Secure Locking for Untrusted Clouds

Chiu C. Tan, Qin Liu, and Jie Wu

Department of Computer and Information Science
Temple University
Introduction

Cloud Computing: Dell cloud computing solutions | Dell

Components

Implementing Strategy

The Value of Cloud Software for the Cloud

Servers Optimized Infrastructure as a Service Platform as a Service Software as a Cloud Consulting Architectures Performance and Cloud

Using a Public Cloud Deploying a Private Partn...
Why Use The Cloud?

<table>
<thead>
<tr>
<th>Feature</th>
<th>Cloud solutions somewhat/significantly worse</th>
<th>Cloud solutions somewhat/significantly better</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total cost of ownership</td>
<td>9%</td>
<td>67%</td>
</tr>
<tr>
<td>Time to value</td>
<td>10%</td>
<td>64%</td>
</tr>
<tr>
<td>Availability</td>
<td>8%</td>
<td>63%</td>
</tr>
<tr>
<td>Ease of deployment</td>
<td>11%</td>
<td>62%</td>
</tr>
<tr>
<td>Ease of integration</td>
<td>14%</td>
<td>61%</td>
</tr>
<tr>
<td>Customizability</td>
<td>17%</td>
<td>57%</td>
</tr>
<tr>
<td>User adoption</td>
<td>14%</td>
<td>55%</td>
</tr>
<tr>
<td>Reliability</td>
<td>19%</td>
<td>53%</td>
</tr>
<tr>
<td>Security</td>
<td>23%</td>
<td>53%</td>
</tr>
<tr>
<td>Vendor lock-in</td>
<td>12%</td>
<td>52%</td>
</tr>
</tbody>
</table>
Main Concerns

- Ensuring security and compliance: 1% Unimportant, 35% Somewhat important, 48% Very important
- Improving manageability: 16% Unimportant, 45% Somewhat important, 36% Very important
- Deeper integration with on-premises apps: 18% Unimportant, 48% Somewhat important, 29% Very important
- Deeper integration with other cloud apps: 19% Unimportant, 50% Somewhat important, 26% Very important
- Providing mobile access: 19% Unimportant, 42% Somewhat important, 33% Very important
- Performance monitoring/management: 19% Unimportant, 43% Somewhat important, 32% Very important
- Improving data quality: 23% Unimportant, 34% Somewhat important, 39% Very important
- Reducing SaaS silos: 19% Unimportant, 42% Somewhat important, 31% Very important
- Improving application governance: 20% Unimportant, 39% Somewhat important, 34% Very important
- Reporting and analytics: 19% Unimportant, 39% Somewhat important, 33% Very important
- Driving user adoption/productivity: 25% Unimportant, 35% Somewhat important, 31% Very important
Cloud Computing Security

In-house system

• Control over security policy, e.g. key management
• Conduct system, personal audits, e.g. background checks
• System administration policies, e.g. patches
Cloud Computing Security

In-house system
- Control over security policy, e.g. key management
- Conduct system, personal audits, e.g. background checks
- System administration policies, e.g. patches

Cloud system
- SLA agreements
- Likely to be generic, not tailored to specific needs
- No easy mechanism for verification
Many Efforts To Improve Security

• Privacy preserving operations
  • Search
  • Computation
• Virtual machine security
  • Trusted computing
• Verify system operations
  • Backups
  • Delete
  • Concurrency control
Motivation

• Most current cloud systems do not provide concurrency control
• Main reason is performance
• Useful for certain applications, e.g. financial applications, certain database applications
Motivation

- Locks can be used to regulate read and write operations
- A read lock is a **shared** mode lock
- A write lock is an **exclusive** mode lock
- Users must obtain a lock before executing an operation

Write-write conflict
Motivation

• The goal is not to examine how to implement locking algorithms efficiently.
• Assuming a cloud provider claims to provide this service, how do we verify?
• Once outsourced to 3rd party cloud, we do not have the same control as in-house system.
Motivation

• Assume that there is an SLA that requires a certain response time
Motivation

• Assume that there is an SLA that requires a certain response time

Cloud

Read request

Cannot obtain lock

Read response

Stipulated response time

Additional time

Time

Bob
Motivation

• Assume that there is some cost associated with acquiring and releasing the lock
• Cloud wants to charge for the service, but try to avoid incurring the cost

Cloud

Sequence of operations within a short window. Needs to obtain lock 6 times
Motivation

• Assume that there is some cost associated with acquiring and releasing the lock
• Cloud wants to charge for the service, but try to avoid incurring the cost

