Hardware Tracing for Data Race Detection

Farzam Dorostkar with Pr. Michel Dagenais

January 14th, 2022

Polytechnique Montreal

DORSAL Laboratory

Introduction

Research Focus : Hardware tracing on Intel and ARM for low-overhead memory bug detection

Current Research :

Intel Processor Trace (Intel PT) for post-mortem data race detection

- Current tools only rely on code instrumentation
  - TSan : 5x-15x slowdown, 5x-10x memory overhead

Agenda

- Data Race
- Intel Processor Trace (Intel PT)
- Data Race Detection with Intel PT
- Demo
- Conclusion & Future work

Multithread Programming

- Shared variables allow threads to communicate quickly
- A bug when two+ threads access the same shared variable concurrently and at least one access is a write (Data Race!)

Data Race

Multithread Programming

- Shared variables allow threads to communicate quickly
- A bug when two+ threads access the same shared variable concurrently and at least one access is a write (Data Race!)

t1 R	t2 R	t1 R
t1 W	t2 W	t2 R
t2 R	t1 R	t1 W
t2 W	t1 W	t2 W
count = 2	count = 2	Count = 1

```
int count = 0; //Shared variable

void *routine_one(void *arg) {
    count++;
}

void *routine_two(void *arg) {
    count++;
}

int main() {
    pthread_t t1, t2;
    pthread_create(&t1, NULL, &routine_one, NULL);
    pthread_create(&t2, NULL, &routine_two, NULL);
    pthread_join(t[0], NULL);
    pthread_join(t[1], NULL);
    return 0;
}
```

How to Determine Concurrent Events?

Equivalent question: “Can we determine partial ordering?”

- Lamport’s Happens-before relation [1]

  i. If *a* and *b* in the same thread & *a* comes before *b* : $a \rightarrow b$

  ii. If *a* and *b* are a synchronization-pair (lock/unlock) : $a \rightarrow b$

  iii. If $a \rightarrow b$ & $b \rightarrow c$: $a \rightarrow c$ (Transitivity)

  If $a! \rightarrow b$ & $b! \rightarrow a$: *a* and b are concurrent!

  Used by many tools including TSan, Helgrind, and GO’s built-in data race detector

[1] L. Lamport, “Time, clocks, and the ordering of events in a distributed system,” Commun. ACM, 21(7):558–565, 1978.

How to Implement Happens-before?

Vector clocks

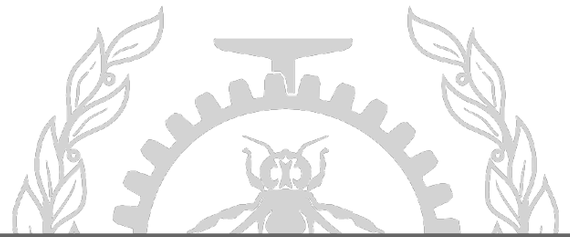
- A program with N threads: each thread maintains a vector of N logical clocks

  i. All ckocks are zero initially

  ii. For own events increments own logical clock by one

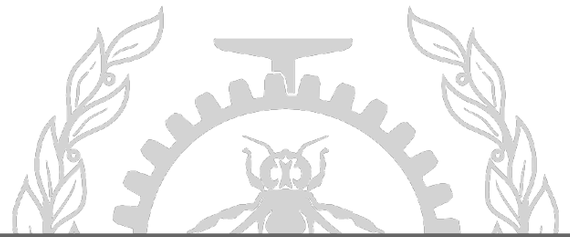
  iii. Synchronization transmits the state of unlocking thread to the locking thread

If every element of *a*'s vector clock is less than *b*'s : $a \rightarrow b$



Intel Processor Trace (Intel PT)

- A hardware feature that logs information about software execution with minimal impact
- A non-intrusive means to trace control flow
  - <5% performance overhead
  - Decoder can reconstruct the precise execution flow
- Trace data is generated only for non statically known control flow changes
- Can store both cycle count and timestamp information
- No need to modify source code!
  - Run under Intel PT-enabled debug and profiling tools (like Linux perf & GDB)



Intel Processor Trace (Intel PT)

Control Flow Tracing

- TNT (Taken Not-Taken) : direct conditional branches (generates only a 1-bit indication)
- TIP (Target IP) : target address of indirect branches, exception, and interrupts

Instructions	Encoding
push	
mov	
cmp	
je .L1	TNT (Taken Not Taken)
mov	
add	
L1:	
Call (edx) // virtual function	TIP (Target IP)

PT trace
T
0x4012a4

Intel Processor Trace (Intel PT)

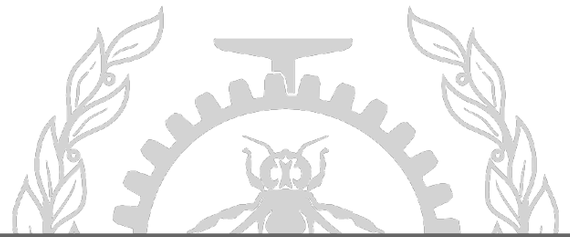
Reconstructing the Control Flow

Decoder can determine the exact execution flow from trace log

Instructions
push
mov
cmp
je .L1
mov
add
L1:
Call (edx) // virtual function

PT trace
T
0x4012a4

Reconstructed Control Flow
push
mov
cmp
je .L1
mov
add
L1:
Call (0x4012a4)



