Mitigating Multi-Tenancy Risks in IaaS Cloud Through Constraints-Driven Virtual Resource Scheduling

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

ACM Symposium on Access Control Models and Technologies (SACMAT 2015)
Vienna, Austria
June 1-3, 2015
Cloud Service Models

- **Software as a Service (SaaS)**
  - Network accessible software

- **Platform as a Service (PaaS)**
  - App dev environment with cloud characteristics

- **Infrastructure as a Service (IaaS)**
  - Virtualized hardware infrastructure
IaaS Cloud: Virtual to Physical Mappings
Multi-tenancy is unavoidable in cloud platforms
- Hypervisor provides isolation, albeit tricky
  - E.g. Ristenpart et al
Constraints-Driven Co-location

• Toward a programmable cloud platform for resource isolation that can satisfy constraints such as:
  – “Do not co-locate sensitive VMs with low-sensitive”
  – “Do not co-locate high-availability VMs in the same rack”
  – “Do not co-locate Exxon VMs with those of BP”

• Must not co-locate vs. must co-locate
  – Scheduling problems
Attribute-Based Conflict Specification for VM Co-location

• Name-value pairs on VMs
  – E.g. sensitivity(vm₁)="high", tenant(vm₂)="Acme"
  – Specified for VMs of each tenant

• Intra-tenant (tenant-specified)
  – Varies from tenant to tenant
  – E.g. “sensitivity”, “group”, etc.

• Inter-tenant (cloud service provider specified)
  – Available to VMs of all tenants
  – E.g. “tenant”, “flavor”, etc.
Sample Attributes for a Tenant

\[ \text{ATTR}_{\text{VM}} = \{ \text{sensitivity, tenant} \} \]
\[ \text{SCOPE}_{\text{sensitivity}} = \{ \text{high, low} \} \]
\[ \text{SCOPE}_{\text{tenant}} = \{ \text{tnt1, tnt2, tnt3, tnt4, tnt5, tnt6} \} \]
\[ \text{ConSet}_{\text{sensitivity}} = \{ \{ \text{high, low} \} \} \]
\[ \text{ConSet}_{\text{tenant}} = \{ \{ \text{tnt1, tnt2} \}, \{ \text{tnt4, tnt6} \}, \{ \text{tnt2, tnt3} \} \} \]
Conflict-Free Partitioning of Attributes

Step 1:

Finding MIN_PARTITION is similar to k-coloring: NP-Complete

Step 2:

\[ O(|ATTR_{VM}| \times |PARTITION_{att}|) \]
Co-Resident VM Scheduling

Step 3: Partitions of co-resident VMs

\[ O(|\text{VM}| \times |\text{ConflictFreeATTR}| \times |\text{ATTR}_{\text{VM}}|) \]

Step 4: Scheduling of co-resident VMs into physical hosts

Similar to bin-packing: NP-Hard

Not a problem introduced by this work
Experimental Setup

• OpenStack deployed on 5 physical machines
  – Each is a Dell R710 with 16 cores, 2.53 GHz and 98GB RAM
  – Each VM simulated as a physical host to simulate 100s of physical hosts
Conflict-Free Partition Using Backtracking

Small-ish scope and conflict set

Large scope and conflict set
Scheduling Latency After Partitioning

![Graph showing scheduling latency over the number of VMs]

- X-axis: Number of VMs
- Y-axis: Time (sec)
- Graph title: Conflict-Free VM Scheduler

UTSA Engineering

The Institute for Cyber Security
#Hosts

With varying number of elements in Conflict-Set

With varying number of maximum degree of conflicts
Host Utilization

![Host Utilization Graph]

- Server Utilization (Percentage)
- Number of Vms

- Without Conflicts
- Max Degree-of-Conflicts 5
- Max Degree-of-Conflicts 10
- Max Degree-of-Conflicts 15
- Max Degree-of-Conflicts 20

UTSA Engineering
Conflict Changes

• Conflict specification can change over time!
• Changes can be of different types
  – Type 1: remove an element from the ConSet_{att}
  – Type 2: add an element to ConSet_{att}
    • PARTITION_{att} remains unchanged
  – Type 3: add an element to ConSet_{att}
    • PARTITION_{att} changes -> may need to migrate
Migrations

% of Conflict for a Given Scope

% of Total VMs that Require Migration
Ongoing/Future Directions

• Constraints that span further levels of abstractions
  ○ PaaS and SaaS
Ongoing/Future Directions (continued)

• Constraints involving other virtual resources
  – Storage, Network, etc.
• Managing conflict changes over time
• Incremental conflict specification
• Attribute computation to inform conflict specification
Summary

• A conflict specification framework for resources in IaaS
  – Conflict-free partitioning is NP-Complete

• Prototyped and experimented in OpenStack
Thank you!