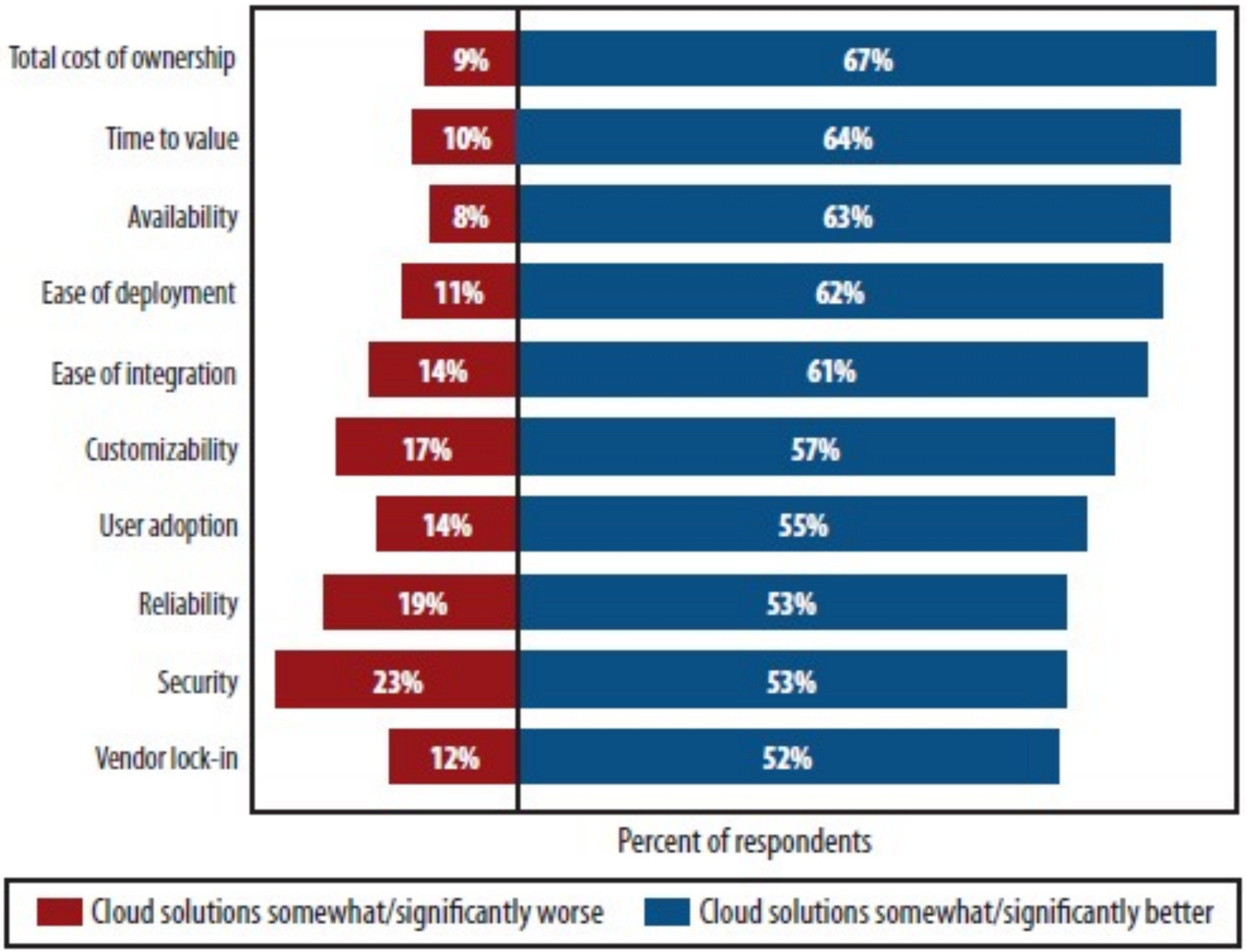
Secure Locking for Untrusted Clouds

Chiu C. Tan, Qin Liu, and Jie Wu

Department of Computer and Information Science
Temple University

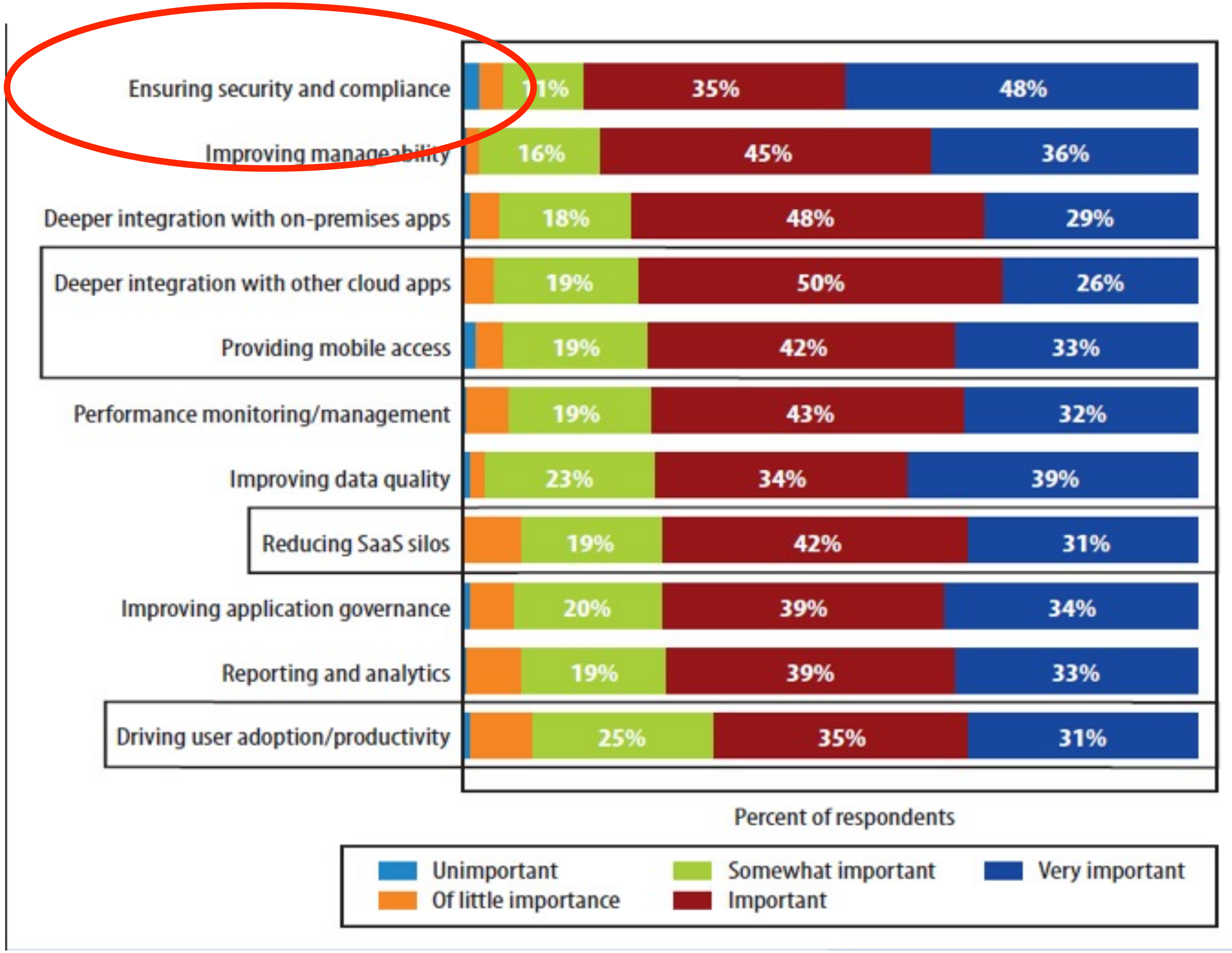
Introduction

Why Use The Cloud?



IEEE Computer 2011 Vol 44 Issue3

Main Concerns



