RaceMob: Crowdsourced Data Race Detection

Baris Kasikci, Cristian Zamfir, and George Candea

School of Computer & Communication Sciences

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Data Races

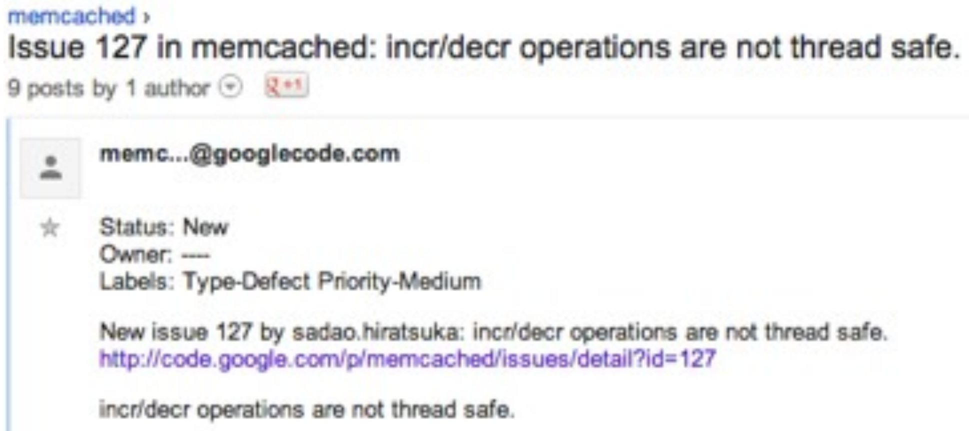
- Accesses to shared memory location
  - *By multiple threads*
  - *At least one of the accesses is a write*
  - *Synchronization operations do not enforce an order among the accesses*

`shared x`

Thread 1	**Thread 2**
`x = 1`	`x = 2`

Spectrum of Data Races

Kept for performance



memcached ›

Issue 127 in memcached: incr/decr operations are not thread safe.

9 posts by 1 author

memc...@googlecode.com

Status: New
Owner: ----
Labels: Type-Defect Priority-Medium

New issue 127 by sadao.hiratsuka: incr/decr operations are not thread safe.
http://code.google.com/p/memcached/issues/detail?id=127

incr/decr operations are not thread safe.

???

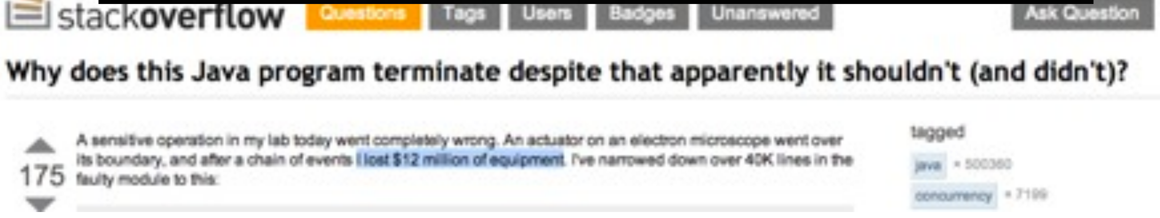
Caused massive losses

An Investigation of the Therac-25 Accidents

Nancy
Clark

Reprint

2003 Blackout



Why does this Java program terminate despite that apparently it shouldn't (and didn't)?

175

A sensitive operation in my lab today went completely wrong. An actuator on an electron microscope went over its boundary, and after a chain of events I lost $12 million of equipment. I've narrowed down over 40K lines in the faulty module to this:

tagged

java × 500360

concurrency × 7199

Pitfalls

- Programs with data races are incorrect according to POSIX & C/C++ standards
- Compilers can break correctness of programs with data races [HotPar'11]
    - *Harmless data races can become harmful*

Developers need to know every true data race

How to Find All Data Races?

- Static race detectors
  - ***Full path analysis*** ✓
  - ***Cheap*** *(0 runtime overhead)*✓
  - ***Few false negatives*** ✓
  - ***Many false positives*** *(~80%)*✗

- Dynamic race detectors
  - ***Per-run analysis*** ✗
  - ***Expensive*** *(≤ 200x)*✗
  - ***Many false negatives*** ✗
  - ***Few false positives*** *(~0%)* ✓

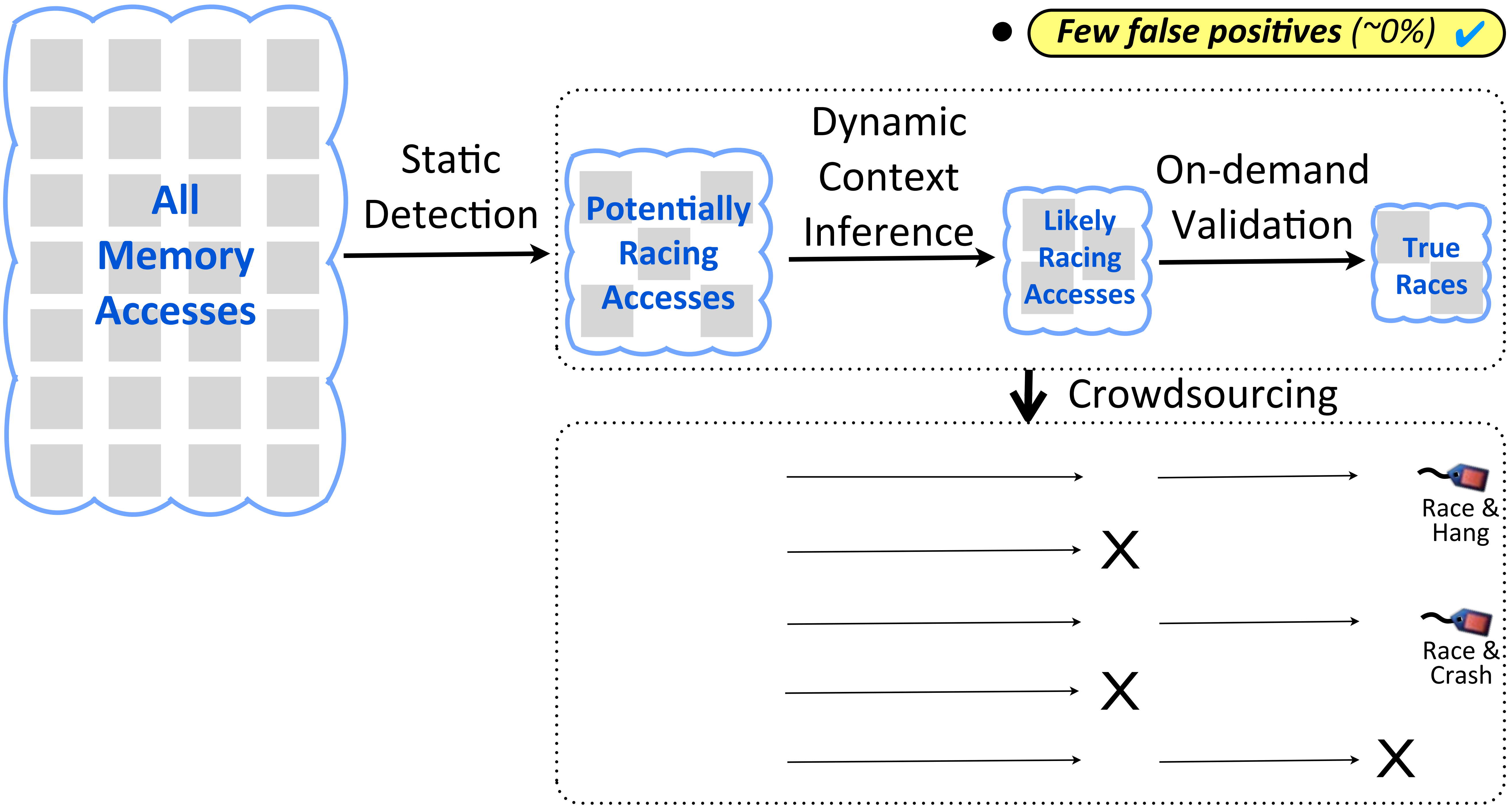
Existing detectors are not practical

How to Find All Data Races?

