开源社
kaiyuanshe

EightFish: A MVC Framework for Decentralized App

Mike Tang EightFish.org/ RustCC

The 8th China Open Source Conference (COSCon 20

EightFish

A MVC Framework for Decentralized App

Mike Tang
2023-10

What is EightFish

EightFish is a development framework for decentralized application with powerful performance and easy experience.

Why another framework

There are many blockchain development frameworks, but no good Web3 frameworks. Merely blockchain framework is not enough for the Web3 world.

Substrate is a blockchain framework

Use Substrate, we can build a blockchain network rapidly and easily.

But not enough for Web3.0 App/Dapp.

The daily web app requires:

1. high performance
2. cheap storage
3. powerful query capability
4. working with existing databases

The Substrate off-chain worker

Story starts from the Substrate OCW (off-chain worker)

What OCW offers.

- Ability to submit transactions—either signed or unsigned—to the chain to publish computation results.
- A fully-featured HTTP client allowing the worker to access and fetch data from external services.
- Access to the local keystore to sign and verify statements or transactions.
- An additional, local key-value database shared between all offchain workers.
- A secure, local entropy source for random number generation.
- Access to the node's precise local time.
- The ability to sleep and resume work.

& Forkless upgrade.

The limits of Substrate OCW

- The interval stream programming pattern: be called at every block imported, it is hard to master, and should be careful to treat
- No execution when in sync state
- Can not spawn new task manually and concurrently
- Hard to interact with sql databases (only http interface right now)

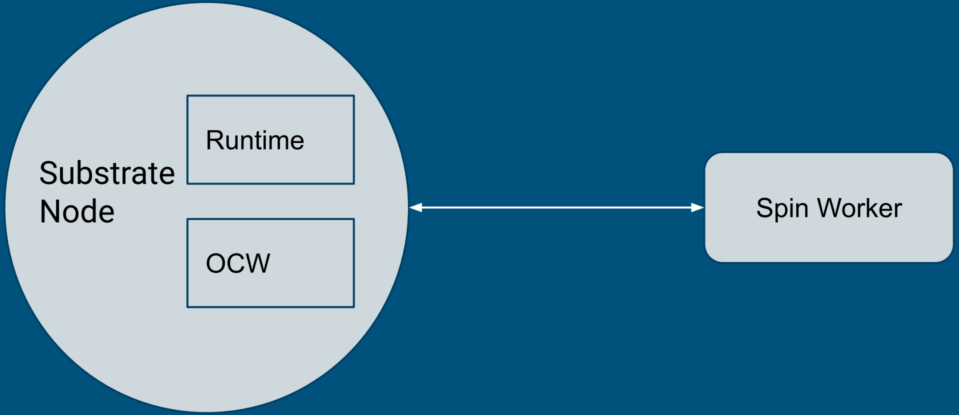
While the OCW plays its important role in Substrate ecosystem, we need another flexible and powerful solution to extend the ability of Substrate.