IEEE Computer 2011 Vol 44 Issue3

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 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

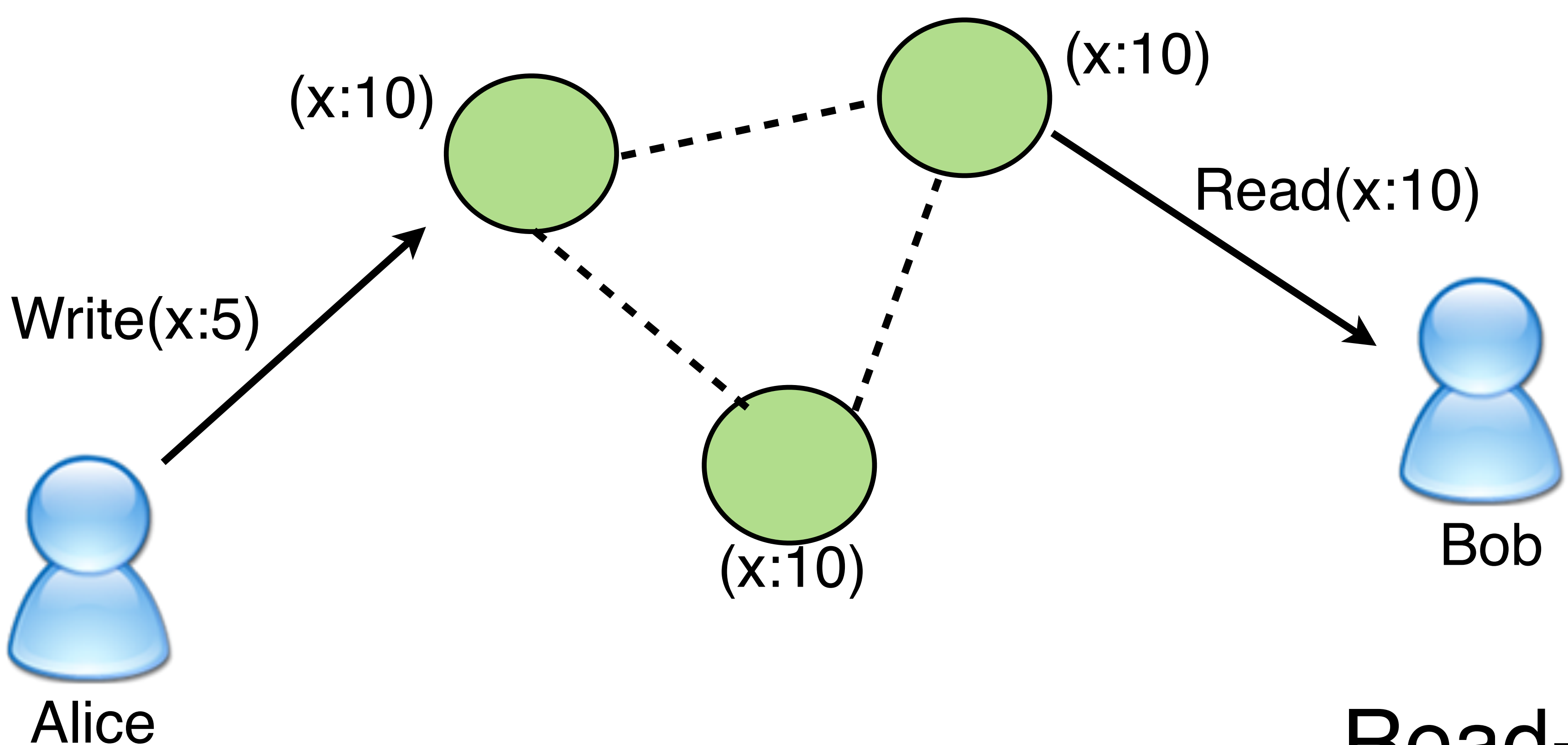
• SLA agreements
• Likely to be generic, not tailored to specific needs
• No easy mechanism for verification

