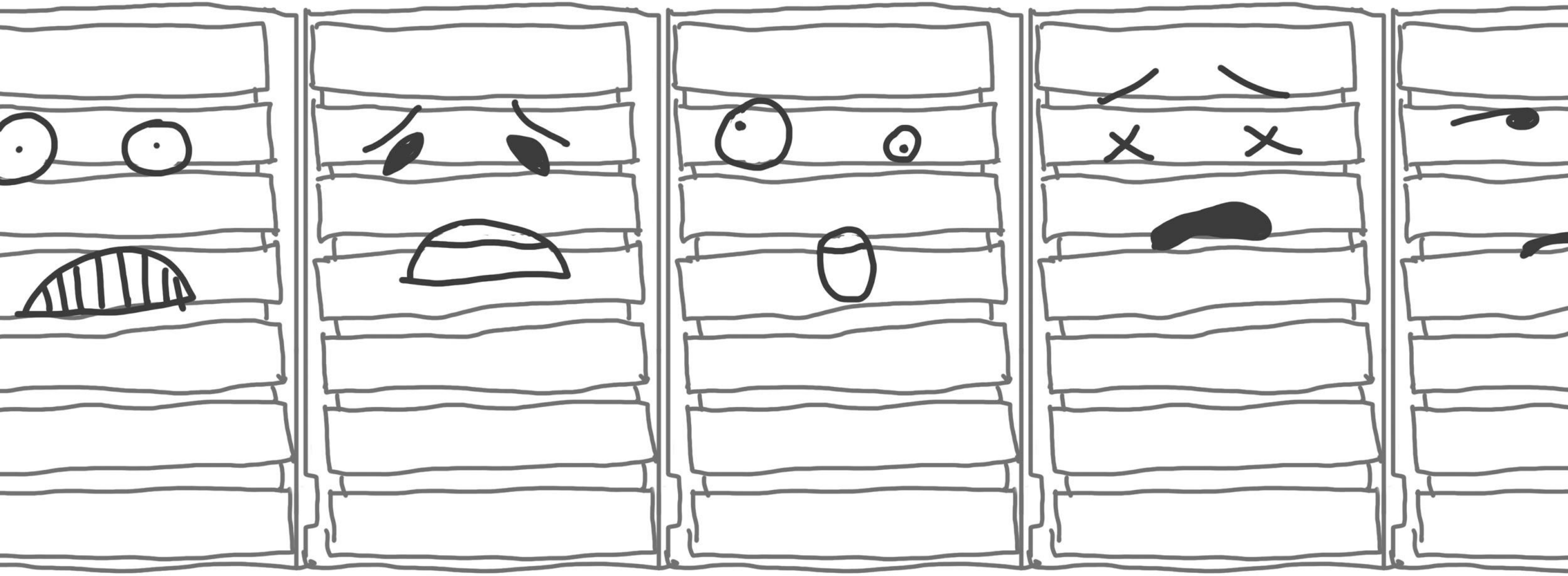
Server Memory



@listochkin

fb.com/tektonna

Not a JavaScript talk

No CSS
No frameworks
No Node

A bit deeper than most talks on similar subject

Medium Well 👌

How
programming languages
work

Just-in-time compilers

Garbage Collection

Memory Management

Server runs our programs

CPU

Caches

RAM

Swap

Disk

Network

Operating System

Lazy

^_^

Pages

4k

Virtual

2 processes

[0 … ∞]

Oh, I got a pointer at 4096
What’s at 4095?

Page fault

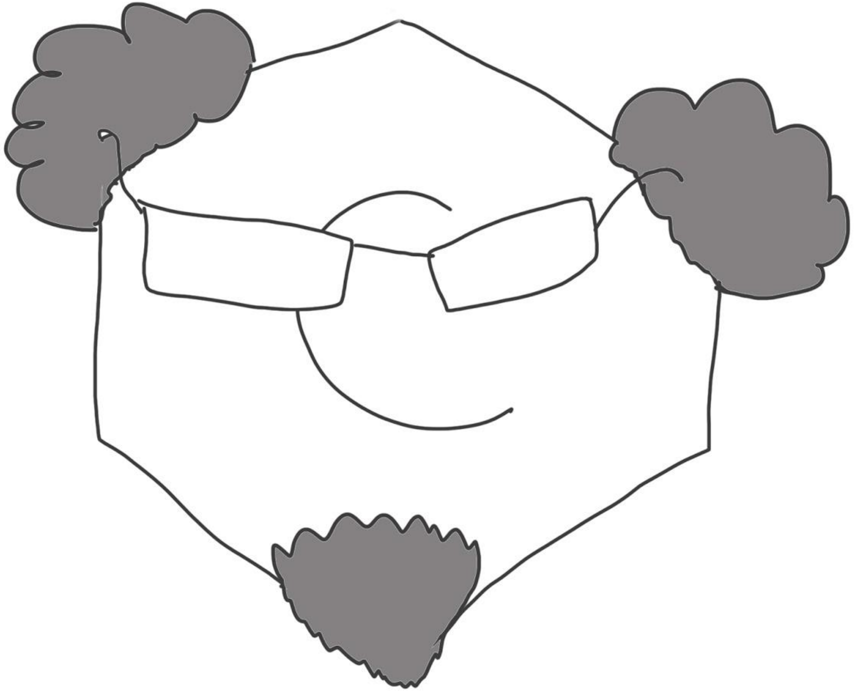
Allocate page
But not really

Once you start writing to it

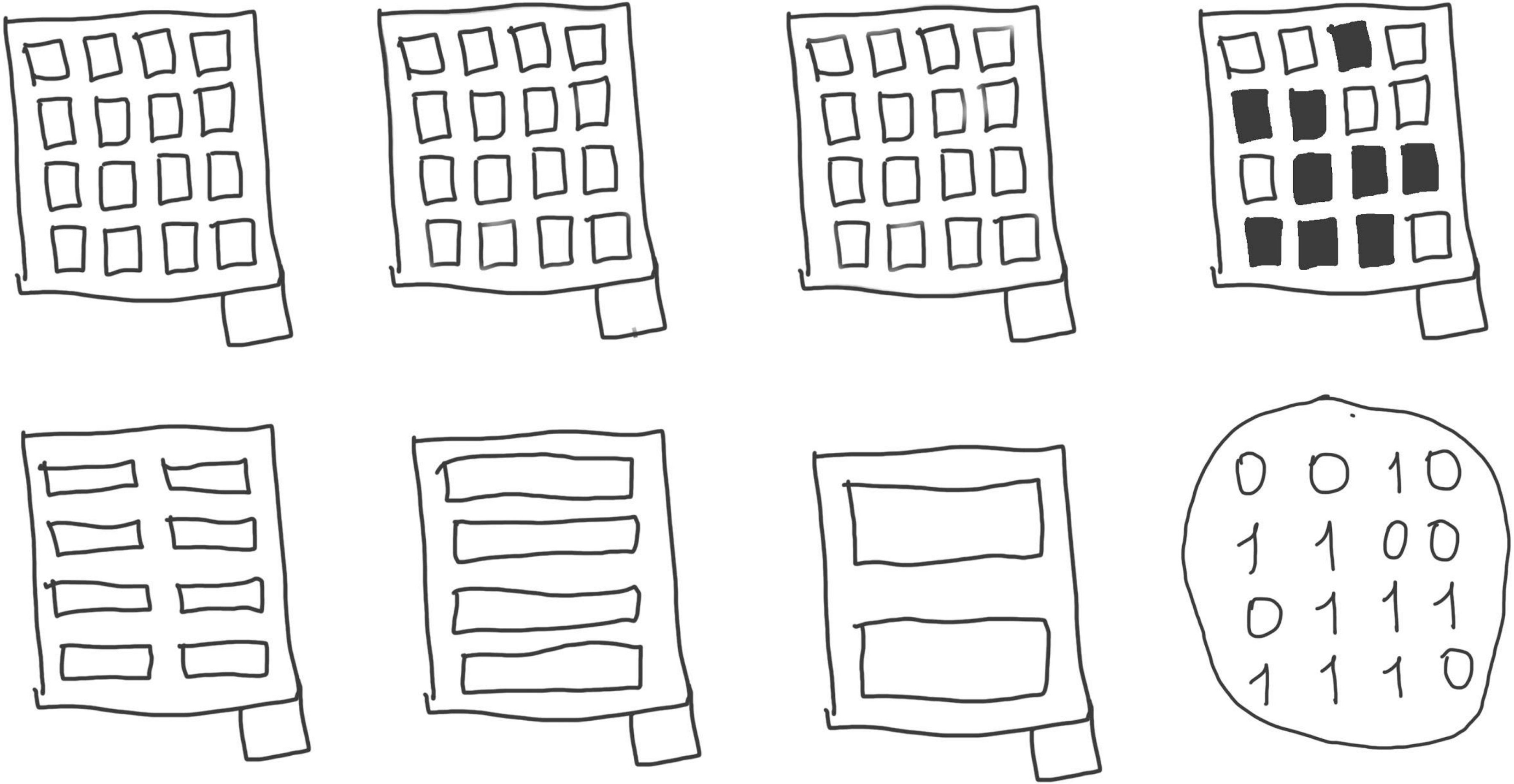
not 4k?

our Stars

C - old school cool 😎



`malloc`


0 0 1 0
1 1 0 0
0 1 1 1
1 1 1 0

malloc me some memory

a pointer

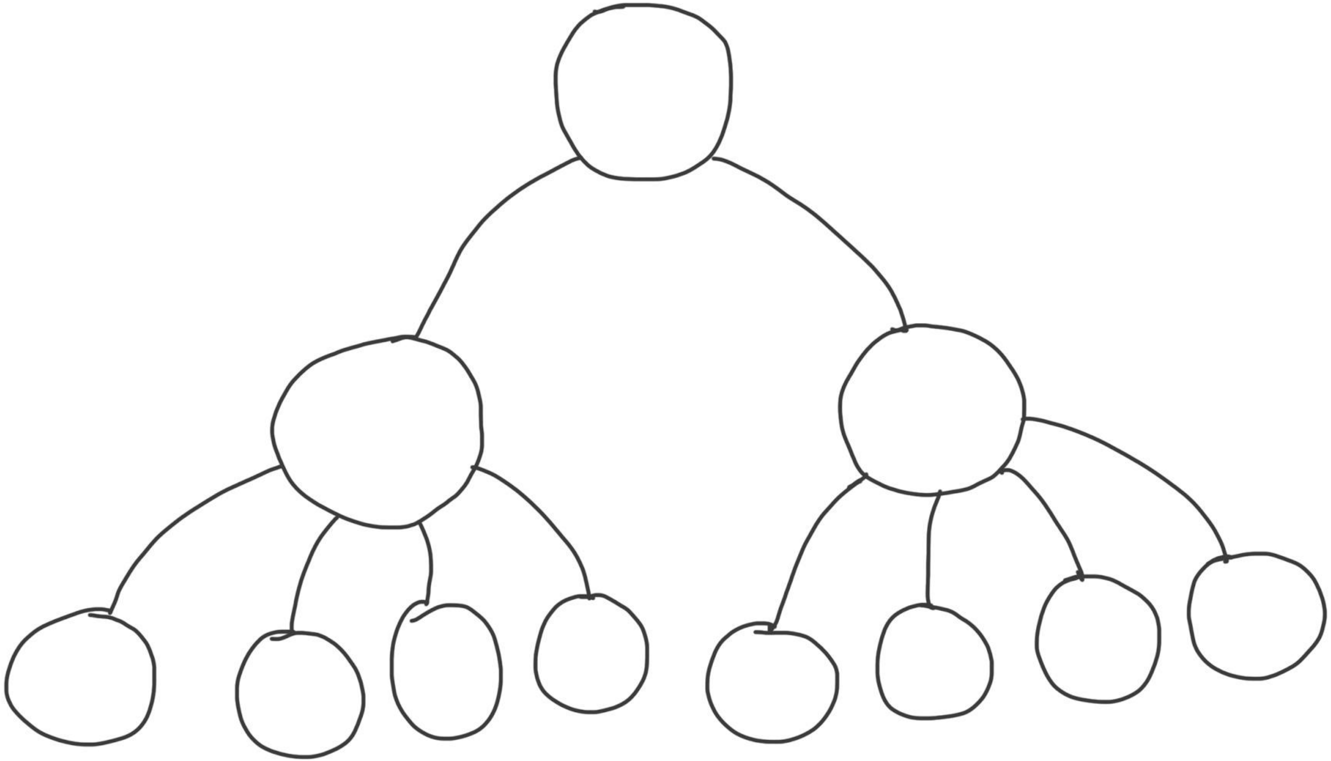
```
free(pointer)
```

```
malloc
malloc
malloc
free
malloc
free
free
```

Forget to free
Free the wrong pointer
Read from unallocated memory

Can we do better?