Data Race Detection with Intel PT

TID	IP	Addr	Sym+off	Insn
20295	0	4011d0	routine_two+0x0	pushq %rbp
20295	0	4011d1	routine_two+0x1	mov %rsp, %rbp
20295	0	4011d4	routine_two+0x4	sub $0x10, %rsp
20295	0	4011d8	routine_two+0x8	movq %rdi, -0x8(%rbp)
20295	0	4011dc	routine_two+0xc	mov $0x404060, %rdi
20295	401080	4011e6	routine_two+0x16	callq 0xfffffffffffffe9a
20295	0	4011eb	routine_two+0x1b	movl 0x40405c, %ecx
20295	0	4011f2	routine_two+0x22	add $0x1, %ecx
20295	0	4011f5	routine_two+0x25	movl %ecx, 0x40405c
20295	0	4011fc	routine_two+0x2c	mov $0x404060, %rdi
20295	0	401206	routine_two+0x36	movl %eax, -0xc(%rbp)
20295	401050	401209	routine_two+0x39	callq 0xfffffffffffffe47
20295	0	40120e	routine_two+0x3e	xor %ecx, %ecx
20295	0	401210	routine_two+0x40	mov %ecx, %edx
20295	0	401212	routine_two+0x42	movl %eax, -0x10(%rbp)
20295	0	401215	routine_two+0x45	mov %rdx, %rax
20295	0	401218	routine_two+0x48	add $0x10, %rsp
20295	0	40121c	routine_two+0x4c	popq %rbp
20295	7f4def4f2609	40121d	routine_two+0x4d	retq
20294	0	401180	routine_one+0x0	pushq %rbp
20294	0	401181	routine_one+0x1	mov %rsp, %rbp
20294	0	401184	routine_one+0x4	sub $0x10, %rsp
20294	0	401188	routine_one+0x8	movq %rdi, -0x8(%rbp)
20294	0	40118c	routine_one+0xc	mov $0x404060, %rdi
20294	401080	401196	routine_one+0x16	callq 0xfffffffffffffeea
20294	0	40119b	routine_one+0x1b	movl 0x40405c, %ecx
20294	0	4011a2	routine_one+0x22	add $0x1, %ecx
20294	0	4011a5	routine_one+0x25	movl %ecx, 0x40405c
20294	0	4011ac	routine_one+0x2c	mov $0x404060, %rdi
20294	0	4011b6	routine_one+0x36	movl %eax, -0xc(%rbp)
20294	401050	4011b9	routine_one+0x39	callq 0xfffffffffffffe97
20294	0	4011be	routine_one+0x3e	xor %ecx, %ecx
20294	0	4011c0	routine_one+0x40	mov %ecx, %edx
20294	0	4011c2	routine_one+0x42	movl %eax, -0x10(%rbp)
20294	0	4011c5	routine_one+0x45	mov %rdx, %rax
20294	0	4011c8	routine_one+0x48	add $0x10, %rsp
20294	0	4011cc	routine_one+0x4c	popq %rbp
20294	7f4def4f2609	4011cd	routine_one+0x4d	retq

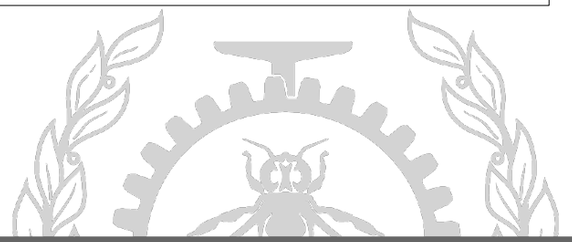
Decoded PT trace of the sample code (perf)

```
int count = 0; //Shared variable
pthread_mutex_t mutex;

void *routine_one(void *arg) {
    pthread_mutex_lock(&mutex);
    count++;
    pthread_mutex_unlock(&mutex);
}

void *routine_two(void *arg) {
    pthread_mutex_lock(&mutex);
    count++;
    pthread_mutex_unlock(&mutex);
}

int main() {
    pthread_t t1, t2;
    pthread_mutex_init(&mutex, NULL);
    pthread_create(&t1, NULL, &routine_one, NULL);
    pthread_create(&t2, NULL, &routine_two, NULL);
    pthread_join(t1, NULL);
    pthread_join(t2, NULL);
    pthread_mutex_destroy(&mutex);
    return 0;
}
```



