Elastic Secure Marketplace
For
Trading Bare-metal Servers

MACS 2018

Sahil Tikale
(PhD Candidate)
ECE, Boston University

Amin Mosayyebzadeh
(PhD Student)
ECE, Boston University

MACS (12/07/2018)
Public Cloud
Public Cloud

Private cloud
Hybrid Cloud

Public Cloud

Private cloud
CAN WE DO BETTER THAN THIS?
Share excess capacity with others
Common shared pool

Bare Metal Servers
HPC/HTC Cluster

- Unlimited CPU demand.
- Aggregated CPU usage per month
- Happy to share if monthly CPU usage > HPC owned CPUtime

Common shared pool

Bare Metal Servers
HPC/HTC Cluster

- Unlimited CPU demand.
- Aggregated CPU usage per month
- Happy to share if monthly CPU usage > HPC owned CPUtime

Common shared pool

Bare Metal Servers
- Unlimited CPU demand.
- Aggregated CPU usage per month
- Happy to share if monthly CPU usage > HPC owned CPUtil

Interactive demand: Short term peaks.
- Let other use than running idle
HPC/HTC Cluster

- Unlimited CPU demand.
- Aggregated CPU usage per month
- Happy to share if monthly CPU usage > HPC owned CPU time

OpenStack Cluster

- Interactive demand: Short term peaks.
- Let other use than running idle
HPC/HTC Cluster

- Unlimited CPU demand.
- Aggregated CPU usage per month
- Happy to share if monthly CPU usage > HPC owned CPUtime

OpenStack Cluster

- Interactive demand: Short term peaks.
- Let other use than running idle

OS researchers:
Deterministic Experiments

- Need “Exact-same-hardware”
- Willing to share if guaranteed availability “exact-same-hardware” is guaranteed to be available on demand.
- Peak demand: paper deadlines
HPC/HTC Cluster

- Unlimited CPU demand.
- Aggregated CPU usage per month
- Happy to share if monthly CPU usage > HPC owned CPUtime

OpenStack Cluster

- Interactive demand: Short term peaks.
- Let other use than running idle

Common shared pool

Bare Metal Servers

OS researchers:
Deterministic Experiments

- Need “Exact-same-hardware”
- Willing to share if guaranteed availability “exact-same-hardware” is guaranteed to be available on demand.
- Peak demand: paper deadlines
HPC/HTC Cluster

- Unlimited CPU demand.
- Aggregated CPU usage per month
- Happy to share if monthly CPU usage > HPC owned CPUtime

OpenStack Cluster

- Interactive demand: Short term peaks.
- Let other use than running idle

OS researchers:
Deterministic Experiments

- Need “Exact-same-hardware”
- Willing to share if guaranteed availability “exact-same-hardware” is guaranteed to be available on demand.
- Peak demand: paper deadlines

Scalability Lab
@ Red Hat

- High volume demand: 1000s of servers
- Predictable cyclical demands.
HPC/HTC Cluster

- Unlimited CPU demand.
- Aggregated CPU usage per month
- Happy to share if monthly CPU usage > HPC owned CPU time

Interactive demand: Short term peaks.

Let other use than running idle

OpenStack Cluster

OS researchers:
Deterministic Experiments

- Need “Exact-same-hardware”
- Willing to share if guaranteed availability “exact-same-hardware” is guaranteed to be available on demand.
- Peak demand: paper deadlines

Common shared pool

Bare Metal Servers

Scalability Lab
@ Red Hat

- High volume demand: 1000s of servers
- Predictable cyclical demands.
- Unlimited CPU demand.
- Aggregated CPU usage per month
- Happy to share if monthly CPU usage > HPC owned CPUtime

Interactive demand: Short term peaks.
Let other use than running idle

OS researchers:
Deterministic Experiments

- Need “Exact-same-hardware”
- Willing to share if guaranteed availability “exact-same-hardware” is guaranteed to be available on demand.
- Peak demand: paper deadlines

Scalability Lab
@ Red Hat

- High volume demand: 1000s of servers
- Predictable cyclical demands.