Reference Counting



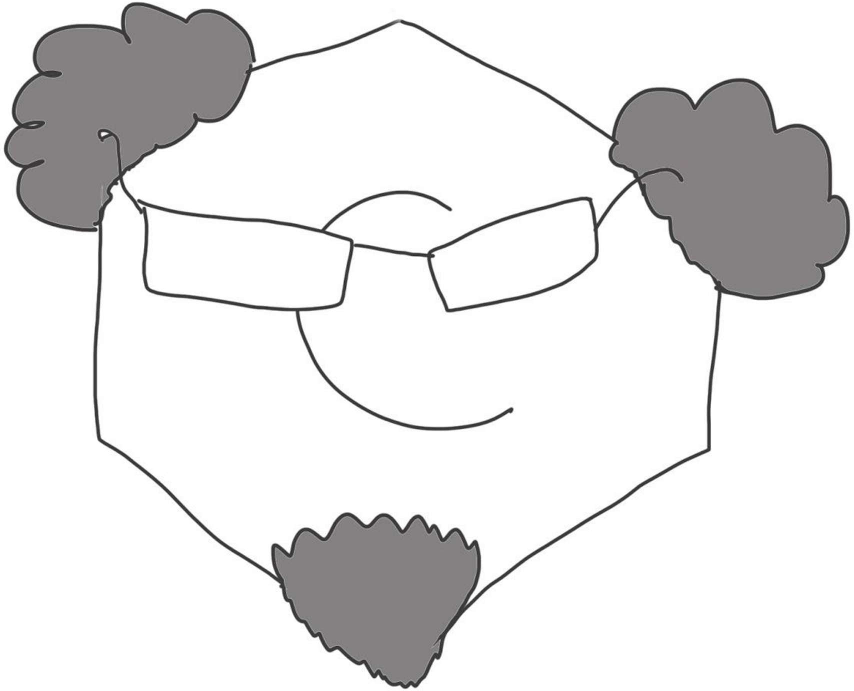
```
const players = [ … ]
players.forEach(p =>
makeMove(p))
```

```
a = …
doX(a)
yield a
```

scope

```
if ( … ) {
  let a = …
  …
}
```

C - old school cool 😎



C++: std::shared_ptr

Best from the BEST ♥

Optimizations

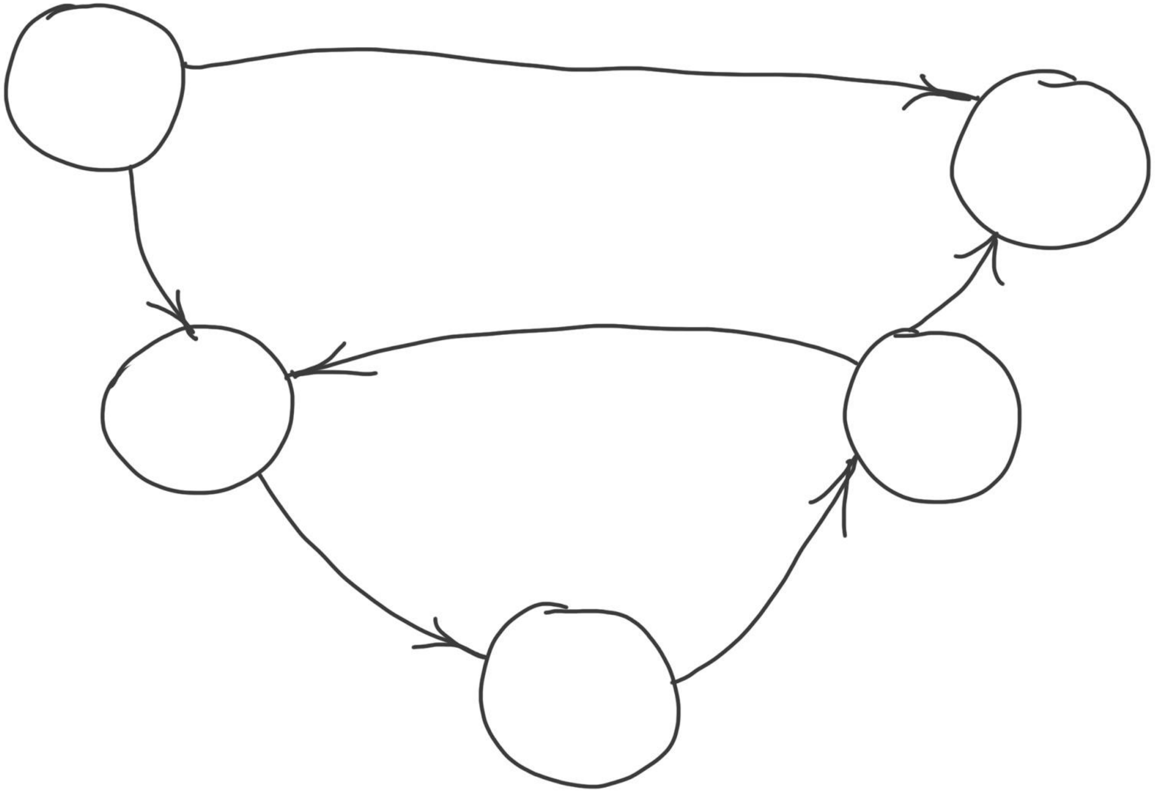
Delayed Counting

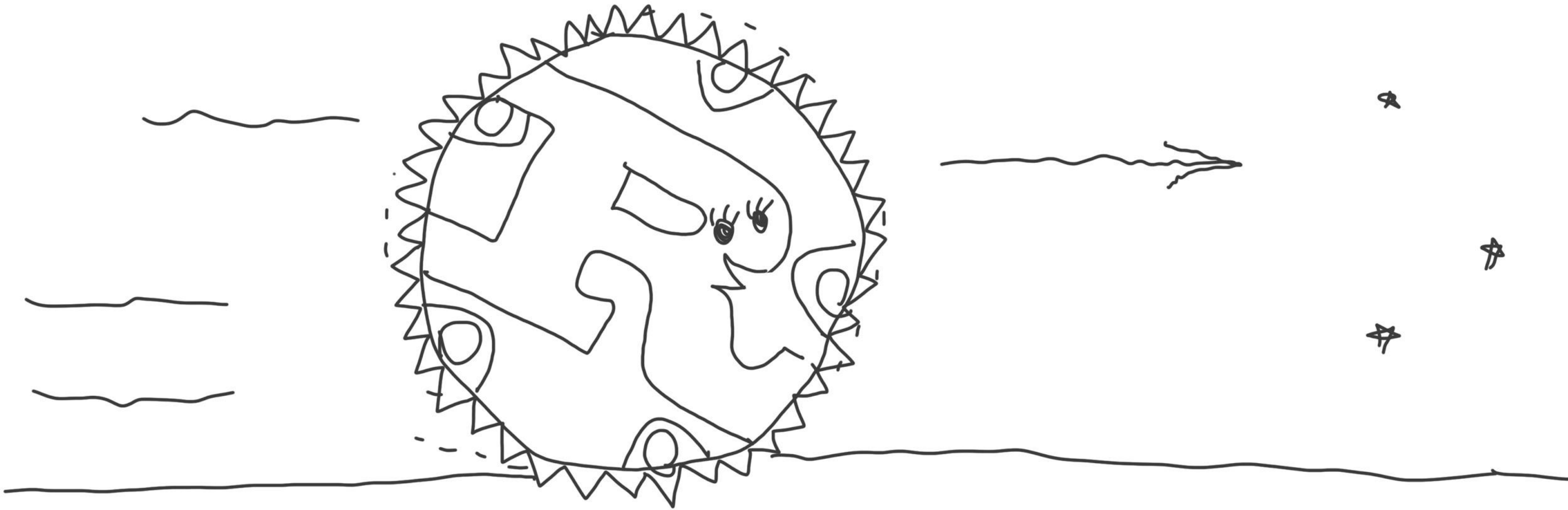
Don’t count local references

```
function (user) {
  let email = user.email;
  …
  …
}
```

