LibreOffice Conference 2021

# COOL performance

making collaboration slick & quick.

By Michael Meeks

General Manager Collabora Online

@michael_meeks michael.meeks@collabora.com

*"Stand at the crossroads and look; ask for the ancient paths, ask where the good way is, and walk in it, and you will find rest for your souls..." - Jeremiah 6:16*

LibreOffice Conference 2021

## Outline

**Basics of how COOL works**

**LibreOffice core Technology**

- Wiggly lines

**LOOLWSD / Kit**

- I/O and queueing

**Javascript:**

- Websocket
- String / Image handling & async
- DOM mutation
- JQuery / Select2

Collabora Online

LibreOffice
Conference 2021

# How COOL works:

## Browser

- Thin Javascript.
- Overlays for cursor / selection etc.
- Pan / zoom interpolation / shape overlays for fluid movement

## WSD

- Web Services Daemon – multiplexes all messages to/from the Kit

## Kit

- A securely contained & isolated LibreOffice
- Streams 'tiles' to the client as PNG images
  - has view of whole document: unusually zoomed out.
  - Has multiple views – one per user.

## User

- cognitive biases & perceptual fun.

Collabora Online

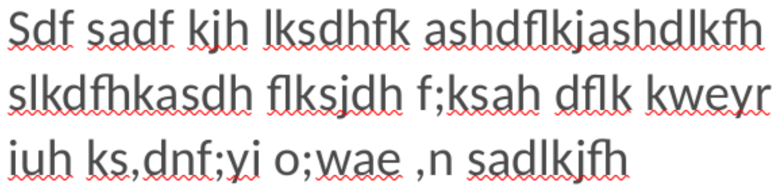
LibreOffice
Conference 2021

# LibreOffice core Tech.

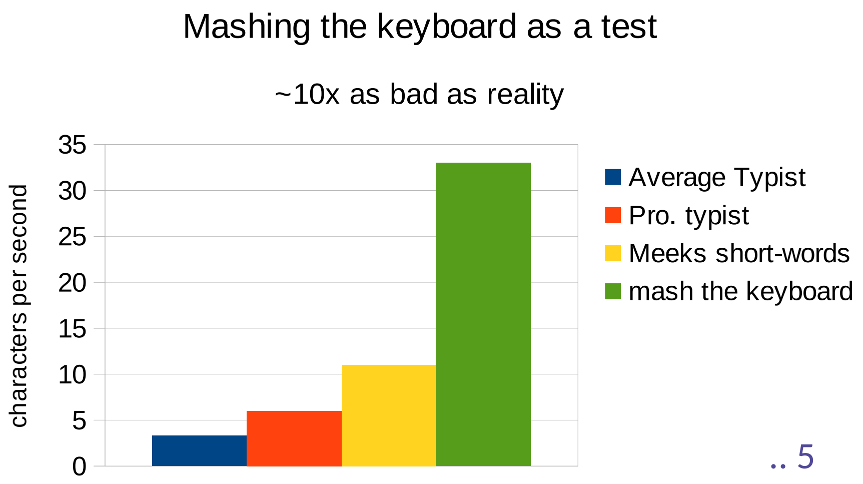
Collabora
Online

# Performance Testing & typing ...

- Customer feedback: *“we tested it with eight people doing random typing”*
- Profiled this use-case; it is/was slow
  - The mis-spelling squiggly-line (cf. wrong language setting?) ...
    - an unfeasible amount of CPU ~90% of rendering time
    - A most beautiful sub-divided, AA b-spline but ... ~2 pixels high mostly.
    - Fixed in 6.4.10
- Mashing the keyboard a pathological case: we’re still working on improving.
  - *Test your speed here*