Many Efforts To Improve Security

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

Motivation

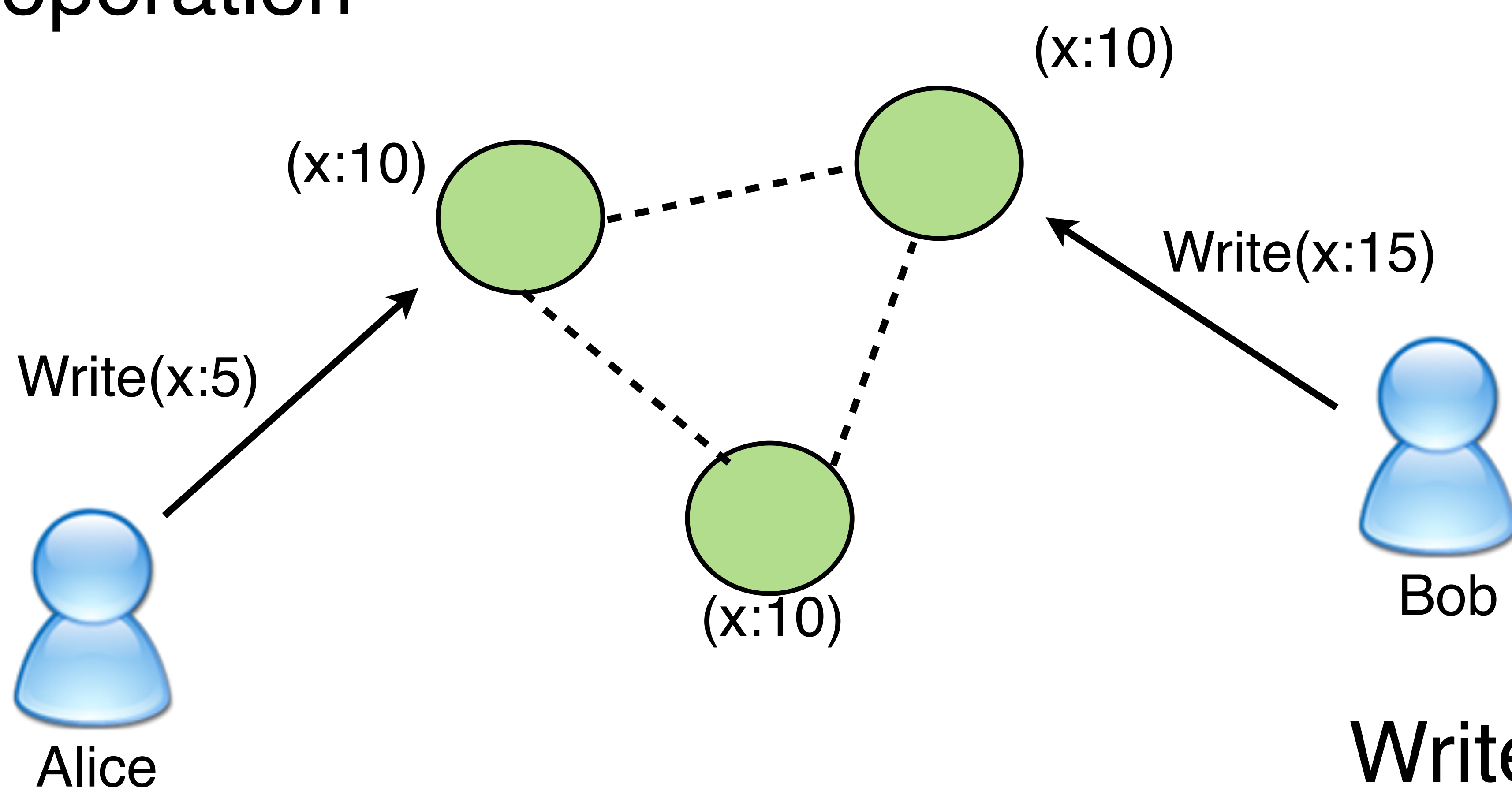
- Most current cloud systems do not provide concurrency control
- Main reason is performance
- Useful for certain applications, e.g. financial applications, certain database applications



Read-write conflict

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



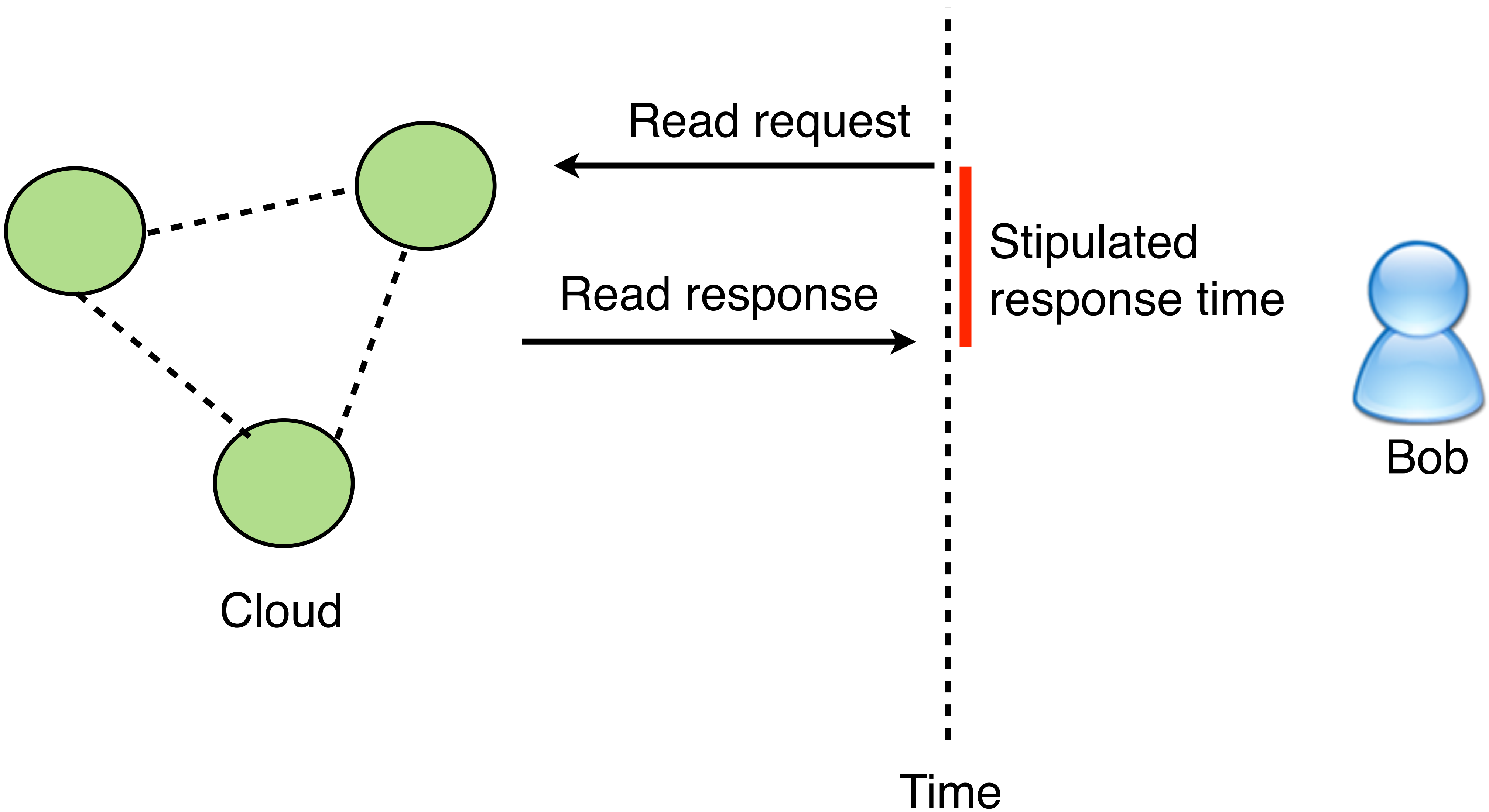
Write-write conflict

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.

