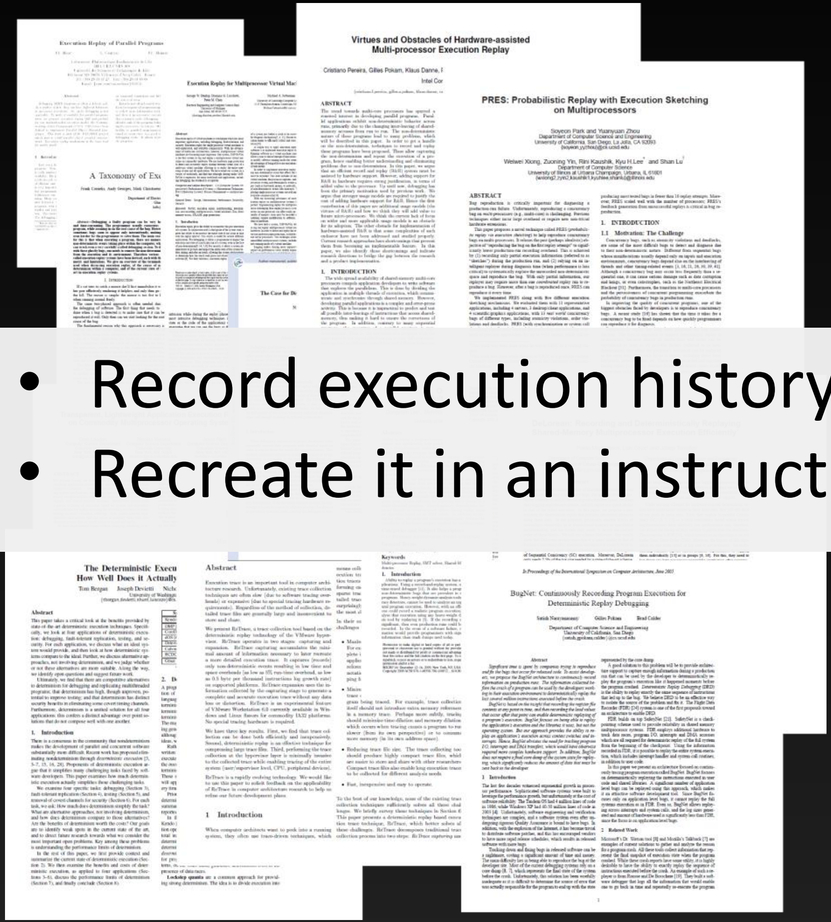
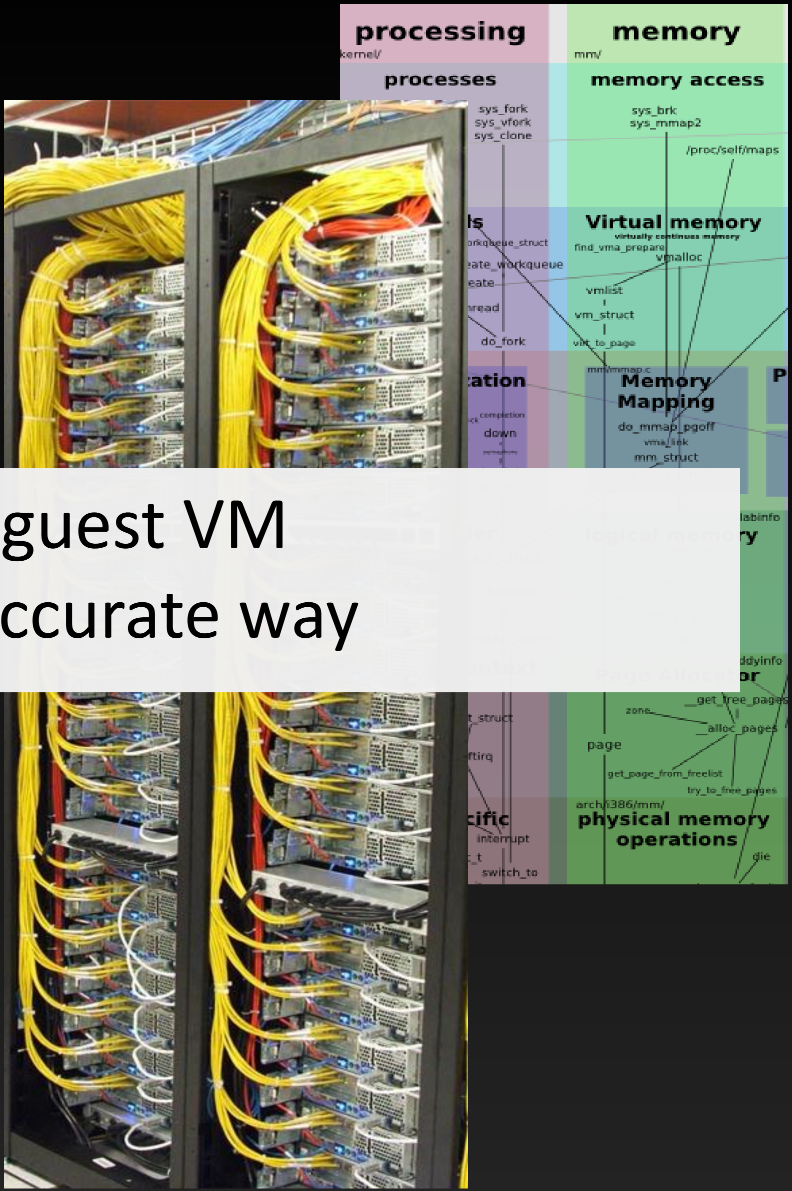
XenTT: Deterministic Systems Analysis in Xen

Anton Burtsev, David Johnson, Chung Hwan Kim, Mike Hibler, Eric Eide, John Regehr

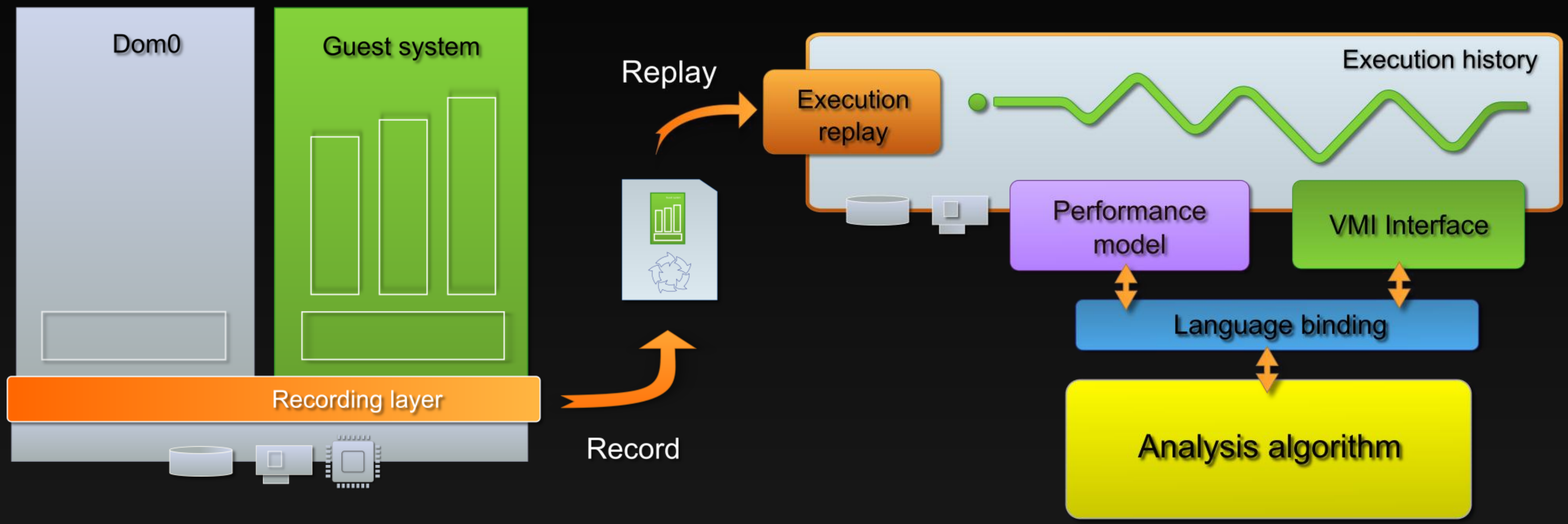
University of Utah

August 27-28, 2012
San Diego, CA, USA


Execution Replay of Parallel Programs
Virtues and Obstacles of Hardware-assisted Multi-processor Execution Replay
Execution Replay for Multiprocessor Virtual Ma
A Taxonomy of Ex
PRES: Probabilistic Replay with Execution Sketching on Multiprocessors
The Deterministic Execu How Well Does it Actually
BugNet: Continuously Recording Program Execution for Deterministic Replay Debugging



- Record execution history of a guest VM
- Recreate it in an instruction-accurate way

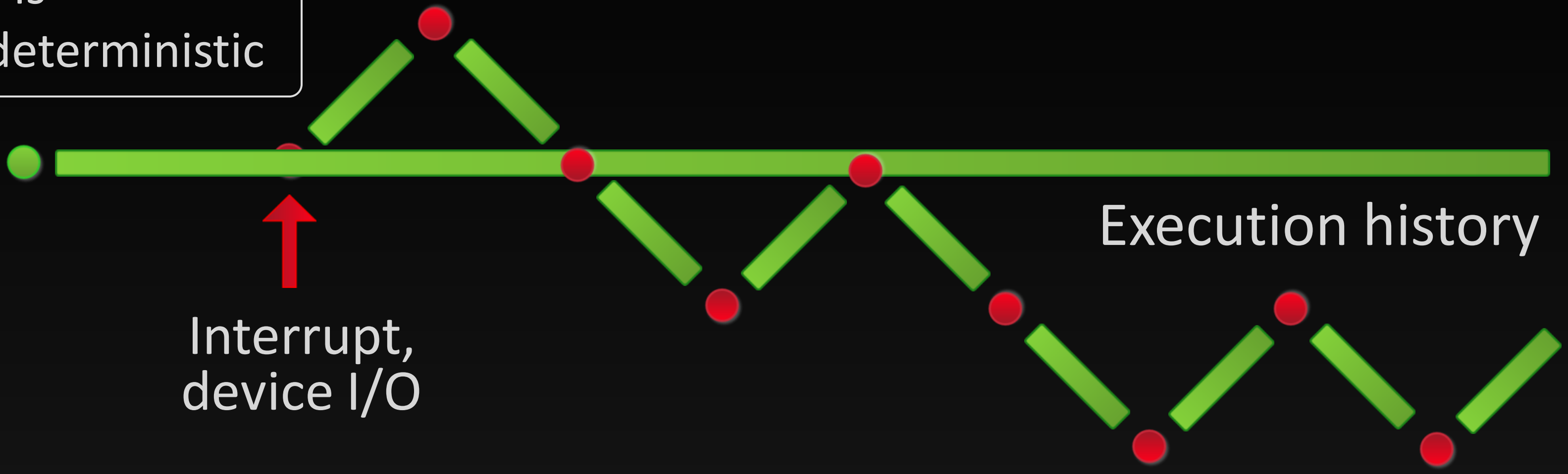


- Execution replay is a right way to analyze systems
- We need a practical tool!

Deterministic replay

Determinism

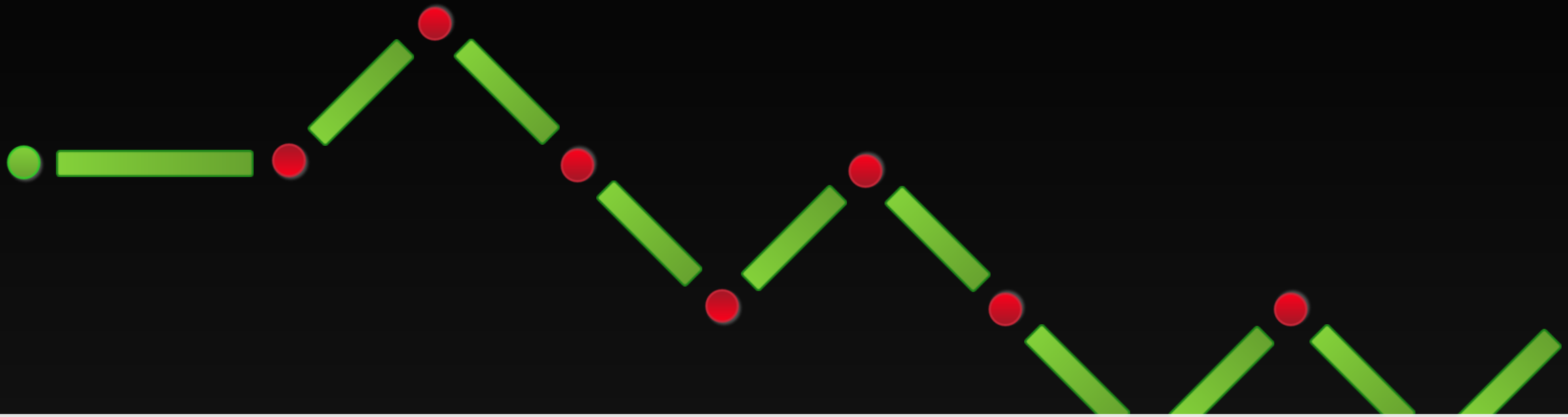
CPU is deterministic



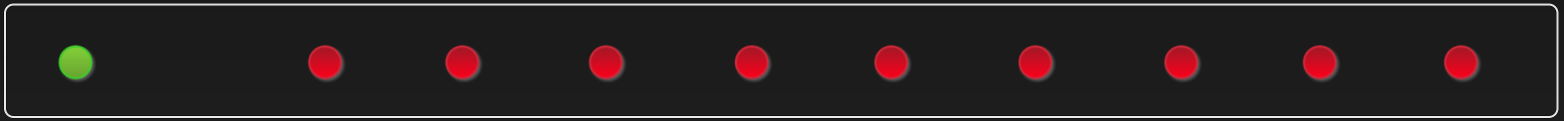
Execution history

Interrupt, device I/O

Recording



- Determinism of the execution environment
- Instruction-accurate position of events



