Virtual Machines for ROC: Initial Impressions

Pete Broadwell
pbwell@cs.berkeley.edu
Talk Outline

1. Virtual Machines & ROC: Common Paths
2. Quick Review of VMware Terminology
3. Case Study: Using VMware for Fault Insertion
4. Future Directions
Background

- Virtual machine: an efficient, isolated duplicate of a real machine – Popek & Goldberg

- VMware: an x86-based virtual machine environment
  - Runs on PCs, workstations, servers
  - Supports Linux and Windows
  - Began as a research project at Stanford
ROC & Virtual Machines: A Perfect Match?
Recovery-Oriented Features of VMs

- VM “sandboxing” provides effective **isolation**.
- Multiple VMs on one machine yields **redundancy**.
- Suspend/resume capability means fast failover and **restartability**.

- Support for checkpointing, **undo**able sessions
- Significant support for **monitoring** and **diagnostics**
- **Online verification** of recovery mechanisms?
Type I VM: Stand-Alone

- Virtual machine monitor runs on bare hardware, supports multiple virtual machines.
- Examples: VMware ESX Server, IBM z/VM
Type II VM: Hosted

- VM app uses driver to load VMM at privileged level. VMM uses host OS I/O services through VM app.

- Examples: VMware Workstation, VMware GSX Server, Connectix Virtual PC, Plex86
Hosted VM I/O Virtualization

- Host OS
- Virtual Disk
  - Host OS device drivers
- VM app
- Apps
- Guest OS
- VM
  - Virt IDE
  - Virt NIC
- VMM
- PC hardware
- Host OS device drivers
-.vmnet
- vmmon
- virt bridge

Apps
VM
Guest OS
VMM
Case Study: Opportunities for Online Fault Injection in VMware GSX Server
Why VMs for Fault Injection?

Fault injection is old news!

- ROC goals for fault injection:
  - Integrated with operating environment
  - Capable of injecting multiple types
  - Low overhead, high configurability
  - Able to expose latent errors in production systems
Which Faults are Important to Inject?

• Consider errors that have been observed on x86 PCs.

• Of these errors,
  – Which can be inserted using the existing capabilities of VMware?
  – Which require that VMware source code must be modified?
  – Which can’t be injected at all?
VMware does checking of its own!
Memory/Processor Errors

• Want to simulate processor faults, memory ECC errors.

• Problem: in VMware, processor ops & memory accesses execute directly on hardware (not simulated).

• Need to allow VM to return “machine check” exception to guest OS.

Not difficult to guess what will happen: kernel panic or blue screen.
Memory Corruption

• VMs use file system as backing for pinned memory pages – point for inserting corruption errors.

• VM driver (open source) interposes upon memory requests between VMs & host OS – can insert memory errors here.

Easy to do, but not very interesting or realistic.
Disk Fault Injection

• By default, a VM’s virtual disk image is a flat file.
• Failures: catch read/write calls to the file, return errors indicating bad blocks, device failures to OS.
• Transient failures: overwrite random portions of disk image. Should be relatively straightforward.
Network Device Faults

• VMware’s virtual network module is open-source.

• Modify module, introduce failure code at virtual bridges and hubs
  – Drop packets
  – Corrupt packets
  – Simulate slowdown
  – Simulate DOS attacks
Virtual Hub: No Faults

[root@localhost /root]# ping 10.1.2.129
PING 10.1.2.129 (10.1.2.129) from 10.1.2.128 : 56(84) bytes of data.
64 bytes from 10.1.2.129: icmp_seq=0 ttl=128 time=38.936 msec
Warning: time of day goes back, taking countermeasures.
64 bytes from 10.1.2.129: icmp_seq=1 ttl=128 time=2.918 msec
64 bytes from 10.1.2.129: icmp_seq=2 ttl=128 time=3.640 msec
64 bytes from 10.1.2.129: icmp_seq=3 ttl=128 time=3.759 msec
64 bytes from 10.1.2.129: icmp_seq=4 ttl=128 time=1.136 msec
64 bytes from 10.1.2.129: icmp_seq=5 ttl=128 time=1.352 msec
64 bytes from 10.1.2.129: icmp_seq=6 ttl=128 time=1.043 msec
64 bytes from 10.1.2.129: icmp_seq=7 ttl=128 time=3.167 msec
64 bytes from 10.1.2.129: icmp_seq=8 ttl=128 time=988 usec
64 bytes from 10.1.2.129: icmp_seq=9 ttl=128 time=1.200 msec
64 bytes from 10.1.2.129: icmp_seq=10 ttl=128 time=1.025 msec
64 bytes from 10.1.2.129: icmp_seq=11 ttl=128 time=1.300 msec
64 bytes from 10.1.2.129: icmp_seq=12 ttl=128 time=991 usec

