

Seminar WS25/26

Data Race Detection

Demo

```
int x;
```

```
int x;
```

⋮

Thread 1

Thread 2

```
int x;
```

⋮

Thread 1

```
x = x + 1;
```

Thread 2

```
x = x + 1;
```

```
int x;
```

⋮

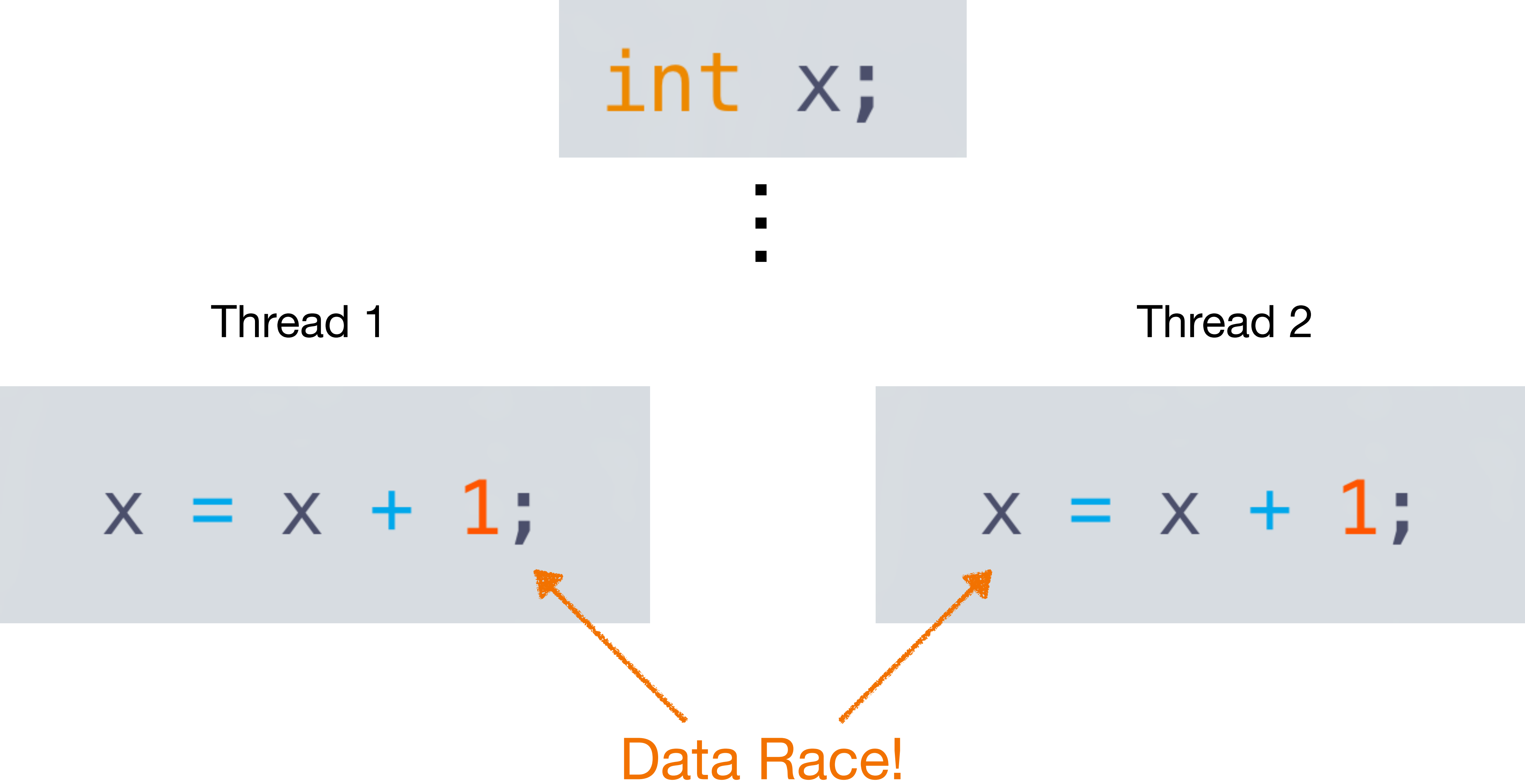
Thread 1

```
x = x + 1;
```

Thread 2

```
x = x + 1;
```

Data Race!



