Be Kind, Please Rewind

Adventures in creating a macOS record/replay debugger

LEGAL


FBI
WARNING
Federal law provides severe civil and criminal penalties for the unauthorized reproduction, distribution or exhibition of copyrighted motion pictures, video tapes or video discs.
Criminal copyright infringement is investigated by the FBI and may constitute a felony with a maximum penalty of up to five years in prison and/or a $250,000 fine.
DEPARTMENT OF JUSTICE
FEDERAL BUREAU OF INVESTIGATION

REWIND <<

Why?

```
NSURLSessionDataTask *task = [_session
     dataTaskWithRequest:request
       completionHandler:^(NSData *_Nullable data, NSURLResponse *_Nullable response,
                           NSError *_Nullable err) {
         …
           *stop = YES;
```

```
Thread 2 Crashed::  Dispatch queue:
com.apple.NSXPCConnection.user.com.google.santa.metricservice.63335
0   libobjc.A.dylib       objc_msgSend + 29
1   Foundation            -[NSError copyWithZone:] + 107
2   santametricservice    -[SNTMetricHTTPWriter write:toURL:error:] +
1372
3   santametricservice    -[SNTMetricService exportForMonitoring:] + 475
…
```

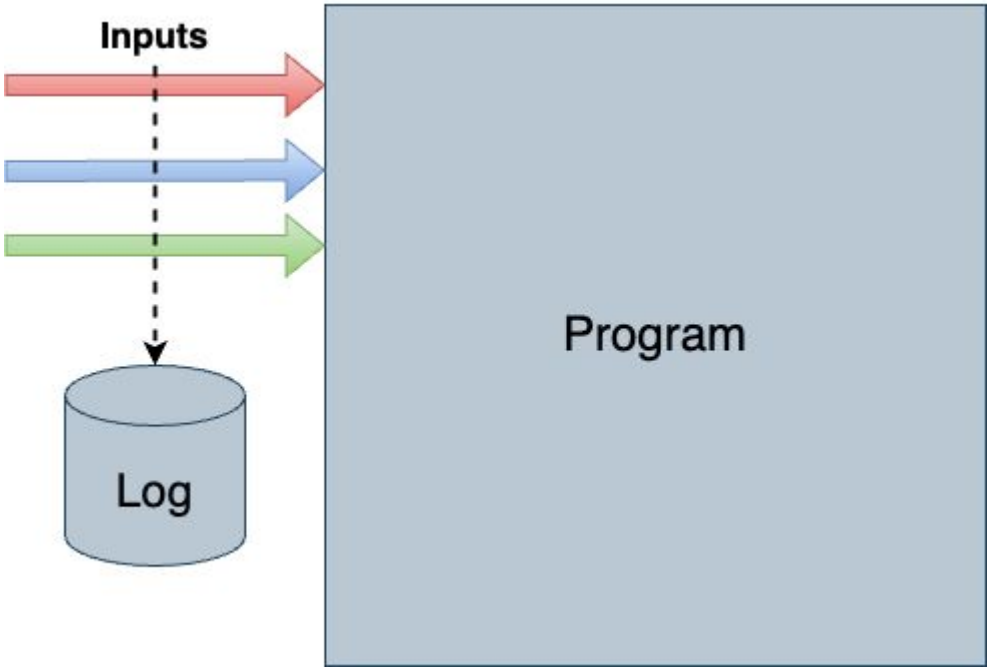
Record/Replay: Prior Art

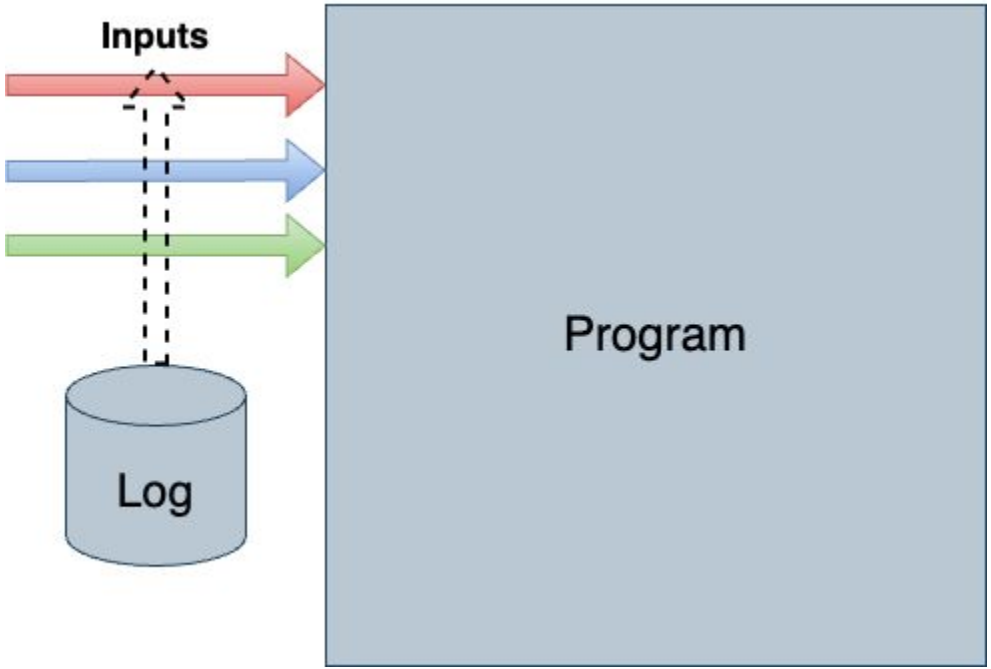
- PANDA (2020)
  - Whole system!
- WinDbg (2017)
- RR (2014)
- Scribe (2010)
- Jockey (2005)
- Flashback (2004)
- ReTrace
- QuickRec
- Revirt (1999)
  - Whole system!

