope}>$

$<\text{connector}> ::= \wedge \mid \vee$

$<\text{comparator}> ::= = \mid \neq$

$<\text{Cls}> ::= \mathsf{c}_1 \mid \mathsf{c}_2 \mid \ldots \mid \mathsf{c}_n$

$<\text{resoruce}> ::= \text{vr1} \mid \text{vr2}$

$<\text{digit}> ::= 0|1|2| \ldots |8|9$

$<\text{letter}> ::= \text{a}|\text{b}|\ldots|\text{y}|\text{z}|\text{A}|\text{B}|\ldots|\text{Y}|\text{Z}$

Backup

Field	Type
id	Integer
name	String(255)
project_id	String(255)

(a) The *attribute* Table

Field	Type
id	Integer
relation_name	String(255)
expression	String(255)
project_id	String(255)

(c) The *constraints* Table

Field	Type
id	Integer
name	String(255)
value	String(255)
project_id	String(255)

(a) The *scope* Table

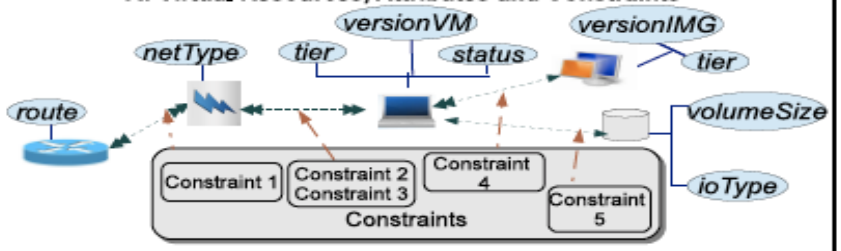
Field	Type
vm_id	Integer
vm_id	String(255)
meta	String(255)

(d) The instance-*metadata* Table

Backup

3-Tier Application System Configurations

A: Virtual Resources, Attributes and Constraints



B: Scopes of the Attributes

$$ATTR^{VM}_{3\text{-tier}} = \{ tier, versionVM, status \}$$

$$SCOPE_{tier} = \{presentation, application, database\}$$

$$SCOPE_{versionVM} = \{psV1, psV2, app1, dbV1, dbV2\}$$

$$SCOPE_{status} = \{running, stop, suspended\}$$

$$ATTR^{IMG}_{3\text{-tier}} = \{ tier, versionIMG \}$$

$$SCOPE_{tier} = \{presentation, application, database\}$$

$$SCOPE_{versionIMG} = \{psI1, psI2, appI1, dbI1\}$$

$$ATTR^{NET}_{3\text{-tier}} = \{ netType \}$$

$$SCOPE_{netType} = \{outerNet, psNet, appNet, dbNet\}$$

$$ATTR^{RTR}_{3\text{-tier}} = \{ route \}$$

$$SCOPE_{route} = \{outerR, psToappR, appTodbR\}$$

$$ATTR^{STR}_{3\text{-tier}} = \{ ioType, volumeSize \}$$

$$SCOPE_{ioType} = \{regular, fast, fastest\}$$

$$SCOPE_{volumeSize} = \{regular, large, huge\}$$

C: Constraint Specification

Constraints for router-network connection mapping:

Const 1: *If route attribute of a router is outerRoute then only network with netType outerNet can connect to it.*

$$((route(\text{router})=\text{outerR}) \rightarrow ((netType(\text{network})=\text{outerNet}) \vee (netType(\text{network})=\text{psNet})))$$

Constraints for network-vm connection mapping:

Const 2: *Only presentation layer vm can connect to a psNet network. Similar, constraints can be generated for other layer vms and networks.*

$$((netType(\text{network})=\text{psNet}) \rightarrow (tier(\text{vm})=\text{presentation}))$$

Const 3: *A network can be removed from a vm if the vm status is stop.*

$$((status(\text{vm})=\text{stop}))$$

Constraints for image-vm connection mapping:

Const 4: *If tier of IMG is presentation and versionImg is psI1, it cannot be used by presentation VM with versionVM psV2.*

$$(((tier(\text{image})=\text{presentation}) \wedge (versionImg(\text{image})=\text{psI1})) \rightarrow (((tier(\text{vm})=\text{presentation}) \wedge (versionVM(\text{vm})\neq\text{psV2})) \vee (tier(\text{vm})\neq\text{presentation})))$$

Constraints for storage-vm connection mapping:

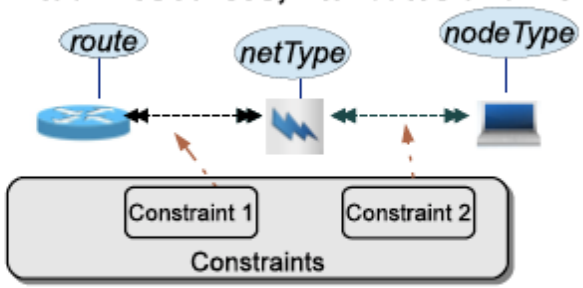
Const 5: *A presentation vm cannot get fastest ioType storages.*

$$((tier(\text{vm})=\text{presentation}) \rightarrow (ioType(\text{storage})\neq\text{fastest}))$$

Backup

Hadoop Cluster Configurations

A: Virtual Resources, Attributes and Constraints



B: Scopes of the Attributes

$\mathbf{ATTR}^{\mathbf{VM}}_{\mathbf{hadoop}} = \{ nodeType \}$

$\mathbf{SCOPE}_{\mathbf{nodeType}} = \{ \text{clientNode}, \text{nameNode}, \text{jobTracker}, \text{mapTask}, \text{reduceTask} \}$

$\mathbf{ATTR}^{\mathbf{NET}}_{\mathbf{hadoop}} = \{ netType \}$

$\mathbf{SCOPE}_{\mathbf{netType}} = \{ \text{outerNet}, \text{clientNet}, \text{nameNet}, \text{jobNet}, \text{mapNet}, \text{reduceNet} \}$

$\mathbf{ATTR}^{\mathbf{RT}}_{\mathbf{hadoop}} = \{ route \}$

$\mathbf{SCOPE}_{\mathbf{route}} = \{ \text{outerRoute}, \text{nameRoute}, \text{jobRoute}, \text{taskRoute} \}$

C: Constraint Specification

Constraints for router-network connection mapping:

Constraint 1: *If route attribute of a router is outerRoute then only network with netType outerNet and clientNet can connect to it and if If route attribute is taskRoute then it cannot be connected with nameNet, outerNet and clientNet.*

$$((route(\text{router})=\text{outerRoute}) \rightarrow ((netType(\text{network})=\text{outerNet}) \vee (netType(\text{network})=\text{clientNet}))) \wedge ((route(\text{router})=\text{taskRoute}) \rightarrow (((netType(\text{network}) \neq \text{nameNet}) \wedge (netType(\text{network})=\text{outerNet})) \wedge (netType(\text{network})=\text{clientNet})))$$

Constraints for network-vm connection mapping:

Constraint 2: *In a nameNet network only nameNode and jobTracker vm can be connected.*

$$((netType(\text{network})=\text{nameNet}) \rightarrow ((nodeType(\text{vm})=\text{nameNode}) \vee (nodeType(\text{vm})=\text{jobTracker})))$$