Perf Insights, RAIL & Happiness

https://goo.gl/n2l4Fd

10 Nov 2015

nduca@chromium.org


DOM
LAYOUT
60 FPS
CDN
FIRST PAINT
JANK
DNS
LOOKUP
LAYOUT
THRASHING
SLOW
SUNSPIDER
JAVASCRIPT
GARBAGE COLLECTION
CRITICAL PATH
INLINE CSS

rail score plz

RESPONSE
100ms

ANIMATION
10ms

IDLE
50ms

LOAD
1000ms

how do we measure RAIL?


fetching

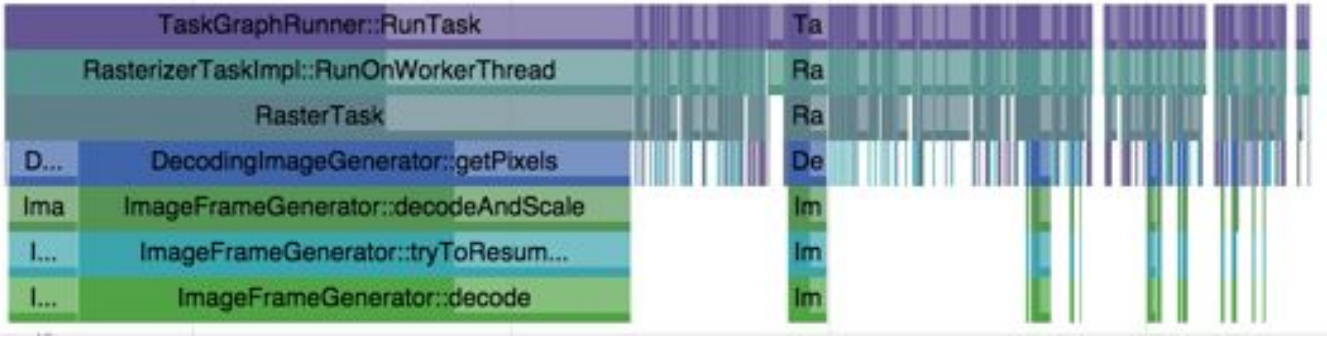
"Just"....

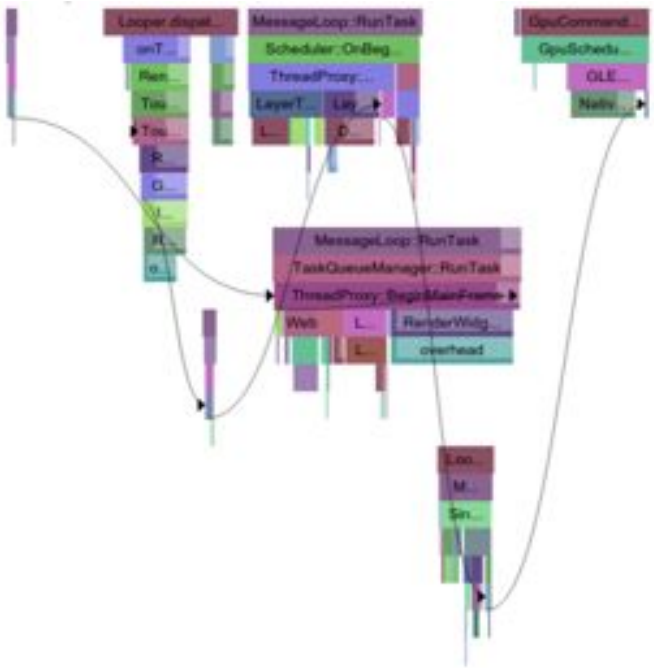
On CrRenderer main, track which stage we're in...

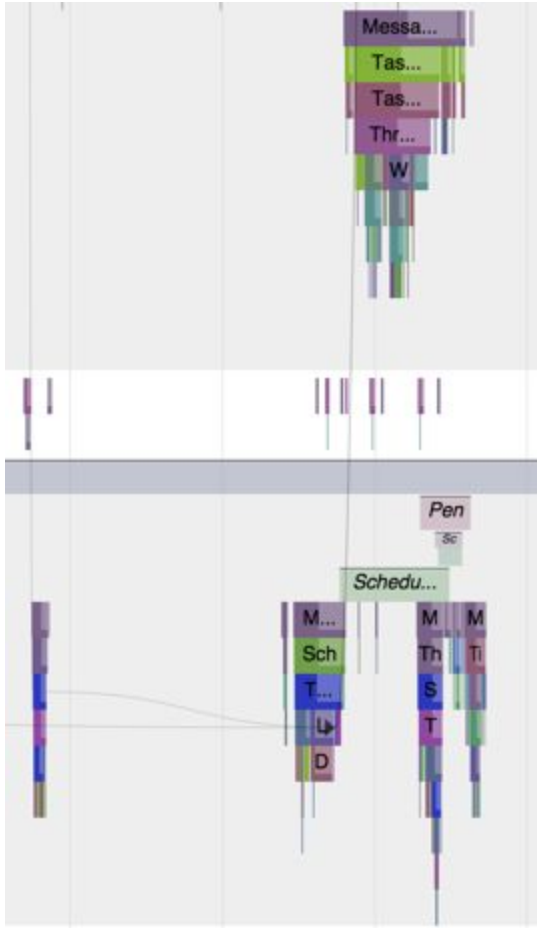
Time when the stage begins, when it ends

