How we measure and optimize for RAIL in V8's GC

mlippautz@, ulan@

BlinkOn 6, Munich

Contents

V8 GC and RAIL → **Update on Optimizations** → **Reflection and Future**

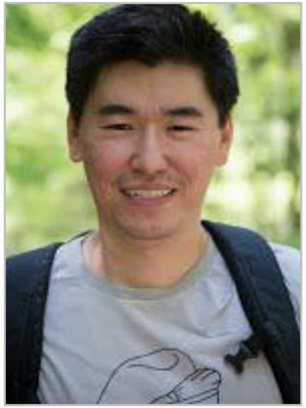
Who We Are - V8 GC Team Munich

Hannes
Payer

Jochen
Eisinger

Marcel
Hlopko

Michael
Lippautz

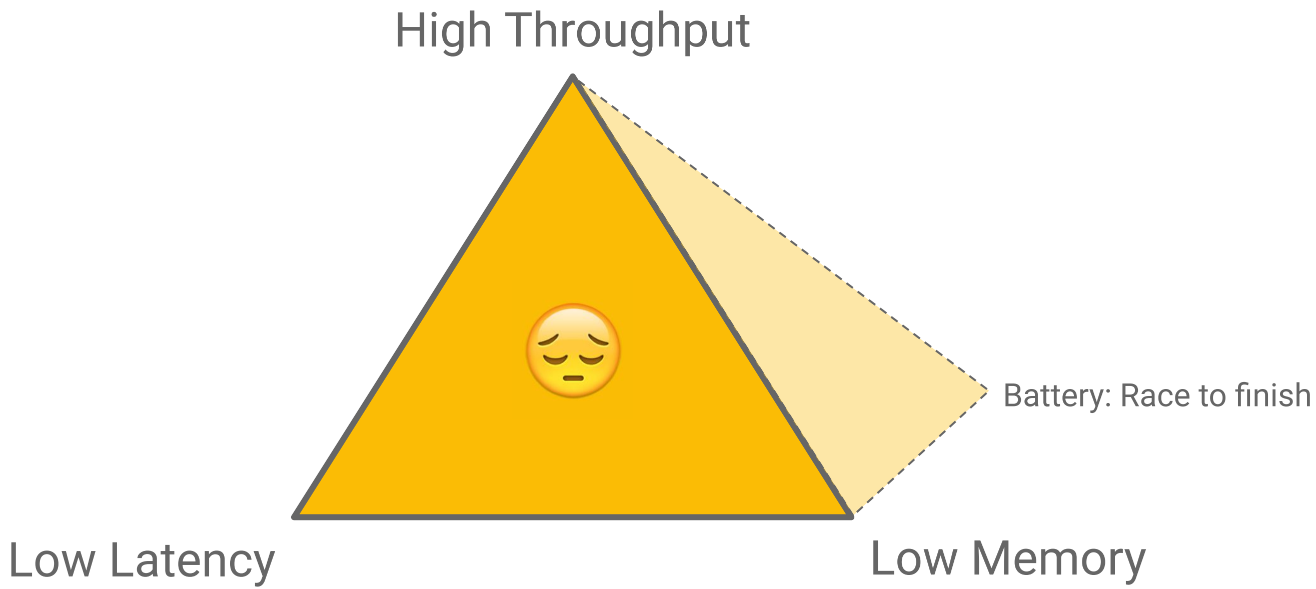


Ulan
Degenbaev

V8 GC and RAIL

Google

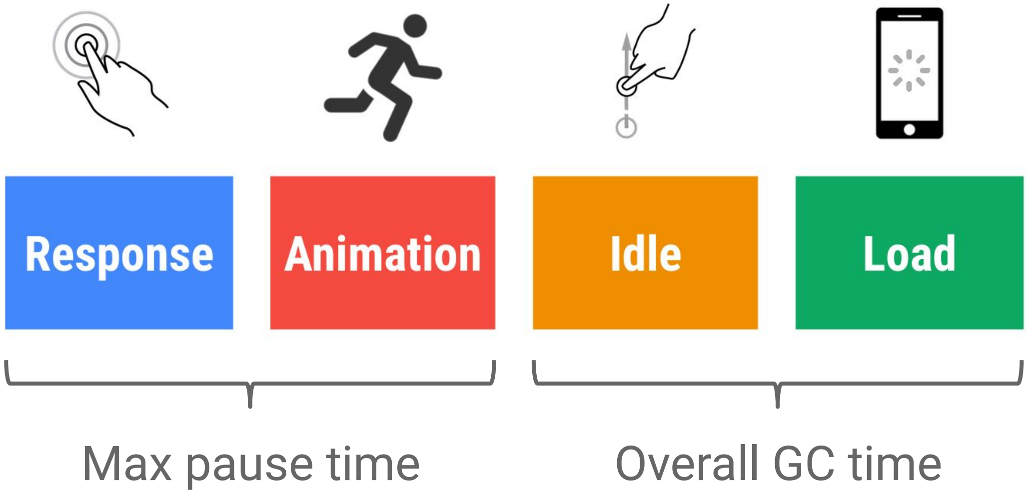
The Impossible Garbage Collection Triad



Google

RAIL and GC Metrics

- Ideally: No GC… ever
- In practice: Prioritize

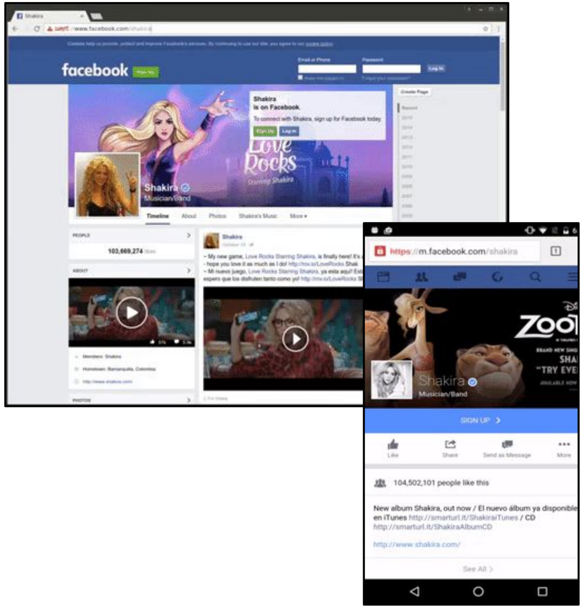


Fine print

Without consuming too much memory 😊

Real World Benchmarking

- Catapult (Telemetry)
- Record/replay for real-world websites
- Mobile and desktop

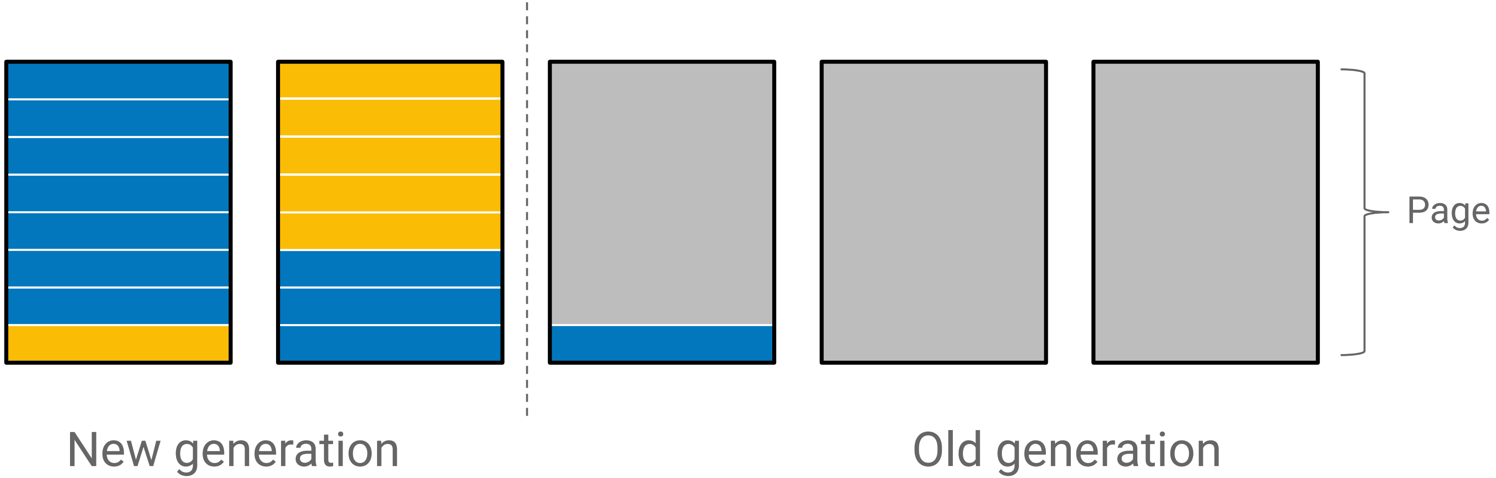


Google

V8's Generational Garbage Collector

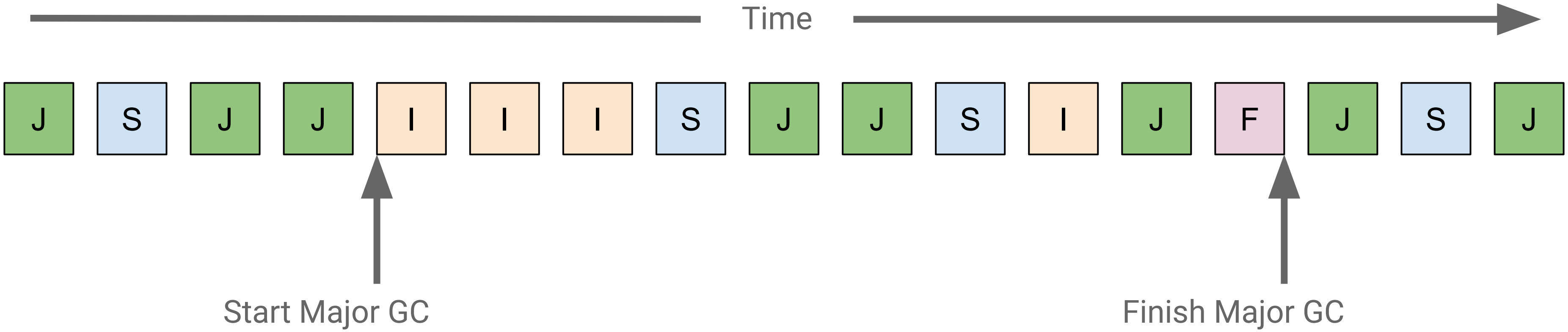
Generational hypothesis: "Most objects die shortly after their allocation"

- New generation: Semi-space Scavenger (Cheney)
- Old generation: Mark-Compact (and Sweep)



GC Events

- **J** JavaScript code
- **S** Minor GC: Scavenger (~0-10 ms)
- **I** Major GC: Incremental Marking (~0.01-*CONFIGURABLE* ms)
- **F** Major GC: Final Mark-Compact Collection (~4-20 ms)