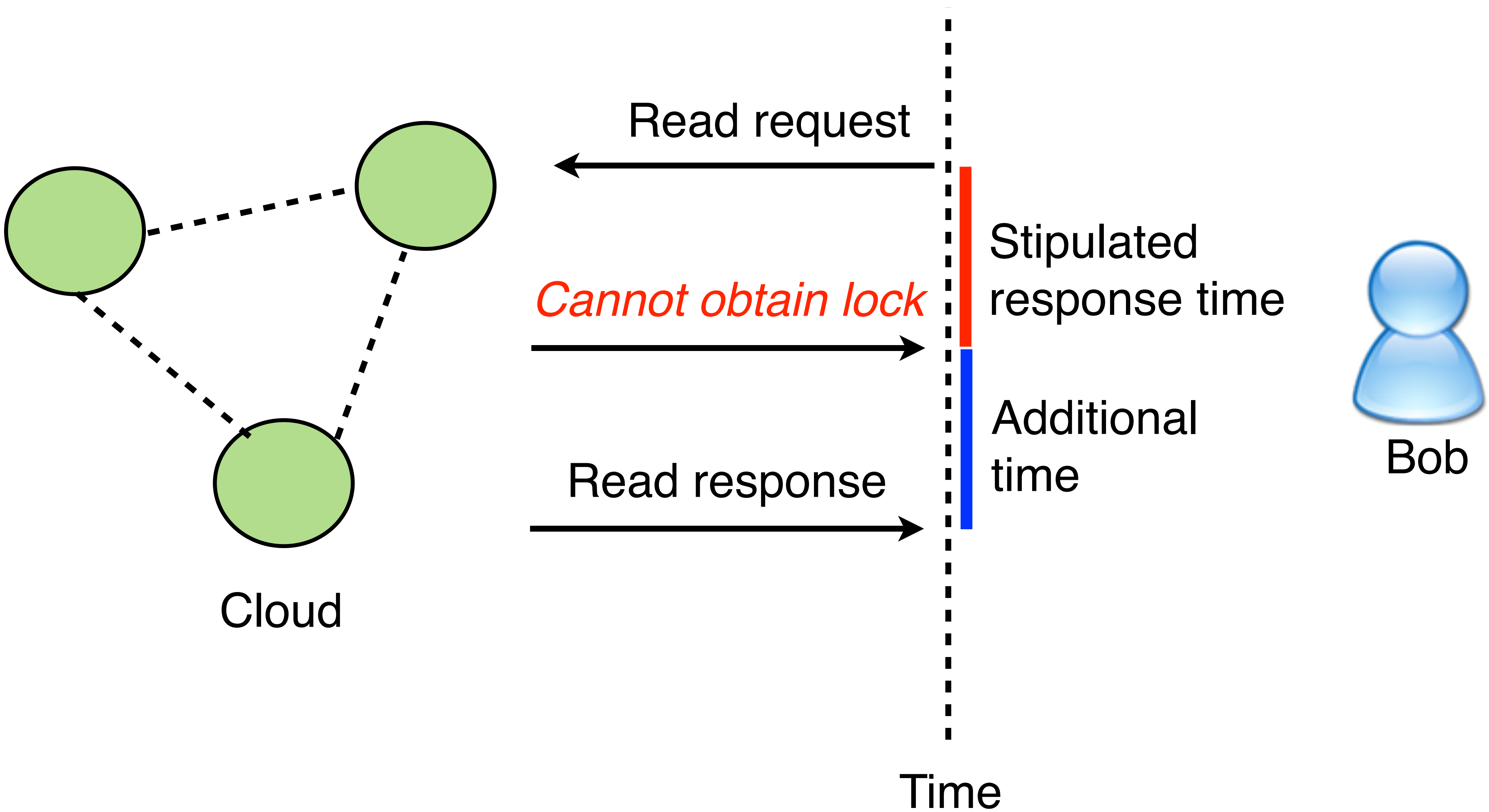
Motivation

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



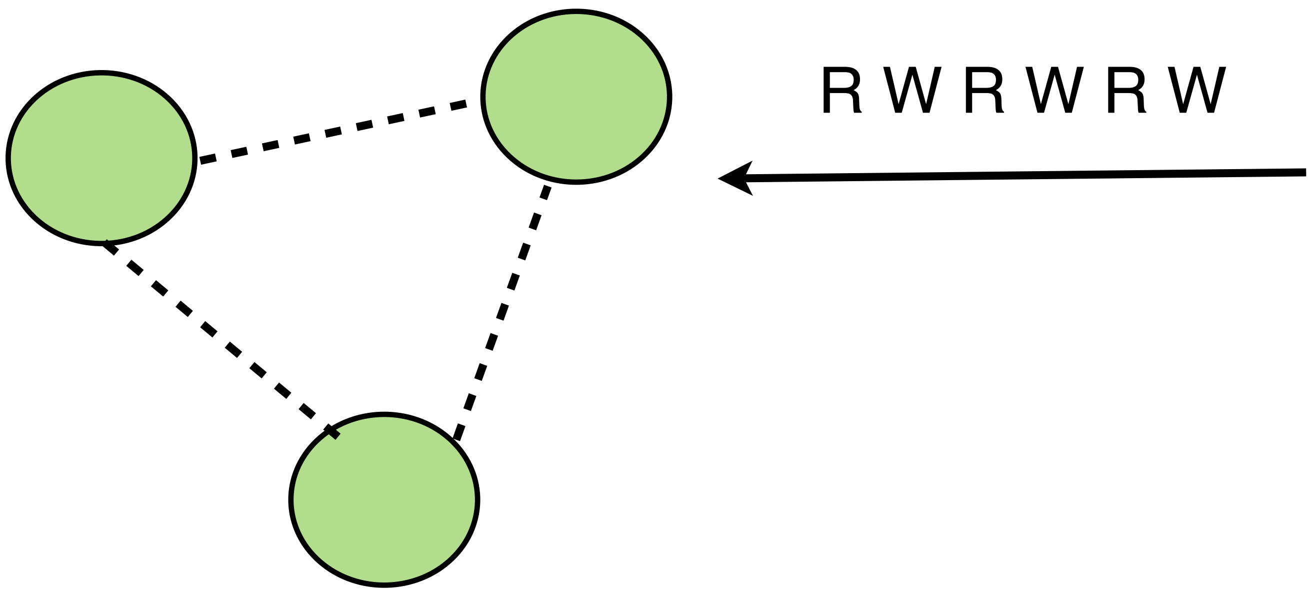
Motivation

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



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

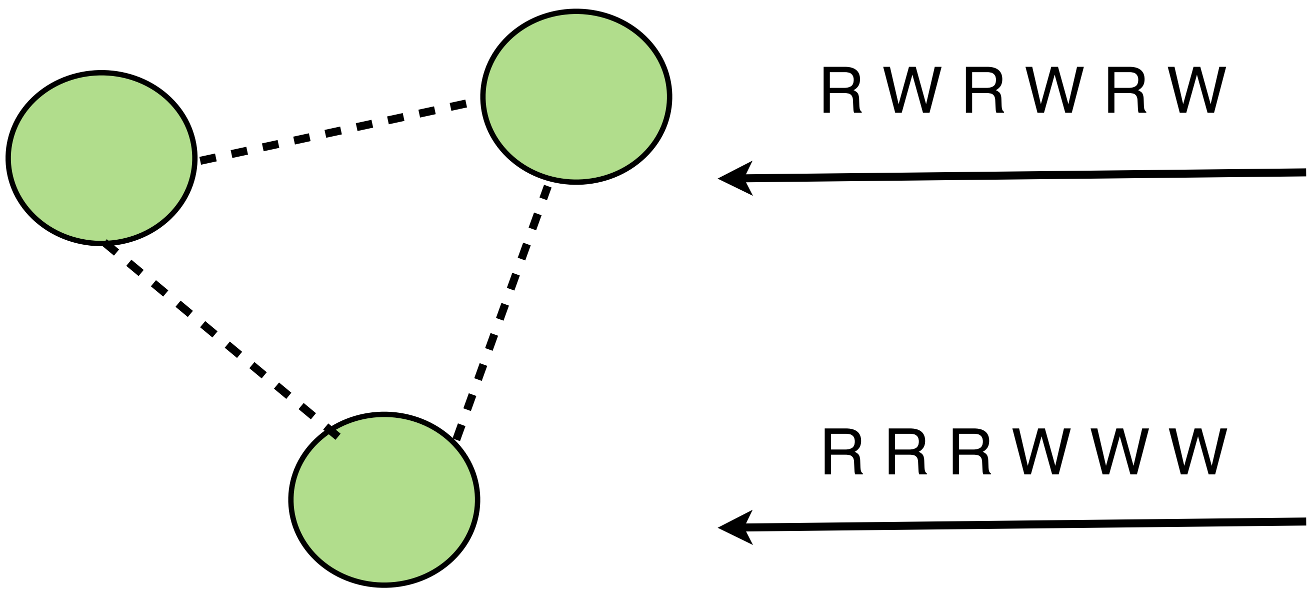


