CRDT and Local-first Architecture

A step forward from PWA

Yifeng Wang @ Toeverything

About Me

- 👨‍💻 Co-founder @Toeeverything, currently building AFFiNE & BlockSuite
- 👨‍🎨 Spent years working on text, graphics, and collaboration
- 🤐 Former tech blogger behind the Great Firewall
- 📜 Amateur JavaScript historian, translated *JavaScript the First 20 Years*
- 🌈 #OpenSourceEnthusiast, find me on GitHub @doodlewind

Outline

- 🚩 **Challenge**: Limitations with existing data storage and sharing approaches
- 💾 **Local-first software**: Prioritizing the usage of local resources over remote servers
- 🔗 **CRDT**: The technology that enables local-first architecture
- 💡 **AFFiNE**: A real-world use case

The Challenge

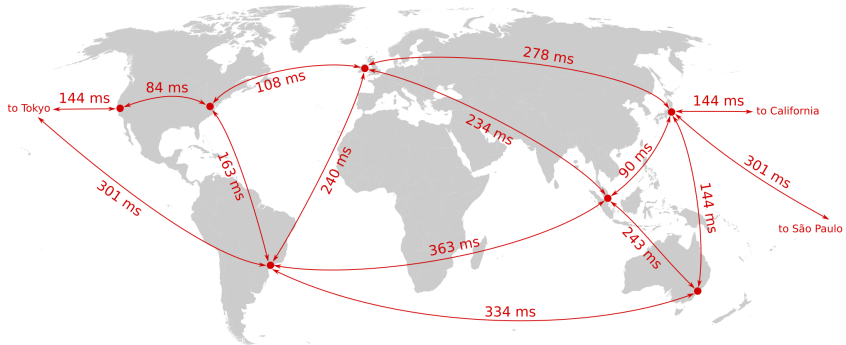
A quick recap of web apps

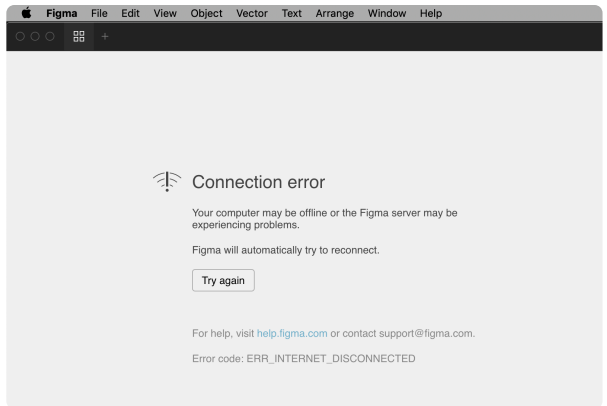
Web app comes with a lot of benefits...