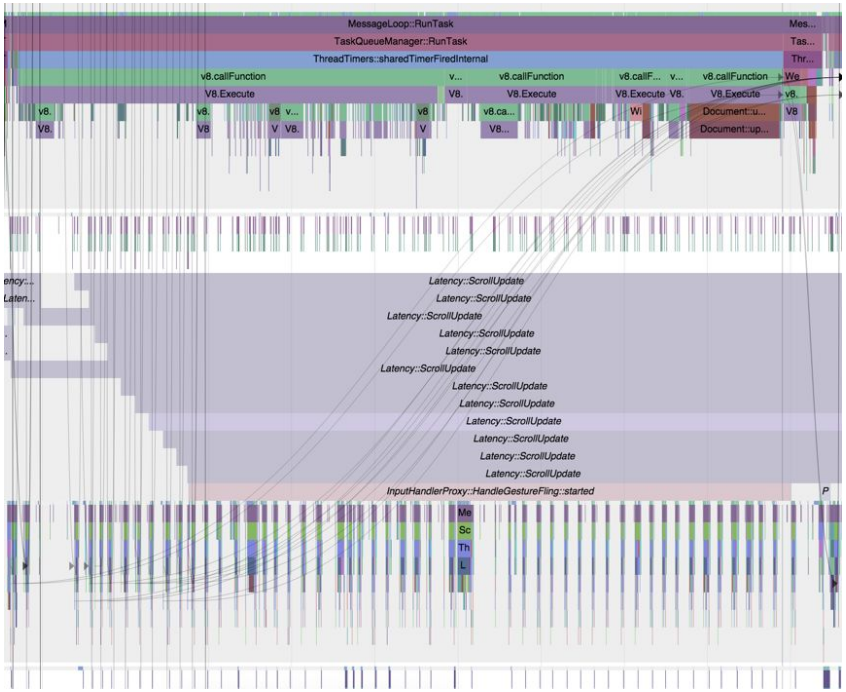
If it exceeds the goal, then you've had a violation

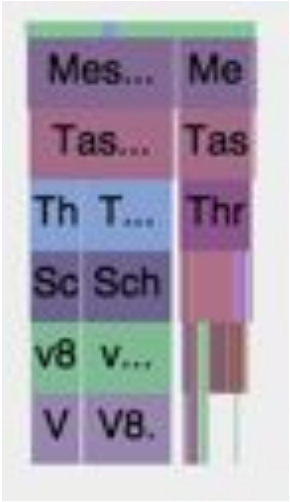
Track the # of violations, or the amount


TaskGraphRunner::RunTask
RasterizerTaskImpl::RunOnWorkerThread
RasterTask
D...
DecodingImageGenerator::getPixels
Ima
ImageFrameGenerator::decodeAndScale
I...
ImageFrameGenerator::tryToResum...
I...
ImageFrameGenerator::decode
Ta
Ra
Ra
De
Im
Im
Im


Looper.dispat...
MessageLoop::RunTask
GpuCommand...
Scheduler::OnBeg...
GpuScheds...
ThreadProxy:...
GLE...
LayerT...
Lay...
Nativ...
MessageLoop::RunTask
TaskQueueManager::RunTask
ThreadProxy::BeginMainFrame
Web
L...
RenderWidg...
overhead
Loo...
M...
$n...


Messa...
Tas...
Tas...
Thr...
W
Pen
Sc
Schedu...
M...
Sch
T...
D
M
Th
S
T
M
Ti


MessageLoop::RunTask
TaskQueueManager::RunTask
ThreadTimers::sharedTimerFiredInternal
v8.callFunction
V8.Execute
v8.callFunction
V8.Execute
v8.callFunction
V8.Execute
Document::u...
Document::up...
Latency::ScrollUpdate
InputHandlerProxy::HandleGestureFling::started


Mes...
Me
Tas...
Tas
Th
T...
Thr
Sc
Sch
v8
v...
V
V8.