Cloud

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



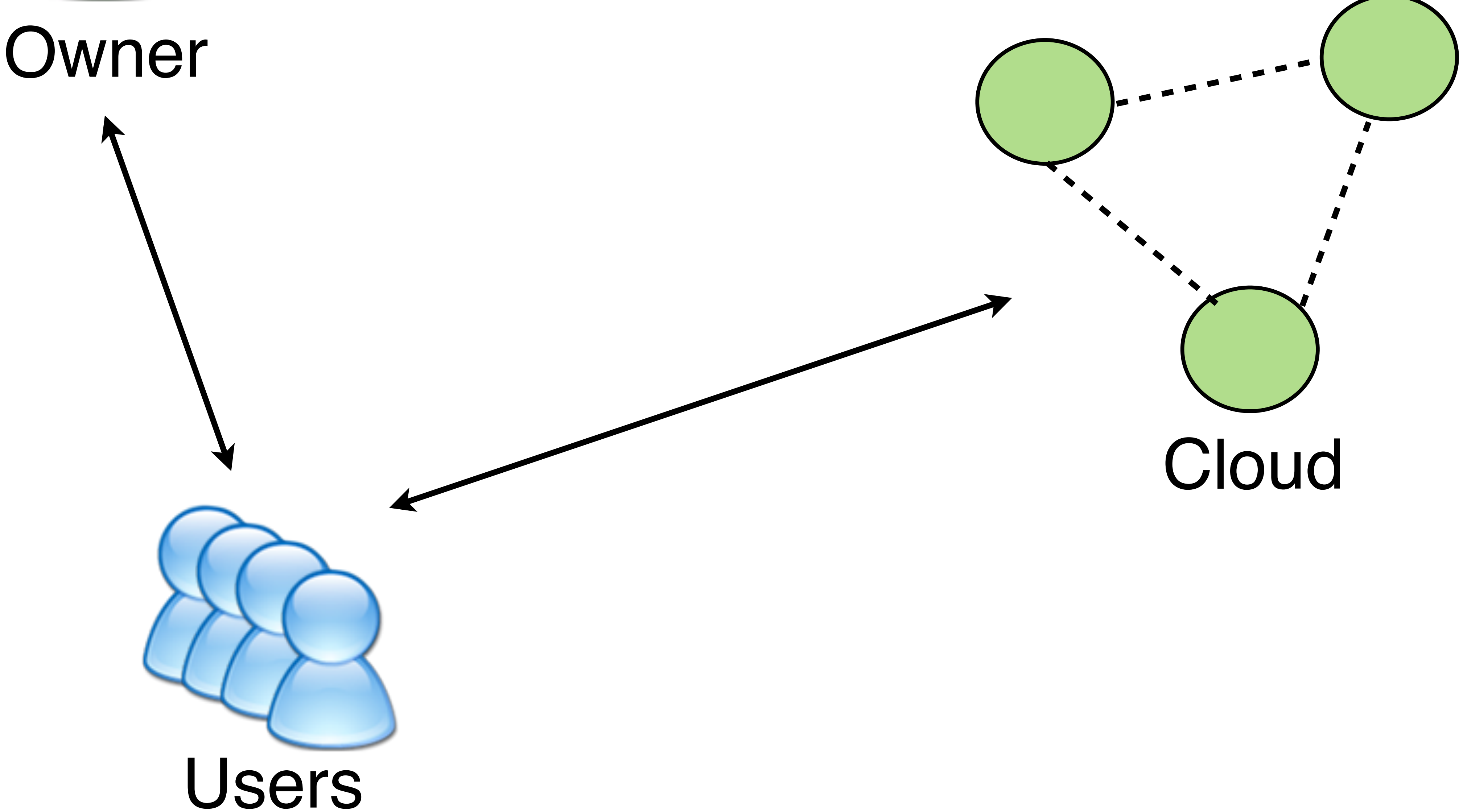
Cloud

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

After reordering, needs to obtain lock 4 times

Our Solution

Lock number	Requested	Received	Returned
...	...	...	...