Event log

Instruction-accurate position of events

```
label:  ...
        mov
        shr
        mov
→   rep movsl
        test
        jne label
        ...
```

- Number of instructions since boot
  - Intel has a hardware counter
  - It’s not accurate

- Hardware instruction counter
  - Preempt execution of a system at the same instruction
  - Hardware instruction counter + single-stepping

Determinism in Xen

Nondeterministic events

Simple Model

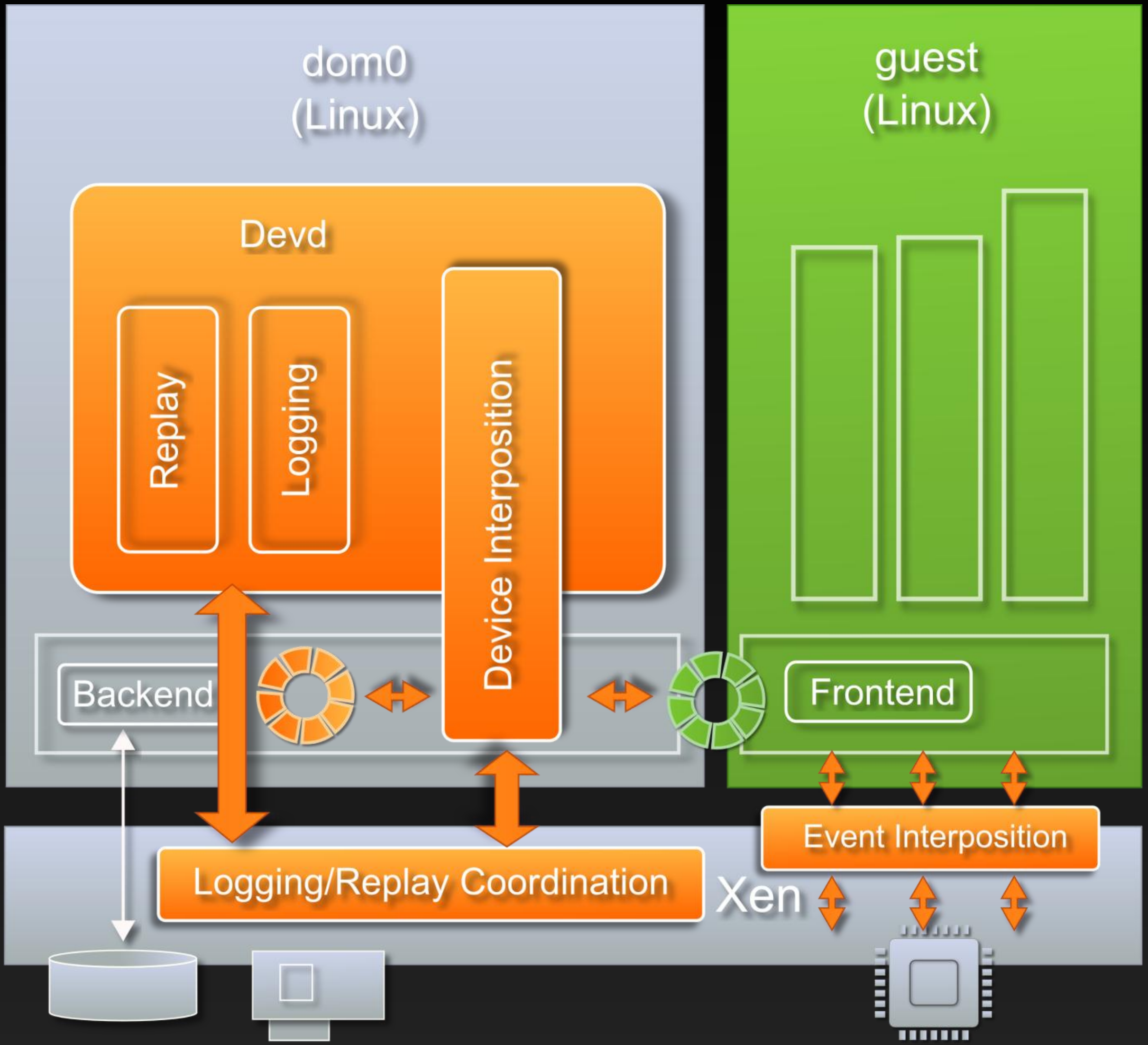
System = memory pages + registers

Events = memory updates
(time, device I/O, system calls)

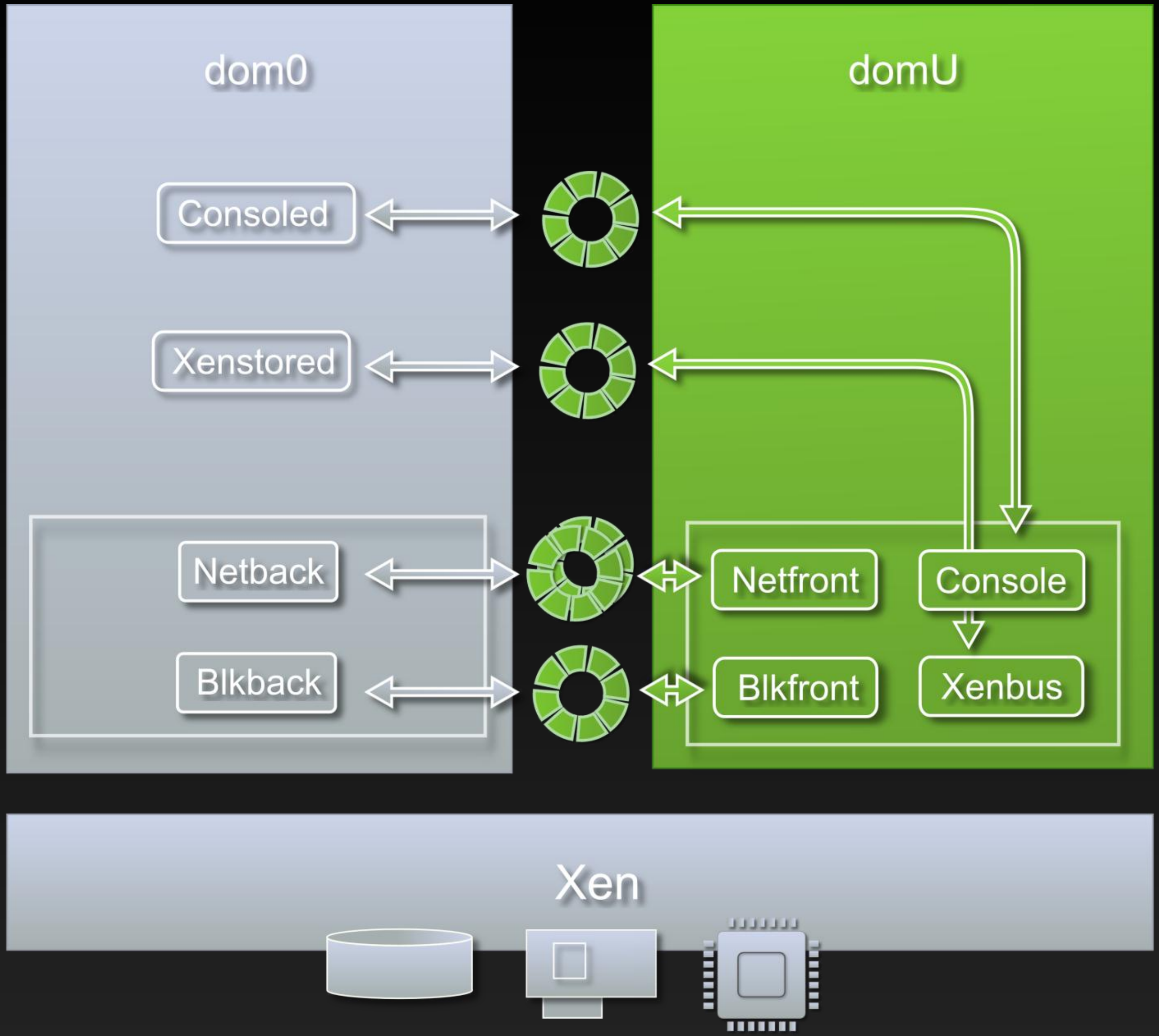
Control flow updates = registers + stack
(interrupts, events)

Some examples

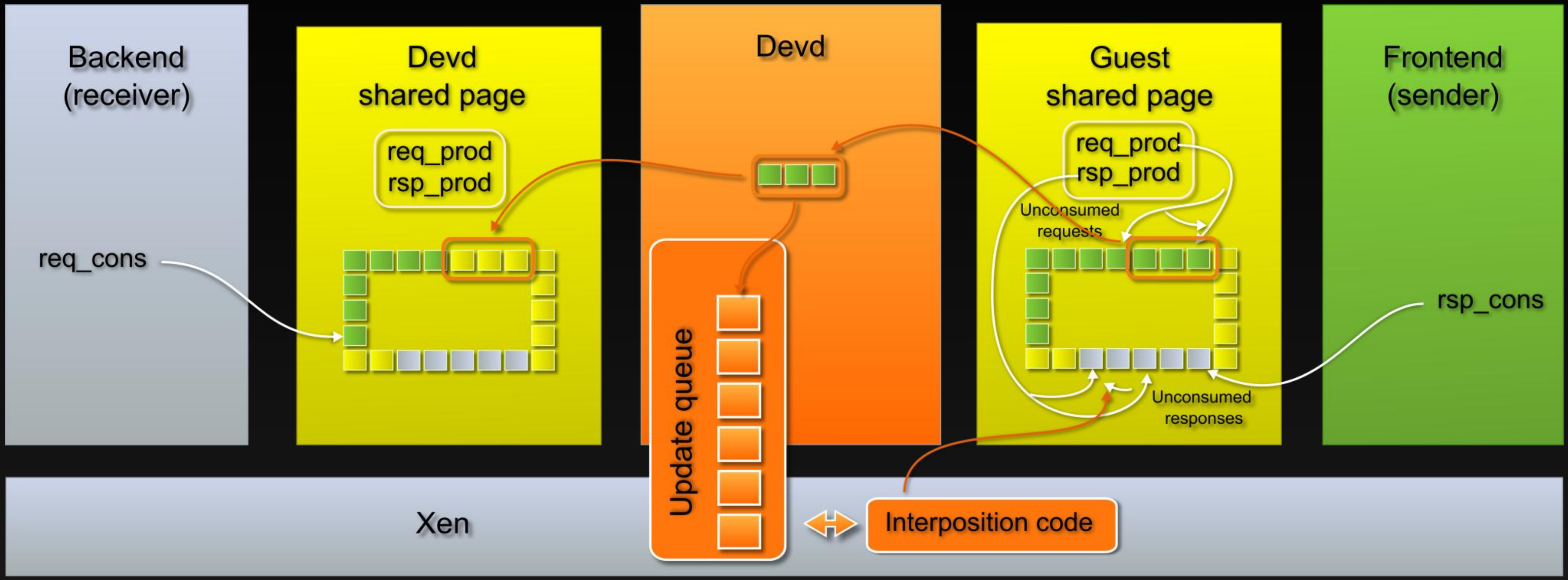
- Instruction emulation (e.g. cpuid, rdtsc, in/out)
  - Return the values of the original run
- Hypercalls
  - Re-execute to ensure determinism of the hypervisor
- Time
  - Shared info page + rdtsc
- Exceptions
  - Deterministic, re-execute
- Interrupts
  - Force re-execution of the interrupt frame (bounce frame) code in entry.S
- Shared info updates
  - Replay original values
- Memory
  - Shadow page tables


dom0
(Linux)
guest
(Linux)
Devd
Replay
Logging
Device Interposition
Backend
Frontend
Event Interposition
Logging/Replay Coordination
Xen

Xen devices


dom0
domU
Consoled
Xenstored
Netback
Blkback
Netfront
Console
Blkfront
Xenbus
Xen