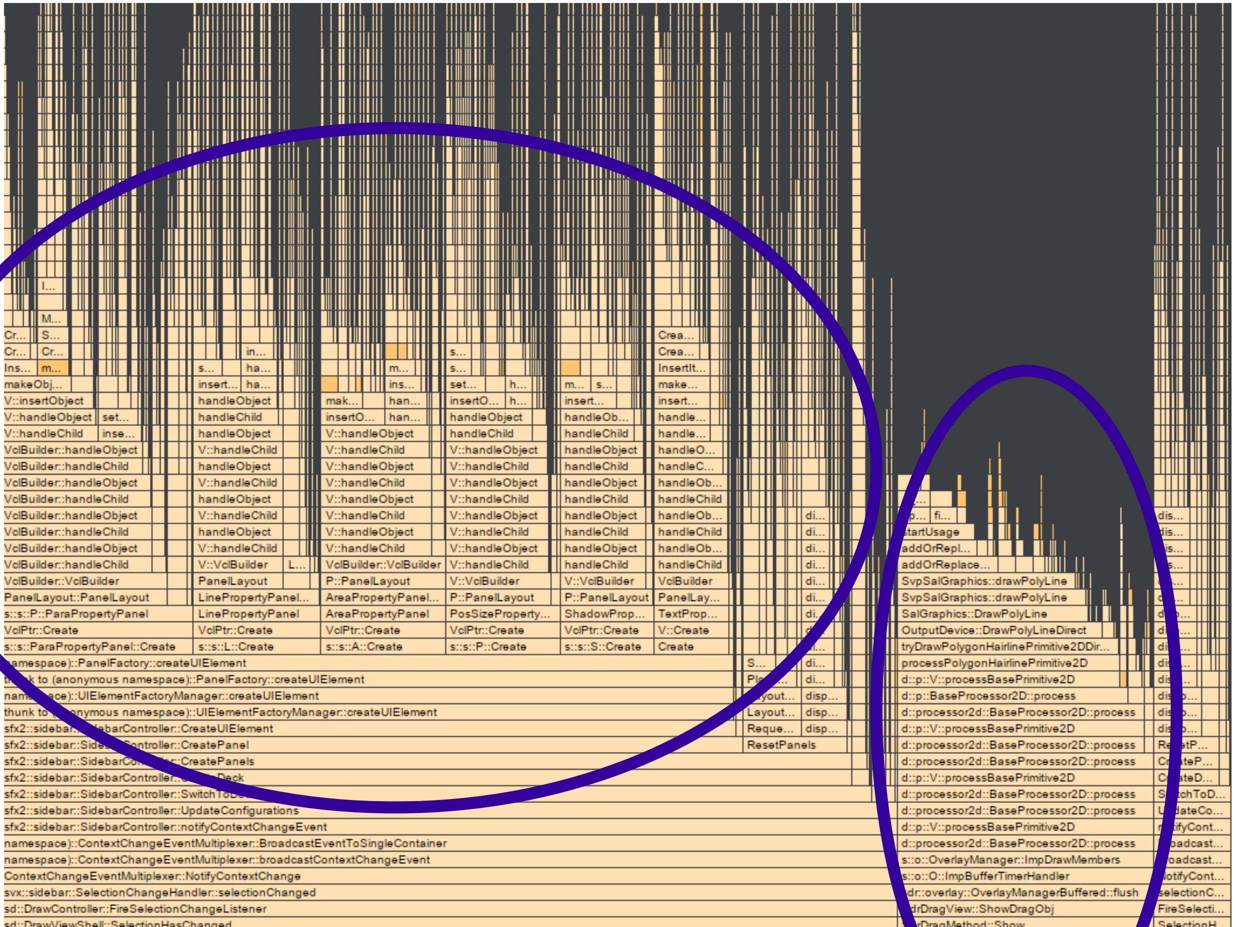


- Sdf sadf kjh lksdhfk ashdflkjashdlkfh slkdfhkasdh flksjdh f;ksah dflk kweyr iuh ks,dnf;yi o;wae ,n sadlkjfh



LibreOffice Conference 2021

# Slow to edit text / shapes in impress

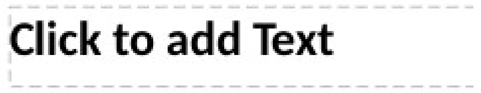


**Android: 5+ seconds to switch to edit mode:**

- **4s** – setting up property panels
- **1s** – rendering the dashed line around the edited text box.

Collabora Online

## Easyish fixes:


Click to add Text

**Sidebar**

- Android:
  - Defer sidebar setup to wizard / context being used.
- Cache sidebar panels
  - They are always there after first load: just hidden
    - Some layout excitement.
  - Much faster – after initial setup
- Quicker for all desktop users too.

**Dashed line rendering**

- Split into a large number of individual lines.
- Each of these then intersected with each other with an expensive algorithm
- Short-circuited:
  - Accelerated dashed line rendering.
- *Thanks to Armin Le-Grand*

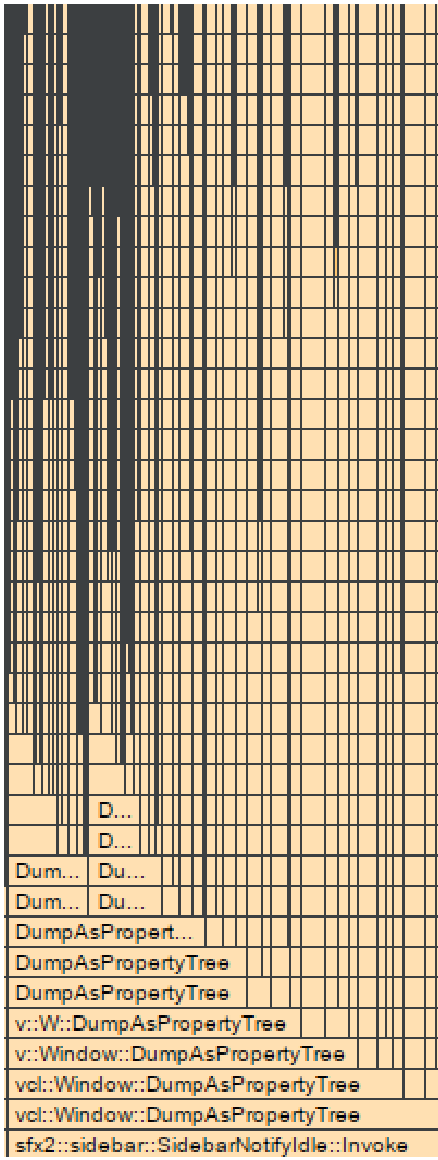
# JSON generation

**Lots of events generate JSON**

- Particularly side-bar & dialog – description of widgets:
  - Looots of JSON: DumpAsPropertyTree
- Switch from:

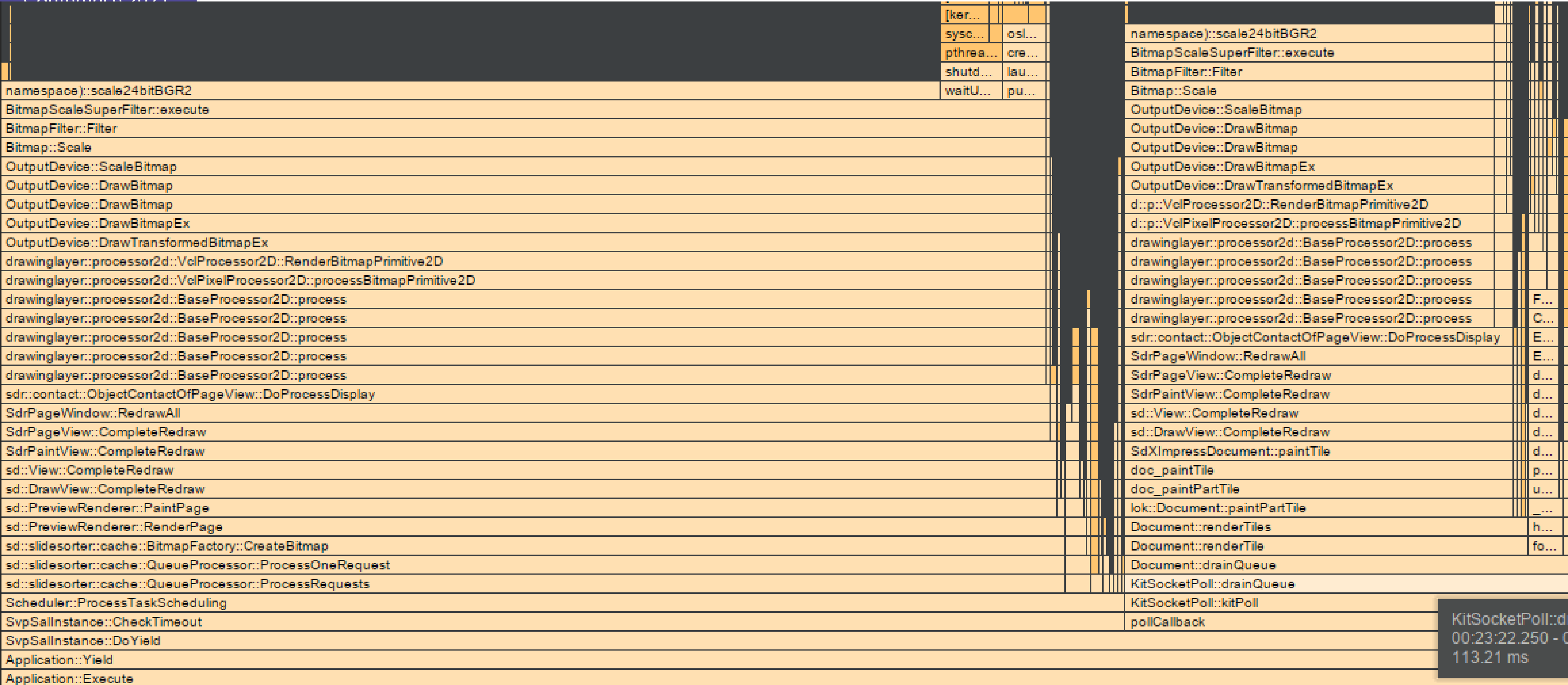
```
-boost::property_tree::ptree DumpAsPropertyTree()
+void DumpAsPropertyTree(tools::JsonWriter& rJsonWriter)
```

