Seminararbeit

Dynamische Programmanalysen für nebenläufige Programme - Data Race Prediction mit TSan

Frank Ling

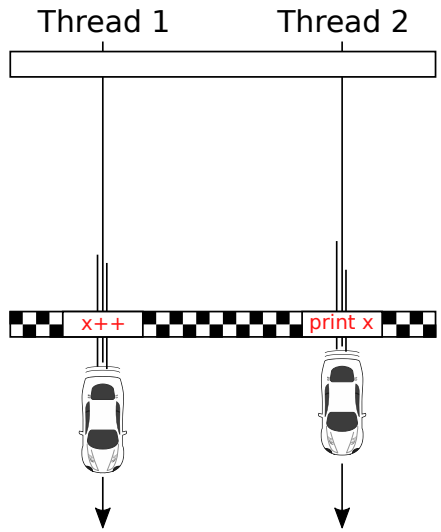
June 13, 2023

Table of contents

Hochschule Karlsruhe
University of
Applied Sciences

Introduction

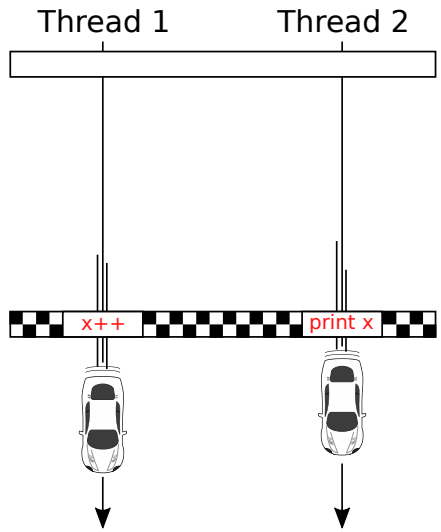
- What are data races?



Source: https://programming.guide/go/data-races-explained.html

Introduction

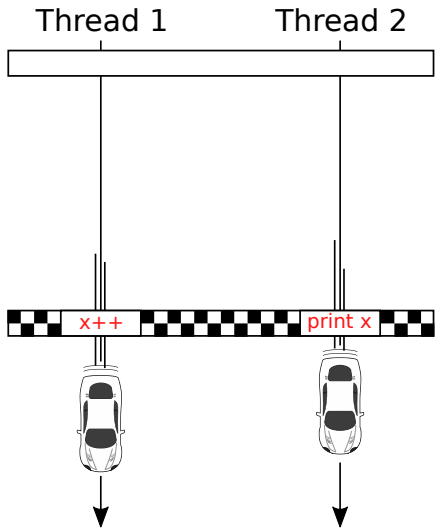
- What are data races?
- Why fix data races?



Source: https://programming.guide/go/data-races-explained.html

Introduction

- What are data races?
- Why fix data races?
- How to detect data races?



Source: https://programming.guide/go/data-races-explained.html

Background


```
1  int x;
2  pthread_mutex_t y;
3
4  void *Thread1(void *x) {
5          x++;
6          pthread_mutex_lock(&y);
7          pthread_mutex_unlock(&y);
8          return NULL;
9  }
10
11 void *Thread2(void *x) {
12         pthread_mutex_lock(&y);
13         x--;
14         pthread_mutex_unlock(&y);
15         return NULL;
16 }
```

Listing: program exhibiting a data race

Background


```
1  int x;
2  pthread_mutex_t y;
3
4  void *Thread1(void *x) {
5          x++;
6          pthread_mutex_lock(&y);
7          pthread_mutex_unlock(&y);
8          return NULL;
9  }
10
11 void *Thread2(void *x) {
12         pthread_mutex_lock(&y);
13         x--;
14         pthread_mutex_unlock(&y);
15         return NULL;
16 }
```

Listing: program exhibiting a data race

	1#	2#
1.	w(x)	
2.	acq(y)	
3.	rel(y)	
4.		acq(y)
5.		w(x)
6.		rel(y)

Table: obtained trace

	1#	2#
4.		acq(y)
5.		w(x)
1.	w(x)	
6.		rel(y)
2.	acq(y)	
3.	rel(y)	

Table: Trace 1 reordered

Data race! $\Leftarrow$

	1#	2#
4.		acq(y)
5.		w(x)
1.	w(x)	
6.		rel(y)
2.	acq(y)	
3.	rel(y)	

Table: Trace 1 reordered

Background

- Dynamic data race prediction

Background

- Dynamic data race prediction
- Vector clocks

Background

- Dynamic data race prediction
- Vector clocks
- Epochs

Background

- Dynamic data race prediction
- Vector clocks
- Epochs
- Lamport's HB relation

Lamport's HB relation

Program order condition

- For events e and f in same thread, if e appears before f in trace then $e <_{HB} f$
- inc(V, j) applied after processing event in same thread

Critical section

- For rel(y) in thread i and acq(y) in thread j if rel(y) appears beofre acq(y) then rel$(y) <_{HB}$ acq(y)
- sync(V_1, V_2) is applied after rel(y) in thread

FastTrack and TSan V2

FastTrack

- Lamport's HB relation
- epoch-based
- semi-adaptive
  - dynamically sized epochs expensive
  - initially epochs only
  - read $\rightarrow$ VC, stays VC after (subsequent concurrent reads frequent)
  - writes are epochs only (all writes are totally ordered)

ThreadSanitizer (TSan) V2

- slightly modified version of FastTrack
- shadow memory

Shadow memory

- Shadow word (64 bits)

TID (Thread Id)	16 bits
Scalar Clock	42 bits
IsWrite	1 bit
Access Size	2 bits
Address Offset	3 bits

Shadow memory

- Shadow word (64 bits)

TID (Thread Id)	16 bits
Scalar Clock	42 bits
IsWrite	1 bit
Access Size	2 bits
Address Offset	3 bits

- **N** shadows words

  ↓

  application word (memory location)

Shadow memory

- Shadow word (64 bits)

TID (Thread Id)	16 bits
Scalar Clock	42 bits
IsWrite	1 bit
Access Size	2 bits
Address Offset	3 bits

- **N** shadows words

  ↓

  application word (memory location)

- N is configurable (2, 4, 8) but default is 4

  ⇓

  For every memory location 1 write and up to concurrent 3 reads are stored

Shadow memory

- Shadow word (64 bits)

TID (Thread Id)	16 bits
Scalar Clock	42 bits
IsWrite	1 bit
Access Size	2 bits
Address Offset	3 bits

- **N** shadows words

  ↓

  application word (memory location)

- N is configurable (2, 4, 8) but default is 4

  ⇓

  For every memory location 1 write and up to concurrent 3 reads are stored
- for every subsequent concurrent read a random read in shadow memory will be evicted

Limitations

- Unsound due to epochs

```
1  void thread1() {
2          y := x+5;
3  }
4
5  void thread2() {
6          if (y == 5)
7                  x := 10;
8          else
9                  while(true);
10 }
```

Listing: program showing unordered writes

	1#	2#
1.	r(x)	
2.	w(y)	
3.		r(y)
4.		w(x)

Table: Obtained trace

Limitations

- Incomplete due to:
  - HB relation (see first example)
  - shadow memory

Conclusion

- no race detection tool right now is entirely complete and sound
- FastTrack and TSan produce good results even though missing data races under certain circumstances
- they are reasonably efficient compared to other data race detection tools