CloudRank-V: a Benchmark for Desktop Cloud Systems

Lin Cai

Xi’an Jiaotong University and Institute of Computing Technology, Chinese Academy of Sciences

HVC tutorial
in conjunction with The 19th IEEE International Symposium on High Performance Computer Architecture (HPCA 2013)
Outline

• Background & Motivation
• CloudRank-V Design and Implementation
• User Guide
• Use Case
Desktop Cloud (1)

• What is Desktop Cloud?
  – From the service provider’s perspective
    • Consolidate *virtual desktops*
      on a Desktop Cloud server
Desktop Cloud (2)

• What is Desktop Cloud?
  – From the user’s perspective
    • hosted virtual desktops
    • Remotely accessed
      – thin-client
      – other on-line devices
  – Used in Office systems
Desktop Cloud (2)
A Typical Desktop Cloud Deployment

Virtual Desktops
- Virtual Desktop
- Virtual Desktop
- Virtual Desktop
- Virtual Desktop

Hypervisor
- Xen Hypervisor

Hardware
- Servers
- Storages
- Management Infrastructure

Network
- Mobile Device
- Laptop
- PC
- Thin Client
- Thin Client
Why Benchmark Desktop Cloud?

• A large market share
  – more and more organizations shift toward Desktop Cloud from traditional office systems.

• Emerging workloads
  – more interactions with guaranteed QoS

• Users and Researchers
  – Evaluate different Desktop Cloud solutions
  – Optimize system and architecture
CloudRank-V

• A benchmark for Desktop Cloud server
  – Simulate representative scenarios
    • Popular office applications and complex user behaviors
  – Performance Evaluation of Desktop Cloud servers
## Related Benchmarks

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Target</th>
<th>Workload model</th>
<th>Metrics</th>
<th>Methodology</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeskBench</td>
<td>Evaluate impact of resource</td>
<td>Identical mouse/keyboard event stream for each VM</td>
<td>Client-side latency</td>
<td>Match the received screen image</td>
</tr>
<tr>
<td></td>
<td>contention and network latency</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VNCplay</td>
<td>Evaluate performance of VNC</td>
<td>Identical mouse/keyboard event stream for each VM</td>
<td>Client-side latency</td>
<td>Match the small regions around mouse event</td>
</tr>
<tr>
<td></td>
<td>protocol</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VDBench</td>
<td>Compare performance of different remote protocols</td>
<td>Identical Delay-injected packet stream for each VM</td>
<td>Client-side latency &amp; Bandwidth</td>
<td>Inject delay between every two packets when replay packets</td>
</tr>
<tr>
<td>LoginVSI</td>
<td>Capacity test</td>
<td>Identical batch of applications for each VM</td>
<td>Maximum VMs supported</td>
<td>Scheduling applications</td>
</tr>
</tbody>
</table>

9/
Why CloudRank-V

• Different methodology
  – First capacity test, then latency test

• Model realistic and complex application scenarios
  – Simulate representative mixed scenarios according to reality workloads
Outline

• Background & Motivation
• CloudRank-V Design and Implementation
• User Guide
• Use Case
A Typical Desktop Cloud Deployment

**Virtual Desktops**
- Virtual Desktop
- Virtual Desktop
- Virtual Desktop
- Virtual Desktop

**Hypervisor**
- Xen Hypervisor

**Hardware**
- Servers
- Storages
- Management Infrastructure

**Network**
- Mobile Device
- Laptop
- PC
- Thin Client

**Process of CloudRank-V**
- Design of CloudRank-V
- Workload Generation
Deployment of CloudRank-V

Model complex application scenarios

- Design of CloudRank-V
- Process of CloudRank-V
- Workload Generation
Methodology of CloudRank-V

Evaluating a Desktop Cloud server needs
1. VM configuration (peak capacity)
2. VM workloads (application loads)

Meta-load preparation

Primitive workloads used to model complex scenarios

Design of CloudRank-V

>>Process of CloudRank-V

Workload Generation

CloudRank-V

Decide Capacity

Peak value

Capacity

Performance Evaluation

Workloads

Workloads

Workload generation
Prepare Workloads

1. Choose representative application scenarios
2. Play each scenario (run the applications on each VMs)
3. Capture the workloads (packet streams of the request-reponse services)
Generate Meta-load