Record/Replay: Prior Art

- PANDA (2020)
  - Whole system!
- WinDbg (2017)
- RR (2014)
- Scribe (2010)
- Jockey (2005)
- Flashback (2004)
- ReTrace
- QuickRec
- Revirt (1999)
  - Whole system!

Record / Replay Basics & Goals


Inputs
Program
Log


Inputs
Program
Log

Goals for our tool

- Only focusing on user-space programs
- Easy to use and deploy – needs to support a stock MBP with/M1,M2
- No DBI / code instrumentation
- Small investment of effort to maintain
- Fast enough to use on real programs

RR’s Requirements for User-Space Replay

Requirement	Does macOS Meet This Out of the Box?
Ability to Record Syscalls	✅
Ability to Record Syscalls Outside Libc	☹️
Ability to determine if a Syscall is blocking	🟡
Ability to Intercept Signals	✅

RR’s Requirements for User-Space Replay (Part 2)

Requirement	Does macOS Meet This Out of the Box?
Ability to pin a process to a single core (cpuset)	
Ability to trap non-deterministic instructions	
Ability to access reliable and deterministic hardware performance counters	


PUSH-OPEN
STEREO
NICAM DIGITAL STEREO
SUPER 4-HEAD
VHS
CLOCK
PROG
TRACKING
NEXT
MEMORY
AUDIO RECORDING
ON
OFF
AUDIO DUB
INSERT
REWIND
PLAY
FF
PAUSE/STILL
STOP
REC
Hi-Fi REC LEVEL
REMOTE CONTROL UNIT
VTR
CLOCK/COUNTER
AUDIO OUT
REC/PLAY
STILL ADV
SHUTTLE
SEARCH
TRANSMIT

Recording

Recording: It’s not that simple…

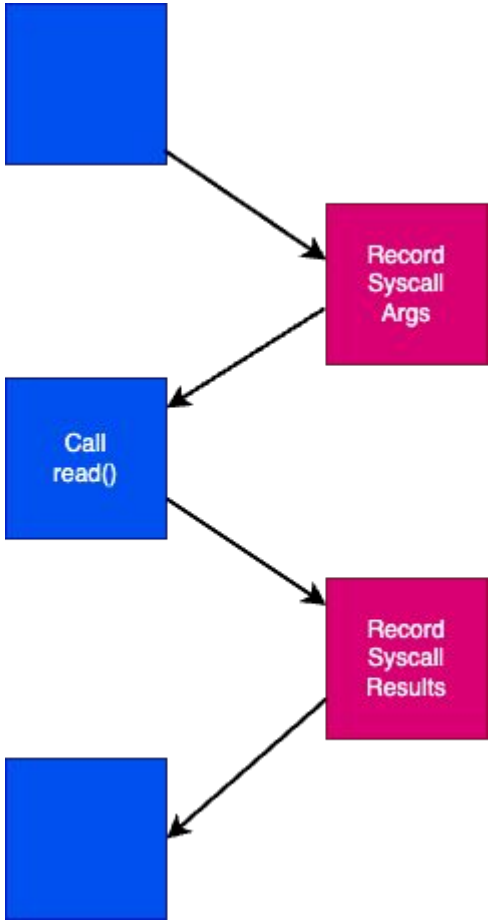
- Mach traps
  - Close enough to syscalls, no big deal
  - … except for a few traps which don’t have normal hook points
- Signals
  - The outside world can “asynchronously” poke the target
- mmap
- Multithreading
  - Well-formed programs shouldn’t have issues (data races), but…
  - Aside: thanks Apple for not giving us cpuset - no easy way to pin to one core
- Commpage
  - Similar to vvar (normally accessed via vDSO) on Linux
- Non-deterministic instructions – mrs x0, cntvct_el0

Recording: Syscalls & Traps

Recording: Syscalls on macOS

- 3 types – BSD, mach traps, machine dependent

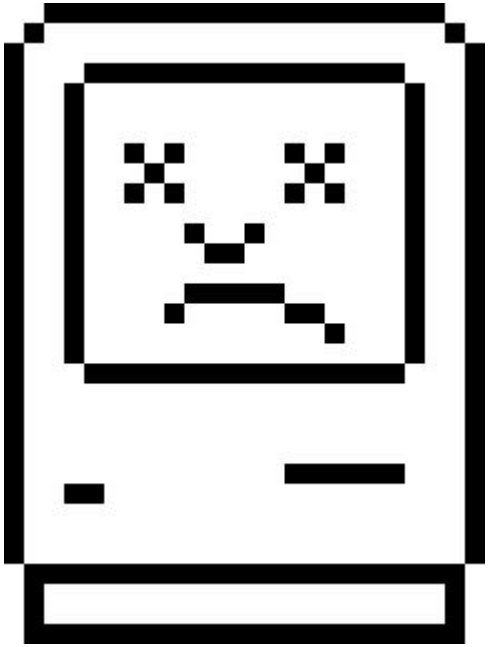
- Need pre- and post-hooks for data gathering



Thanks Apple

- Gutted ptrace implementation – no sysemu

- No seccomp-bpf equivalent



One Option: dtrace

- dtrace hooks, storage, etc.

- Not enough to capture arbitrary syscall data though
  - No conditionals for example - not possible to `switch` in “multiplexed” syscalls

- Strictly async
  - How to pause so userland can get what it needs?
  - Luckily there are “destructive” actions
    - signal(STOP)
    - stop() - mach_task_suspend
  - These only take effect *after* the syscall is processed though…

One Option: dtrace

Seatbelt / Sandbox?