How to improve it

How to do？

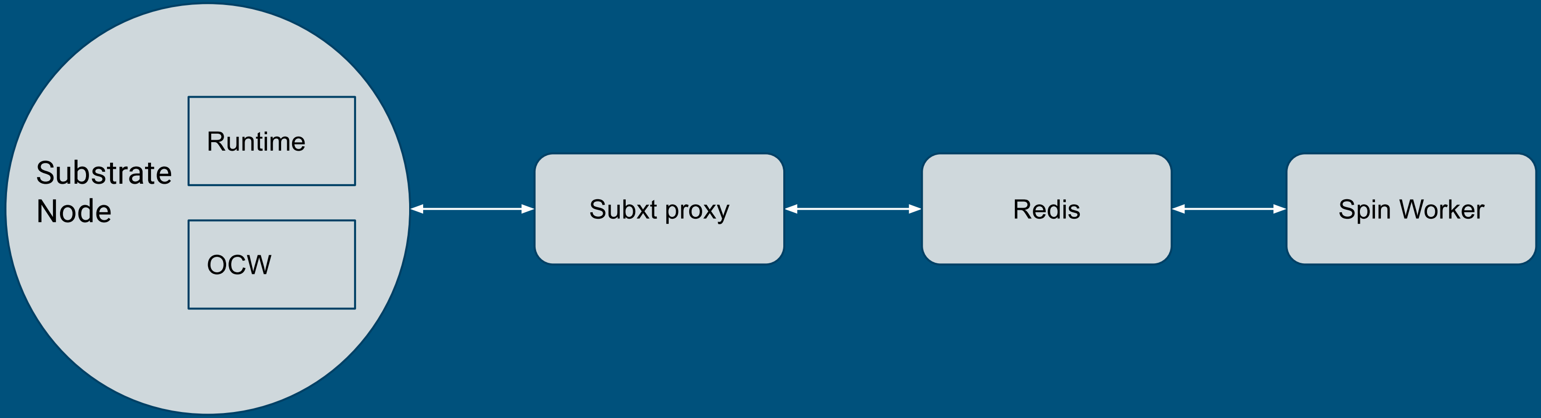
Step 1: move OCW outside

Use the spin wasm runtime to serve as the off-chain work, listening to the events emitted from the Substrate Runtime.