The diagram illustrates a data race between two threads, Thread 1 and Thread 2, accessing a shared variable `x`. At the top, a gray box contains the declaration `int x;`. Below it, three vertical dots indicate shared memory. Thread 1 and Thread 2 are shown as separate gray boxes, each containing the statement `x = x + 1;`. Two orange arrows originate from the text 'Data Race!' at the bottom and point to the `x` variable in both thread boxes, highlighting the conflict where both threads attempt to modify the same memory location simultaneously.

Thread 1



Thread 2

```
std::mutex mtx;  
int x;
```

⋮

Thread 1

Thread 2

```
std::mutex mtx;  
int x;
```

▪
▪
▪

Thread 1

```
mtx.lock();  
x = x + 1;  
mtx.unlock();
```

Thread 2

```
mtx.lock();  
x = x + 1;  
mtx.unlock();
```

```
std::mutex mtx;  
int x;
```

⋮

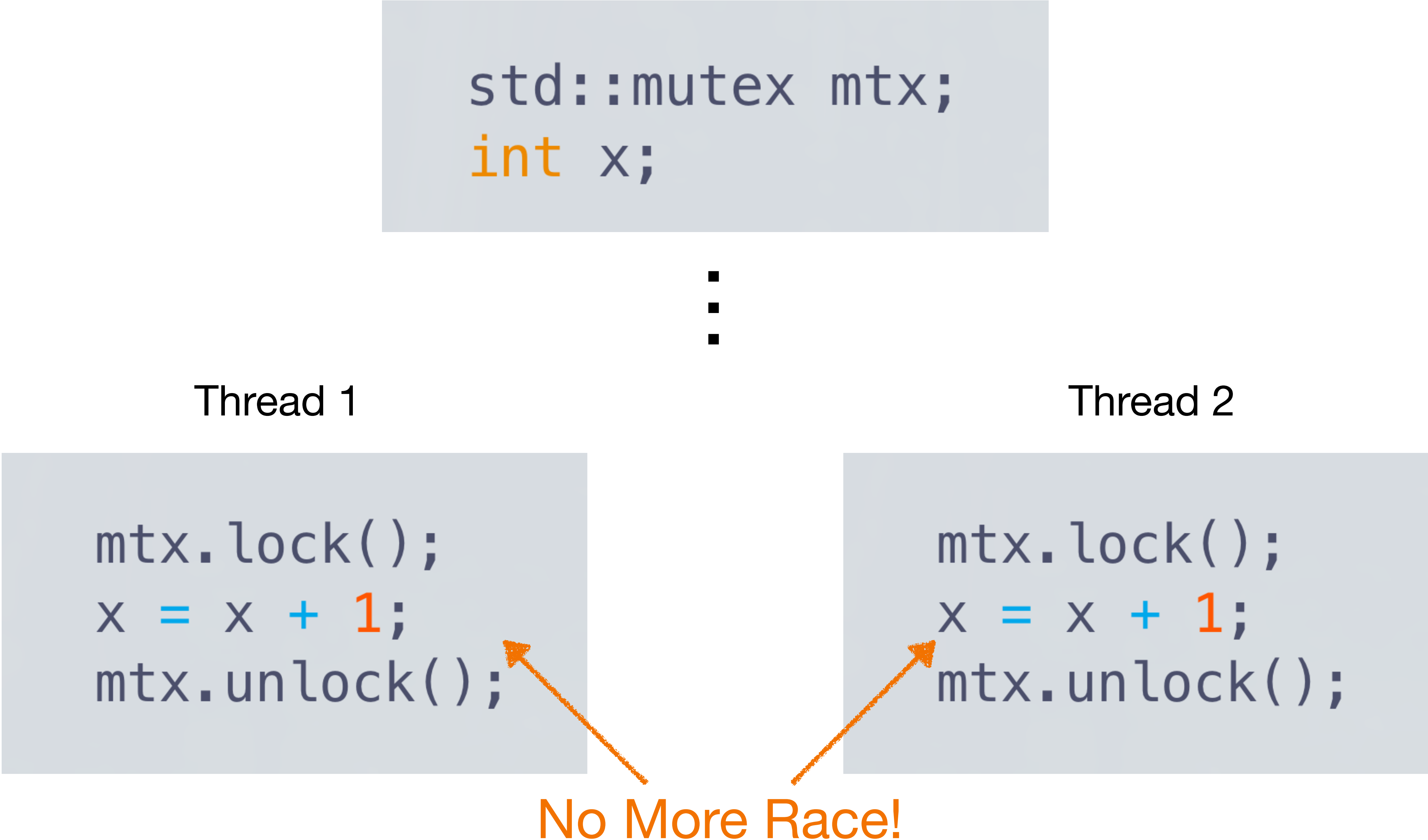
Thread 1

```
mtx.lock();  
x = x + 1;  
mtx.unlock();
```