Delayed deallocation

Cycle Collector





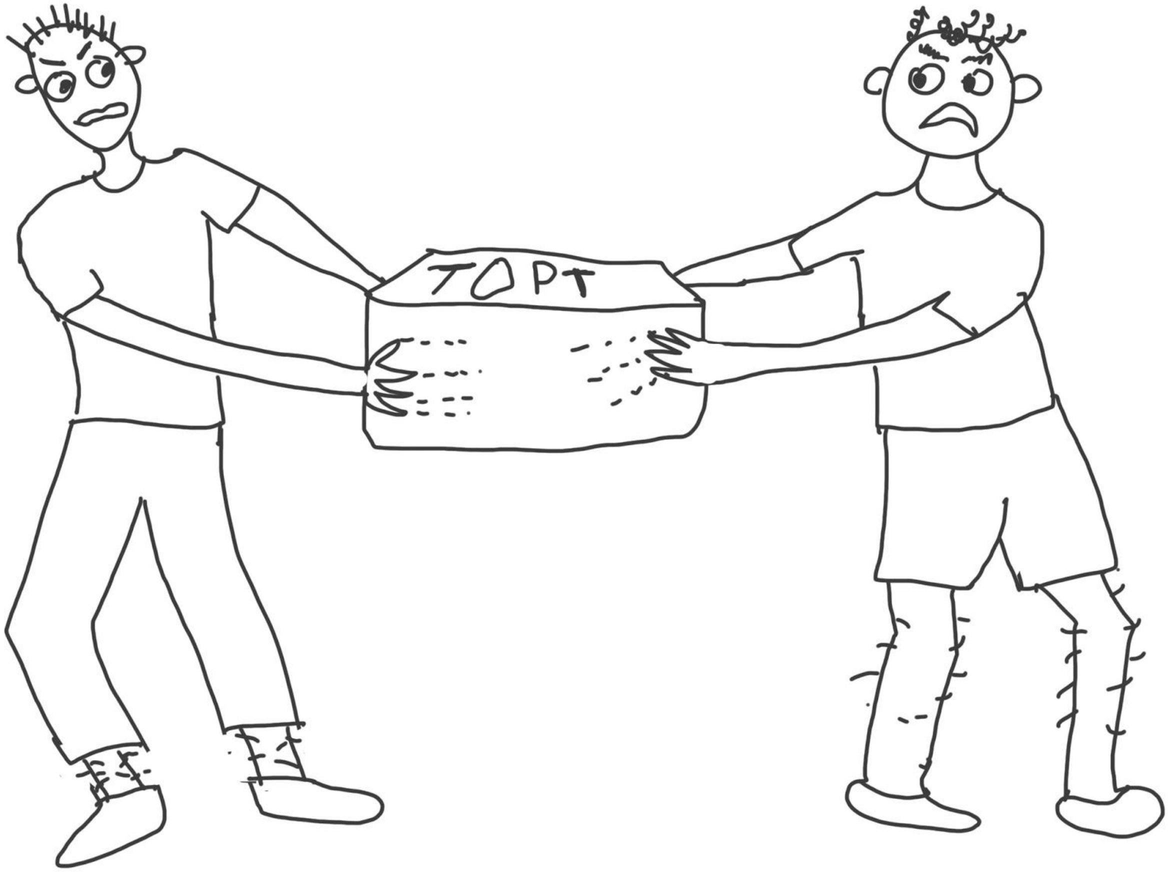
Rust - Future today

Compile time ref-counting

Several types of pointers

Ownership

Forbids cycles
by forbidding several owning
references
at the same time


ТОРТ

⋆ Perl ⋆

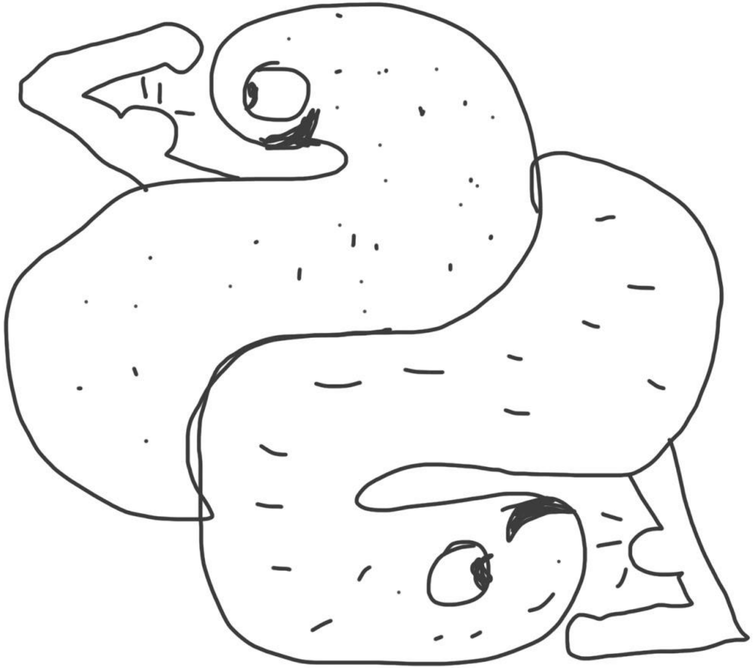
PRide of NASA

and

camels

Python

Python



when you need a real language

Perl

1987 - today

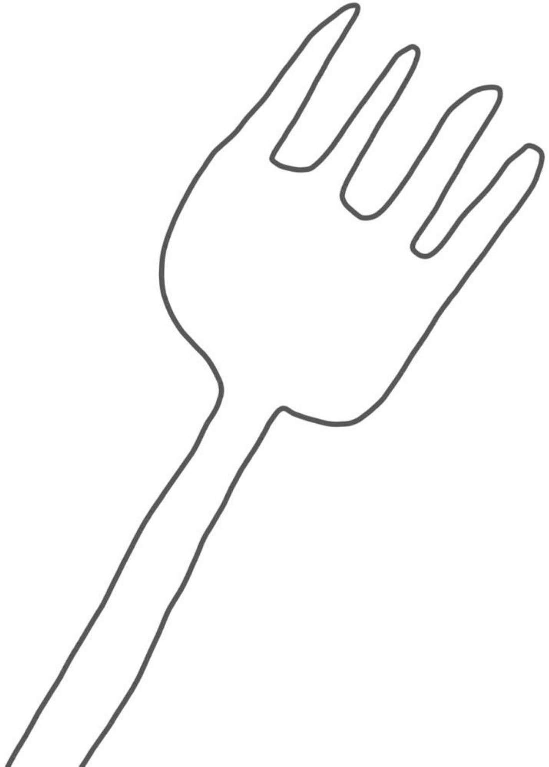
Python

1991 - 2001

PHP

1994 - 2009

Multiprocess deploys



fork

Copy on Write

Worker
Accepts requests

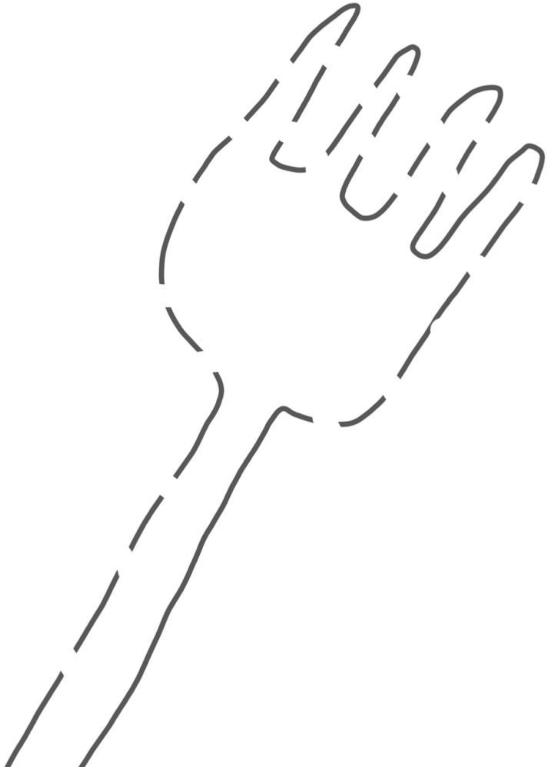
As memory use raises:
Stop accepting requests
Complete in-flight requests
Terminate

Master

Keep track on workers

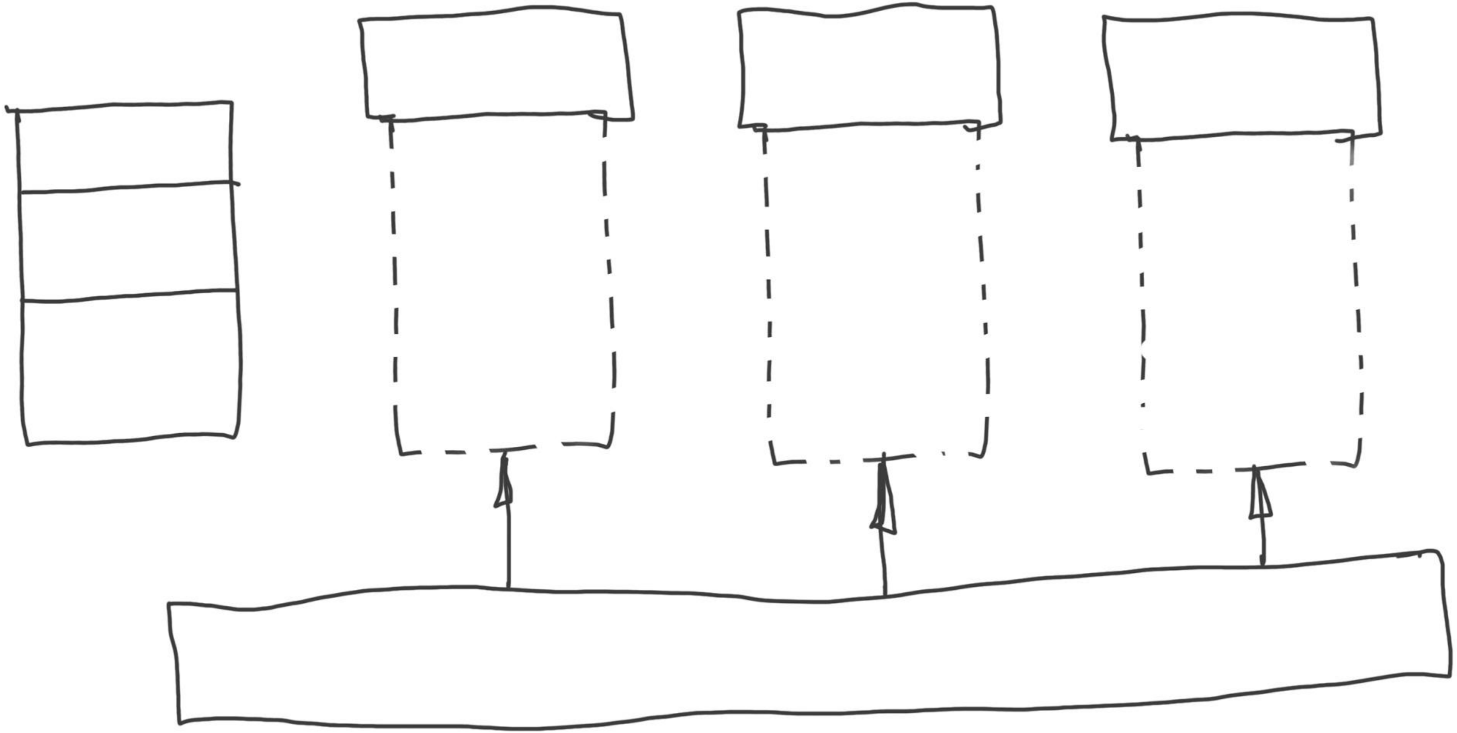
Start new workers

Signal them to terminate when memory pressure is high



“Pre-fork”