Cloud

Sequence of operations within a short window. Needs to obtain lock 6 times

After reordering, needs to obtain lock 4 times
We adopt the rule from [9], where a read lock is considered. When a user wants to write to the data, we need to exclude other users. Currently, since the data remains consistent, only when a user is a bounded delay ordering of operations from the cloud. We assume that there is a timestamp mechanism querying the cloud for data. The purpose is to detect any re-ordering of operations to minimize locking operations. This could have a negative impact on users by delaying important updates. The cloud may also schedule the order of operations to mask latency delays. The cloud may pretend that a lock is unavailable to mask changes from other users. To ensure changes are propagated to all the replicas, or that the cloud did not devote sufficient resources, the cloud provider did not devote sufficient resources. Therefore, the cloud provider did not devote sufficient resources. The cloud's motivation behind the locking attacks is to ensure changes are propagated to all the replicas, or that the cloud provider did not devote sufficient resources. The cloud may pretend that a lock is unavailable to mask changes from other users. To ensure changes are propagated to all the replicas, or that the cloud did not devote sufficient resources.

A. Overview

The CloudLock protocol consists of interactions between the cloud data owner and multiple users. A data object locked in a shared mode can allow multiple users to read the data concurrently. A data object locked in an exclusive mode, no other user can lock that object in either exclusive or shared mode. Once a data object is locked in a shared mode lock, and a write lock is considered an exclusive lock. When a user holds a write lock, denoted as \( WLOCK(i) \), will use the lock number and write lock number are the same. The cloud, after waiting for \( \Delta t \) and \( n \), will respond. The types of locking attacks are as follows:

1. Issue incorrect lock number.
2. Issue the same lock to two or more users.
3. Deny issuing a lock.
4. Re-order user requests before issuing a lock.
5. Fraudulently claim that the lock is busy.

Our solution relies on maintaining two sets of lock numbers, each time the cloud returning both a read lock number and write lock number are the same. The cloud's motivation behind the locking attacks is to ensure changes are propagated to all the replicas, or that the cloud provider did not devote sufficient resources. The cloud may pretend that a lock is unavailable to mask changes from other users. To ensure changes are propagated to all the replicas, or that the cloud did not devote sufficient resources. Therefore, the cloud provider did not devote sufficient resources. The cloud's motivation behind the locking attacks is to ensure changes are propagated to all the replicas, or that the cloud provider did not devote sufficient resources. The cloud may pretend that a lock is unavailable to mask changes from other users. To ensure changes are propagated to all the replicas, or that the cloud did not devote sufficient resources.

B. Protocol description

CloudLock uses two locks, a read lock and a write lock. A data object locked in a shared mode can allow multiple users to read the data concurrently. A data object locked in an exclusive mode, no other user can lock that object in either exclusive or shared mode. Only when a user is a bounded delay ordering of operations from the cloud. We assume that there is a timestamp mechanism querying the cloud for data. The purpose is to detect any re-ordering of operations to minimize locking operations. This could have a negative impact on users by delaying important updates. The cloud may also schedule the order of operations to mask latency delays. The cloud may pretend that a lock is unavailable to mask changes from other users. To ensure changes are propagated to all the replicas, or that the cloud did not devote sufficient resources, the cloud provider did not devote sufficient resources. Therefore, the cloud provider did not devote sufficient resources. The cloud's motivation behind the locking attacks is to ensure changes are propagated to all the replicas, or that the cloud provider did not devote sufficient resources. The cloud may pretend that a lock is unavailable to mask changes from other users. To ensure changes are propagated to all the replicas, or that the cloud did not devote sufficient resources.

Our Solution

<table>
<thead>
<tr>
<th>Lock number</th>
<th>Requested</th>
<th>Received</th>
<th>Returned</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Our Solution

RLOCK (i) : read lock i, signature by cloud
WLOCK (k) : write lock k, signature by cloud

HIST(k,i) : h(k, data),
  signature of h(k, data) by user,
  signature of h(k, i) by user
One Particular Case

- RLOCK has already been assigned, and a user wants to obtain a WLOCK
One Particular Case

- RLOCK has already been assigned, and a user wants to obtain a WLOCK

User

Wait

Request to perform a write

Assigned WLOCK(k)

Last RLOCK(i)

Cloud
One Particular Case

• RLOCK has already been assigned, and a user wants to obtain a WLOCK
Our Solution

- Do not require strict synchronization between users
- Users can use HIST to detect some violations
- Owner’s table used to detect re-ordering and other operations

<table>
<thead>
<tr>
<th>Lock number</th>
<th>Requested</th>
<th>Received</th>
<th>Returned</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Our Solution

• There are other cases

<table>
<thead>
<tr>
<th></th>
<th>Request read</th>
<th>Request write</th>
</tr>
</thead>
<tbody>
<tr>
<td>Issued $RLOCK$</td>
<td>Case 1</td>
<td>Case 2</td>
</tr>
<tr>
<td>Issued $WLOCK$</td>
<td>Case 3</td>
<td>Case 4</td>
</tr>
</tbody>
</table>