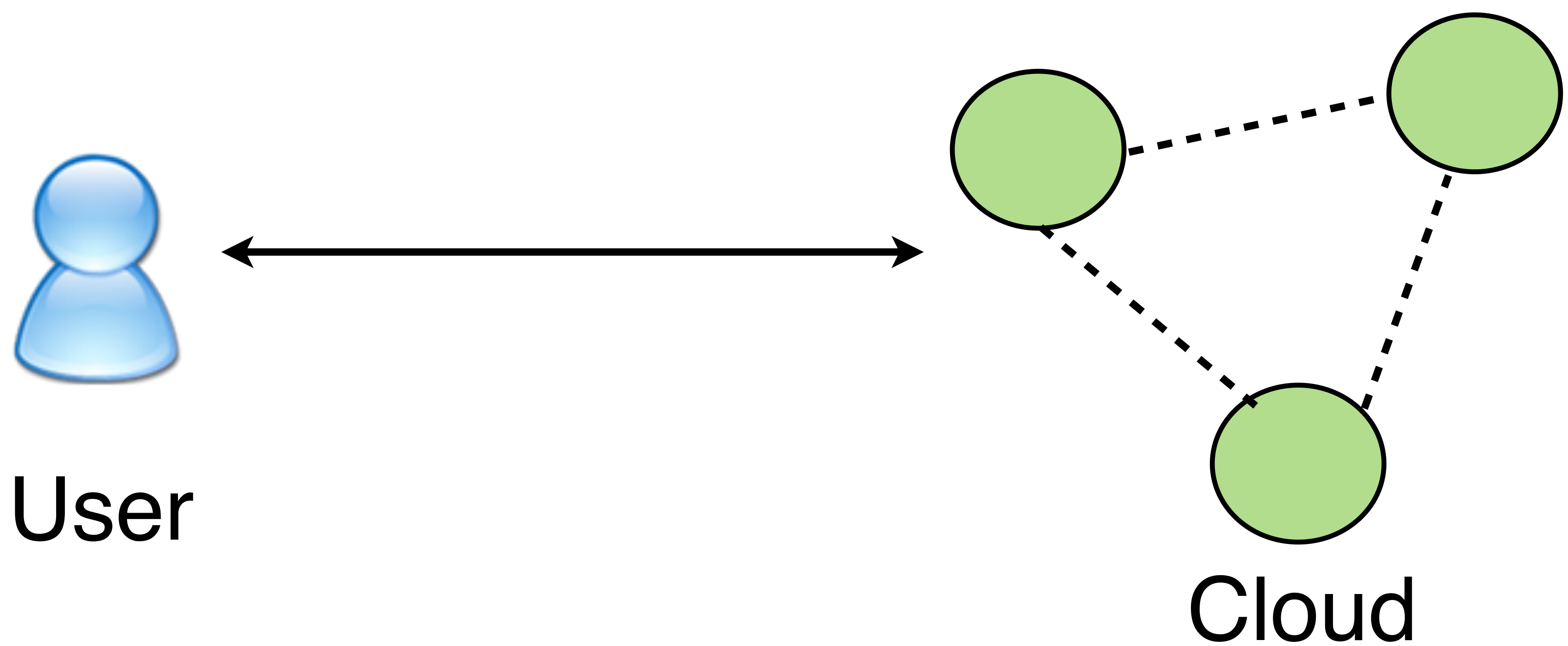


Our Solution

RLOCK (i) : read lock i, signature by cloud

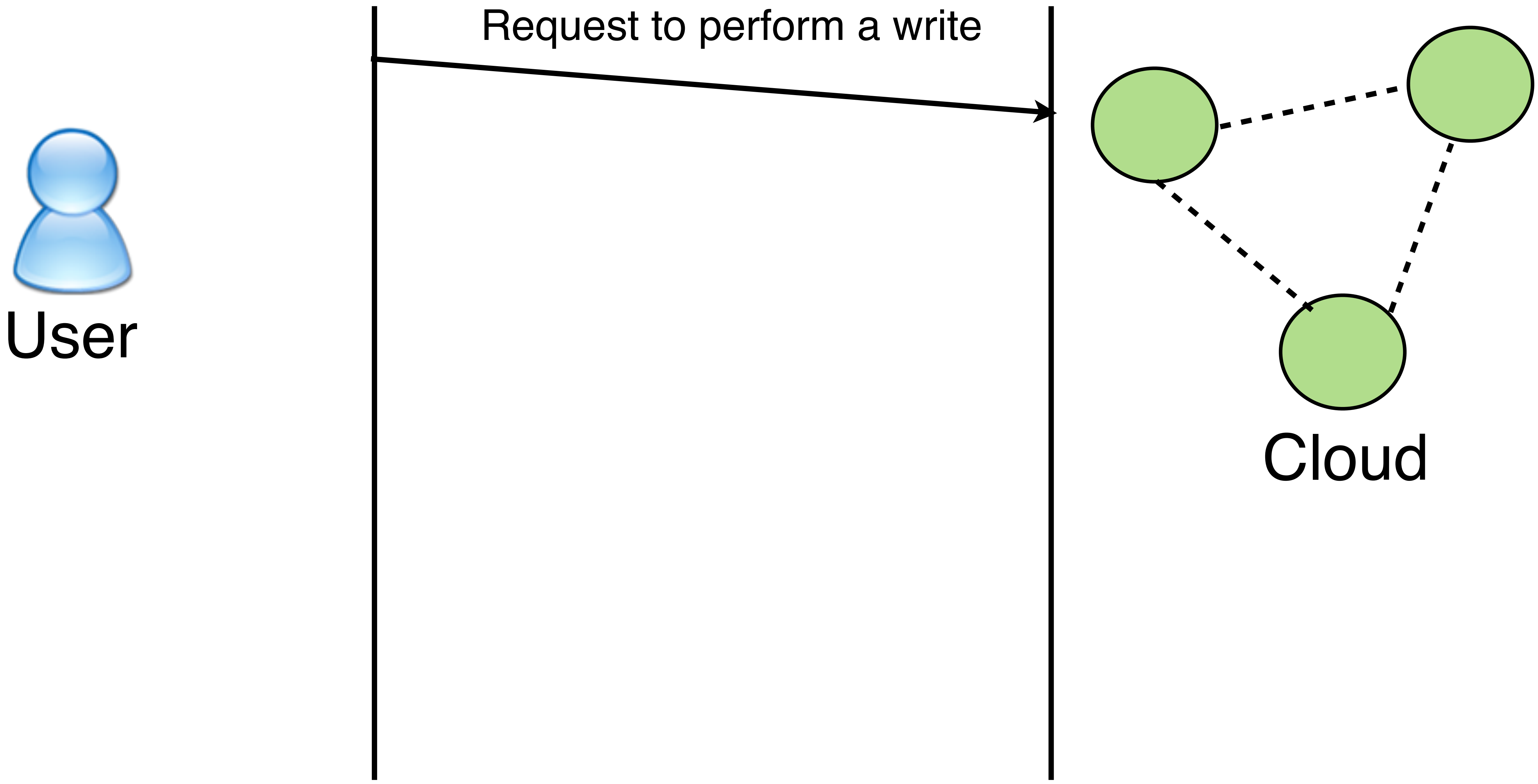
WLOCK (k) : write lock k, signature by cloud

