

TA70

VProbes: Diagnostics for Production Software

Keith Adams

Sr. Staff Engineer
VMware, Inc.

Robert Benson

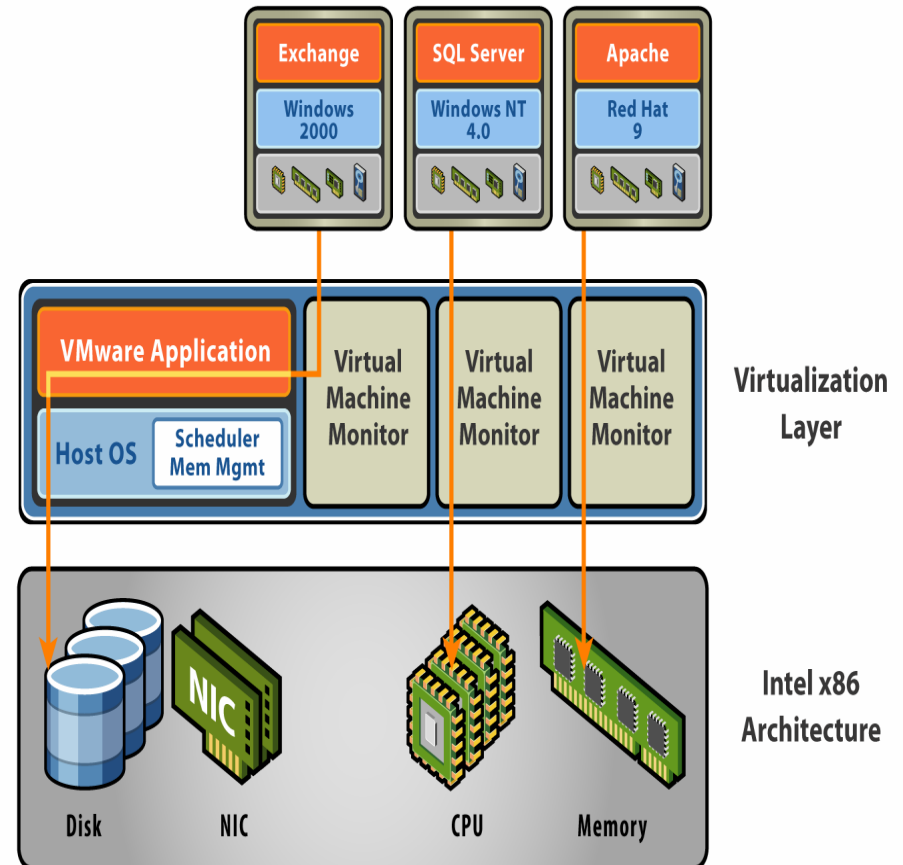
Engineer
VMware, Inc.

Alex Mirgorodskiy

Technical Staff
VMware, Inc.