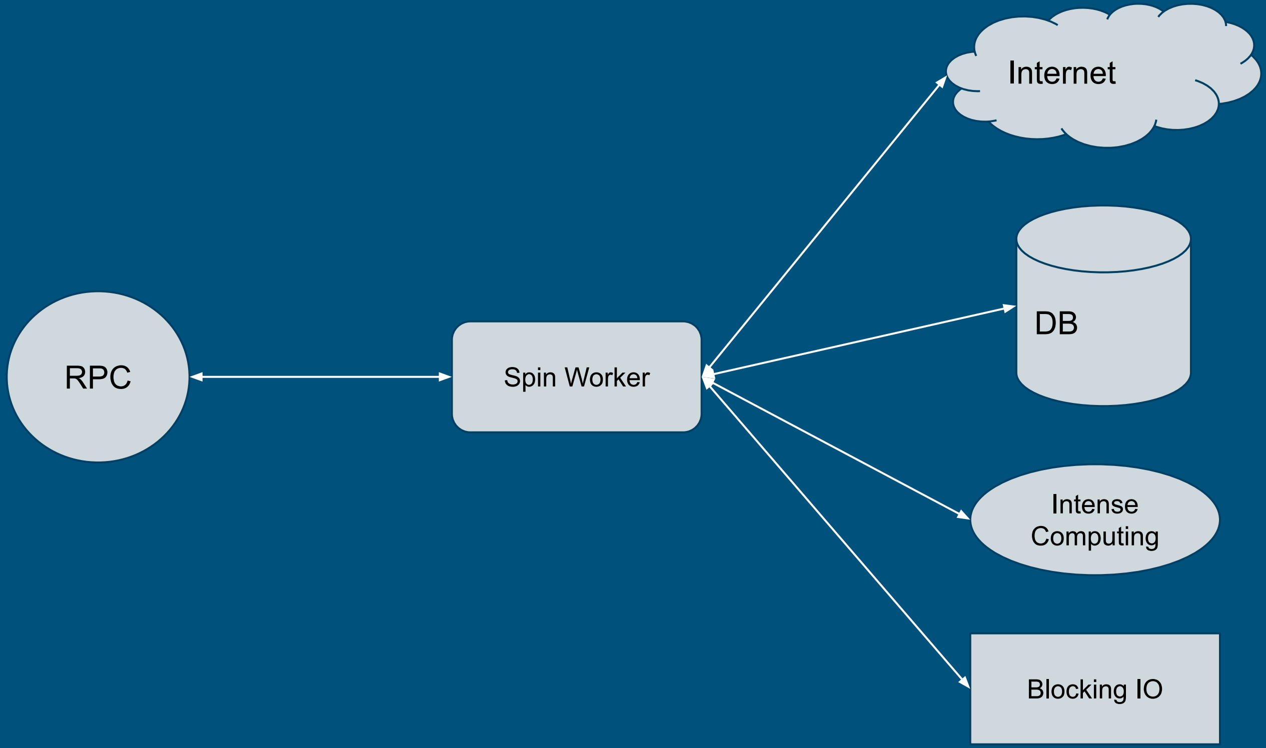


https://www.fermyon.com/spin

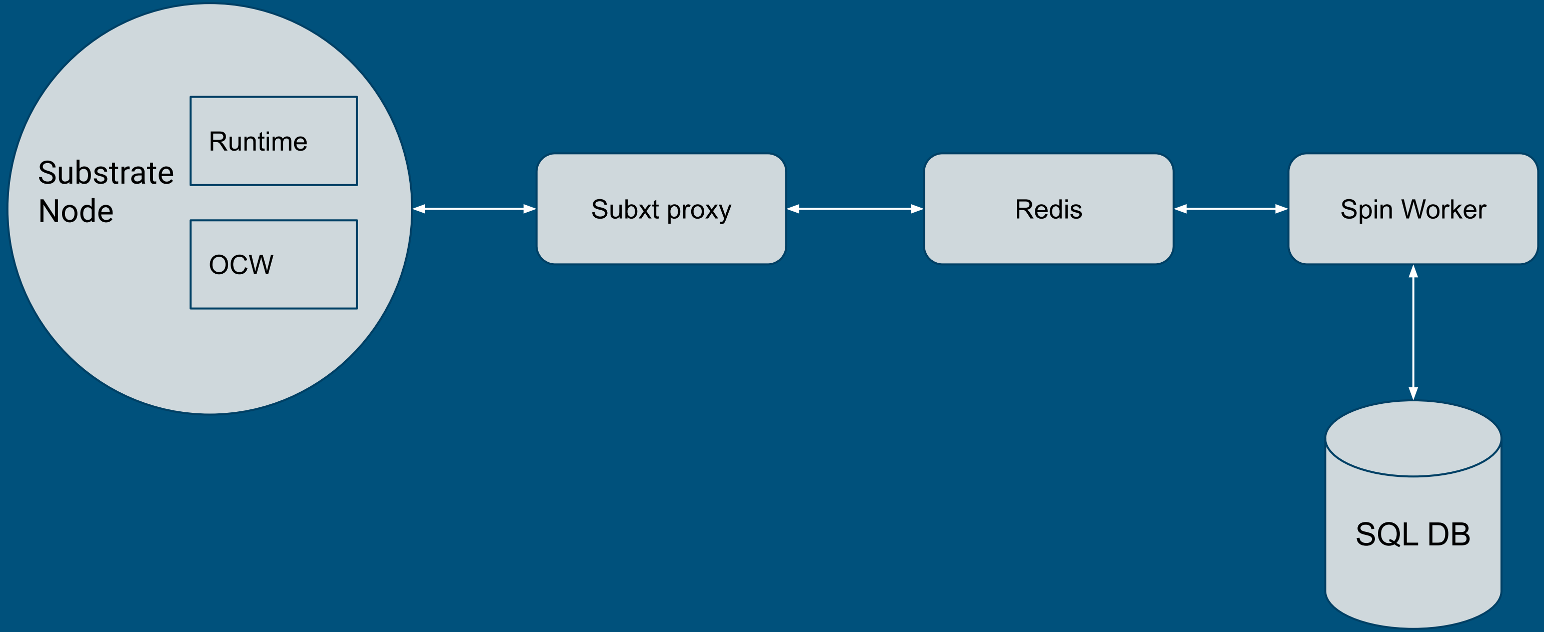
Technical requirements



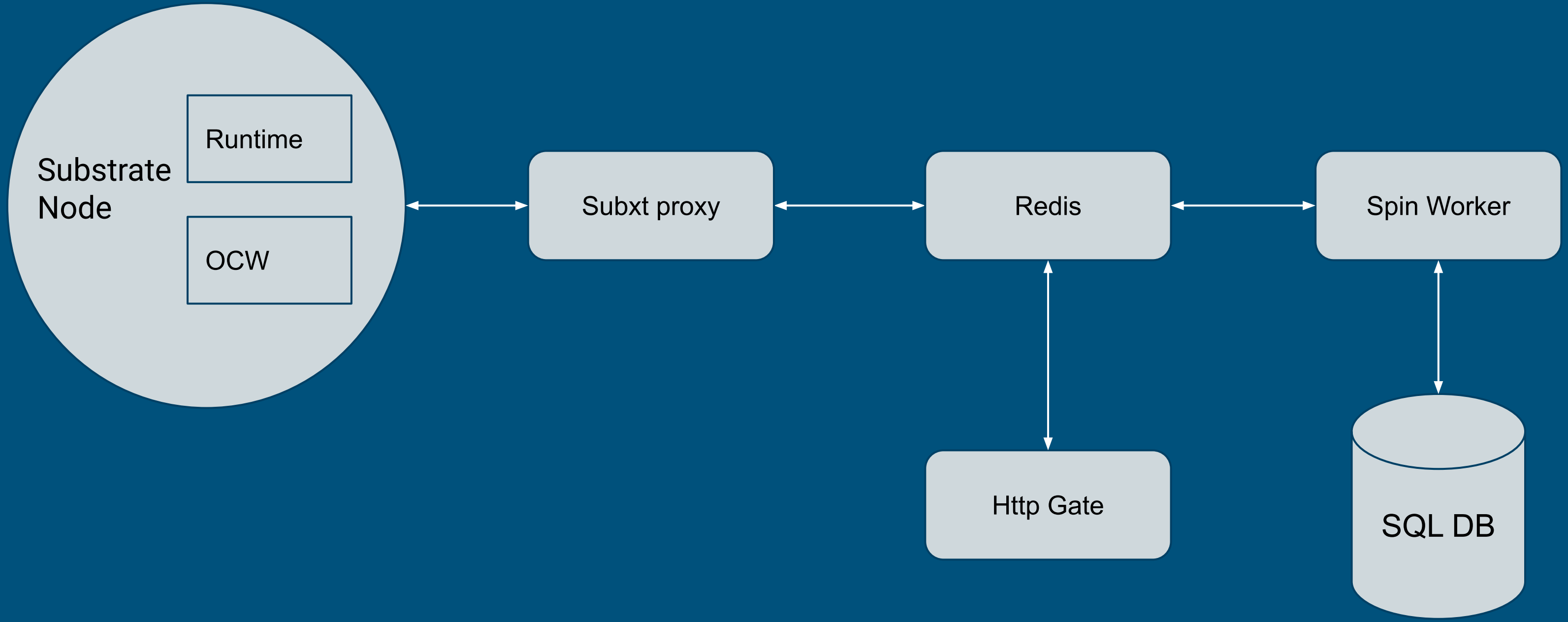