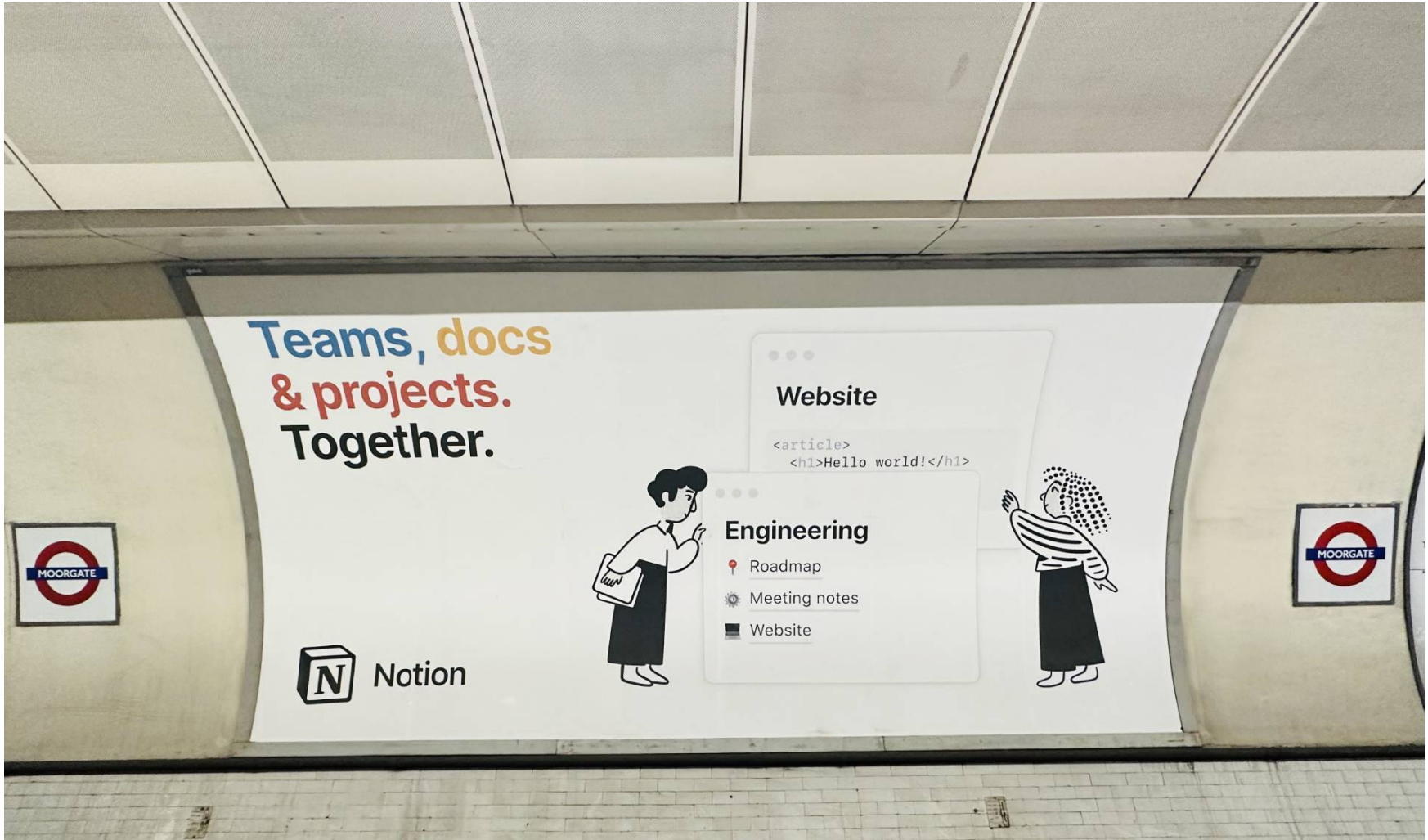
- ✅ Easy to distribute
- ✅ Easy to cross-platform
- ✅ Easy to collaborate

But also some tradeoffs…

- 😐 Network latency
- 😐 Connectivity requirement
- 😐 Data control and ownership






Teams, docs
& projects.
Together.
Website
<article>
<h1>Hello world!</h1>
Engineering
Roadmap
Meeting notes
Website
Notion
MOORGATE
MOORGATE

Progressive Improvement: The Role of PWA

A quick recap of PWA

- 📩 App shell architecture for optimal page init performance
- 🔄 Service worker for offline caching and content serving
- 💾 Local storage for offline data persistence
- 📶 Background data synchronization when back online

Do we truly need the caching mindset and a central server?

The Alternative: Local-first Architecture

Local state as single source of truth - or what if things are **local only**?