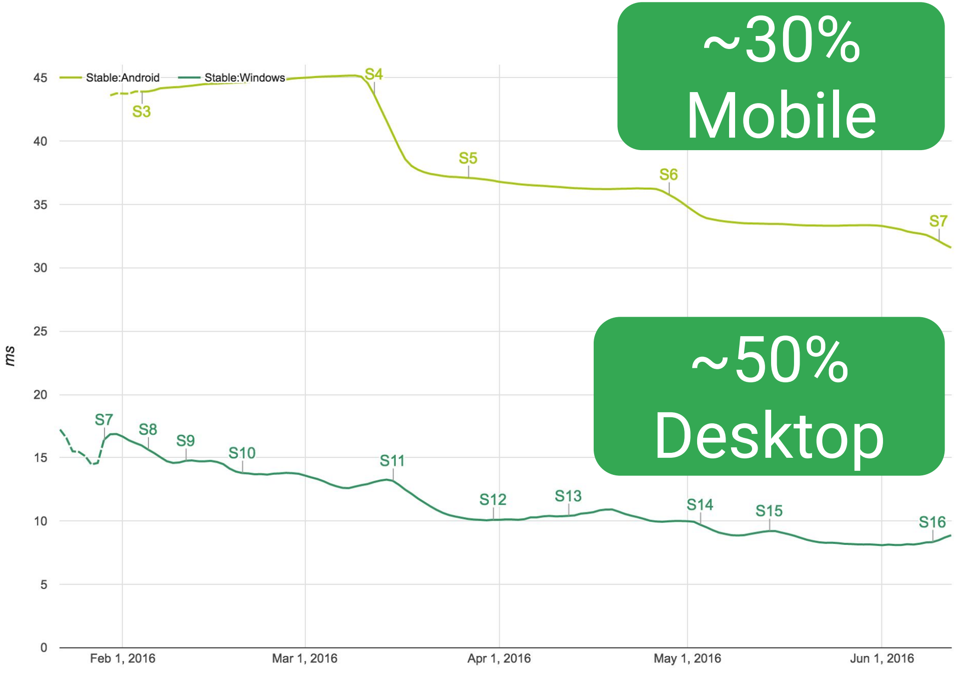
V8 GC Update

Focusing on A

Orinoco

Mostly parallel and concurrent GC
without strict generational boundaries
to
reduce jank and memory consumption
while providing high throughput

... landing incrementally



UMA: Major GC final pause **F** (50 %-ile)

Evacuation

- *Copy* objects within semi space of new generation
- *Move* objects from new to old generation
- *Copy* objects within the old generation
- Re-*write* remembered set pointers

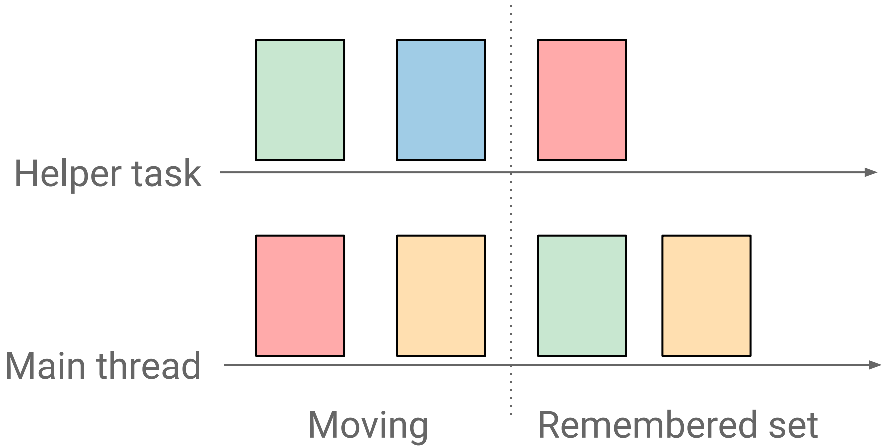
Writing memory is expensive

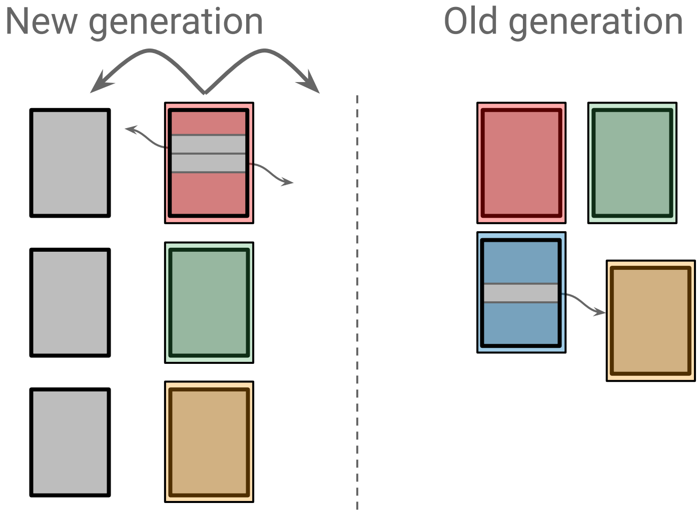
Orinoco

Design GC to utilize available resources

Parallel Evacuation during Major GC

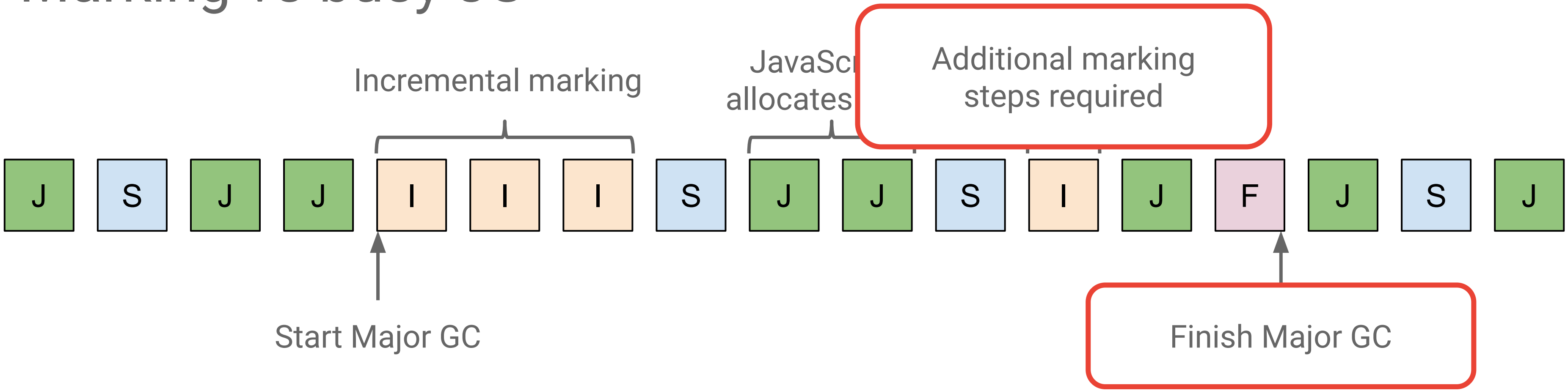
- Lock-step
  - Parallelize moving of memory based on pages
  - Parallelize processing remembered set pointers





Google

Marking vs busy JS

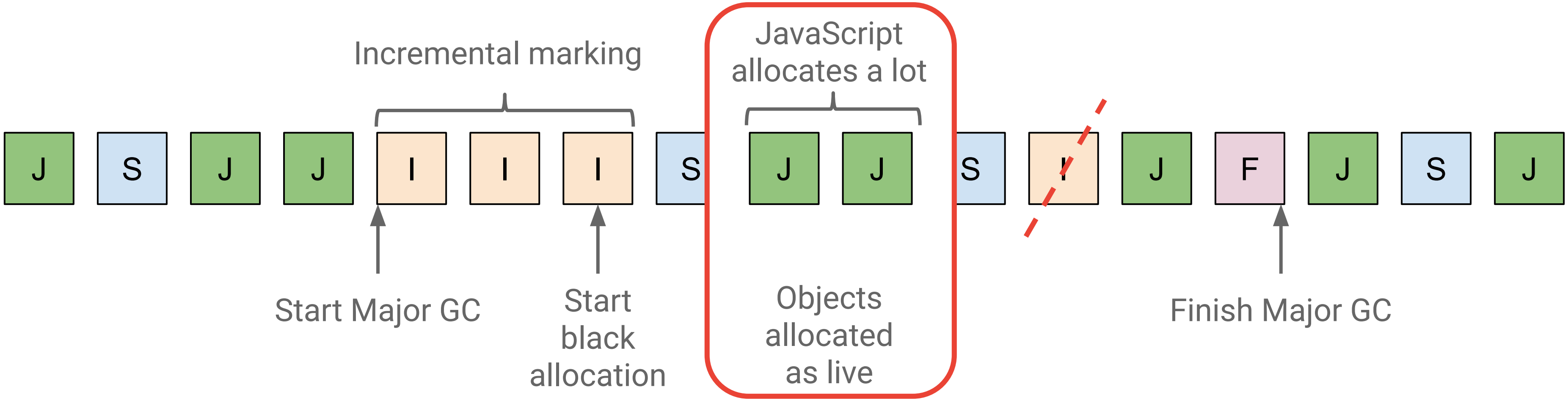


Live heap gets bigger

More marking steps, longer finalization pause

Google

🤘 Black Allocation

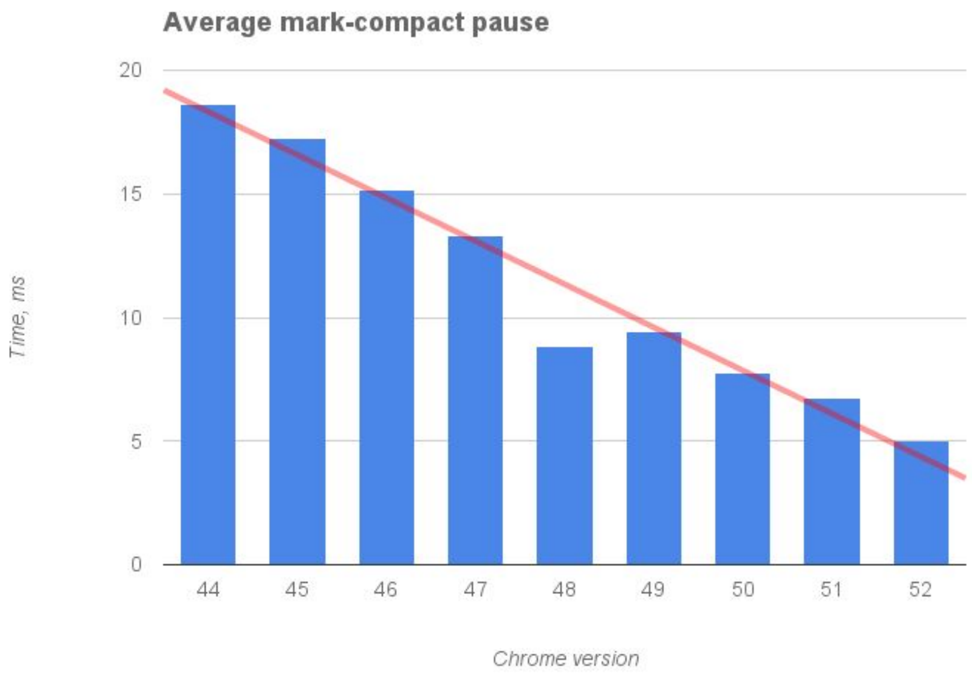


Assumption: Objects allocated during marking will survive the following GC

Objects are allocated as live (already marked)