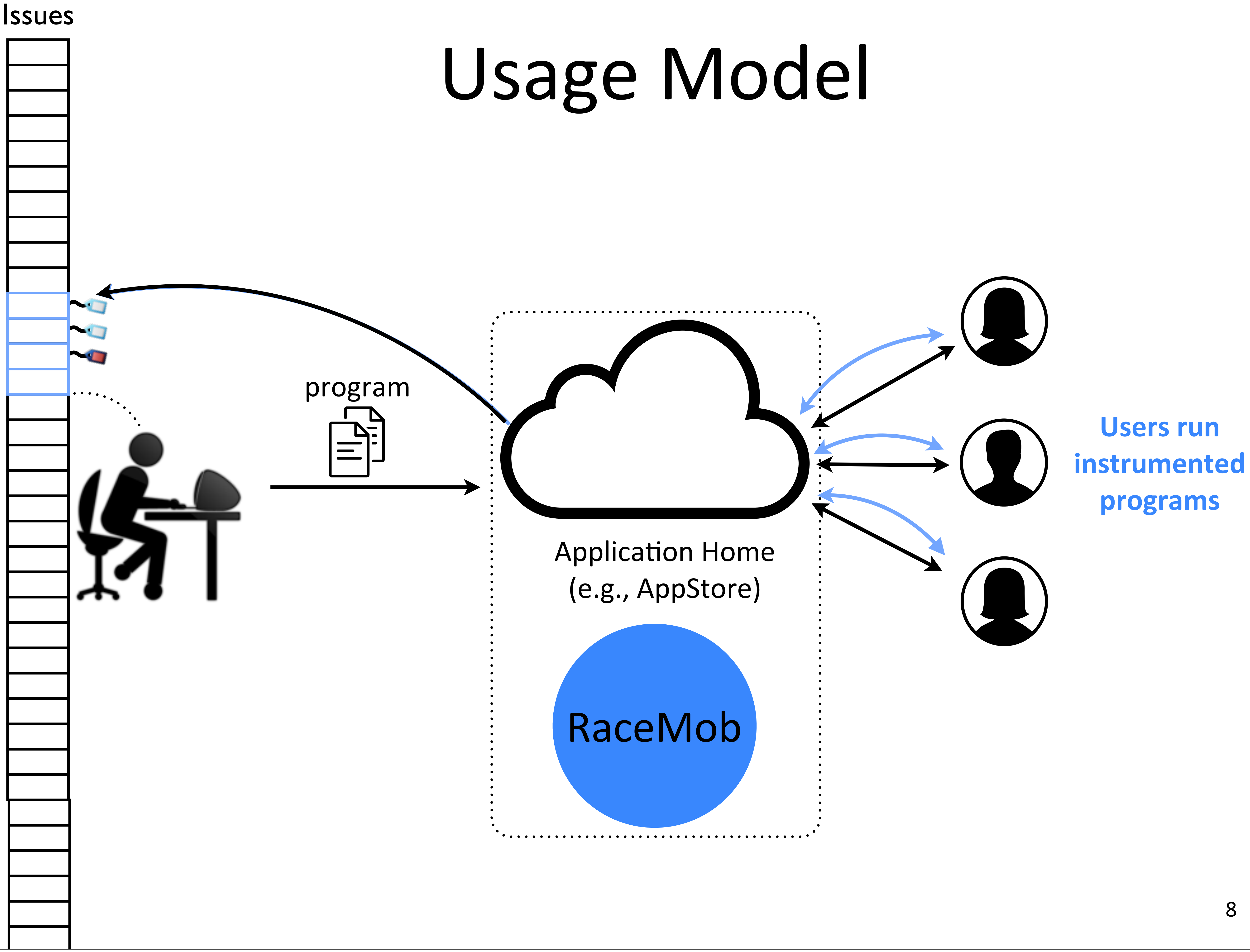
- ***Full path analysis*** ✓
- ***Cheap*** *(0 runtime overhead)*✓
- ***Few false negatives*** ✓
- ***Few false positives*** *(~0%)* ✓

RaceMob

- ***Full path analysis*** ✓
- ***Cheap** (0 runtime overhead)* ✓
- ***Few false negatives*** ✓
- ***Few false positives** (~0%)* ✓



Usage Model



Issues

program

Application Home
(e.g., AppStore)

