Collabora Productivity

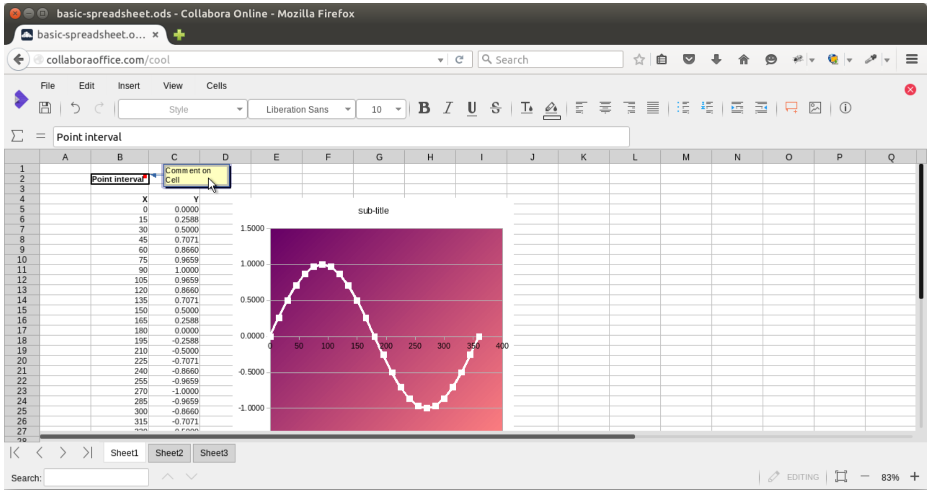
LibreOffice Online: Deep Dive

Ashod Nakashian

Collabora Productivity

LibreOffice Online

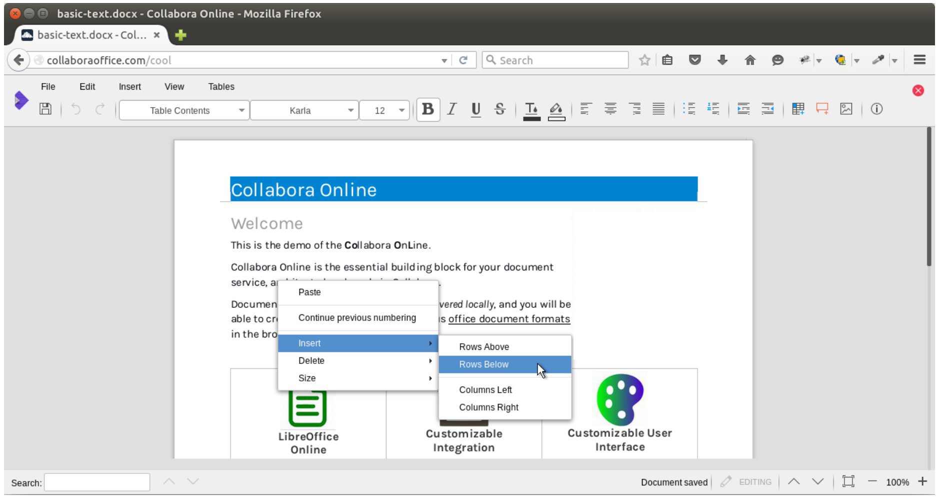
Calc with comment and graph



Point interval

	A	B	C	D
1				
2		Point interval		
3				
4		X	Y	
5		0	0.0000	
6		15	0.2588	
7		30	0.5000	
8		45	0.7071	
9		60	0.8660	
10		75	0.9659	
11		90	1.0000	
12		105	0.9659	
13		120	0.8660	
14		135	0.7071	
15		150	0.5000	
16		165	0.2588	
17		180	0.0000	
18		195	-0.2588	
19		210	-0.5000	
20		225	-0.7071	
21		240	-0.8660	
22		255	-0.9659	
23		270	-1.0000	
24		285	-0.9659	
25		300	-0.8660	
26		315	-0.7071	