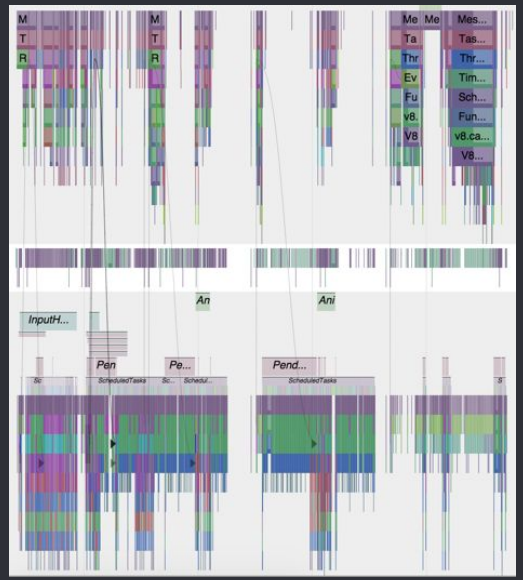

Me
Ta
Thr

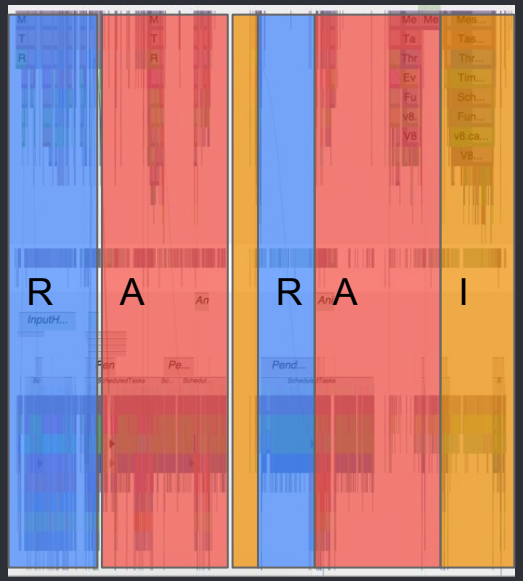
actually measuring RAIL

RAILScoring

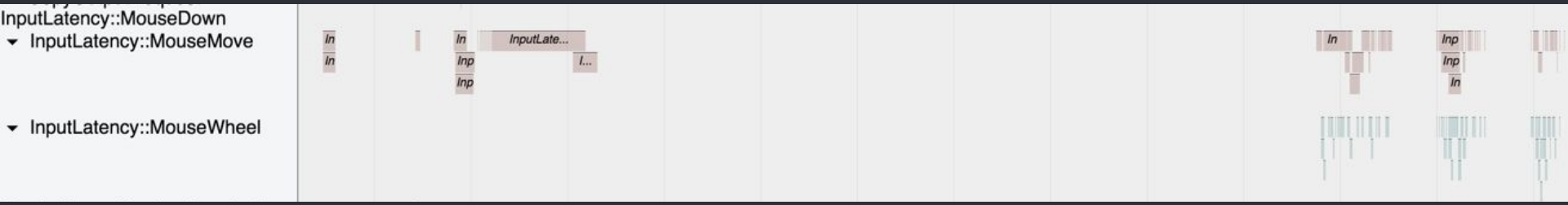
IR: **I**nteraction **R**ecord

~= RAIL Stage ... naming is hard.



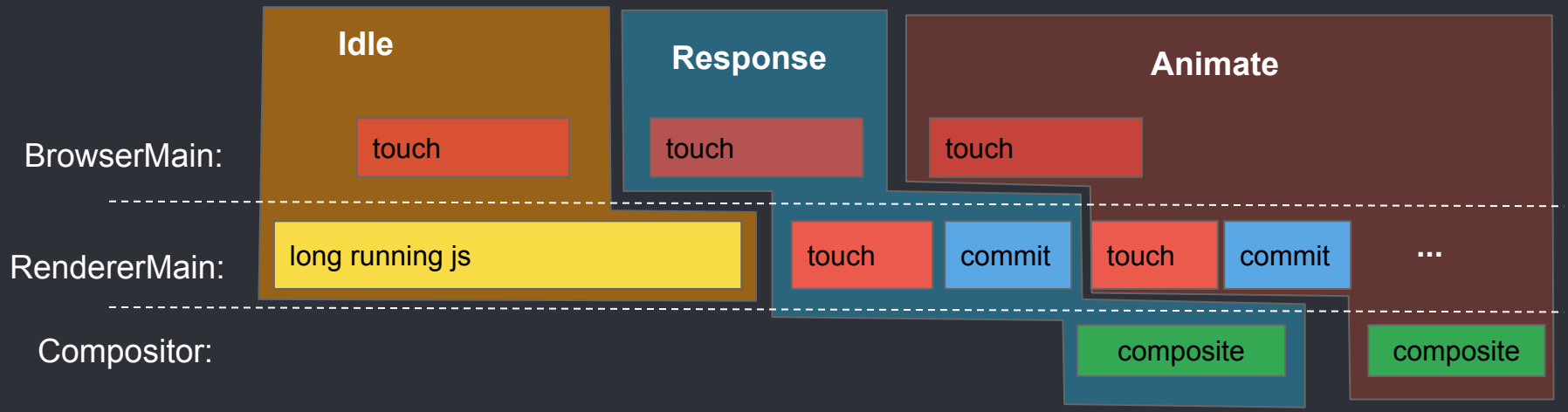


IR finding reconstructs user expectations based on things like input, loading state, window state:

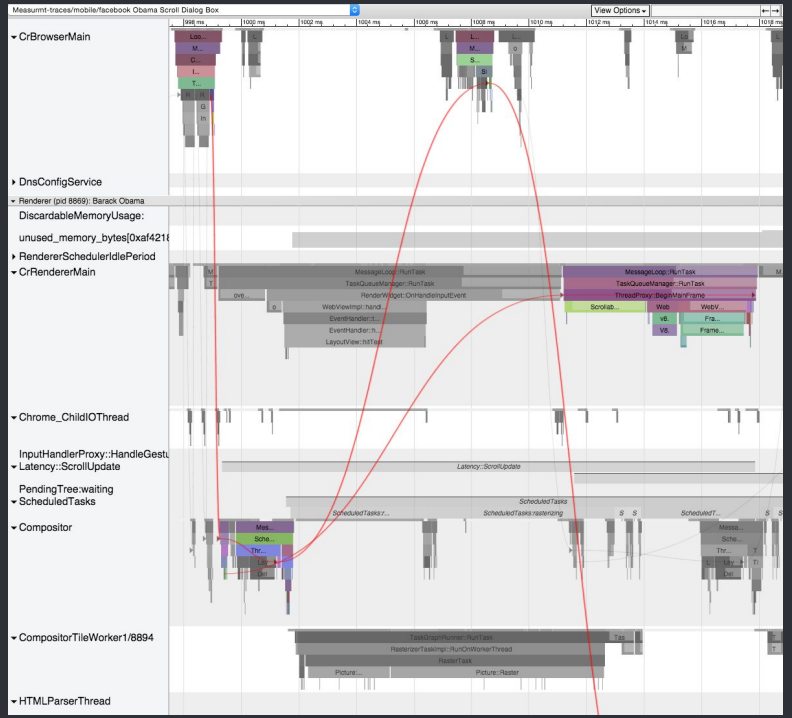


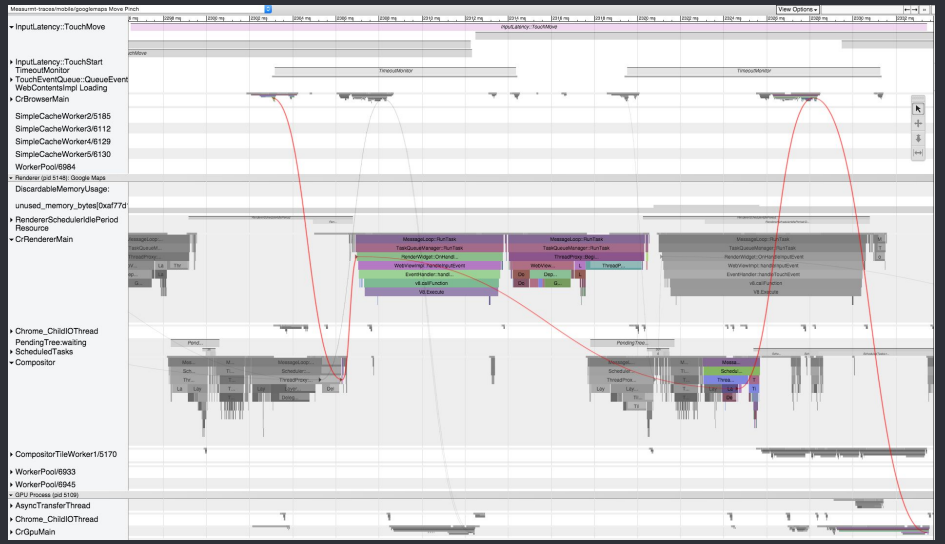
Final result is:

Threading means that IRs aren't nice pretty squares:



So we reconstruct what work is associated with an IR, using flow events and other tricks:

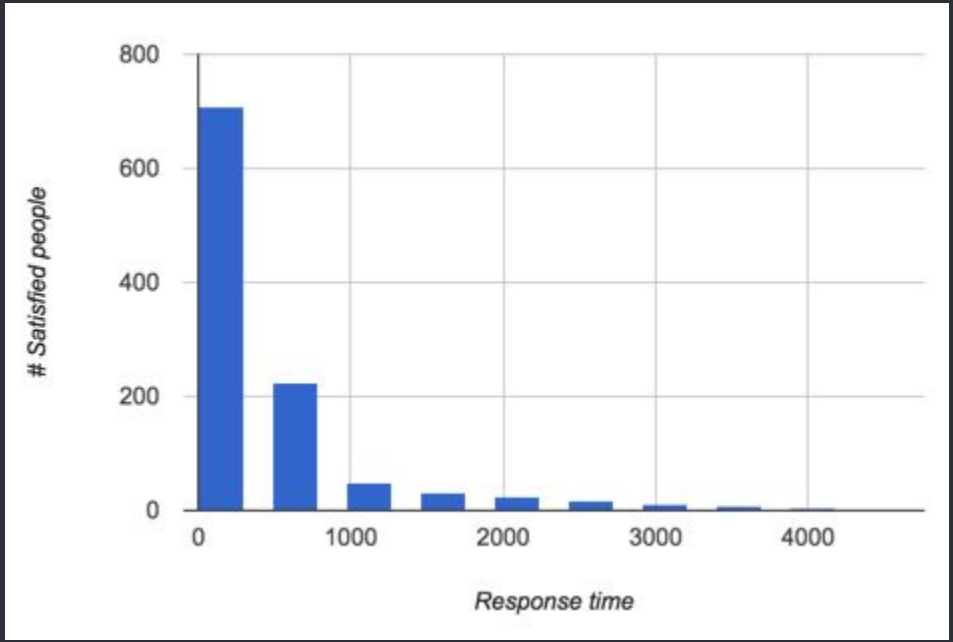




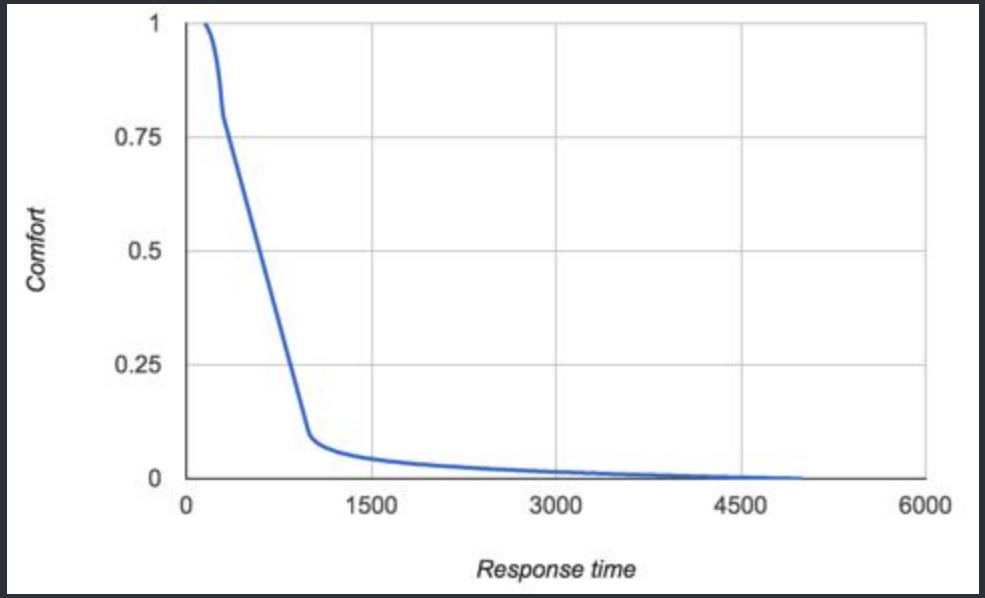
Great metrics are continuous, not discrete

discrete metrics are noise amplifying.

Key insight: though 150ms response is the ideal, the underlying data is more like this:



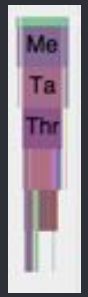
Bit of interpolation gives you comfort:



If you're bored, you can help out!

http://goo.gl/Z3XM79

But what about efficiency?

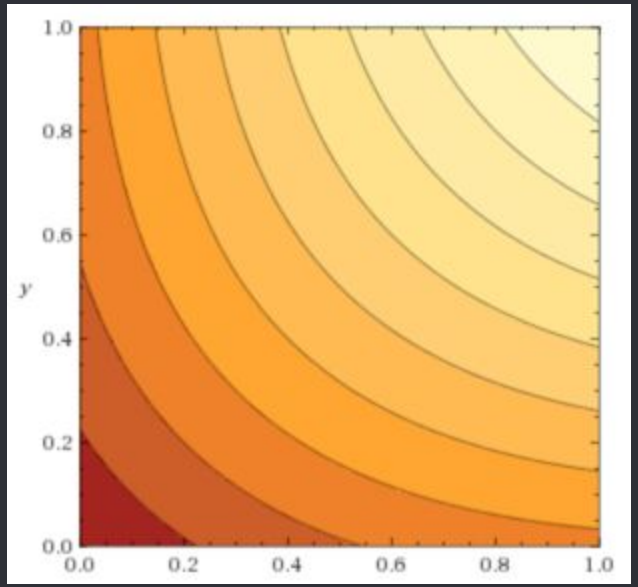
Eg: 

is better than 

Efficiency ~= cputime used / #cpus

Nicely catches this: 