- Instead of deep duplicating & returning ptree's
- Implement a new JsonWriter
  - Ultimately a stream type interface anyway.
  - Disappears from the profile.
- *Thanks to Noel Grandin*



LibreOffice Conference 2021

# Image scaling & rendering



```
[ker...
sysc...      osl...
pthrea...    cre...
shutd...     lau...
waitU...     pu...
namespace)::scale24bitBGR2
BitmapScaleSuperFilter::execute
BitmapFilter::Filter
Bitmap::Scale
OutputDevice::ScaleBitmap
OutputDevice::DrawBitmap
OutputDevice::DrawBitmap
OutputDevice::DrawBitmapEx
OutputDevice::DrawTransformedBitmapEx
drawinglayer::processor2d::VclProcessor2D::RenderBitmapPrimitive2D
drawinglayer::processor2d::VclPixelProcessor2D::processBitmapPrimitive2D
drawinglayer::processor2d::BaseProcessor2D::process
drawinglayer::processor2d::BaseProcessor2D::process
drawinglayer::processor2d::BaseProcessor2D::process
drawinglayer::processor2d::BaseProcessor2D::process
drawinglayer::processor2d::BaseProcessor2D::process
sdr::contact::ObjectContactOfPageView::DoProcessDisplay
SdrPageWindow::RedrawAll
SdrPageView::CompleteRedraw
SdrPaintView::CompleteRedraw
sd::View::CompleteRedraw
sd::DrawView::CompleteRedraw
sd::PreviewRenderer::PaintPage
sd::PreviewRenderer::RenderPage
sd::slidesorter::cache::BitmapFactory::CreateBitmap
sd::slidesorter::cache::QueueProcessor::ProcessOneRequest
sd::slidesorter::cache::QueueProcessor::ProcessRequests
Scheduler::ProcessTaskScheduling
SvpSalInstance::CheckTimeout
SvpSalInstance::DoYield
Application::Yield
Application::Execute
```

```
namespace)::scale24bitBGR2
BitmapScaleSuperFilter::execute
BitmapFilter::Filter
Bitmap::Scale
OutputDevice::ScaleBitmap
OutputDevice::DrawBitmap
OutputDevice::DrawBitmap
OutputDevice::DrawBitmapEx
OutputDevice::DrawTransformedBitmapEx
d::p::VclProcessor2D::RenderBitmapPrimitive2D
d::p::VclPixelProcessor2D::processBitmapPrimitive2D
drawinglayer::processor2d::BaseProcessor2D::process
drawinglayer::processor2d::BaseProcessor2D::process
drawinglayer::processor2d::BaseProcessor2D::process
drawinglayer::processor2d::BaseProcessor2D::process            F...
drawinglayer::processor2d::BaseProcessor2D::process            C...
sdr::contact::ObjectContactOfPageView::DoProcessDisplay        E...
SdrPageWindow::RedrawAll                                       E...
SdrPageView::CompleteRedraw                                    d...
SdrPaintView::CompleteRedraw                                   d...
sd::View::CompleteRedraw                                       d...
sd::DrawView::CompleteRedraw                                   d...
SdXImpressDocument::paintTile                                  d...
doc_paintTile                                                  p...
doc_paintPartTile                                              u...
lok::Document::paintPartTile                                   _...
Document::renderTiles                                          h...
Document::renderTile                                           fo...
Document::drainQueue
KitSocketPoll::drainQueue
KitSocketPoll::kitPoll
pollCallback
```

```
KitSocketPoll::d
00:23:22.250 - (
113.21 ms
```

LibreOffice
Conference 2021

## Continual re-scaling of bitmaps

**We had a nice image scaling cache:**

- Problem: only caches one size per image
- For (random) reasons: not working nicely on Android.
- Now we have a multi-resolution scaled image cache:
    - Hugely faster, particularly for large zoom-out

**Online**

- Now we scale the cache size based on the number of open views
- Great for multiple users at different zooms
- *Thanks to Lubos Lunak*

Collabora Online

LibreOffice Conference 2021

## Pointless ~O(n^3) in SwRegionRects

**SwRegionRects::Compress()**

- Notionally saves effort & space by compressing invalidated rectangles together.
- Particularly problematic with COOL – since the document is always visible in a gigantic pseudo-view.

**Now only ~O(n^2) in number of regions**

- https://gerrit.libreoffice.org/c/core/+/122121

*Thanks to Lubos Lunak*

**Should accelerate all large writer documents with complex invalidations.**

Collabora Online

LibreOffice Conference 2021

## Calc: ScDocument::GetPrintArea

**Called surprisingly often**

- Switching views, when re-rendering a region etc.

**Pixel area dependent on zoom**

- Row heights vary in real height based on zoom level
    - But all look the same height.

**So – scan from the beginning ...**

**Cost is all in:**

- ScTable::GetRowForHeight(sal_uLong nHeight)

**Now massively faster**

- Walks both ‘hidden’ and ‘height’ span-trees concurrently – in jumps.
- Instead of iterating row by row.

Collabora Online

LibreOffice Conference 2021

# And much more in core ...

**Checkout Noel's talk:**

- I feel the need … The need for speed …

**Lots of misc other pieces**

- Faster file opening
- Better font caching to accelerate text rendering
- Quicker scrolling
- Quicker spreadsheet filtering
- Faster large chart insertion/setup

**Don't paint to windows**

- In LOK mode we used to often calculate & paint to an invisible 1x1 pixel window
- Avoid repeated writer layout calls too.

**Detail overload …**

Collabora Online

LibreOffice
Conference 2021

## Web Service Daemon / Kit

Collabora
Online

LibreOffice Conference 2021

## Shuffling vectors ...

**Buffering outgoing socket data: std::vector<char>**

- Transmit from the beginning and then erase(begin(), begin() + sentBytes)
- Unfortunately: SSL: 16k max writable chunks
- 20Mb images / document downloads common
- Shuffling ~10Mb average - 1200x times down a vector – not fast.