Device interposition

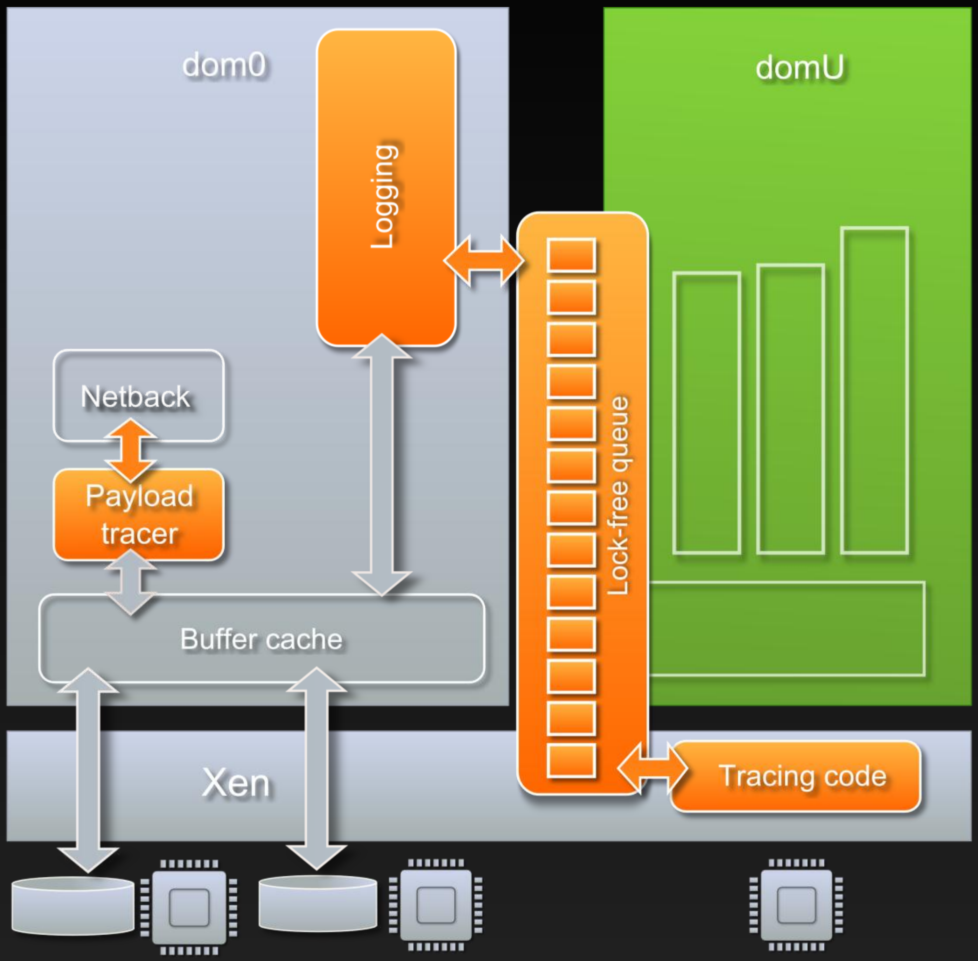


- Devd ensures determinism of updates to the guest’s shared ring buffers

Replay touches many parts of Xen

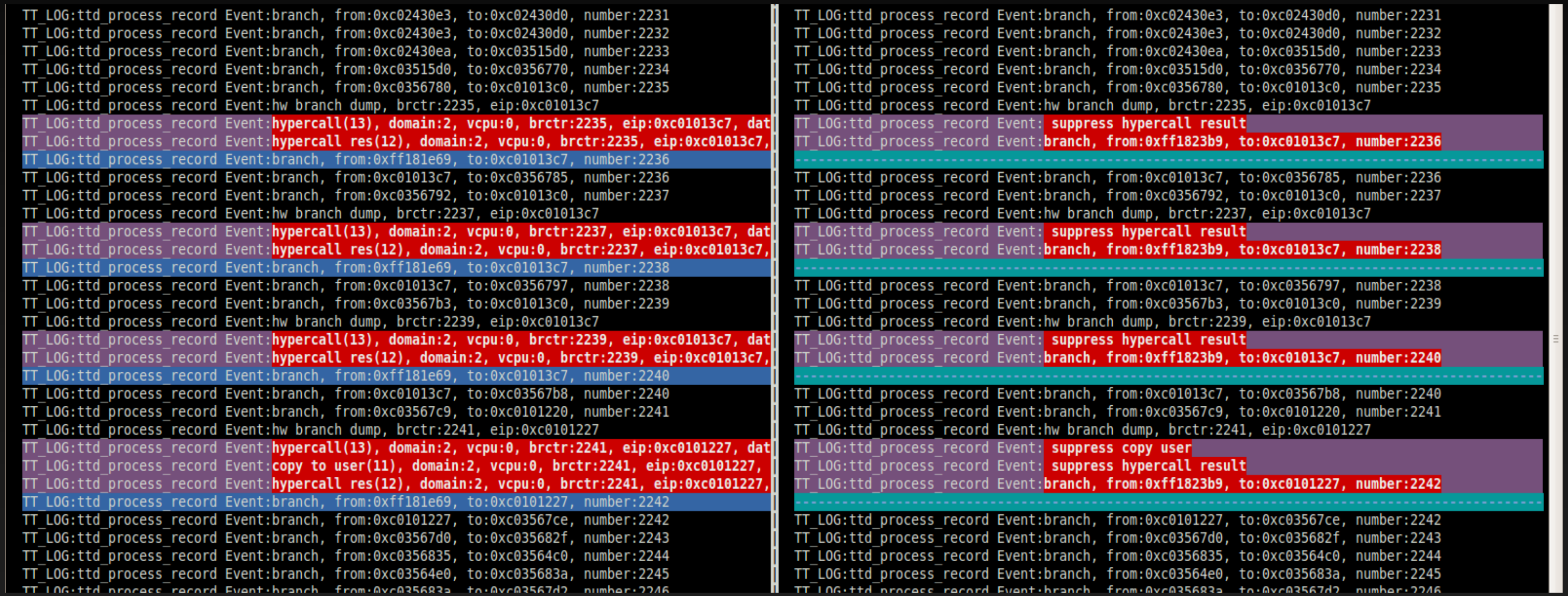
- Device discovery
  - Devd implements a concept of a device bus
  - Discovers new devices in Xenstore
  - Binds new devices with drivers
- Xenstore transactions
  - During replay, transactions from replayed guest can’t fail
  - They will not be re-executed
- Out-of-order device responses
  - Disk and network responses can arrive out-of-order
- Disk logging
  - Disk payload is deterministic
  - LVM snapshots
- Network logging
  - In-kernel logging of the network payload

Low-overhead logging



Are we sure that executions identical?

- Intel branch store trace facility
- Record all taken branches in a memory buffer
  - Compare original and replay runs

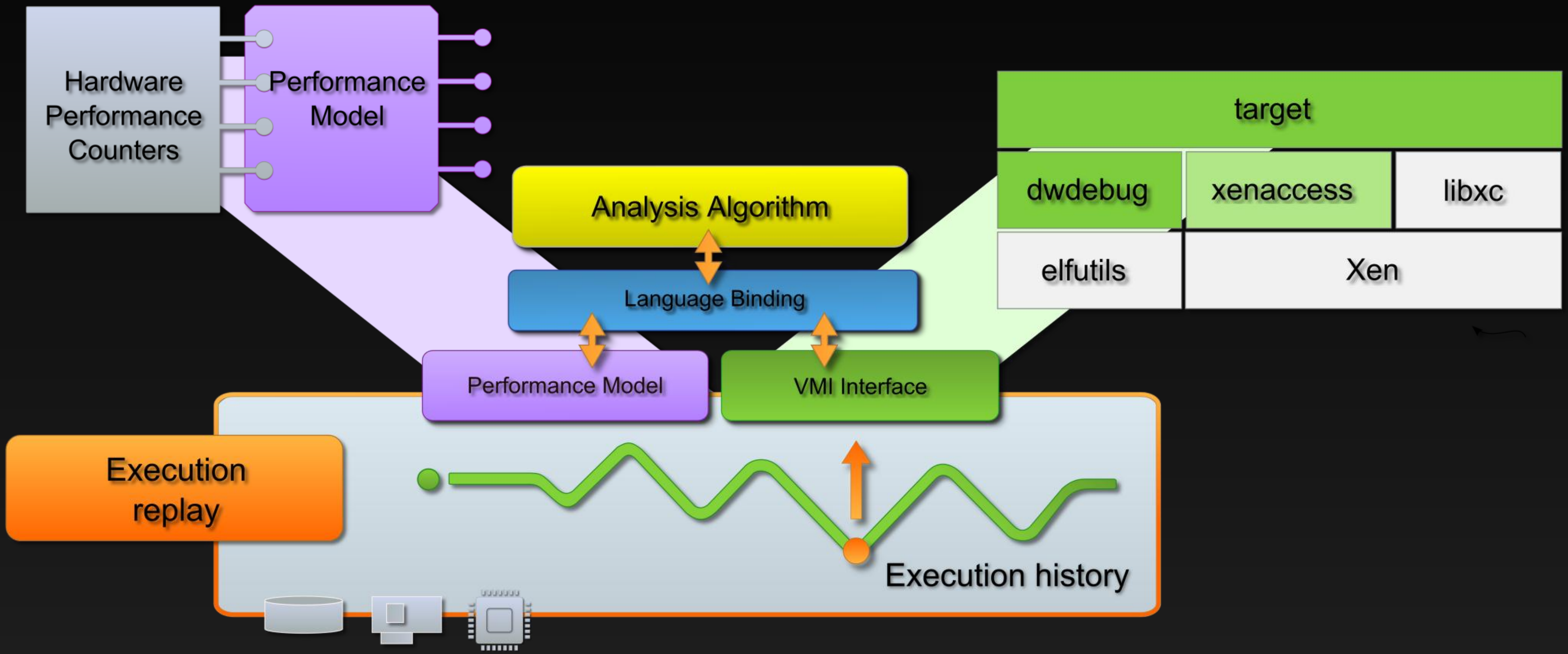


```
TT_LOG:ttd_process_record Event:branch, from:0xc02430e3, to:0xc02430d0, number:2231
TT_LOG:ttd_process_record Event:branch, from:0xc02430e3, to:0xc02430d0, number:2232
TT_LOG:ttd_process_record Event:branch, from:0xc02430ea, to:0xc03515d0, number:2233
TT_LOG:ttd_process_record Event:branch, from:0xc03515d0, to:0xc0356770, number:2234
TT_LOG:ttd_process_record Event:branch, from:0xc0356780, to:0xc01013c0, number:2235
TT_LOG:ttd_process_record Event:hw branch dump, brctr:2235, eip:0xc01013c7
TT_LOG:ttd_process_record Event:hypercall(13), domain:2, vcpu:0, brctr:2235, eip:0xc01013c7, dat
TT_LOG:ttd_process_record Event:hypercall res(12), domain:2, vcpu:0, brctr:2235, eip:0xc01013c7,
TT_LOG:ttd_process_record Event:branch, from:0xff181e69, to:0xc01013c7, number:2236
TT_LOG:ttd_process_record Event:branch, from:0xc01013c7, to:0xc0356785, number:2236
TT_LOG:ttd_process_record Event:branch, from:0xc0356792, to:0xc01013c0, number:2237
TT_LOG:ttd_process_record Event:hw branch dump, brctr:2237, eip:0xc01013c7
TT_LOG:ttd_process_record Event:hypercall(13), domain:2, vcpu:0, brctr:2237, eip:0xc01013c7, dat
TT_LOG:ttd_process_record Event:hypercall res(12), domain:2, vcpu:0, brctr:2237, eip:0xc01013c7,
TT_LOG:ttd_process_record Event:branch, from:0xff181e69, to:0xc01013c7, number:2238
TT_LOG:ttd_process_record Event:branch, from:0xc01013c7, to:0xc0356797, number:2238
TT_LOG:ttd_process_record Event:branch, from:0xc03567b3, to:0xc01013c0, number:2239
TT_LOG:ttd_process_record Event:hw branch dump, brctr:2239, eip:0xc01013c7
TT_LOG:ttd_process_record Event:hypercall(13), domain:2, vcpu:0, brctr:2239, eip:0xc01013c7, dat
TT_LOG:ttd_process_record Event:hypercall res(12), domain:2, vcpu:0, brctr:2239, eip:0xc01013c7,
TT_LOG:ttd_process_record Event:branch, from:0xff181e69, to:0xc01013c7, number:2240
TT_LOG:ttd_process_record Event:branch, from:0xc01013c7, to:0xc03567b8, number:2240
TT_LOG:ttd_process_record Event:branch, from:0xc03567c9, to:0xc0101220, number:2241
TT_LOG:ttd_process_record Event:hw branch dump, brctr:2241, eip:0xc0101227
TT_LOG:ttd_process_record Event:hypercall(13), domain:2, vcpu:0, brctr:2241, eip:0xc0101227, dat
TT_LOG:ttd_process_record Event:copy to user(11), domain:2, vcpu:0, brctr:2241, eip:0xc0101227,
TT_LOG:ttd_process_record Event:hypercall res(12), domain:2, vcpu:0, brctr:2241, eip:0xc0101227,
TT_LOG:ttd_process_record Event:branch, from:0xff181e69, to:0xc0101227, number:2242
TT_LOG:ttd_process_record Event:branch, from:0xc0101227, to:0xc03567ce, number:2242
TT_LOG:ttd_process_record Event:branch, from:0xc03567d0, to:0xc035682f, number:2243
TT_LOG:ttd_process_record Event:branch, from:0xc0356835, to:0xc03564c0, number:2244
TT_LOG:ttd_process_record Event:branch, from:0xc03564e0, to:0xc035683a, number:2245
TT_LOG:ttd_process_record Event:branch, from:0xc035683a, to:0xc03567d2, number:2246
```