Thread 2

```
mtx.lock();  
x = x + 1;  
mtx.unlock();
```

No More Race!



```
int x;
```

⋮

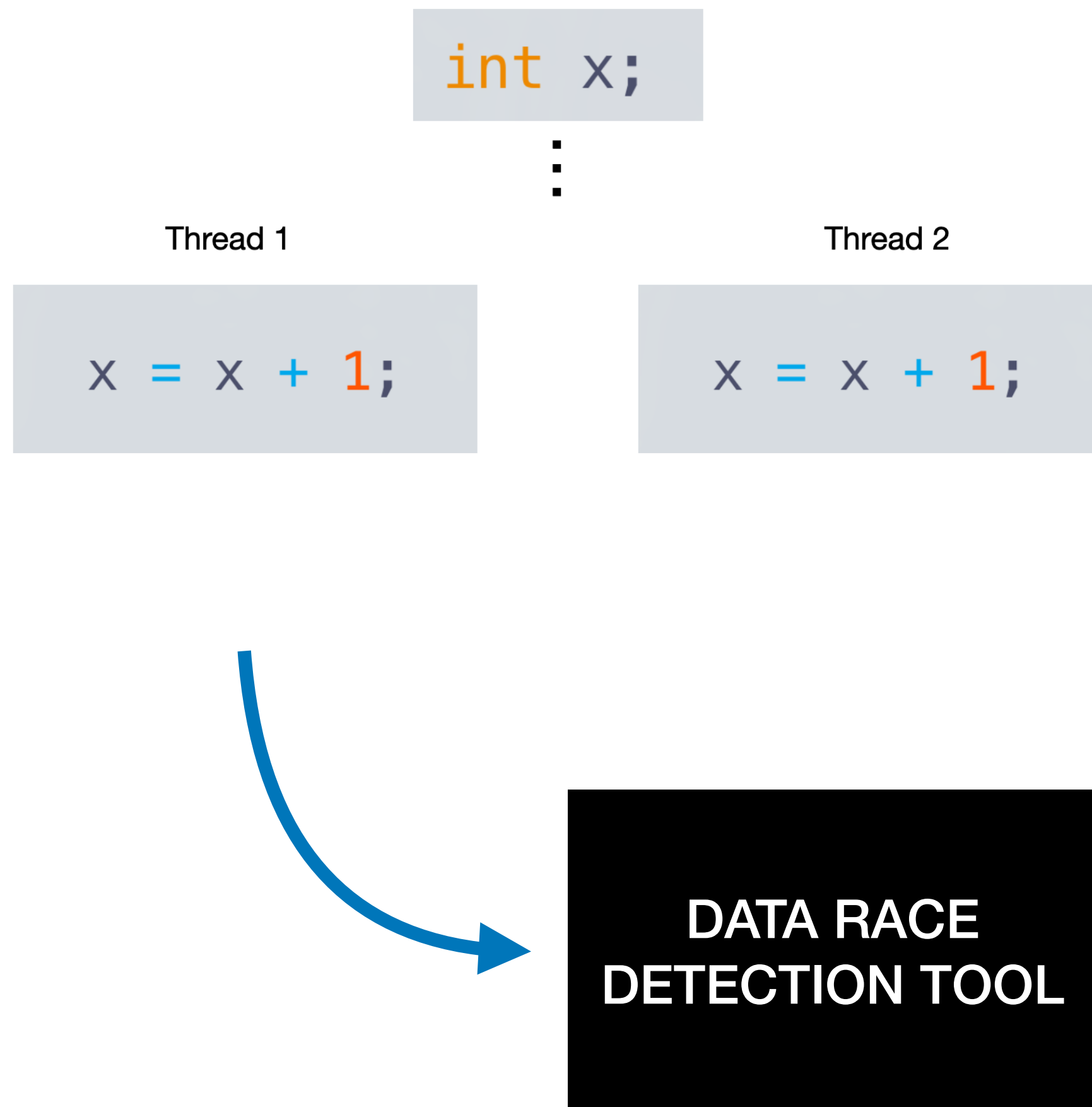
Thread 1