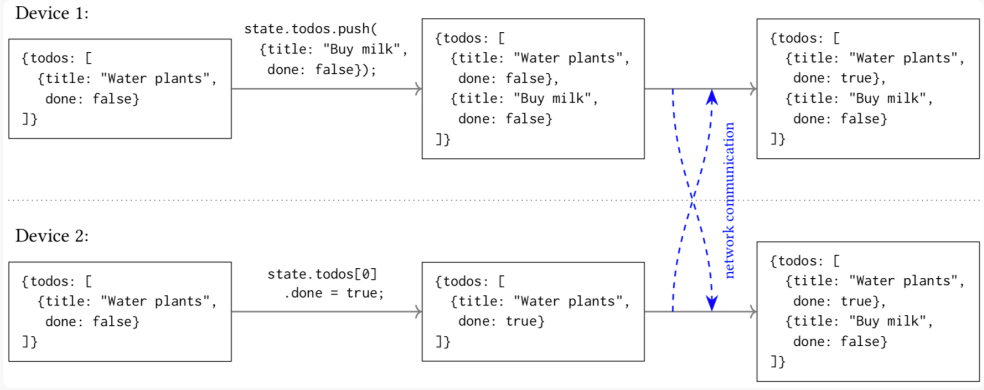
- **Complete offline functionality**: Fully usable without internet access
- **Reduced latency**: Faster data access by eliminating server roundtrips
- **Data ownership**: Users have full control over their data and storage
- **Enhanced privacy**: Data stays on the user's device, reducing exposure to third parties
- **Simplified DX**: Reduced server complexity by offloading data management to clients

But when it comes to collaboration...

CRDT: Prerequisite for local-first collaboration

Conflict-free replicated data type - What it is?

```
import * as Y from 'yjs'

// Model states can be hosted in a CRDT container
const doc = new Y.Doc()

// Different top-level YModel instances can be created
const yRoot = doc.getMap('root')

// Using class constructors
const yPoint = new Y.Map()
yPoint.set('x', 0)
yPoint.set('y', 0)

// Composing nested structure
yRoot.set('point', yPoint)

// And essential rich text support
const yName = new Y.Text()
yName.insert(0, 'Kevin')
yRoot.set('name', yName)
```



CRDT: Prerequisite for local-first collaboration

Conflict-free replicated data type - How it works?

- Recalling the classical Redux way: defining serializable actions -> *event sourcing*!
- Similar in command driven editors: defining `add_element`, `change_element`, `remove_element`…
- Working with two kinds of data: **model** and **operation** (*commands*, *actions*…).
- So when it comes to handling conflicts:
  - Transforming **operations** - **OT** (used by Google Docs, Lark, Etherpad…)
  - Making **models** conflict-free - **CRDT** (used by Figma)
- To make this happen, operation-based CRDTs essentially record all history operations
- Every operation contains `clientID` and **logical timestamp**, making it decentralized and deterministic

CRDT: Git That Doesn’t Conflict

Both git and CRDT would track the history of changes!

Lifecycle of a CRDT-based application:

1. Duplicate the "repository" (`git clone`)
2. Make changes locally (`git commit`)
3. Push changes to the "remote" (`git push`)

Differences:

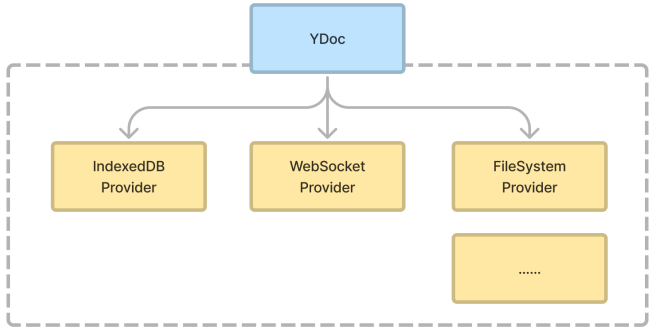
1. No need for manual `git commit`
2. No conflict on `git merge`
3. No need for manual `git pull` and `git push`

See `Y.encodeStateAsUpdate(doc: Y.Doc): Uint8Array` and y-protocols

Provider-based Persistence

Now we have encoded the model as `Uint8Array`, then let's persist and distribute it…

- CRDT model APIs are synchronous like `localStorage.setItem`, but **very fast**!
- Underlying network and database IO are asynchronous
- Data syncing works just like using git over SSH or HTTPS with `git remote add`



Some FAQs for CRDT

- What if *A blabla, B blabla, A blabla…*
  - For merge result, mathematical correctness is more important than intention keeping
  - In real-world, the conflict resolution part in Yjs is rarely used 🤫
- Encoded binaries are highly optimized and tombstone mechanism is used
- Don't put blob content here!