Some Results: Facebook during A

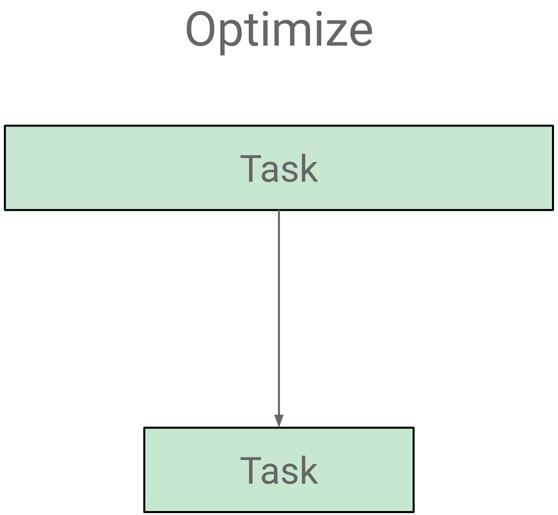
Some Results: Facebook during A

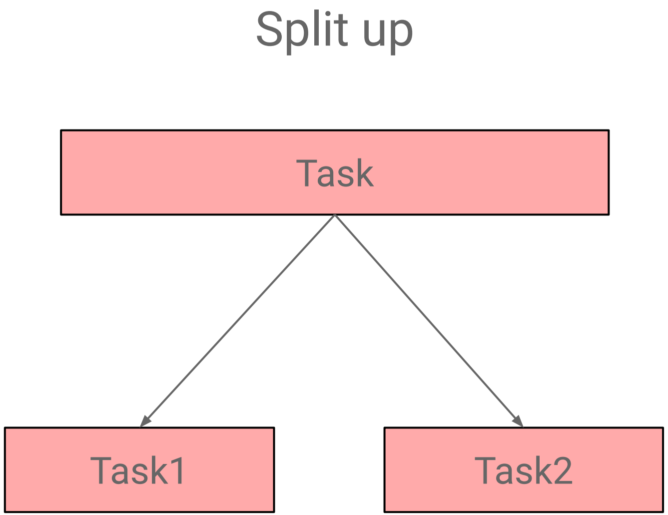


Google

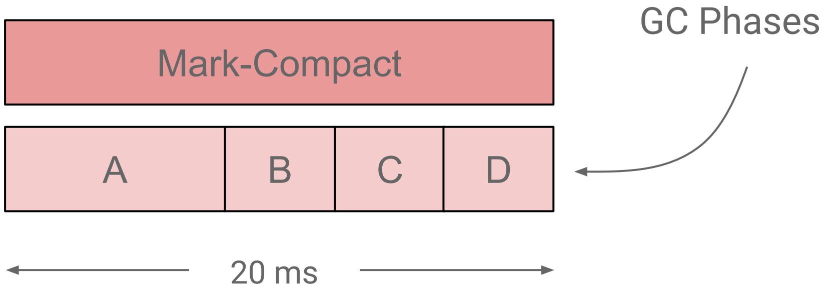
Reflection and Future

Two ways to reduce latency





Making GC more incremental

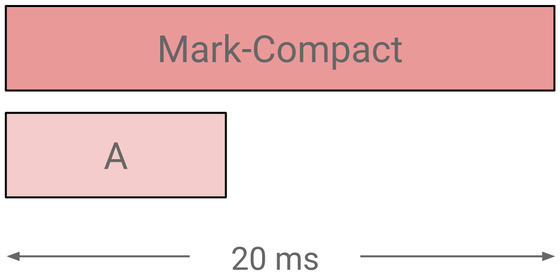


Google

Making GC more incremental

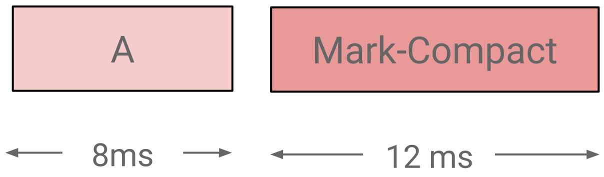
Before:

max-pause = 20ms



After:

max-pause = 12ms



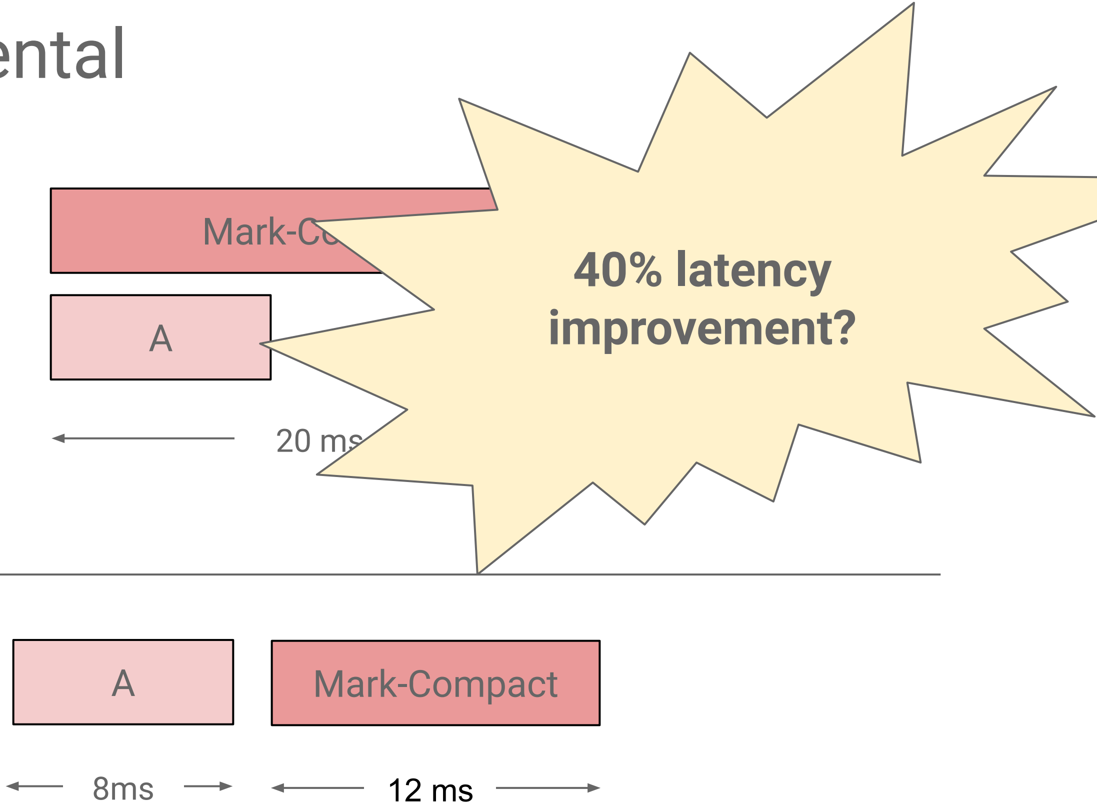
Making GC more incremental

Before:

max-pause = 20ms

After:

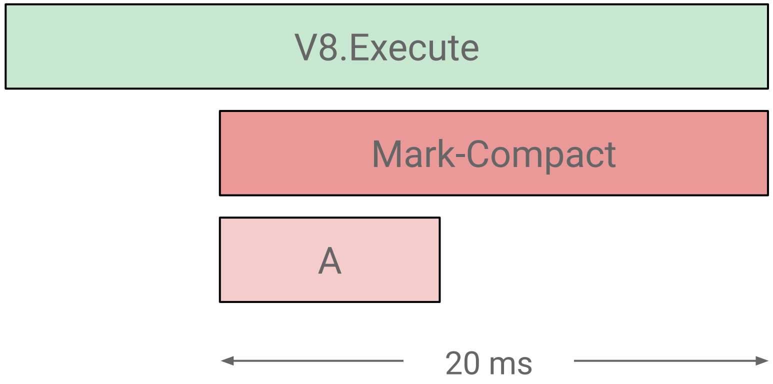
max-pause = 12ms



Making GC more incremental

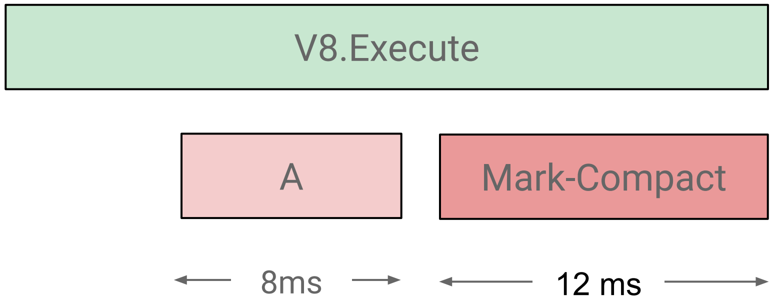
Before:

max-pause = 20ms



After:

max-pause = 12ms



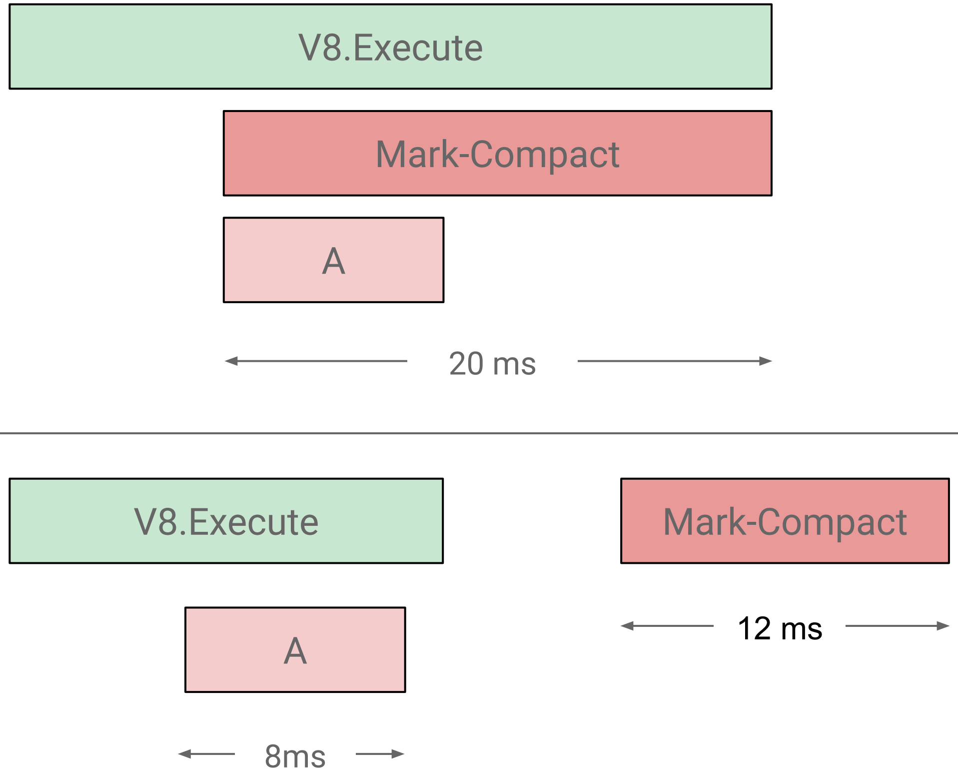
Making GC more incremental

Before:

max-pause = 20ms

After:

max-pause = 12ms



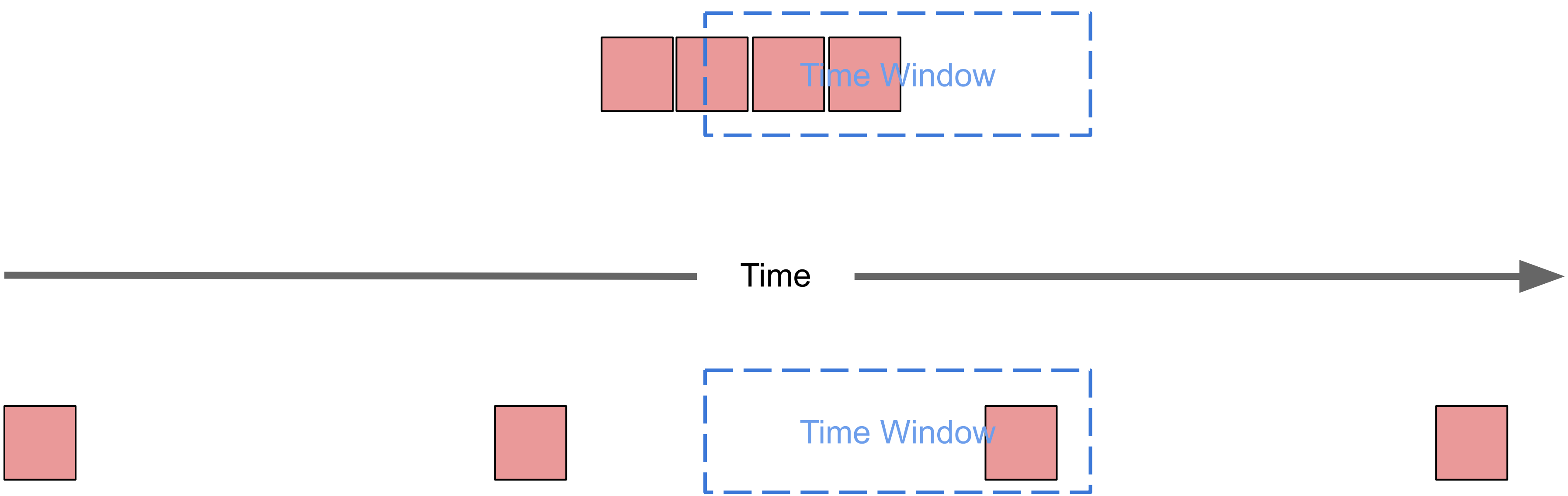
User facing metrics from PWM and system health