```
x = x + 1;
```

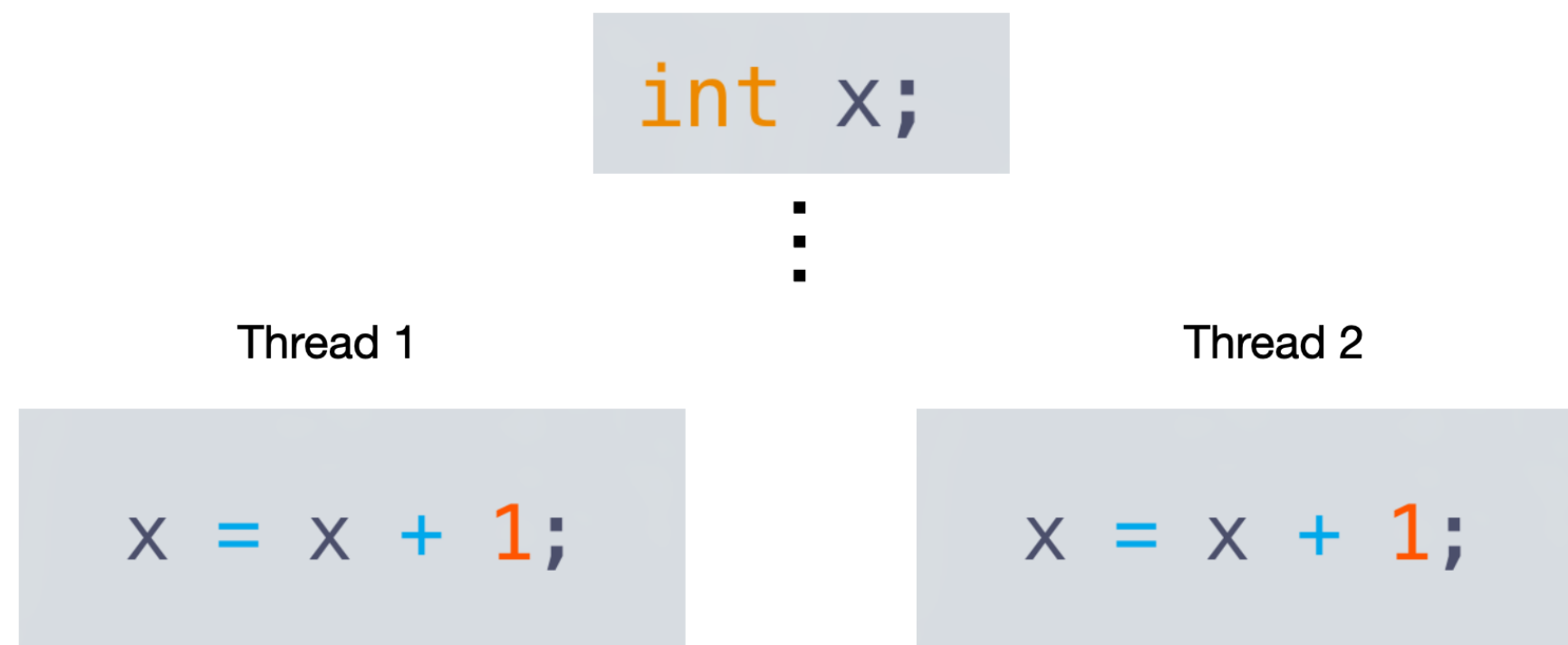
Thread 2

```
x = x + 1;
```

Hard to stop!



Hard to stop!



Hard to stop!