Data Race Detection with Intel PT

TID	IP	Addr	Sym+off	Insn
20295	0	4011d0	routine_two+0x0	pushq %rbp
20295	0	4011d1	routine_two+0x1	mov %rsp, %rbp
20295	0	4011d4	routine_two+0x4	sub $0x10, %rsp
20295	0	4011d8	routine_two+0x8	movq %rdi, -0x8(%rbp)
20295	0	4011dc	routine_two+0xc	mov $0x404060, %rdi
20295	401080	4011e6	routine_two+0x16	callq 0xfffffffffffffe9a
20295	0	4011eb	routine_two+0x1b	movl 0x40405c, %ecx
20295	0	4011f2	routine_two+0x22	add $0x1, %ecx
20295	0	4011f5	routine_two+0x25	movl %ecx, 0x40405c
20295	0	4011fc	routine_two+0x2c	mov $0x404060, %rdi
20295	0	401206	routine_two+0x36	movl %eax, -0xc(%rbp)
20295	401050	401209	routine_two+0x39	callq 0xfffffffffffffe47
20295	0	40120e	routine_two+0x3e	xor %ecx, %ecx
20295	0	401210	routine_two+0x40	mov %ecx, %edx
20295	0	401212	routine_two+0x42	movl %eax, -0x10(%rbp)
20295	0	401215	routine_two+0x45	mov %rdx, %rax
20295	0	401218	routine_two+0x48	add $0x10, %rsp
20295	0	40121c	routine_two+0x4c	popq %rbp
20295	7f4def4f2609	40121d	routine_two+0x4d	retq
20294	0	401180	routine_one+0x0	pushq %rbp
20294	0	401181	routine_one+0x1	mov %rsp, %rbp
20294	0	401184	routine_one+0x4	sub $0x10, %rsp
20294	0	401188	routine_one+0x8	movq %rdi, -0x8(%rbp)
20294	0	40118c	routine_one+0xc	mov $0x404060, %rdi
20294	401080	401196	routine_one+0x16	callq 0xfffffffffffffeea
20294	0	40119b	routine_one+0x1b	movl 0x40405c, %ecx
20294	0	4011a2	routine_one+0x22	add $0x1, %ecx
20294	0	4011a5	routine_one+0x25	movl %ecx, 0x40405c
20294	0	4011ac	routine_one+0x2c	mov $0x404060, %rdi
20294	0	4011b6	routine_one+0x36	movl %eax, -0xc(%rbp)
20294	401050	4011b9	routine_one+0x39	callq 0xfffffffffffffe97
20294	0	4011be	routine_one+0x3e	xor %ecx, %ecx
20294	0	4011c0	routine_one+0x40	mov %ecx, %edx
20294	0	4011c2	routine_one+0x42	movl %eax, -0x10(%rbp)
20294	0	4011c5	routine_one+0x45	mov %rdx, %rax
20294	0	4011c8	routine_one+0x48	add $0x10, %rsp
20294	0	4011cc	routine_one+0x4c	popq %rbp
20294	7f4def4f2609	4011cd	routine_one+0x4d	retq

Filtering the trace

- Memory access events
- synchronization events

Symbol	Addr
count	0x40405c
mutex	0x404060
pthread_mutex_lock	0x401080
pthread_mutex_unlock	0x401050

Data Race Detection with Intel PT

TID	IP	Addr	Sym+off	Insn
20295	0	4011dc	routine_two+0xc	mov $0x404060, %rdi
20295	401080	4011e6	routine_two+0x16	callq 0xfffffffffffffe9a
20295	0	4011eb	routine_two+0x1b	movl 0x40405c, %ecx
20295	0	4011f5	routine_two+0x25	movl %ecx, 0x40405c
20295	0	4011fc	routine_two+0x2c	mov $0x404060, %rdi
20295	401050	401209	routine_two+0x39	callq 0xfffffffffffffe47
20294	0	40118c	routine_one+0xc	mov $0x404060, %rdi
20294	401080	401196	routine_one+0x16	callq 0xfffffffffffffeea
20294	0	40119b	routine_one+0x1b	movl 0x40405c, %ecx
20294	0	4011a5	routine_one+0x25	movl %ecx, 0x40405c
20294	0	4011ac	routine_one+0x2c	mov $0x404060, %rdi
20294	401050	4011b9	routine_one+0x39	callq 0xfffffffffffffe97

Vector Clocks

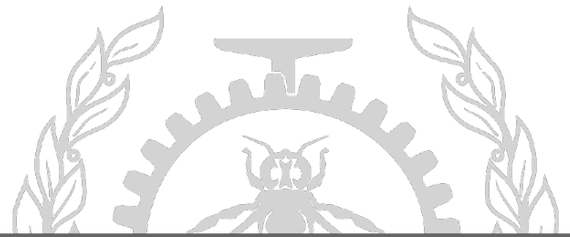
t1 (20294)

t1	t2
0	0

t2 (20295)

t1	t2
0	0

Shadow for 0x40405c		
TID	Vector Clock TID	WR



Data Race Detection with Intel PT

TID	IP	Addr	Sym+off	Insn
20295	0	4011dc	routine_two+0xc	mov $0x404060, %rdi
20295	401080	4011e6	routine_two+0x16	callq 0xfffffffffffffe9a
20295	0	4011eb	routine_two+0x1b	movl 0x40405c, %ecx
20295	0	4011f5	routine_two+0x25	movl %ecx, 0x40405c
20295	0	4011fc	routine_two+0x2c	mov $0x404060, %rdi
20295	401050	401209	routine_two+0x39	callq 0xfffffffffffffe47
20294	0	40118c	routine_one+0xc	mov $0x404060, %rdi
20294	401080	401196	routine_one+0x16	callq 0xfffffffffffffeea
20294	0	40119b	routine_one+0x1b	movl 0x40405c, %ecx
20294	0	4011a5	routine_one+0x25	movl %ecx, 0x40405c
20294	0	4011ac	routine_one+0x2c	mov $0x404060, %rdi
20294	401050	4011b9	routine_one+0x39	callq 0xfffffffffffffe97