- Response: response latency
  - minimize GC work in critical time window
- Animation: animation latency
  - minimize GC work in critical time window
- Idle: responsiveness risk a.k.a responsiveness hazard
  - minimize GC work in tasks longer than 50ms
- Load: time until load finishes
  - minimize GC work during page load

Overfitting: delay GC in RAL, post many 49ms tasks in I

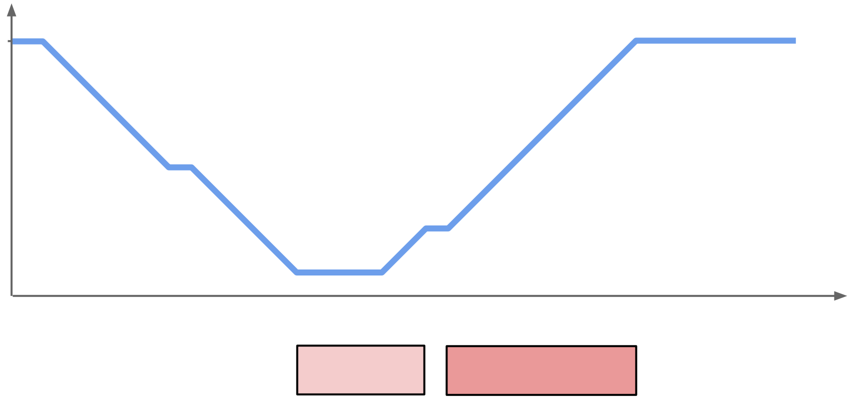
Simpler model: consider only RA, critical time window position is arbitrary

Uniform incremental steps



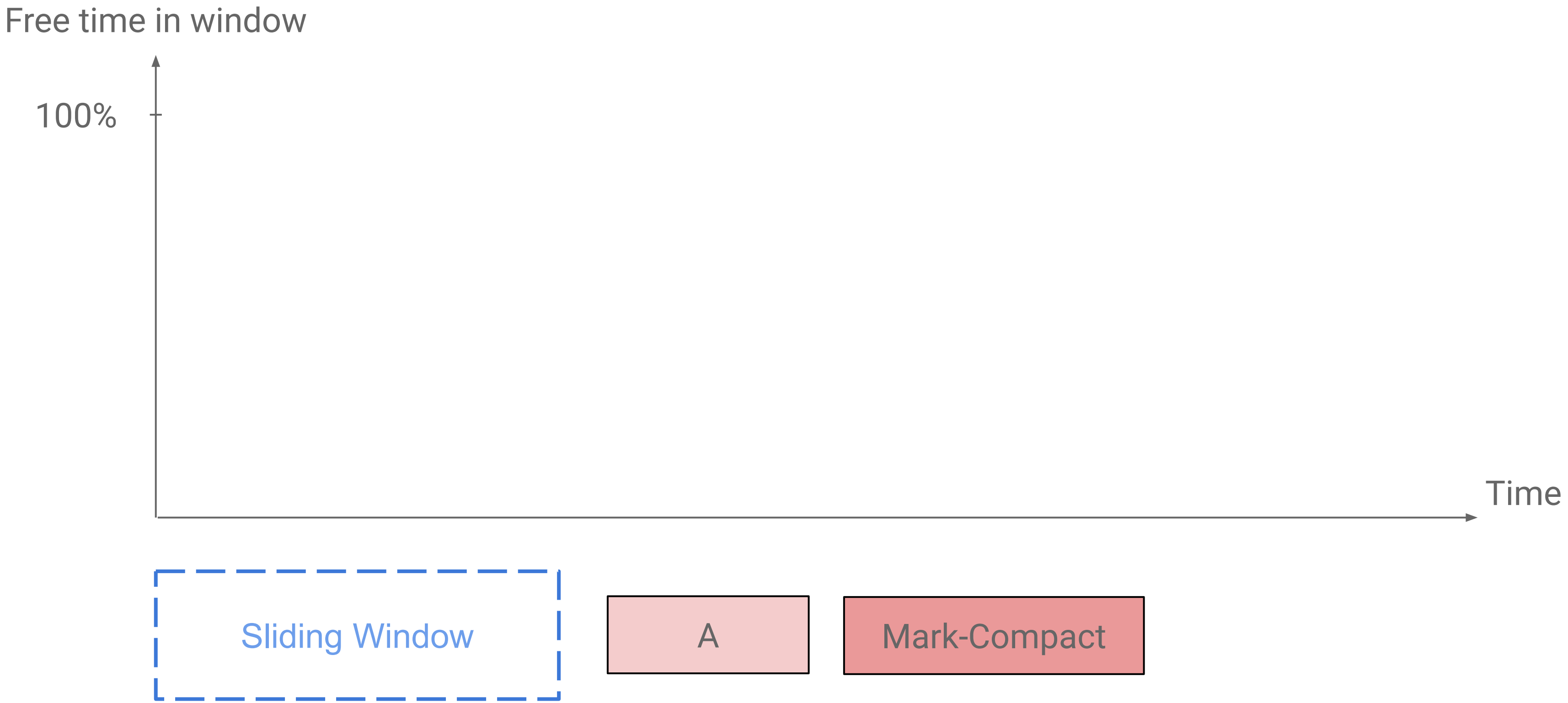
Google

Mutator Utilization

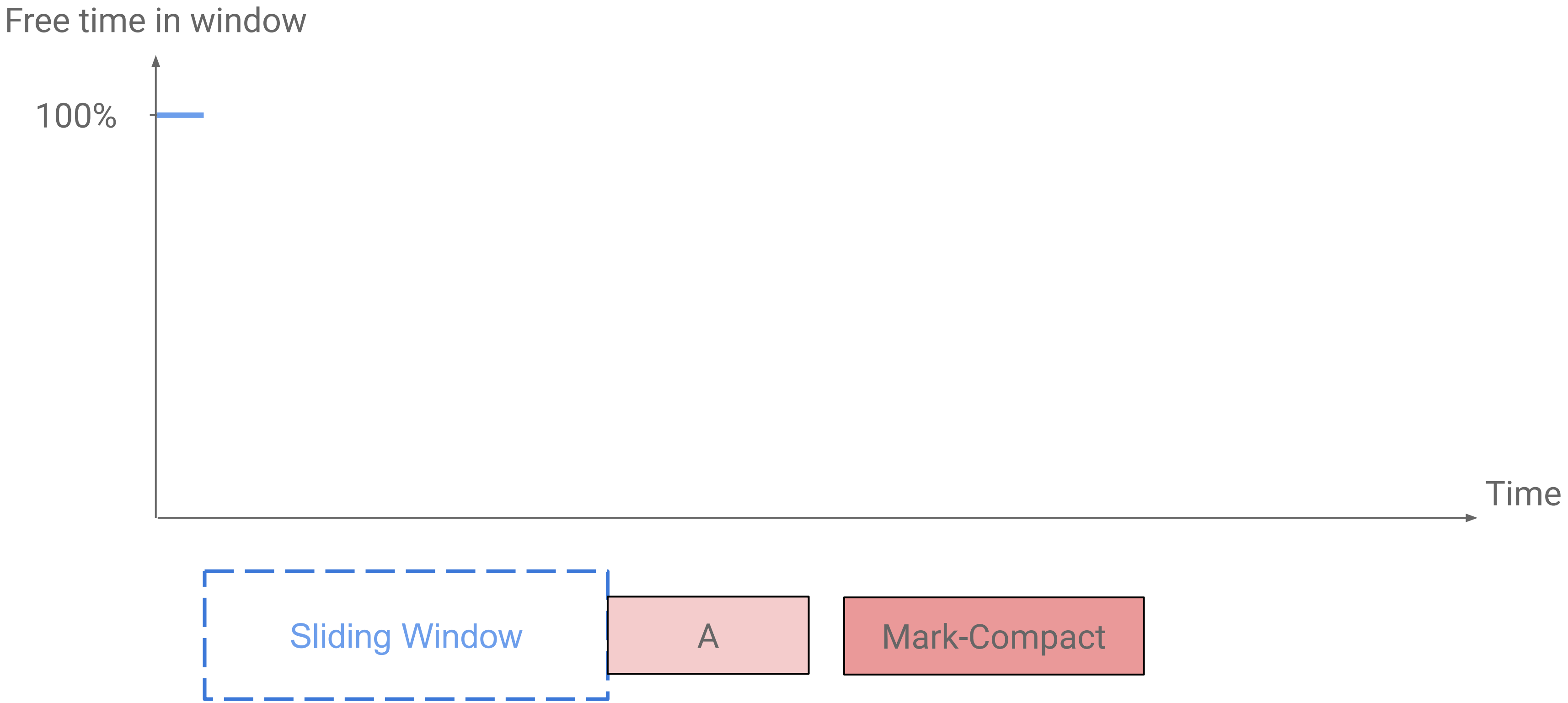


- Introduced by Cheng and Blelloch in 2001 for real-time GC.
- Called ***mutator utilization*** - application *mutates* reachability graph from GC POV
- Can be generalized to other tasks in Chrome