DATA RACE
DETECTION TOOL

```
WARNING: ThreadSanitizer: data race (pid=12812)
Write of size 4 at 0x000104530170 by thread T2:
  #0 writer() concurrent.cpp:8 (conc:arm64+0x100002478)
  #1 void* std::__1::__thread_proxy[abi:ne190102]<std::__1::tuple<std::__1::unique_ptr<std::__1::thr

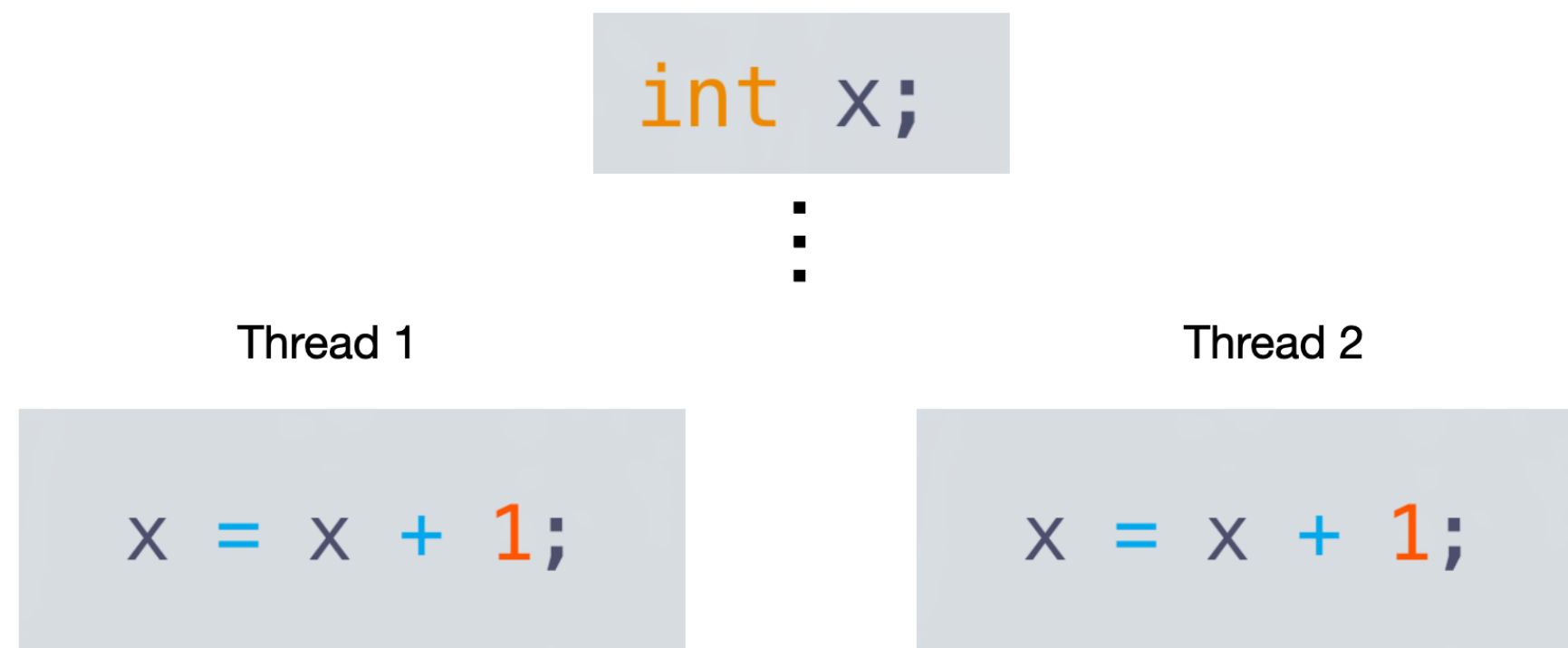
Previous write of size 4 at 0x000104530170 by thread T1:
  #0 writer() concurrent.cpp:8 (conc:arm64+0x100002478)
  #1 void* std::__1::__thread_proxy[abi:ne190102]<std::__1::tuple<std::__1::unique_ptr<std::__1::thr

Location is global 'x' at 0x000104530170 (conc+0x100008170)

Thread T2 (tid=23440608, running) created by main thread at:
  #0 pthread_create <null> (libclang_rt.tsan_osx_dynamic.dylib:arm64+0x37040)
  #1 std::__1::thread::thread<void (&)(), 0>(void (&)()) thread.h:217 (conc:arm64+0x100002be0)
  #2 main concurrent.cpp:18 (conc:arm64+0x10000252c)

Thread T1 (tid=23440607, finished) created by main thread at:
  #0 pthread_create <null> (libclang_rt.tsan_osx_dynamic.dylib:arm64+0x37040)
  #1 std::__1::thread::thread<void (&)(), 0>(void (&)()) thread.h:217 (conc:arm64+0x100002be0)
  #2 main concurrent.cpp:18 (conc:arm64+0x10000252c)

SUMMARY: ThreadSanitizer: data race concurrent.cpp:8 in writer()
```



Hard to stop!