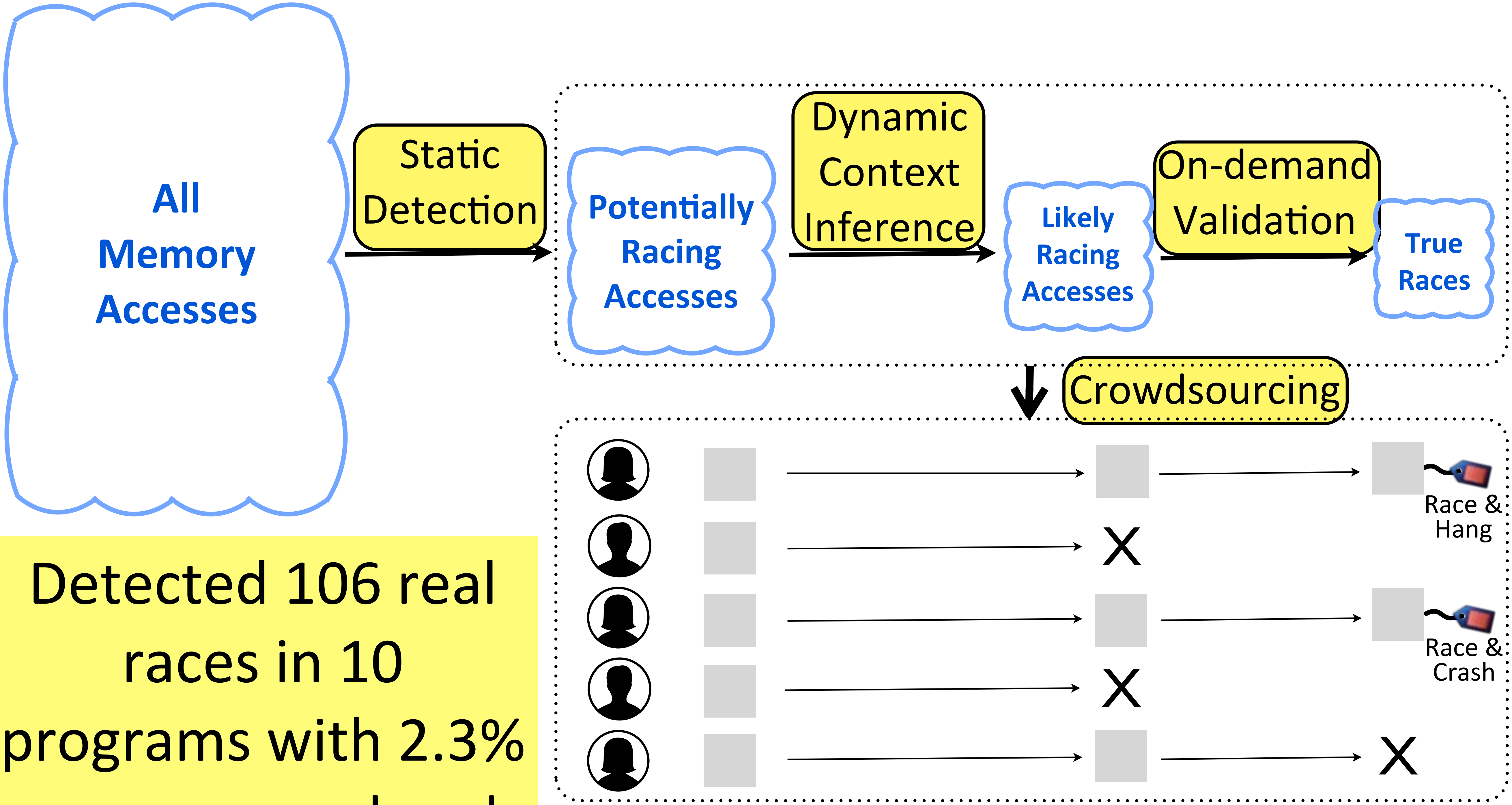
RaceMob

Users run instrumented programs

Insights

- Use static race detection to prune memory accesses that need not be monitored
- Cost of dynamic detection can be amortized across many users
- Using the crowd, we can detect races that ”matter”

RaceMob



All Memory Accesses

Static Detection

Potentially Racing Accesses

Dynamic Context Inference

Likely Racing Accesses

On-demand Validation

True Races

Crowdsourcing

Race & Hang

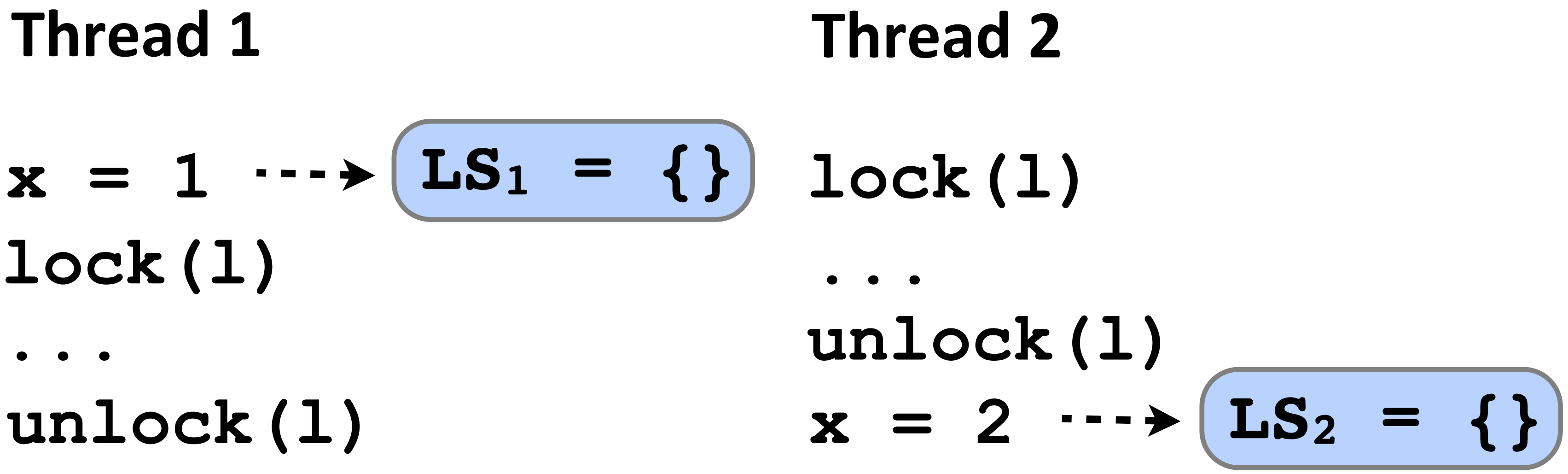
Race & Crash

Detected 106 real races in 10 programs with 2.3% average overhead

Static Data Race Detection

- We use RELAY [FSE’07]
  - *Analyzes entire program paths at once*
  - *Computes & composes per-function summaries to scale*
  - *Tracks locks*

Example



Thread 1

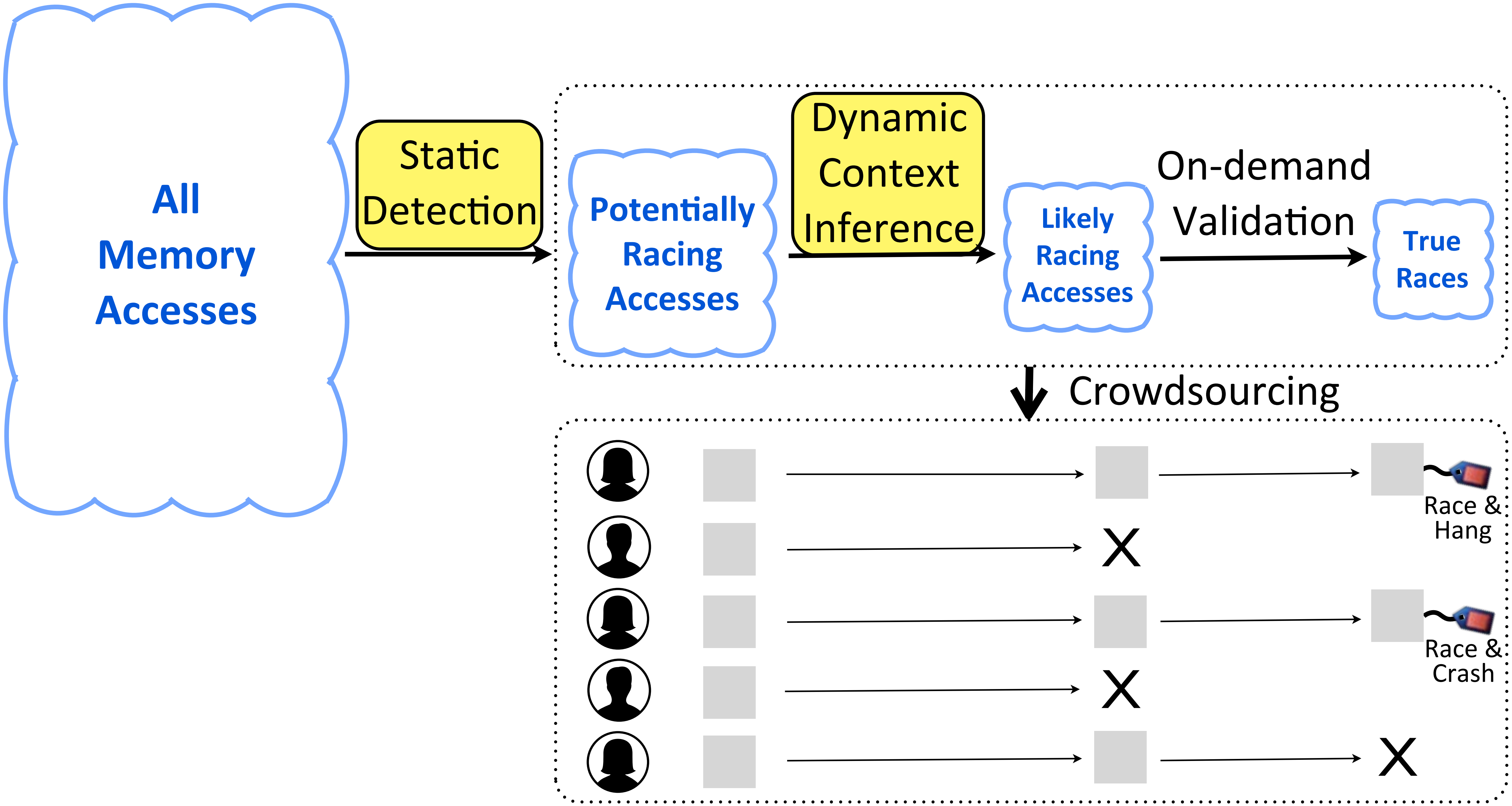
```
x = 1 ---> LS1 = {}
lock(l)
...
unlock(l)
```