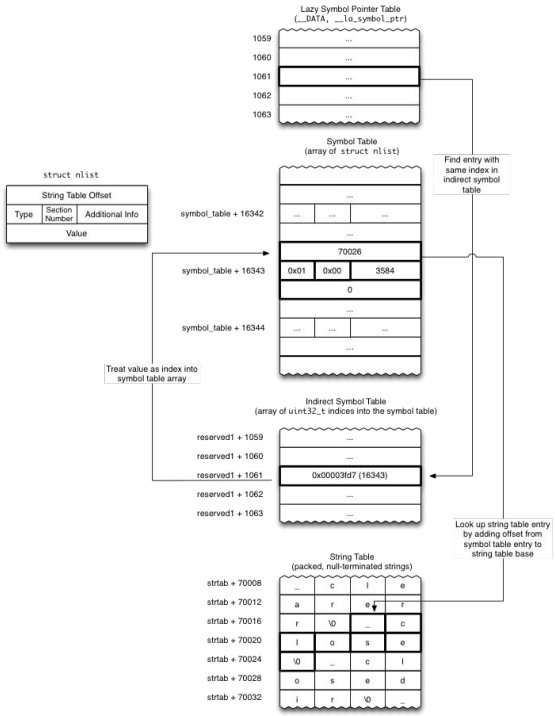
- Seatbelt is wired up into every syscall maybe?

- Trace mode for recording
  - Not a good API, minimal log entries

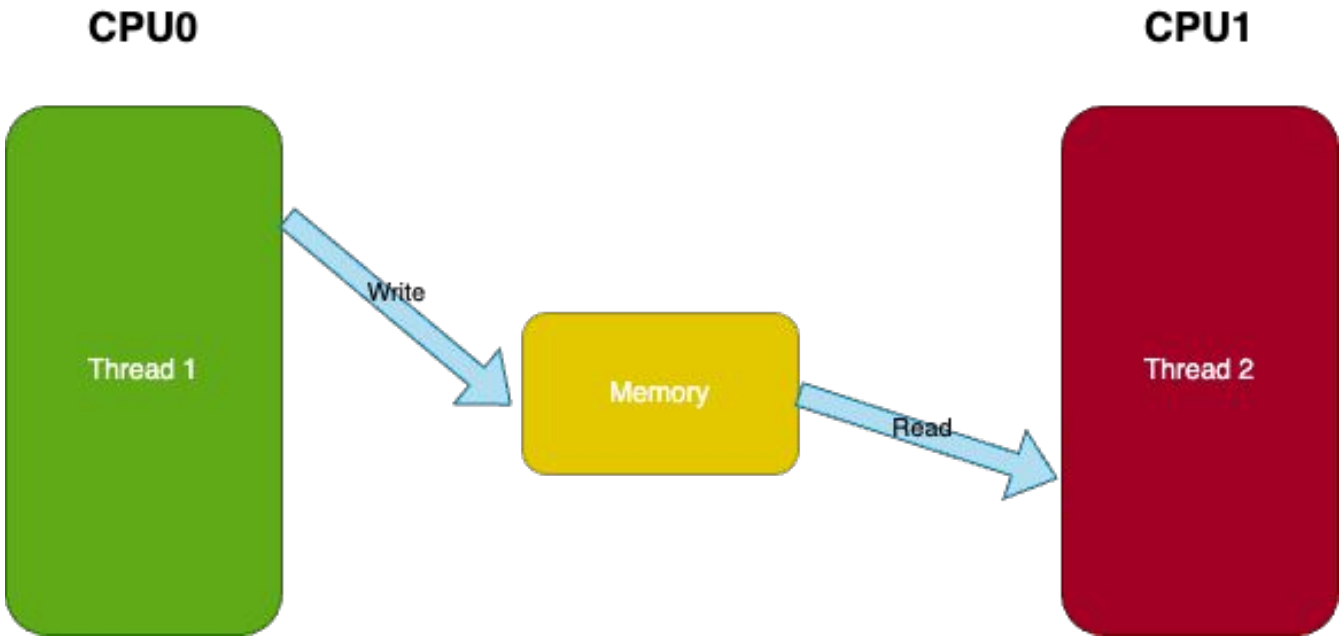
- No way to not kill on replay

Interposing / Dynamic Interposing / Symbol Rebinding

- macOS is a BSD!
  - ABI compatibility is at the libc level not the kernel
  - Can we just hook libsystem_kernel?
- We can interpose on the symbol
  - Could use [fishhook]
- Doesn’t catch direct syscalls…



Recording: Dealing with Data Races

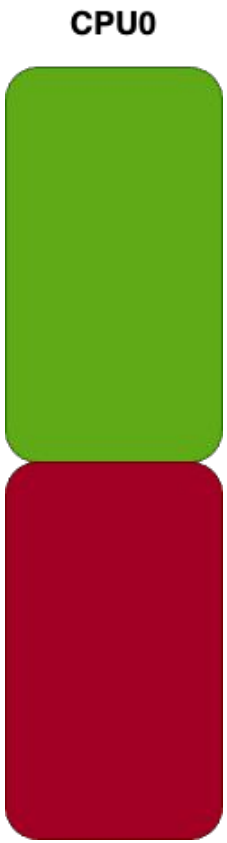

CPU0
CPU1
Thread 1
Write
Memory
Read
Thread 2

How Does RR Handle This?