DATA RACE
DETECTION TOOL

```
WARNING: ThreadSanitizer: data race (pid=12812)
Write of size 4 at 0x000104530170 by thread T2:
  #0 writer() concurrent.cpp:8 (conc:arm64+0x100002478)
  #1 void* std::__1::__thread_proxy[abi:ne190102]<std::__1::tuple<std::__1::unique_ptr<std::__1::thr

Previous write of size 4 at 0x000104530170 by thread T1:
  #0 writer() concurrent.cpp:8 (conc:arm64+0x100002478)
  #1 void* std::__1::__thread_proxy[abi:ne190102]<std::__1::tuple<std::__1::unique_ptr<std::__1::thr

Location is global 'x' at 0x000104530170 (conc+0x100008170)

Thread T2 (tid=23440608, running) created by main thread at:
  #0 pthread_create <null> (libclang_rt.tsan_osx_dynamic.dylib:arm64+0x37040)
  #1 std::__1::thread::thread<void (&)(), 0>(void (&)()) thread.h:217 (conc:arm64+0x100002be0)
  #2 main concurrent.cpp:18 (conc:arm64+0x10000252c)

Thread T1 (tid=23440607, finished) created by main thread at:
  #0 pthread_create <null> (libclang_rt.tsan_osx_dynamic.dylib:arm64+0x37040)
  #1 std::__1::thread::thread<void (&)(), 0>(void (&)()) thread.h:217 (conc:arm64+0x100002be0)
  #2 main concurrent.cpp:18 (conc:arm64+0x10000252c)

SUMMARY: ThreadSanitizer: data race concurrent.cpp:8 in writer()
```

How does this work?!?

DATA RACE
DETECTION TOOL

← How does this work?!?

Well, you tell me!

Seminar Contents

← How does this work?!?
Well, you tell me!

Seminar Contents

1. Read Paper, understand Technique

← How does this work?!?

Well, you tell me!

Seminar Contents

1. Read Paper, understand Technique
2. Decide what to present.

← How does this work?!?

Well, you tell me!

Seminar Contents

1. Read Paper, understand Technique
2. Decide what to present.
3. Write Seminar Paper

← How does this work?!?

Well, you tell me!

Seminar Contents

1. Read Paper, understand Technique
2. Decide what to present.
3. Write Seminar Paper
4. Give a Talk

1. Read Paper, understand Technique

1. Read Paper, understand Technique



How does this work?!?

1. Read Paper, understand Technique



How does this work?!?

There is this Thing that they don't explain, but I don't understand?

1. Read Paper, understand Technique



How does this work?!?

There is this Thing that they don't explain, but I don't understand?

1. Open the Rabbit-Hole: Find Papers for the Thing.

1. Read Paper, understand Technique



How does this work?!?

There is this Thing that they don't explain, but I don't understand?

1. Open the Rabbit-Hole: Find Papers for the Thing.
2. This can contribute to your Background/Related Work Section!

2. Decide what to present!

2. Decide what to present!

Less space, different content!

2. Decide what to present!

Less space, different content!

1. What do you include?

2. Decide what to present!

Less space, different content!

1. What do you include?
2. Are there Things missing in the paper that you want to include?

3. Write Seminar Paper

Structure

3. Write Seminar Paper

Structure

1. Introduction

- Think: Lead the reader to your specific topic.
- Reader: A first year student should understand this!

3. Write Seminar Paper

Structure

1. Introduction

- Think: Lead the reader to your specific topic.
- Reader: A first year student should understand this!

2. Background

- What does the reader need to know to understand this seminar paper?

3. Write Seminar Paper

Structure

1. Introduction
 - Think: Lead the reader to your specific topic.
 - Reader: A first year student should understand this!
2. Background
 - What does the reader need to know to understand this seminar paper?
3. Your Content.
 - Think: What sections make sense? What content to move where?