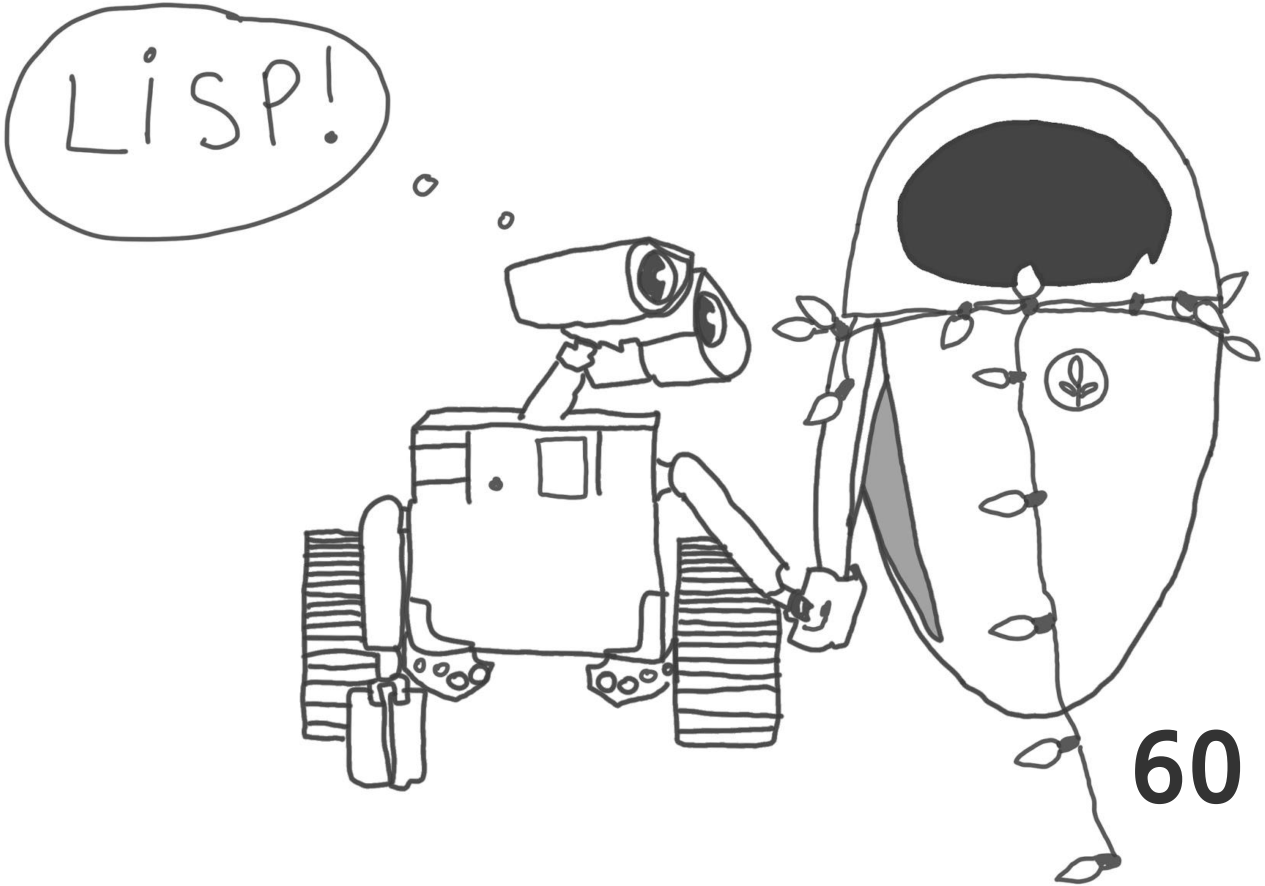
Load the framework
Load app code
Run full GC
Start forking process to
accept requests



Battle hardened

“Processes”

Tracing GC



1959

60 years ago

Start at Root references
Follow all references
Build a live objects tree

Delete all objects not part of the tree

Ruby
Java
JavaScript
Lua
Go

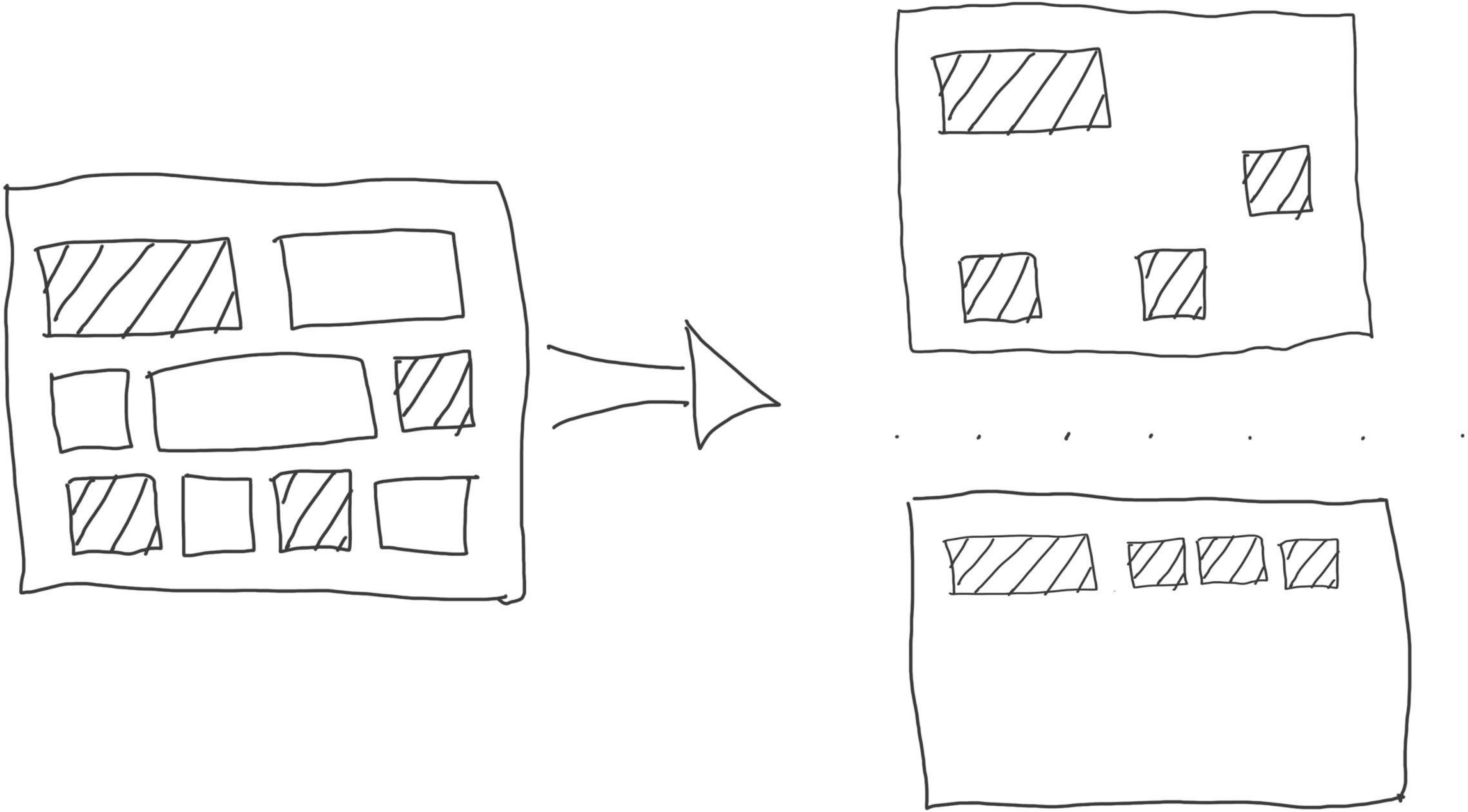
Roots?

Constants
Global variables
Local variables
Closures
Thread-locals
...

Mark & Sweep

vs

Mark-Compact



Can you move objects after mark?

Team Sweep:

Go

Ruby*

Lua

Embedded JS engines

Erlang

Pros:

Pointers don’t change

Native extensions

Easier to implement

Team Compact:

Java

JavaScript

Ruby*

Haskell

Pros:

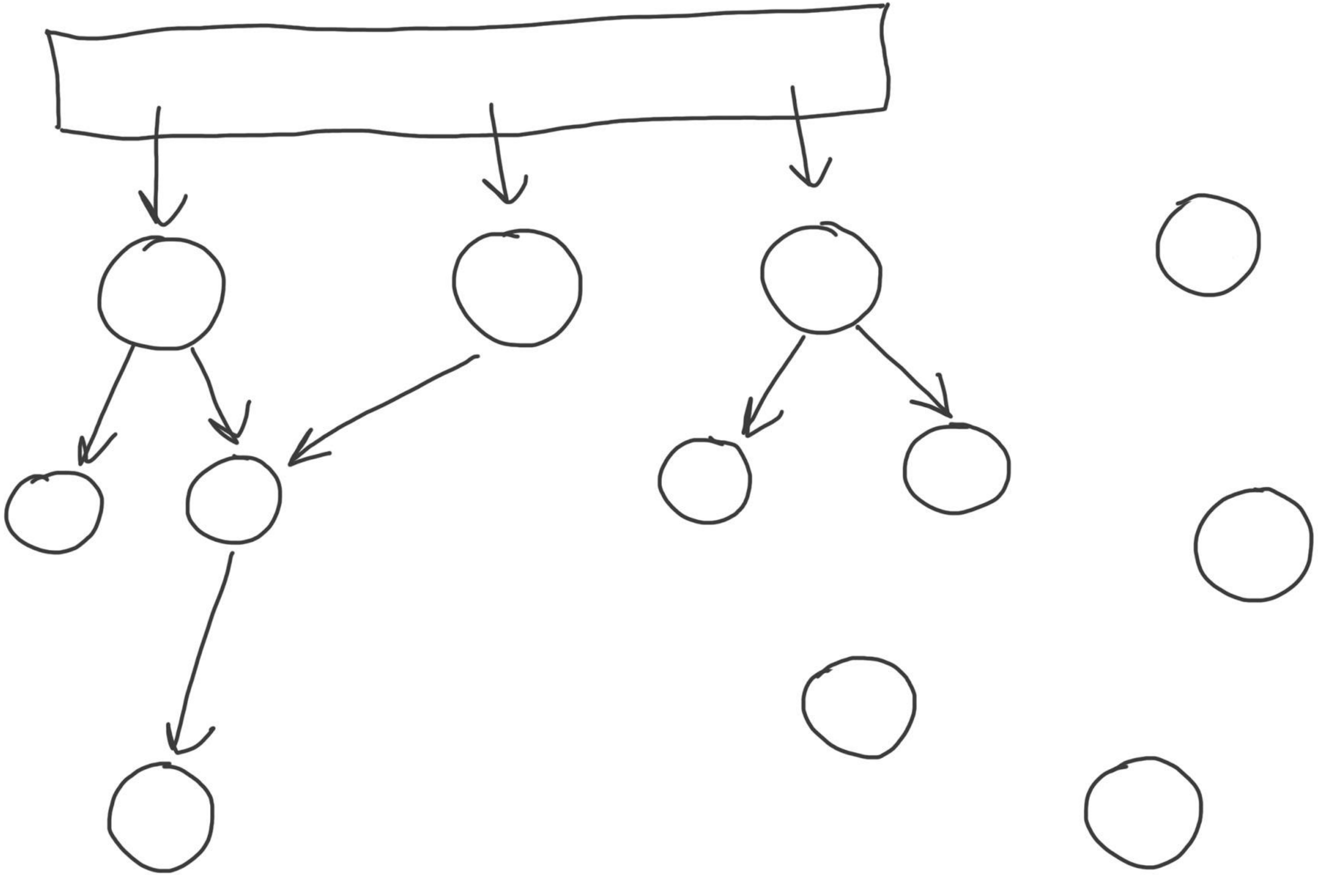
Less memory fragmentation over time

Cons:

Harder

Takes longer to do a GC

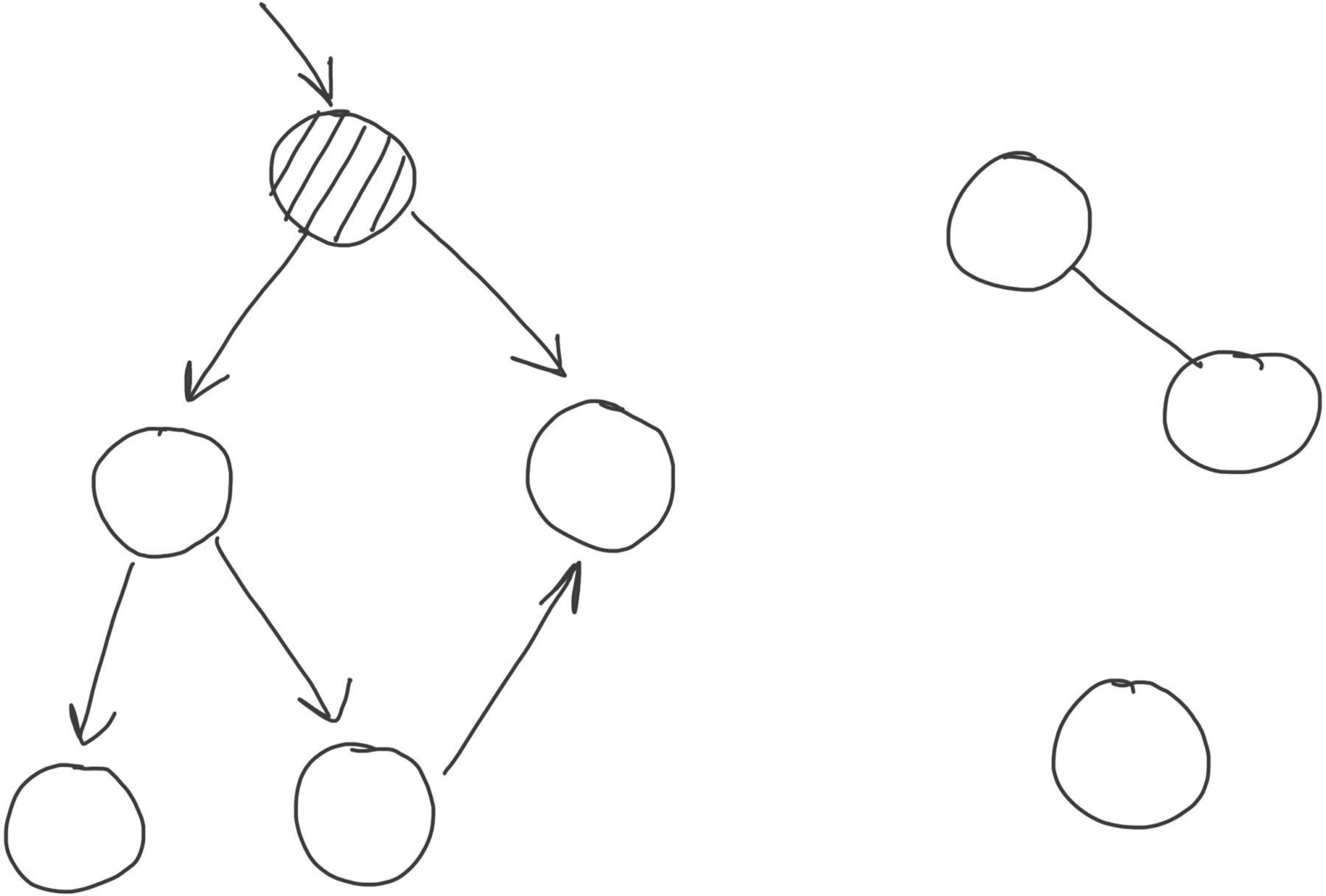
Mark all memory

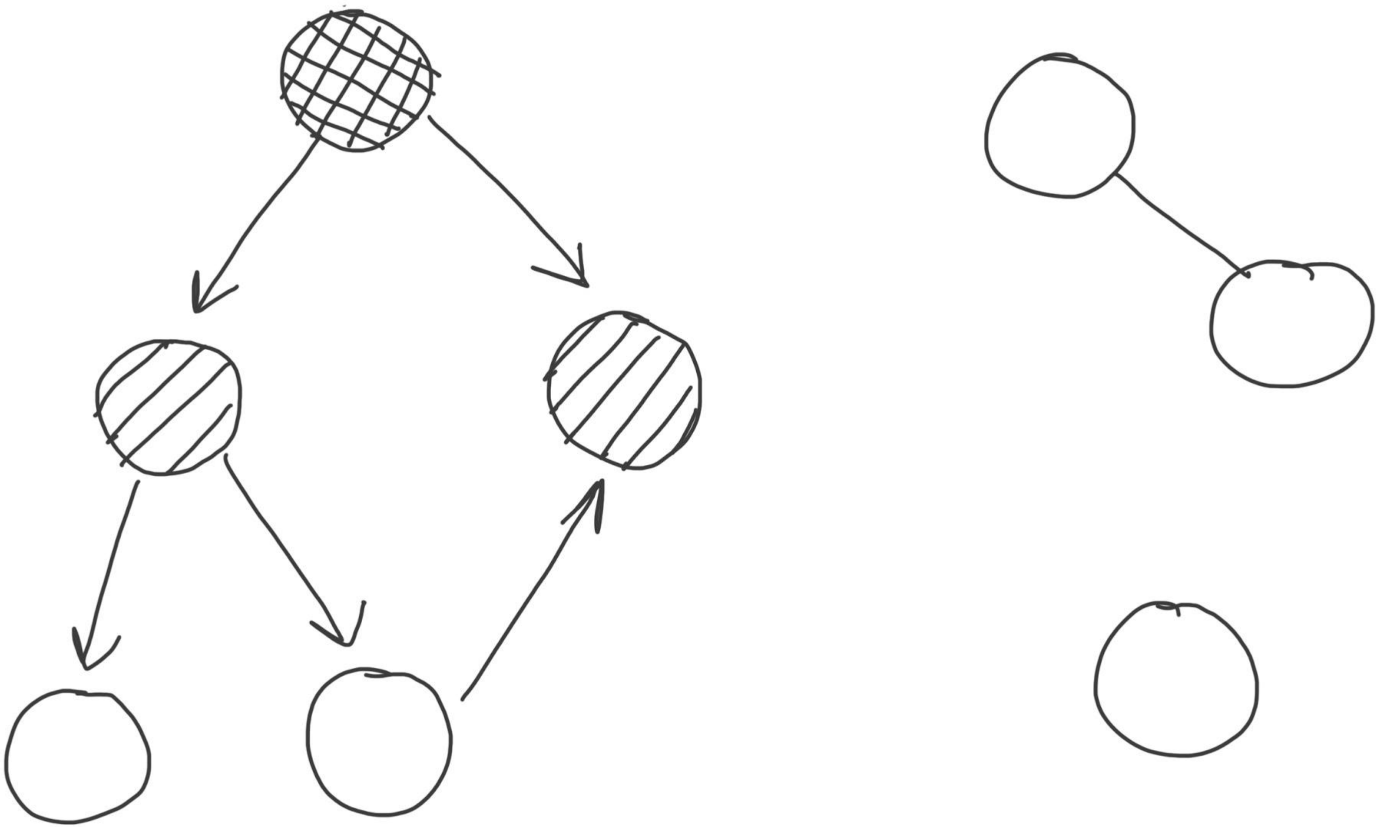


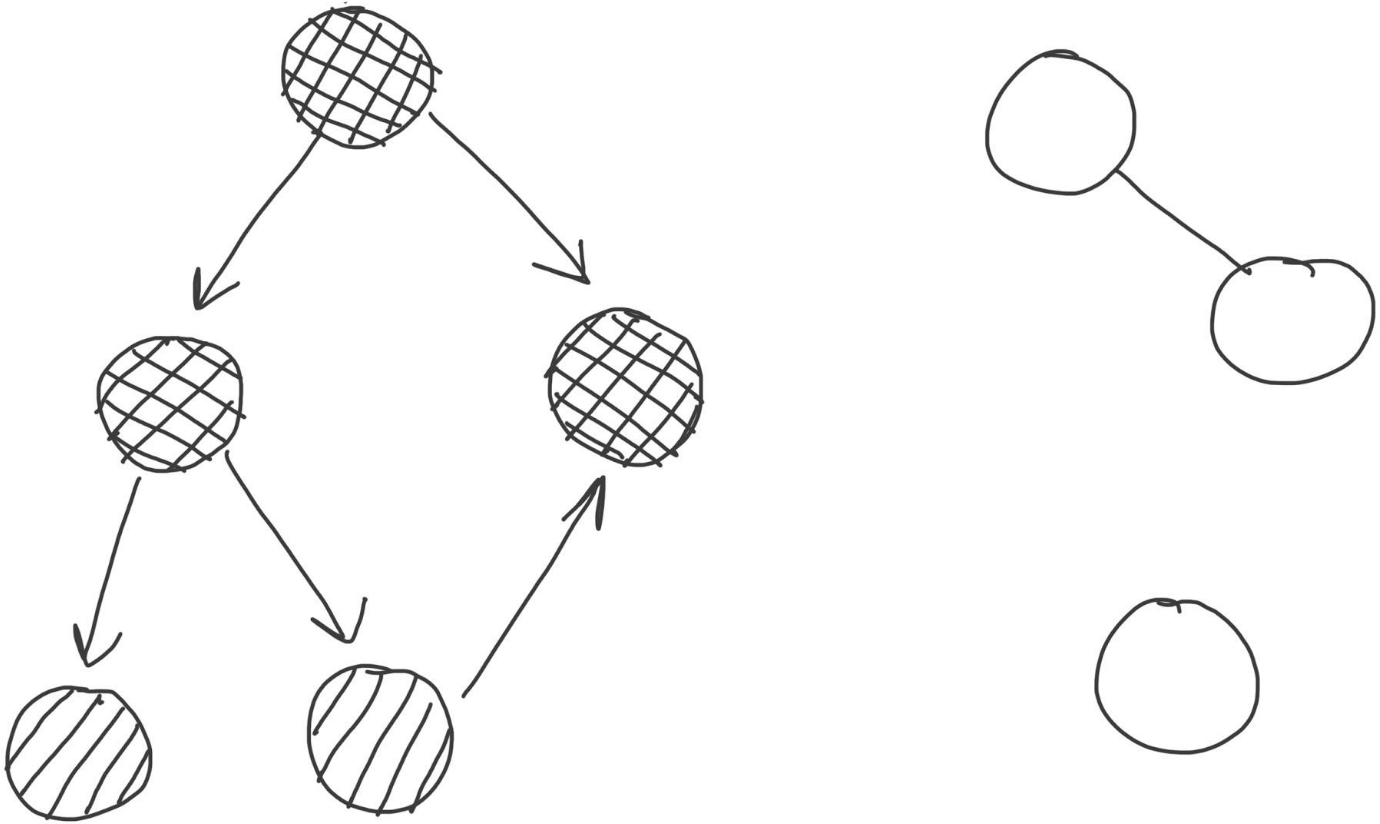
Incremental Marking