Writer tables and context menu



Collabora Online

Welcome

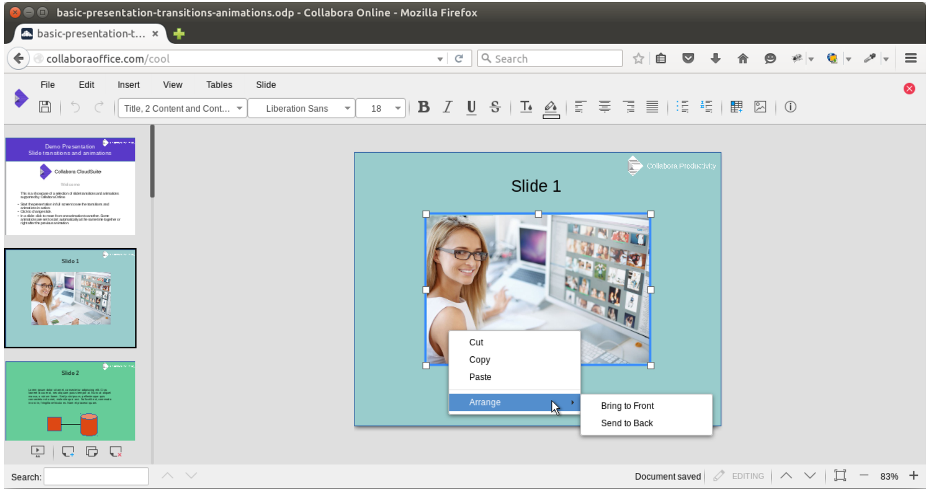
This is the demo of the **Co**llabora **O**n**L**ine.

Collabora Online is the essential building block for your document
service, a

Documen *vered locally*, and you will be
able to cr s office document formats
in the bro

LibreOffice Online	Customizable Integration	Customizable User Interface

Impress with embedded image and context menu



Overview

- Moving Online
  - With benefits comes challenges
  - Flexibility, mobility, availability
- Architecture
  - Self-serving Web-Services Daemon
  - One process per document
  - Process isolation (Jailing)
  - Flexible document storage integration
- Challenges
  - Fast, Interactive Rendering
  - Scalability