AFFiNE: Example of Local-first App

- Built with the **one model, multiple views** philosophy
  - Same block tree for list view, kanban view and table view
  - Smooth transition between document mode and whiteboard mode
- Local-first, privacy-first, collaboration-ready
- Extensible block-based editor based on BlockSuite
- Data persistence based on OctoBase

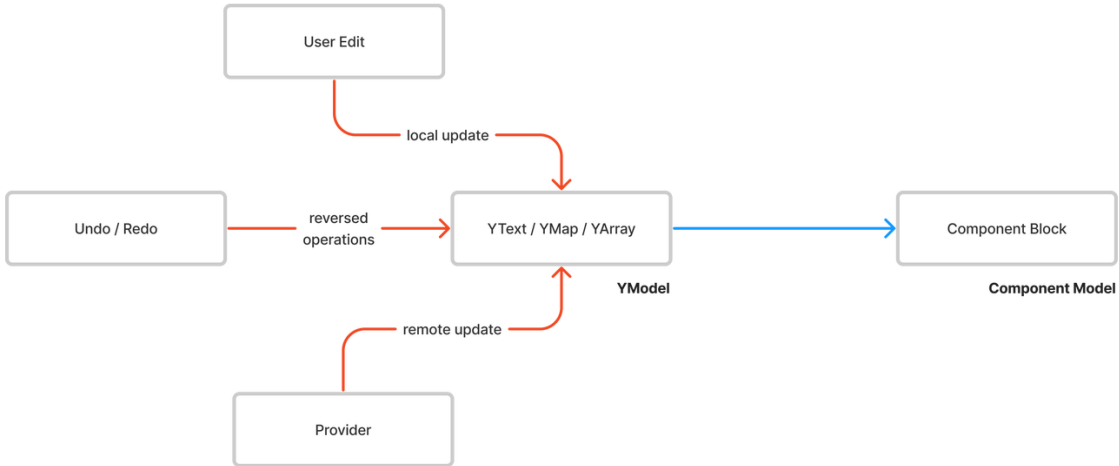
Fundamental Concepts in AFFiNE

Working with `Workspace`, `Page` and `Block`

```
import { Workspace, Page } from '@blocksuite/store';
import { AffineSchemas } from '@blocksuite/blocks/models';
import { EditorContainer } from '@blocksuite/editor';

// Create a workspace with one default page
const workspace = new Workspace({ id: 'test' }).register(AffineSchemas);
const page = workspace.createPage('page0');

// Create default blocks in the page
const pageBlockId = page.addBlock('affine:page');
const frameId = page.addBlock('affine:frame', {}, pageBlockId);
page.addBlock('affine:paragraph', {}, frameId);

// Init editor with the page store
const editor = new EditorContainer();
editor.page = page;
document.body.appendChild(editor);
```

CRDT-driven: State Management in AFFiNE

- Type-safe block tree built on top of CRDT primitives
- Always update YModel first, rather than using two-way binding
- `YEvent` triggered for all YModel updates coming from different origins
- No need to distinguish local and remote updates anymore
- See the `handleYEvents` method in BlockSuite



CRDT Outside of WebView: OctoBase

- Based on Yrs, the Rust port of Yjs, for binary compatibility
- Sending binary updates between WebView and native process
- Do searching and cross-page content analysing in native environment
- SQLite and Postgres persistence support
- Plug-n-play in AFFiNE

We will advocate this infra in the future, stay tuned!

New Challenges

- High-level data schema and consistency
- Content migration and forward compatibility
- Content streaming

Recap

- Local-first app takes the advantages of both local and web apps
- CRDT is the key to local-first collaboration
- Incremental adoption of local-first features is practical

Hope to see more in the future!