Spin worker: an enhanced OCW



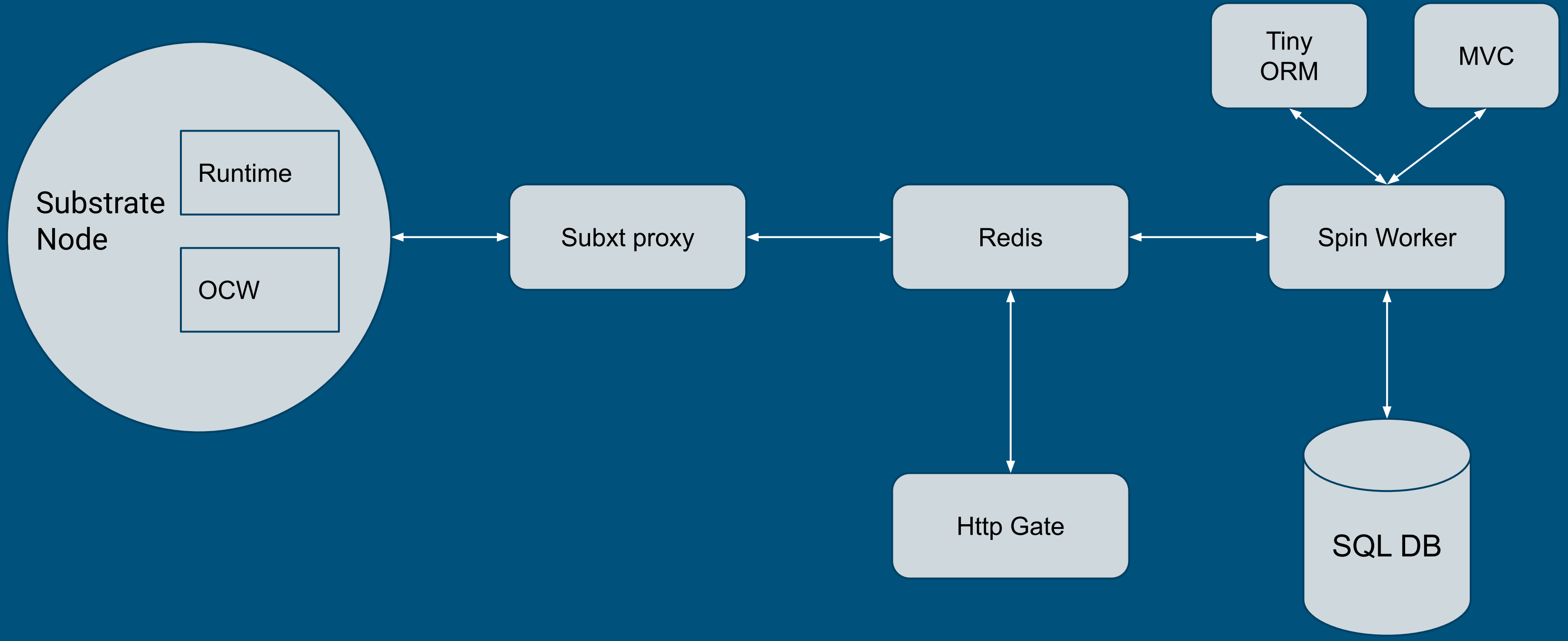
Step 2: make it more useful



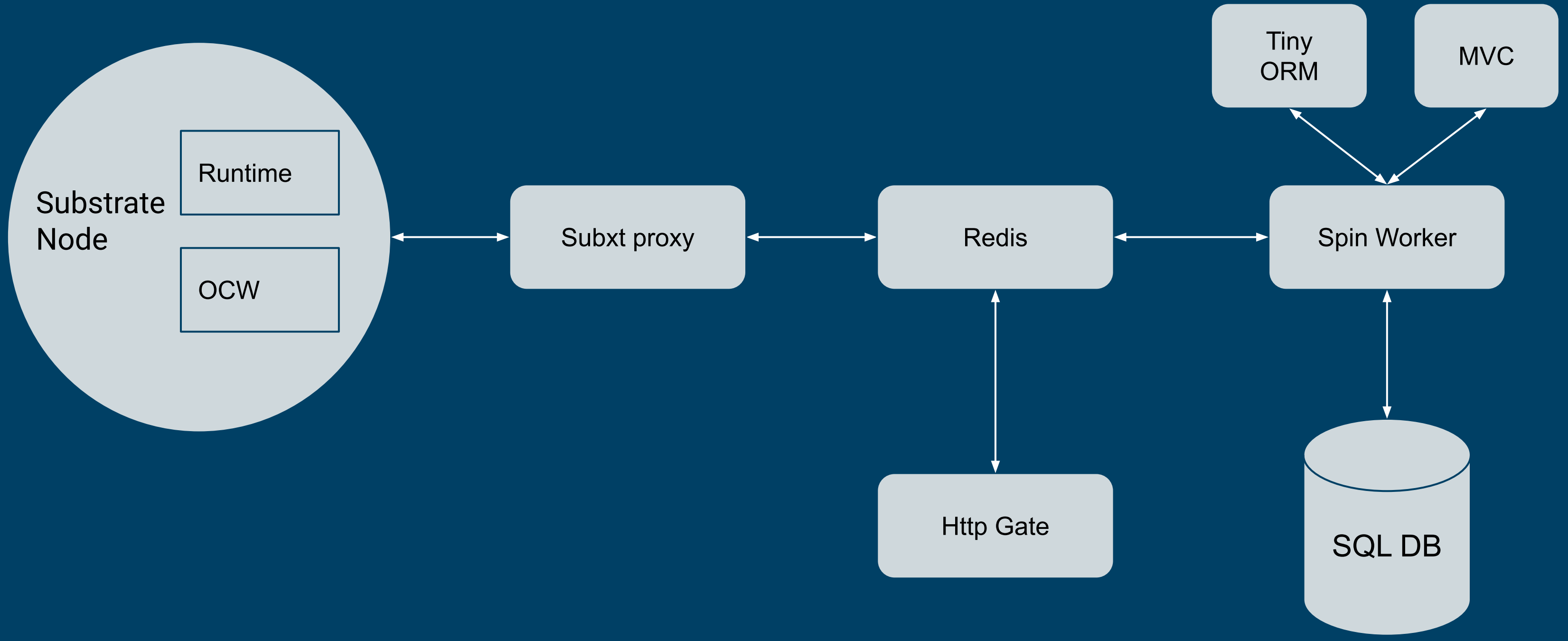
