The Blink Scheduler

London is minding your tasks

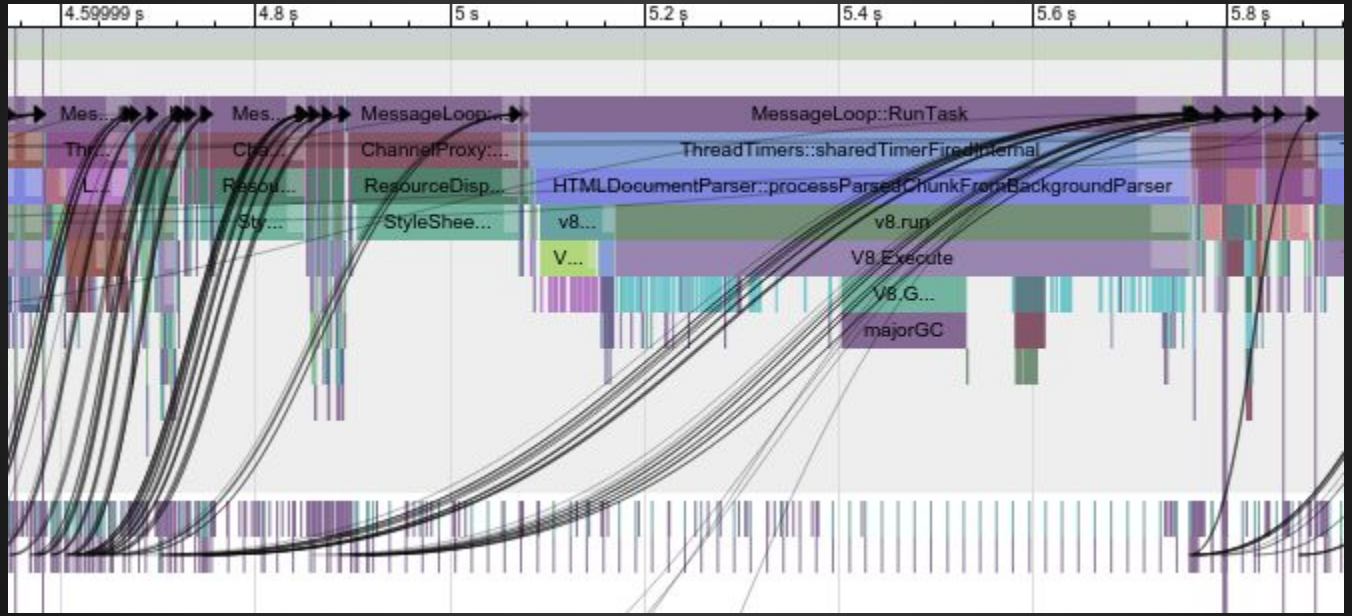
BlinkOn 3, Nov 5th 2014

{alexclarke, petrcermak, picksi, rmcilroy, skyostil}@chromium.org

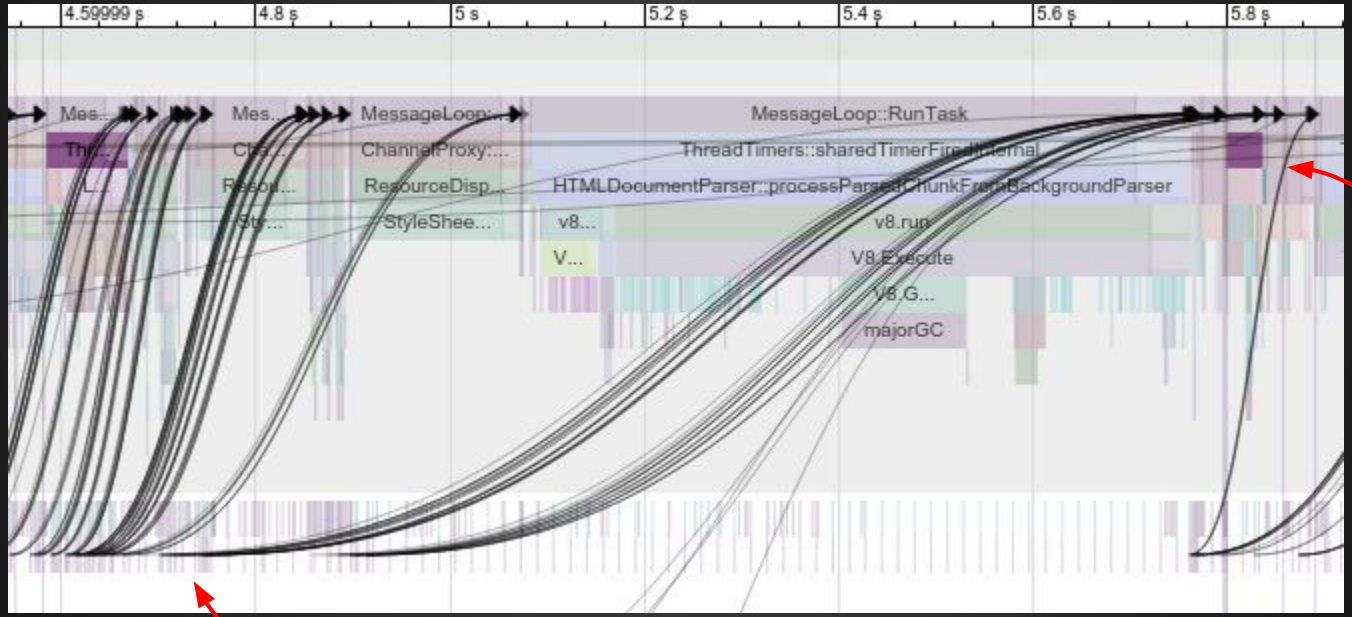
The main problem: Traffic jams

- High priority tasks (e.g. user input) can get stuck behind slow tasks.
- Delays of up to 1 second when scrolling

The main problem: Traffic jams



The main problem: Traffic jams

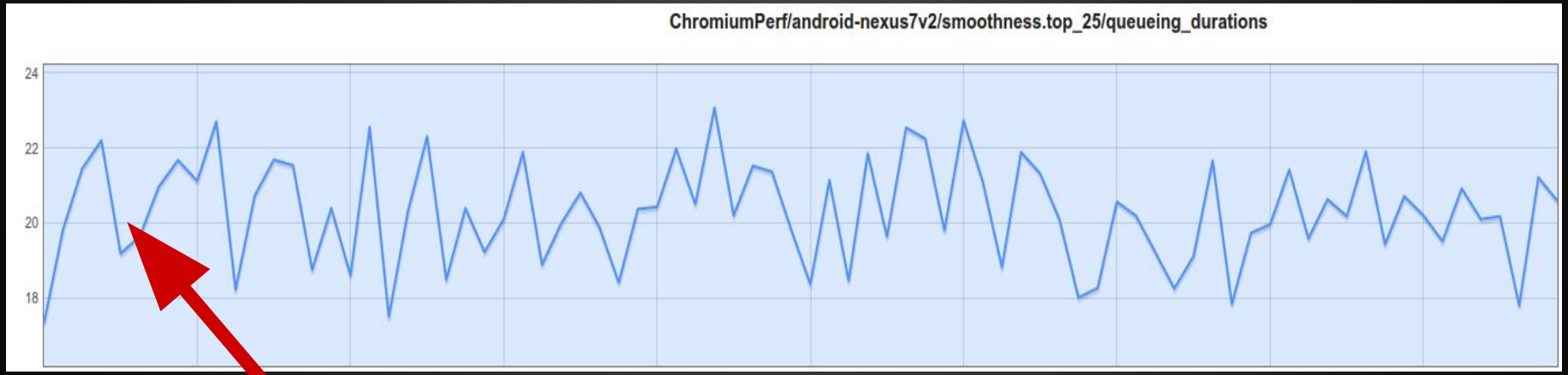


...and executed here, after 1 second

Compositor task posted here

Issue: How do we measure traffic jams?