Thread 2

```
lock(l)
...
unlock(l)
x = 2 ---> LS2 = {}
```

x = 1 and x = 2 are potentially racing

RaceMob



All Memory Accesses

Static Detection

Potentially Racing Accesses

Dynamic Context Inference

Likely Racing Accesses

On-demand Validation

True Races

Crowdsourcing

Race & Hang

Race & Crash

Dynamic Context Inference (DCI)

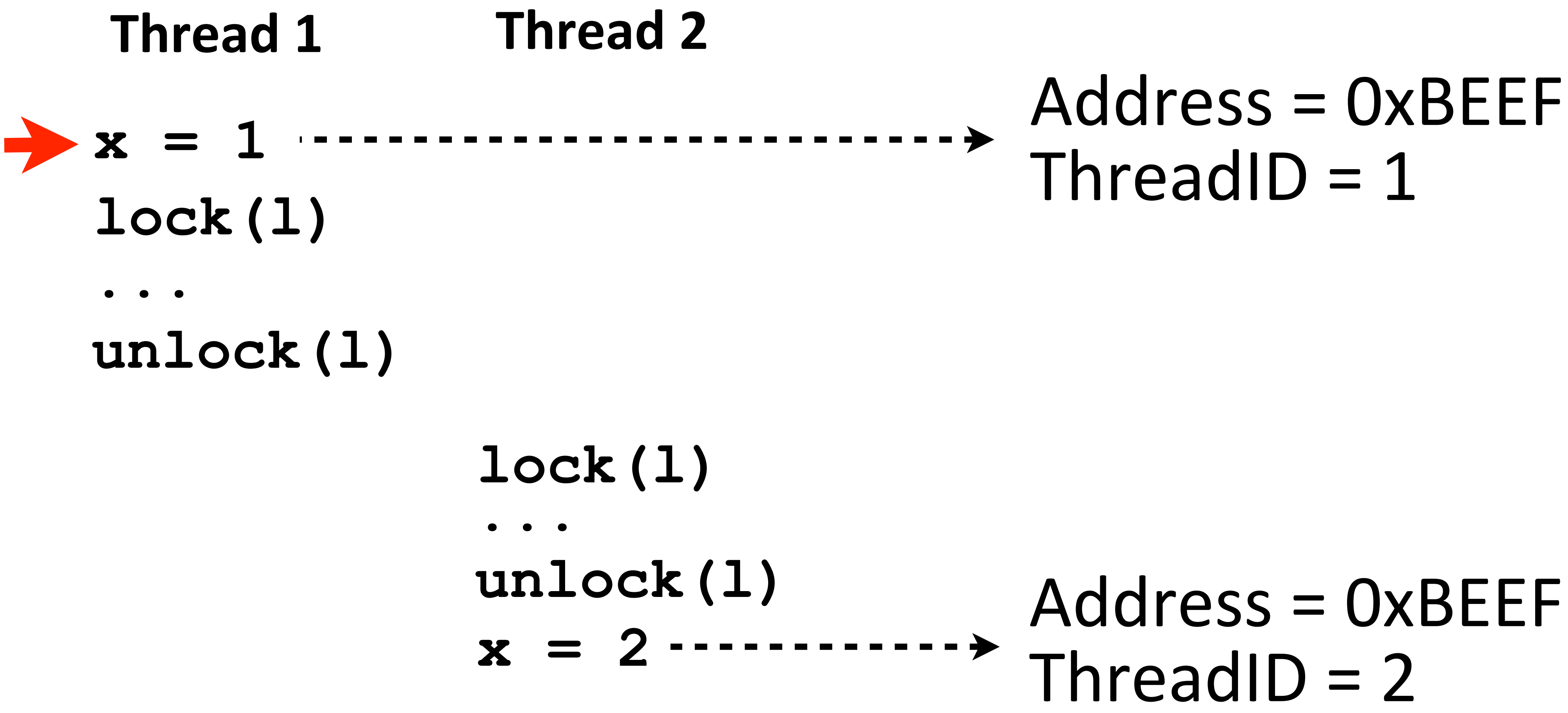
- Inaccuracy in static data race detection
  - *Pointer alias analysis errors*
  - *Inability to infer multithreaded program context*
- DCI checks at runtime:
  - *If accesses are from different threads*
  - *If accesses alias*