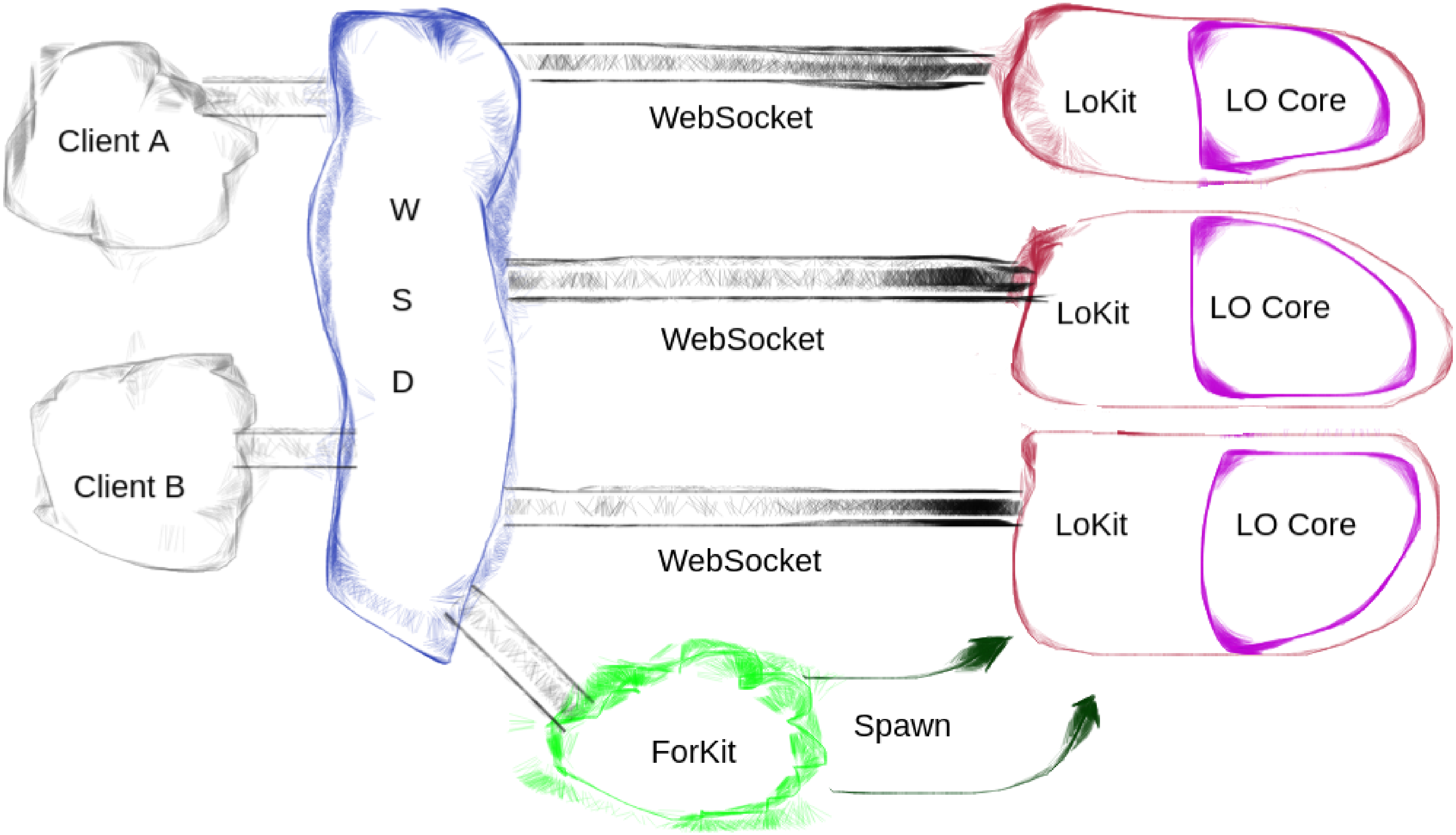
Moving Online

Moving Online

- Leverage LO Core
- Flexibility, mobility, availability
- With benefits comes challenges
  - Designing for low latency
  - Designing for high-scalability