Vector Clocks

t1 (20294)

t1	t2
0	0

t2 (20295)

t1	t2
0	0
0	1

Shadow for 0x40405c		
TID	Vector Clock TID	WR

Data Race Detection with Intel PT

TID	IP	Addr	Sym+off	Insn
20295	0	4011dc	routine_two+0xc	mov $0x404060, %rdi
20295	401080	4011e6	routine_two+0x16	callq 0xfffffffffffffe9a
20295	0	4011eb	routine_two+0x1b	movl 0x40405c, %ecx
20295	0	4011f5	routine_two+0x25	movl %ecx, 0x40405c
20295	0	4011fc	routine_two+0x2c	mov $0x404060, %rdi
20295	401050	401209	routine_two+0x39	callq 0xfffffffffffffe47
20294	0	40118c	routine_one+0xc	mov $0x404060, %rdi
20294	401080	401196	routine_one+0x16	callq 0xfffffffffffffeea
20294	0	40119b	routine_one+0x1b	movl 0x40405c, %ecx
20294	0	4011a5	routine_one+0x25	movl %ecx, 0x40405c
20294	0	4011ac	routine_one+0x2c	mov $0x404060, %rdi
20294	401050	4011b9	routine_one+0x39	callq 0xfffffffffffffe97

Vector Clocks

t1 (20294)

t1	t2
0	0

t2 (20295)

t1	t2
0	0
0	1
0	2

Shadow for 0x40405c		
TID	Vector Clock TID	WR
20295	(0,2)	R

Data Race Detection with Intel PT

TID	IP	Addr	Sym+off	Insn
20295	0	4011dc	routine_two+0xc	mov $0x404060, %rdi
20295	401080	4011e6	routine_two+0x16	callq 0xfffffffffffffe9a
20295	0	4011eb	routine_two+0x1b	movl 0x40405c, %ecx
20295	0	4011f5	routine_two+0x25	movl %ecx, 0x40405c
20295	0	4011fc	routine_two+0x2c	mov $0x404060, %rdi
20295	401050	401209	routine_two+0x39	callq 0xfffffffffffffe47
20294	0	40118c	routine_one+0xc	mov $0x404060, %rdi
20294	401080	401196	routine_one+0x16	callq 0xfffffffffffffeea
20294	0	40119b	routine_one+0x1b	movl 0x40405c, %ecx
20294	0	4011a5	routine_one+0x25	movl %ecx, 0x40405c
20294	0	4011ac	routine_one+0x2c	mov $0x404060, %rdi
20294	401050	4011b9	routine_one+0x39	callq 0xfffffffffffffe97

Vector Clocks

t1 (20294)

t1	t2
0	0

t2 (20295)

t1	t2
0	0
0	1
0	2
0	3

Different threads? No, so it's not a race

Shadow for 0x40405c		
TID	Vector Clock TID	WR
20295	(0,2)	R

Write by 20295

Data Race Detection with Intel PT

TID	IP	Addr	Sym+off	Insn
20295	0	4011dc	routine_two+0xc	mov $0x404060, %rdi
20295	401080	4011e6	routine_two+0x16	callq 0xfffffffffffffe9a
20295	0	4011eb	routine_two+0x1b	movl 0x40405c, %ecx
20295	0	4011f5	routine_two+0x25	movl %ecx, 0x40405c
20295	0	4011fc	routine_two+0x2c	mov $0x404060, %rdi
20295	401050	401209	routine_two+0x39	callq 0xfffffffffffffe47
20294	0	40118c	routine_one+0xc	mov $0x404060, %rdi
20294	401080	401196	routine_one+0x16	callq 0xfffffffffffffeea
20294	0	40119b	routine_one+0x1b	movl 0x40405c, %ecx
20294	0	4011a5	routine_one+0x25	movl %ecx, 0x40405c
20294	0	4011ac	routine_one+0x2c	mov $0x404060, %rdi
20294	401050	4011b9	routine_one+0x39	callq 0xfffffffffffffe97

Vector Clocks

t1 (20294)

t1	t2
0	0

t2 (20295)

t1	t2
0	0
0	1
0	2
0	3

Update the Shadow

Shadow for 0x40405c		
TID	Vector Clock TID	WR
20295	(0,3)	W

Data Race Detection with Intel PT

TID	IP	Addr	Sym+off	Insn
20295	0	4011dc	routine_two+0xc	mov $0x404060, %rdi
20295	401080	4011e6	routine_two+0x16	callq 0xfffffffffffffe9a
20295	0	4011eb	routine_two+0x1b	movl 0x40405c, %ecx
20295	0	4011f5	routine_two+0x25	movl %ecx, 0x40405c
20295	0	4011fc	routine_two+0x2c	mov $0x404060, %rdi
20295	401050	401209	routine_two+0x39	callq 0xfffffffffffffe47
20294	0	40118c	routine_one+0xc	mov $0x404060, %rdi
20294	401080	401196	routine_one+0x16	callq 0xfffffffffffffeea
20294	0	40119b	routine_one+0x1b	movl 0x40405c, %ecx
20294	0	4011a5	routine_one+0x25	movl %ecx, 0x40405c
20294	0	4011ac	routine_one+0x2c	mov $0x404060, %rdi
20294	401050	4011b9	routine_one+0x39	callq 0xfffffffffffffe97