Utilization function

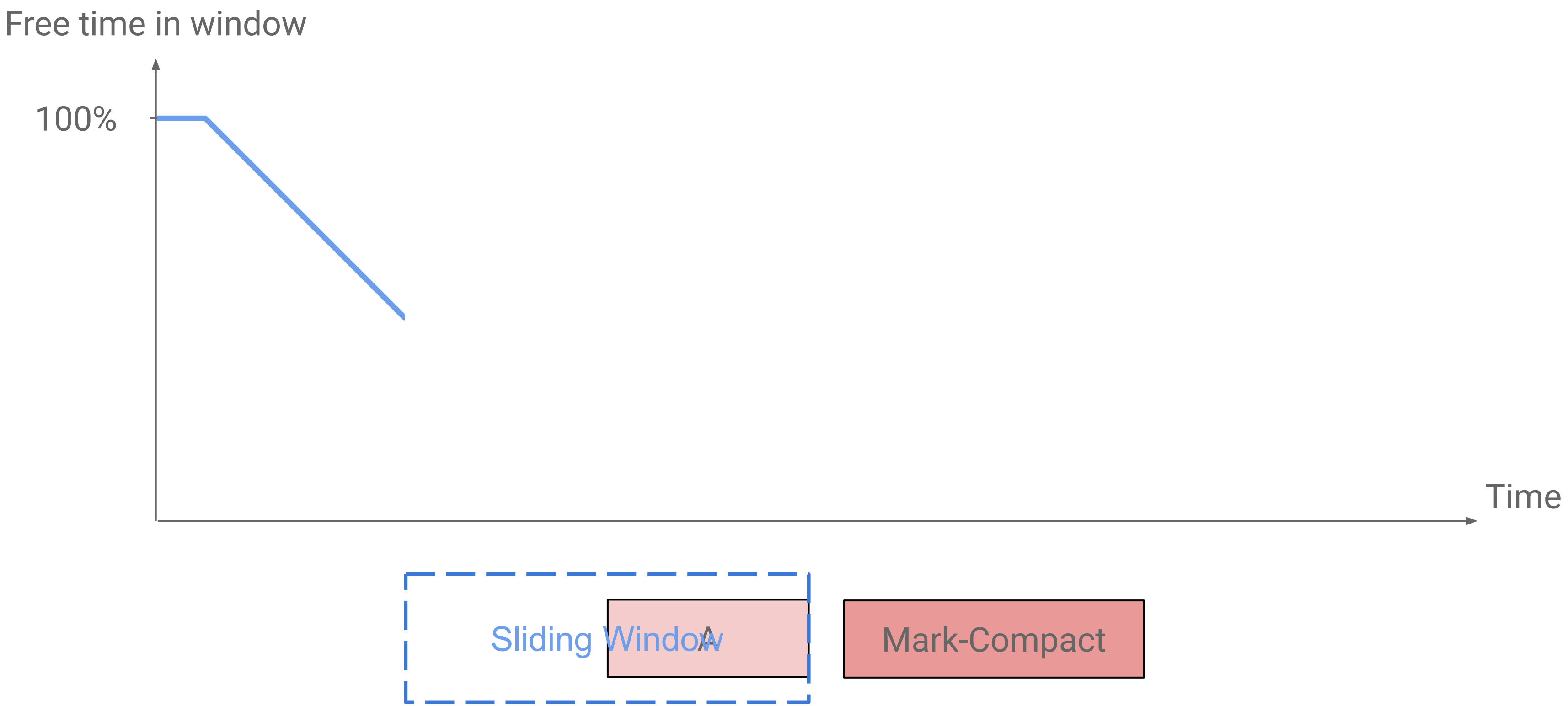


Utilization function



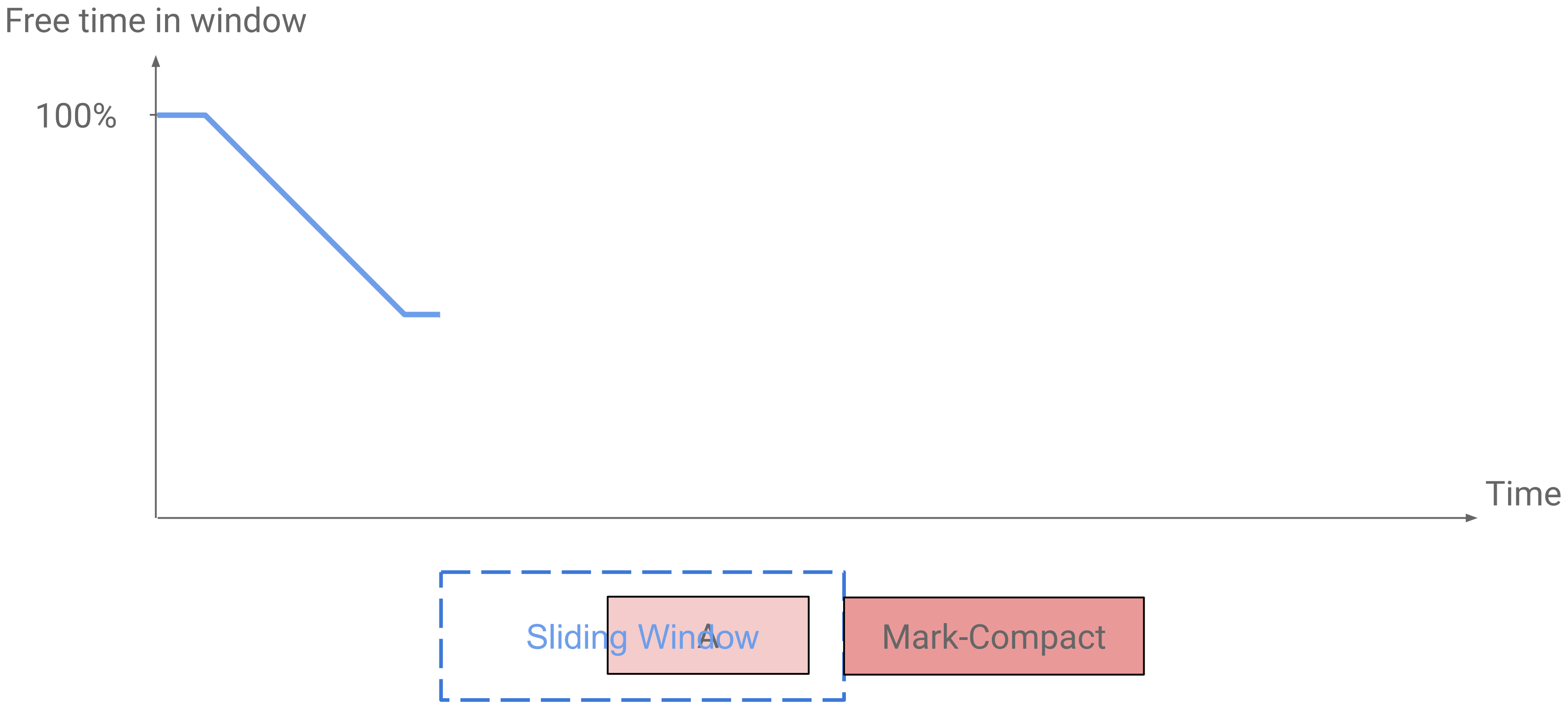
Google

Utilization function



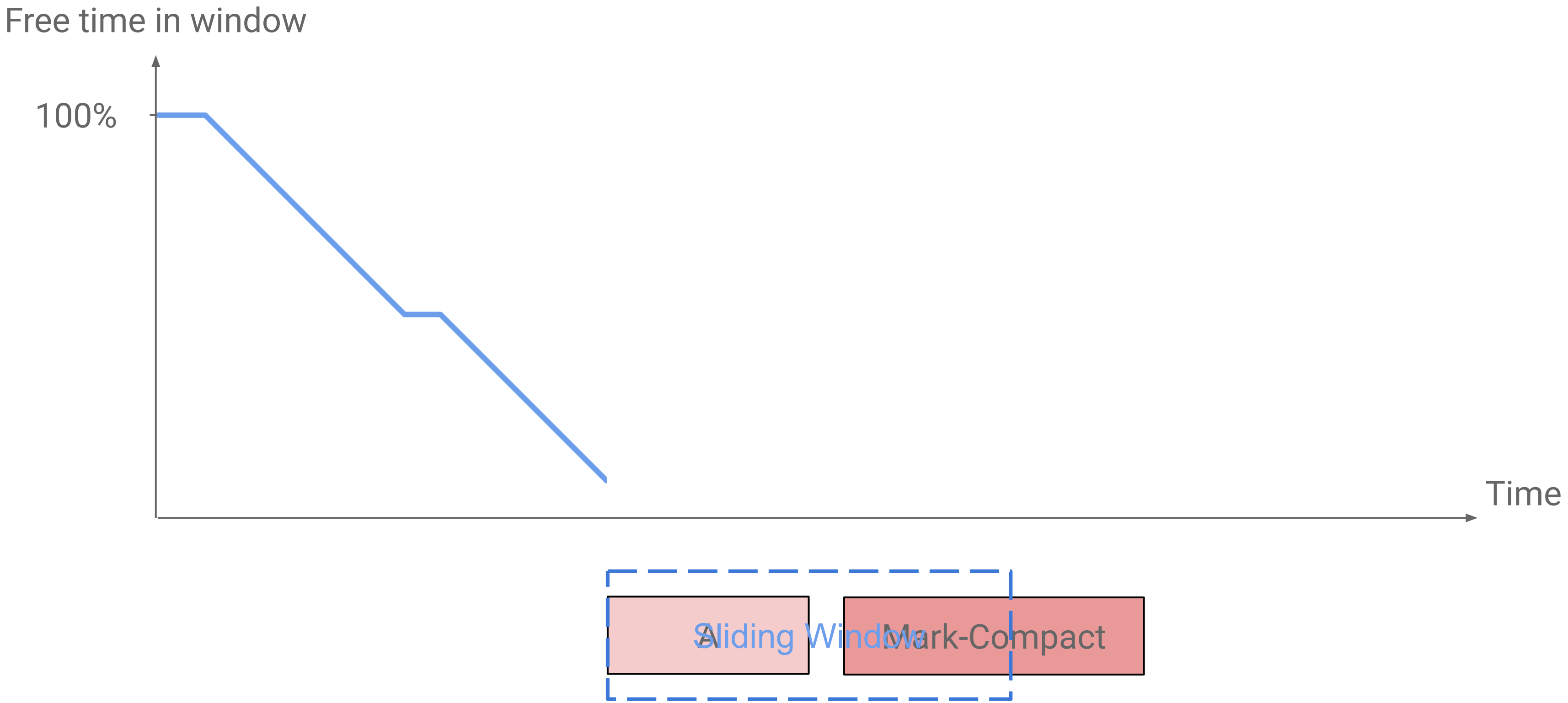
Google

Utilization function



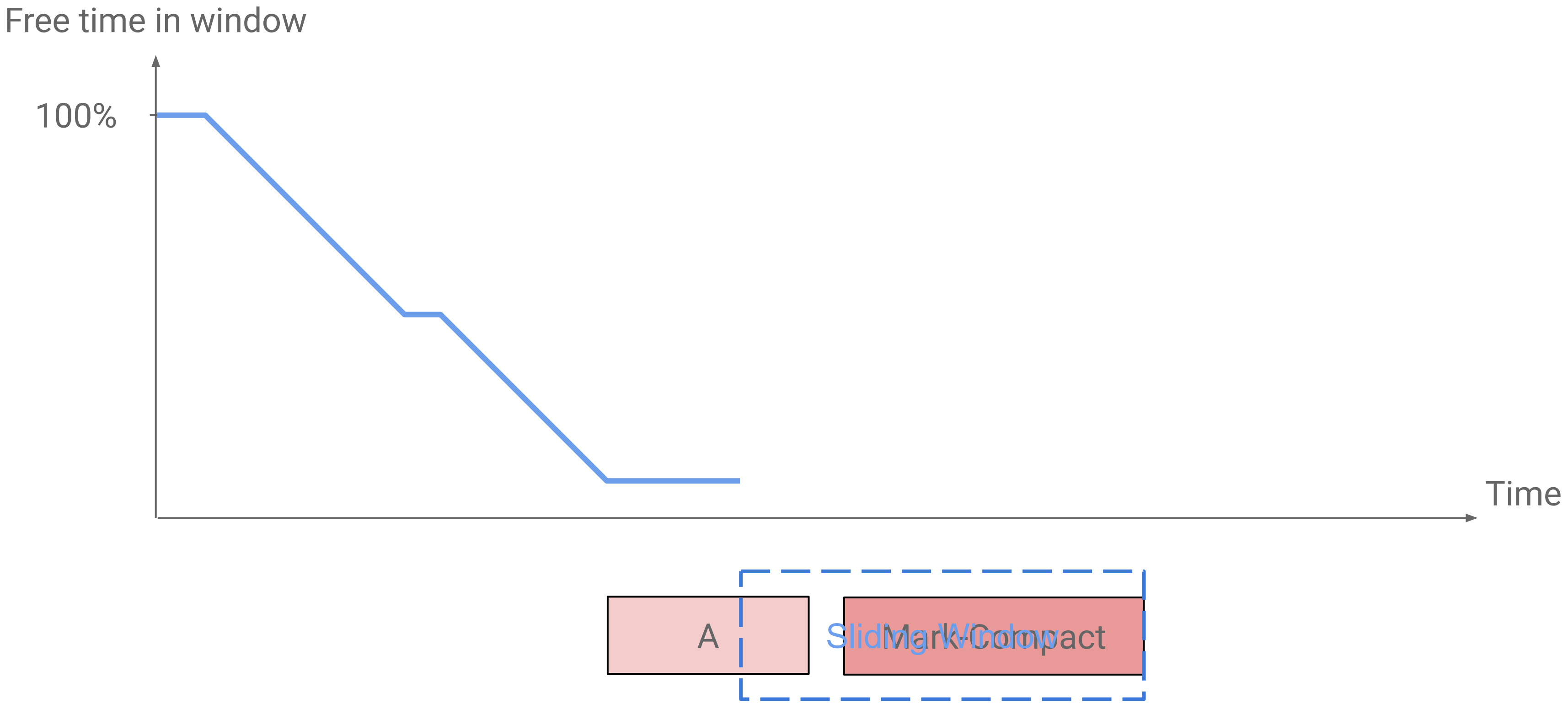
Google

Utilization function

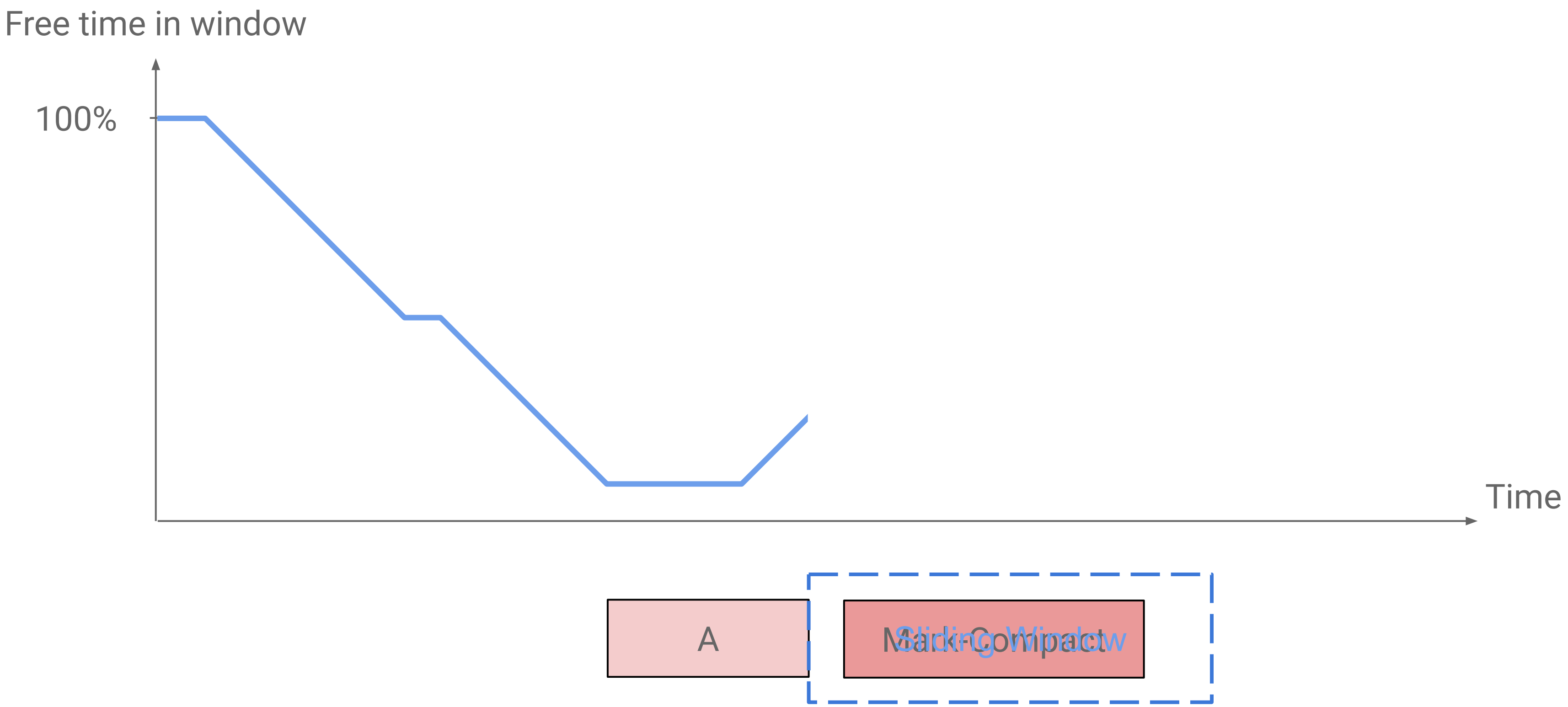


Google