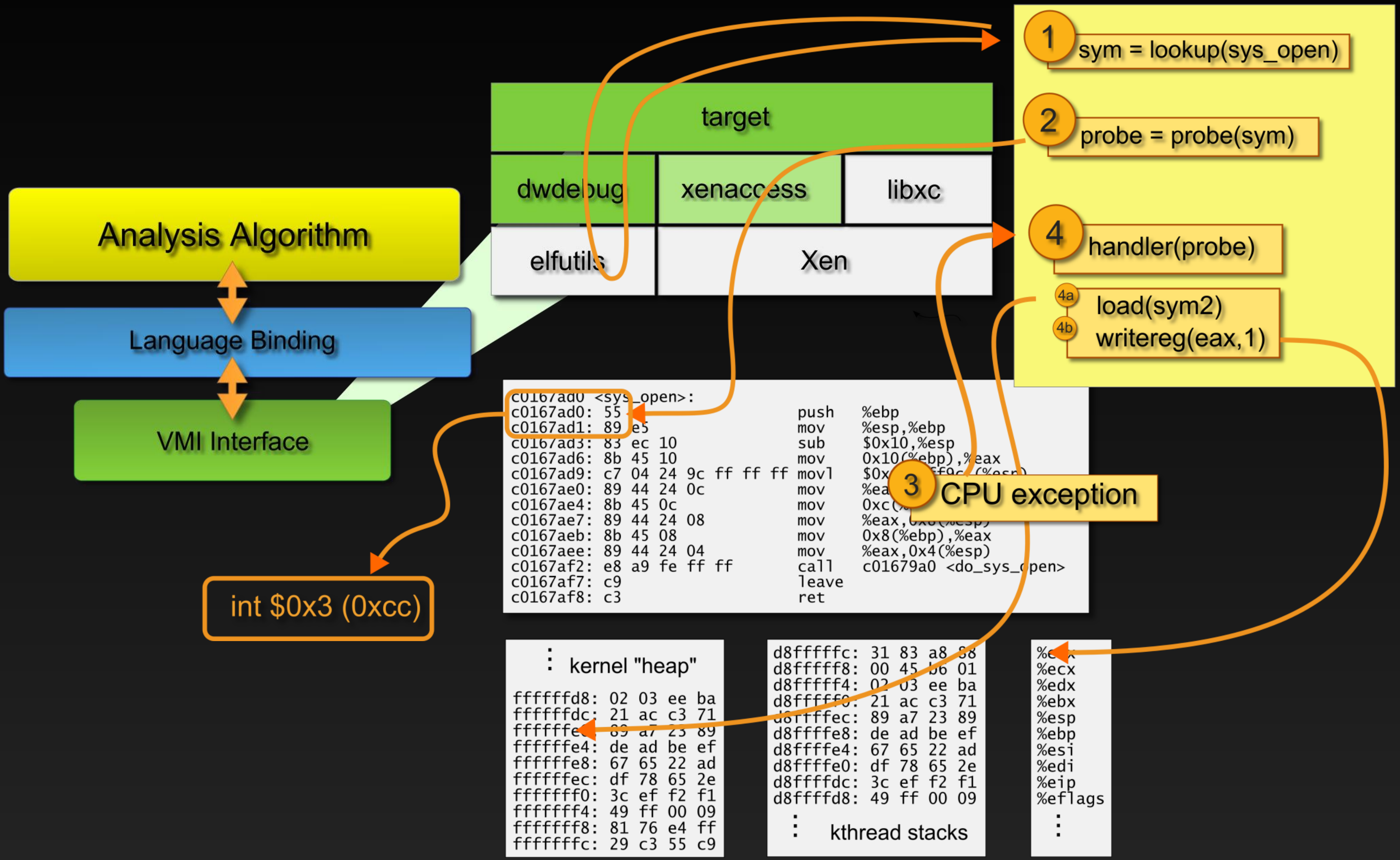
```
TT_LOG:ttd_process_record Event:branch, from:0xc02430e3, to:0xc02430d0, number:2231
TT_LOG:ttd_process_record Event:branch, from:0xc02430e3, to:0xc02430d0, number:2232
TT_LOG:ttd_process_record Event:branch, from:0xc02430ea, to:0xc03515d0, number:2233
TT_LOG:ttd_process_record Event:branch, from:0xc03515d0, to:0xc0356770, number:2234
TT_LOG:ttd_process_record Event:branch, from:0xc0356780, to:0xc01013c0, number:2235
TT_LOG:ttd_process_record Event:hw branch dump, brctr:2235, eip:0xc01013c7
TT_LOG:ttd_process_record Event: suppress hypercall result
TT_LOG:ttd_process_record Event:branch, from:0xff1823b9, to:0xc01013c7, number:2236
-----------------------------------------------------------------------------------------
TT_LOG:ttd_process_record Event:branch, from:0xc01013c7, to:0xc0356785, number:2236
TT_LOG:ttd_process_record Event:branch, from:0xc0356792, to:0xc01013c0, number:2237
TT_LOG:ttd_process_record Event:hw branch dump, brctr:2237, eip:0xc01013c7
TT_LOG:ttd_process_record Event: suppress hypercall result
TT_LOG:ttd_process_record Event:branch, from:0xff1823b9, to:0xc01013c7, number:2238
-----------------------------------------------------------------------------------------
TT_LOG:ttd_process_record Event:branch, from:0xc01013c7, to:0xc0356797, number:2238
TT_LOG:ttd_process_record Event:branch, from:0xc03567b3, to:0xc01013c0, number:2239
TT_LOG:ttd_process_record Event:hw branch dump, brctr:2239, eip:0xc01013c7
TT_LOG:ttd_process_record Event: suppress hypercall result
TT_LOG:ttd_process_record Event:branch, from:0xff1823b9, to:0xc01013c7, number:2240
-----------------------------------------------------------------------------------------
TT_LOG:ttd_process_record Event:branch, from:0xc01013c7, to:0xc03567b8, number:2240
TT_LOG:ttd_process_record Event:branch, from:0xc03567c9, to:0xc0101220, number:2241
TT_LOG:ttd_process_record Event:hw branch dump, brctr:2241, eip:0xc0101227
TT_LOG:ttd_process_record Event: suppress copy user
TT_LOG:ttd_process_record Event: suppress hypercall result
TT_LOG:ttd_process_record Event:branch, from:0xff1823b9, to:0xc0101227, number:2242
-----------------------------------------------------------------------------------------
TT_LOG:ttd_process_record Event:branch, from:0xc0101227, to:0xc03567ce, number:2242
TT_LOG:ttd_process_record Event:branch, from:0xc03567d0, to:0xc035682f, number:2243
TT_LOG:ttd_process_record Event:branch, from:0xc0356835, to:0xc03564c0, number:2244
TT_LOG:ttd_process_record Event:branch, from:0xc03564e0, to:0xc035683a, number:2245
TT_LOG:ttd_process_record Event:branch, from:0xc035683a, to:0xc03567d2, number:2246
```

Analysis Engine and Virtual Machine Introspection

Analysis framework



Virtual Machine Introspection (VMI)



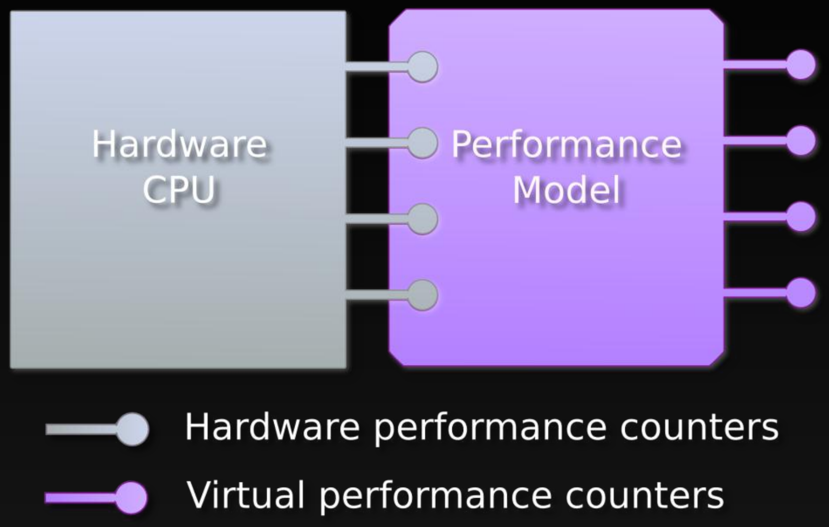
```
c0167ad0 <sys_open>:
c0167ad0: 55                      push   %ebp
c0167ad1: 89 e5                   mov    %esp,%ebp
c0167ad3: 83 ec 10                sub    $0x10,%esp
c0167ad6: 8b 45 10                mov    0x10(%ebp),%eax
c0167ad9: c7 04 24 9c ff ff ff    movl   $0xffffff9c,(%esp)
c0167ae0: 89 44 24 0c             mov    %eax,0xc(%esp)
c0167ae4: 8b 45 0c                mov    0xc(%ebp),%eax
c0167ae7: 89 44 24 08             mov    %eax,0x8(%esp)
c0167aeb: 8b 45 08                mov    0x8(%ebp),%eax
c0167aee: 89 44 24 04             mov    %eax,0x4(%esp)
c0167af2: e8 a9 fe ff ff          call   c01679a0 <do_sys_open>
c0167af7: c9                      leave
c0167af8: c3                      ret
```

```
  ⋮
ffffffd8: 02 03 ee ba
ffffffdc: 21 ac c3 71
ffffffe0: 89 a7 23 89
ffffffe4: de ad be ef
ffffffe8: 67 65 22 ad
ffffffec: df 78 65 2e
fffffff0: 3c ef f2 f1
fffffff4: 49 ff 00 09
fffffff8: 81 76 e4 ff
fffffffc: 29 c3 55 c9
```

```
d8fffffc: 31 83 a8 88
d8fffff8: 00 45 b6 01
d8fffff4: 02 03 ee ba
d8fffff0: 21 ac c3 71
d8ffffec: 89 a7 23 89
d8ffffe8: de ad be ef
d8ffffe4: 67 65 22 ad
d8ffffe0: df 78 65 2e
d8ffffdc: 3c ef f2 f1
d8ffffd8: 49 ff 00 09
  ⋮
```

```
%eax
%ecx
%edx
%ebx
%esp
%ebp
%esi
%edi
%eip
%eflags
  ⋮
```

Performance model

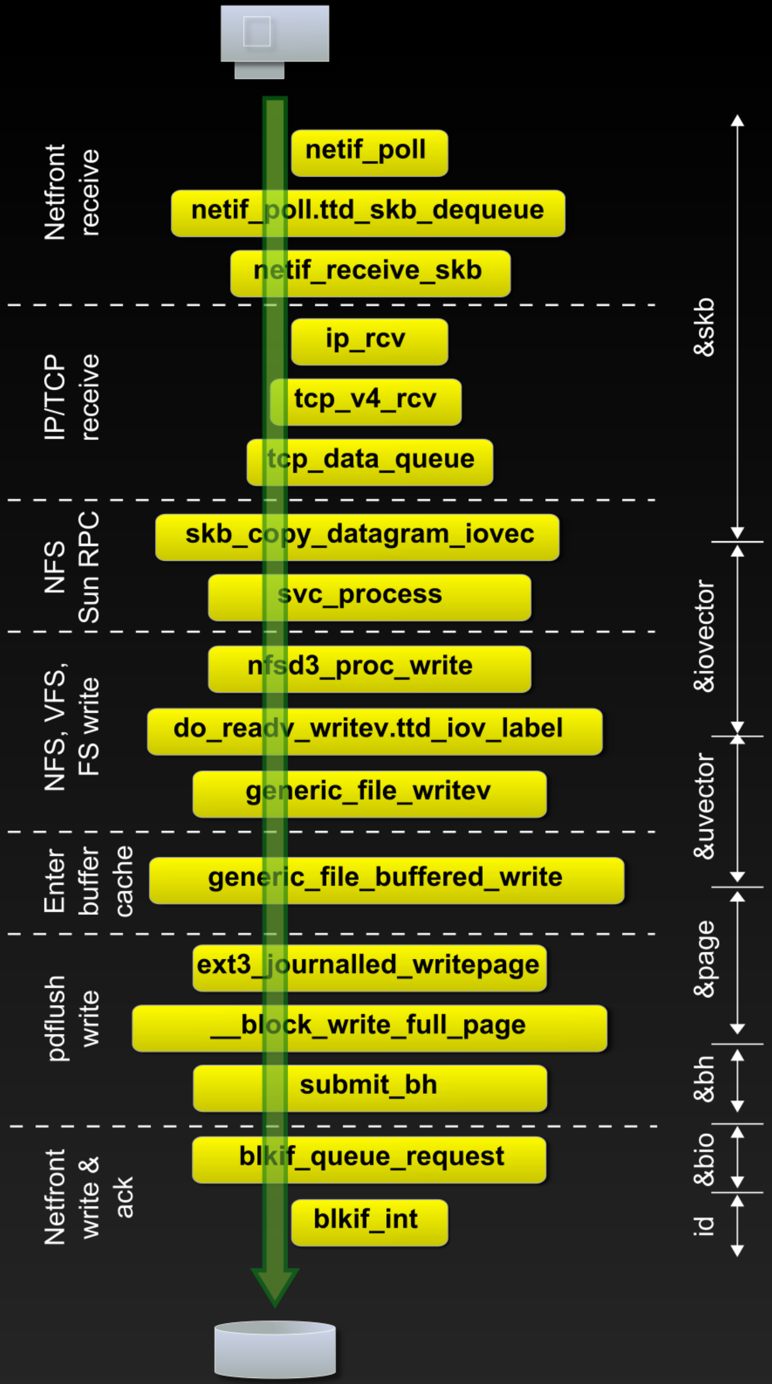


- Account for effects of replay
- Translate performance between original and replay runs

- Re-execution approach to performance

$$\boxed{\text{Virt cntr} = \text{Virt cntr}_{\text{start}} + \Delta\,(\text{Real cntr})}$$