Vector Clocks

t1 (20294)

t1	t2
0	0

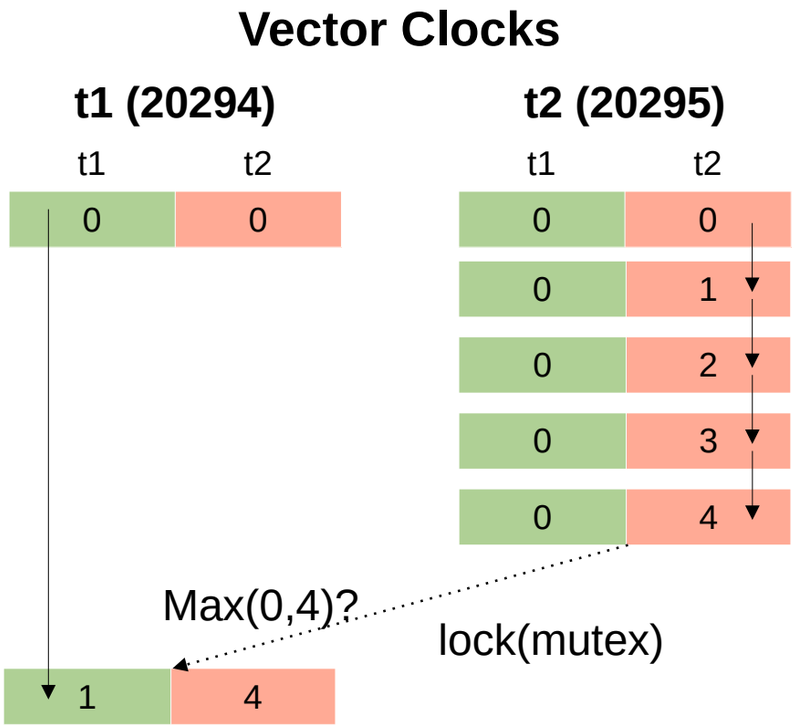
t2 (20295)

t1	t2
0	0
0	1
0	2
0	3
0	4

Shadow for 0x40405c		
TID	Vector Clock TID	WR
20295	(0,3)	W

Data Race Detection with Intel PT

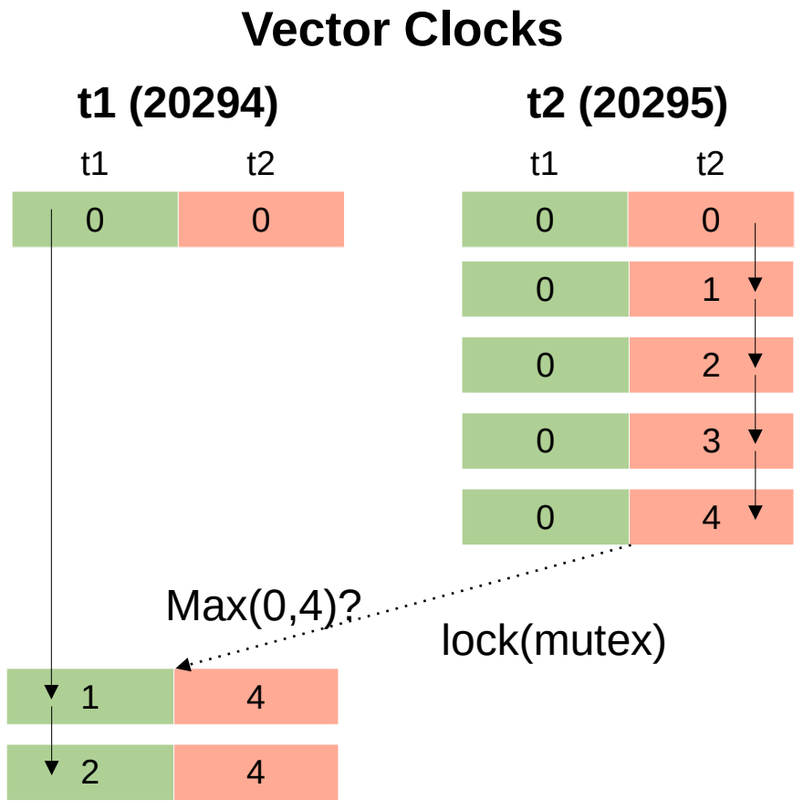
TID	IP	Addr	Sym+off	Insn
20295	0	4011dc	routine_two+0xc	mov $0x404060, %rdi
20295	401080	4011e6	routine_two+0x16	callq 0xfffffffffffffe9a
20295	0	4011eb	routine_two+0x1b	movl 0x40405c, %ecx
20295	0	4011f5	routine_two+0x25	movl %ecx, 0x40405c
20295	0	4011fc	routine_two+0x2c	mov $0x404060, %rdi
20295	401050	401209	routine_two+0x39	callq 0xfffffffffffffe47
20294	0	40118c	routine_one+0xc	mov $0x404060, %rdi
20294	401080	401196	routine_one+0x16	callq 0xfffffffffffffeea
20294	0	40119b	routine_one+0x1b	movl 0x40405c, %ecx
20294	0	4011a5	routine_one+0x25	movl %ecx, 0x40405c
20294	0	4011ac	routine_one+0x2c	mov $0x404060, %rdi
20294	401050	4011b9	routine_one+0x39	callq 0xfffffffffffffe97