Final IR scoring: blend comfort & efficiency



*we're still fiddling with weights of comfort vs efficiency

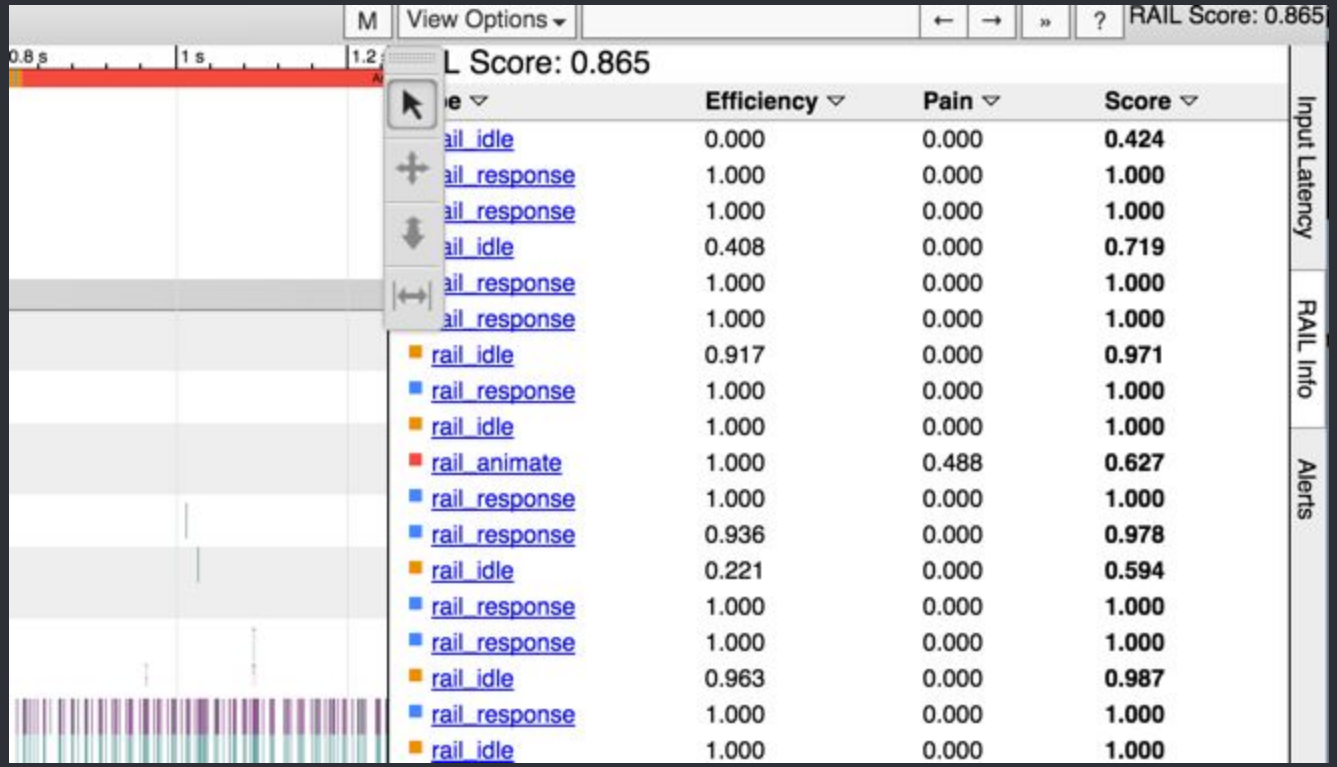
Final RAILscore based on peak end rule

~= worst experience dominates your opinion of the overall

E.g. a weighted average

- Each IR gets a score from [0-1]
- Better scores are damped, e.g. pow(ir.score, 2)
- Sum up and divide

Demo


M
View Options
RAIL Score: 0.865
0.8 s
1 s
1.2
L Score: 0.865
Efficiency
Pain
Score
ail_idle 0.000 0.000 0.424
ail_response 1.000 0.000 1.000
ail_response 1.000 0.000 1.000
ail_idle 0.408 0.000 0.719
ail_response 1.000 0.000 1.000
ail_response 1.000 0.000 1.000
rail_idle 0.917 0.000 0.971
rail_response 1.000 0.000 1.000
rail_idle 1.000 0.000 1.000
rail_animate 1.000 0.488 0.627
rail_response 1.000 0.000 1.000
rail_response 0.936 0.000 0.978
rail_idle 0.221 0.000 0.594
rail_response 1.000 0.000 1.000
rail_response 1.000 0.000 1.000
rail_idle 0.963 0.000 0.987
rail_response 1.000 0.000 1.000
rail_idle 1.000 0.000 1.000
Input Latency
RAIL Info
Alerts