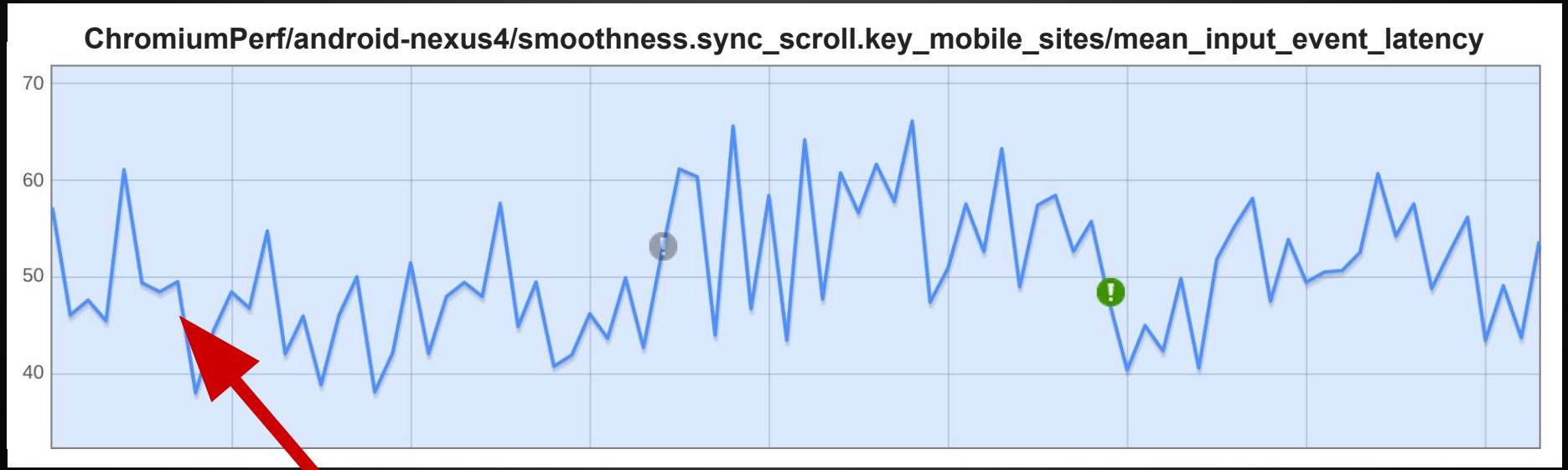
queueing_durations track how long compositor tasks are stuck in the message loop queue.



20 ms median delay

Issue: How do we measure traffic jams?

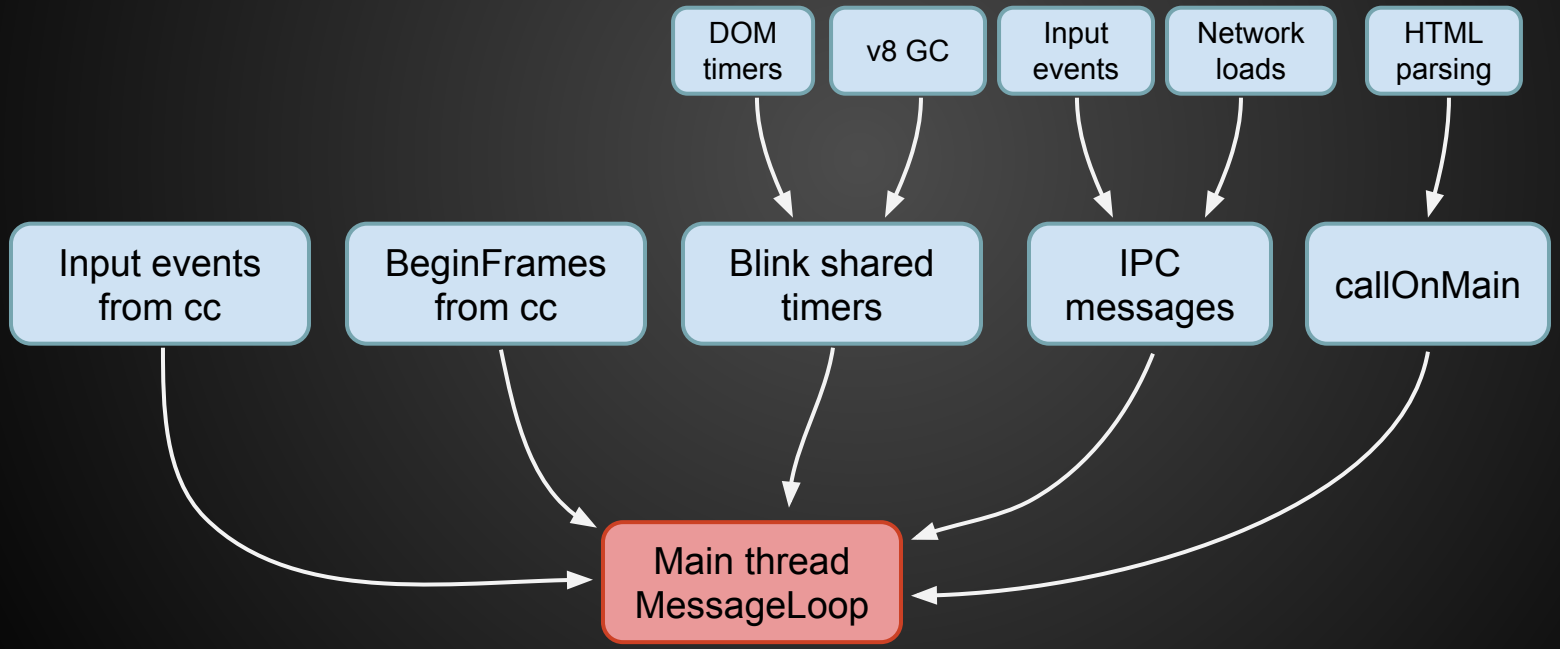
mean_input_event_latency tracks how long user input takes to hit the GPU.



50 ms median delay!

Task sources

Renderer main thread tasks come from many sources:

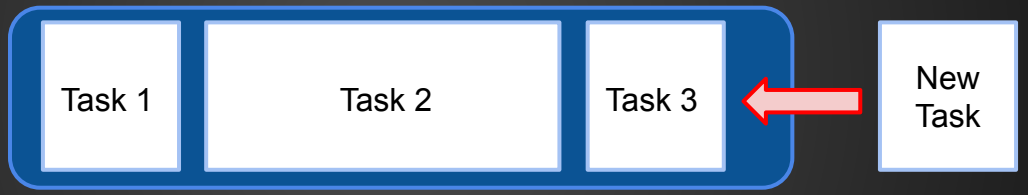


Goal: Perfect Scheduling

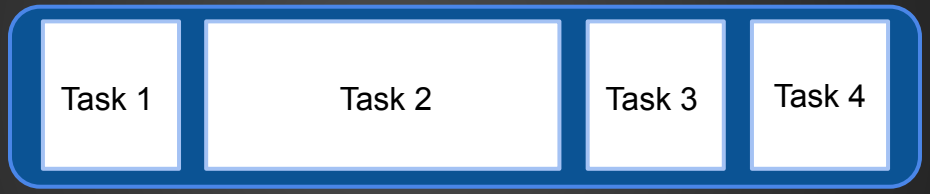
- *Perfect scheduling* makes sure the right things get done at the right time.
- We're doing the same stuff, just in a sensible order.
- Enable well-written content to take advantage of good scheduling.