Shadow for 0x40405c		
TID	Vector Clock TID	WR
20295	(0,3)	W

Data Race Detection with Intel PT

TID	IP	Addr	Sym+off	Insn
20295	0	4011dc	routine_two+0xc	mov $0x404060, %rdi
20295	401080	4011e6	routine_two+0x16	callq 0xfffffffffffffe9a
20295	0	4011eb	routine_two+0x1b	movl 0x40405c, %ecx
20295	0	4011f5	routine_two+0x25	movl %ecx, 0x40405c
20295	0	4011fc	routine_two+0x2c	mov $0x404060, %rdi
20295	401050	401209	routine_two+0x39	callq 0xfffffffffffffe47
20294	0	40118c	routine_one+0xc	mov $0x404060, %rdi
20294	401080	401196	routine_one+0x16	callq 0xfffffffffffffeea
20294	0	40119b	routine_one+0x1b	movl 0x40405c, %ecx
20294	0	4011a5	routine_one+0x25	movl %ecx, 0x40405c
20294	0	4011ac	routine_one+0x2c	mov $0x404060, %rdi
20294	401050	4011b9	routine_one+0x39	callq 0xfffffffffffffe97



Shadow for 0x40405c		
TID	Vector Clock TID	WR
20295	(0,3)	W

Read by 20294

Different threads? Yes

Is at least one of the accesses a write? Yes

Are they concurrent? No, so it's not a race

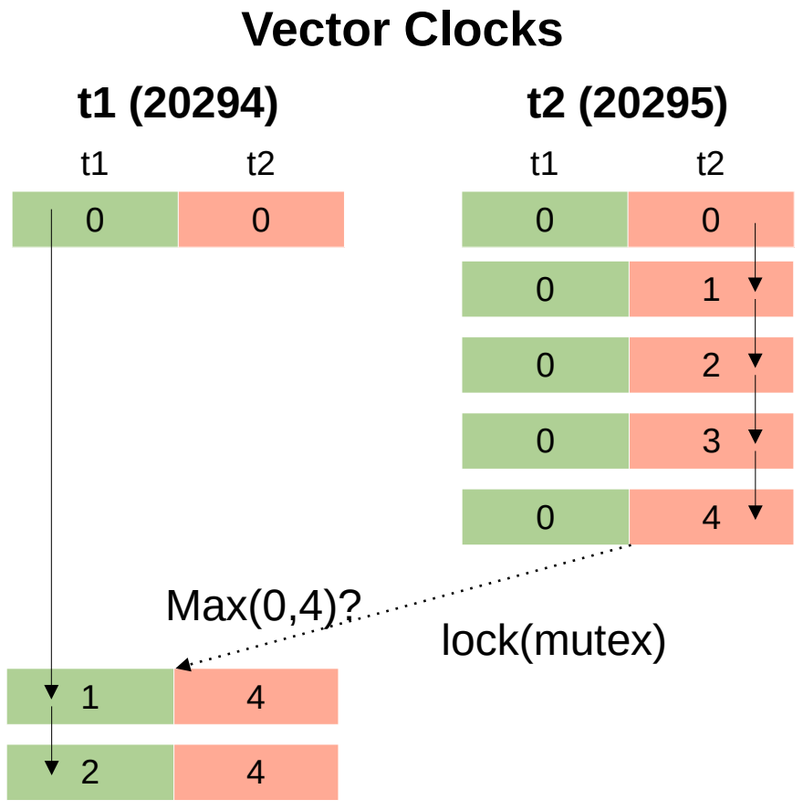
(0,3) < (2,4), happened before!

Data Race Detection with Intel PT

TID	IP	Addr	Sym+off	Insn
20295	0	4011dc	routine_two+0xc	mov $0x404060, %rdi
20295	401080	4011e6	routine_two+0x16	callq 0xfffffffffffffe9a
20295	0	4011eb	routine_two+0x1b	movl 0x40405c, %ecx
20295	0	4011f5	routine_two+0x25	movl %ecx, 0x40405c
20295	0	4011fc	routine_two+0x2c	mov $0x404060, %rdi
20295	401050	401209	routine_two+0x39	callq 0xfffffffffffffe47
20294	0	40118c	routine_one+0xc	mov $0x404060, %rdi
20294	401080	401196	routine_one+0x16	callq 0xfffffffffffffeea
20294	0	40119b	routine_one+0x1b	movl 0x40405c, %ecx
20294	0	4011a5	routine_one+0x25	movl %ecx, 0x40405c
20294	0	4011ac	routine_one+0x2c	mov $0x404060, %rdi
20294	401050	4011b9	routine_one+0x39	callq 0xfffffffffffffe97