3. Write Seminar Paper

Structure

1. Introduction
 - Think: Lead the reader to your specific topic.
 - Reader: A first year student should understand this!
2. Background
 - What does the reader need to know to understand this seminar paper?
3. Your Content.
 - Think: What sections make sense? What content to move where?
4. Related Work
 - Differences/Commonalities with closest works from the literature

4. Give a Talk

4. Give a Talk



Todos:

- Wochentag + Uhrzeit Vorträge
- Paper Zuweisen
- Betreuer Zuweisen

Papers

Static Tools

1. **[Engler, Ashcraft 2003]**
RacerX: Effective, Static Detection of Race Conditions and Deadlocks (Lockset Analysis)
<https://dl.acm.org/doi/10.1145/1165389.945468>
2. **[Naik, Aiken, Whaley 2006]**
Effective Static Race Detection for Java (Java-Lockset analysis)
<https://dl.acm.org/doi/10.1145/1133981.1134018>
3. **[Young, Jhala, Lerner 2007]**
RELAY: Static Race Detection on Millions of Lines of Code (Follow Up Work of RacerX)
<https://dl.acm.org/doi/10.1145/1287624.1287654>
4. **[Vojdani, Apinis, Rotov, Seidl, Vene, Vogler 2016]**
Static Race Detection for Device Drivers: The Goblint Approach
<https://dl.acm.org/doi/10.1145/2970276.2970337>
5. **[Blackshear, Gorogiannis, O'Hearn, Sergey 2018]**
RacerD: Compositional Static Race Detection (Infer's DRC)
<https://dl.acm.org/doi/10.1145/3276514>

Papers

Type Systems

6. **[Poyapati, Rinard 2001]**

A parameterized type system for race-free Java programs (Uniqueness)

<https://dl.acm.org/doi/10.1145/504311.504287>

7. **[Haller, Odersky 2010]**

Capabilities for Uniqueness and Borrowing (Borrowing)

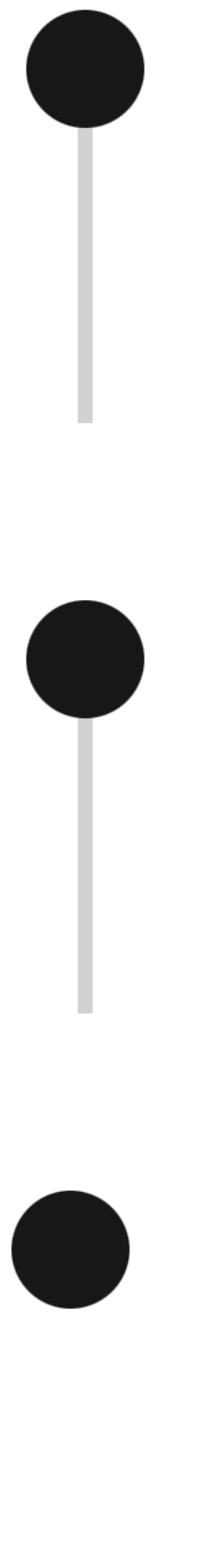
https://link.springer.com/chapter/10.1007/978-3-642-14107-2_17

Papers

Dynamic Tools

8. **[Savage, Burrows, Nelson, Sobalvarro, Anderson 1997]**
Eraser: A Dynamic Data Race Detector for Multi-Threaded Programs (Locksets)
<https://dl.acm.org/doi/10.1145/265924.265927>
9. **[O'Callahan, Choi 2003]**
Hybrid Dynamic Data Race Detection. (Extending on Locksets with HB-relations)
<https://dl.acm.org/doi/10.1145/966049.781528>
10. **[Pozniansky, Schuster 2003]**
MultiRace: Efficient on-the-fly data race detection in multithreaded C++ programs. (Vector Clocks)
<https://dl.acm.org/doi/10.1145/781498.781529>
11. **[Flanagan, Freund 2009]**
FastTrack: Efficient and precise dynamic race detection. (Epochs instead of VC)
<https://dl.acm.org/doi/10.1145/1543135.1542490>

Timeline

- 
- 16/01/26
Hand in Paper for Peer-Review
 - 30/01/2026
Review Paper
 - 08/02/2026
Hand in Final Version

TBD: Talks