HIPAA Complaint Clusters

- Tedious and time consuming to built
- Utilization < 1%
- Willing to share if compliant hardware available when required.
HPC/HTC Cluster

- Unlimited CPU demand.
- Aggregated CPU usage per month
- Happy to share if monthly CPU usage > HPC owned CPUtime

OpenStack Cluster

- Interactive demand: Short term peaks.
- Let other use than running idle

OS researchers:
Deterministic Experiments

- Need “Exact-same-hardware”
- Willing to share if guaranteed availability “exact-same-hardware” is guaranteed to be available on demand.
- Peak demand : paper deadlines

Scalability Lab
@ Red Hat

- High volume demand: 1000s of servers
- Predictable cyclical demands.

HIPAA Complaint Clusters

- Tedious and time consuming to built
- Utilization < 1%
- Willing to share if compliant hardware available when required.
HPC/HTC Cluster

- Unlimited CPU demand.
- Aggregated CPU usage per month
- Happy to share if monthly CPU usage > HPC owned CPUtime

OpenStack Cluster

- Interactive demand: Short term peaks.
- Let other use than running idle

Common shared pool

Bare Metal Servers

OS researchers:
Deterministic Experiments

- Need “Exact-same-hardware”
- Willing to share if guaranteed availability “exact-same-hardware” is guaranteed to be available on demand.
- Peak demand: paper deadlines

Scalability Lab
@ Red Hat

- High volume demand: 1000s of servers
- Predictable cyclical demands.

HIPAA Complaint Clusters

- Tedious and time consuming to built
- Utilization < 1%
- Willing to share if compliant hardware available when required.

Dedicated data-centers for National emergencies utilized mostly around 2%

- Willing to share if they can use the shared pool to ramp up their systems in during emergencies.
HPC/HTC Cluster

- Unlimited CPU demand.
- Aggregated CPU usage per month
- Happy to share if monthly CPU usage > HPC owned CPUtime

OpenStack Cluster

- Interactive demand: Short term peaks.
- Let other use than running idle

Common shared pool

Bare Metal Servers

OS researchers:
Deterministic Experiments

- Need “Exact-same-hardware”
- Willing to share if guaranteed availability “exact-same-hardware” is guaranteed to be available on demand.
- Peak demand: paper deadlines

Scalability Lab
@ Red Hat

- High volume demand: 1000s of servers
- Predictable cyclical demands.

HIPAA Complaint Clusters

- Tedious and time consuming to built
- Utilization < 1%
- Willing to share if compliant hardware available when required.
HPC/HTC Cluster

- Unlimited CPU demand.
- Aggregated CPU usage per month
- Happy to share if monthly CPU usage > HPC owned CPUtime

OpenStack Cluster

- Interactive demand: Short term peaks.
- Let other use than running idle

OS researchers:
Deterministic Experiments

- Need “Exact-same-hardware”
- Willing to share if guaranteed availability “exact-same-hardware” is guaranteed to be available on demand.
- Peak demand: paper deadlines

HIPAA Complaint Clusters

- Tedious and time consuming to built
- Utilization < 1%
- Willing to share if compliant hardware available when required.

Common shared pool

Bare Metal Servers

G. Hinton
M. Schuster
Y. Bengio
K. Duraiswami

Scalability Lab @ Red Hat

- High volume demand: 1000s of servers
- Predictable cyclical demands.
How do we achieve this?

- **Goal 1: Elastic sharing of hardware between different deployment system**
  - Mechanism that supports movement of bare-metal nodes between different clusters.
  - Allows clusters to choose their own method of deploying operating system and application software.

- **Goal 2: Minimize the cost of moving nodes between clusters.**
  - Minimize the time to setup a cluster.
  - Reduce dependency of state of clusters on the underlying hardware.

- **Goal 3: Security for sharing bare-metal servers between non-trusting entities.**
  - Protecting incumbent users of bare-metal nodes from malicious previous tenants.
  - Protecting incumbent users of bare-metal nodes from future malicious tenants.

- **Goal 4: A system to incentivize sharing of bare-metal servers.**
  - Encourage users to give up their nodes when they do not need them.
  - Incentivize users to proactively make nodes available to others who may need it more.
How do we achieve this?

- **Goal 1: Elastic sharing of hardware between different deployment systems**
  - Mechanism that supports movement of bare-metal nodes between different clusters.
  - Allows clusters to choose their own method of deploying operating system and application software.

- **Goal 2: Minimize the cost of moving nodes between clusters.**
  - Minimize the time to setup a cluster.
  - Reduce dependency of state of clusters on the underlying hardware.