} If yes, accesses are likely racing

Dynamic context inference reduced the set of accesses to monitor by 53%

DCI Example

Alice



Thread 1 **Thread 2**

```
x = 1
lock(l)
...
unlock(l)
```

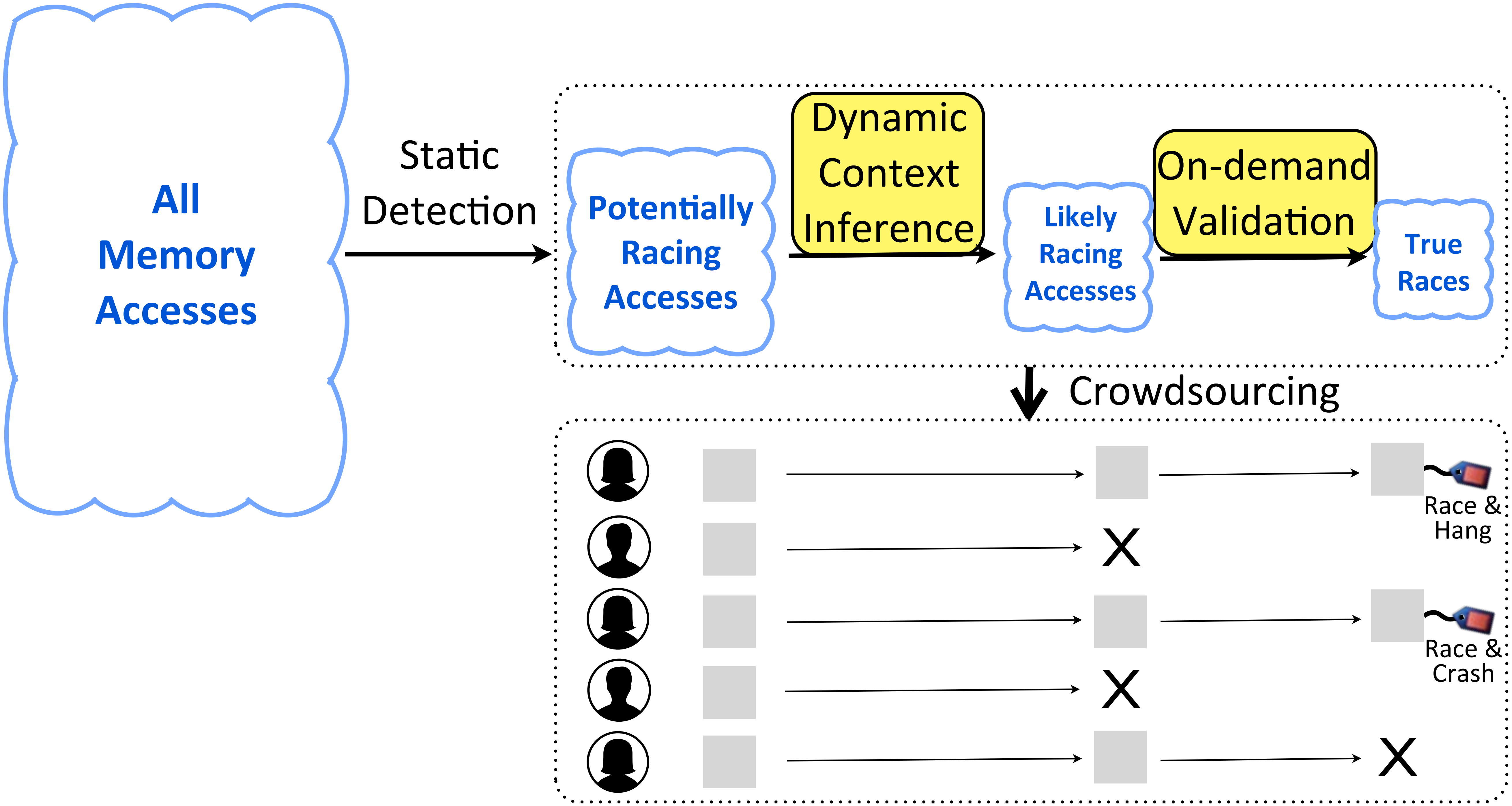
Address = 0xBEEF
ThreadID = 1

```
lock(l)
...
unlock(l)
x = 2
```

Address = 0xBEEF
ThreadID = 2

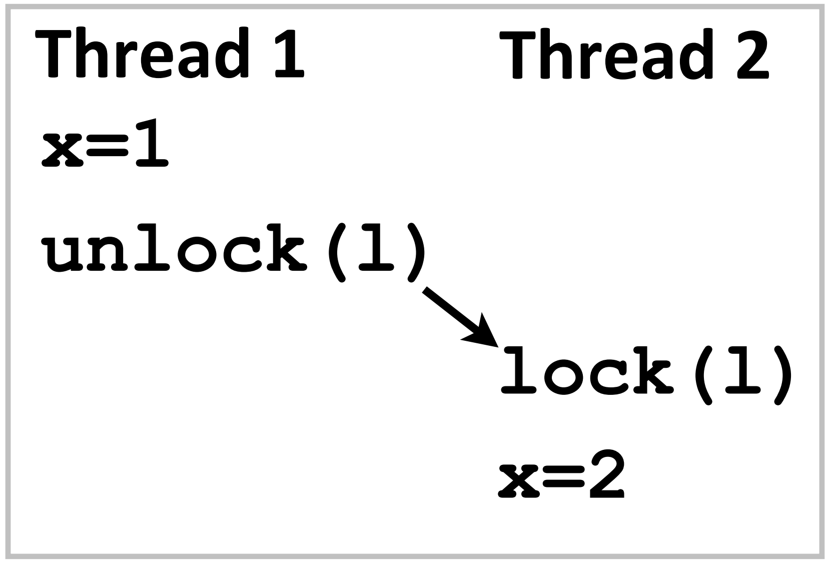
Proceed to on-demand data race validation

RaceMob


All Memory Accesses
Static Detection
Potentially Racing Accesses
Dynamic Context Inference
Likely Racing Accesses
On-demand Validation
True Races
Crowdsourcing
X
X
X
Race & Hang
Race & Crash

On-demand Data Race Validation

- Happens-before based
  - *Track synchronization*
  - *Few false positives*
- Minimal tracking
  - *Only memory accesses of the target data race*
  - *Synchronization in between these accesses*
    - Until enough happens-before edges form
- Steers thread schedule to expose races



On-demand Validation Example

Alice

```
Thread 1          Thread 2

x = 1
lock(l)
...
unlock(l)   HB
                  lock(l)
                  ...
                  unlock(l)
                  x = 2
```

No Race

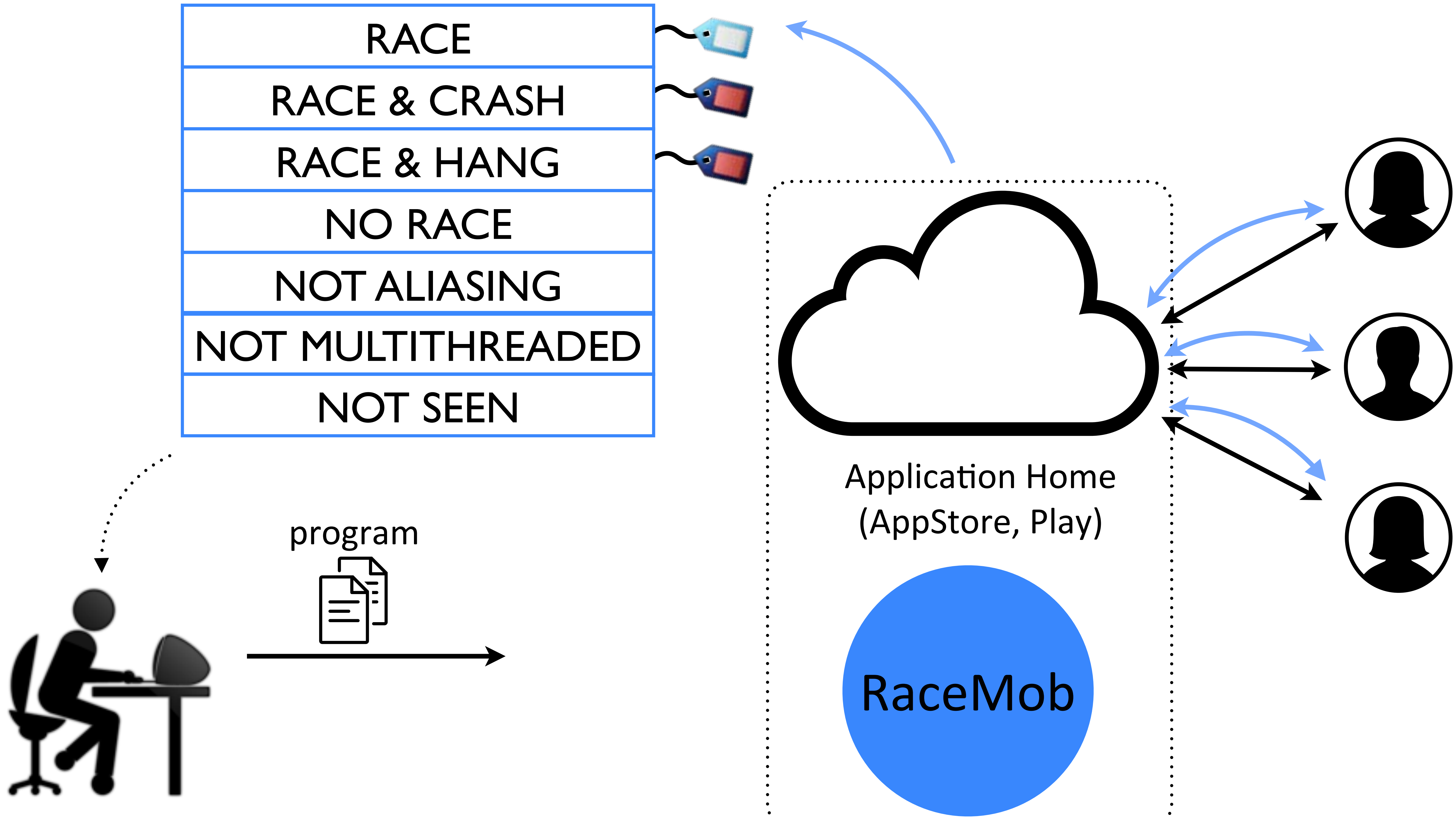
Bob

```
Thread 1          Thread 2

x = 1             lock(l)
                  ...
                  unlock(l)
     No HB        x = 2
x = 1
lock(l)
...
unlock(l)
```