All about scheduling

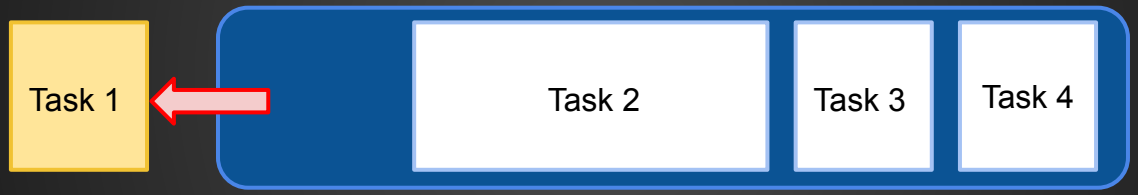
Pre-Scheduler world



Pre-Scheduler world

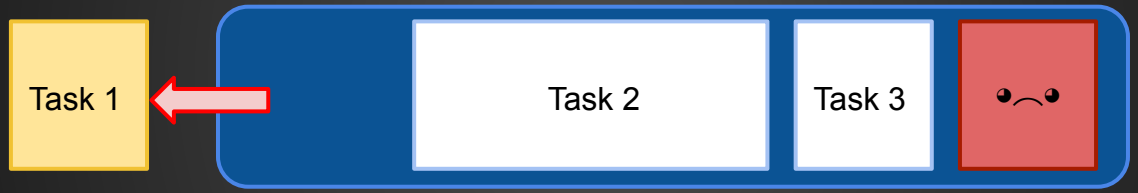


Pre-Scheduler world



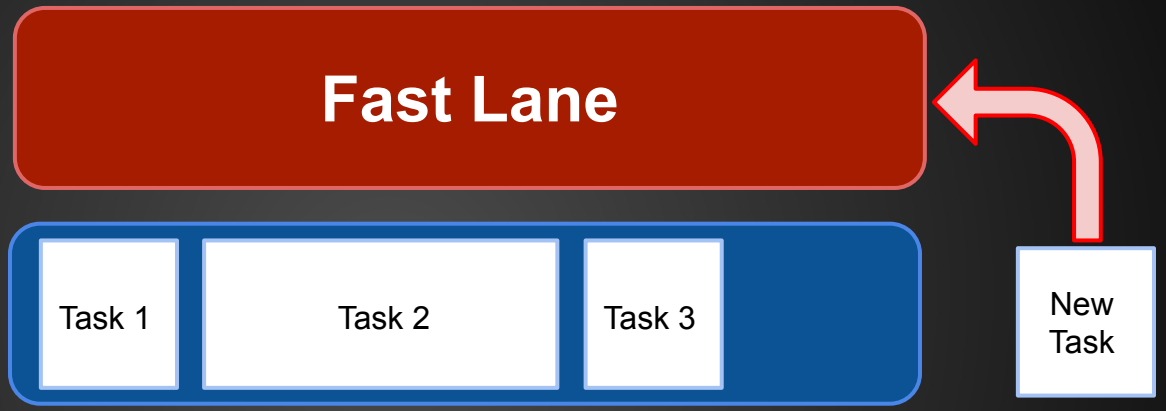
Task 1 taken from the front of the queue and executed.

Pre-Scheduler world



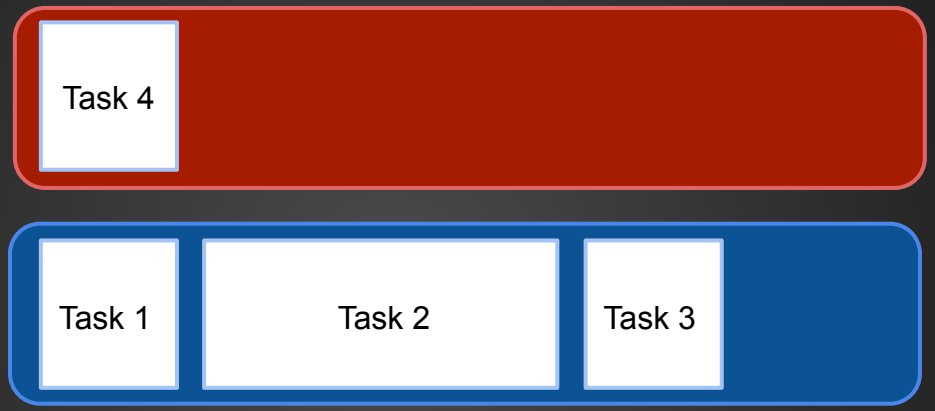
A new task will need to wait until all the preceding tasks have been executed.

Scheduler world

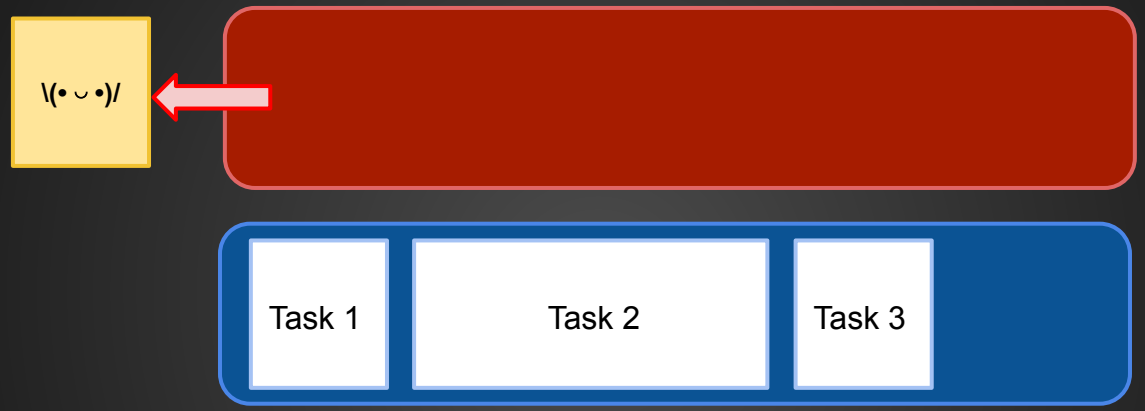


Ideally we'd like to be able to fast-track high priority tasks.