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



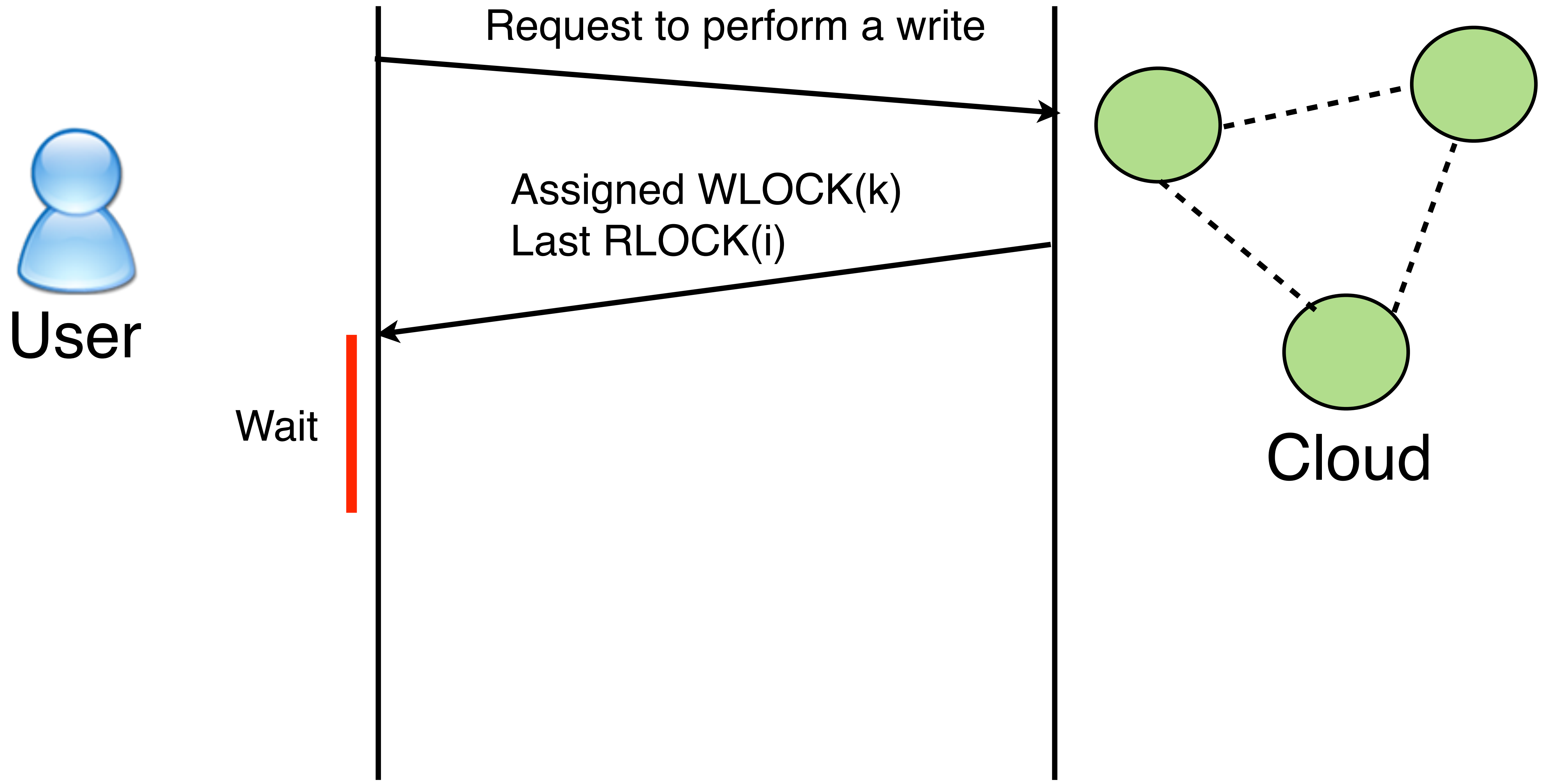
One Particular Case

- 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

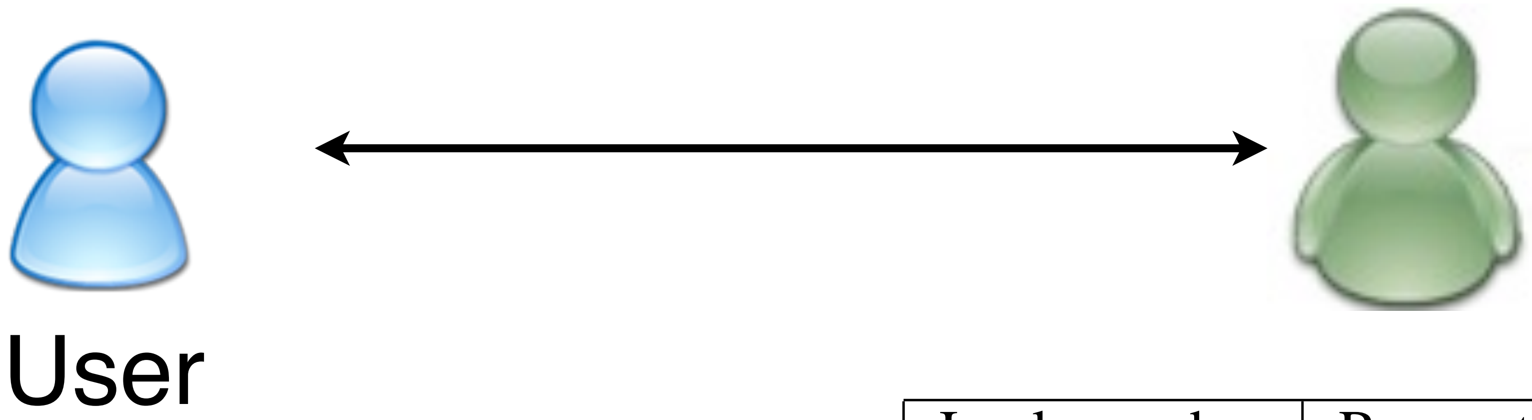


One Particular Case

- 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



Lock number	Requested	Received	Returned
...	...	...	...

