Collabora Online: Async-Saving Design and Testing

By Ashod Nakashian

Consultant Collabora Online

ash@collabora.com

Agenda

- Problem
- Challenge
- Risks
- Solution
- Results

Collabora Online

The Problem

Saving

- Document contents must be saved to disk first.
- Once saving is completed, the file needs to be uploaded to storage.
- Saving is asynchronous; no UI interruptions.

Uploading

- Uploading to storage, via WOPI, can be unreliable.
- It can be very slow, get interrupted, or return temporary errors.
- Used blocking sockets, resulting in unresponsive UI while uploading.

Collabora Online

The Challenge

Sockets

- Upload using asynchronous connection instead of blocking.
  - But over-reliance on Poco makes this harder; drop-in replacement, ideally.
- Avoid per-request threads, avoid complex library integration.

Protocols, Application

- Support SSL, Terminating-SSL (for reverse proxies), and non-SSL setups.
- Support chunked transfer, redirection, connection reuse, etc.

Risks

HTTP

- Reinventing the wheel always problematic.
  - Requires extensive automated testing infrastructure.
- Significant reliance on Poco means less flexibility in new API design.

Application

- Saving and uploading are not always independent activities.
  - Both are done as part of more complex operations: renaming, converting, conflict resolution.
  - Requires state modelling and management.

Collabora Online

Solution

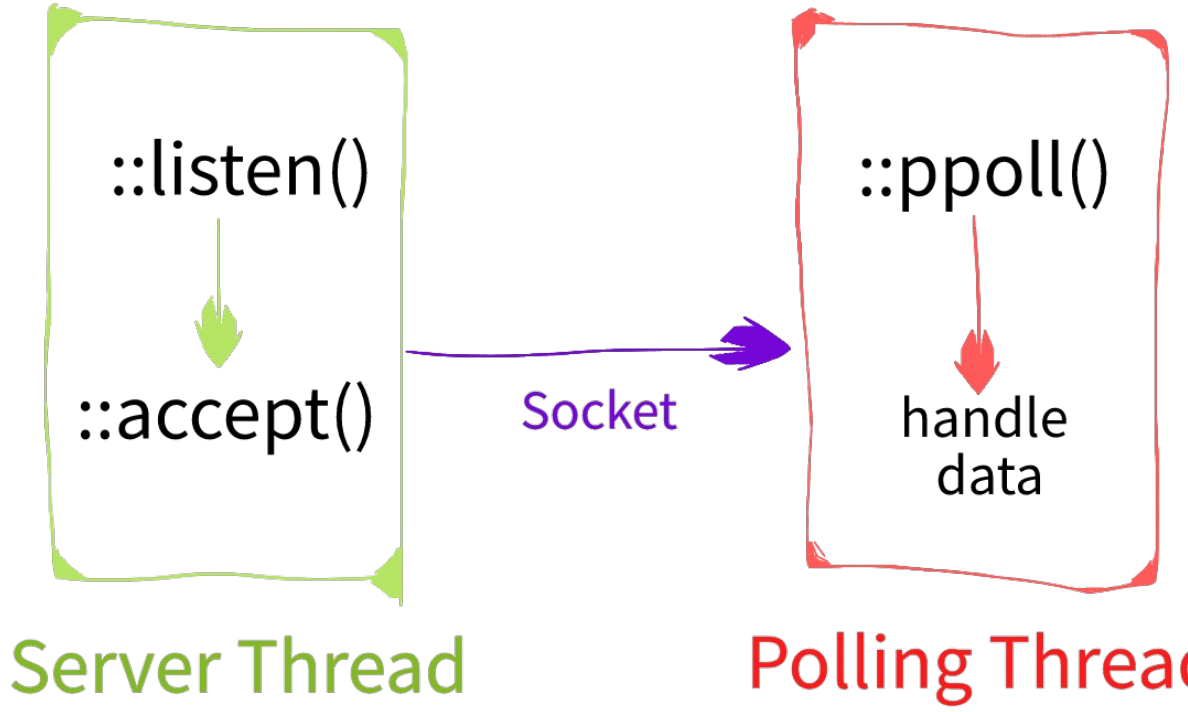
Sockets

- Leverage own asynchronous sockets for HTTP.
  - Used for all WebSocket communication, in production for years.
- Shared socket polling thread that polls on multiple sockets simultaneously.
- Support socket reuse, or per-request sockets.

Implementation

- New HTTP Request/Response handlers, with compliant parsers.
- Can run in blocking and non-blocking modes.
- Used for all HTTP requests, replacing Poco (mostly).

Low-Overhead Design



- One ppoll syscall
- One thread
- High scalability

Testing

Testing

- HTTP is reasonably self-contained and reasonably easy to test.
  - Leaving out networking.
- Extensive class-level unit-tests.
- Exhaustive coverage for incomplete data parsing.
- Built-in server: emulates all status codes, returns random data, echo, etc.
- External test-server tests, for compliance and regression testing.

Fuzzing

- Fuzzing for the HTTP protocol parsers.
- Fuzzing of the HTTP round-trip, with built-in server:
  - Client Request, Server Response, Client handling of response.

Collabora Online

Results

Performance

- UI completely unaffected by storage performance.
- Independent and potentially overlapping Save and Upload operations.
- Transparent and automatic retrying of failed Uploading.
- Smooth auto-saving user experience.
- Fuzzing shows thousands of loopback round-trips per-second, per core.

Implementation

- More homogeneous codebase with own sockets, polling, and now HTTP layers.
  - Fewer overlapping libraries reduces defect surface-area, improves readability.
- Better code integration, more maintainable, better performance.

Collabora Online

Thanks!
Happy to take questions

Collabora Online

By Ashod Nakashian
ash@collabora.com

@CollaboraOffice
hello@collaboraoffice.com
Collaboraoffice.com