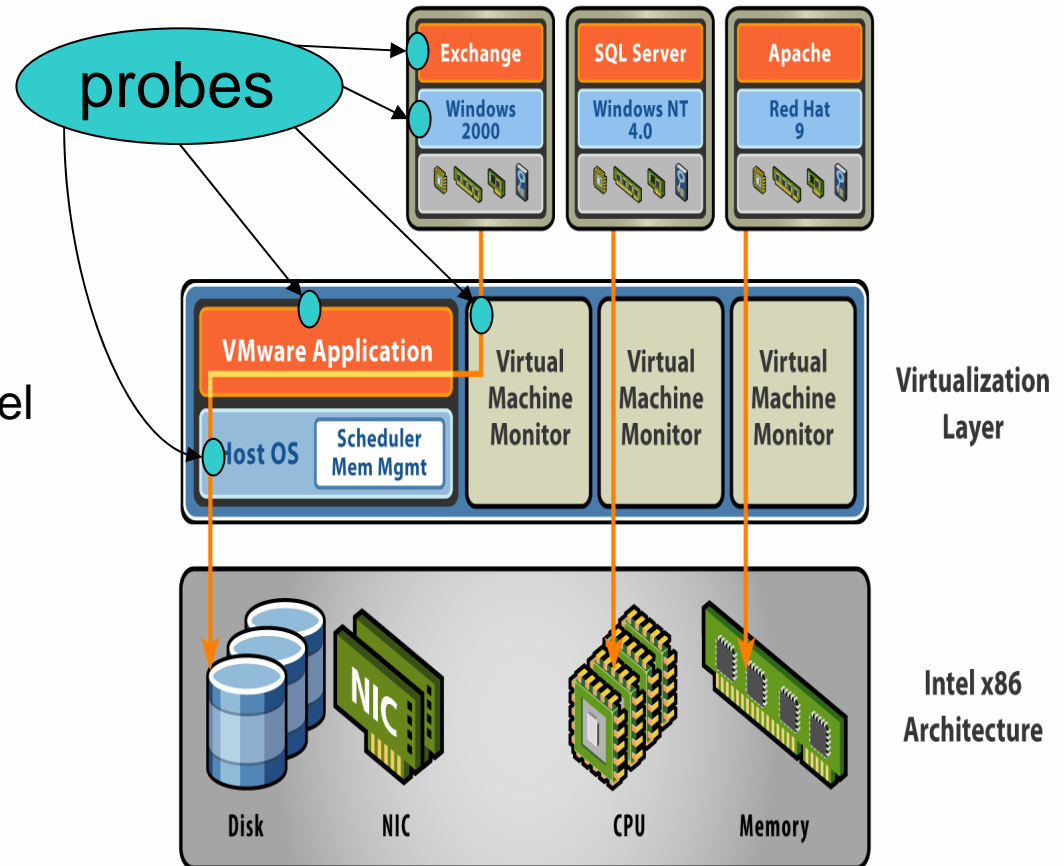
Software Trouble-Shooting is Hard

- Applications are built from multiple components
- Production environments are fragile
- Problems can be transient, site-specific, etc.



VProbes Vision

- Use V12n Layer to collect user-specified data
 - > Exposes entire software stack
 - > Correlate application-level events with system-level ones
 - > Instrumentation external to application, OS, etc.



VProbes Properties.

- ◉ **Dynamic.**

- > No restart, recompile

- ◉ **Safe.**

- > Inspect/record, but not modify, system state
 - > Cannot impede forward progress

- ◉ **OS-independent.**

- > System events described abstractly
 - > Most scripts port easily across versions, even across operating systems

- ◉ **Pre-alpha**

- > Work in progress

VProbe Workflow

- **Specify event of interest**

- E.g.: entry to function “Foo” in application bar.exe, “open file” system call, network IRQ ...

- **Specify data to gather on each event**

- Value of variables, CPU registers, current process executable name, hardware time stamp counter ...

- **Gather data**

- Load probe into running VMware guest
- Output accumulates while guest runs

- **Iterate as needed**

- Can remove, modify, or add new probes
- Mix with replay for diagnosing “one-off” error conditions...

Example: guest profiler.

```
Guest_TimerTick {  
    ticks[EIP]++;  
}
```

Example: guest top.

```
Guest_TimerTick {  
    ticks[curprocname( )]++;  
}
```

Demo: guest “top”

...

- ◉ **Sort of like “top”, except ...**

- > It works on Windows (and Linux. And Solaris. And ...)
- > Doesn't depend on “sane” guest environment

Callstack profiler

```
Guest_TimerTick {  
    ticks[gueststack()]++;  
}
```

Strace.vp

```
Guest_SystemCall {  
    printf("%s : %s\n",  
        curprocname(),  
        syscallname(RAX));  
}
```

Questions?

TA70

**VProbes: Diagnostics for
Production Software**

**Keith Adams, Robert Benson and
Alex Mirgorodskiy**

VMware, Inc.

TA70

VAssert: Write your own bug detectors under VMware's record and replay

Min Xu

VMware, Inc.

Dmitry Grinberg (UIUC)

VMware, Inc.

Bug Detectors

```
void writeArray(int index, int value)
{
    static int array[16];
    assert(index >= 0 && index < 16); ← A very simple bug detector
    array[index] = value;
}
```

- In real life
- So useful and effective → Java has runtime array bound check!
- As code and # of coders grow → increasingly more important
- Program evolves → detector evolves too
- Semantic gap → detectors often need to be compiled with the programs
- Bottom line
- You need to write your own detectors (which are specific to your code)

Summary of VAssert

○ Problems of current detectors

- > Slowdown due to checking, no-go for production
- > Probe-effects, undermine the effectiveness
- > Re-compilation, limit usefulness

