Martin is getting the projector to work with his laptop.

Threads, Critical Sections, and Termination

Martin McClure

Thread

vs.

Process

Shared State

…sigh

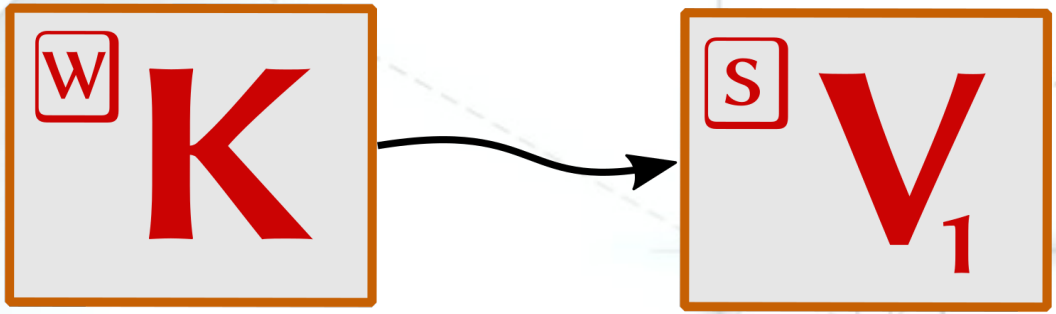
GemBuilder® for Smalltalk

GBS

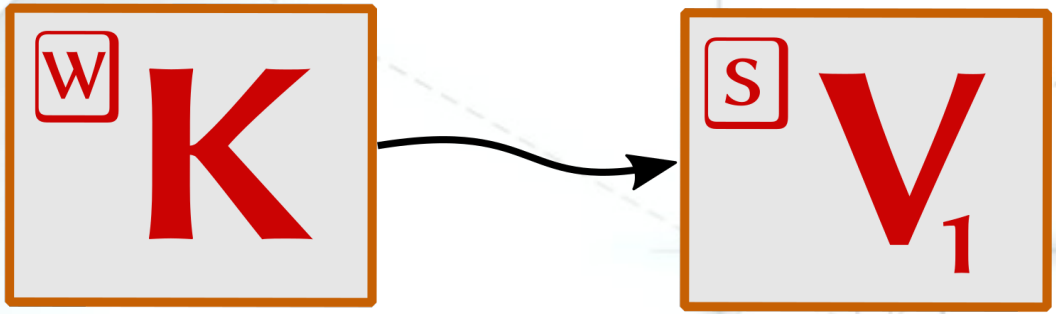
Mapping Dictionaries

ClientMap

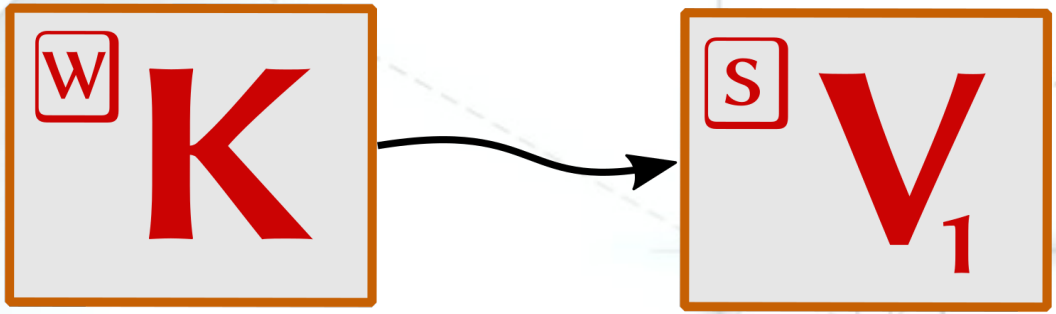
- Identity Dictionary
- ~2M keys
- ~100K updates/s
- ~(1-10)×100K lookups/s