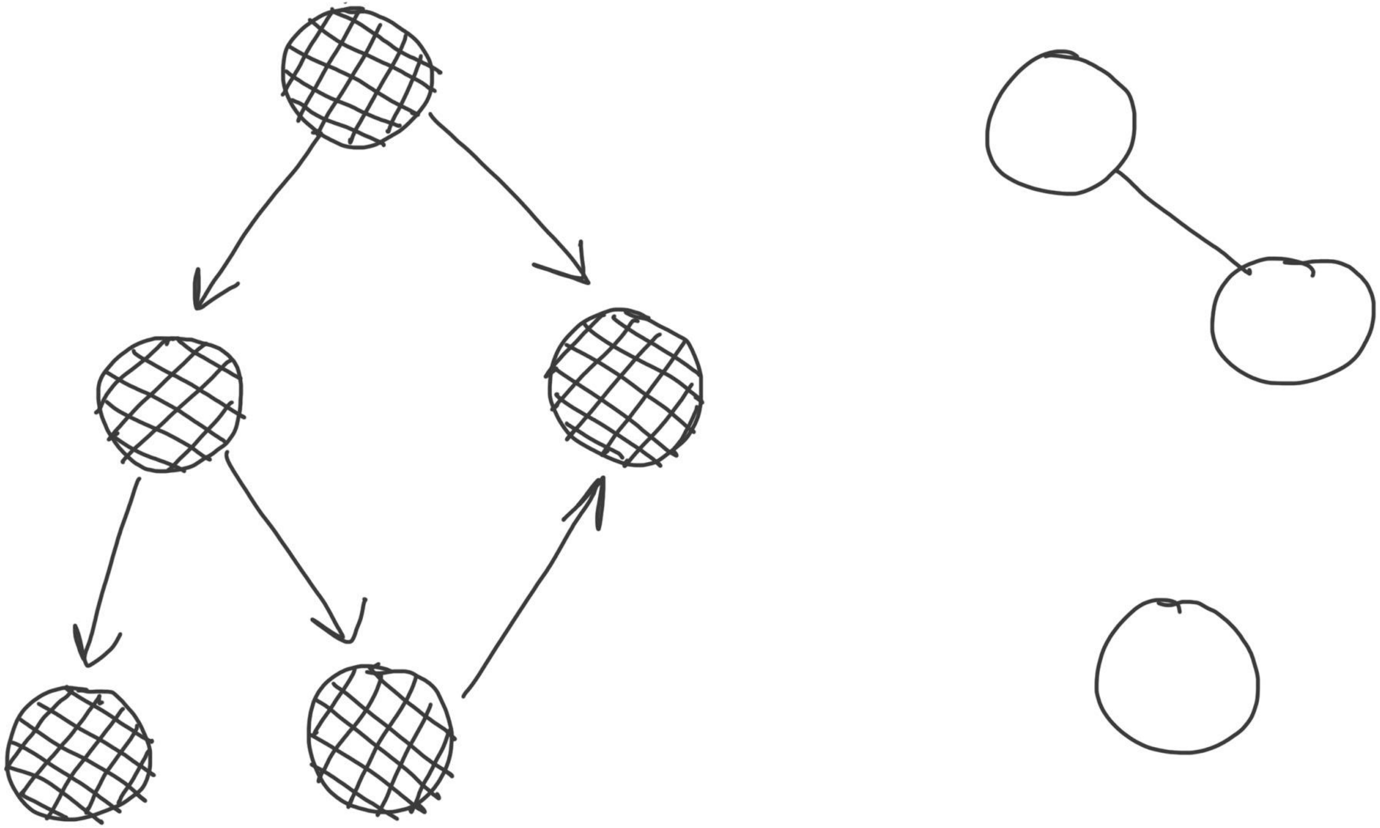
3-colored algorithm

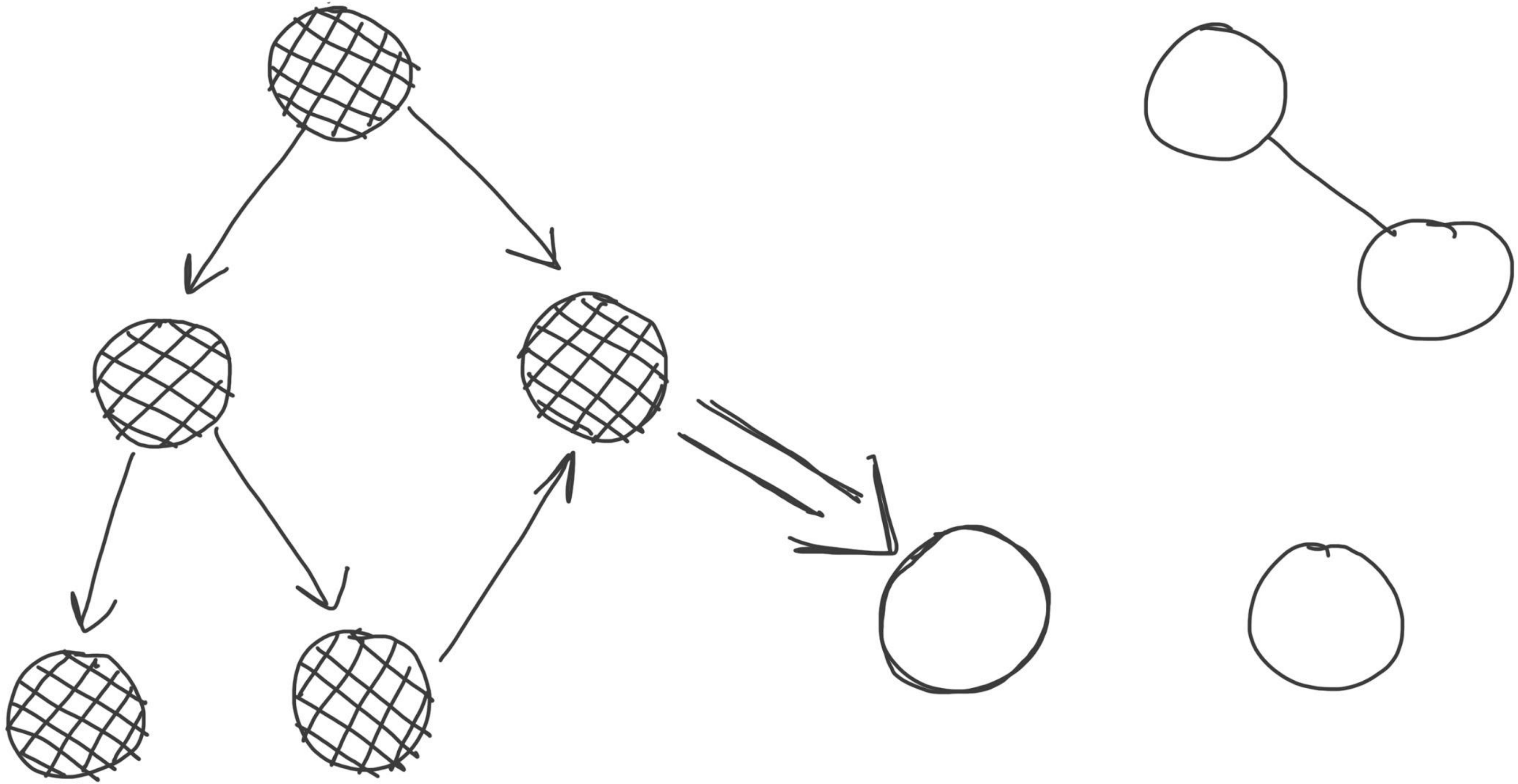
by Dijkstra™











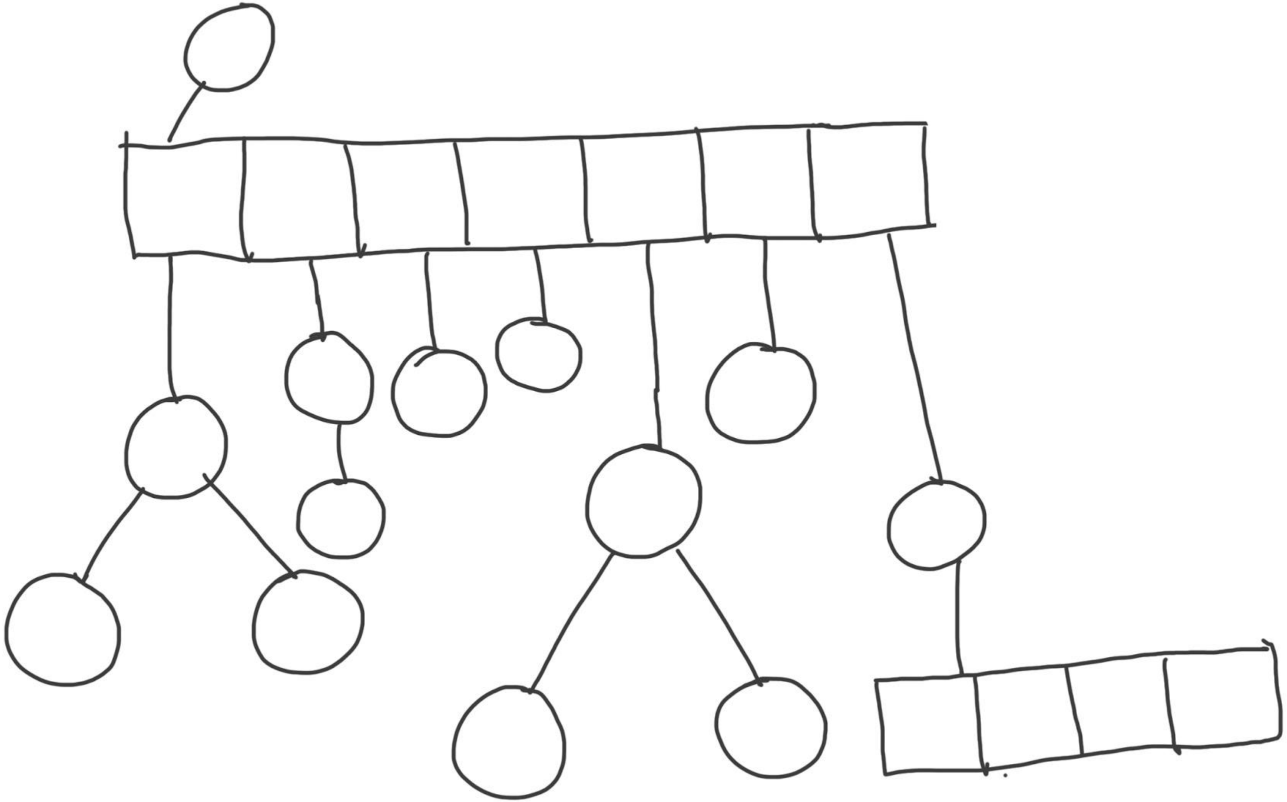
What is a barrier?