Update the Shadow

Shadow for 0x40405c		
TID	Vector Clock TID	WR
20294	(2,4)	R



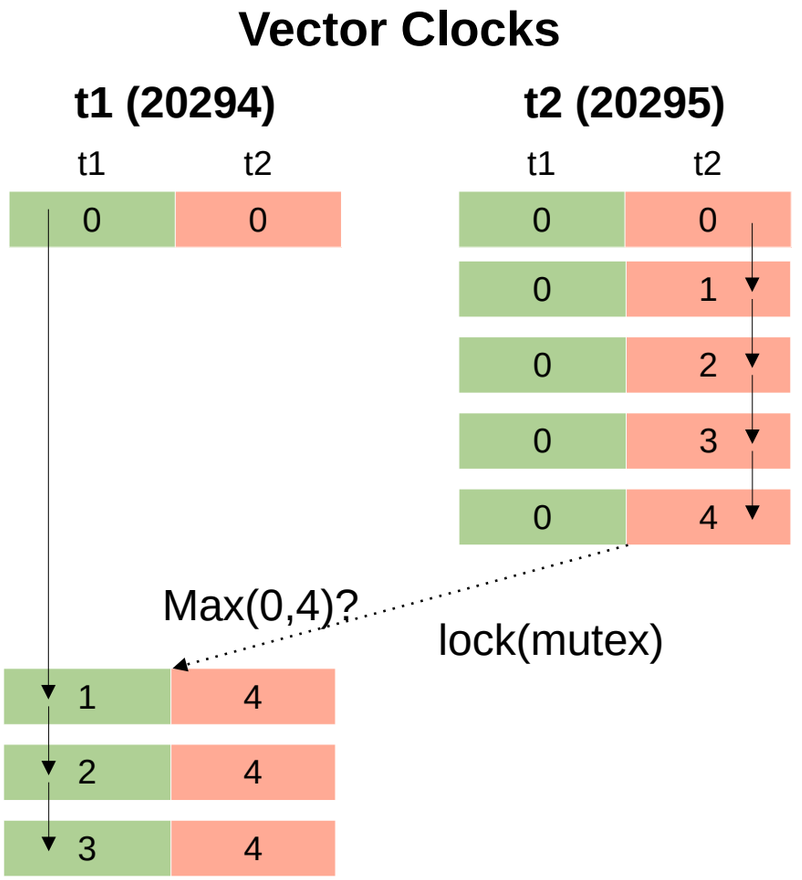
Data Race Detection with Intel PT

TID	IP	Addr	Sym+off	Insn
20295	0	4011dc	routine_two+0xc	mov $0x404060, %rdi
20295	401080	4011e6	routine_two+0x16	callq 0xfffffffffffffe9a
20295	0	4011eb	routine_two+0x1b	movl 0x40405c, %ecx
20295	0	4011f5	routine_two+0x25	movl %ecx, 0x40405c
20295	0	4011fc	routine_two+0x2c	mov $0x404060, %rdi
20295	401050	401209	routine_two+0x39	callq 0xfffffffffffffe47
20294	0	40118c	routine_one+0xc	mov $0x404060, %rdi
20294	401080	401196	routine_one+0x16	callq 0xfffffffffffffeea
20294	0	40119b	routine_one+0x1b	movl 0x40405c, %ecx
20294	0	4011a5	routine_one+0x25	movl %ecx, 0x40405c
20294	0	4011ac	routine_one+0x2c	mov $0x404060, %rdi
20294	401050	4011b9	routine_one+0x39	callq 0xfffffffffffffe97

Different threads? No, so it's not a race

Shadow for 0x40405c		
TID	Vector Clock TID	WR
20294	(2,4)	R

Write by 20294

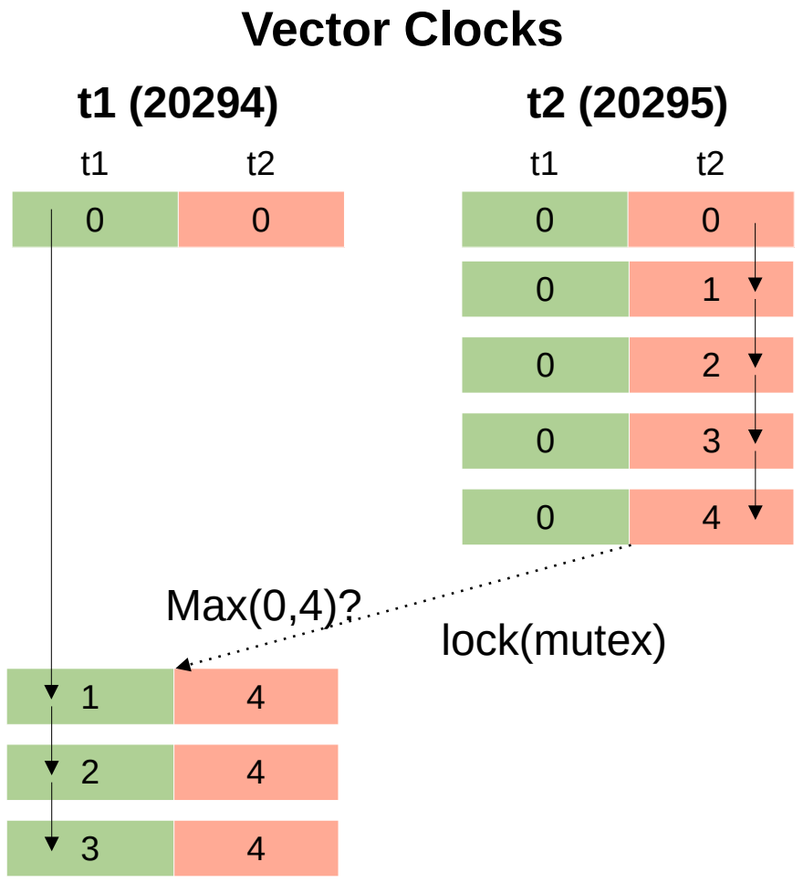


Data Race Detection with Intel PT

TID	IP	Addr	Sym+off	Insn
20295	0	4011dc	routine_two+0xc	mov $0x404060, %rdi
20295	401080	4011e6	routine_two+0x16	callq 0xfffffffffffffe9a
20295	0	4011eb	routine_two+0x1b	movl 0x40405c, %ecx
20295	0	4011f5	routine_two+0x25	movl %ecx, 0x40405c
20295	0	4011fc	routine_two+0x2c	mov $0x404060, %rdi
20295	401050	401209	routine_two+0x39	callq 0xfffffffffffffe47
20294	0	40118c	routine_one+0xc	mov $0x404060, %rdi
20294	401080	401196	routine_one+0x16	callq 0xfffffffffffffeea
20294	0	40119b	routine_one+0x1b	movl 0x40405c, %ecx
20294	0	4011a5	routine_one+0x25	movl %ecx, 0x40405c
20294	0	4011ac	routine_one+0x2c	mov $0x404060, %rdi
20294	401050	4011b9	routine_one+0x39	callq 0xfffffffffffffe97