--- 10.1.2.129 ping statistics ---
13 packets transmitted, 13 packets received, 0% packet loss
round-trip min/avg/max/mdev = 0.988/4.727/38.936/9.929 ms
[root@localhost /root]# _
Virtual Hub: Injected Faults

```
[root@localhost /root]# ping 10.1.2.129
PING 10.1.2.129 (10.1.2.129) from 10.1.2.128 : 56(84) bytes of data.
  Warning: time of day goes back, taking countermeasures.
64 bytes from 10.1.2.129: icmp_seq=0 ttl=128 time=3.880 msec
64 bytes from 10.1.2.129: icmp_seq=4 ttl=128 time=7.645 msec
64 bytes from 10.1.2.129: icmp_seq=8 ttl=128 time=1.110 msec
64 bytes from 10.1.2.129: icmp_seq=12 ttl=128 time=959 usec
64 bytes from 10.1.2.129: icmp_seq=16 ttl=128 time=1.201 msec
64 bytes from 10.1.2.129: icmp_seq=20 ttl=128 time=1.254 msec
64 bytes from 10.1.2.129: icmp_seq=24 ttl=128 time=1.515 msec
64 bytes from 10.1.2.129: icmp_seq=28 ttl=128 time=1.095 msec
64 bytes from 10.1.2.129: icmp_seq=32 ttl=128 time=1.215 msec
64 bytes from 10.1.2.129: icmp_seq=34 ttl=128 time=1.442 msec
64 bytes from 10.1.2.129: icmp_seq=38 ttl=128 time=1.429 msec
64 bytes from 10.1.2.129: icmp_seq=42 ttl=128 time=1.071 msec
64 bytes from 10.1.2.129: icmp_seq=46 ttl=128 time=1.260 msec

--- 10.1.2.129 ping statistics ---
52 packets transmitted, 13 packets received, 75% packet loss
round-trip min/avg/max/mdev = 0.959/1.928/7.645/1.801 ms
[root@localhost /root]# _
```
Cluster-Level Faults

• Use VMware’s built-in remote management interface to hard-suspend nodes in a cluster, remove network bridges.

• Verify recovery/failover routines in cluster management software.
  - Dell Scalable Enterprise Computing
  - MS Cluster Server
  - NetWare Cluster Services
  - Microsoft SQL Server!
(Virtual) Cluster Management Interface

VMware GSX Server

VMs on clone2.CS.Berkeley.EDU (System CPUs: 2, System RAM: 581 MB)

<table>
<thead>
<tr>
<th>Virtual Machine</th>
<th>Rights</th>
<th>%HE</th>
<th>Up Time</th>
<th>% CPU</th>
<th>% RAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux (PID: 11942)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VMware (PID: 11494)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Windows 2000 Professional</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

System Summary: 94.2%, 23% 24h, 55% 38%

Installing the VMware Remote Console

Installing a Console in a Windows NT 4.0 or Windows 2000 Host

Download the installer:
- VMware-console-x86-1882.exe

To install the remote console, double-click VMware-console-x86-1882.exe and follow the instructions in the installation wizard.

Installing a Console in a Linux Host

Download the installer appropriate for your Linux distribution:
- VMware-console-x86-1882-i386.rpm
- VMware-console-x86-1882.i386.tar.gz

In a terminal window, become root (su) so you can carry out the initial installation steps. Then do one of the following.
Analysis

- Levels of difficulty for different fault injection types:
  - CPU, cache, & memory (non-corruption) are hard to do.
  - Memory corruption, disk, NIC, peripherals may be medium.
  - Network, cluster level is easy.
The Big Picture

- Want to develop models for multiple correlated faults & implement them.
- Combine fault injection with introspection tools for anomaly detection & root-cause analysis.