Race

Detection Results



RACE

RACE & CRASH

RACE & HANG

NO RACE

NOT ALIASING

NOT MULTITHREADED

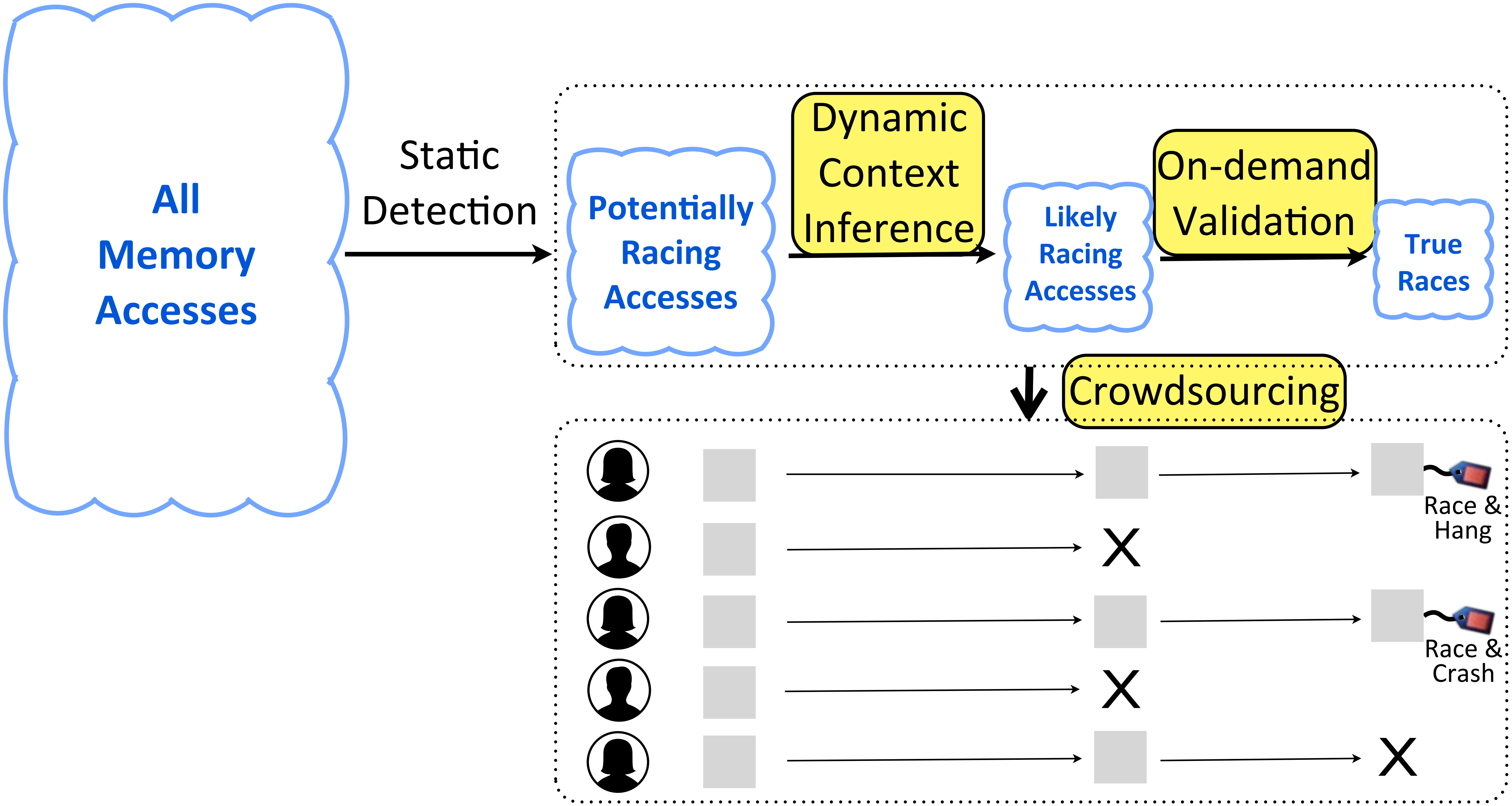
NOT SEEN

program

Application Home
(AppStore, Play)

RaceMob

RaceMob



All Memory Accesses

Static Detection

Potentially Racing Accesses

Dynamic Context Inference

Likely Racing Accesses

On-demand Validation

True Races

Crowdsourcing

Race & Hang

Race & Crash

Evaluation

memcached

Pbzip2 Ctrace

Apache SQLite pfscan SPLASH-2

- Detection effectiveness
- Contribution of techniques to reducing overhead
- Breakdown of overhead
- Comparison to other detectors
- Comparison to concurrency testing tools
- Scalability analysis