○ Your new detector is built on VMware's Record/Replay

- > Eliminate detector slowdown
 - Enable expensive checks
- > Eliminate detector probe-effects
 - 100% reproducible

- > Avoid re-compilation



Virtual Machine & Record/Replay

○ VMware Virtual Machine

- > A software layer (virtual machine monitor)
- > Between a (or multiple) “guest” OS and a “host” OS
- > Guest thinks it “owns” the hardware
- > Reality: hardware shared among guests and host

○ Record and Replay

- > Recreate a program execution deterministically
- > Instruction-exact between recording and replaying
- > Low overhead, compact log file

Recorder implementation

- **VMM (virtual machine monitor) is an ideal place**

- > Observe all inputs
- > Control all internal interactions

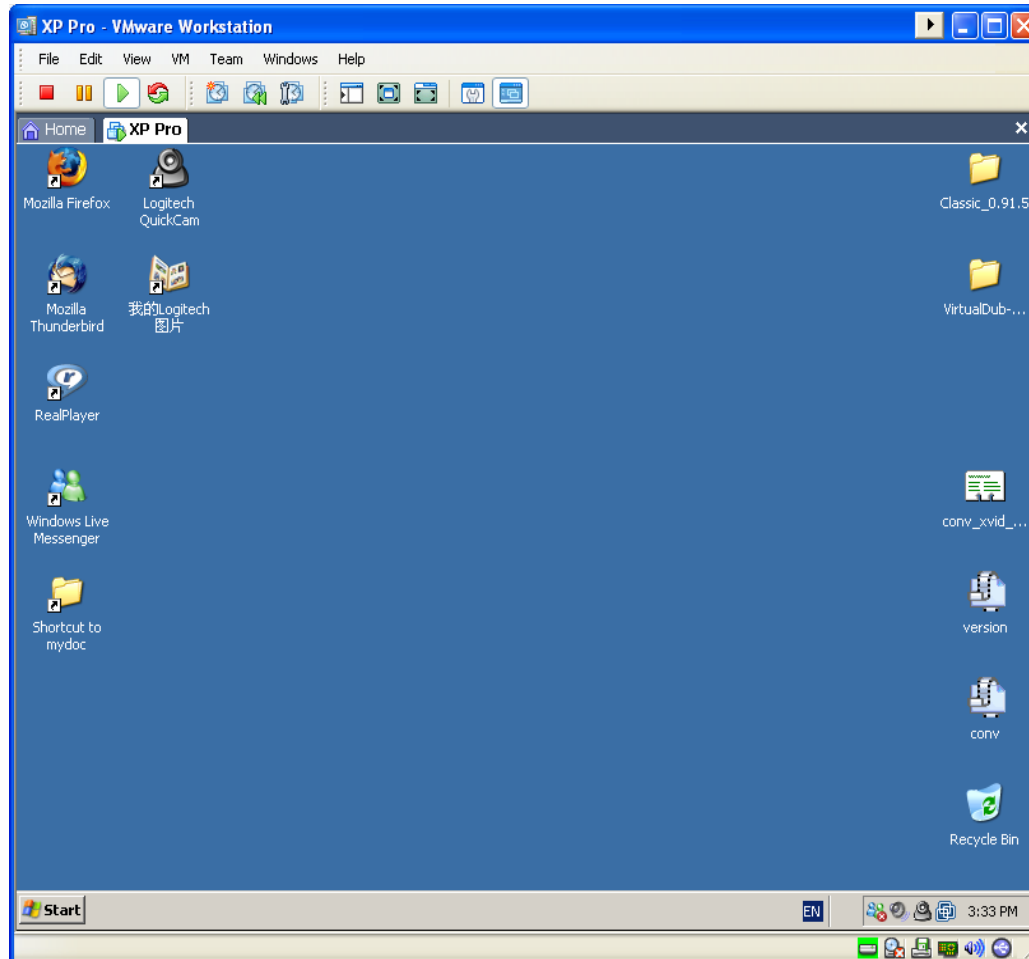
- **Challenges**

- > Instruction determinism correctness!
- > Instruction countingperformance!
- > Input loggingperformance!