W
K
S
V1

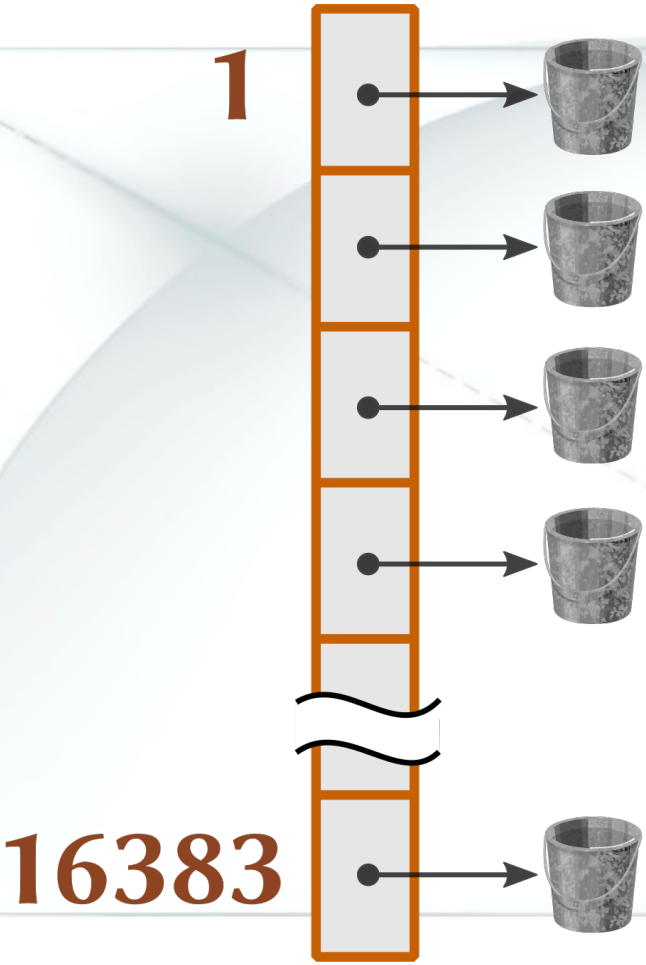
32-bit VW

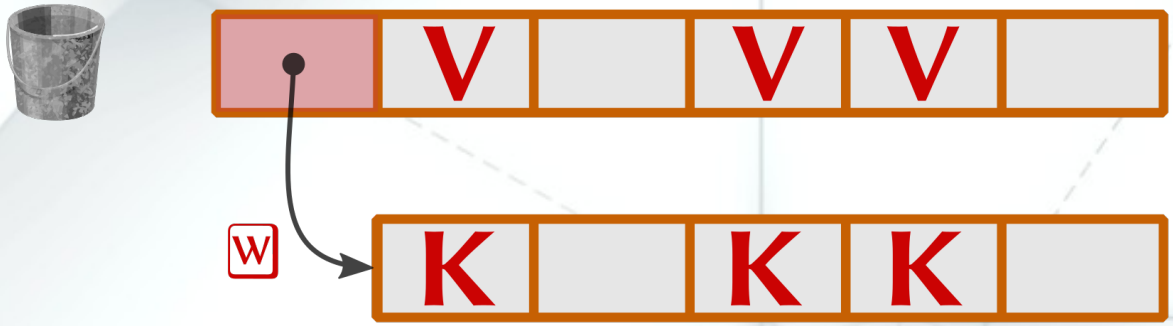


32-bit VW



2^{14} hash values


1
16383


V
V
V
W
K
K
K

Safe sharing of this structure?

Semaphore

Semaphore

...performance?

Performance

"Premature optimization is the root of all evil."

--Donald Knuth

Simplicity

Clarity

Performance

Performance

5ms

Semaphore

...performance?

Time microsecondsToRun: [1000000 timesRepeat: []]

```
Time microsecondsToRun:
    [1000000 timesRepeat: []]
```

3ns-4ns

3ns-4ns

[sem critical: []]

[sem critical: []]

~70ns

```
[dict
  delegateAtClient: key]
```

[dict
delegateAtClient: key]

~40ns

```
[sem critical:
 [dict
  delegateAtClient: key]]
```

```
[sem critical:
 [dict
  delegateAtClient: key]]
```

~130ns

Batched semaphore for modifying

No semaphore for lookup

Hash Bucket Lookup

```
index := linear search for key.
^index == 0
  ifTrue: [nil]
  ifFalse:
    [value := self basicAt: index.
    (keys at: index) == key
      ifTrue: [value] ifFalse: [nil]]
```

Hash Bucket Add

```
index := freeHead.
[] ensure:
    [freeHead := keys at: freeHead.
    self basicAt: index put: value.
    keys at: index put: key.
    size := size + 1].
```

…but there's a hole

Hash Bucket Lookup

```
index := linear search for key.
^index == 0
  ifTrue: [nil]
  ifFalse:
    [value := self basicAt: index.
    (keys at: index) == key
      ifTrue: [value] ifFalse: [nil]]
```

Hash Bucket Add

```
index := freeHead.
[] ensure:
    [freeHead := keys at: freeHead.
    self basicAt: index put: value.
    keys at: index put: key.
    size := size + 1].
```

[“empty block”] ensure: [“non-empty block”].