- **Goal 3: Security for sharing bare-metal servers between non-trusting entities.**
  - Protecting incumbent users of bare-metal nodes from malicious previous tenants.
  - Protecting incumbent users of bare-metal nodes from future malicious tenants.

- **Goal 4: A system to incentivize sharing of bare-metal servers.**
  - Encourage users to give up their nodes when they do not need them.
  - Incentivize users to proactively make nodes available to others who may need it more.
Goal 1: Elastic sharing of hardware between different deployment systems
Goal 1: Elastic sharing of hardware between different deployment systems
Goal 1: Elastic sharing of hardware between different deployment system
Goal 1: Elastic sharing of hardware between different deployment system
Goal 1: Elastic sharing of hardware between different deployment system

Hardware Isolation Layer (HIL)

A fundamental new layer in the data center that decouples server allocation from how they are provisioned.

J. Hennessey, et al., "HIL: Designing an Exokernel for the Data Center", SoCC '16
Goal 1: Elastic sharing of hardware between different deployment system

Hardware Isolation Layer (HIL)

Colocated pool of
Bare Metal Server
Goal 1: Elastic sharing of hardware between different deployment system

Hardware Isolation Layer (HIL)

Allocate
Bare Metal Servers
Goal 1: Elastic sharing of hardware between different deployment system

Hardware Isolation Layer (HIL)

Connect Network
Hardware Isolation Layer (HIL)

Goal 1: Elastic sharing of hardware between different deployment systems.

Install using your favourite Provisioning System.
Goal 1: Elastic sharing of hardware between different deployment systems

Hardware Isolation Layer (HIL)

Just 2 api calls: Move nodes between clusters
Hardware Isolation Layer (HIL)

Goal 1: Elastic sharing of hardware between different deployment systems

Just 2 api calls: Move nodes between clusters
Goal 1: Elastic sharing of hardware between different deployment system

Hardware Isolation Layer (HIL)

- **Minimal Attack Surface**: Core code ~3000 LoC
- **Standard proxy interface**:
  - Out of band management of servers
  - Network calls of switches.
- **Extensible**:
  - Cisco, Brocade, Dell, Openvswitch
  - Authentication: Database, Keystone
- **Compatible with any provisioning system**:
  - IRONIC, MaaS, emulab,
  - Forman, Geni, xCAT, M2, etc
- **Used in production for over two years at MOC**
How do we achieve this?

● **Goal 1**: Elastic sharing of hardware between different deployment systems
  ○ Mechanism that supports movement of bare-metal nodes between different clusters.
  ○ Allows clusters to choose their own method of deploying operating system and application software.

● **Goal 2**: Minimize the cost of moving nodes between clusters.
  ○ Minimize the time to setup a cluster.
  ○ Reduce dependency of state of clusters on the underlying hardware.

● **Goal 3**: Security for sharing bare-metal servers between non-trusting entities.
  ○ Protecting incumbent users of bare-metal nodes from malicious previous tenants.
  ○ Protecting incumbent users of bare-metal nodes from future malicious tenants.

● **Goal 4**: A system to incentivize sharing of bare-metal servers.
  ○ Encourage users to give up their nodes when they do not need them.
  ○ Incentivize users to proactively make nodes available to others who may need it more.
Goal 2: Minimize the cost of moving nodes between clusters.

Existing Bare Metal Offerings Provision to Local Disk - Stateful

Over the network from an ISO or a Pre-installed image

Heroic approaches have been proposed:
Goal 2: Minimize the cost of moving nodes between clusters.

Problems with Stateful provisioning

- **Slow Provisioning**
  Upto tens of minutes to provision

- **Boot Storms**
  Heavy network traffic

- **Single point of failure.**
  Loss of both OS and application

- **Bad for moving between services.**
  Have to provision from scratch, everytime.
Goal 2: Minimize the cost of moving nodes between clusters.

Could we provision Bare Metal like Virtual Machines

Distributed Storage

Bare Metal server

NETBOOT
Goal 2: Minimize the cost of moving nodes between clusters.

Problems with Stateful provisioning

- **Slow Provisioning**
  Upto tens of minutes to provision

- **Boot Storms**
  Heavy network traffic

- **Single point of failure.**
  Loss of both OS and application

- **Bad for moving between services.**
  Have to provision from scratch, everytime.
Problems with Stateful provisioning

- **Slow Provisioning**
  
  Upto tens of minutes to provision

- **Boot Storms**
  
  Heavy network traffic

- **Single point of failure.**
  
  Loss of both OS and application

- **Bad for moving between services.**
  
  Have to provision from scratch, everytime.