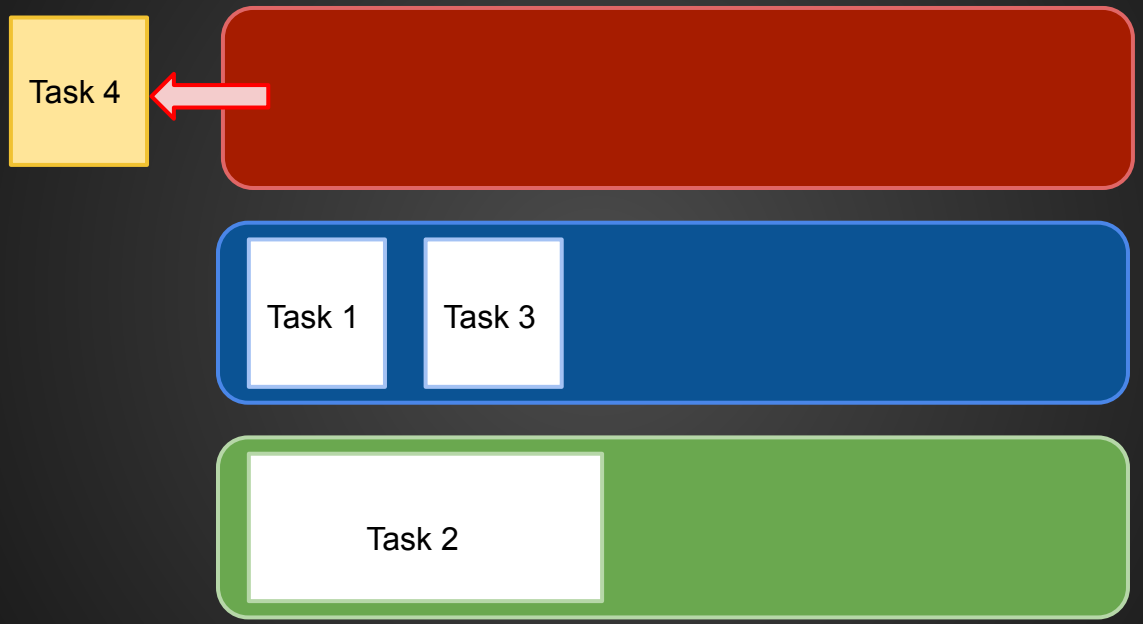
Scheduler world



Scheduler world



Multiple queues give more flexibility



Issue: Maintaining execution order

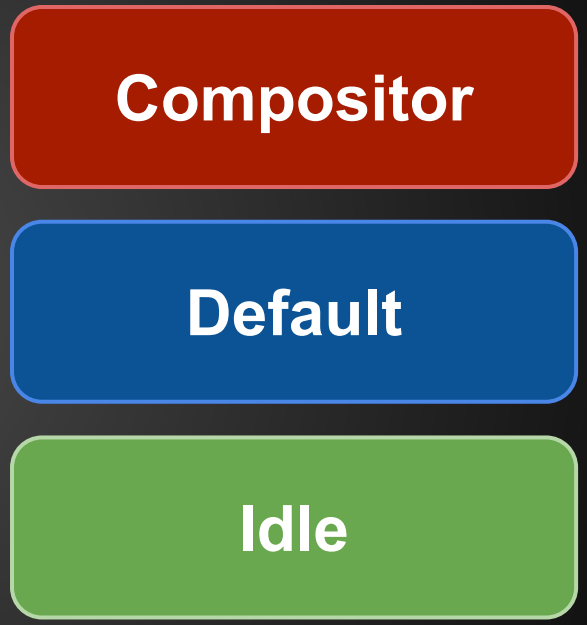
Most tasks need to have their relative execution order maintained.

- Can’t composite the results of input processing before the input has been processed
- Ordering assumptions baked into existing code

Only reorder tasks that have opted into prioritization.

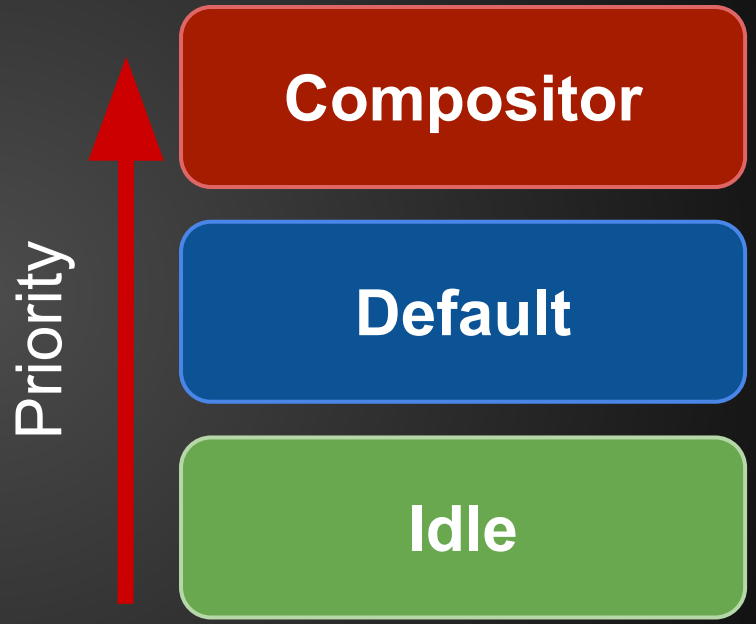
Solution: Tasks are grouped by type

- Maintains relative execution order
- Provides a clear API for posting tasks



Solution: Tasks are grouped by type

Always prioritise compositor?



Issue: Static prioritisation doesn’t work

Always prioritising compositor tasks regressed page loading time by 14%.

Solution: Dynamic prioritisation

Use contextual awareness to determine prioritisation *policy*.

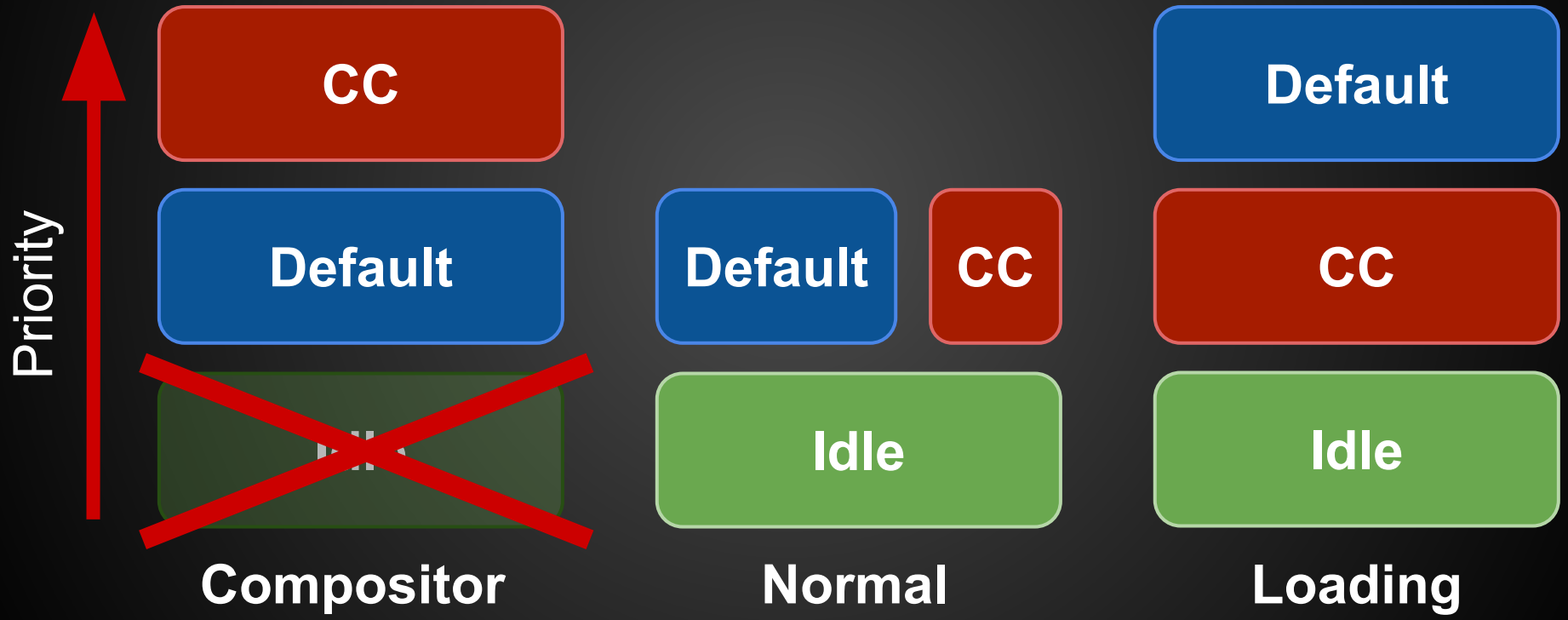
- User interaction makes:
  - **Compositor tasks** *more* important
  - **Resource loading** *less* important
- When finished drawing a frame, allow background tasks:
  - pro-active **GC**
  - background **HTML parsing**