**Buffer class**

- Wrap a std::vector<char>
- Don’t erase – have an offset: send 1Mb at a time before shuffling
    - bingo – 64x faster.

Collabora Online

LibreOffice
Conference 2021

## STL / Android amazement

**STL on Android is abysmal**

- Thankfully we no longer have to binary-patch it at run-time; but …

**vector::~vector<char>**

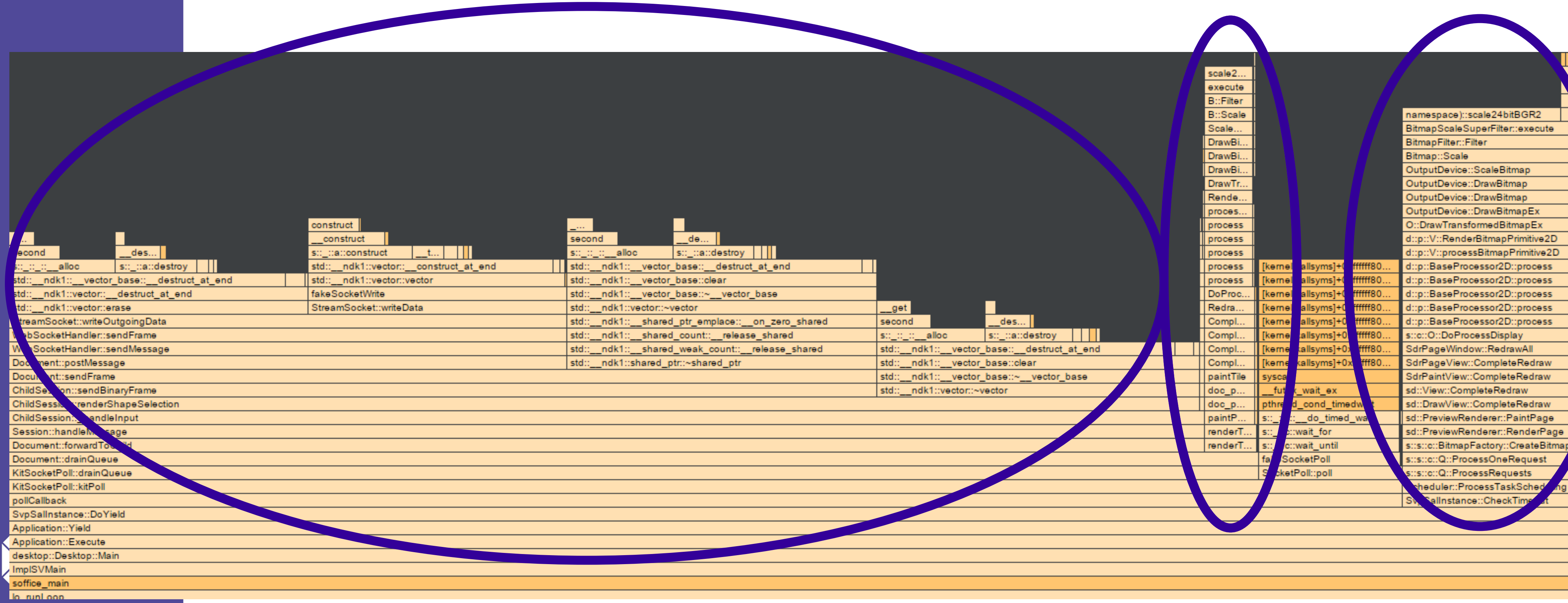
- Very high on the profile – doing some ‘0’ assignment in a loop while destroying ?
- allocation – understandably slow – but freeing [!] …
- More time spent allocating, wiping & freeing std::vector<char>
- Than rendering document content: huh !
- calloc buffer to render into instead.

Collabora
Online

# And here it is:

Vector folly:

Scaling bitmaps,
rendering tiles etc.



# Merge key-events

**Under heavy-load**

- Can't process key-events in the time they come in:

**Input event compression:**

- Kill un-necessary keyup events, then:

```
    child-foo textinput id=0 text=f
    child-foo textinput id=0 text=o
    child-foo textinput id=0 text=o   → Turn it into:
    child-foo textinput id=0 text=foo
```

- So we can catch-up … (also for removetextcontext (backspace/delete) events)
- *Thanks to Tor Lillqvist.*

# Asynchronous save ...

**Previously**

- Paused all document editing during save + up-load

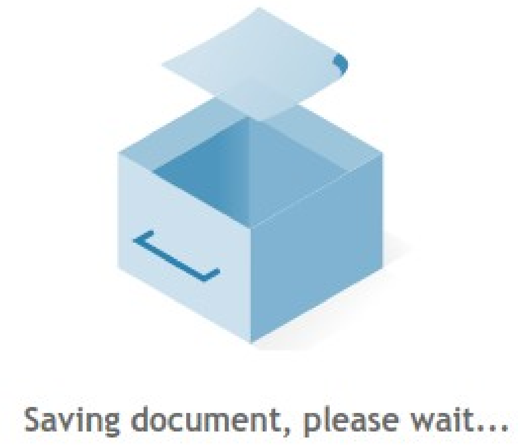
**Up-load**

- Thought to be fast: data-center ↔ data-center internal network link & storage.
- But ... some backends: several seconds
- So re-worked to continue editing while we up-load.
- *Thanks to Ashod Nakasian*

**Solves autosave 'stalls' while typing**

**Even so some things sync still:**

- Rename for example
- So be pretty there:


Saving document, please wait...

LibreOffice Conference 2021

## Javascript

Collabora Online

LibreOffice
Conference 2021

# End to end profiling

**Catching badness across the board**

- Found that we had been optimizing the wrong piece.
- So implemented a new end-to-end profiler.

**Core: ProfileZone**

- Passing data back from Kit → WSD

**JS: TraceEvent logging**

- Passing data back from browser → WSD

**WSD:**

- ProfileZone code too.

**To enable:**

- Press ‘t’ in Help→About
- Needs: trace_event[@enable] config option in loolwsd.xml.

**Visualize:**

- Chrome profile renderer: see everything.

*Thanks to Tor Lillqvist*

Collabora Online

## Profiling: Javascript – the surprise

**We thought JS in the browser is fast**

- We obsessed about network latency & server-side performance.
  - We were mostly wrong.
  - (though lots of sillies on the server-side too ...)