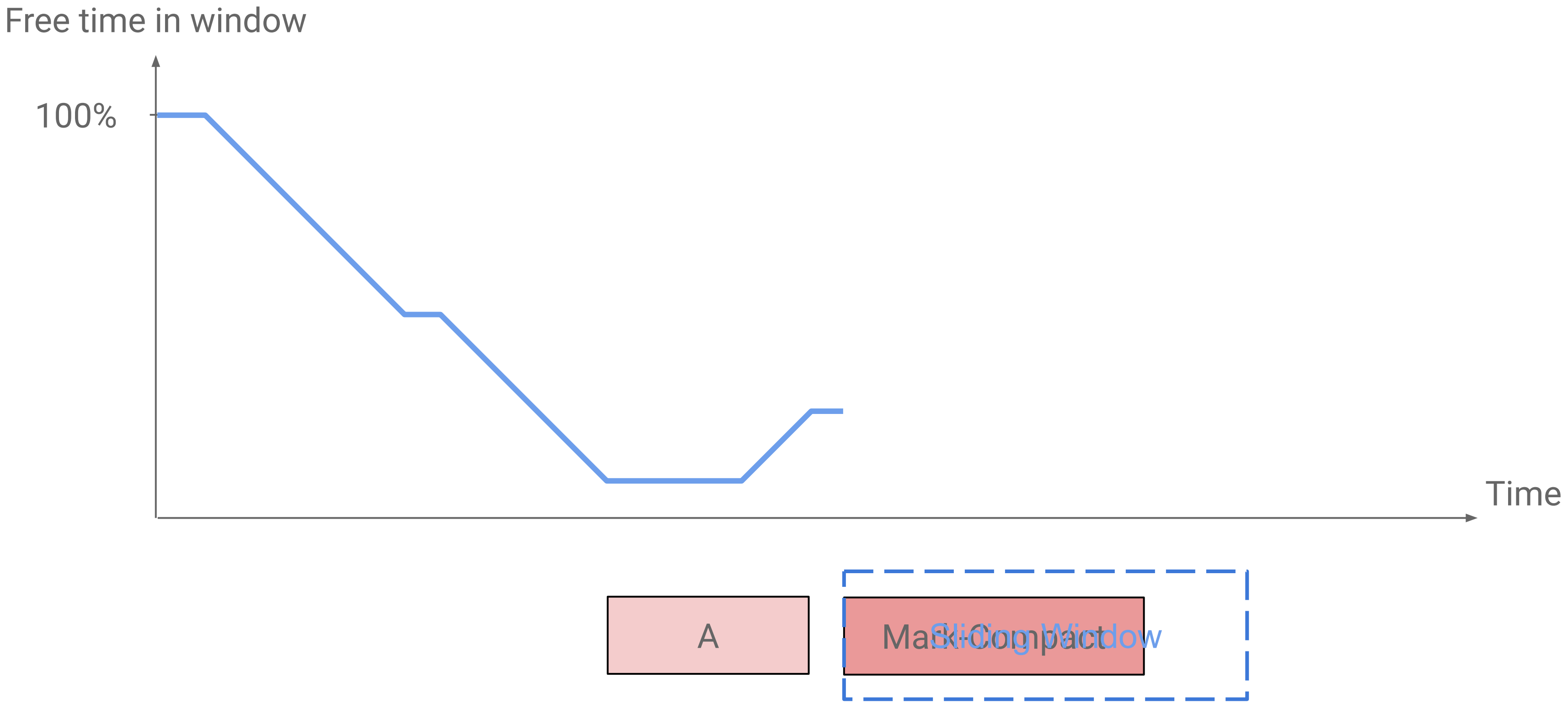
Utilization function



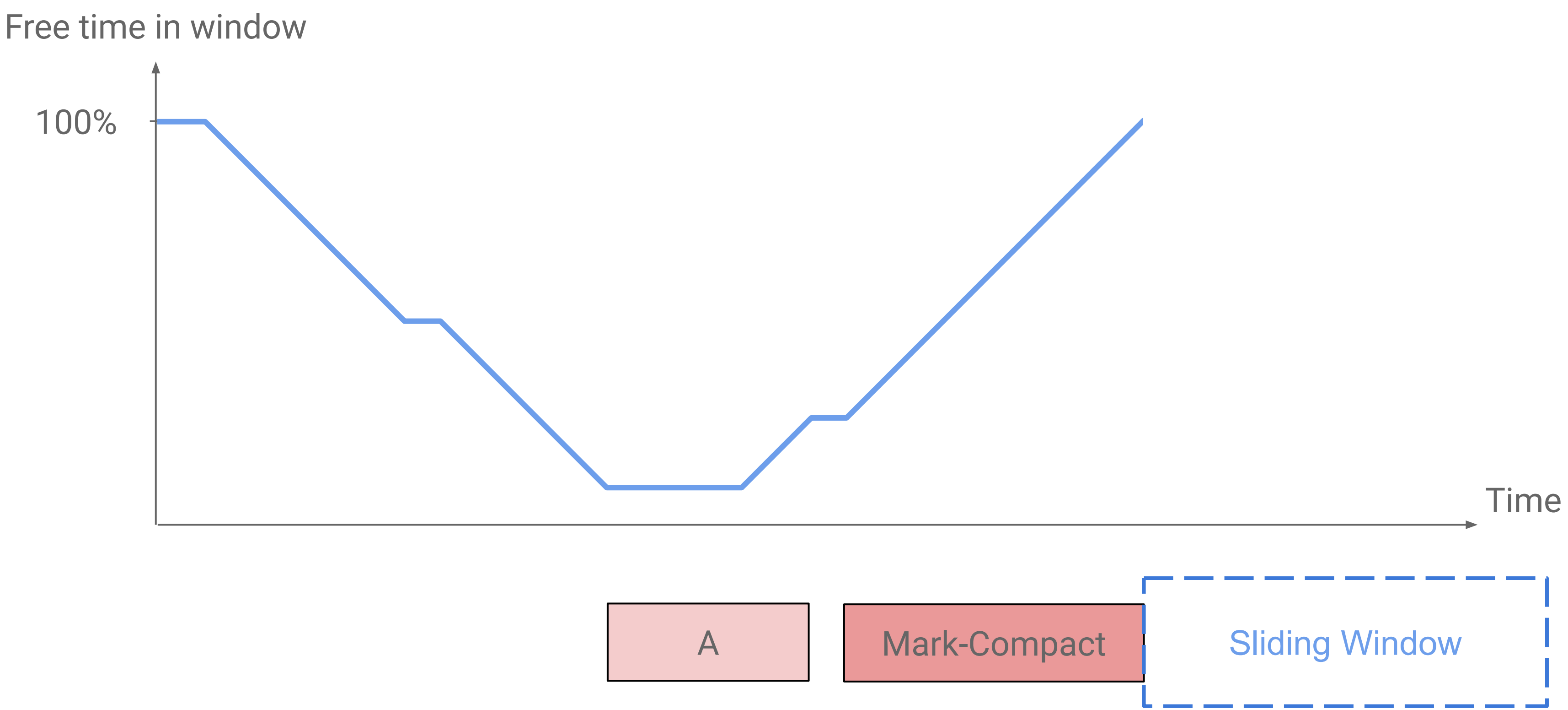
Utilization function



Utilization function

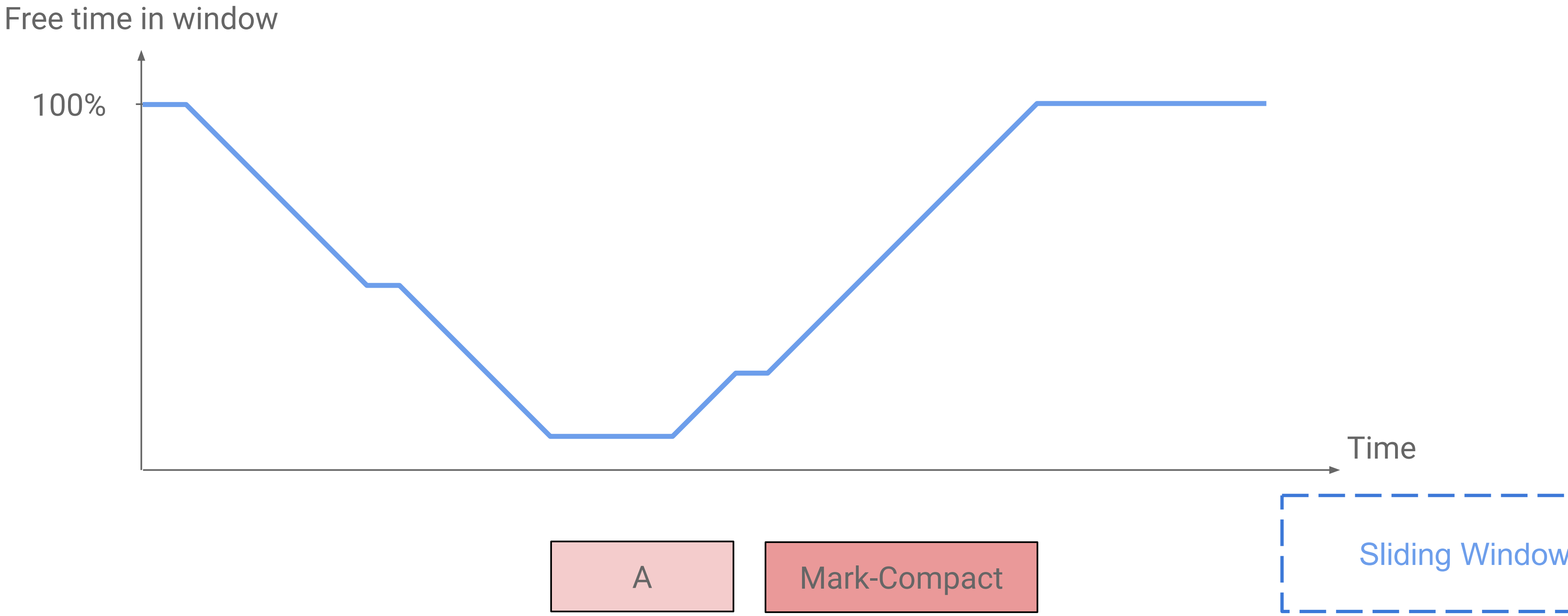


Utilization function



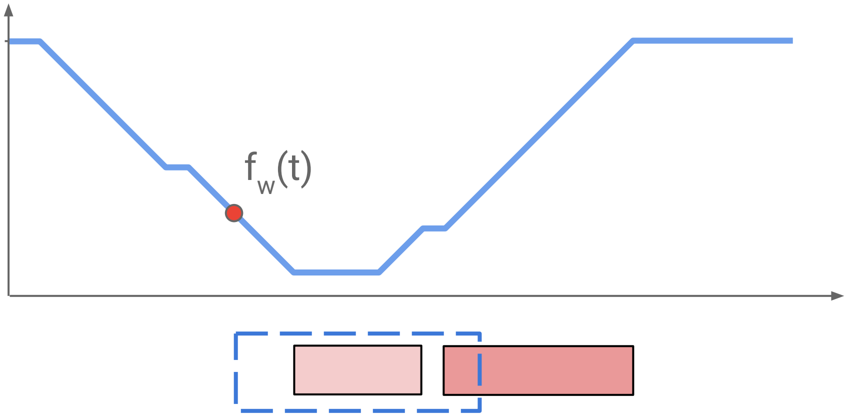
Google

Utilization function



Google

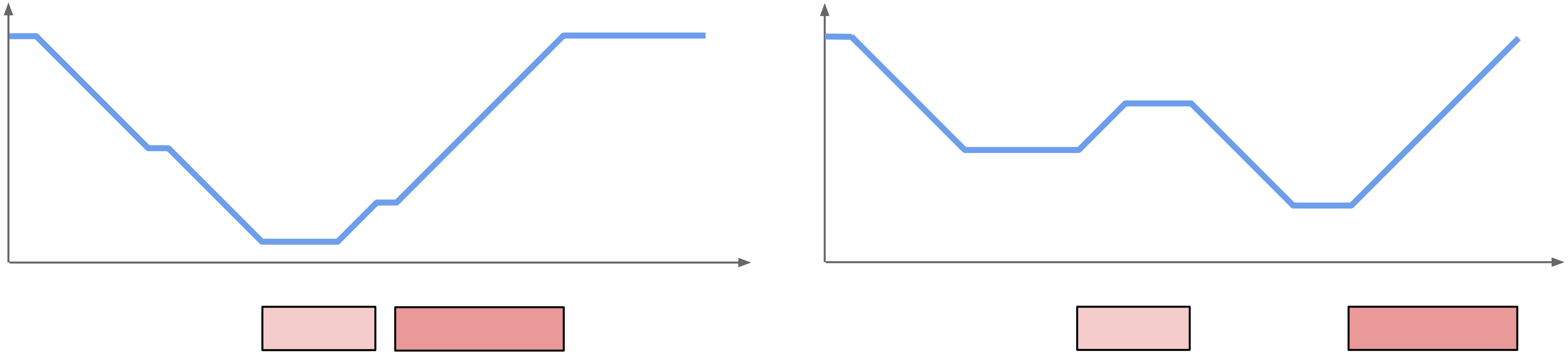
Interpretations