Step 3: make it further useful



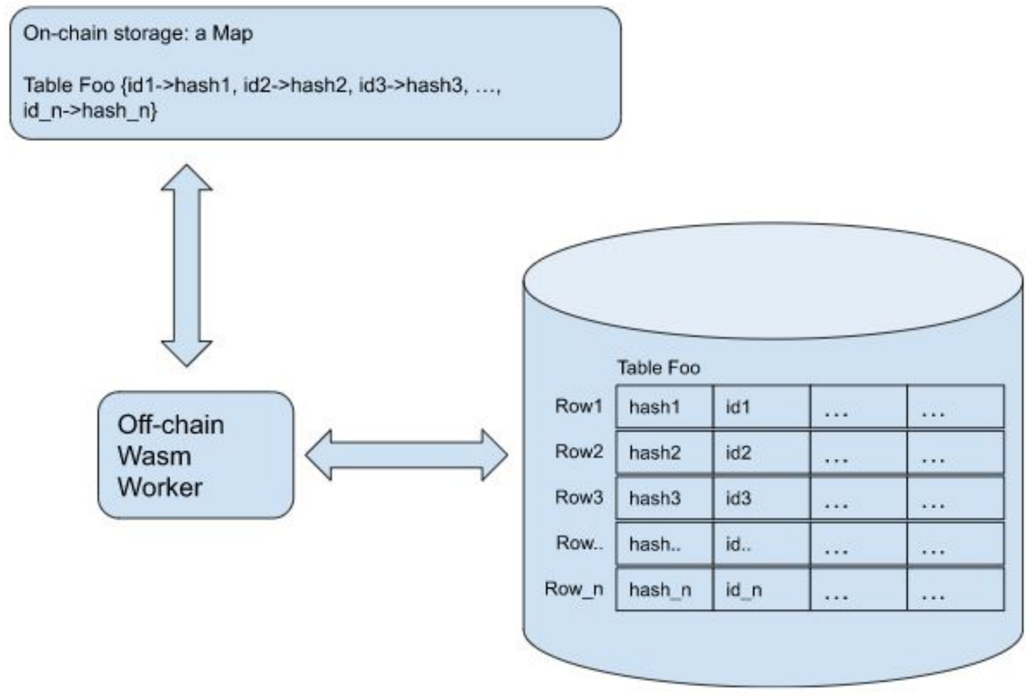
Step 4: make dev more comfortable



EightFish Framework Architecture



Problem transformation