Analysis Examples



NFS request processing path

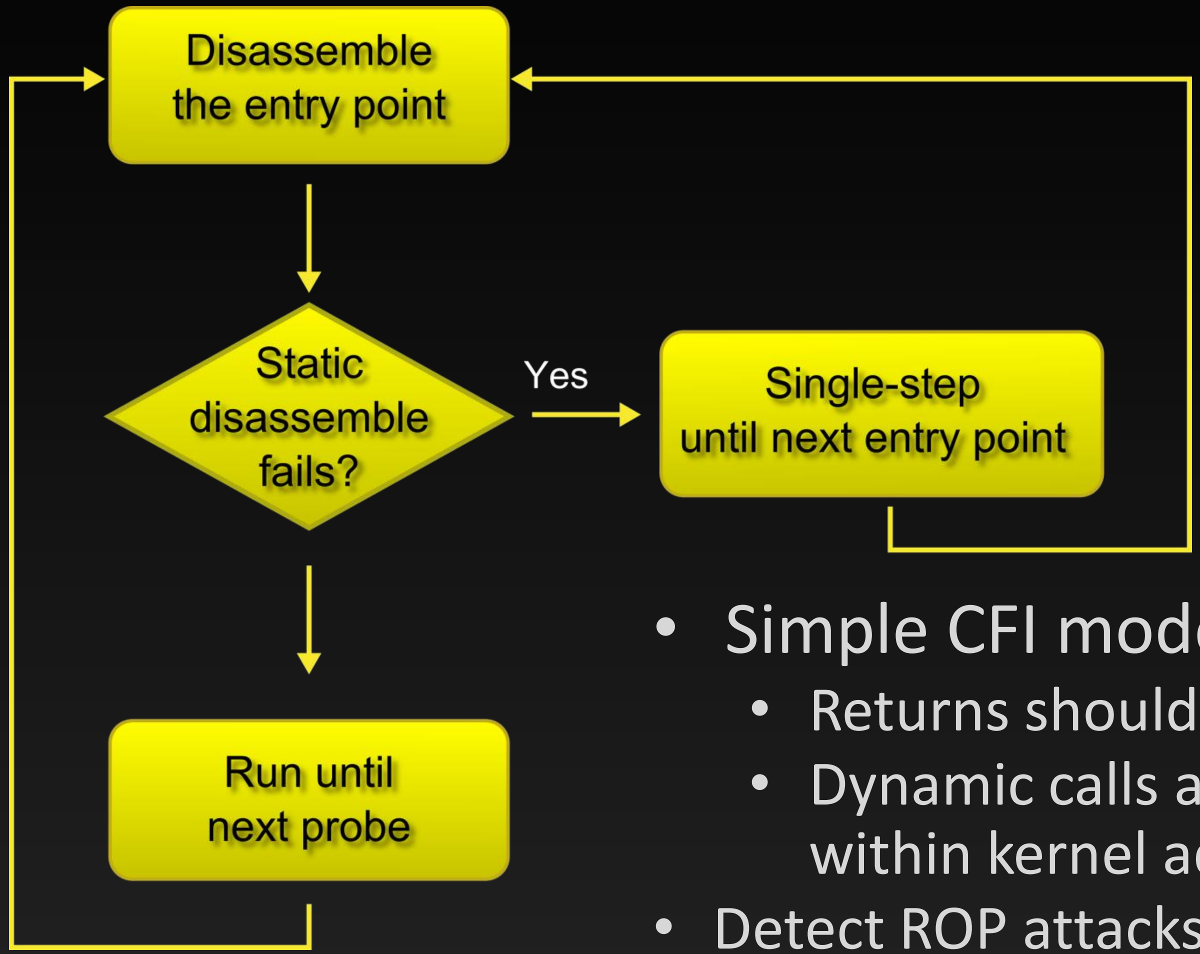
- How much time requests spend in each subsystem?

- Request tracking
  - Address of the kernel data structure is a unique identifier
  - Join identifiers when requests move between subsystems

Execution context tracking

- Execution context
    - Context switches
        - `schedule.switch_tasks`
    - User/kernel
        - System call transitions
        - `system_call`
    - Interrupts and exceptions
        - `do_IRQ`
        - `do_pagefault`
        - `do_* (divide_error, debug, nmi, int3...)`
- Make analysis context aware
    - Filter probes by context, e.g.
    - All pagefaults from the process “foo”

Control Flow Integrity (CFI)


Disassemble the entry point
Static disassemble fails?
Yes
Single-step until next entry point
Run until next probe

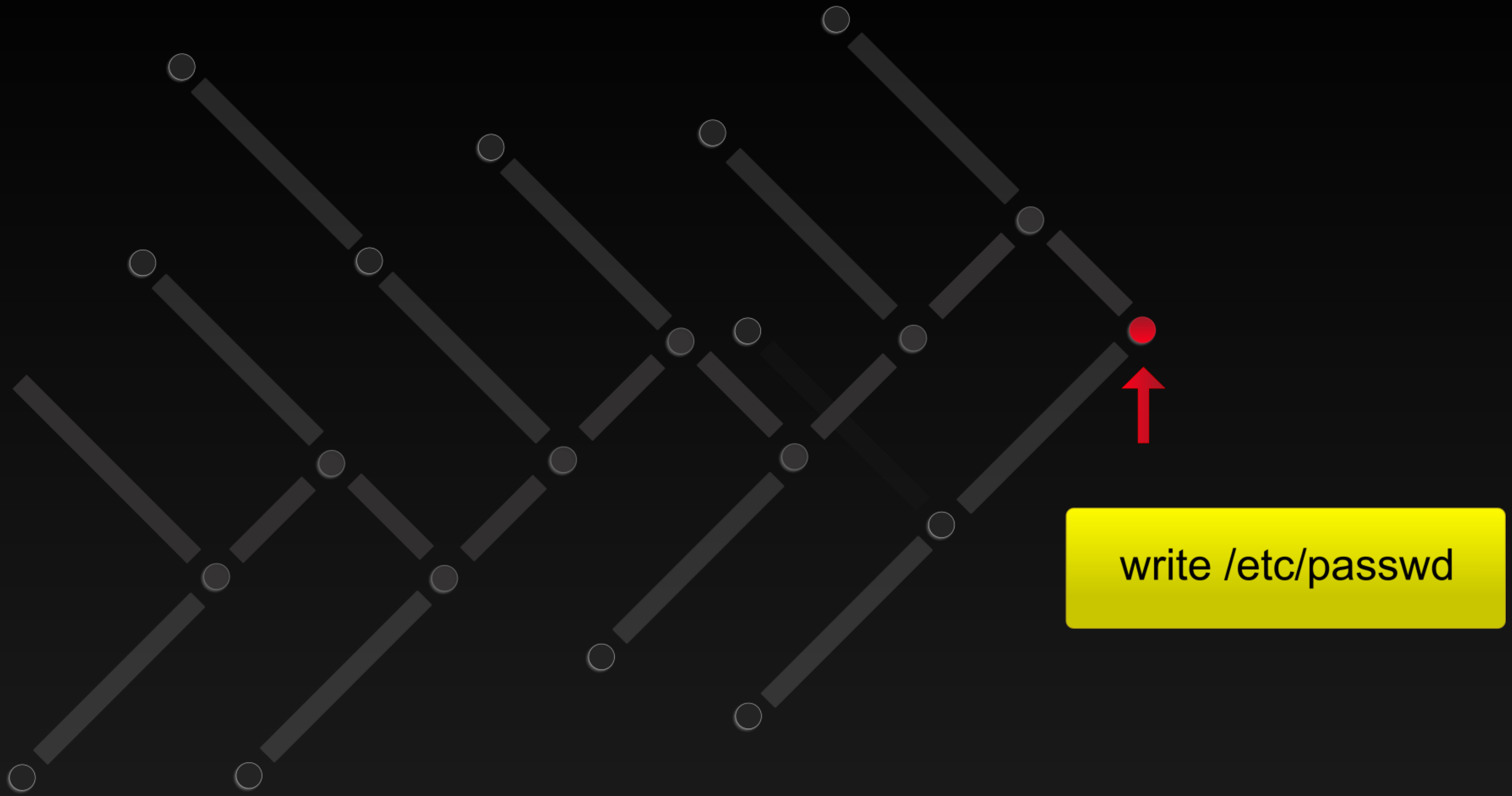
- Simple CFI model:
  - Returns should match calls
  - Dynamic calls are "sane", e.g. within kernel address space
- Detect ROP attacks, stack smashing, etc.

Execution trace

```
sys_sendfile
  do_sendfile
    fget_light
    rw_verify_area
    fget_light
    rw_verify_area
    shmem_file_sendfile
      do_shmem_file_read
        shmem_getpage
          find_lock_page
            radix_tree_lookup
          shmem_recalc_inode
          shmem_swp_alloc
            shmem_swp_entry
              kmap_atomic
                __kmap_atomic
                  page_address
          kunmap_atomic
          find_get_page
            radix_tree_lookup
        file_send_actor
          sock_sendpage
            UNKNOWN FUNCTION
            (addr:0x00000000)
```

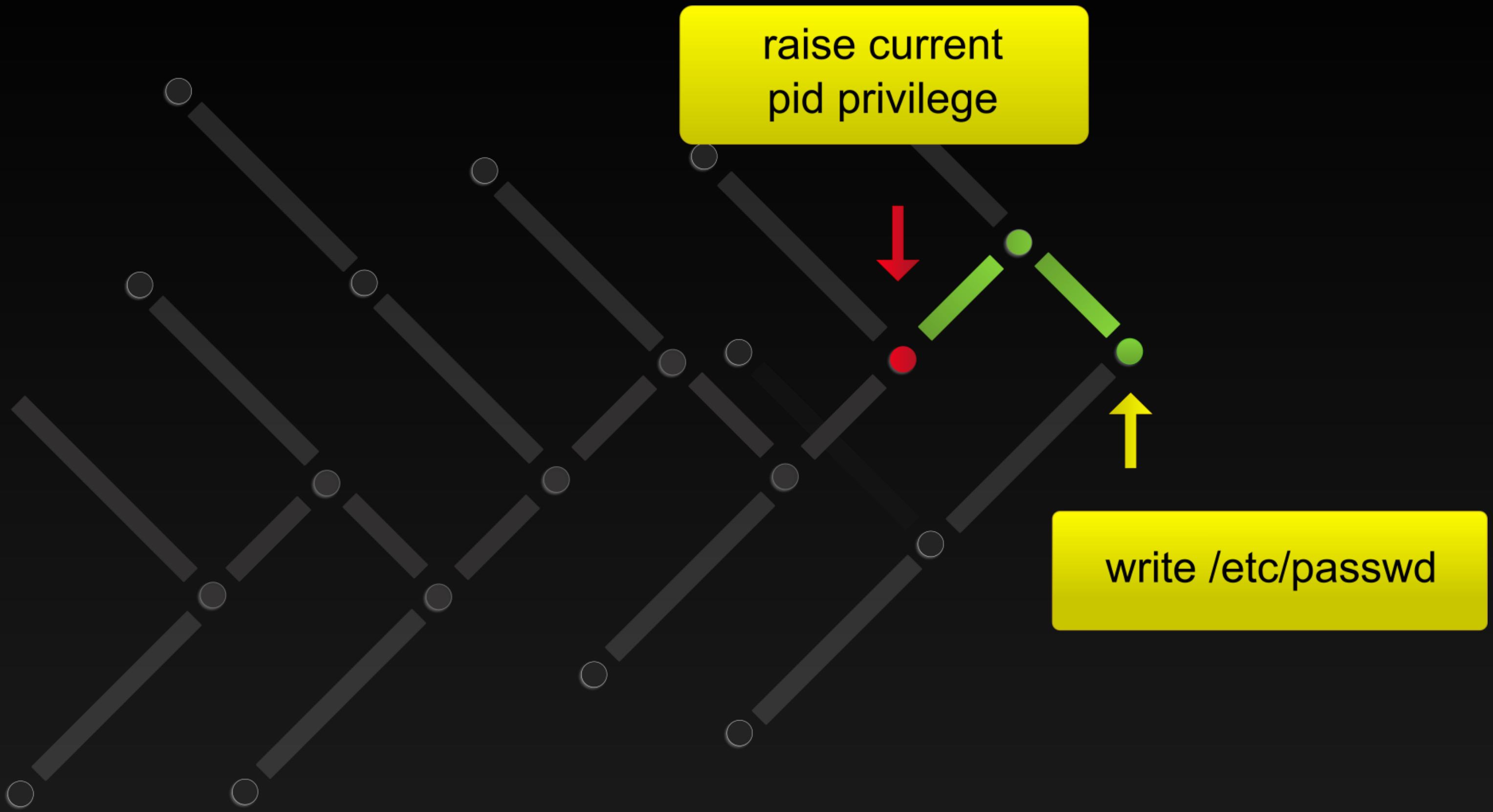
- CFI records a trace of function calls
  - `sys_sendfile` is the last system call before control flow jumps to 0x0