★ Only copy what you need.
Problems with Stateful provisioning

- **Slow Provisioning**
  Upto tens of minutes to provision

- **Boot Storms**
  Heavy network traffic

- **Single point of failure.**
  Loss of both OS and application

- **Bad for moving between services.**
  Have to provision from scratch, everytime.

Goal 2: Minimize the cost of moving nodes between clusters.

- ★ **Only copy what you need.**

- ★ **Multiple NICs and Distributed File System**
Problems with Stateful provisioning

- **Slow Provisioning**
  Upto tens of minutes to provision

- **Boot Storms**
  Heavy network traffic

- **Single point of failure.**
  Loss of both OS and application

- **Bad for moving between services.**
  Have to provision from scratch, everytime.

★ Only copy what you need.

★ Multiple NICs and Distributed File System

★ Reboot from a saved Image

Goal 2: Minimize the cost of moving nodes between clusters.
Goal 2: Minimize the cost of moving nodes between clusters.

M2: Malleable Metal as a Service

Simple Microservice

for

Rapid Provisioning and Image Management

"An Experiment on Bare-Metal BigData Provisioning", HotCloud 16
"M2: Malleable Metal as a Service." IC2E 2018
Goal 2: Minimize the cost of moving nodes between clusters.

Provisioning/Re-Provisioning Times Comparison For Single OpenStack Node

~ 25 Minutes
Goal 2: Minimize the cost of moving nodes between clusters.

Provisioning Times Comparison For Single OpenStack Node
Goal 2: Minimize the cost of moving nodes between clusters.

Provisioning Times Comparison For Single OpenStack Node

**Provisioning Times**

- **Foreman Provision**
- **Power-on Self-test (POST)**
Goal 2: Minimize the cost of moving nodes between clusters.

Provisioning Times Comparison For Single OpenStack Node

Foreman Provision

PXE Request
Goal 2: Minimize the cost of moving nodes between clusters.

Provisioning Times Comparison For Single OpenStack Node

Foreman Provision

Kernel Download & Local Disk Installation
Goal 2: Minimize the cost of moving nodes between clusters.

Provisioning Times Comparison For Single OpenStack Node
Goal 2: Minimize the cost of moving nodes between clusters.

Provisioning Times Comparison For Single OpenStack Node

- Foreman Provision
- Power-on Self-test (POST) & PXE Request
Goal 2: Minimize the cost of moving nodes between clusters.

Provisioning Times Comparison For Single OpenStack Node

Booting OS from Local Disk
Goal 2: Minimize the cost of moving nodes between clusters.

Provisioning Times Comparison For Single OpenStack Node

Foreman Provision

OpenStack Package Installation
Goal 2: Minimize the cost of moving nodes between clusters.

Provisioning Times Comparison For Single OpenStack Node
Goal 2: Minimize the cost of moving nodes between clusters.

Provisioning Times Comparison For Single OpenStack Node

~ 25 Minutes
Goal 2: Minimize the cost of moving nodes between clusters.

Provisioning Times Comparison For Single OpenStack Node

- Foreman Provision
- Foreman Re-Provision

~ 25 Minutes
Goal 2: Minimize the cost of moving nodes between clusters.

Provisioning Times Comparison For Single OpenStack Node

~ 25 Minutes
Goal 2: Minimize the cost of moving nodes between clusters.

Provisioning Times Comparison For Single OpenStack Node

- Foreman Provision or Re-Provision: ~ 25 Minutes
- Power-on Self-test (POST) & PXE Request
- M2 Provision
Goal 2: Minimize the cost of moving nodes between clusters.

Provisioning Times Comparison For Single OpenStack Node

- Foreman Provision or Re-Provision
- OS Chain Booting (iPXE)
- M2 Provision

~ 25 Minutes
Goal 2: Minimize the cost of moving nodes between clusters.

Provisioning Times Comparison For Single OpenStack Node

~ 25 Minutes
Goal 2: Minimize the cost of moving nodes between clusters.

Provisioning Times Comparison For Single OpenStack Node

Foreman Provision or Re-Provision

M2 Provision

~ 25 Minutes

~ 11 Minutes
Goal 2: Minimize the cost of moving nodes between clusters.