- Only runs one thread to run at a time (non-parallel)

- Limits threads to the same core using processor affinity

- Schedules threads and records the choice in the log (**can mixup order on replay to find bugs**)

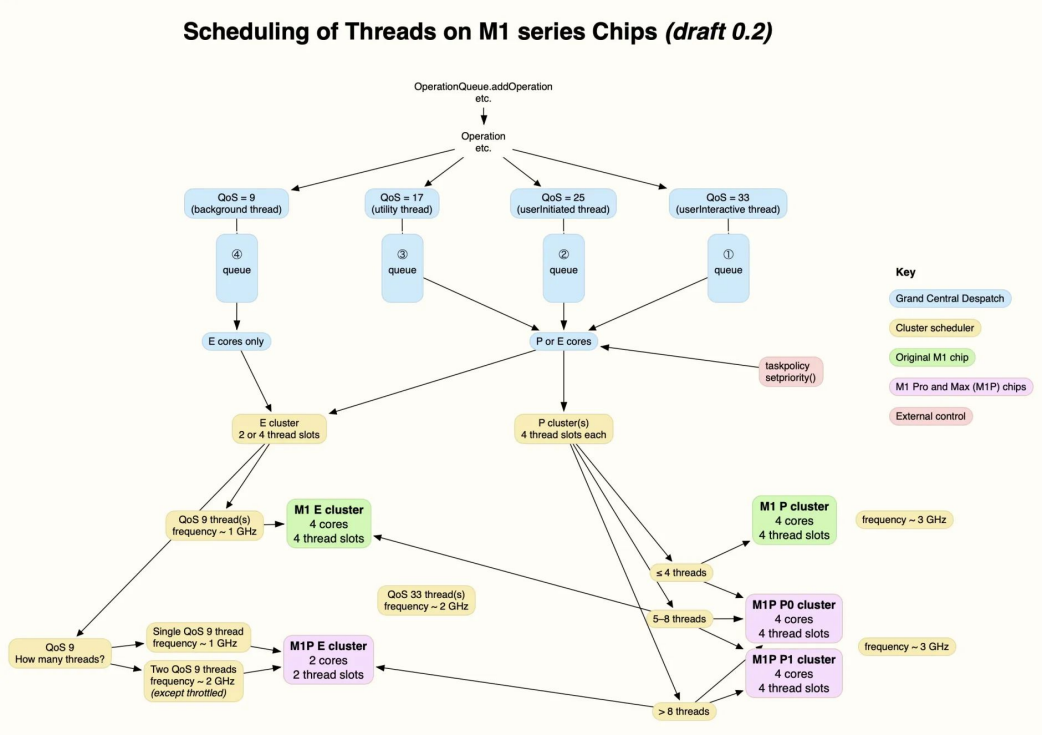


Thread scheduling on macOS not guaranteed

- No `cpu_set(3)`
- Can we use `THREAD_AFFINITY_POLICY`?

```
/*
 *      thread_bind:
 *
 *      Force the current thread to execute on the specified processor.
 *      Takes effect after the next thread_block().
 *
 *      Returns the previous binding.  PROCESSOR_NULL means
 *      not bound.
 *
 *      XXX - DO NOT export this to users - XXX
 */
processor_t
thread_bind(
        processor_t             processor)
{
        thread_t                self = current_thread();
        processor_t             prev;
```

P-cores and E-cores



From: https://eclecticlight.co/2022/01/13/scheduling-of-processes-on-m1-series-chips-first-draft/

Can we shutdown cores?

- In the old OS X internals books there was an example showing how to shutdown cores using `processor_exit`
- Can we just limit ourselves to a core?

```
[ user@watervile ~ ]
$  sudo ./print_processors
Password:
Number of processors: 12
CPU: slot 0(master)
CPU: slot 1
//snipped.
CPU: slot 11
[ user@waterville ~ ]
$  sudo ./processor_xable
processor_exit: (os/kern) service not supported
```

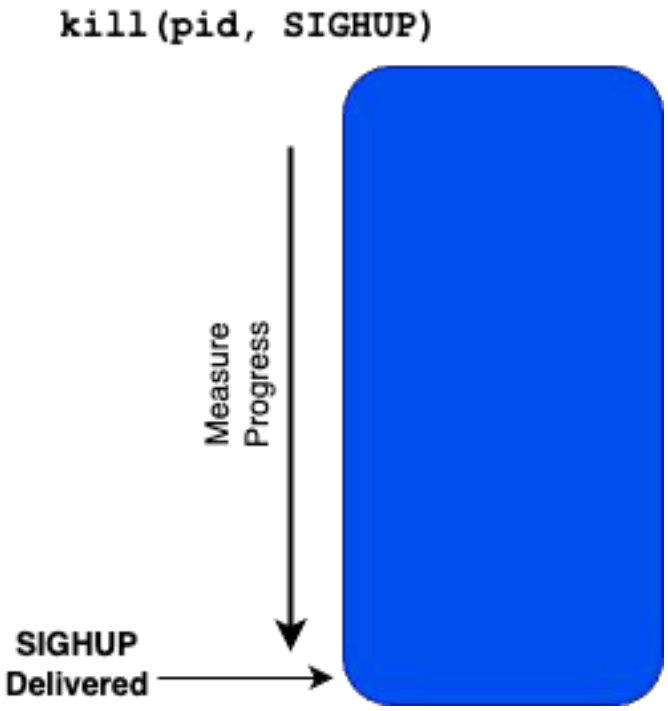
Recording: Asynchronous Events

Signals & Scheduling

- Need to be able to intercept signals and record register state of where the signal was delivered or program interrupted for scheduling.

- Need to know where you are in the programs execution so you can inject your signals in the right place during replay

- Replay: when using something interrupt driven must account for late firing interrupts



Using PMUs from macOS

- RR works on Asahi Linux and uses the PMU can we?
  - Uses the count of retired conditional branches as progress indicator (0x8c)
  - Can reset for an interrupt when replaying

- macOS does not have an interface for setting PMUs from EL0

```
[ user@waterville  ~/src/pmu_counters  ]
$  sudo ./counter_test
loaded db: a15 (Apple A15)
number of fixed counters: 2
number of configurable counters: 8
counters value:
        cycles: 41865278
  instructions: 91998218
      branches: 21071096
 branch-misses: 53779
[ user@waterville  ~/src/pmu_counters  ]
$  sudo ./counter_test
loaded db: a15 (Apple A15)
number of fixed counters: 2
number of configurable counters: 8
counters value:
        cycles: 41946121
  instructions: 92093331
      branches: 0
 branch-misses: 0
```

panic(cpu 5 caller 0xfffffe0017c66cd8): kperf: timer fired at 2793246644070, but sampling is disabled @kptimer.c:328
Debugger message: panic

Supporting Nondeterministic Execution in Fault-Tolerant Systems*

J. Hamilton Slye
Dept. of Electrical and Computer Engineering
Carnegie Mellon University
ham+@cmu.edu

E.N. Elnozahy
Department of Computer Science
Carnegie Mellon University
mootaz@cs.cmu.edu

Abstract

We present a technique to track nondeterminism resulting from asynchronous events and multithreading in log-based rollback-recovery protocols. This tech-

with the end users [11]. Efficient tracking of nondeterminism is thus crucial to supporting interactive applications [14].