**Please be careful with your JS**

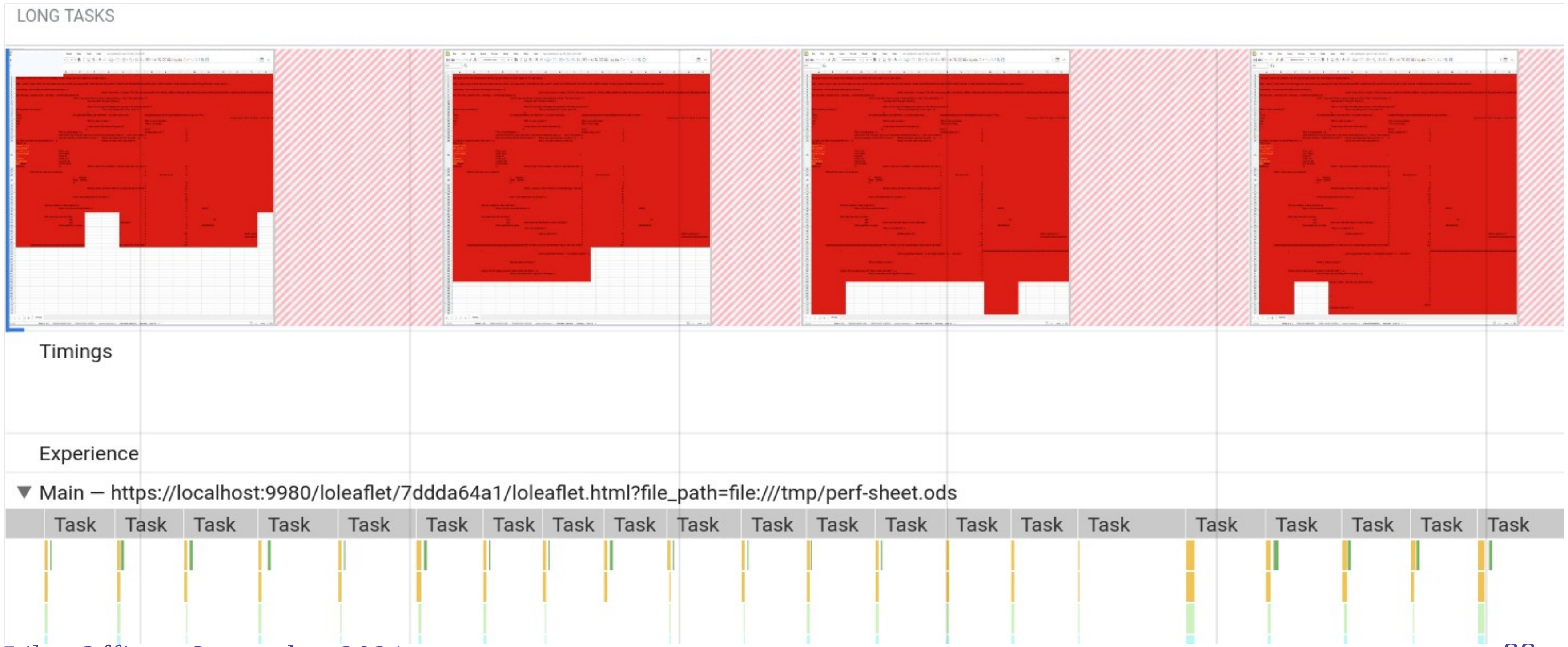
- DOM mutation, Canvas re-rendering, ‘elegant’ code using unusual libraries.

LibreOffice
Conference 2021

## Watch each tile render: ( spreadsheet with red background)

Websocket messages processed one by one at idle …

do a re-render → we see an animation of each tile rendering


LONG TASKS
Timings
Experience
▼ Main — https://localhost:9980/loleaflet/7ddda64a1/loleaflet.html?file_path=file:///tmp/perf-sheet.ods
Task Task Task Task Task Task Task Task Task Task Task Task Task Task Task Task Task Task Task Task Task

Collabora Online

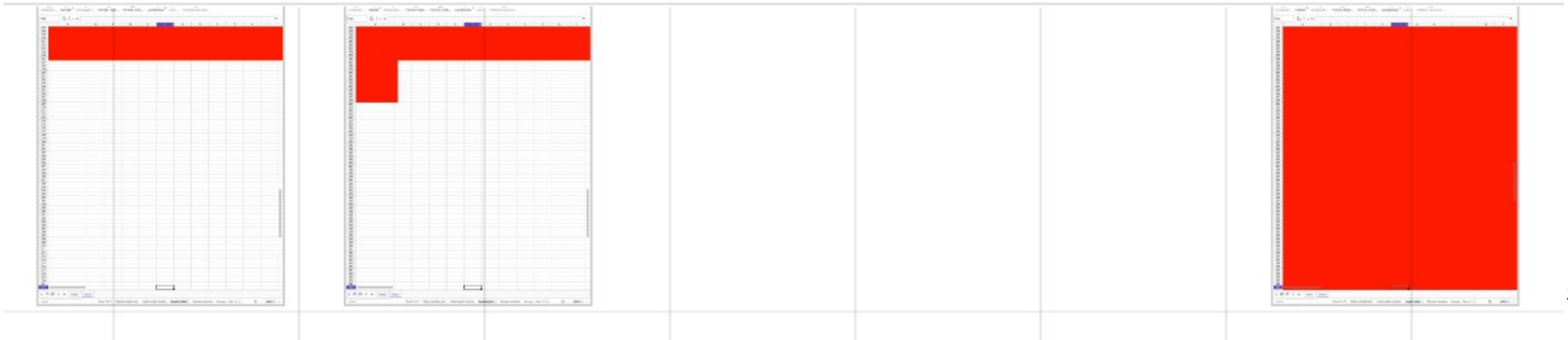
## Simple solution: (worth avoiding Promises too?)

```
// The problem: if we process one websocket message at a time, the
// browser -loves- to trigger a re-render as we hit the main-loop,
// this takes ~200ms on a large screen, and worse we get
// producer/consumer issues that can fill a multi-second long
// buffer of web-socket messages in the client that we can't
// process so - slurp and the emit at idle - its faster to delay!
_slurpMessage: function(e) {
    var that = this;
    if (!this._slurpQueue || !this._slurpQueue.length) {
        this._queueSlurpEventEmission(); // process in 1ms timer
        that._slurpQueue = [];
    }
    this._extractTextImg(e);
    that._slurpQueue.push(e);
},
```