Policy selection

Events will change the active policy:

- User input events → Prioritize Compositor
- Compositor committed frame → Idle
- Compositor begin frame → Normal
- Page navigation → Page Load

Queue priorities set by policy

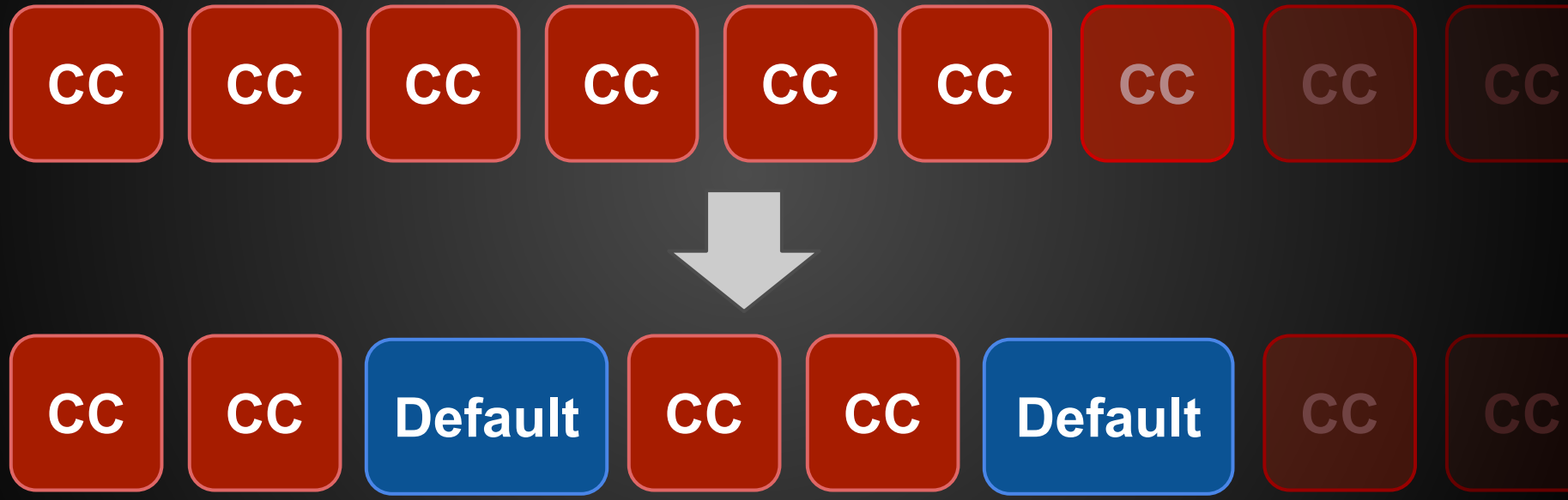


Issue: Task starvation

We need a scheduler which cannot be ‘gamed’ by sociopathic tasks to completely starve out other tasks.

Issue: Task starvation

Solution: Weighted round robin to avoid starvation



Architecture

Finding a home for the Scheduler

Version 1 implementation was in Blink

- Re-implemented much of the message loop
- Posting tasks required a lot of Blink-Chromium plumbing
- Task execution required Russian doll closures

Finding a home for the Scheduler

Version 2 implementation was in base/message_loop

- Required low level changes in base
- Changing mission critical but crufty code
- Changes would be inherited by all message loops

Finding a home for the Scheduler

Version 3 (current) lives in content/renderer/scheduler/

- Sits on top of base::MessageLoop
- Lets us use more powerful primitives from base
- Avoids crossing layers unnecessarily

Scheduler architecture overview

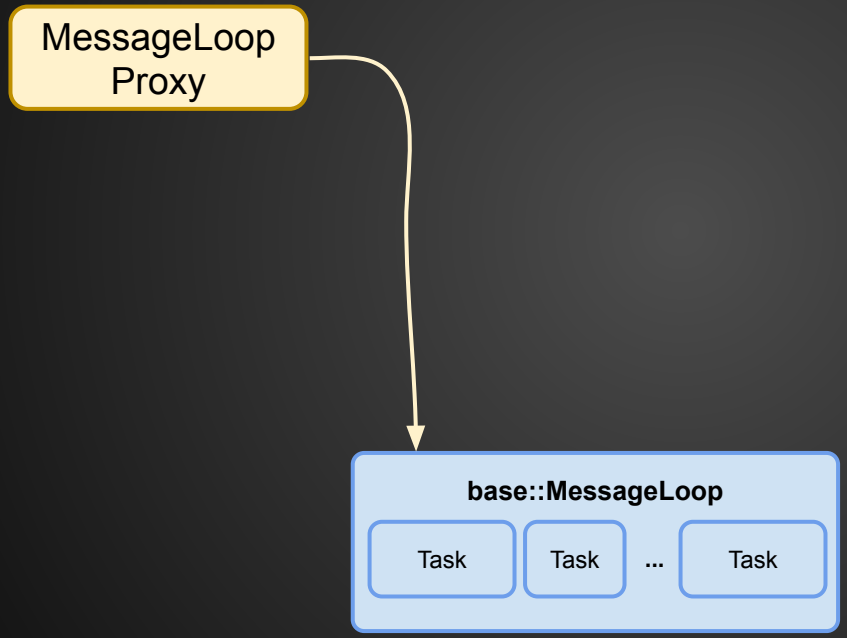
- Scheduler provides N task queues
- Policies determine which task queues are prioritised
- System behaviour determines policies