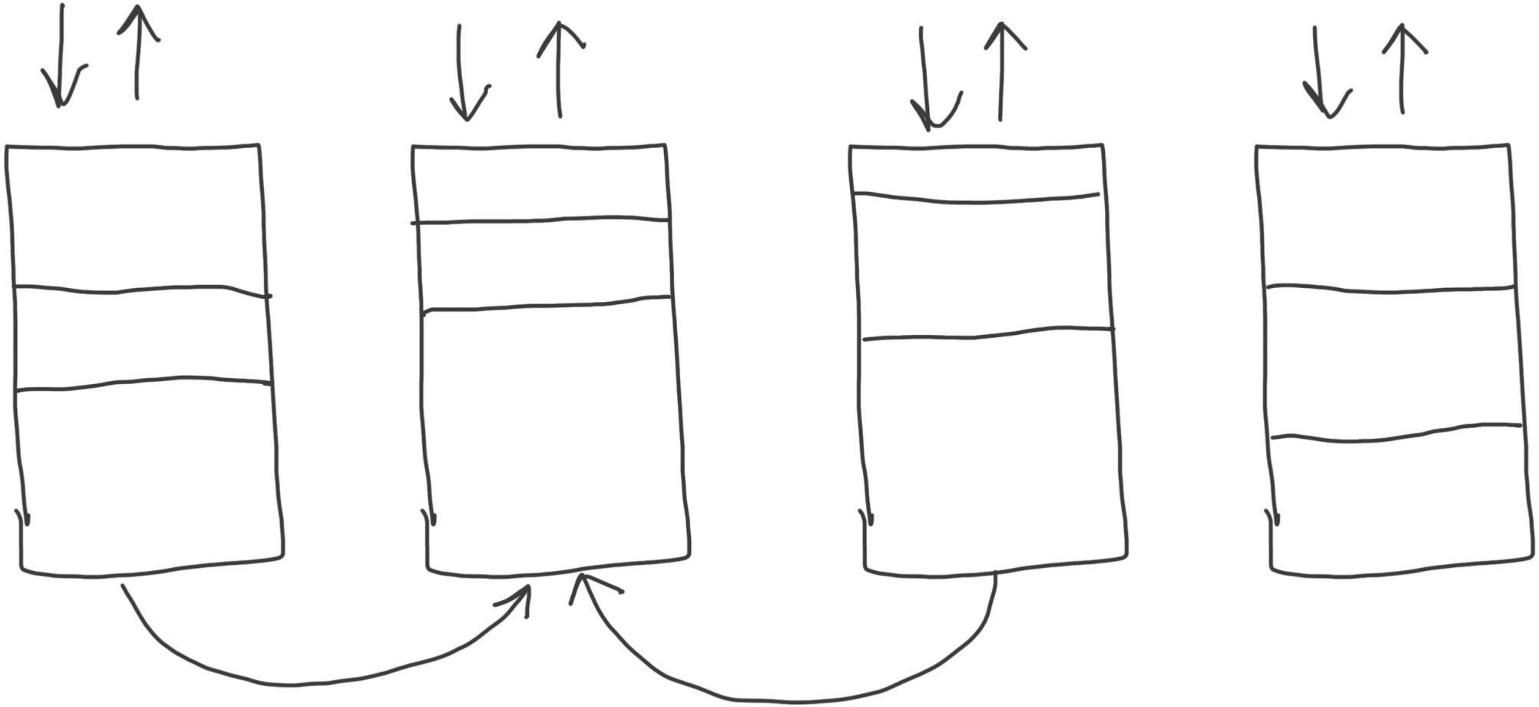
if statement

else branch is very rare

CPU branch predictor

Parallel marking





Lazy Sweep

Generational

Temporary data

Major vs Minor GC

Pointers from Old objects to new objects

Remembered Set

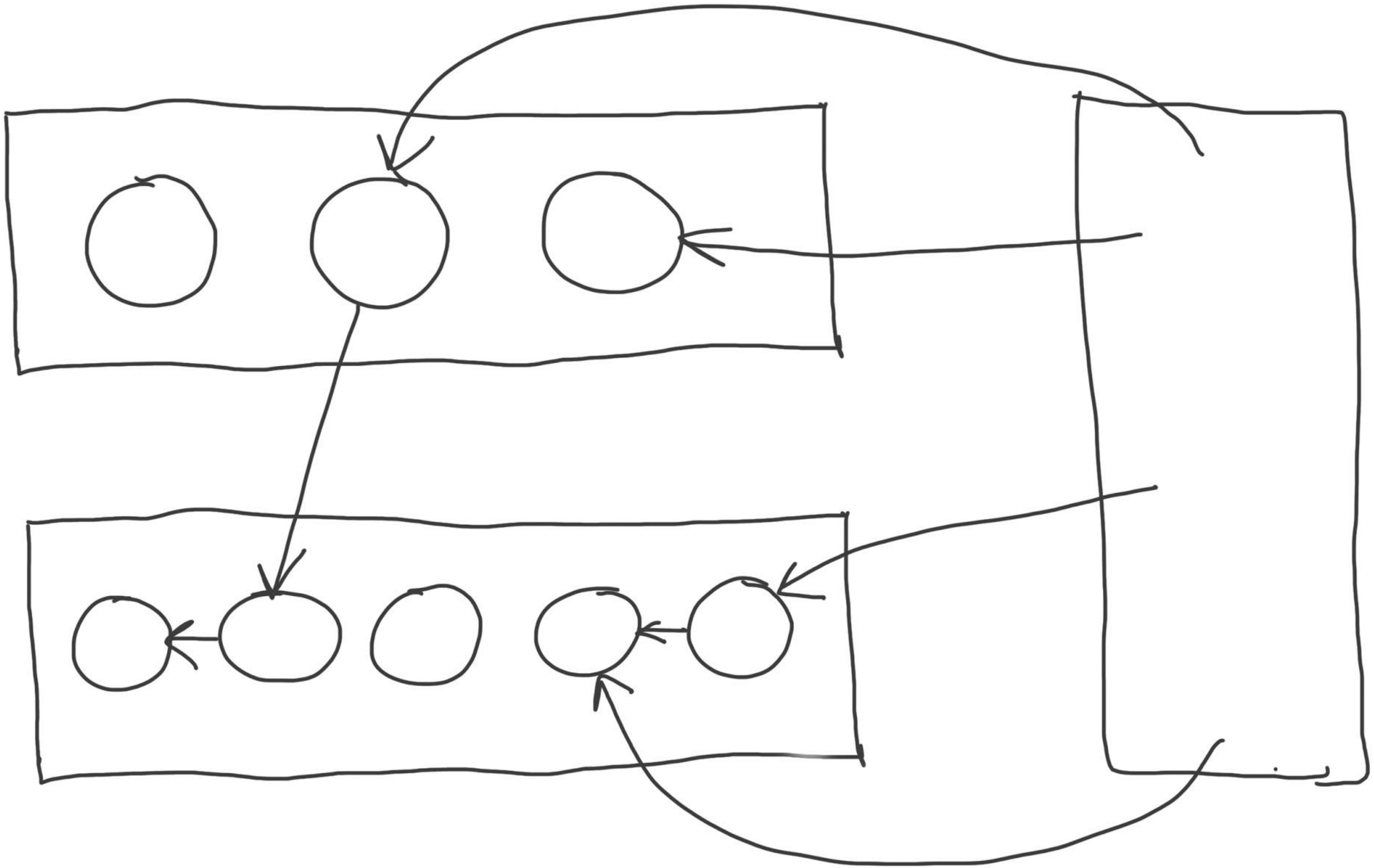
Major GC

Scan roots only
Bigger object graph

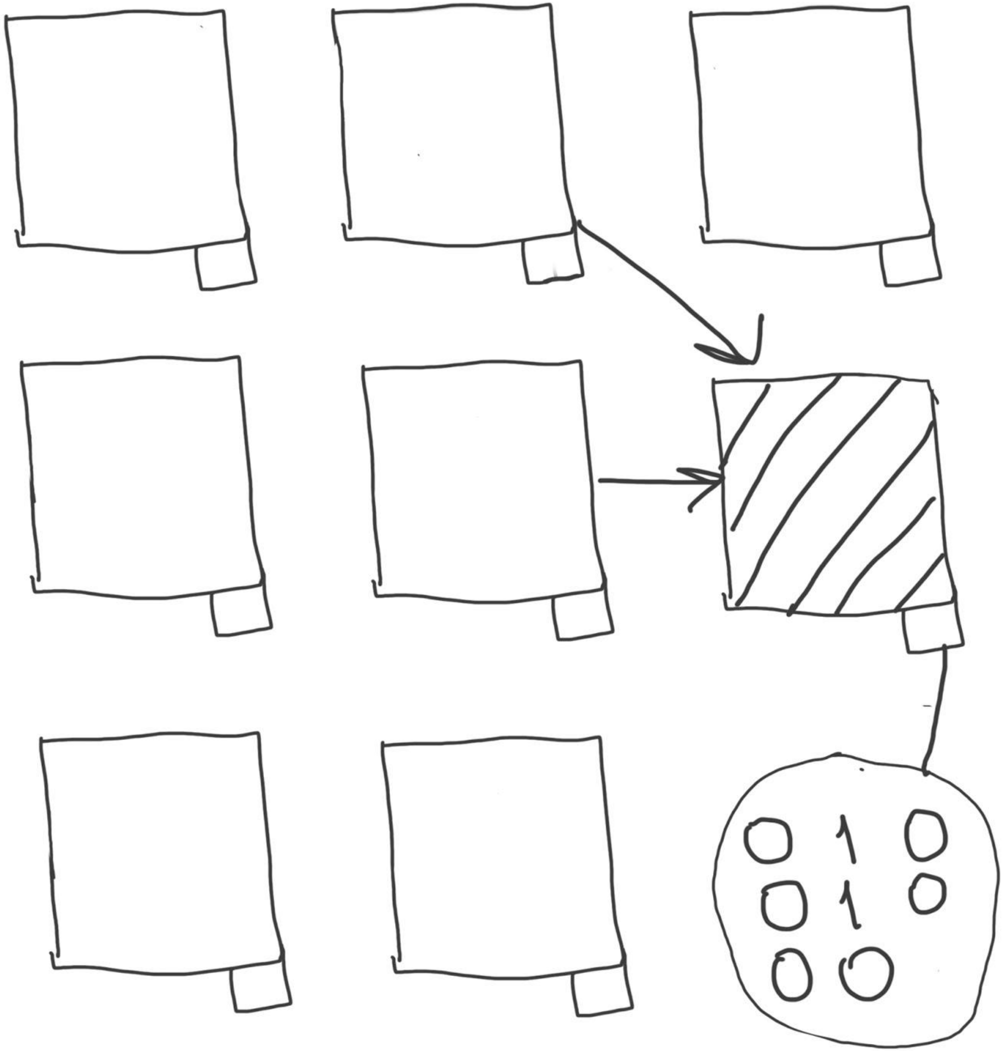
Minor GC

Scan roots + RS

Small object graph

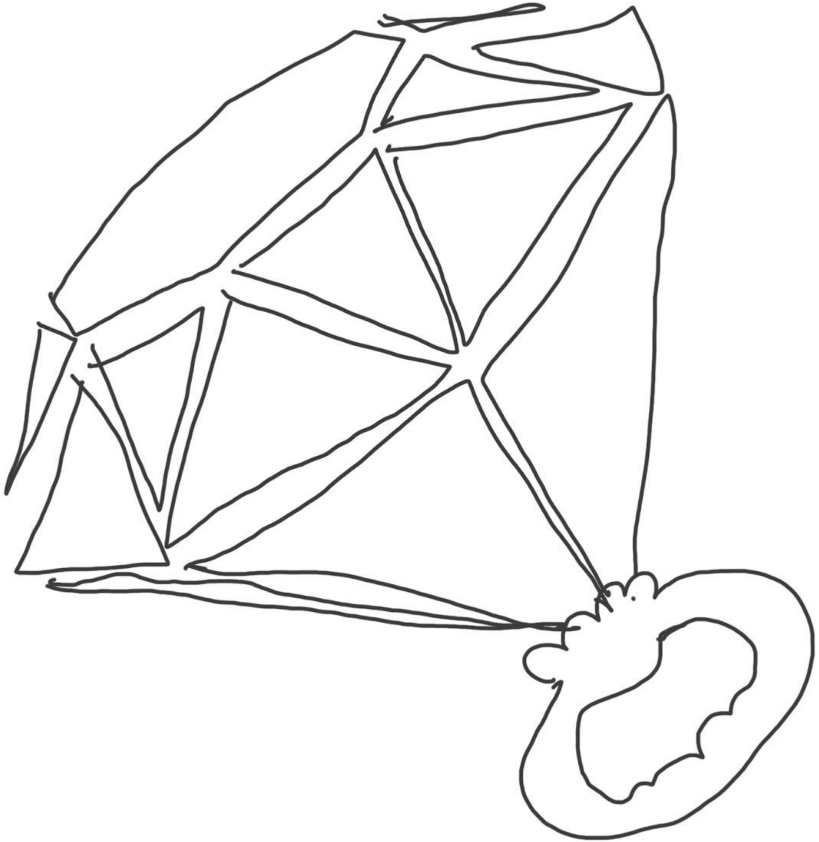


Modern GCs are hybrid


0 1 0
0 1 0
0 0

♡ Ruby ♡

Because you care ♡



C-extensions

Can’t move objects if their references are passed to an extension

Can’t add WB

RGen GC

2 types of objects

WB-protected
WB-unprotected

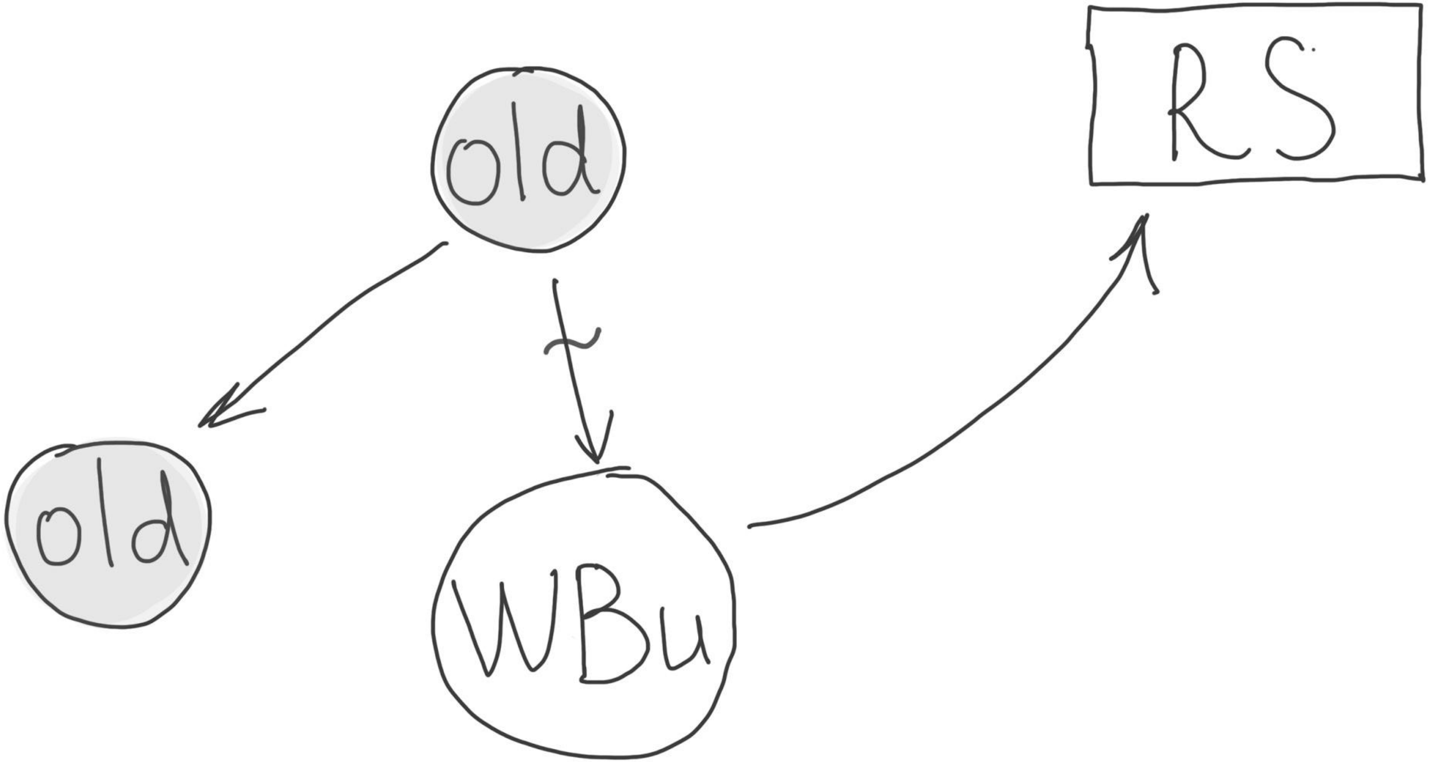