Our Solution

- There are other cases

	Request read	Request write
Issued $RLOCK$	Case 1	Case 2
Issued $WLOCK$	Case 3	Case 4

- 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

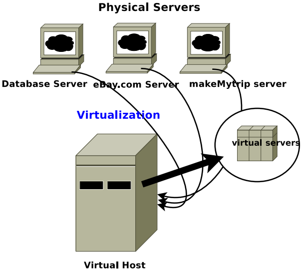
Prajakta Patil, Purushottam Kulkarni, Umesh Bellur

Dept. of Computer Science and Engineering
IIT Bombay

July 5, 2011

Outline

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

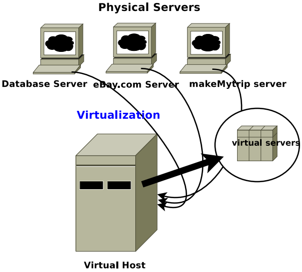
Solution

Eliminate old model - *"One Server, One Application"*

Virtualization

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

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

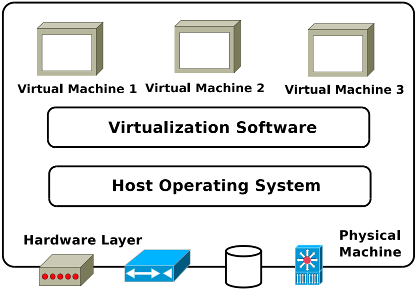


Figure: Virtualization Platform

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

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

	XenMon [3]	**Xenoprof** [1]	**Hyperic** [4]
Support for virtual platforms	Xen	Xen	Xen
Inbuilt load generator	Not present	Not present	Not present
Profiling with multiple resource allocations	No	No	No
CPU overhead charge back to specific VMs	Implemented but with assumption of Page flipping	No	No
Profiling with VM Migration	No	No	No

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.**

---

[a] Piyush Shivam et. al. Cutting Corners: Workbench Automation for Server Benchmarking. In USENIX Annual Technical Conference, 2008

[b] John A. Dilley. Web Server Workload Characterization. Technical Report HPL-96-160, Hewlett Packard Laboratories, 1996.

[c] Aaron Beitch et.al. Rain: A Workload Generation Toolkit for Cloud Computing Applications. Technical Report EECS-2010-14, UC Berkeley, 2010

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

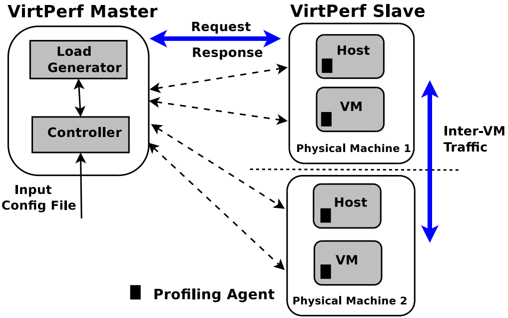


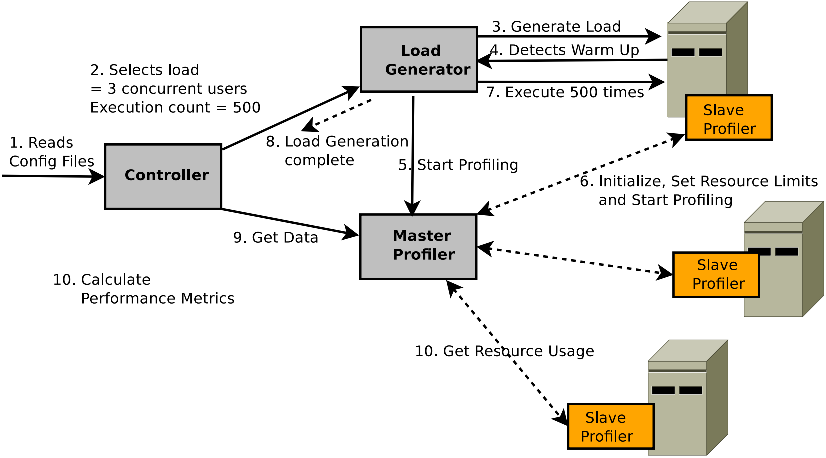
Figure: *VirtPerf* Architecture

Modules

1. Controller
2. Load Generator
3. Profiler

Profiler Module

Profiling Scenario



Input to *VirtPerf* - I

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

Examples

```
<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

```
<farmer>
    <name>Farmer1</name>
    <executioncount>100</executioncount>
    <distribution>poisson</distribution>
    <thinktime>150</thinktime>
    <usetransaction>DomU</usetransaction>
</farmer>
<farm>
    <name>Message</name>
    <usefarmer count="5">Farmer1</usefarmer>
</farm>
```

Input to *VirtPerf* - II

- **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

```
<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

Xen Measurements

Host measurements : xentop [11]
Guest measurements : ps, netstat, iostat

KVM Measurements

Host and Guest measurements : ps, netstat, iostat

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

Profiling 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

< *coreinfo* >

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

< *cpuinfo* >

```
<cpuinfo>
    <cpucap>192.168.50.72:20:80</cpucap>
    <step>20</step>
</cpuinfo>
```

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

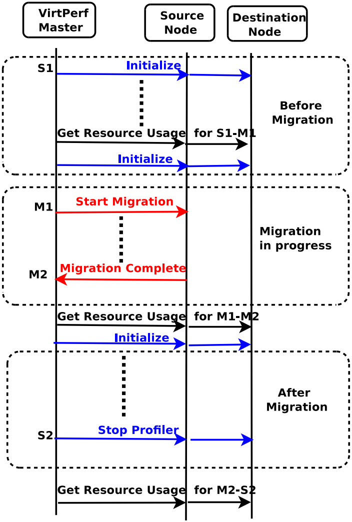
< *meminfo* >

```
<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



< *migrate* >

```
<migrate>
   <targetvm>10.129.41.173</targetvm>
   <destinationpm>10.129.112.84</destinationpm>
   <when>20</when>
</migrate>
```

Profiling Phases

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

Back

Thank You !