Computing RAILscores at scale

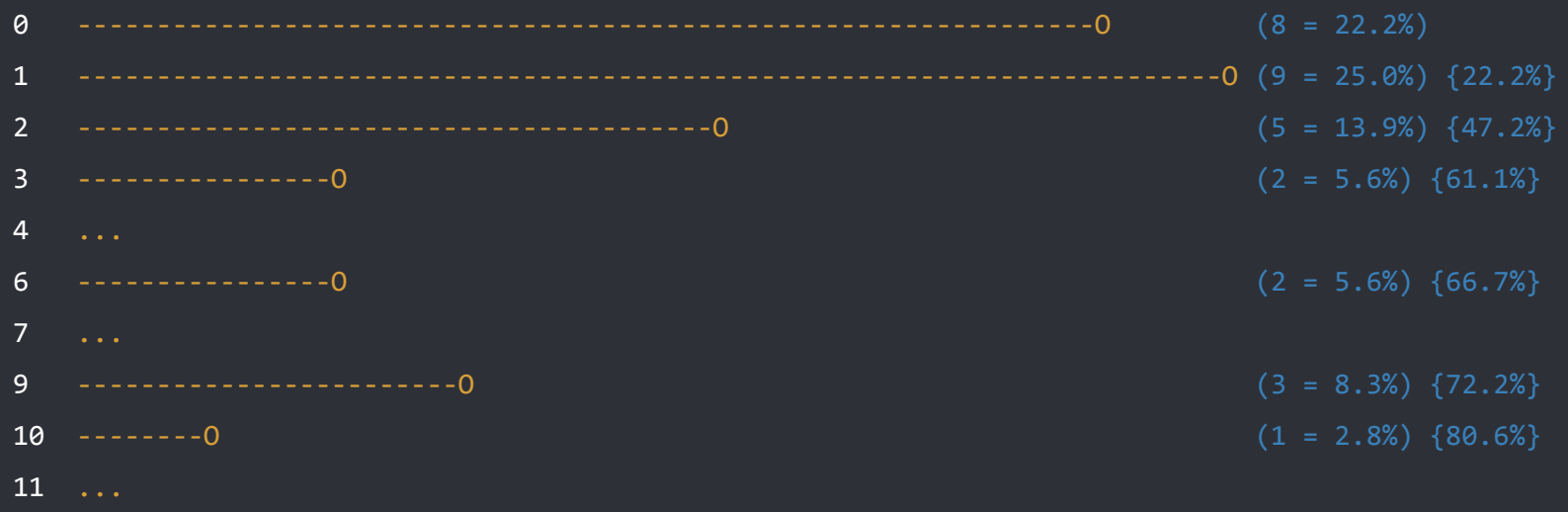
Perf Insights & Pi Reports

Core idea

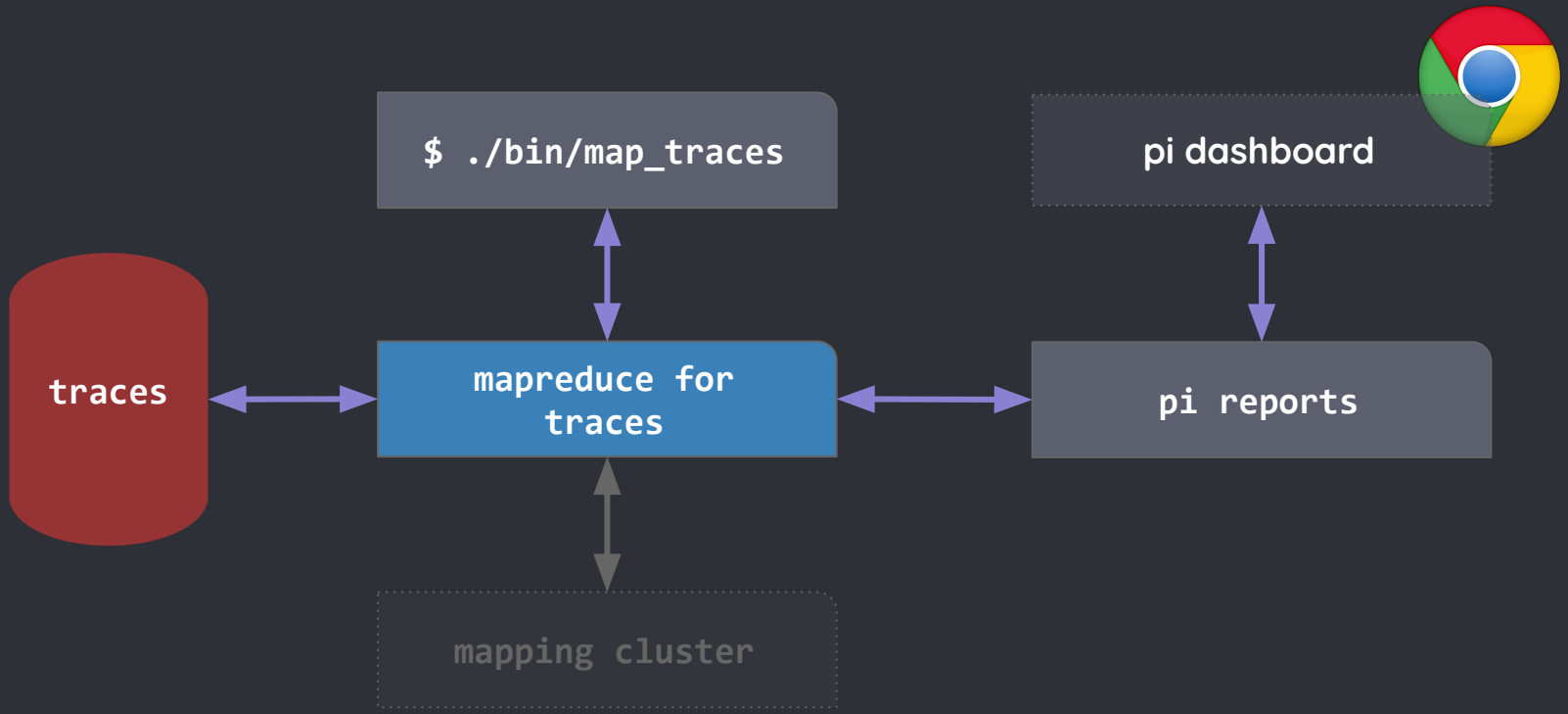
Lets bulk process traces.

LOTS of traces.

```
$ ./map_traces -j500 task_durations_histogram.html
```



Lë blôçk diagråm!