Select representative applications

Abstract operations

Generate meta-load

Open
Close
Play
Edit

Capture packet stream while playing each scenario
Example: Microsoft Word Scenario

• Application scenario
  - Open file: project.doc
  - Browse and modify the document
  - Save and close the file

• Abstract operations
  - [“Open”, “Browse”, “Modify”, “Save”, “Close”]

• Create meta-load
  - Perform operations and capture packets
## Designed Application Scenarios

<table>
<thead>
<tr>
<th>No.</th>
<th>Application</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Windows OS</td>
<td>[“OpenFloder”, “CreateFile”, “CloseFloder”, “Minimize”, “Maximize”]</td>
</tr>
<tr>
<td>2</td>
<td>Windows picture and fax viewer</td>
<td>[“Open”, “Zoom In”, “Rotate”, “Close”]</td>
</tr>
<tr>
<td>3</td>
<td>Adobe Reader</td>
<td>[“Open”, “Browse”, “Close”, “Minimize”, “Maximize”]</td>
</tr>
<tr>
<td>4</td>
<td>Microsoft Word</td>
<td>[“Open”, “Browse”, “Modify”, “Save”, “Close”]</td>
</tr>
<tr>
<td>5</td>
<td>Microsoft Excel</td>
<td>[“Open”, “Entry”, “Compute”, “Graph”, “Save”, “Close”]</td>
</tr>
<tr>
<td>6</td>
<td>Microsoft Powerpoint</td>
<td>[“Open”, “RunSlideshow”, “ModifySlides”, “AppendSlides”, “Save”, “Close”]</td>
</tr>
<tr>
<td>7</td>
<td>Internet Explorer</td>
<td>[“Open”, “Browse”, “Close”] (”pure text”)</td>
</tr>
<tr>
<td>8</td>
<td>Internet Explorer</td>
<td>[“Open”, “Browse”, “Close”] (”text/image mixed”)</td>
</tr>
<tr>
<td>9</td>
<td>Windows Media Player</td>
<td>[“Open”, “Play”, “Close”]</td>
</tr>
</tbody>
</table>
Decide peak capacity

1. Generate workload for each VM
2. Replay all workloads on clients
3. Capture the packet streams
4. Calculate the average latency
5. Report the number of running VMs using feedback threshold
6. Finally determine the peak capacity
Design of CloudRank-V

>> Process of CloudRank-V

Workload Generation

CloudRank-V

Decide peak capacity (cont.)

- $c_n = 1$;
- Generate workloads;
- Replay & Capture packets;
- Calculate latencies;
- Overloaded?
  - N: $c_n = c_n + 1$;
  - Y: Capacity = $c_n - 1$;
- Start a new VM;

**cn**: current VM number

$L_i$: the average latency of $i$th VM ($i = 1, 2, ..., c_n$)

Denote $L_0 = L_1$ where $c_n = 1$

Feedback control threshold:

- If (there exists any $i$, such that $L_0 / L_i < 90\%$)
  - The platform is overloaded, hence the maximum number of VMs that the platform can support is gotten
- Else
  - Start another VM and repeat above steps, until the platform is overloaded
Evaluate the Server

1. Run N VMS
2. Generate workload for each VM
3. Replay all workloads on clients
4. Monitor packet stream and calculate the average latency
5. Obtain other system or architecture metrics

Meta-load preparation

Workload generation

Decide Capacity

Performance Evaluation

- Meta-load
- Workloads
Workload Generation

- Specified workloads
  - For different goals, we need to generate different kinds of workloads
Specified Workloads

• For deciding capacity
  – Each VM’s workloads are different
    • Different applications require different resources
  – Set the short duration of each VM’s workload
    • Reduce total time of experiments
Specified Workloads (cont.)