WBu are never OldGen

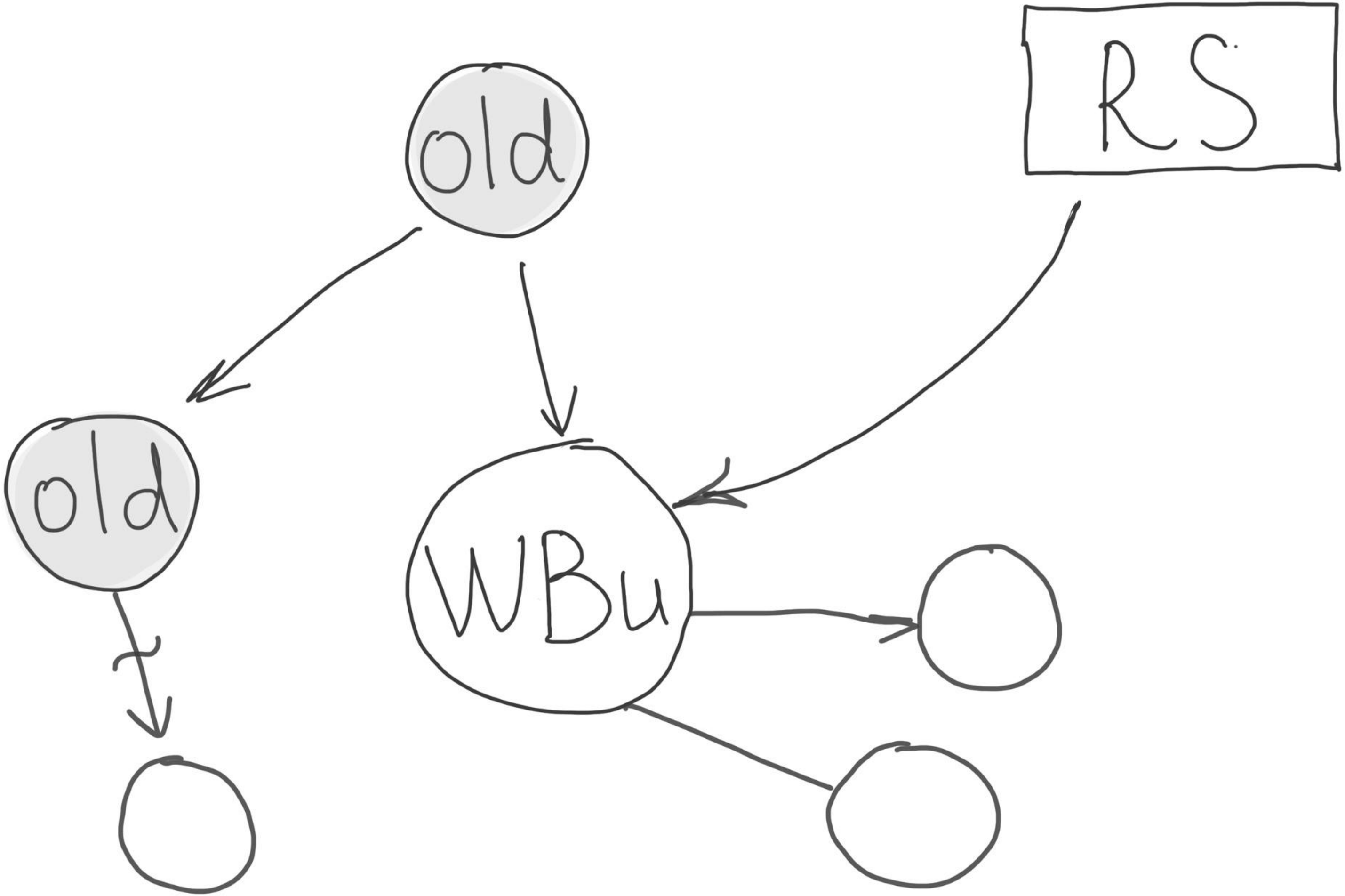
OldGen -> WBu

WBu to Remembered Set

Mark WBus on every minor GC

1 stw to mark all WBu in RS


RS
old
old
WBu


RS
old
old
WBu

Adding compaction for WBp

Javascript

YES WE CAN

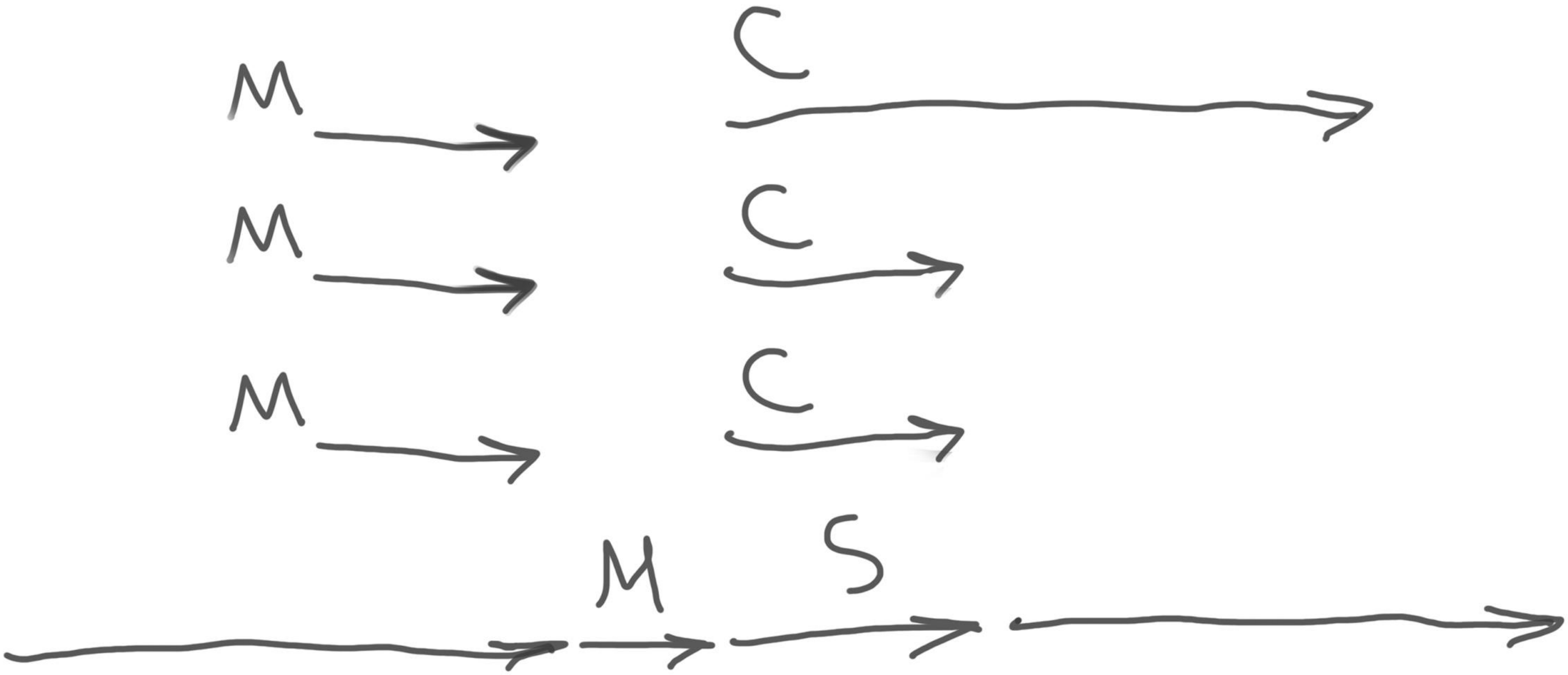
V8

Minor GC
Parallel

Major

Parallel marking

Parallel / Concurrent
Compact || Sweep


C
M
C
M
C
M
M
S

When to trigger GC?

Out-of-Bounds GC

request? Minor GCs

response is sent? Major GC

Firefox
Run GC in background tabs
first instead of current tab

Chrome
Animation frame

Walking the memory

Cache locality

GPU

OS Pages
pre-forked processes
malloc zones
GC Pages
Remembered Sets
Barriers

- Memory management is hard and...

...fascinating!

STAY CURIOUS!