Writing a mapper

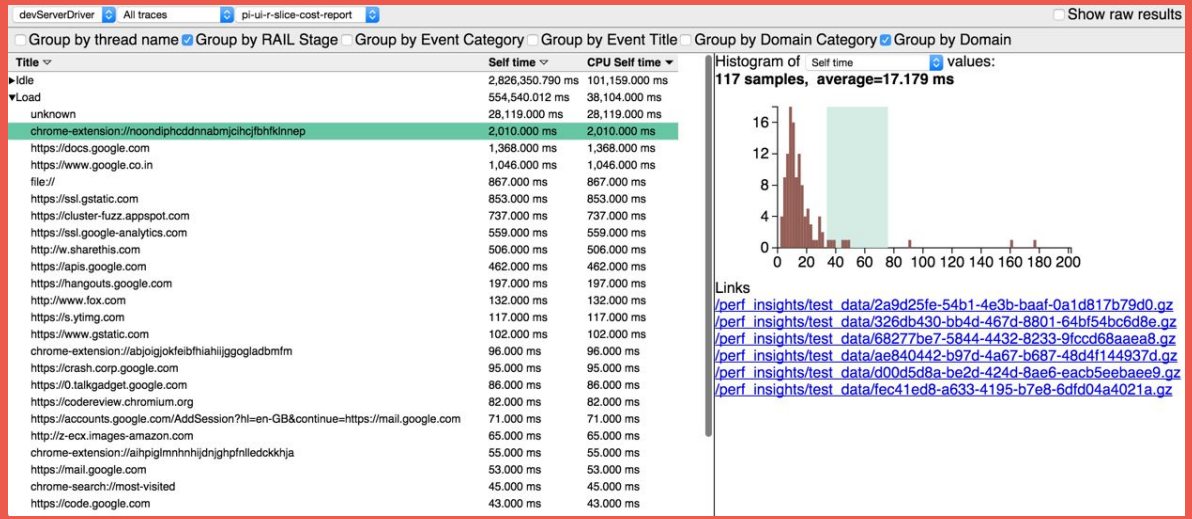
```
function myMapFunction(mapInput, mapResults, args) {
  ...
}
```

mapInput	An object `{ traceHandle, model }`.
model	Where the real stuff lives: events, flows, metadata... Learn about it at go/perfinsights-reports
mapResults	Call `addResults(strKey, jsonableObj)` as needed.

Writing a mapper

```
function myMapFunction(mapInput, mapResults, args) {
  mapInput.model.iterateAllSlices(function(slice) {
     var cat = UserFriendlyCategory.getForSlice(slice);
     if (!slice.isTopLevel) return;
     var qd = getQueuingDuration(slice);
     mapResults.addValue(cat, qd);
  });
}
```

Demo!


devServerDriver
All traces
pi-ui-r-slice-cost-report
Show raw results
Group by thread name
Group by RAIL Stage
Group by Event Category
Group by Event Title
Group by Domain Category
Group by Domain
Title
Self time
CPU Self time
▸Idle 2,826,350.790 ms 101,159.000 ms
▾Load 554,540.012 ms 38,104.000 ms
unknown 28,119.000 ms 28,119.000 ms
chrome-extension://noondiphcddnnabmjcihcjfbhfklnnep 2,010.000 ms 2,010.000 ms
https://docs.google.com 1,368.000 ms 1,368.000 ms
https://www.google.co.in 1,046.000 ms 1,046.000 ms
file:// 867.000 ms 867.000 ms
https://ssl.gstatic.com 853.000 ms 853.000 ms
https://cluster-fuzz.appspot.com 737.000 ms 737.000 ms
https://ssl.google-analytics.com 559.000 ms 559.000 ms
http://w.sharethis.com 506.000 ms 506.000 ms
https://apis.google.com 462.000 ms 462.000 ms
https://hangouts.google.com 197.000 ms 197.000 ms
http://www.fox.com 132.000 ms 132.000 ms
https://s.ytimg.com 117.000 ms 117.000 ms
https://www.gstatic.com 102.000 ms 102.000 ms
chrome-extension://abjoigjokfeibfhiahiijggogladbmfm 96.000 ms 96.000 ms
https://crash.corp.google.com 95.000 ms 95.000 ms
https://0.talkgadget.google.com 86.000 ms 86.000 ms
https://codereview.chromium.org 82.000 ms 82.000 ms
https://accounts.google.com/AddSession?hl=en-GB&continue=https://mail.google.com 71.000 ms 71.000 ms
http://z-ecx.images-amazon.com 65.000 ms 65.000 ms
chrome-extension://aihpiglmnhnhijdnjghpfnlledckkhja 55.000 ms 55.000 ms
https://mail.google.com 53.000 ms 53.000 ms
chrome-search://most-visited 45.000 ms 45.000 ms
https://code.google.com 43.000 ms 43.000 ms
Histogram of Self time values:
117 samples, average=17.179 ms
16
12
8
4
0
0 20 40 60 80 100 120 140 160 180 200
Links
/perf_insights/test_data/2a9d25fe-54b1-4e3b-baaf-0a1d817b79d0.gz
/perf_insights/test_data/326db430-bb4d-467d-8801-64bf54bc6d8e.gz
/perf_insights/test_data/68277be7-5844-4432-8233-9fccd68aaea8.gz
/perf_insights/test_data/ae840442-b97d-4a67-b687-48d4f144937d.gz
/perf_insights/test_data/d00d5d8a-be2d-424d-8ae6-eacb5eebaee9.gz
/perf_insights/test_data/fec41ed8-a633-4195-b7e8-6dfd04a4021a.gz

how to get lots of traces?

Contact fmeawad@chromium.org and rschoen@chromium.org for more information

Try map_traces yourself...

https://goo.gl/6gZ8vC

What next?

Memory

Battery

Emerging Markets

Great User Experience is Not Just RAIL

Security

Connectivity Resilience

Annoyance

RAIL++

Kevin Dooley

And, if we can...

Analysis at a frame-level: which iframes are heaviest?

Make bulk trace analyses available to everyone, probably via a dashboard :)

Figure out how to expose some of this to web devs...