Intrusion backtracking



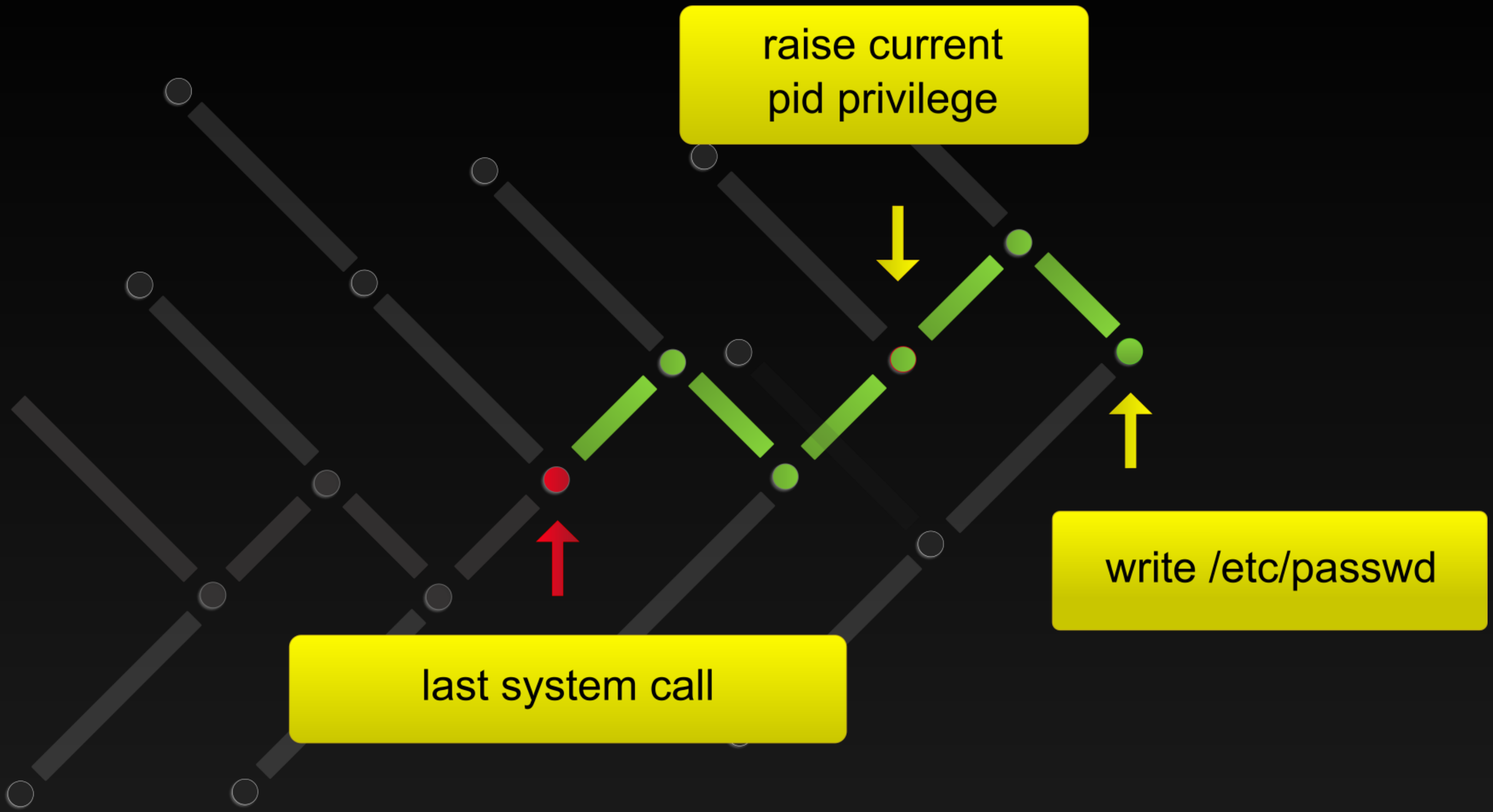
- Track accesses to “/etc/passwd”
- Probe sys_open
  - Filter by file name
  - Find process ID, branch counter

Intrusion backtracking: pass 1



- Process or it's parent escalated privileges
- Watch write accesses to &task->uid
  - Filter by parents of the offending process

Intrusion backtracking: pass 1



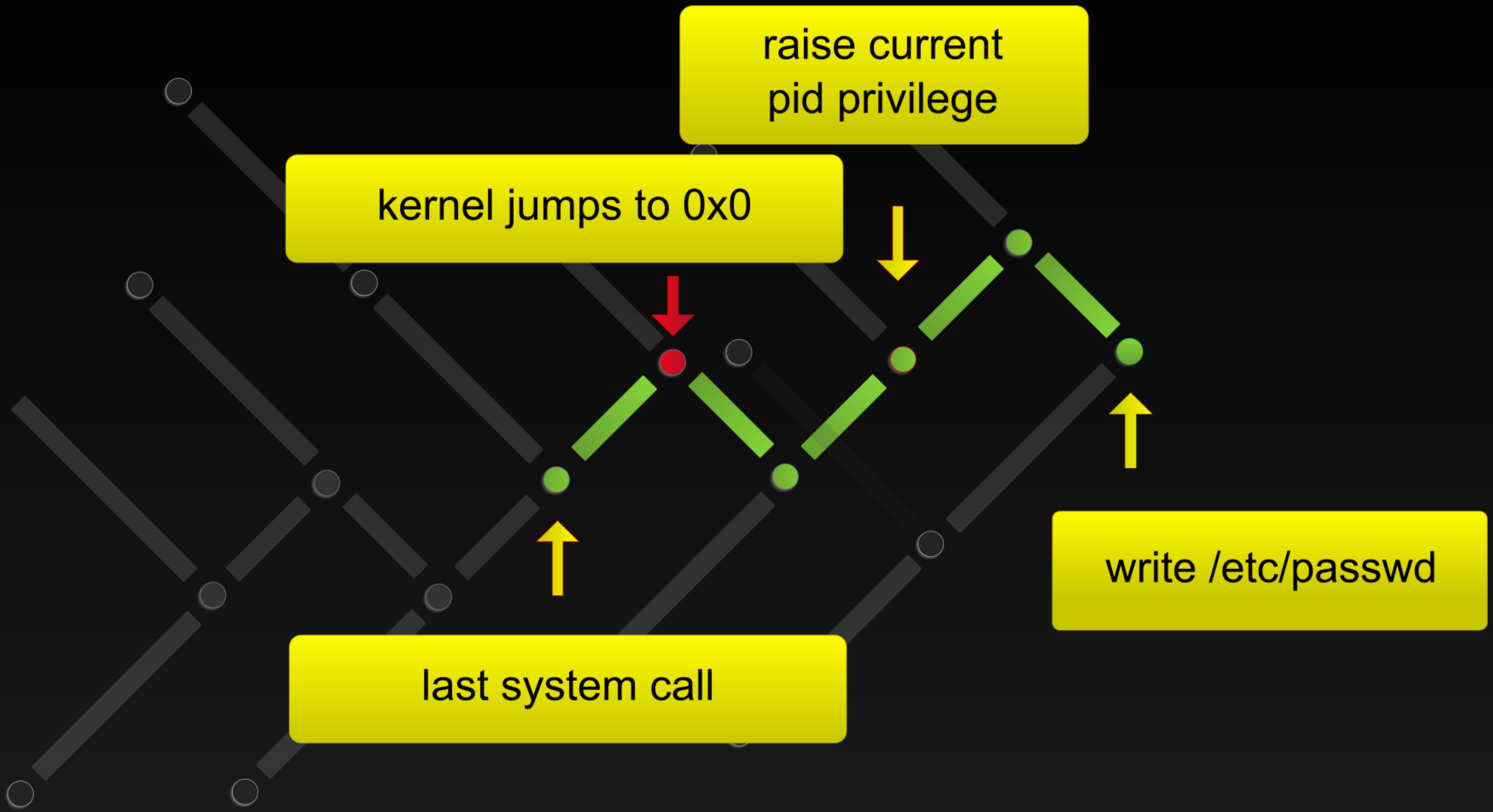
raise current pid privilege

write /etc/passwd

last system call

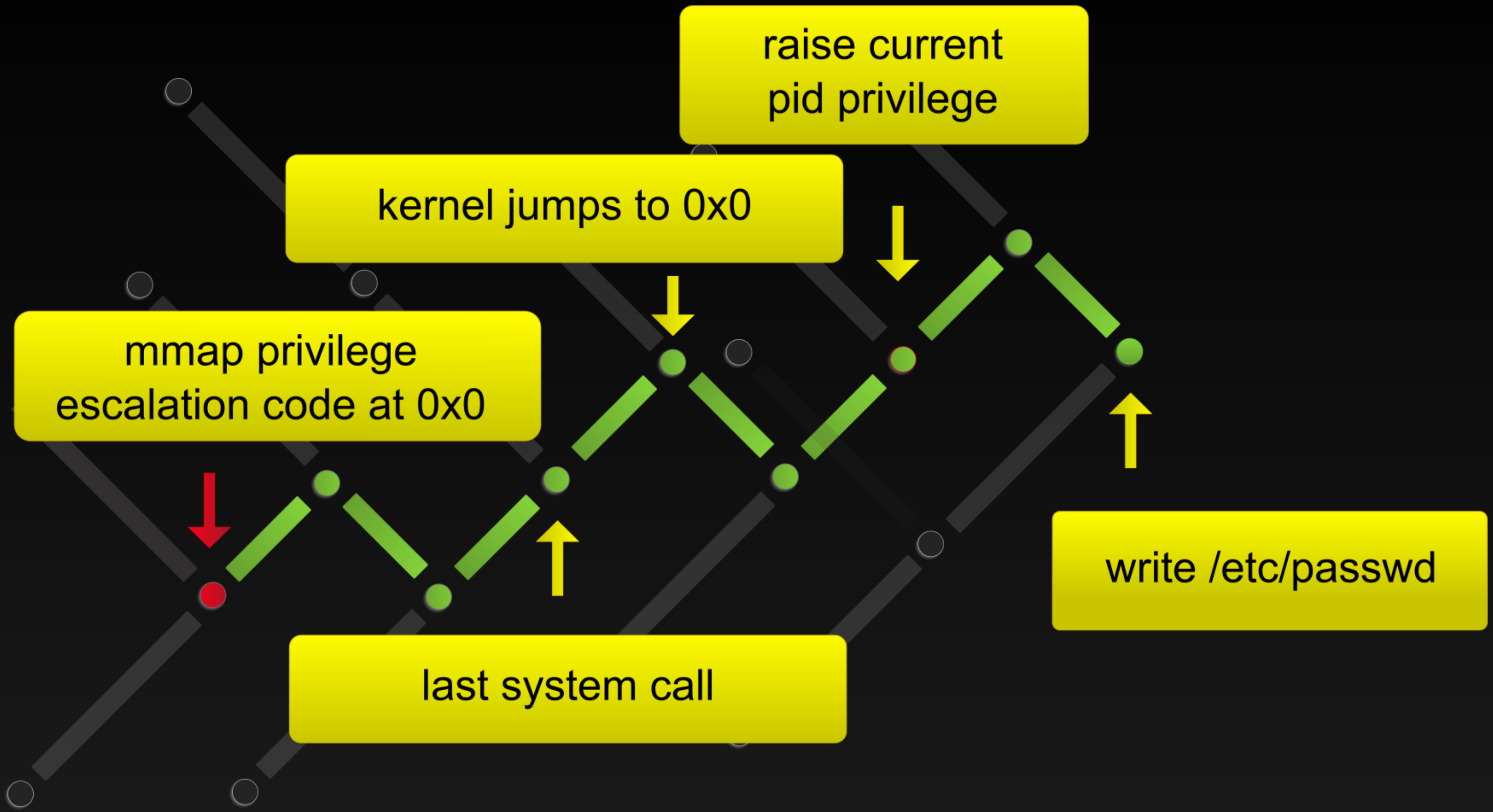
- Find the syscall inside which privileges are escalated
- Probe sys_* - all system call entry and exit points
  - Filter by the offending process ID

Intrusion backtracking: pass 1



- Privilege escalation is a CFI violation
- Start CFI analysis from the last system call
- Find %EIP at which CFI is violated, and location of the shell code (0x0)

Intrusion backtracking: pass 1



raise current pid privilege

kernel jumps to 0x0

mmap privilege escalation code at 0x0

write /etc/passwd

last system call

- Find at which point address 0x0000000 gets mapped
- Probe do_page_fault

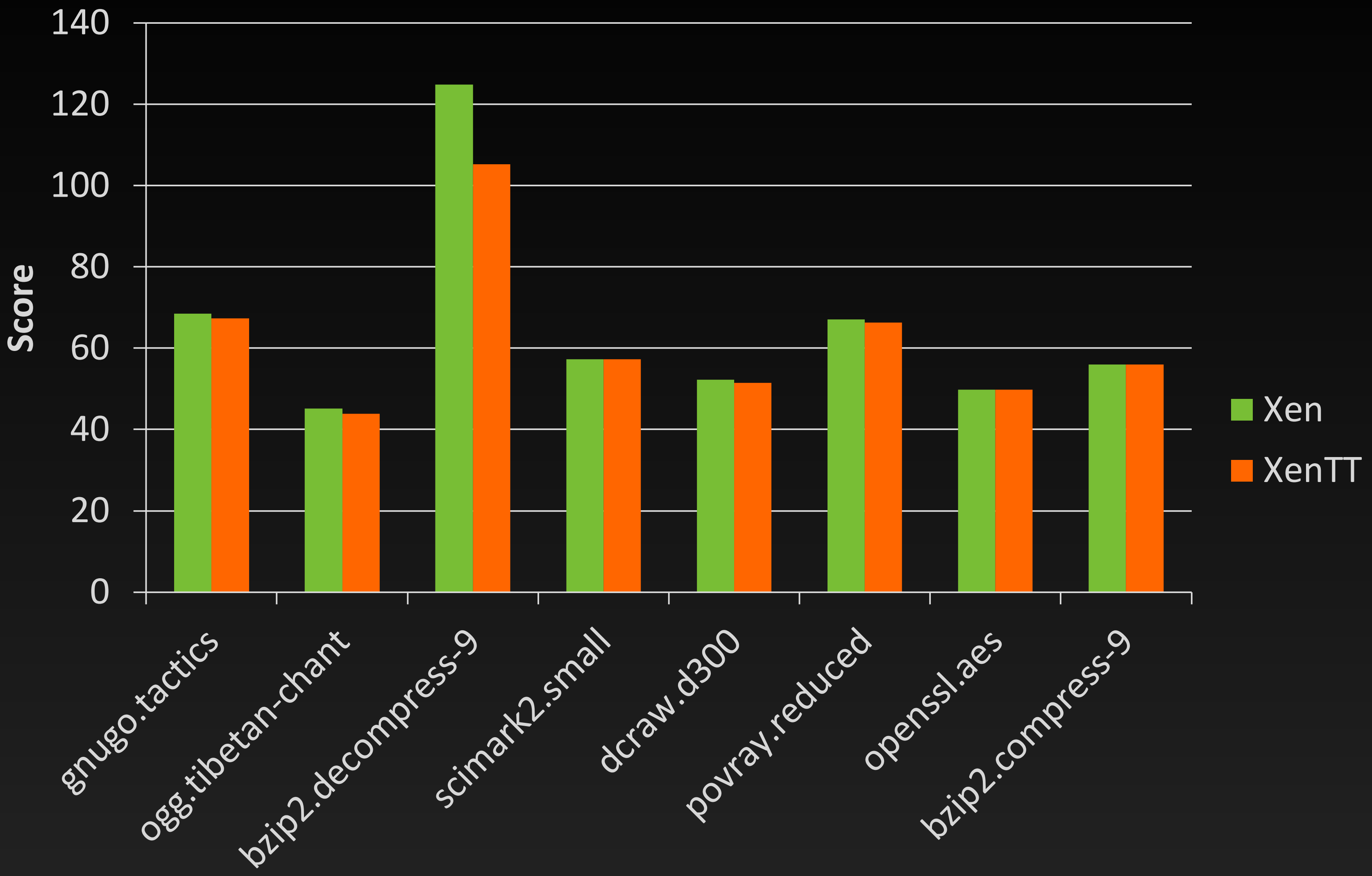
More mechanisms

- Execution traces
  - BTS trace of all taken branches
  - Instruction traces
- Memory (variable) access traces
  - Intersect with the execution trace
  - See where variables get accessed

How much overhead?

