Recitation 5: Eraser

Henry Corrigan-Gibbs
6.033 - Spring 2022

Plan

- Data races - what are they
- Race detectors: Go
- Eraser
  * Group discussion
  * Summary

Logistics

* Feedback summary
* Volunteers for next time ans rec Qs?

Very excited to get feedback!

- Don't spend enough time waiting for answers to questions before moving on.
- Don't feel confident enough in my answers to speak up in class.
- Worried about losing points.

→ Ask a question! Don't need to answer
→ Volunteer for prepared Q.
→ Group activities

⇨ Participation check in coming soon...

Real life race condition

Parents with eggs

e.g. Shopping list on fridge.

1 Read items on list.

2. Go to store, buy item. (slow)

3. Cross item off of list.

```
list = { .... ]

get_home() {
    if (list.len > 0) {
        buy(list);
        list = [];
    }
}
```

PROBLEM: Can buy multiple items!

↳ ... but might not.

Depends on traffic on way to store :-

Shared access to a variable (shopping list)

1) At least one is a write.

2) No way to prevent simultaneous access.

Solution?

* Take list with you?
  ↳ failure?
* Mark item on list?
* Single threaded?

Race detection in modern languages.

→ Amir has comments ¡

Example in Go

1. Single thread — no race!
   ↳ Run race detector

2. Multi thread - race!
   ↳ Run race detector

3. Add locks ... careful where to add
   sync. Mutex

Eraser is implementing a race detector just like this one.

For impl details, see Kavya Joshi talk on Youtube.

Group activity: Eraser

- Split by numbers

1. Three words you didn't know/understand.

2. In three not-too-long sentences, how does Eraser detect data races?

3. Has 10-30x overhead.
   Why slow?
   What could you do to improve efficieny?

Erase: Basic idea

- For each piece of data (addr in mem) Keep track of which locks held while R/W it.

↳ If none, complain!

- Why hard
  - →Read/Write locks
  - → Initialization w/o locks
  - → Memory reuse

Walk throgh trick OS code?

↳ Example of why threaded code is hard