**Same problem with async image load from .src=<base64 URL>**

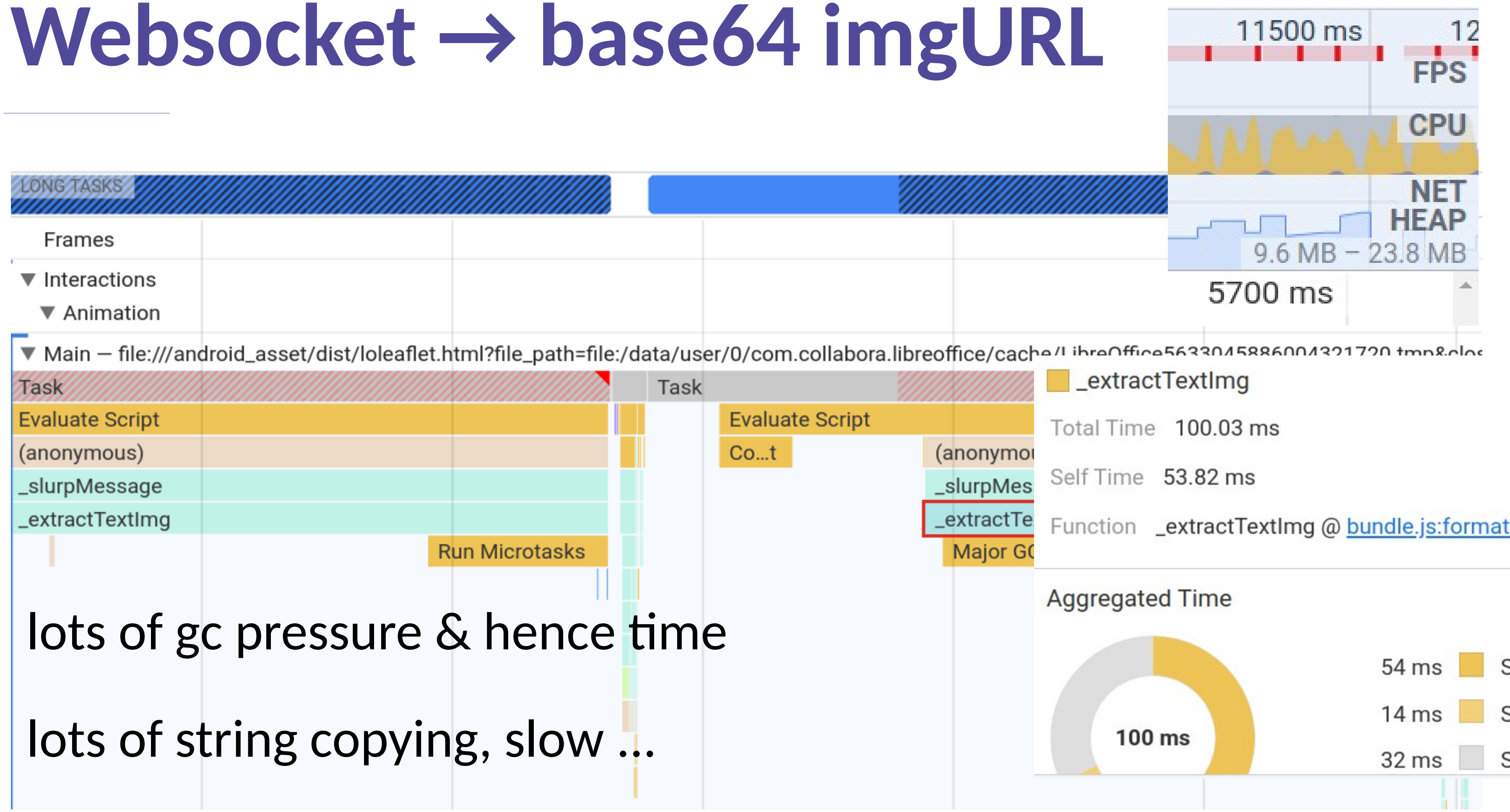
## Event emission:

```
_emitSlurpedEvents: function() {
    this._map._docLayer.pauseDrawing();

    try {
        for (var i = 0; i < queueLen; ++i) {
            var evt = this._slurpQueue[i];

            if (evt.isComplete()) {
                try {
                    // it is - are you ?
                    this._onMessage(evt);
                }
```



LibreOffice
Conference 2021

# Websocket → base64 imgURL



Main — file:///android_asset/dist/loleaflet.html?file_path=file:/data/user/0/com.collabora.libreoffice/cache/LibreOffice5633045886004321720.tmp&clos

| Task | Task |
|---|---|
| Evaluate Script | Evaluate Script |
| (anonymous) | Co...t / (anonymou |
| _slurpMessage | _slurpMes |
| _extractTextImg | _extractTe |
| Run Microtasks | Major GC |

_extractTextImg

Total Time 100.03 ms

Self Time 53.82 ms

Function _extractTextImg @ bundle.js:format

Aggregated Time

100 ms

54 ms
14 ms
32 ms

lots of gc pressure & hence time

lots of string copying, slow ...

Collabora Online

## Before code:

```
// read the tile data
var strBytes = '';
for (var i = 0; i < data.length; i++) {
    strBytes += String.fromCharCode(data[i]);
}
img = 'data:image/png;base64,' + window.btoa(strBytes);
```

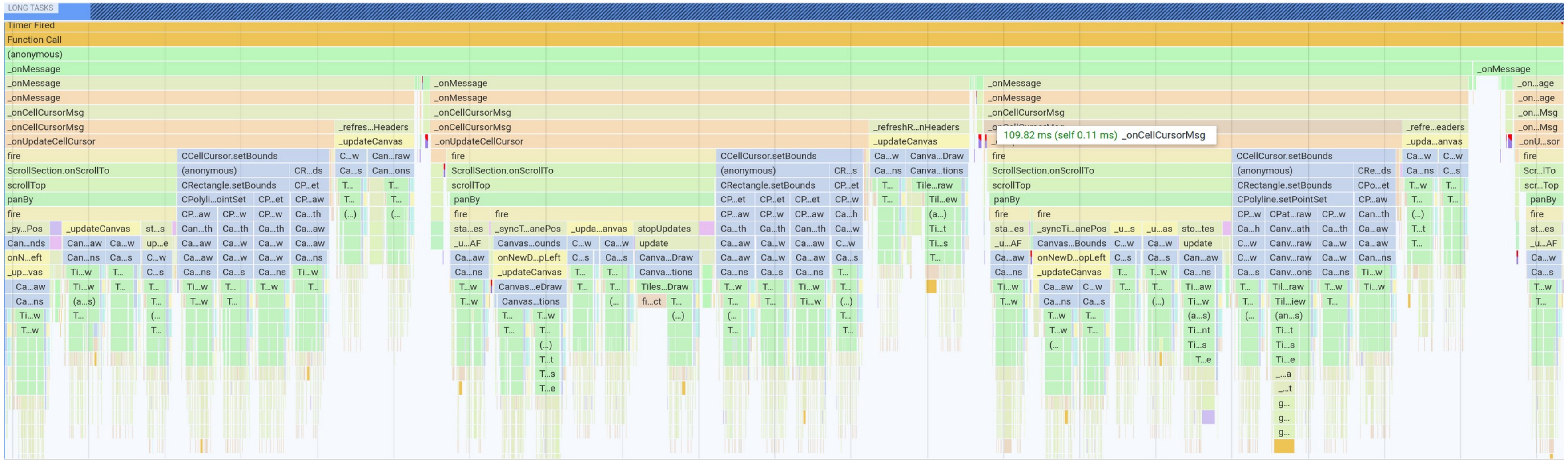
## After code:

```
// convert to string of bytes without blowing the stack if data is large.
_strFromUint8: function(data) {
    var i, chunk = 4096;
    var strBytes = '';
    for (i = 0; i < data.length; i += chunk)
        strBytes += String.fromCharCode.apply(null, data.slice(i, i + chunk));
    strBytes += String.fromCharCode.apply(null, data.slice(i));
    return strBytes;
},
…
img = 'data:image/png;base64,' + window.btoa(this._strFromUint8(data));
```