• Also other types of attacks
  • Issue incorrect lock number
  • Incorrect lock operations
  • Fraudulently claim lock is busy
  • Deny issuing lock
  • Re-ordering user requests

• Please see paper for details
Conclusions and Future work

- Existing cloud places emphasis on availability, which may not be sufficient for some applications
- Reusing existing, proven, distributed system algorithms is a good idea
- Need to consider execution by, a possibly malicious, party
- Lightweight verification and attestation will be increasingly important
VirtPerf: A Performance Profiling Tool for Virtualized Environments

Prajakta Patil, Purushottam Kulkarni, Umesh Bellur

Dept. of Computer Science and Engineering
IIT Bombay

July 5, 2011
Outline

1. Introduction
2. Related Work
   - Existing Profiling Tools
   - Profiling Techniques
3. Problem Definition
4. VirtPerf
   - Architecture Details
   - Input to Tool
   - Reports and Analysis Produced by VirtPerf
   - Key Features
5. Experiments and Results
   - Experimental Setup
   - Capabilities of VirtPerf
6. Future Work
7. Conclusions
Introduction

Virtualization: An Overview

Each web application is hosted on a separate high-end server

- Popularity of web applications increase ⇒ “Server Sprawl” [9, 8]
- High infrastructure cost but low resource utilization

Solution

Eliminate old model - “One Server, One Application”

Virtualization

- Pooling common infrastructure resources
- Lowering IT costs, increasing the efficiency, utilization, flexibility and availability
Server Consolidation with Virtual Infrastructure

- Each web application is hosted on a separate high-end server
- Popularity of web applications increase ⇒ "Server Sprawl" [9, 8]
- High infrastructure cost but low resource utilization

**Solution**

Eliminate old model - "One Server, One Application"

**Virtualization**

- Pooling common infrastructure resources
- Lowering IT costs, increasing the efficiency, utilization, flexibility and availability
Virtualization: An Overview

Server Virtualization - Running multiple virtual execution environments on a single physical machine

Virtualization Models
- Paravirtualization - Xen [6]
- Full virtualization - KVM [5]
- Hardware virtualization

Benefits
- High resource utilization
- Savings on cost, energy
- Software easier to migrate
- Multiple execution environments on single hardware

Figure: Virtualization Platform
Challenges in Hosting Application in Virtual Environments

1. Estimating the peak resource requirements for each workload to decide resource provisioning
2. Analyzing effect of virtualization overheads on application performance
3. Understanding behaviour of multi-tier web applications under different workload patterns in virtual environments
4. For performance guarantee according to SLAs. - Resource usage estimation and capacity analysis is must!

Need of a profiling tool which stress applications with different workloads, monitor resource usage and performance levels in virtualized environments
Existing Profiling Tools

AutoPerf

Autoperf is an automated tool for resource profiling and capacity analysis of web-based systems deployed in physical environment.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Support for virtual platforms</td>
<td>Xen</td>
<td>Xen</td>
<td>Xen</td>
</tr>
<tr>
<td>Inbuilt load generator</td>
<td>Not present</td>
<td>Not present</td>
<td>Not present</td>
</tr>
<tr>
<td>Profiling with multiple resource allocations</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CPU overhead charge back to specific VMs</td>
<td>Implemented but with assumption of Page flipping</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Profiling with VM Migration</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

Summary

- Existing profiling tools are not aware of virtualization
- Need of profiling tool that supports multiple virtualization platforms e.g Xen, KVM
- Capability to emulate real time scenarios - Concurrent users, Thinktime distribution, Resource usage tuning for VM, Profiling while virtual machine migration
- Analysis of multi-tier applications deployed in virtual environment
Profiling Techniques

- a presents a framework for automated server benchmarking. They have concentrated on **automation policies** which are independent of underlying framework e.g server implementation, automated workload generator, resource allocations and virtualization technology.

- b presents the workload characterization of a busy WWW server (NCSA webserver) deployed on non virtualized high end HP server. They explain characteristics of the systems response times. But, they have **not studied system resource utilization patterns as a function of workload**.