Provisioning Times Comparison For Single OpenStack Node

- Foreman Provision or Re-Provision: ~25 Minutes
- M2 Provision: ~11 Minutes
- M2 Re-Provision: ~5 Minutes 30 Seconds

- OpenStack Package Installation overhead removed.
Goal 2: Minimize the cost of moving nodes between clusters.

Provisioning Times Comparison For Single OpenStack Node

- Foreman Re-Provision: ~5 Minutes 30 Seconds
- M2 Re-Provision: ~5 Minutes 30 Seconds

~5X faster than Foreman
Goal 2: Minimize the cost of moving nodes between clusters.

Provisioning Times Comparison For Single OpenStack Node

- M2 Reduces Provisioning/Re-Provisioning Times.
- POST (  ) dominates M2 provisioning time.
Goal 2: Minimize the cost of moving nodes between clusters.

**M2 Architecture Overview**

- Bare Metal Allocation
- Network Isolation (layer 2)
Goal 2: Minimize the cost of moving nodes between clusters.

M2 Architecture Overview

- Data Store
- Pre-Installed Images
Goal 2: Minimize the cost of moving nodes between clusters.

M2 Architecture Overview

- Software iSCSI Server
- TGT Software iSCSI
Goal 2: Minimize the cost of moving nodes between clusters.

M2 Architecture Overview

- HIL
- CEPH

- iSCSI Gateway
- DHCP
- iPXE TFTP

- Diskless Booting from iSCSI target
Goal 2: Minimize the cost of moving nodes between clusters.

M2 Architecture Overview

- HIL
- Orchestration Engine
- CEPH
- REST Service
- iSCSI Gateway
- DHCP
- iPXE TFTP
Goal 2: Minimize the cost of moving nodes between clusters.

M2 Architecture Overview

HIL

USER

REST Service

CEPH
Goal 2: Minimize the cost of moving nodes between clusters.

**M2 Architecture Overview**

1. Reserve Nodes

- HIL
- USER

- REST Service

- Reserved Servers

- CEPH
Goal 2: Minimize the cost of moving nodes between clusters.

**M2 Architecture Overview**

1. Reserve Nodes

2. Provision Reserved Node

HIL

Reserved Servers

USER

REST Service

CEPH
Goal 2: Minimize the cost of moving nodes between clusters.

**M2 Architecture Overview**

1. Reserve Nodes

2. Provision Reserved Node

3. Clone Golden Image

- **HIL**
- **USER**
- **CEPH**
- **REST Service**
  - CEPH Interface
- **Reserved Servers**
- **Cloned Images**
Goal 2: Minimize the cost of moving nodes between clusters.

**M2 Architecture Overview**

1. Reserve Nodes
2. Provision Reserved Node
3. Clone Golden Image
4. Expose Cloned Image as iSCSI Target

- **HIL**
- **USER**
- **Reserved Servers**
- **CEPH**
- **REST Service**
  - **CEPH Interface**
  - **iSCSI Gateway**
- **Cloned Images**
Goal 2: Minimize the cost of moving nodes between clusters.

**M2 Architecture Overview**

1. Reserve Nodes
2. Provision Reserved Node
3. Clone Golden Image
4. Expose Cloned Image as iSCSI Target
5. Configure M2 to PXE boot

**REST Service**
- CEPH Interface
  - iSCSI Gateway
  - DHCP
  - iPXE TFTP

**HIL**

**Reserved Servers**

**CEPH**

**Cloned Images**

**USER**
Goal 2: Minimize the cost of moving nodes between clusters.

M2 Architecture Overview

1. Reserve Nodes
2. Provision Reserved Node
3. Clone Golden Image
4. Expose Cloned Image as iSCSI Target
5. Configure M2 to PXE boot
6. Attach Nodes to Provisioning Network

USER

HIL Interface
CEPH Interface

iSCSI Gateway
DHCP
iPXE TFTP

Reserved Servers

Cloned Images
Goal 2: Minimize the cost of moving nodes between clusters.

M2 Architecture Overview

1. Reserve Nodes

2. Provision Reserved Node

3. Clone Golden Image

4. Expose Cloned Image as iSCSI Target

5. Configure M2 to PXE boot

6. Attach Nodes to Provisioning Network

7. PXE Boot Reserved Nodes
How do we achieve this?

- **Goal 1: Elastic sharing of hardware between different deployment systems**
  - Mechanism that supports movement of bare-metal nodes between different clusters.
  - Allows clusters to choose their own method of deploying operating system and application software.