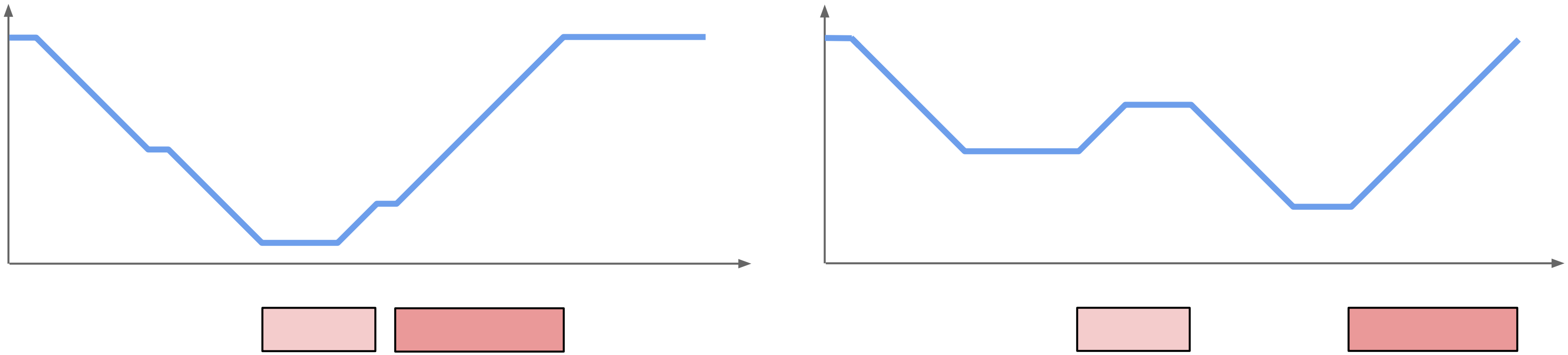
- $f_w(t)$ = fraction of time GC leaves to application in the time window [t, t+w).
- $f_w(t)$ = probability that a high priority task arriving at any moment in [t, t+w) is not queued

Google

Utilization function

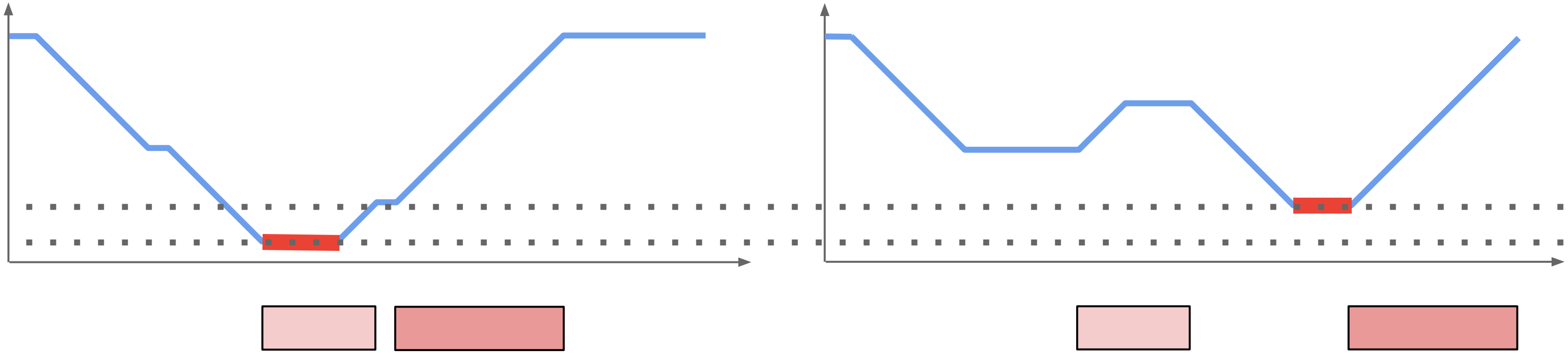


Utilization function



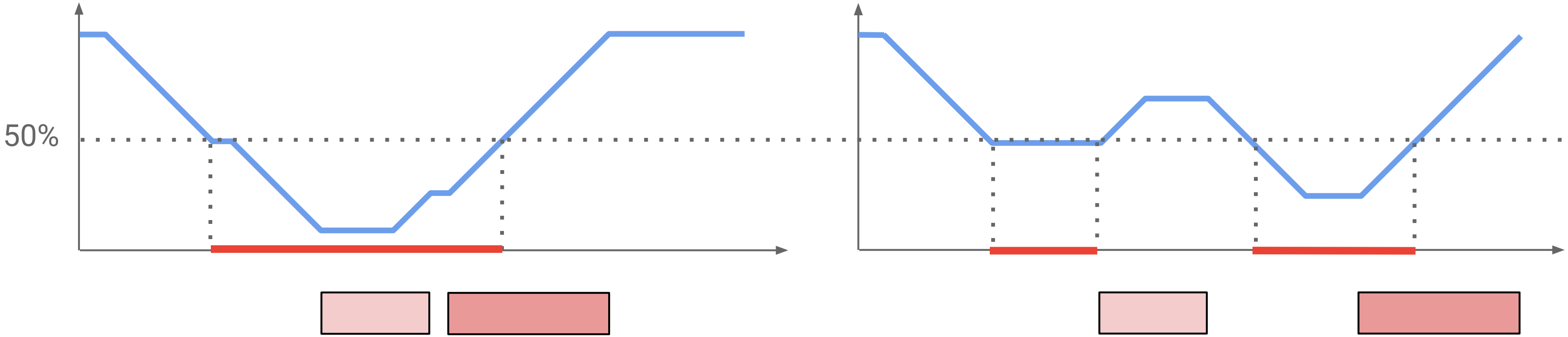
How to compare?