- c presents a workload generation toolkit for virtualized applications, which considers three dimensions for workload generation - variation in amount of load, variation in mix of operations performed by clients and variation in popularity of data accessed. **No emphasis on resource allocation or VM migration.**

---


Virtualization Related Requirements

- Support for profiling with multiple virtualization technologies - e.g. Xen, KVM
- System wide profiling (both at the guest and host levels)
- Support for profiling multi-tier applications and report individual behavior of tiers.
- Support for profiling with setting limits on resource availability for VMs (memory, network bandwidth, CPU cores)
- Support for profiling the behavior of an application during virtual machine migration
Problem Definition

Given input - server deployment configuration, transaction information, load generation information and resource configurations for application deployed in virtualized environments

Goal

To develop a benchmarking tool which measures,

- **Performance Metrics** :
  - Response time
  - Throughput
  - Maximum Achievable Throughput

- **Resource Utilization** :
  - Network I/O
  - Disk I/O
  - CPU Utilization
  - Service Demand
**VirtPerf Architecture**

**Figure:** *VirtPerf Architecture*
Profiling Scenario

1. Reads Config Files
2. Selects load = 3 concurrent users
   Execution count = 500
3. Generate Load
4. Detects Warm Up
5. Start Profiling
6. Initialize, Set Resource Limits and Start Profiling
7. Execute 500 times
8. Load Generation complete
9. Get Data
10. Calculate Performance Metrics
10. Get Resource Usage
Input to *VirtPerf* - 1

- **Transaction Information** - It consists of URL of the server process and sequence-list to generate the dynamic URLs.

  Examples

  ```xml
  <transaction>
    <name>DomU</name>
    <sequencelist>
      <sequence name="k">1,2,3,4,5,6,7,8,9,10</sequence>
    </sequencelist>
    <url>http://192.168.50.71/WebCalendar-1.2.3/day.php?date=201011$k</url>
  </transaction>
  ```

- **Load Description** - Load levels i.e number of concurrent users and execution count for each user thread along with the distribution type (e.g. poisson, uniform) and mean value for thinktime distribution can be specified.

  Examples

  ```xml
  <farmer>
    <name>Farmer1</name>
    <executioncount>100</executioncount>
    <distribution>poisson</distribution>
    <thinktime>150</thinktime>
    <usetransaction>DomU</usetransaction>
  </farmer>
  ```
Deployment Information - It consists of server location (IP address and port number) of privileged and guest domains and name of the server process which is to be profiled. It also contains resource configuration and migration event information.

Examples

```xml
<NodeInfo type="nonJavaNode">
  <Node>10.129.41.58</Node>
  <Process>apache2</Process>
  <Port>2012</Port>
  <migrate>
    <targetvm>10.129.41.173</targetvm>
    <destinationpm>10.129.112.84</destinationpm>
    <when>20</when>
  </migrate>
  <coreinfo>
    <corecount>192.168.50.71:0:1:2:3</corecount>
    <corecount>192.168.50.72:0:1</corecount>
  </coreinfo>
  <cpuinfo>
    <cpucap>10.129.41.173:100:400</cpucap>
    <step>75</step>
  </cpuinfo>
  <meminfo>
    <memset>10.129.41.173:128:1024</memset>
    <step>2</step>
  </meminfo>
</NodeInfo>
```
Reports and Analysis Produced by *VirtPerf*

For each loadlevel (concurrent users accessing web application),

- Host domain gives resource utilization of all active domains at VM level
- Each guest domain gives its own resource usage information at process level
- Performance metrics are measured at master side

Finally,

- Maximum achievable throughput
- Load at which maximum throughput is achieved
Key Features

- Automatic Saturation Load Level Determination
- Warm-up Detection
- Profiling modes: Simple, Multiple Load Level [-c]
- Execution Count Determination
- Capacity Analysis
- Profiling modes - Fixed Multiple Load Level [-n]
- Dynamic Generation Of URLs
- Emulating real user behavior - Think time distributions (Poisson, Uniform)
- Profiling in multiple virtual environment - Xen, KVM
- Profiling with resource usage tuning [CPU, Memory]
- Profiling while virtual machine migration
Profi ling With Resource Usage Tuning

- CPU: *VirtPerf* allows the specification of CPU percentage to be allocated per VM and also configured mapping of VMs to specific CPU cores.

```xml
<coreinfo>
    <coreinfo>
        <corecount>192.168.50.71:0:1:2</corecount>
    </coreinfo>
</coreinfo>
```

- Memory: *VirtPerf* allows the specification of memory (MB) to be allocated per VM.

```xml
<meminfo>
    <memset>192.168.50.72:200:1000</memset>
    <step>200</step>
</meminfo>
```

Tools Used

- Xen: *xm*
- KVM: *virsh* and *cpulimit*
Profiling with Virtual Machine Migration

Profiling Phases
- S1-M1 - Before Migration
- M1-M2 - While Migration
- M2-S2 - After Migration

VirtPerf

< migrate >

<migrate>
  <targetvm>10.129.41.173</targetvm>
  <destinationpm>10.129.112.84</destinationpm>
  <when>20</when>
</migrate>
Thank You!