- **Goal 2: Minimize the cost of moving nodes between clusters.**
  - Minimize the time to setup a cluster.
  - Reduce dependency of state of clusters on the underlying hardware.

- **Goal 3: Security for sharing bare-metal servers between non-trusting entities.**
  - Protecting incumbent users of bare-metal nodes from malicious previous tenants.
  - Protecting incumbent users of bare-metal nodes from future malicious tenants.

- **Goal 4: A system to incentivize sharing of bare-metal servers.**
  - Encourage users to give up their nodes when they do not need them.
  - Incentivize users to proactively make nodes available to others who may need it more.
Today's Bare Metal Clouds

- Don't share machines between tenants: no co-location attacks
- However:
  - Large TCB & attack surface
  - "Trust-me" model
  - Fixed security
  - Hardware vulnerabilities is exposed to the tenants: firmware
  - Provisioning is slow

Goal 3: Security for sharing bare-metal servers between non-trusting entities.
BOLTED: a new architecture for bare metal cloud

- Minimizing trust in the provider
- Supporting even the most security sensitive tenants
- Tenants can make the cost/performance/security tradeoff
- Provisioning time as fast as virtual
- Small Microservices; most can be deployed by tenants and not in TCB

Goal 3: Security for sharing bare-metal servers between non-trusting entities.
Goal 3: Security for sharing bare-metal servers between non-trusting entities.

**Bolted architecture**

1. Allocate a node and move it into a quarantined state where node is isolated
2. Download bootloader and client side attestation software
3. Attest Node’s Firmware
4. Provision the node with tenant’s OS and applications
5. If Attestation passes: move the node to tenant’s enclave
6. If Attestation fails: moves the node to rejected pool
7. Tenant Secure Pool

Airlock

- M2
- Keylime
- HIL

Rejected Pool

Free Node Pool
How do we attest a node?

- Software hash measurements are stored in TPM
- Attestation client side sends these measurement to server side
- Attestation server side check them against a whitelist

Goal 3: Security for sharing bare-metal servers between non-trusting entities.
Goal 3: Security for sharing bare-metal servers between non-trusting entities.

What about the firmware?

- Legacy BIOS, UEFI, ... are huge
  - Vulnerable to attacks;
    - potentially enabling tenants to modify FW
  - No way for tenant to inspect FW

- LinuxBoot: A stripped down linux firmware
  - Small, Open source
  - Deterministically built

- Bolted works with either UEFI or LinuxBoot
Goal 3: Security for sharing bare-metal servers between non-trusting entities.

Answering different needs of different tenants

- Provider's Provisioning Service
- Provider's Attestation Service
- Provider's Isolation Service
- Tenant's Provisioning Service
- Tenant's Attestation Service
- Disk Encryption
- Network Encryption
Goal 3: Security for sharing bare-metal servers between non-trusting entities.

**Bolted - Performance/Security tradeoff**

- Foreman Provision
- ~700 Seconds
Goal 3: Security for sharing bare-metal servers between non-trusting entities.

Bolted - Performance/Security tradeoff

~300 Seconds
Goal 3: Security for sharing bare-metal servers between non-trusting entities.

Bolted - Performance/Security tradeoff

- Foreman
- M2 with UEFI
- M2 with LinuxBoot (~190 Seconds)
Goal 3: Security for sharing bare-metal servers between non-trusting entities.

Bolted - Performance/Security tradeoff

Bolted Provision (UEFI) ~370 Seconds
Goal 3: Security for sharing bare-metal servers between non-trusting entities.

Bolted - Performance/Security tradeoff

- **Foreman**: ~300 Seconds
- **M2 with UEFI**: ~300 Seconds
- **M2 with LinuxBoot**: Better runtime
- **Bolted Provision (UEFI)**: ~270 Seconds
- **Bolted Provision (LinuxBoot)**: Better runtime
Goal 3: Security for sharing bare-metal servers between non-trusting entities.

**Bolted - Performance/Security tradeoff**

- Bolted (UEFI): ~370 Seconds
- Bolted (LinuxBoot): ~270 Seconds
- Bolted – UEFI (Disk and Network Encryption): ~450 Seconds
- Bolted – LinuxBoot (Disk and Network Encryption): ~350 Seconds

35% overhead
Goal 3: Security for sharing bare-metal servers between non-trusting entities.
Goal 3: Security for sharing bare-metal servers between non-trusting entities.