Architecture

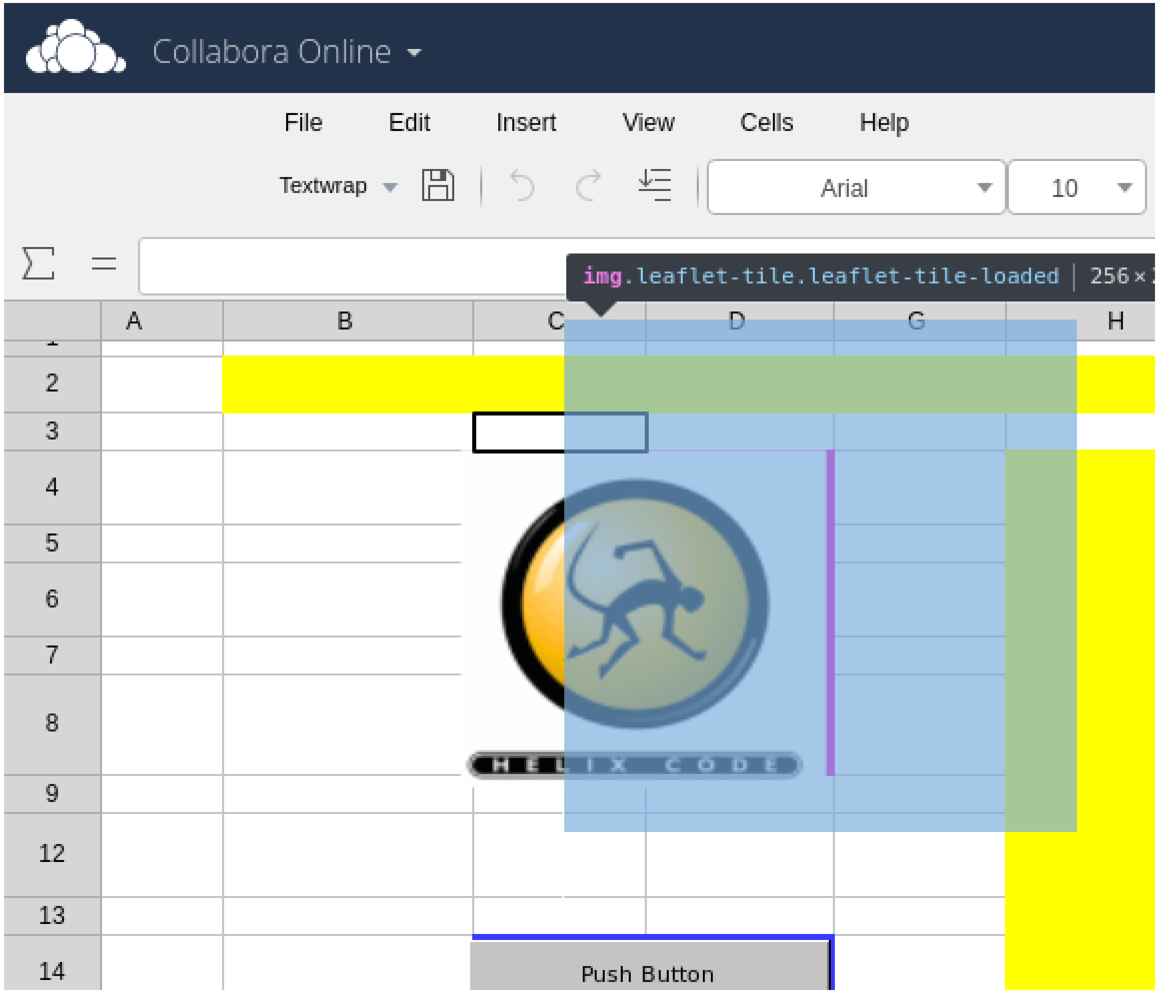
Architecture



Design Features

- Self-serving Web-Services Daemon
  - Powered by LibreOffice Core (see Miklos’s talk from yesterday)
  - One process per document
  - Collaborative Editing
  - Process isolation (Jailing)
  - Flexible document storage integration
- Web UI
  - JavaScript-powered UI
  - Portable, supports all major browsers
  - Built on top of, and extending, Leaflet: mapping UI
  - Integrates with ownCloud/nextCloud, more to come