• For performance evaluation
  – Each VM’s workload should not be identical
    • Different applications require different resources
  – The measurement should performed several times
    • Improve the precision of measured data
Example 1: for Capacity Determination

![Diagram showing capacity determination for VMs](image)

- **VM1**: 1, 5, 2, 6, 9, 4, 7, 8, 3
- **VM2**: 2, 9, 7, 1, 6, 8, 5, 3, 4
- **VM3**: 5, 7, 3, 1, 4, 9, 6, 2, 8
- **VMN**: 4, 9, 7, 1, 6, 8, 5, 3, 2

*Think time intervals marked with vertical dashed lines.*
Workloads for Capacity Determination

• Features
  – Each VM’s workload should be different

• Pattern model
  – Each VM performs every meta-load once and only once

• Methods
  – Selected from 9!=362280 candidates randomly
    • There are 9 different meta-load
Example 2: for Performance Evaluation

Design of CloudRank-V

Process of CloudRank-V

>>Workload Generation

CloudRank-V

Selected by users

think time

VM1

VM2

VM3

VMN

Measure point

Measure point

0

Time
Workloads for Performance Evaluation

• Features
  – Different workloads should be performed concurrently on the server

• Pattern model
  – Each VM performs one meta-load repeatedly

• Methods
  – For each VM, performing which meta-load follows the distribution selected by users
Outline

• Background & Motivation
• CloudRank-V Design and Implementation
• User Guide
• Use Case
Pre-installed Software Package

• On Client-side
  – Tcpreplay
    • version 3.4.4

• On Packet Monitor
  – tcpdump
    • version 4.3.0
  – Wireshark
    • version 1.8.0
Capacity Test (1)

• On Client-side
  – Detect the network address of running VMs
  • Specify the network segment to be searched
  • Detect and list network address of the running VMs

[root@hw097 ~]# ./ListIP
please enter the network: 172.18.50.0
please enter the mask: 255.255.255.192
the active IPs are listed as follow:
172.18.50.11
172.18.50.12
172.18.50.13
172.18.50.14
172.18.50.15
172.18.50.16
172.18.50.17
172.18.50.18
172.18.50.19
172.18.50.20
172.18.50.21
172.18.50.22
172.18.50.23
172.18.50.24
172.18.50.25
[root@hw097 ~]#
Capacity Test (2)

• On Client-side
  – Generate workloads for capacity determination
    • ./GenOrder
      – generates workload order
    • ./GenCapLoad
      – generates workload for each VM
  – Perform each VM’s workload
    • ./ExecCapLoad
Capacity Test (3)

• On Packet Monitor
  – Capture the request and response packets

• Offline analysis
  – Output result
Measure Performance (1)

- On Client-side
  - Maintain the maximum number of VMs running on the platform
  - Specify the distribution of the meta-loads
    - 1. same probabilities
    - 2. one high probability, others same probabilities
    - 3. specified by users

- ./${GenDistr}
  - Select the distribution:
    1. same probabilities
    2. one high probability, others same probabilities
    3. specified by users
  - input the portion

- ./${ExecPerLoad}
  - Set time interval for measuring (min): 20
  - Set times of measuring: 5
  - Sending ...
  - 20min is over, the sending will restart...
  - Sending ...

- ./${GenPerLoad}

- ./${GenPerLoad}
Measure Performance (2)

- **On Packet Monitor**
  - Capture the request and response packets

- **Offline analysis**
  - Output result
Outline

• Background & Motivation
• CloudRank-V Design and Implementation
• User Guide
• Use Case
Use Case 1: micro-architectural evaluation

I want to research the micro-architectural characterization of desktop cloud workloads.

How to setup the environments and do the experiments?
Use Case 1: Micro-architectural Evaluation

- In the past
  - Construct the desktop cloud environments
  - Hire staff to use and collect the micro-architectural level data artificially

- Using CloudRank-V
  - We can generate compound workloads according with the staff pattern and collect the micro-architectural level data automatically
    - less cost, more efficiency
Use Case 2: multi OS evaluation

I want to replace a VMM with a multi-OS for Virtualization on the server

How to evaluate the peak performance of the VMM and multi OS
Use Case 2: multi OS evaluation

VMM

- Workload generation
- Decide Capacity
  - Peak Capacity value
  - Performance Evaluation

Multi OS

- Workload generation
- Decide Capacity
  - Peak Capacity value
  - Performance Evaluation

Prepare the same workloads

Analyze the results
THANK YOU!