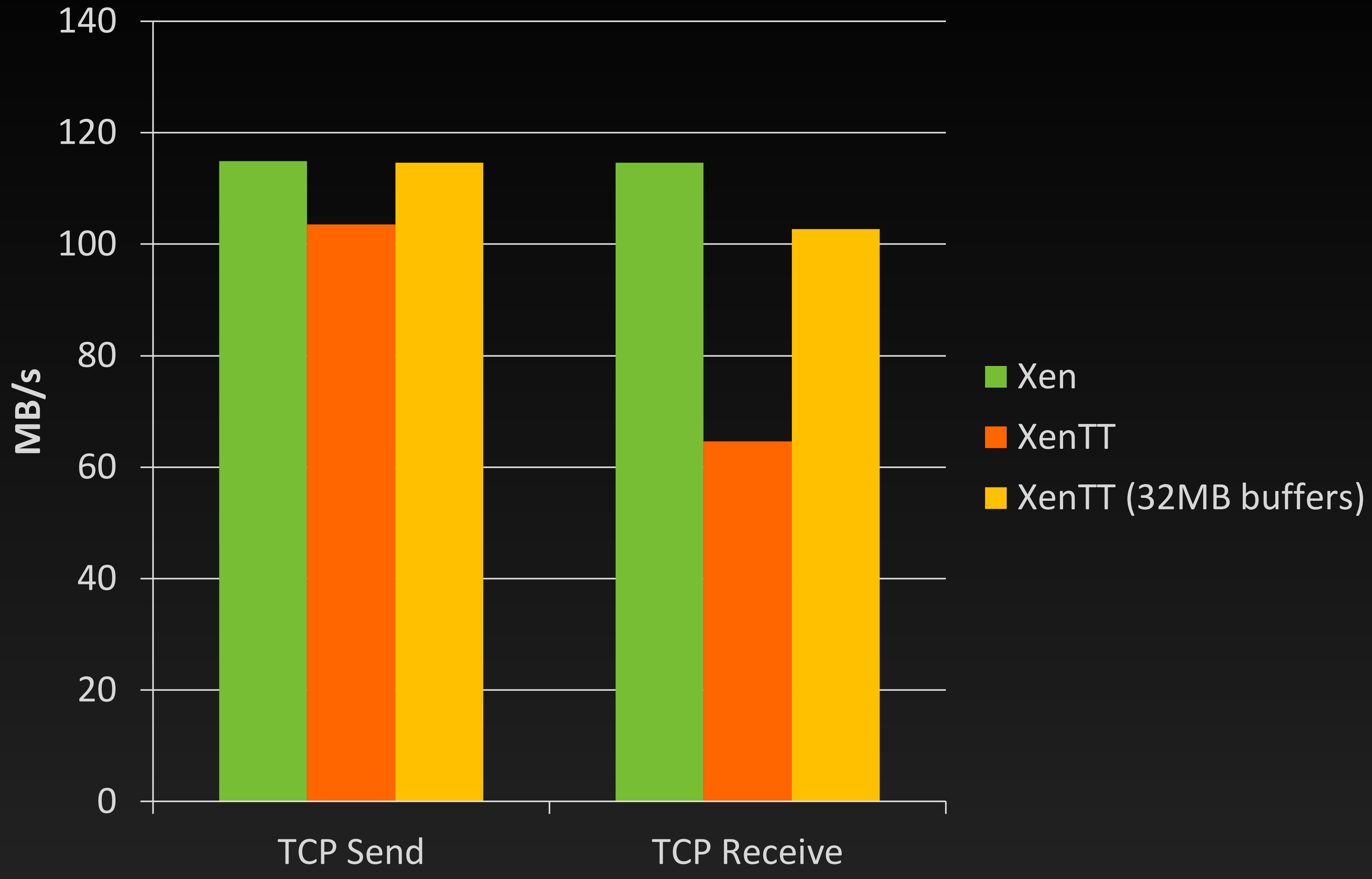
- 32-bit x86 PV-guests
- xen-unstable near v3.0.4
  - We rely on a working shadow page tables
- 1-CPU time-traveling guests
  - No SMP replay
  - Dom0 and Xen are SMP of course
- Test machine
  - 4 cores
  - 1Gbps network
  - 130 MB/s disks

CPU

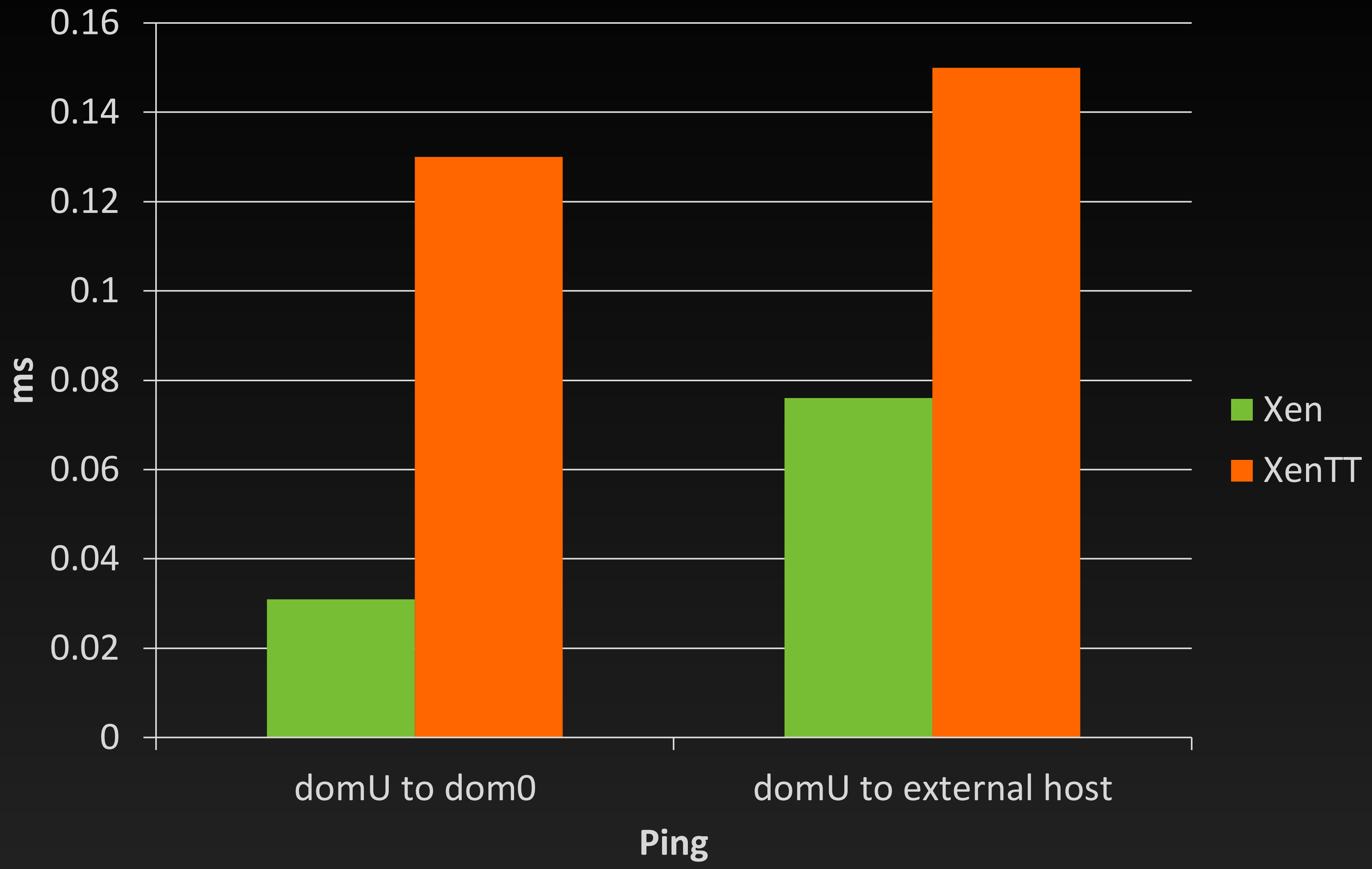


Score
140
120
100
80
60
40
20
0
gnugo.tactics
ogg.tibetan-chant
bzip2.decompress-9
scimark2.small
dcraw.d300
povray.reduced
openssl.aes
bzip2.compress-9
🟩 Xen
🟧 XenTT

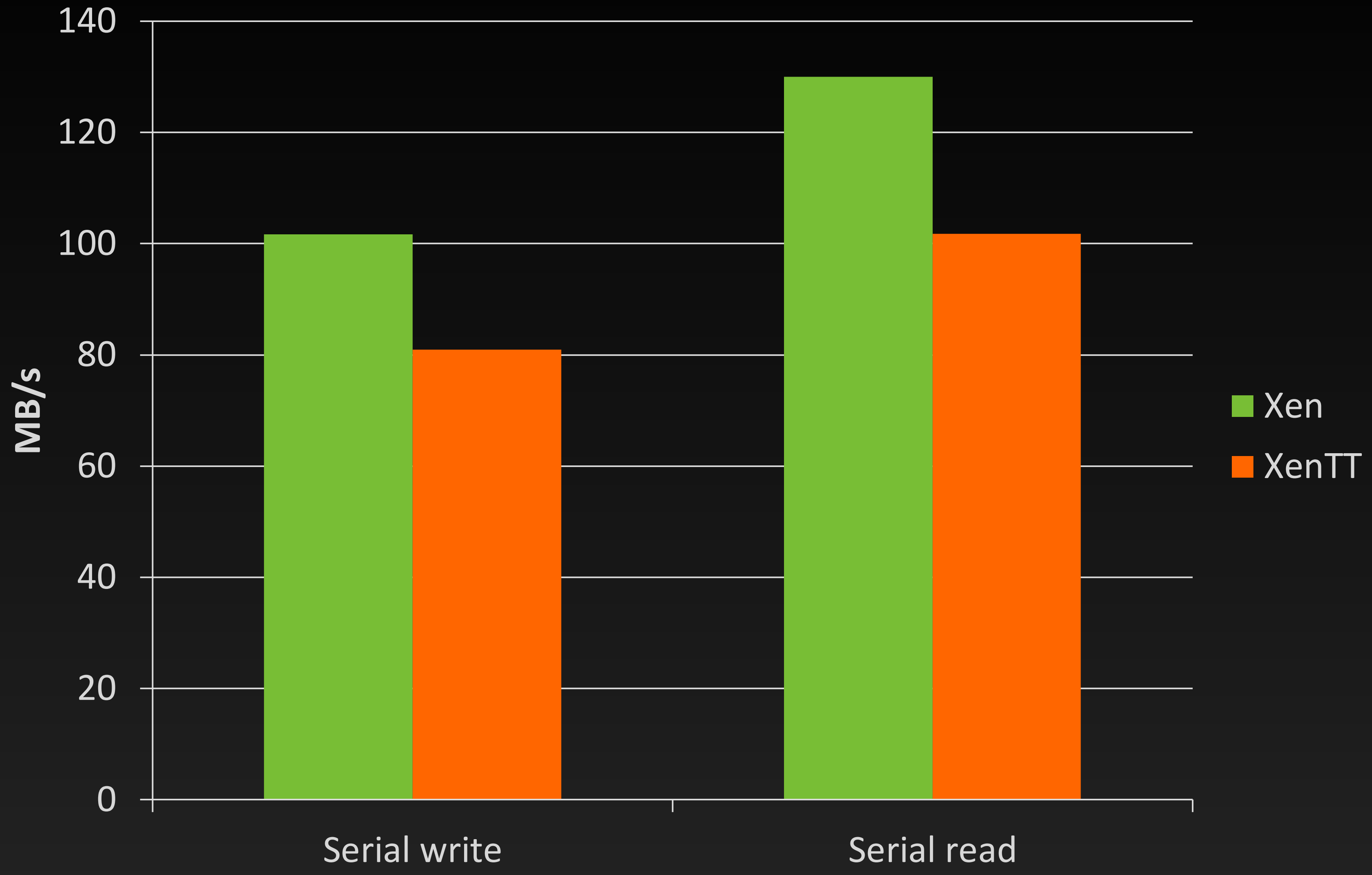
Network throughput



Network delay



Disk I/O


140
120
100
80
60
40
20
0
MB/s
Serial write
Serial read
🟩 Xen
🟧 XenTT

	Raw	Compressed (gzip)
Linux boot		
Event log	4.3 GB	0.8 GB
Idle overnight (14 hours)		
Event log	6.2 GB	1.6 GB
Growth rate	440 MB/hour	114 MB/hour (2.7GB/day)
TCP receive (1.63 GB data stream)		
Event log	1.76 GB	342 MB
Payload log	1.79 GB	Payload dependent
Disk write (4 GB file)		
Event log	1.8 GB	350 MB
Disk read (4 GB file)		
Event log	0.59 GB	

Lessons learnt

Scaling development

- Extending Xen with BTS support
  - Debug crashes in Xen, and dom0

- Execution comparison tools
  - BTS traces to understand what went wrong
  - Support for resolving symbols

- Run-time comparison tools
  - Compare guest's state between original and replay runs

- Trace from all parts of your system
  - Xen, dom0, domU

- Support performance tracing
  - Xentrace messages

What we didn’t predict

- I/O delay goes up
  - Not sure if Linux has adequate low-latency user-level processing support
  - Maybe need an in-kernel interposition component
- Branch counters are fragile
  - Our code works on several server CPUs
  - Fails on a laptop with the CPU from the same model/family line

Conclusions

Practical replay analysis is feasible

- Performance overheads are reasonable
  - Realistic systems
  - Realistic workloads
  - Minor setup costs (just install Xen)
- Analysis engine is an amazing tool
  - And we’re growing it
- We need your help to port it upstream
  - Porting effort
  - Shadow memory for PV guests
  - Support for HVM guests

Thank you.