Scheduler architecture overview

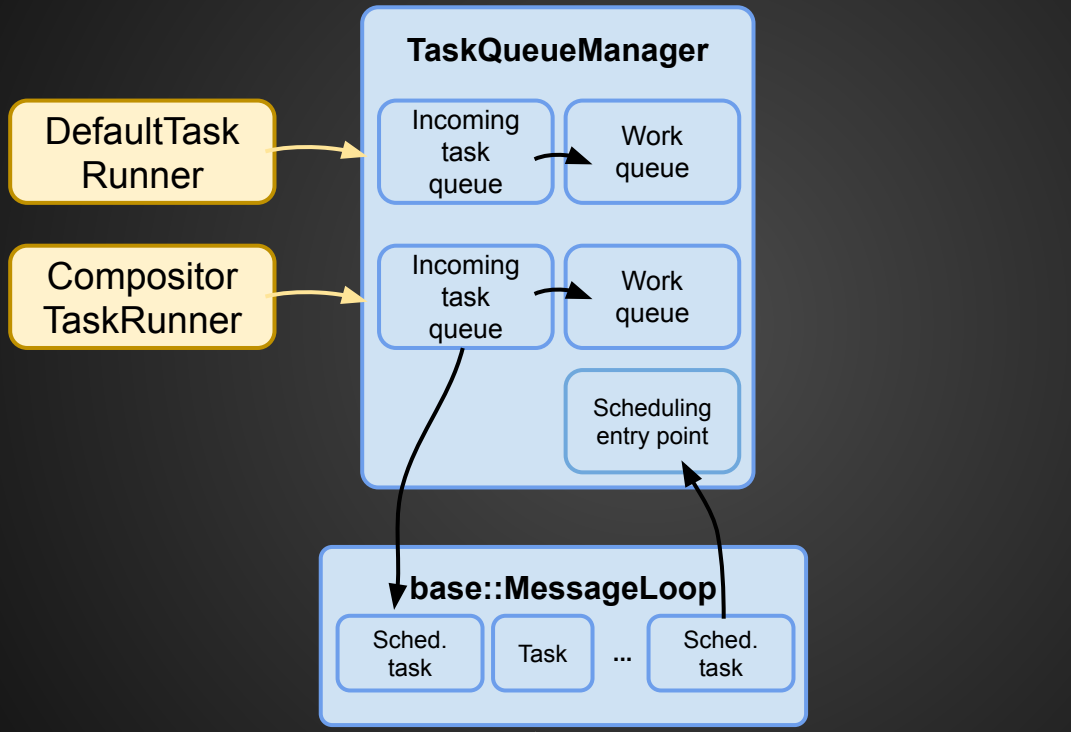
Three key elements

1. API for posting to different queues
2. Interface that selects the next queue to process
3. Mechanism for changing policies

Architecture

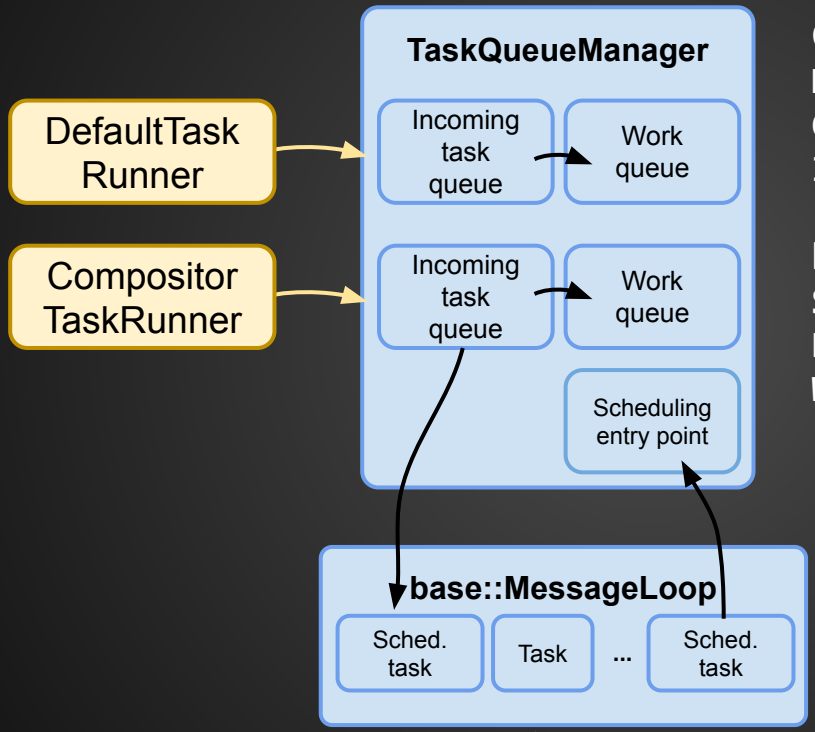


Architecture: Posting API



Other tasks that aren't scheduled (e.g. IPC)

Architecture: Posting API



Other tasks that aren't scheduled (e.g. IPC)

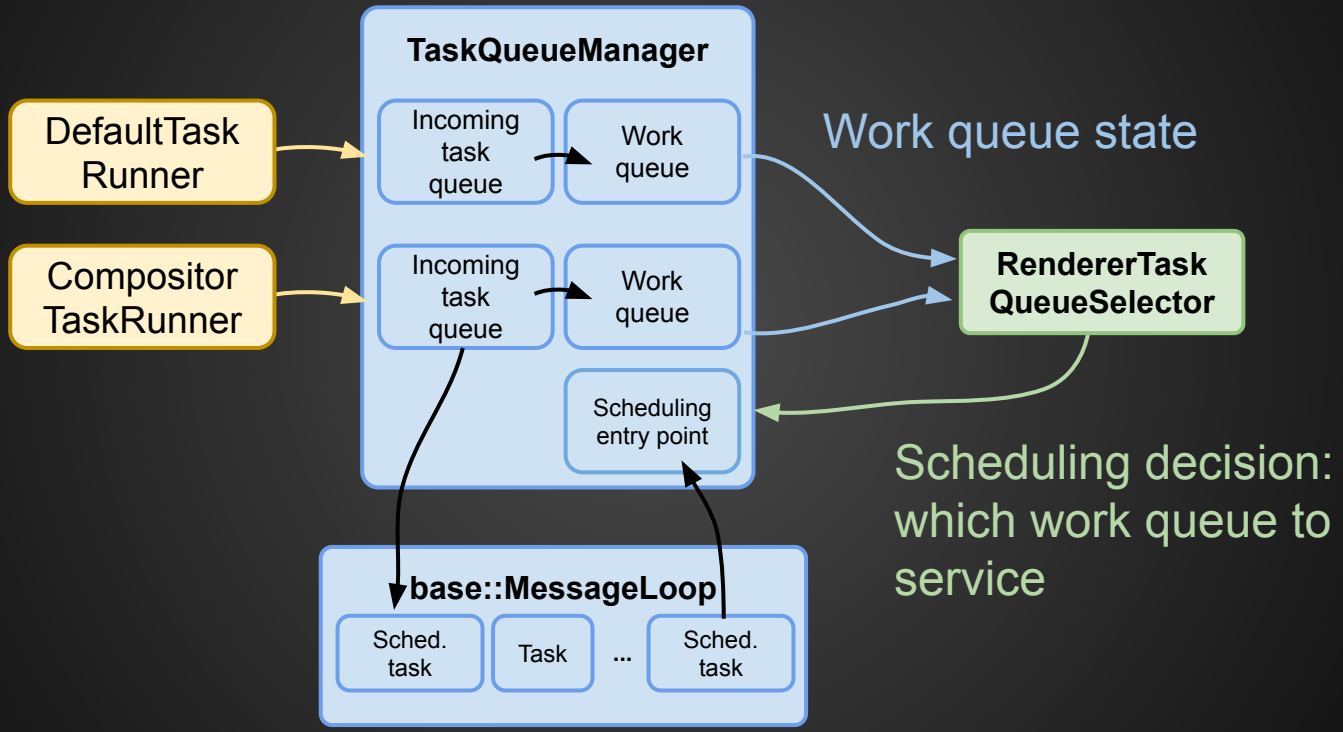
Chromium:

```
DefaultTaskRunner()->PostTask()
CompositorTaskRunner()->PostTask()
IdleTaskRunner()->PostIdleTask()
```