Minimum mutator utilization



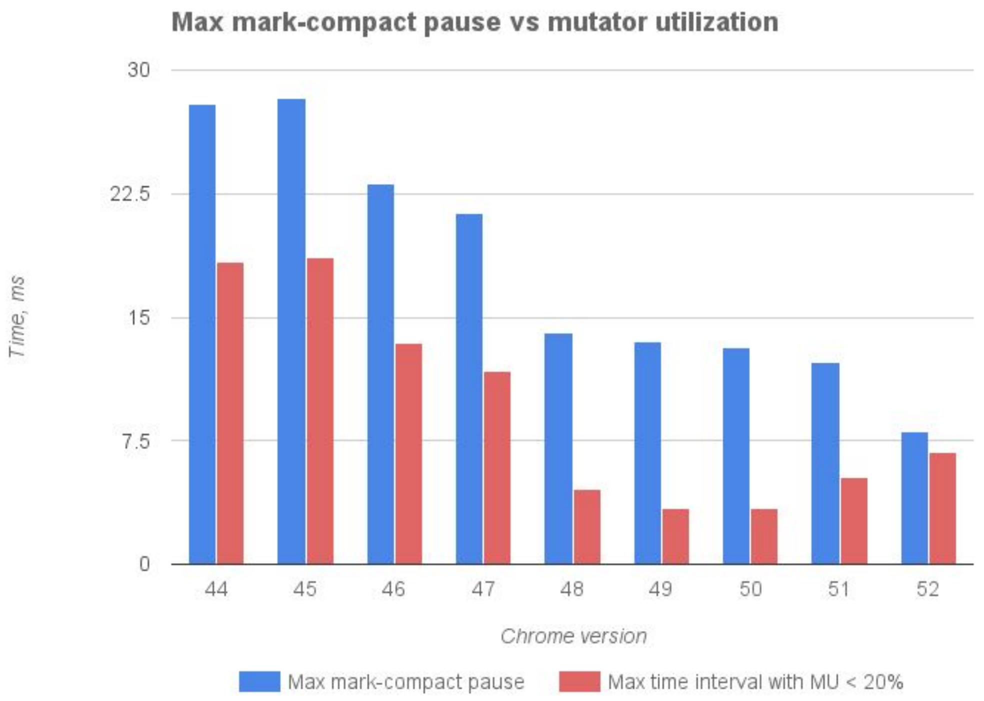
Google

Intervals with low mutator utilization



The largest interval where f(x) ≤ 50% for all points in the interval.

Evaluation



Max mark-compact pause vs mutator utilization
30
22.5
15
7.5
0
Time, ms
44
45
46
47
48
49
50
51
52
Chrome version
Max mark-compact pause
Max time interval with MU < 20%

What we learned

- Incremental development of Orinoco - good idea!
  - Stability feedback via Canary channel
  - Performance feedback via UMA and telemetry
- Pause time distribution is not the whole performance story
  - Density of GC events on the timeline is also important
  - Mutator utilization can be good indicator of latency impact

Thanks!

Google

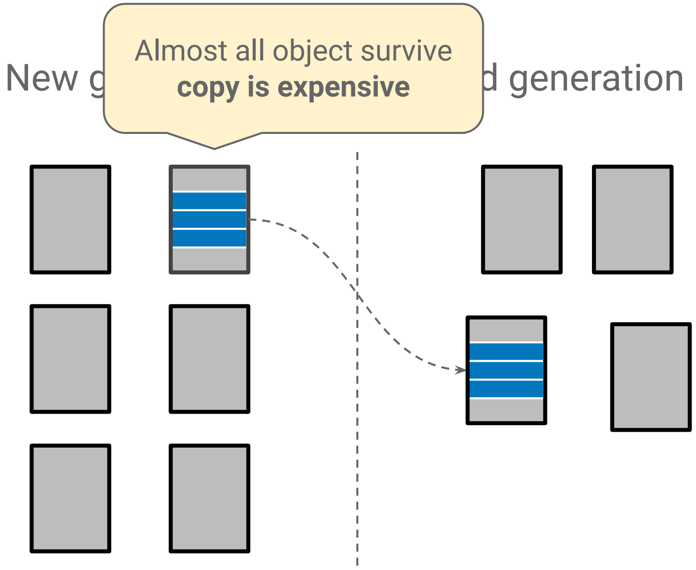
Changes on the way

Experimental

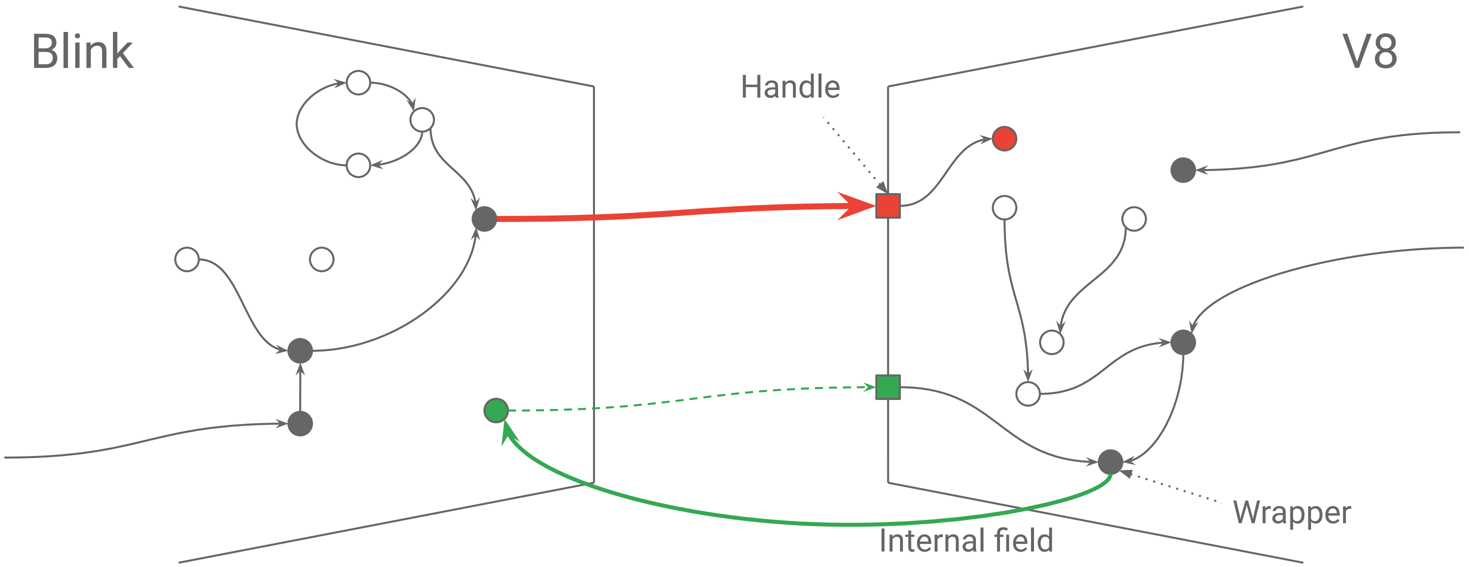
(behind flags)

Unified Heap

- Allow pages to move from young to old generation without copying



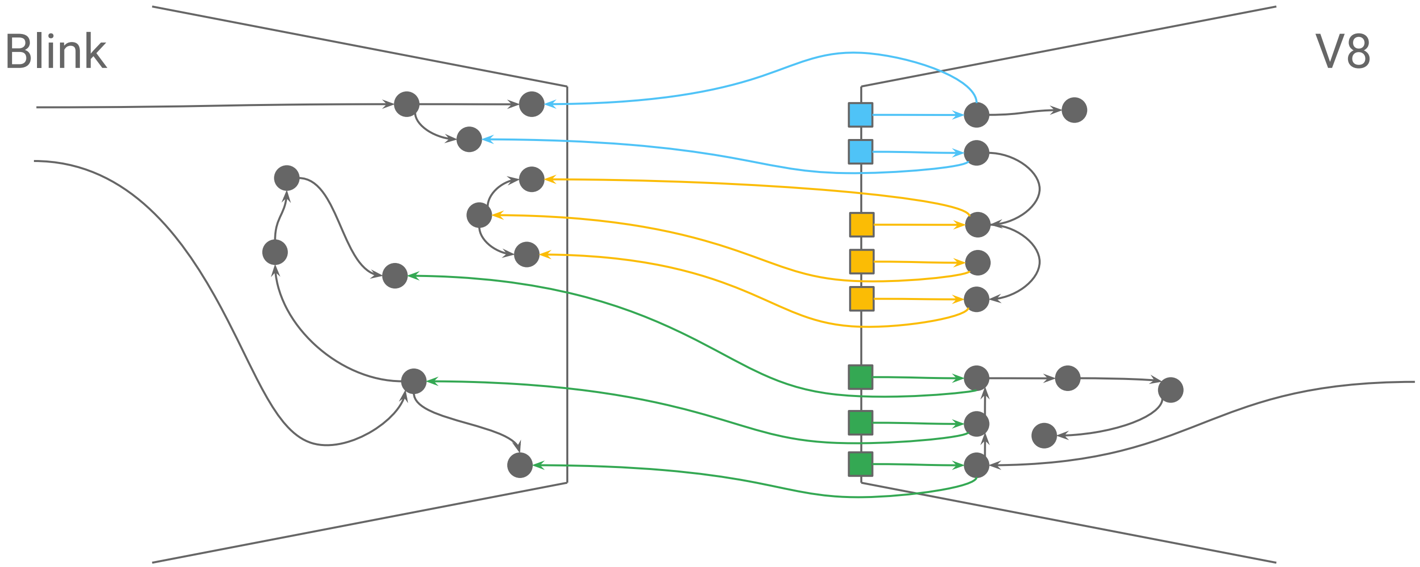
Blink ⇔ V8 - Connecting the Worlds



Handles keep V8 objects alive

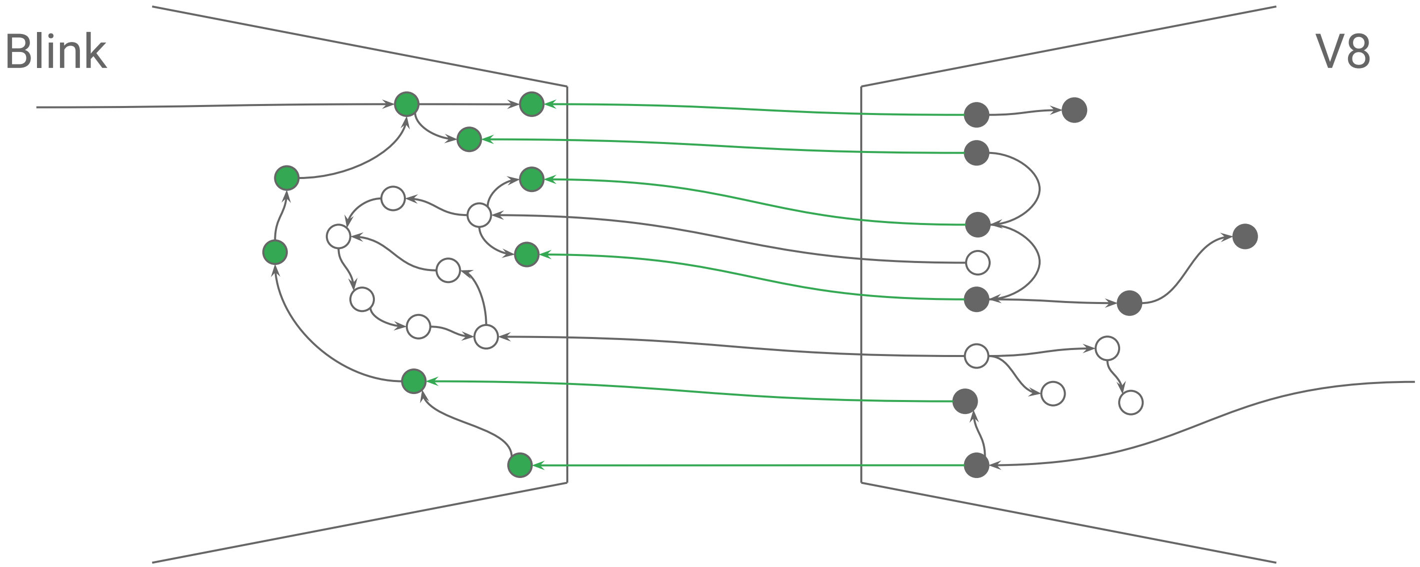
Wrapper keep Blink objects alive

Current Approach: Object Grouping



- Handles know which group they belong too (rule-based)
- Groups keep objects alive

Becoming friends with Blink: Tracing of wrappers



- Tracing through the Blink tree
- No need for handles

Google