- **Solutions**

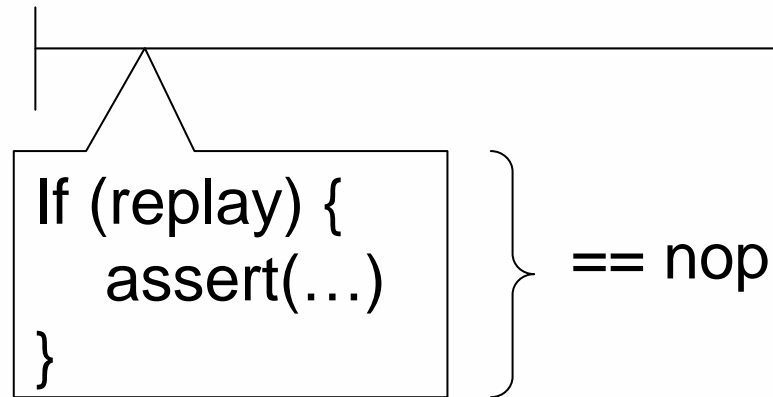
- > Direct execution, binary translation, interpretation

VM Replay Demo

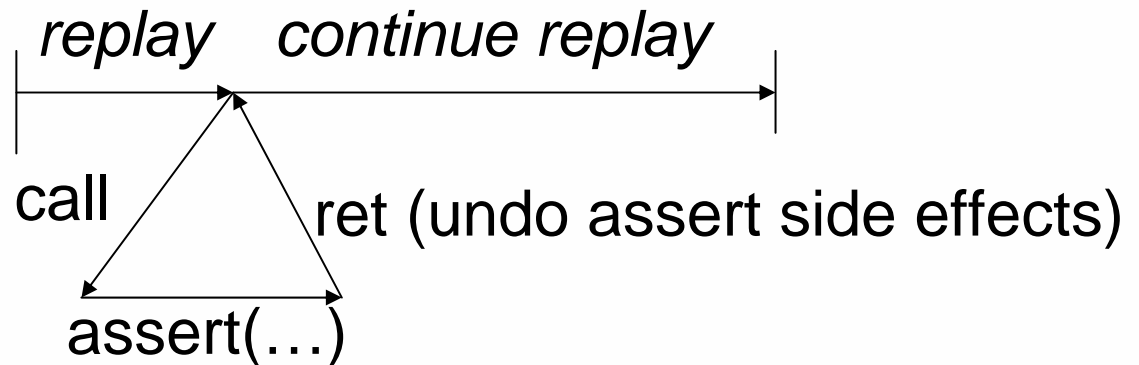


VAssert – the idea

Native or
Not replaying
(w/o assert)



Replay
(w/ assert)



VAssert SDK

- **Vassert(expr)**
 - > Like traditional assert()
- **Vlog(level, fmt, ...)**
 - > Like printf()
- **Vwatchpoint(addr, size, callback)**
 - > Like hardware watchpoint in a debugger
 - > Programmable
 - > Support more watchpoints than hardware
 - > A variation is vbreakpoint(addr, callback)
- **All of above happen only in deterministic replay**

Demo 1

- **Basic “Hello World”**

- > How to use it

Demo 2

◉ Real life scenario

- > Windows: firefox
- > Linux: rxvt-unicode (omitted)
- > Take away 1: avoid traditional assert
- > Take away 2: deterministic & offline automation
- > Take away 3: easy to integrate with existing code
 - Firefox: one (long) night
 - Rxvt-unicode: 2 hours

Demo 3

- **More sophisticated example**

- > A linked list insert()

```
void insert(int value) {  
    node *n = malloc(sizeof *n);  
    goodNodePtr.add(n);  
    vwatchpoint(&n->next, sizeof n->next, &callback);  
    n->next = ...  
}
```

- > Now in the callback()

```
Bool callback(node* n) {  
    assert(goodNodePtr.has(n));  
}
```

- **Take away: powerful tool in building custom bug detectors**

Conclusion

- **VMware record & replay can help software QA**
- **VAssert SDK is**
 - > Avoiding assertion slowdown
 - > **Deterministic** & **automated** offline checking
 - > Powerful tool in building custom bug detectors
- **Feedback wanted**
 - > Will you integrate this into your QA?
 - > Will you write vassert-based bug detectors?

Questions?

TA70

**VAssert: Write your own bug detectors under
VMware's record and replay**

Min Xu and Dmitry Grinberg (UIUC)

VMware, Inc.



VMWORLD 2007

EMBRACING YOUR VIRTUAL WORLD

BREAKOUT SESSION