**What is a data race?**

2 threads access a shared location without synchronization, and at least one access is a write.

Consequences:
- complicated program semantics
- (e.g. non-sequentially-consistent execution)
- undefined semantics for C/C++ programs

**Race detection is big and slow**

Spatial overhead for precise race detectors is **HIGH**. Each shared location needs $\geq 24$ bytes of metadata.

**Race detection metadata is highly redundant**

Redundancy arises when $X, Y, Z$ are accessed without intervening synchronization

*Metadata objects shown in the shaded box have identical values!*

**How much redundancy is there in real programs?**

**How does SlimFast work?**

How are shared locations mapped to their metadata?

In previous race detectors (e.g. FastTrack)

In SlimFast

<table>
<thead>
<tr>
<th>var</th>
<th>meta</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

In SlimFast, metadata instances are **immutable**.

Metadata is updated as follows:

```
start
look up existing metadata
Yes exists? No
return existing metadata allocate new metadata
end
```

**Results**

SlimFast consumes less memory than FastTrack on all benchmarks. Average reduction: 2.02x Max reduction: 3.51x (crypt)

Overall, SlimFast runs faster than FastTrack. Average speedup: 1.2x Max speedup: 2.3x (lufact)