Hash Bucket Add

```
index := freeHead.
[] ensure:
    [freeHead := keys at: freeHead.
    self basicAt: index put: value.
    keys at: index put: key.
    size := size + 1].
```

Semaphore
critical: mutuallyExcludedBlock

```
self waitIfCurtailedSignal.
^mutuallyExcludedBlock
    ensure: [self signal]
```

testTerminateInEnsure

```
| process count random delay |
random := Random new.
10 timesRepeat:
  [process :=
      [count := 0.
        [] ensure:
            [10 timesRepeat:
                [count := count + 1.
                1000000 timesRepeat: [12 factorial]].
            count := count + 1]]
            forkAt: Processor activeProcess priority - 1.
    delay := (random next * 100) asInteger + 10.    "avoid 0-ms delay"
    (Delay forMilliseconds: delay) wait.
    self assert: process isTerminated not.
    process terminate.
    process priority: Processor activeProcess priority + 1.
    self
      assert: process isTerminated;
      assert: count equals: 11]
```

Terminating in a Critical Section

Hash Bucket Add

```
index := freeHead.
[] ensure:
    [freeHead := keys at: freeHead.
    self basicAt: index put: value.
    keys at: index put: key.
    size := size + 1].
```

Is this the best way?

Other Ways

- Semaphore
- #valueUninterruptably
- #valueUnpreemptively

Semaphore

Semaphore

- No termination protection

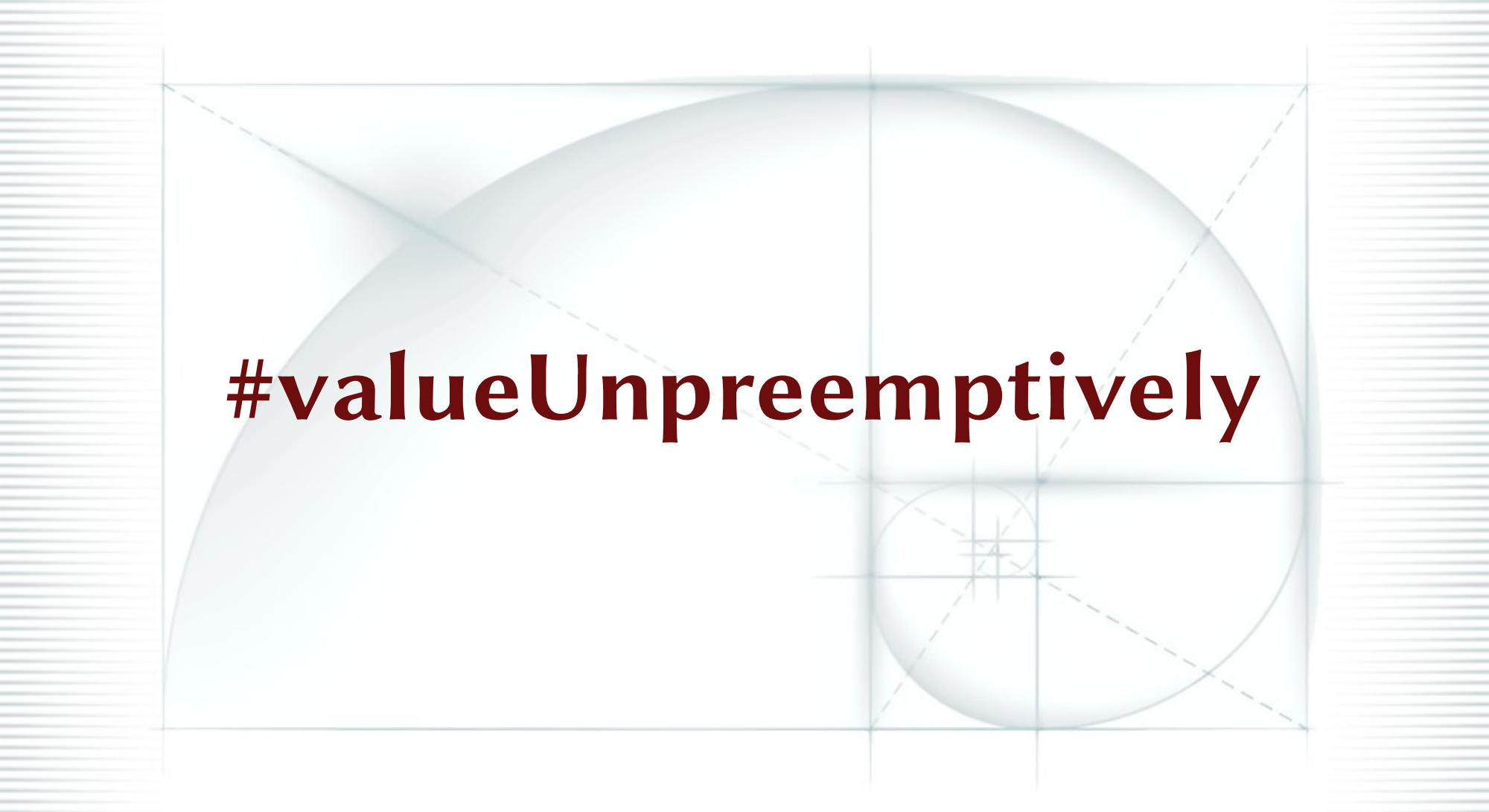
Semaphore

- No termination protection
- 70ns

#valueUninterruptably

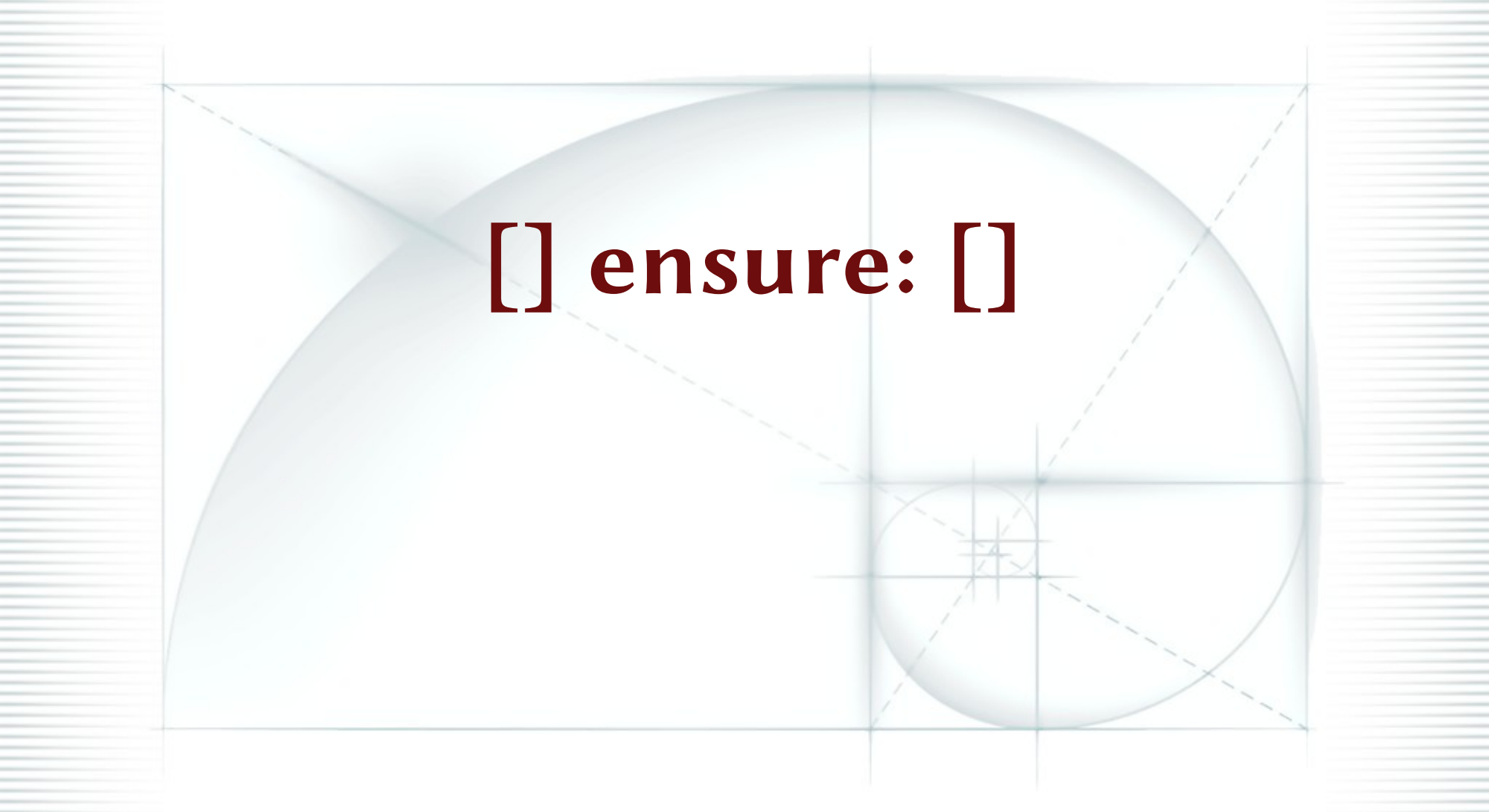
[]valueUninterruptably

~87ns


#valueUnpreemptively

[]valueUnpreemptively

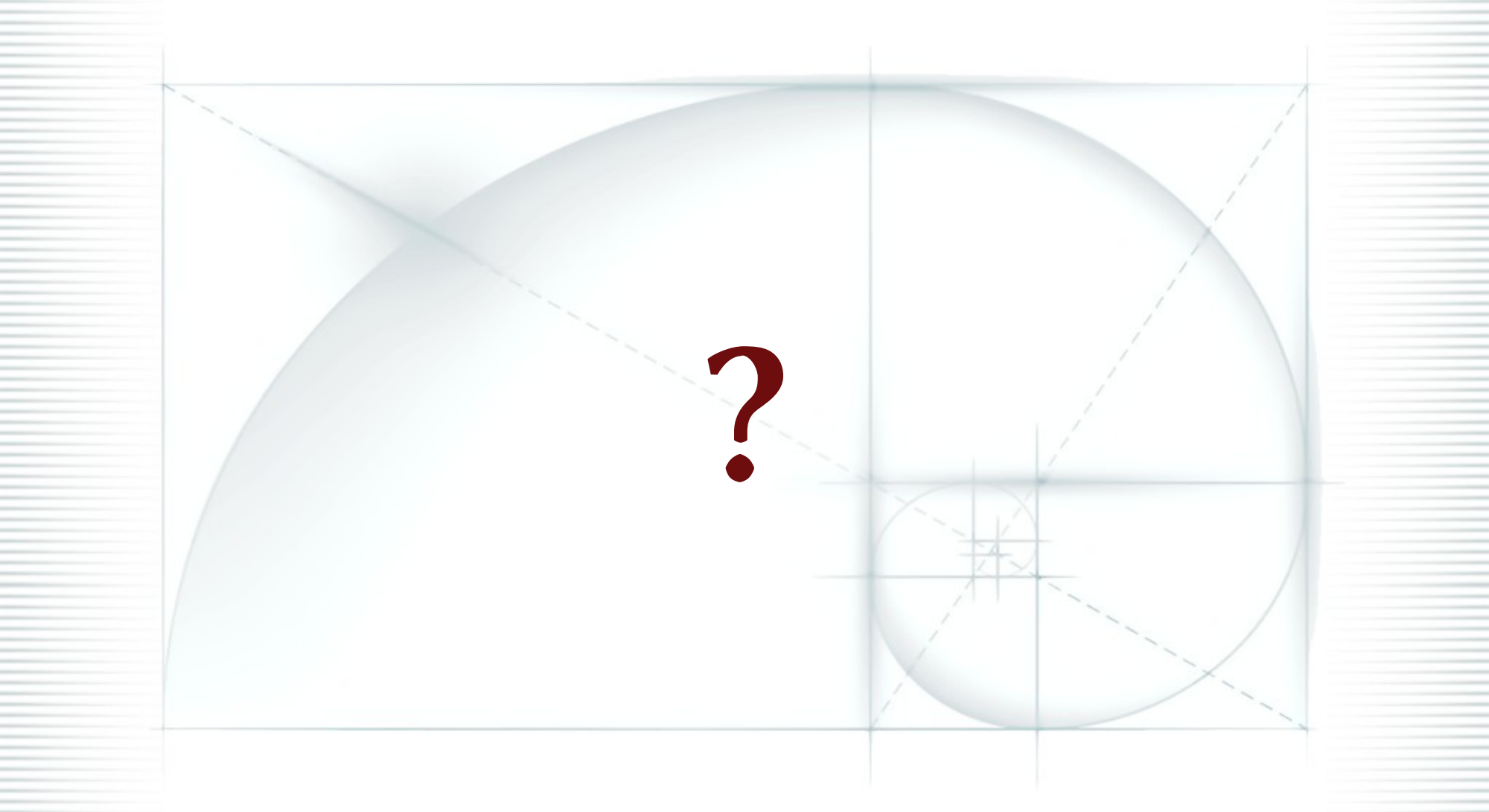
~175ns


[] ensure: []

[] ensure: []

~15ns

Final Thoughts



Threads, Critical Sections, and Termination

Martin McClure