Different flavors of logging have been suggested with different performance and resilience character-

```
Decrement Register
Branch to handler if register = 0
```

Options without PMU or DBI

- We can count the number of syscalls and then single step forward then inject the signal (set a breakpoint and invoke the signal handler)

- Do what scribe(10) does and simply deliver the signal at the next syscall and replay interrupted syscall (special case for signals like SIGSEGV that originate in user space.)

- If we need to go further than say 10,000 instructions we can use an high res clock (e.g. pacman) to trap back to us

```
    // set up the registers for our counter
    asm!(
        "mov x3, {running}",
        "mov x4, x0",    // store backed up copy o
        "ldr x0, [x0]", // load the value of coun
        "1:",
        "isb",
        "ldr x2, [x3]",  // load the value of run
        "sub x0, x0, 1", // decrement the counter
        "cbz x2, 456f",  // break the loop if we'
        "cbnz x0, 1b",   //  if we've not underflo
        "123:",
        "mov {interrupted}, 1", // set interrupte
        "456:",
        "str xzr, [x3]", // set running to false
        "str x0, [x4]",  // store the value of co
        in("x0") count_addr,
        interrupted = out(reg) interrupted,
        running = in(reg) running_addr,
    );
```


PUSH-OPEN
STEREO
NICAM DIGITAL STEREO
SUPER 4-HEAD
VHS
CLOCK
PROG
TRACKING
NEXT
MEMORY
AUDIO RECORDING
ON
OFF
AUDIO DUB
INSERT
REWIND
PLAY
FF
PAUSE/STILL
STOP
REC
Hi-Fi REC LEVEL
REMOTE CONTROL UNIT
VTR
CLOCK/COUNTER
AUDIO OUT
REC/PLAY
STILL ADV
SEARCH
TRANSART

Darling

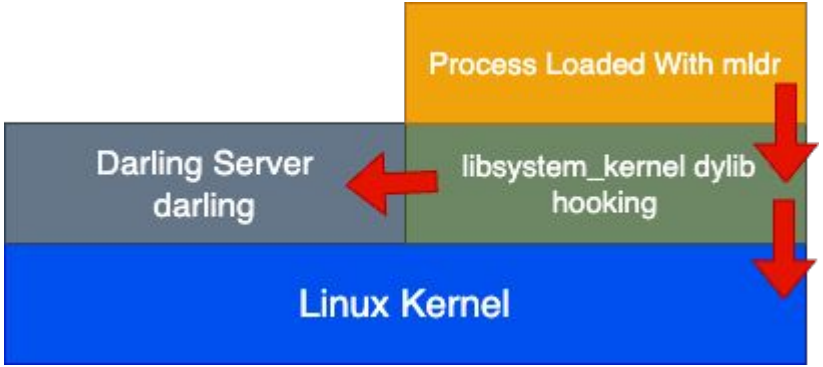
Darling

- “A Translation Layer that lets you run macOS software on Linux”

- Uses a custom loader, interposing of libsystem_kernel, a lot of duct tape code and userland a server to translate macOS syscalls to Linux syscalls

- Can run software like xcode on Linux

High Level: How Darling Works

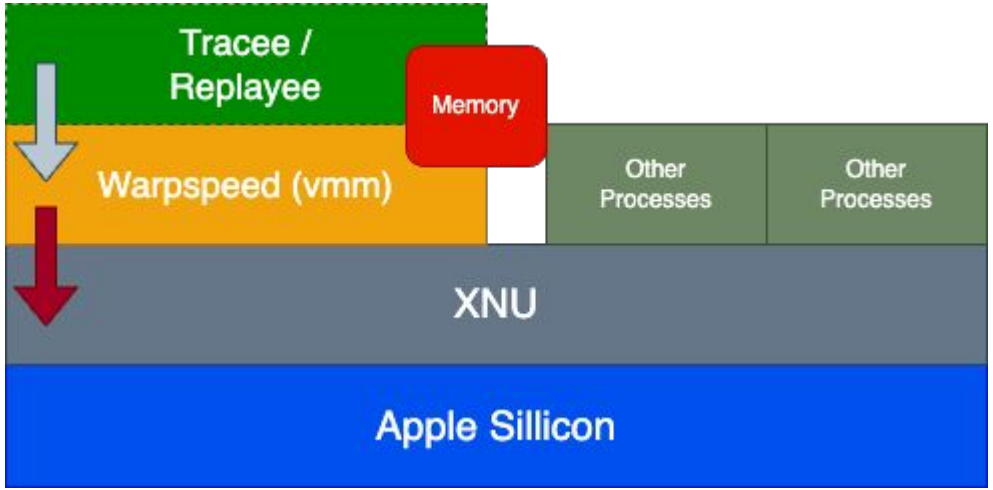



PUSH-OPEN
STEREO
NICAM DIGITAL STEREO
SUPER 4-HEAD
VHS
CLOCK
PROG
TRACKING
NEXT
MEMORY
AUDIO RECORDING
ON
OFF
AUDIO DUB
INSERT
REWIND
PLAY
FF
PAUSE/STILL
STOP
REC
Hi-Fi REC LEVEL
REMOTE CONTROL UNIT
VTR
CLOCK/COUNTER
AUDIO OUT
REC/PLAY
STILL ADV
SHUTTLE
SEARCH
TRANSMIT