Tiled Rendering

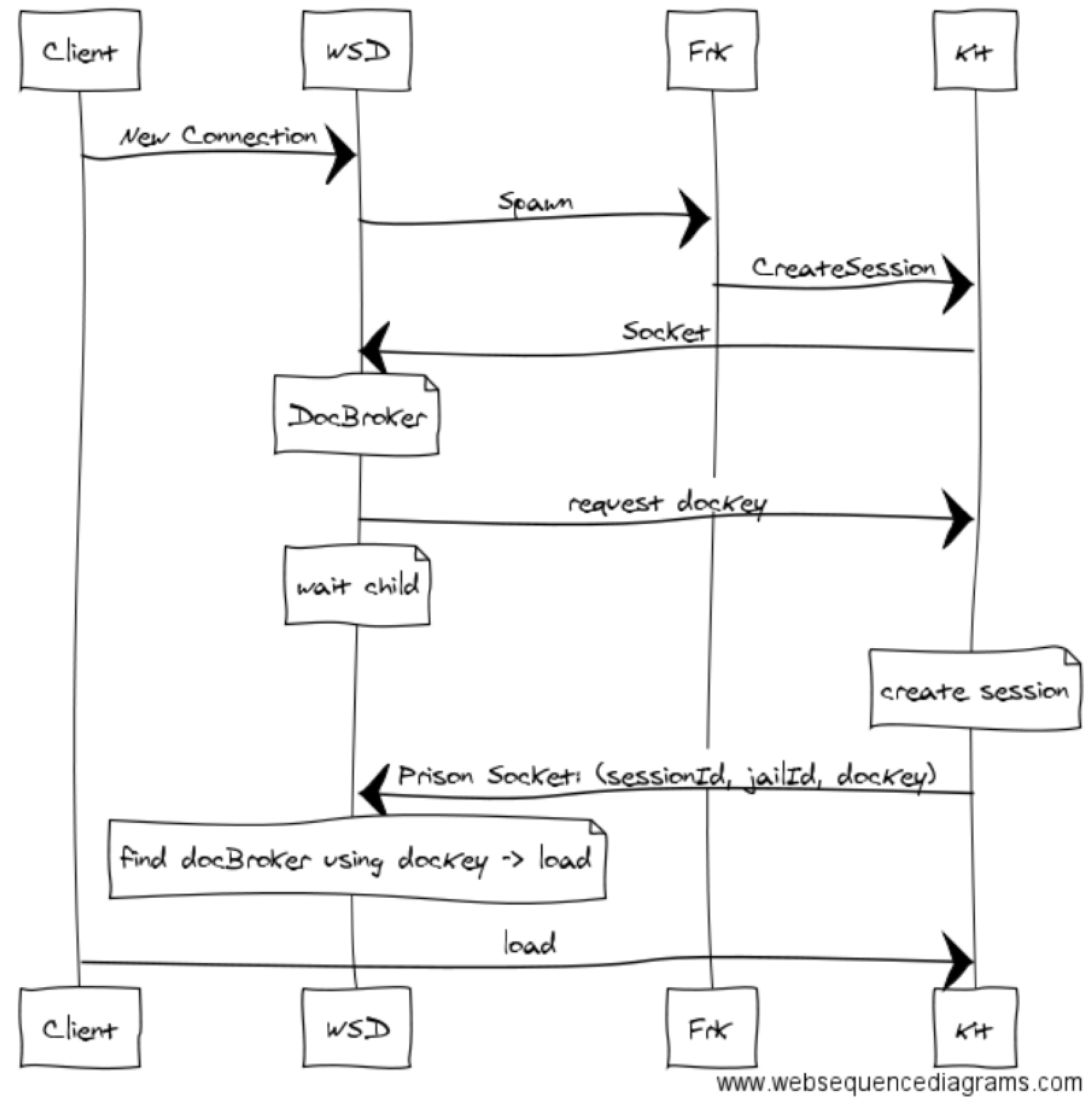


img.leaflet-tile.leaflet-tile-loaded | 256 ×

Tiled Rendering

- Tiles are internally queued and rendered synchronously
- Tiles are rendered once until invalidated
  - Rendering queue removes redundant request
  - Tiles are cached
  - Clients are served once tile is rendered and cached
- Tiles can be rendered in large blocks for better performance
  - Images might need rescaling for each tile
- Clients may cancel previously requested tiles, f.e. when the user jumps to a different page

New Document Load



Protocol

- Client → Server
  - Plain-text commands
  - All-lower command-names
  - Space-separated command arguments
- Server → Client
  - Plain-text responses
  - Only tiles have binary payload
  - JSON payloads for complex data

Protocol

- LO Core Kit Events
  - Plain-text events
  - Payloads space-separated fields or JSON
  - Events queued and pushed on idle
  - Event queue combines and de-duplicates events
  - Pull-model: Clients receive notification and is free to request data, or ignore
    - Possibly push out tiles proactively to reduce latency

LO Core Event Handling

- Two callbacks are registered with LO Kit
  - Global Callbacks: Handles document-specific events, such as status indicator.
  - View Callbacks: All interesting document activity is reported on this callback

- LO Core caches events and fires on Idle
  - Events are deduplicated and compressed
  - Events are queued up during an API call to better compress

Life-cycle of a Change

- Part 1: Input

1) User enters modifying input (ex. Key press)
2) LOLeaflet forwards the input to WSD
3) WSD forwards to the respective LOKit process
4) LOKit invokes respective LO Core API
5) LO Core modifies document, does composition and layouting
6) LO Core issues invalidation events on LOKit callbacks
7) LOKit forwards events to WSD
8) WSD forwards events to the UI

Life-cycle of a Change

- Part 2: Update
  1) UI issues requests for fresh tiles
  2) WSD forwards tile requests to LOKit
  3) LOKit invokes tile rendering API, compresses result to PNG
  4) LOKit sends tile response with PNG payload to WSD
  5) WSD forwards to the UI
  6) UI renders the new tile

Threading

- Internally there is a single LO Kit instance with potentially multiple views
- Each client socket runs on dedicated thread
- But internally calls on LO Kit instance is synchronized
    - SetView called before invoking an API

Scalability

Benchmarking with LoolStress

- We need numbers to tune and optimize Online
- LoolStress is a built-in tool to:
  - Can replay any session with timing precision
    - Recording is enabled via config in WSD
  - Can run a standard benchmark to collect stats in consumable numbers:

    Latency best: 16369 microsecs, 95th percentile: 26837 microsecs.

    Tile best: 13144 microsecs, rendering 95th percentile: 14933 microsecs.

    Cached best: 187 microsecs, tile 95th percentile: 318 microsecs.

    Rendering power: 4.77605 MPixels/sec.

    Cache power: 258.016 MPixels/sec.

Thank You

- <Your Question Here>
-