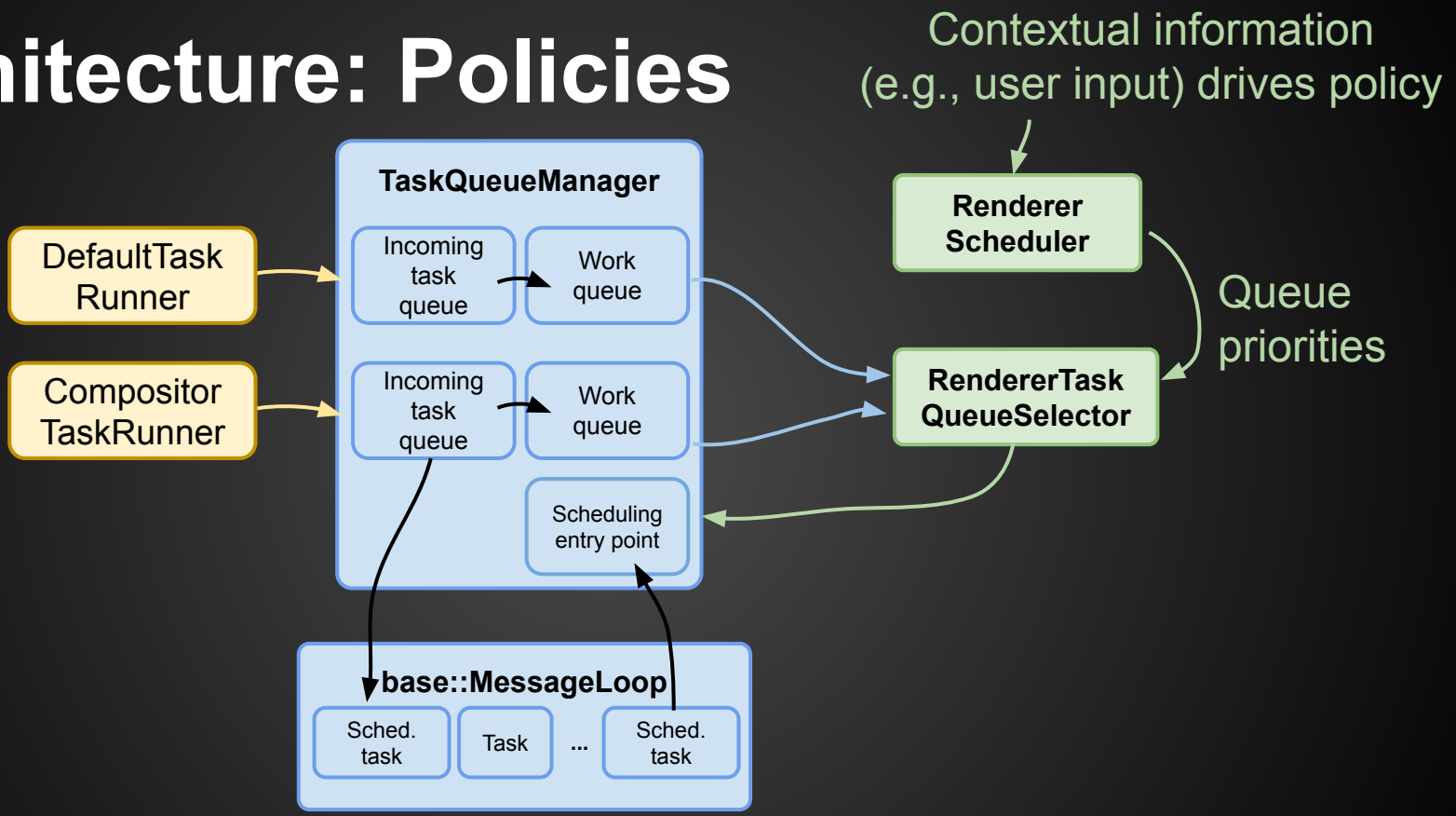
Blink:

```
Scheduler::shared()->scheduler()
Platform::current()->callOnMain()
WebThread::postTask()
```