Warpspeed

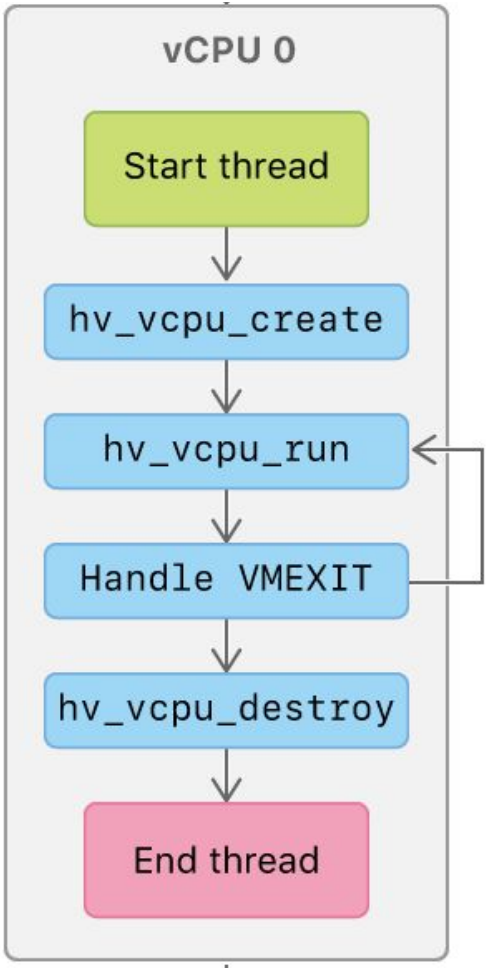
Warpspeed

- Isolate target inside a VM with 1 core
- Proxy syscalls
- Both signal slide + SoftPMU to approximate program progression
- Manual thread scheduling



Hypervisor.framework

- Super light-weight framework
  - Little as possible in the kernel
- Usage:
  - Create a VM
  - Map memory (from hypervisor address space)
  - Create vCPU
  - Set regs
  - Run
  - Trap out to <u>userland</u> on VM exit
  - GOTO 5
  - *That's it*



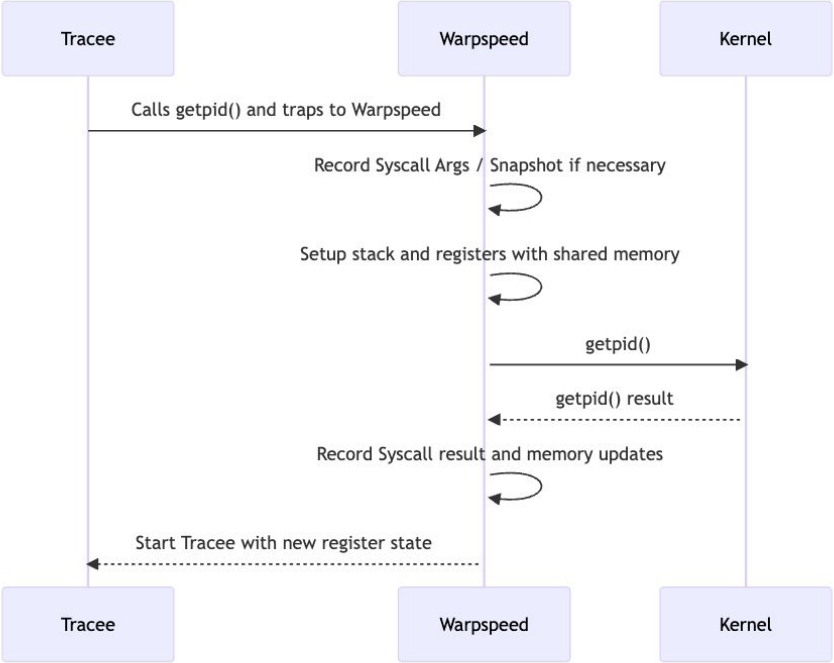
Warpspeed: VM/Hypervisor

- Use modified darling's loader (mldr) to map in target program and dyld
- Load in shared cache
- "Share" an address space with the guest
  - 1:1 map the regions of the loaded target into VM at the same virtual address
- Trap out and forward syscalls

- All based on Hyperpom (Rust!)

- Lets us control the execution of the program perfectly
  - Only have one virtual core
  - Manually schedule threads