LibreOffice
Conference 2021

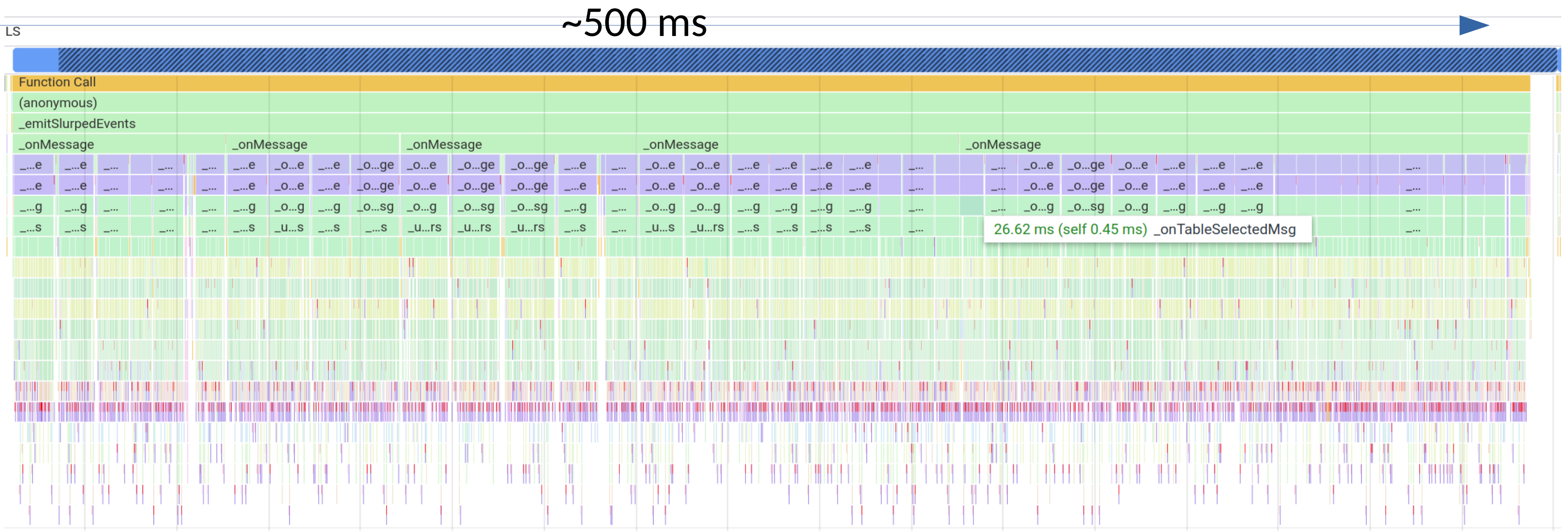
# Invisibly repeating the same work.

Now we: delay all the cursor related onScrollTo work / etc. until we have processed our whole incoming queue



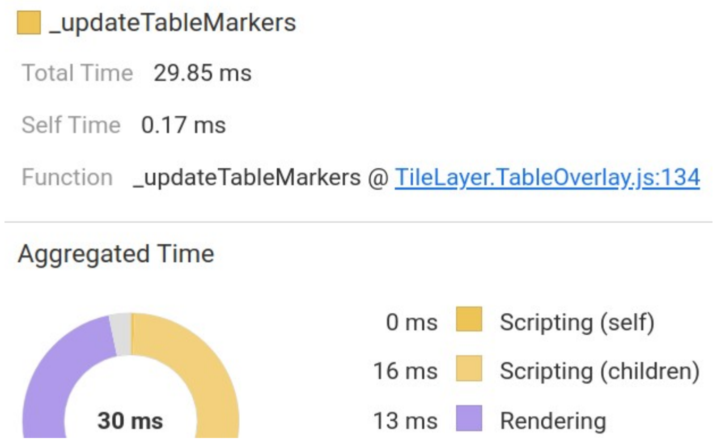
Collabora
Online

# Table handle DOM mutation



We were continually re-creating & destroying table handles for multiple redundant tableselected messages:

# 15x faster do it just once.



🟨 _updateTableMarkers

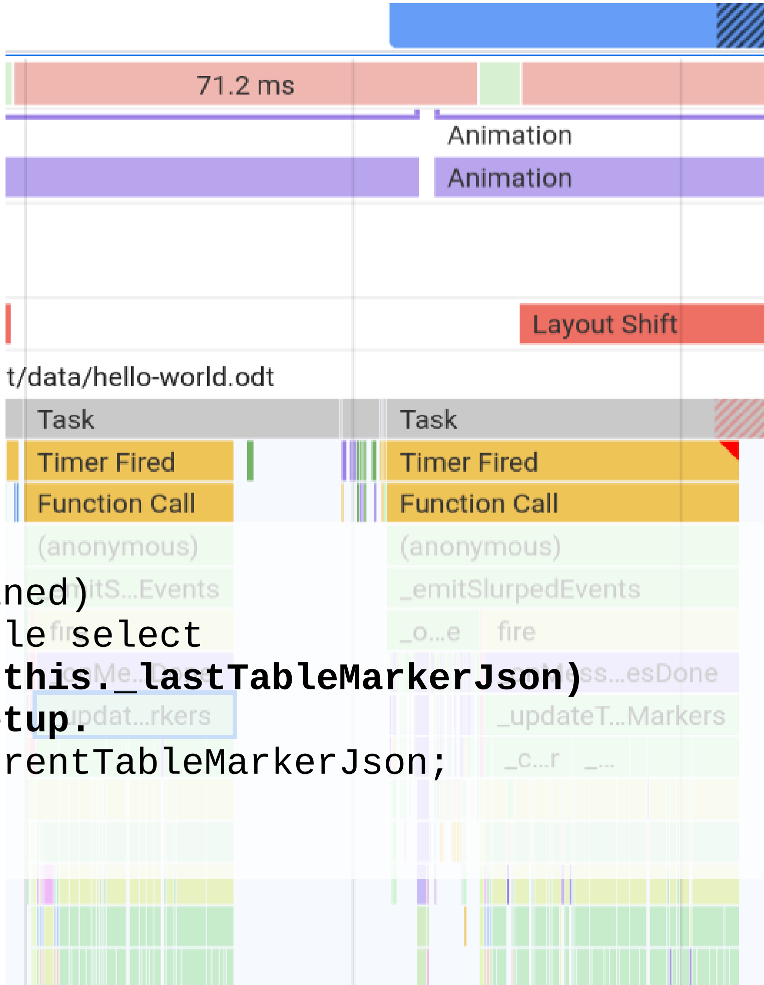
Total Time 29.85 ms

Self Time 0.17 ms

Function _updateTableMarkers @ TileLayer.TableOverlay.js:134

Aggregated Time

| | | |
|---|---|---|
| 0 ms | 🟨 | Scripting (self) |
| 16 ms | 🟨 | Scripting (children) |
| 13 ms | 🟪 | Rendering |



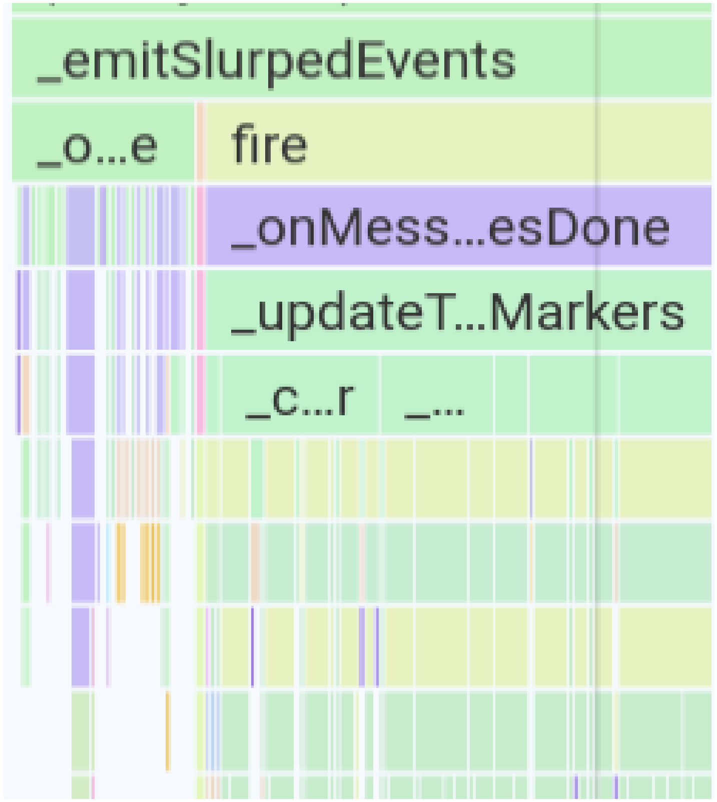
```
_updateTableMarkers: function() {
        if (this._currentTableData === undefined)
                return; // not writer, no table select
        if (this._currentTableMarkerJson === this._lastTableMarkerJson)
                return; // identical table setup.
        this._lastTableMarkerJson = this._currentTableMarkerJson;
```

avoid destroying & re-creating identical table handles

# ‘messagesdone’ to do it right easily:

**New ‘messagesdone’ event**

- fired when we have emitted all complete slurped messages
- If you’re updating view-state, re-render once at the end …



LibreOffice Conference 2021

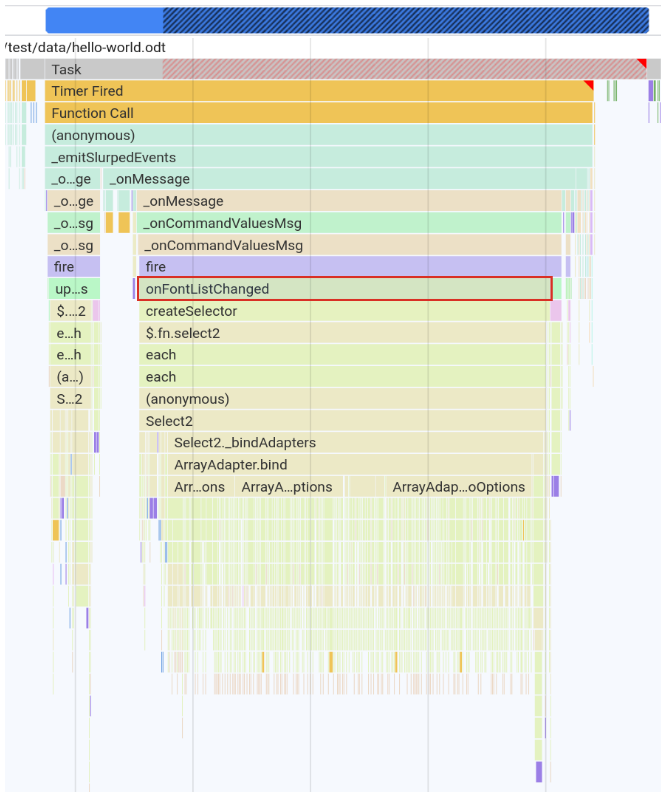
# JQuery plugin thrash:

**Select2 → argh !**

- That 31337 new JQuery plugin
- **800ms** on startup of thrash
- Saw this with jsdom → noticed it … ~5s+ of CPU time

**Thanks to Mert for fixing it**

- Using native JS now



Collabora Online

LibreOffice Conference 2021

## Conclusions: much faster

Much improved performance work for Collabora Online

- Lots of this in LibreOffice 7.2, more coming in 7.3
- Much of it shipping in COOL 6.4.11, more coming in COOL 2021

More work to do here

- more stress & profiling tools being written & used.
- We’re not even nearly done yet.

Collabora Online

# Make Open Source ROCK

# Thanks & Questions

**By Michael Meeks**

@mmeeks @CollaboraOffice

CollaboraOffice.com

CollaboraOffice.com/CODE

michael.meeks@collabora.com

*Oh, that my words were recorded, that they were written on a scroll, that they were inscribed with an iron tool on lead, or engraved in rock for ever! I know that my Redeemer lives, and that in the end he will stand upon the earth. And though this body has been destroyed yet in my flesh I will see God, I myself will see him, with my own eyes - I and not another. How my heart yearns within me. - Job 19: 23-27*