Evaluation

memcached

Pbzip2

Ctrace

Apache

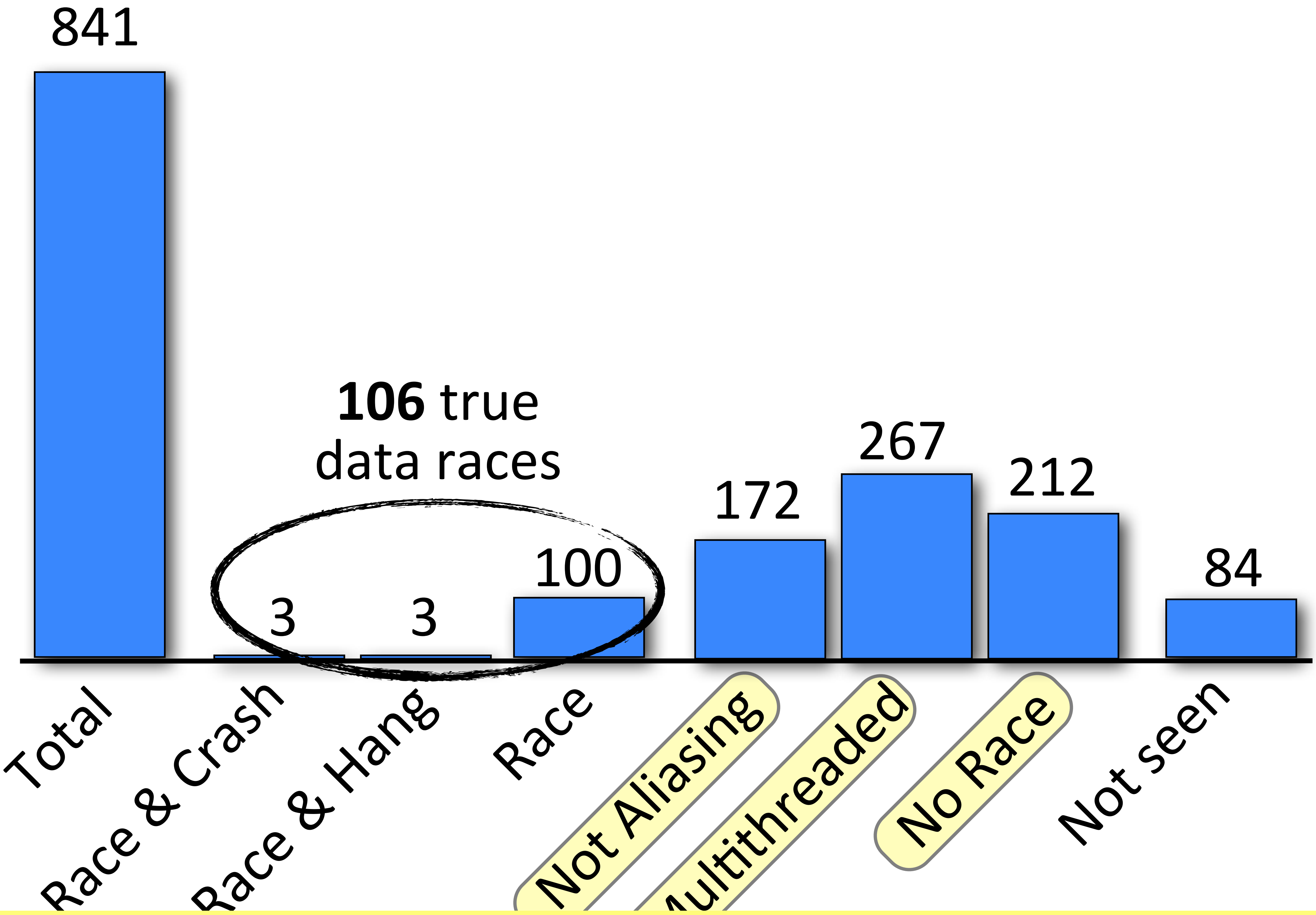
SQLite

pfscan

SPLASH-2

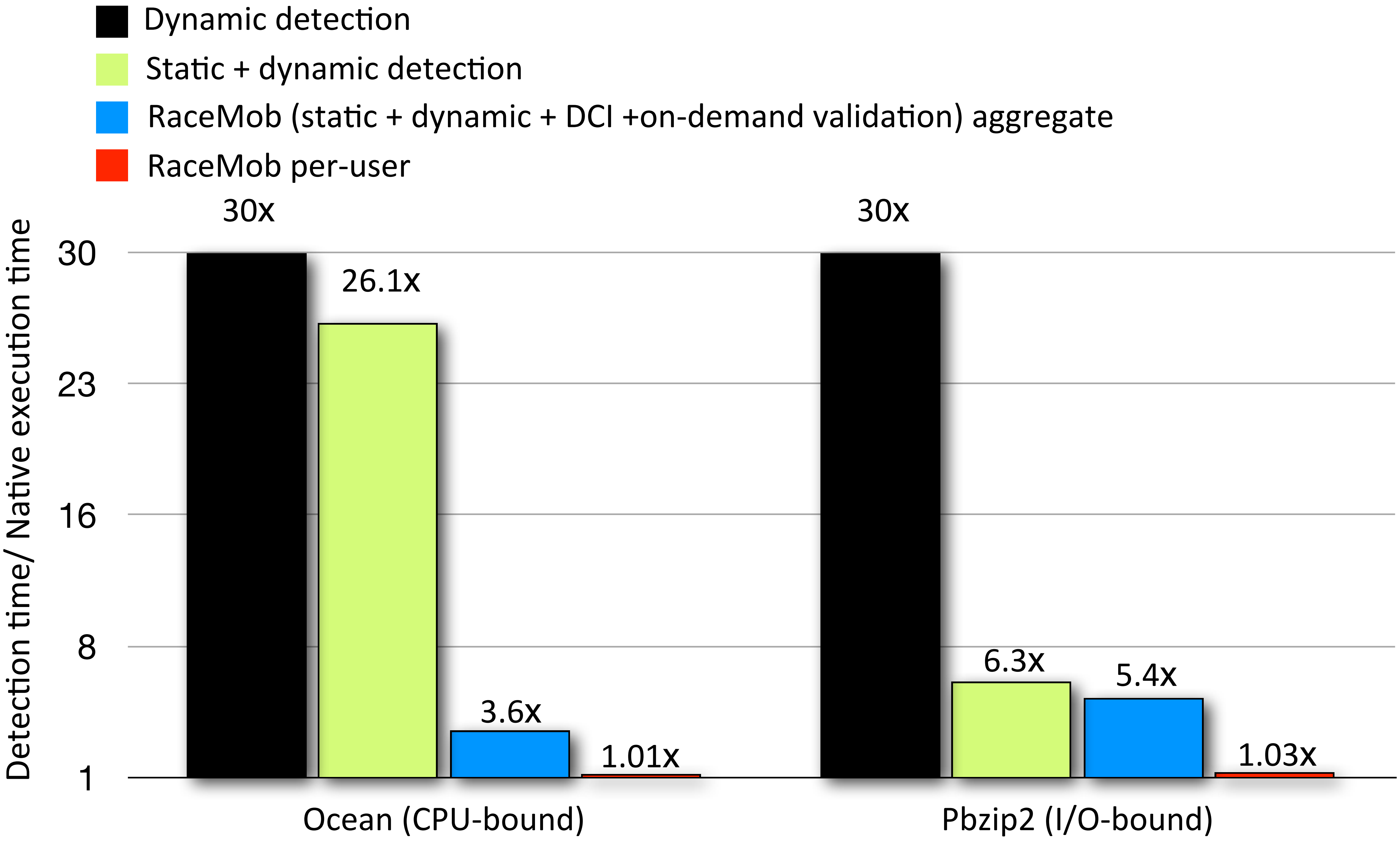
- Detection effectiveness
- Contribution of techniques to reducing overhead
- Breakdown of overhead
- Comparison to other detectors
- Comparison to concurrency testing tools
- Scalability analysis