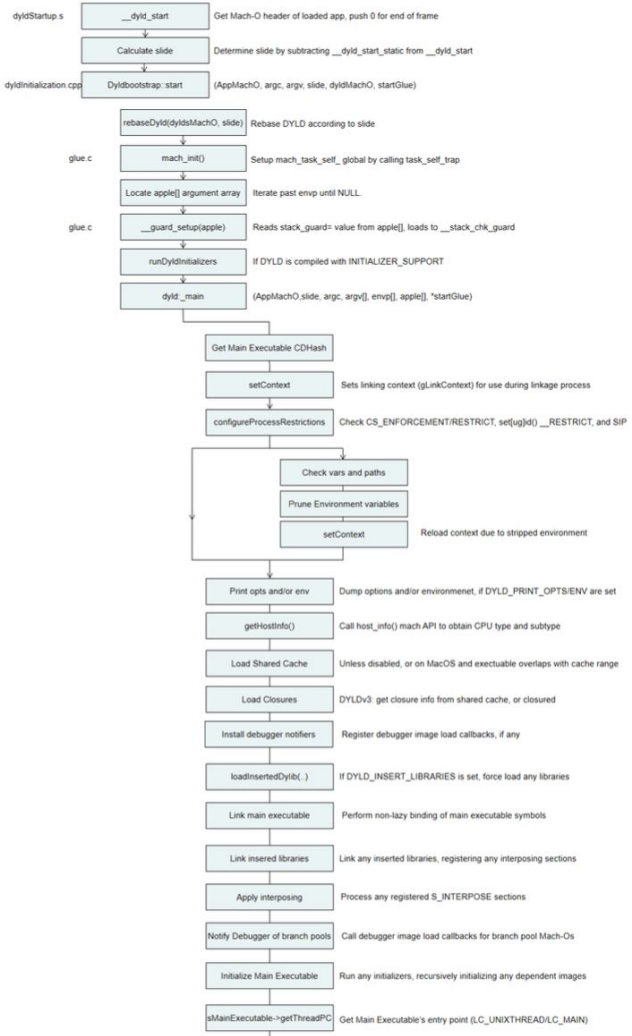
Warpspeed: VM/Hypervisor



```
[2023-06-09T03:19:56Z  INFO  mrr::warpspeed: 267: Forwarding syscall 99(0x[0, 134088000, 0xf, 0, 0, 0, 0, 0x0, 0, 101, 4000
, 5f5e00, 1ae7, 134006530, 2db3ffb40, 2db3ff4a0])
[2023-06-09T03:19:56Z  TRACE  mrr::warpspeed: Changed pages: []
[2023-06-09T03:19:56Z  INFO  mrr::warpspeed: Returning 0xf @ 4000030
[2023-06-09T03:19:56Z  TRACE  mrr::warpspeed: ELR_EL1: 0x18031c8a0
[2023-06-09T03:19:56Z  INFO  mrr::warpspeed: Incoming syscall 6([0x[0, 134088000, 134080000, 0, 4000c402, 2, 2, 0x0, 101, 10
3, 0, 27, 134000000, 1, 134000000, 4000c402])
[2023-06-09T03:19:56Z  INFO  mrr::warpspeed: 268: Forwarding syscall 0x0xD(, 134000000, 134000000, 0, 4000c402, 2, 2, 0x0,
101, 103, 0, 27, 134000000, 1, 134000000, 4000c402])
[2023-06-09T03:19:56Z  TRACE  mrr::warpspeed: Changed pages: []
[2023-06-09T03:19:56Z  INFO  mrr::warpspeed: Returning 0 @ 4000030
[2023-06-09T03:19:56Z  TRACE  mrr::warpspeed: ELR_EL1: 0x18031c8a0
[2023-06-09T03:19:56Z  INFO  mrr::warpspeed: Incoming syscall 5([0x[101b01f00, 1000104, 0, 101b01be1, 0, 50, 0, 0x0, 101b010
e1, 0, 12002027, 1, 1, 134000540, 2db3ffb40, 2db3ff4a0])
[2023-06-09T03:19:56Z  INFO  mrr::warpspeed: open(/System/appleInternal/Library/FeatureFlags/GlobalDiscloseOverrides.plist
)
[2023-06-09T03:19:56Z  INFO  mrr::warpspeed: 269: Forwarding syscall 5(0x[101b01000, 1000104, 0, 101b01b40, 0, 50, 0, 0x0,
101b01b40, 0, 12002027, 1, 1, 134000540, 2db3ffb40, 2db3ff4a0])
[2023-06-09T03:19:56Z  TRACE  mrr::warpspeed: Changed pages: []
[2023-06-09T03:19:56Z  INFO  mrr::warpspeed: Returning 3 @ 600003c0
[2023-06-09T03:19:56Z  TRACE  mrr::warpspeed: ELR_EL1: 0x18031c8a0
[2023-06-09T03:19:56Z  INFO  mrr::warpspeed: Incoming syscall 5([0x[101b01f00, 1000104, 0, 101b01bcf, 0, 50, 0, 0x0, 101b010
cf, dd6037244cf2e8b6, 12002027, 1, 1, 134000540, 2db3ffb40, 2db3ff4a0])
[2023-06-09T03:19:56Z  INFO  mrr::warpspeed: open(/System/Library/FeatureFlags/ProfileDiscloseOverrides.plist)
[2023-06-09T03:19:56Z  INFO  mrr::warpspeed: 270: Forwarding syscall 5(0x[101b01000, 1000104, 0, 101b01bcf, 0, 50, 0, 0x0,
101b01bcf, dd6037244cf2e8b6, 12002027, 1, 1, 134000540, 2db3ffb40, 2db3ff4a0])
[2023-06-09T03:19:56Z  TRACE  mrr::warpspeed: Changed pages: []
[2023-06-09T03:19:56Z  INFO  mrr::warpspeed: Returning 3 @ 600003c0
[2023-06-09T03:19:56Z  TRACE  mrr::warpspeed: ELR_EL1: 0x18031c8a0
[2023-06-09T03:19:56Z  INFO  mrr::warpspeed: Incoming syscall 5([0x[101b01f00, 1000104, 0, 101b01bcf, 0, 50, 0, 0x0, 101b010
cf, dd6037244cf2e8b6, 12002027, 1, 1, 134000540, 2db3ffb40, 2db3ff4a0])
[2023-06-09T03:19:56Z  INFO  mrr::warpspeed: open(/System/Library/FeatureFlags/SettingsDiscloseOverrides.plist)
```