Update the Shadow

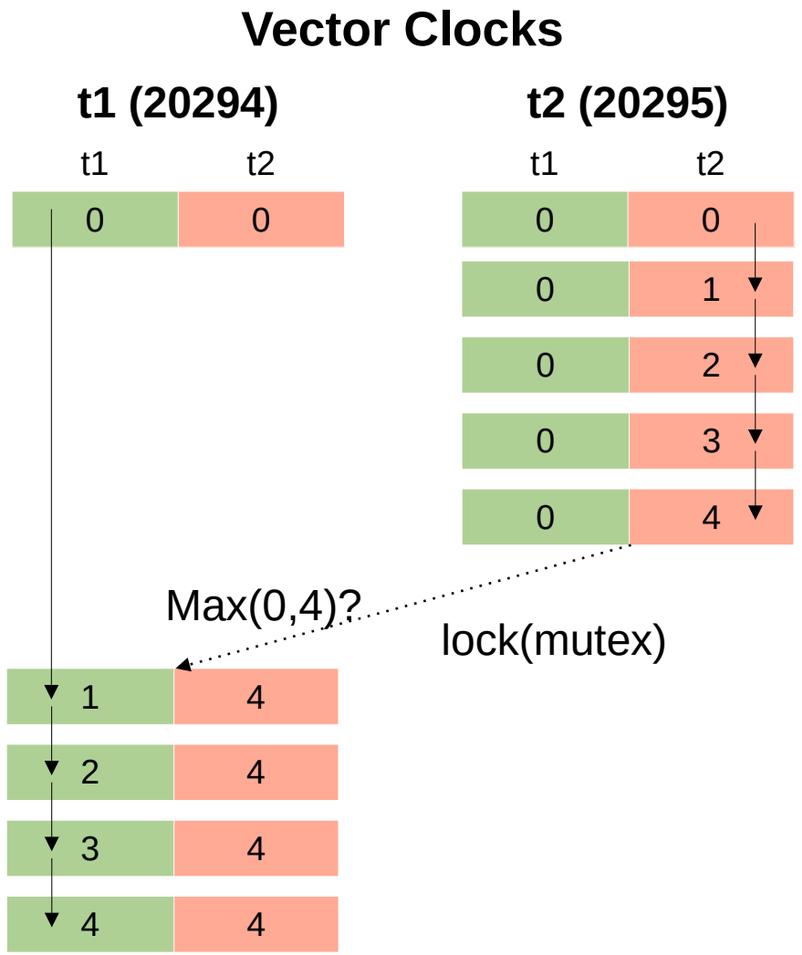
Shadow for 0x40405c		
TID	Vector Clock TID	WR
20294	(3,4)	R



Data Race Detection with Intel PT

TID	IP	Addr	Sym+off	Insn
20295	0	4011dc	routine_two+0xc	mov $0x404060, %rdi
20295	401080	4011e6	routine_two+0x16	callq 0xfffffffffffffe9a
20295	0	4011eb	routine_two+0x1b	movl 0x40405c, %ecx
20295	0	4011f5	routine_two+0x25	movl %ecx, 0x40405c
20295	0	4011fc	routine_two+0x2c	mov $0x404060, %rdi
20295	401050	401209	routine_two+0x39	callq 0xfffffffffffffe47
20294	0	40118c	routine_one+0xc	mov $0x404060, %rdi
20294	401080	401196	routine_one+0x16	callq 0xfffffffffffffeea
20294	0	40119b	routine_one+0x1b	movl 0x40405c, %ecx
20294	0	4011a5	routine_one+0x25	movl %ecx, 0x40405c
20294	0	4011ac	routine_one+0x2c	mov $0x404060, %rdi
20294	401050	4011b9	routine_one+0x39	callq 0xfffffffffffffe97

Shadow for 0x40405c		
TID	Vector Clock TID	WR
20294	(3,4)	R



Demo

Two sample C applications

Demo

Results for routines with 1 billion loops

Sample code with synchronization

	Executable size	Execution time
Native	19240B	104.8 Sec
TSan	×63.8	×5.6
Proposed	×1	×1.4

Sample code without synchronization

	Executable size	Execution time
Native	18448B	2.67 Sec
TSan	×66.5	×226.2
Proposed	×1	×1.08

Our approach:

price to pay for the much lower runtime overhead: the **analysis time** and **trace size**

Conclusion & Future Work

- Some data race detection coverage with zero code instrumentation!
  - TSan totally relies on code instrumentation
- Will add some instrumentation to extend data race detection coverage
  - A hybrid tool that benefits from hardware tracing to mininize the need for code instrumentation
- Will also investigate other possible hardware-tracing-based opportunities for detecting other memory bugs

Questions?

farzam.dorostkar@polymtl.ca