Detection Effectiveness



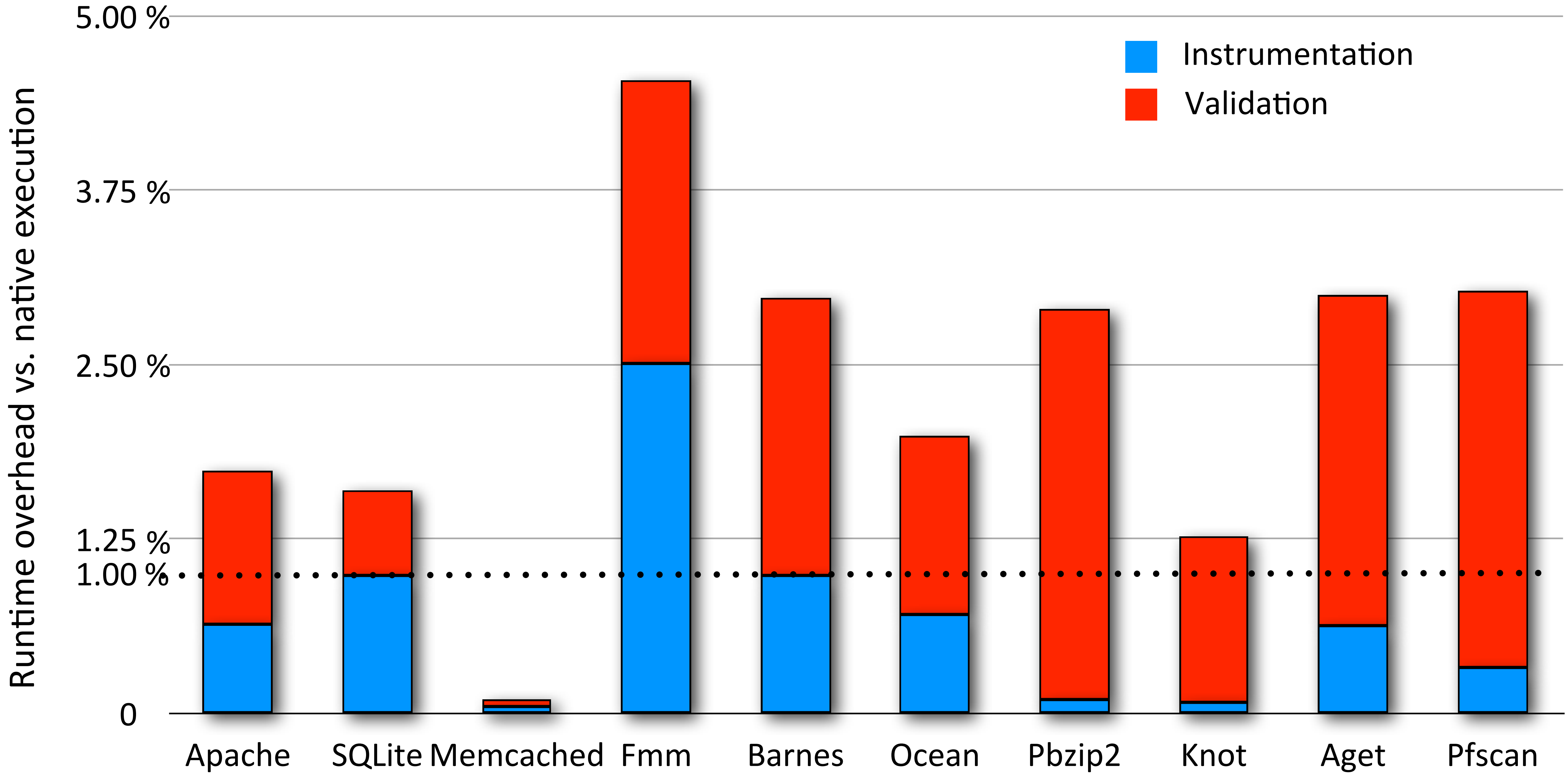
RaceMob detected and confirmed 106 data races

How Does Each Technique Lower the Overhead ?



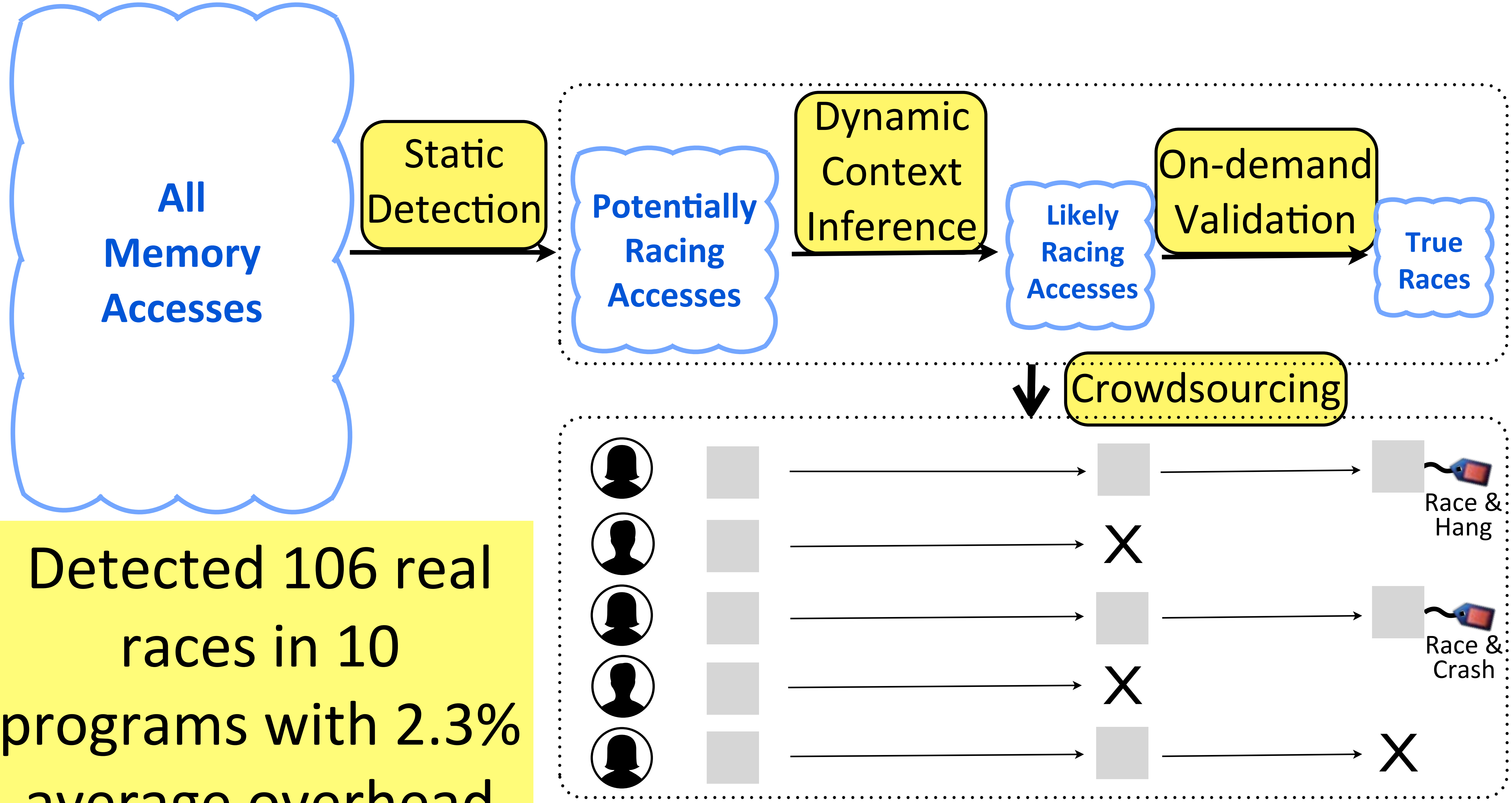
All techniques are required for low overhead

Breakdown of overhead



2.3% average runtime overhead per-user

RaceMob



Detected 106 real races in 10 programs with 2.3% average overhead