LIBREOFFICE IN YOUR BROWSER

WEBASSEMBLY & OTHER NEAT HACKS TO MAKE THAT HAPPEN

oSLO virtual conference, 2020-10-15

Who's talking?

Thorsten Behrens – thorsten.behrens@cib.de

- with CIB since 2015 – built the LibreOffice team here
- one of the LibreOffice f~~orkers~~/founders, and on the TDF board
- working with LibreOffice/OpenOffice code since 2001
- Hacker, computer scientist, fighting for Open Source and Open Standards

The State of the Art (LOOL)

- HTML5-canvas based browser version
- lightweight, tiled rendering
- the heavy lifting happens on the server
  - all documents of all users loaded there
  - all rendering & editing happens in the data center
- Pros:
  - light on the client
  - documents stay on-premise
  - ~easy collaborative editing – just one document instance
- Cons:
  - no offline mode
  - expensive to host
  - no peer2peer editing, or end2end encryption possible

Pricing & TCO for LOOL

- for running LOOL professionally, you have
  - cost of licensing (and support)
  - cost of operation
    - staff / maintenance / user support
    - cost of hosting
      - real-world needs (per *actively working* user):
      - 2-10 active users per CPU thread
      - 100MB per active user (if working on larger documents)
    - same *order of magnitude* as lightweight app virtualisation
  - so that's around 50-100 USD per average active user and year (license, support, and most importantly AWS bill)

Pain points of LOOL’s architecture

- price of hosting
- price of hosting
- and: price of hosting
  - ad-based ARPU industry average is <0.50 USD per year!
  - ARPU for Facebook is around 7.3 USD per year (and the largest)
- also no offline mode, bandwidth & latency requirements :-)

So what now?

LibreOffice WebAssembly – LWA

- Instead, looking at the trajectories of hardware (mobile/laptop)
  - your phone: CPUs with 8 core, up to 2GHz; 12GB RAM on the high-end
  - Ultrabooks with 32GB and 12-thread i7…
- do what we did even since before 2000 – port the core to a new architecture!
  - the new platform is … the browser!
  - WASM – compile native code to run in your browser
  - W3C standard since end of 2019 – WASM core
- where
  - use LibreOffice core
  - cross-compile to WASM (like we do for Android, iOS, Windows ARM etc)
  - use platform APIs whereever feasible (crypto, IO, network)

Project plan & timeline

- Yes, this is an announcement :)

Project plan & timeline (2)

- hope to start next month
  - with getting a cross-build going
- by the end of the year, latest FOSDEM
  - „1st pixel rendered“
- by Summer next year
  - edit text in Writer
- MVP Writer / e2e editing of documents within one year

Archictecture

- we tried that – it didn’t work?!
  - we gave up, as in 2015 emscripten/WASM couldn’t even do exceptions properly
- stars are aligned now
  - W3C standard, wide browser support
  - nothing missing really (except perhaps threading)
  - we know the market, there’s demand
- What needs doing?

  1) low-level cross building

  2) port big blobs to use browser APIs (NSS, I look at you!)

  3) strip down the monolith (target only Writer for a start)

Challenges

- Challenge – size of the binary
  - likely not feasible to load 100MB of WASM & survive
- Single-threaded
  - multi-threading is still experimental
  - then again, Writer is single-threaded since 1990
- Heap size
  - only 2GB (max) with current mem model, so we *really* need to put LibreOffice on a diet

Misc notes

- this is pure-play opensource
- no separate repo – all happens in core
- over time, this will grow JS GUI code, but that should be all below core (like android is already)

CIB

IDEAS WITH A SYSTEM

OUR PRODUCTS:

https://libreoffice.cib.de