Architecture: Queue selector



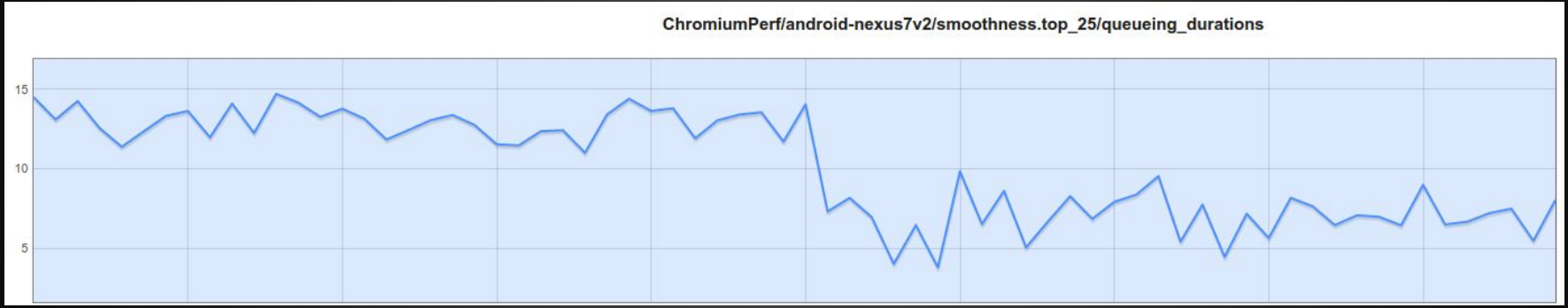
Architecture: Policies



Results

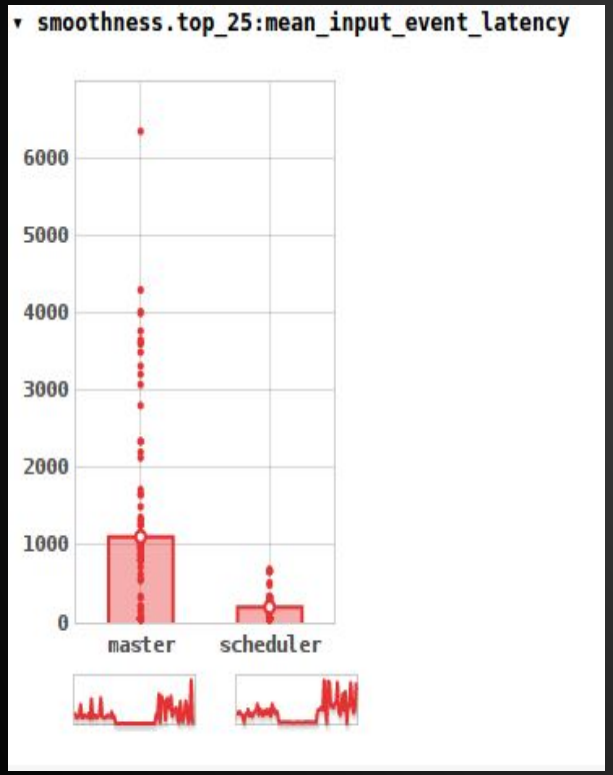
Results from Scheduler V1

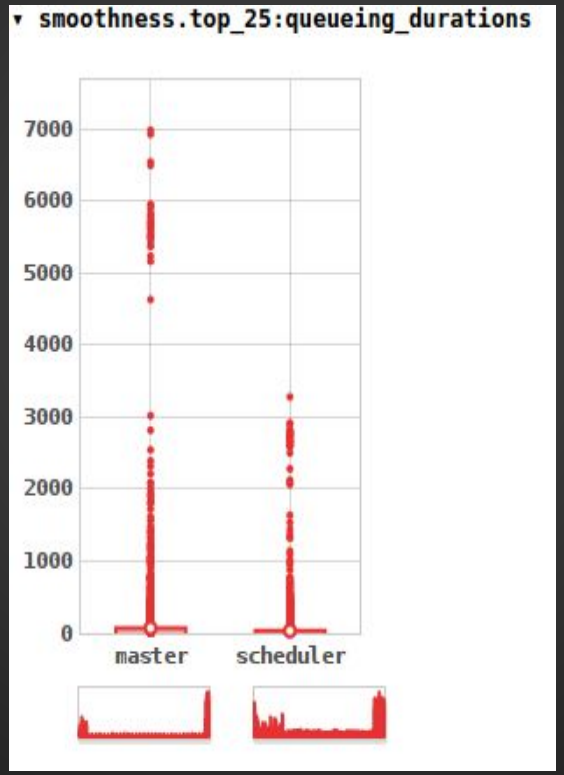
(Nexus 7v2)

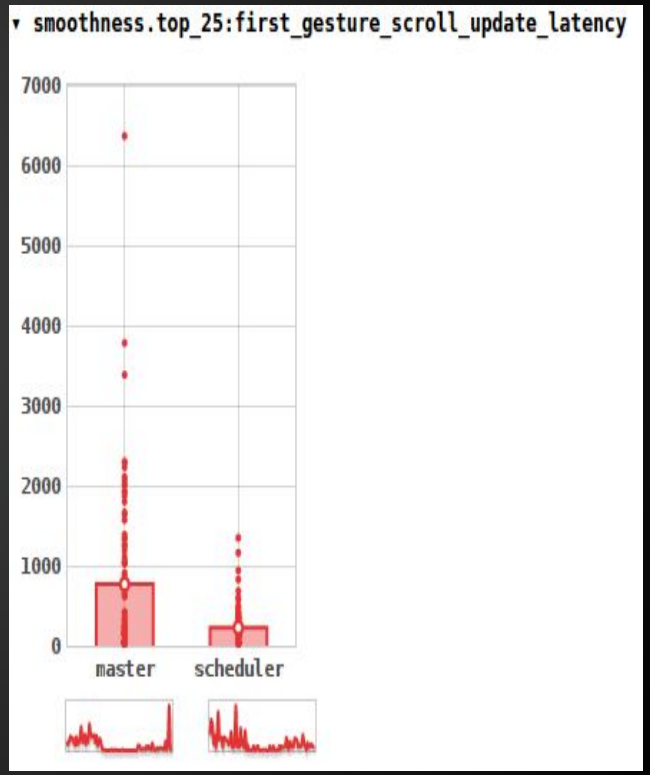


Queueing_durations ~12ms => ~7ms

Preliminary results for V3 (Nexus 7v2)



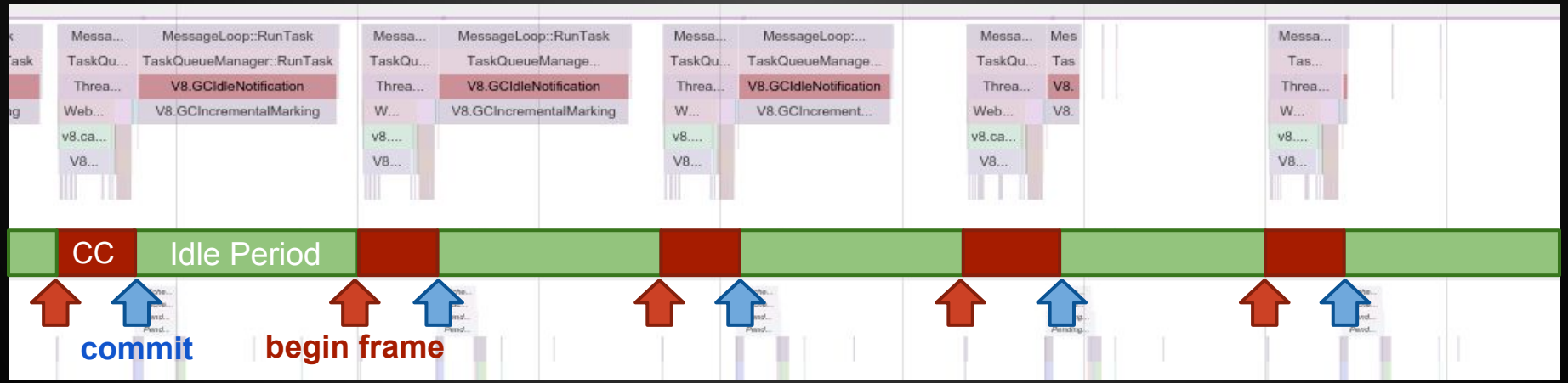




Future steps

Future steps: Idle tasks

- Get non-critical work off the critical path
- Only executed when frame has been committed



Idle task characteristics

- May be starved for arbitrarily long
- Take a deadline argument
  - Should finish by the deadline or earlier
- Posted tasks won’t run until next idle period

First idle task customer: V8 GC

- Do GC work off critical path
- Idle task performs GC incremental marking step
- Estimates how much heap it can mark before deadline
- Repost task if more to do
- Early result: 50% reduction in GC on the critical path

Future Idle task customers…?

- V8 GC sweeping
- V8 optimising compiler
- HTML parsing
- Oilpan incremental marking

Future steps: LoadingTaskQueue

- Adding a queue for page loading tasks
  - Resource loading
  - HTML parsing
  - Javascript compilation
- Add a loading priority policy
  - Prioritises loading tasks to optimise first paint ‘above the fold’
- Deprioritise when in compositor priority policy

Future steps: Half-baked ideas

- Provide scheduling primitives to JS
  - setTimeout(0) should mean setTimeout(0)
  - Posting idle tasks
- Power optimisation
  - Synchronize idle/background ticking in renderers with other wakeups
- Feedback for developers
  - Point out scheduling problems

Future steps: Half-baked ideas

- Provide scheduling primitives to JS
  - setTimeout(0) should mean setTimeout(0)
  - Posting idle tasks
- Power optimisation
  - Synchronize idle/background ticking in renderers with other wakeups
- Feedback for developers
  - Point out scheduling problems

Lessons learned

- How (not) to write a scheduler
  - Existing code unintentionally relies on implicit ordering
  - It’s easy to break low level things in subtle ways
- Failing fast and iterating reaches good solutions
- Live and die by metrics
- Having MTV contacts (eseidel@) was invaluable

Making your tasks scheduler aware

- Avoid blocking the main thread >8ms
- Check `shouldYieldForHighPriorityWork()` from long running tasks
- Post to the most appropriate queue (or add a new one?)
- Do background work using idle tasks
- Join scheduler-dev@chromium.org to get involved.