dyld

Figure 7-2: The flow of dyld



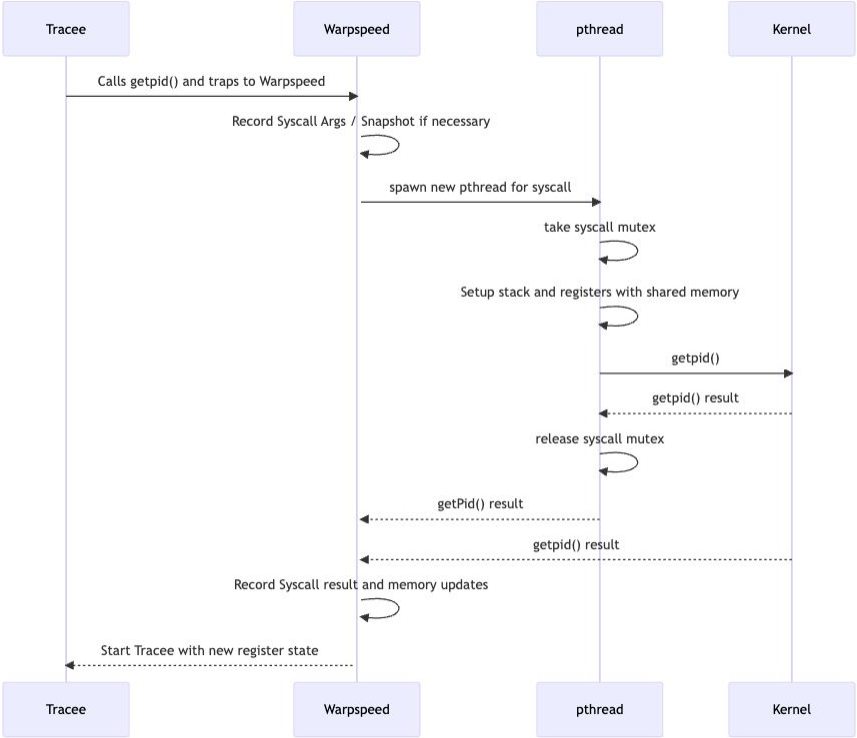
Warpspeed: Unimplemented Features

- LLDB/GDB interface
- Optimizing/compressing log format
- The hypervisor itself is responsible for performing the syscalls
  - What happens on a blocking call?
  - Could deadlock on mutex wait
- Handling blocking syscalls
  - Manually enumerate and perform some non-blocking alternative
  - or…

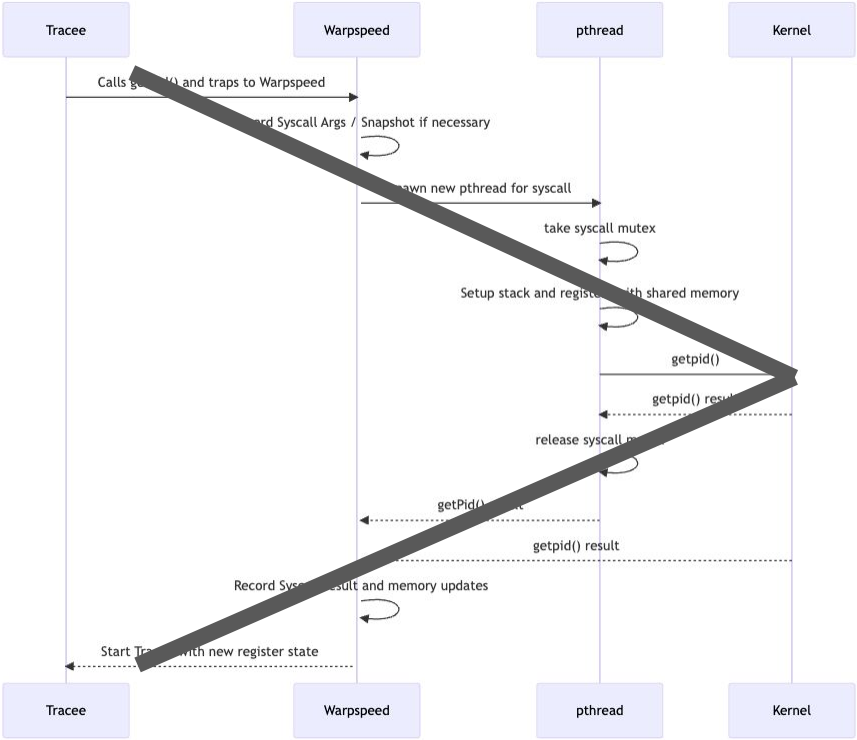
```
nickgregory@Nick-Virtual-Machine ~ $ echo $$
1486
nickgregory@Nick-Virtual-Machine ~ $ ls
```

```
nickgregory@Nick-Virtual-Machine ~ $ sudo ./dtrace_kdebug_blocking 1486

Threads: [0010]
[/usr/demos/kdebug_blocking.rs:50] "{:?}" = "{:?}"
[/usr/demos/kdebug_blocking.rs:51] event = KdBuf {
    timestamp: 1458004516,
    arg1: 1,
        0,
        0,
        0,
        0,
        0x0,
    },
    debugid: 20971500,
    cpuid: 0,
    __unused: 0,

[/usr/demos/kdebug_blocking.rs:50] "{:?}" = "{:?}"
[/usr/demos/kdebug_blocking.rs:51] event = KdBuf {
    timestamp: 1457154600,
    arg1: 1,
        0,
        0,
        0,
        0,
        0x0,
    },
    debugid: 20971500,
    cpuid: 2,
    __unused: 0,
```

Warpspeed: VM/Hypervisor



Warpspeed: VM/Hypervisor



Warpspeed: Outstanding Issues

- MMIO
- Entitlements

That’s Only Half the Battle

Replay

- If you can figure out recording, replay is much simpler
  - Set breakpoints where something happened in recording
  - Mimic side-effects
  - Continue
- SoftPMU needed here in case we end up with an async event in a hot loop

Replay: GUI

- UI is core to macOS
- How can we “pass through” events on replay to the OS (to see the app running) while not introducing nondeterminism?
  - *In theory* it will “just work”
  - No (easy) way to show the UI on replay though

Recap

- Tool is WIP
- But principles work!
- Stay posted for more


INPUT SELECT
HICAM DIGITAL STEREO
SUPER 4-HEAD
VHS
REWIND
PLAY
FF
PAUSE/STILL
STOP
REC
TRACKING
MEMORY
AUDIO DUB
INSERT
REMOTE CONTROL UNIT
AUDIO OUT
TRANSART

Questions?