**Runtime Overhead: Real World Applications**

16 Dell M620 nodes, 64 GB memory, 2 Xeon E5-2650 v2 2.60GHz processors 8 cores
Goal 3: Security for sharing bare-metal servers between non-trusting entities.

Bolted: A Secure Cloud with Minimal Provider Trust

Putting tenants, rather than the provider, in charge to choose the tradeoffs between security, price, and performance

“A Secure Cloud with Minimal Provider Trust”, HotCloud’18
“Tenant Controlled Security for Bare Metal Clouds”, submitted to EuroSys’19
How do we achieve this?

- **Goal 1**: Elastic sharing of hardware between different deployment systems
  - Mechanism that supports movement of bare-metal nodes between different clusters.
  - Allows clusters to choose their own method of deploying operating system and application software.

- **Goal 2**: Minimize the cost of moving nodes between clusters.
  - Minimize the time to setup a cluster.
  - Reduce dependency of state of clusters on the underlying hardware.

- **Goal 3**: Security for sharing bare-metal servers between non-trusting entities.
  - Protecting incumbent users of bare-metal nodes from malicious previous tenants.
  - Protecting incumbent users of bare-metal nodes from future malicious tenants.

- **Goal 4**: A system to incentivize sharing of bare-metal servers.
  - Encourage users to give up their nodes when they do not need them.
  - Incentivize users to proactively make nodes available to others who may need it more.
**HPC/HTC Cluster**

- Unlimited CPU demand.
- Aggregated CPU usage per month.
- Happy to share if monthly CPU usage > HPC owned CPUtime.

**OpenStack Cluster**

- Interactive demand: Short term peaks.
- Let other use than running idle.

**OS researchers:**
Deterministic Experiments

- Need “Exact-same-hardware”
- Willing to share if guaranteed availability “exact-same-hardware” is guaranteed to be available on demand.
- Peak demand: paper deadlines.

**HIPAA Complaint Clusters**

- Tedious and time consuming to build.
- Utilization < 1%
- Willing to share if compliant hardware available when required.

**Scalability Lab @ Red Hat**

- High volume demand: 1000s of servers.
- Predictable cyclical demands.
Goal 4: A system to incentivize sharing of bare-metal servers.

Requirements

How do we satisfy all these divergent needs?

- Access to hardware you own whenever you want.
- Ability to reserve nodes for future use.
- Ability to request and offer specific hardware.
- Strong incentive to give up nodes when
  - You do not need them
  - Or someone else needs them more than you do.

Solution: Marketplace with an underlying economic model
Towards a Simple Marketplace: First-Steps

Assumptions:
- Homogeneous pools of Bare-Metal Servers
- Marketplace Tracks of Tenant Credits and Server Ownership

Incentivization:
- Tenants Accrue Credits when Other Tenants Lease their Servers
- Expend Credits to Lease Servers
- Price High $\Rightarrow$ Release Servers

Goal 4: A system to incentivize sharing of bare-metal servers.
Goal 4: A system to incentivize sharing of bare-metal servers.

FLOCX: Marketplace for Bare-Metal Servers
Goal 4: A system to incentivize sharing of bare-metal servers.

Future Features

- **Bids**: Requesting hardware at desired asking price-range
- **Offers**: Complex time intervals for sharing idle nodes
- **Advanced Reservation System**: Ability to make reservations in future
- **Dynamic Pricing**: Prices reflecting demand and supply fluctuations
Goal 4: A system to incentivize sharing of bare-metal servers.

Agent-Based Trading

- Initially human bid/offer resources in the FLOCX
- Consequently, develop agents for automated trading
  - Exemplary agents for HPC and OpenStack
  - HPC Agent: maximize CPUtime
  - OpenStack Agent: maximize revenue
Future Directions

- Integrate these services in all the clusters at MGHPC.
- Scaling and Productizing:
  - Increase open source community support.
  - Improve robustness for each service.
- Formalizing the security guarantees from hardware isolation using the Universally Composable (UC) security framework.
- Expanding the attestation workflow to include all firmwares.
- Integration of extra layers of encryptions for additional compliance regimes.
- Enable Organization to Deploy and Manage agents for automatic trading of resources.
Questions / Feedback

Elastic Secure Marketplace for Trading Bare-metal Servers

where sharing (servers) is always good !!

Thank You