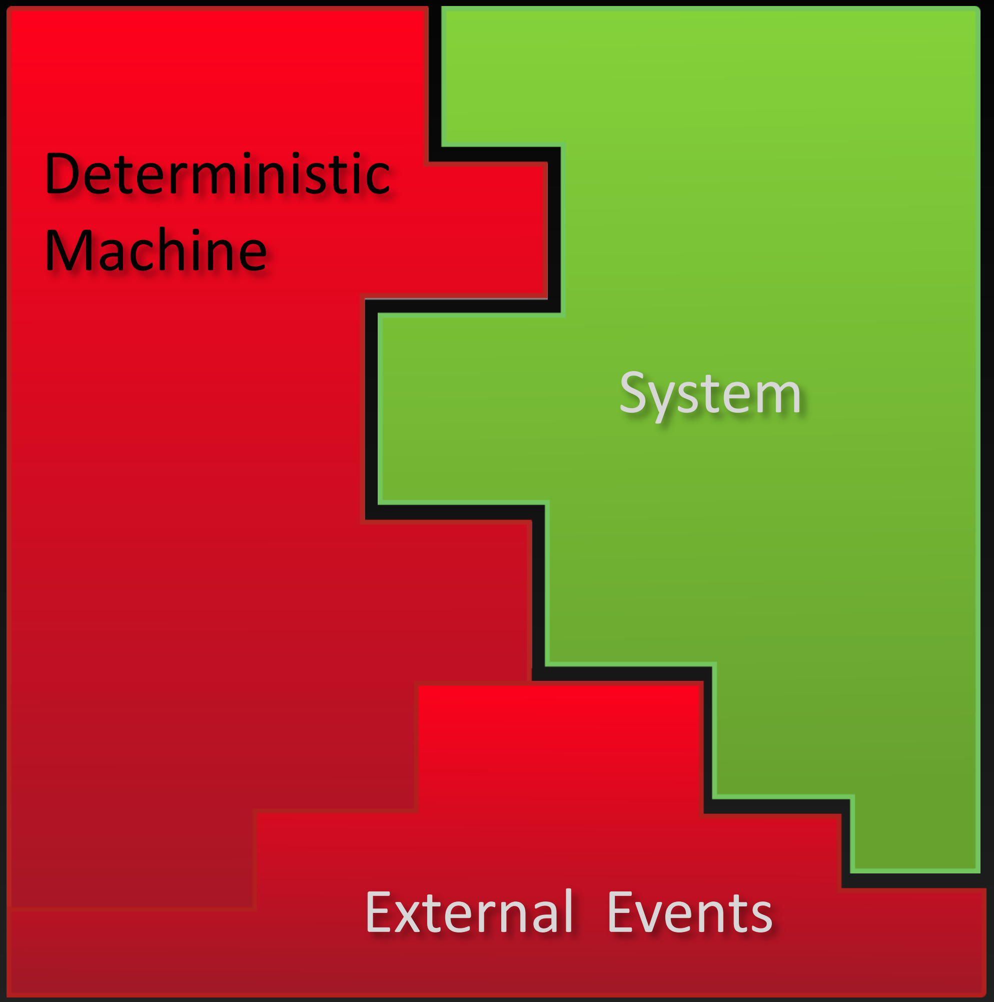
All code is GPLv2 and will be available soon (available on-request now).

Questions: aburtsev@flux.utah.edu

Backup slides

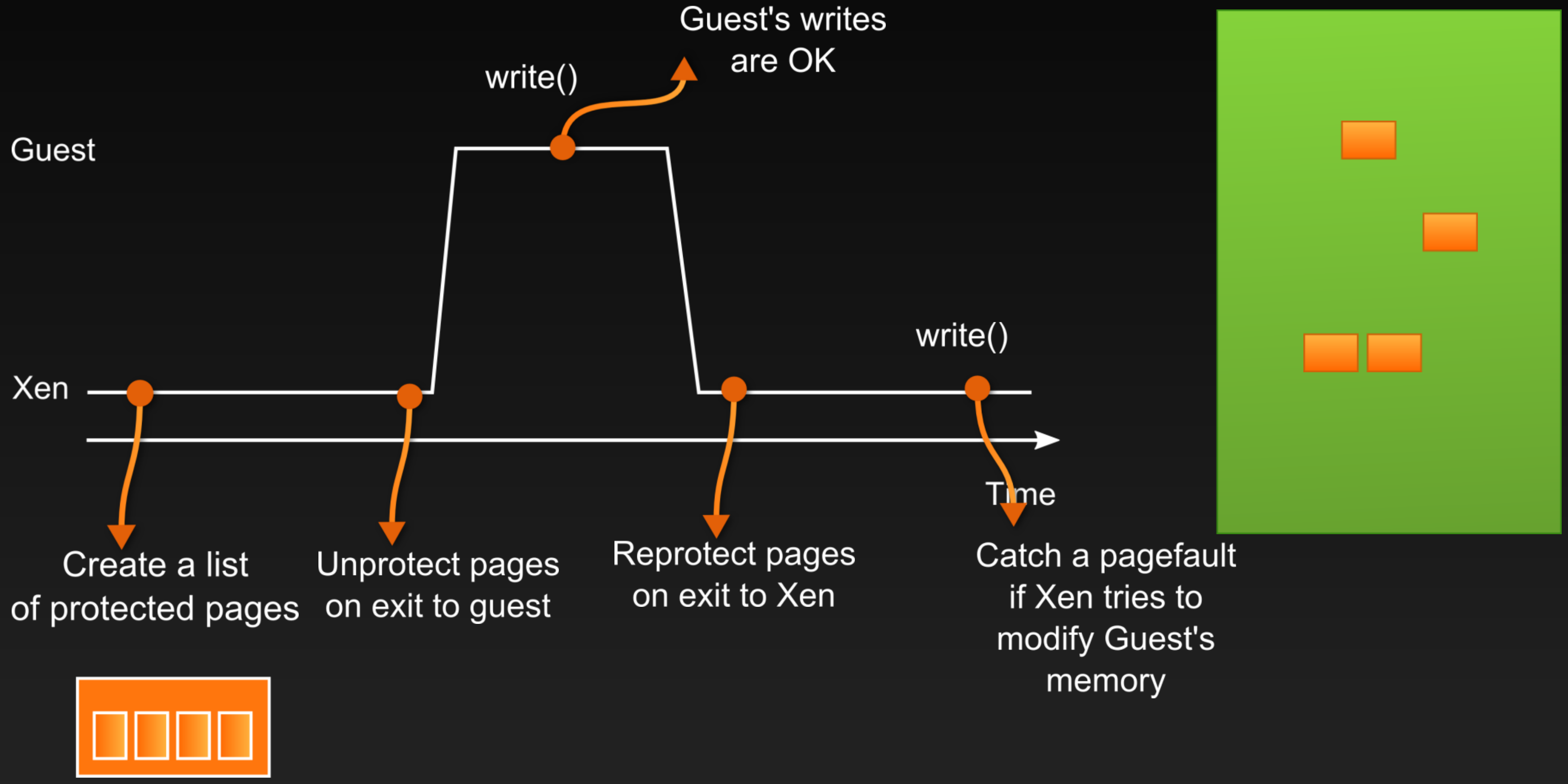
Execution environment

- CPU
- Virtual hardware
  - Memory
  - Disk
- Software

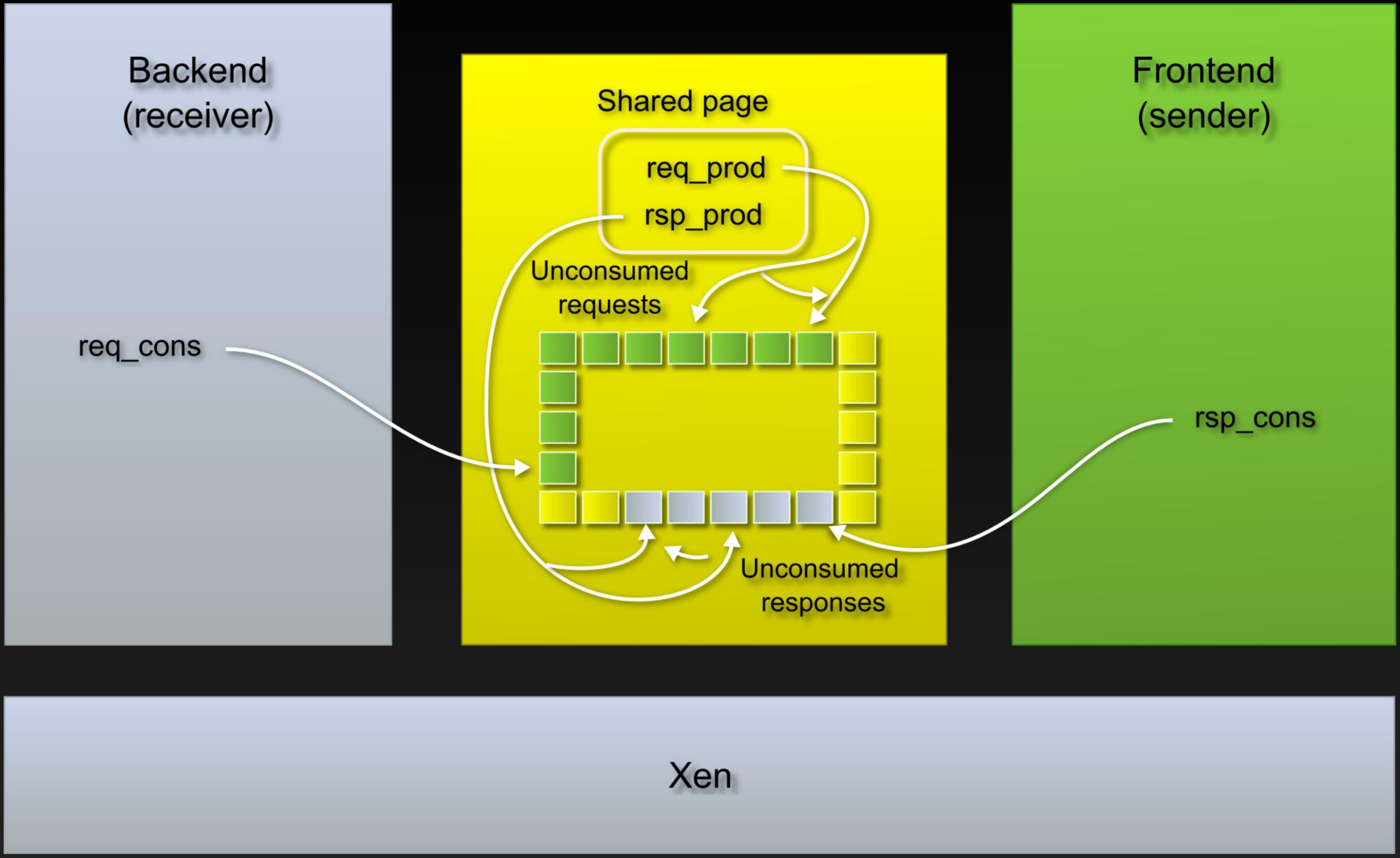


- External I/O

How do we find nondeterministic events?


Guest's writes are OK
write()
Guest
Xen
write()
Time
Create a list of protected pages
Unprotect pages on exit to guest
Reprotect pages on exit to Xen
Catch a pagefault if Xen tries to modify Guest's memory

Device interface



Reading a local variable

```
bsym_skb = target_lookup_sym(probe->target,
                             "netif_poll.skb",
                             ".", NULL, flags);

lval_skb = bsymbol_load(bsym_skb, flags);

skb_addr = *(unsigned long*) lval_skb->buf;
```

Controlled re-execution or replay

- Types of non-deterministic events
  - Synchronous
    - Hypercalls
  - Asynchronous
    - Interrupts
  - Best effort
    - Time updates
- Branch counters
  - The biggest problem of this solution