On-chain storage: a Map

Table Foo {id1->hash1, id2->hash2, id3->hash3, …, id_n->hash_n}

Table Foo

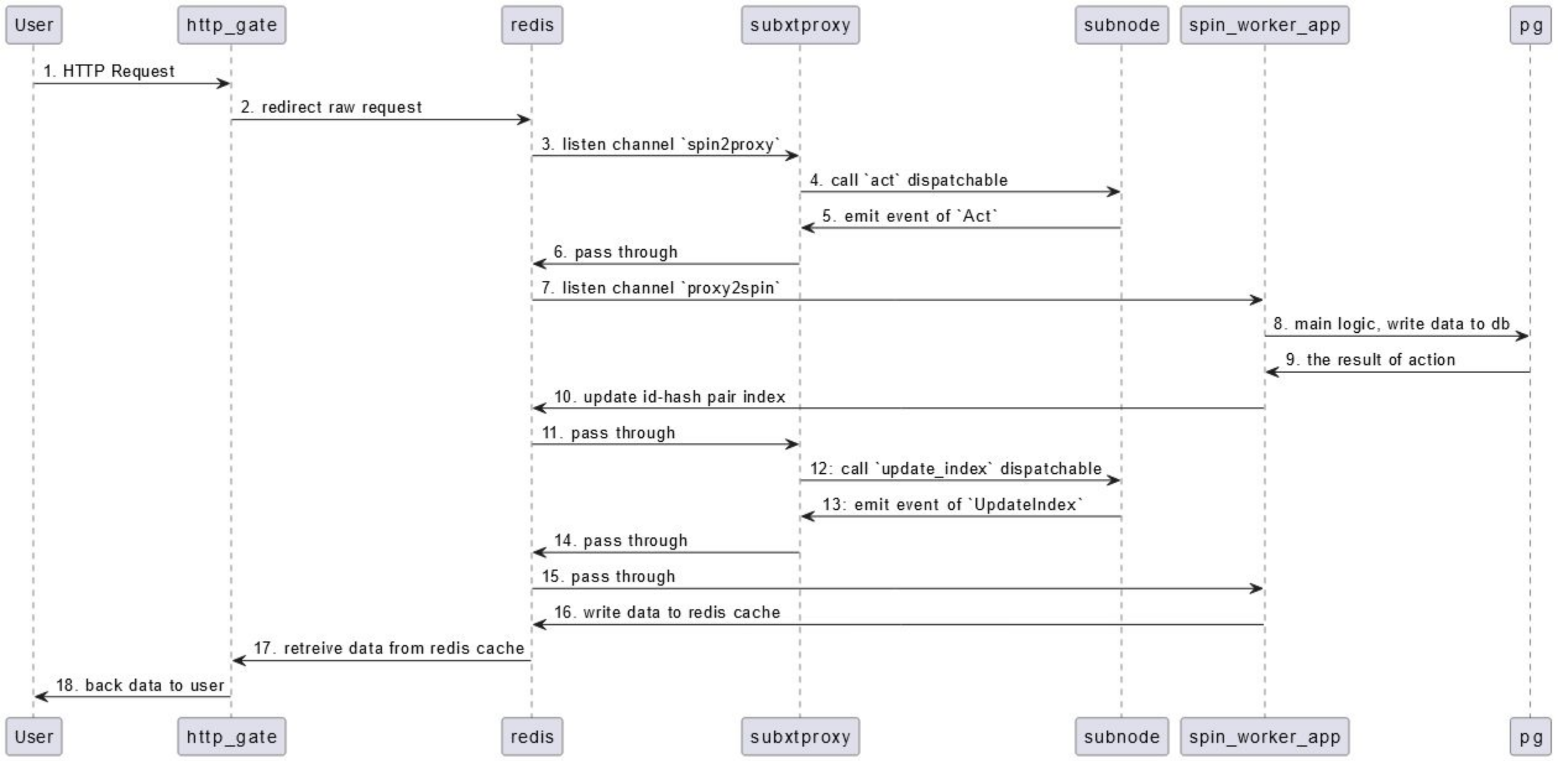
Row1	hash1	id1	…	…
Row2	hash2	id2	…	…
Row3	hash3	id3	…	…
Row..	hash..	id..	…	…
Row_n	hash_n	id_n	…	…

The mapping between on-chain storage items and the SQL db table

Standardization of the on-chain index

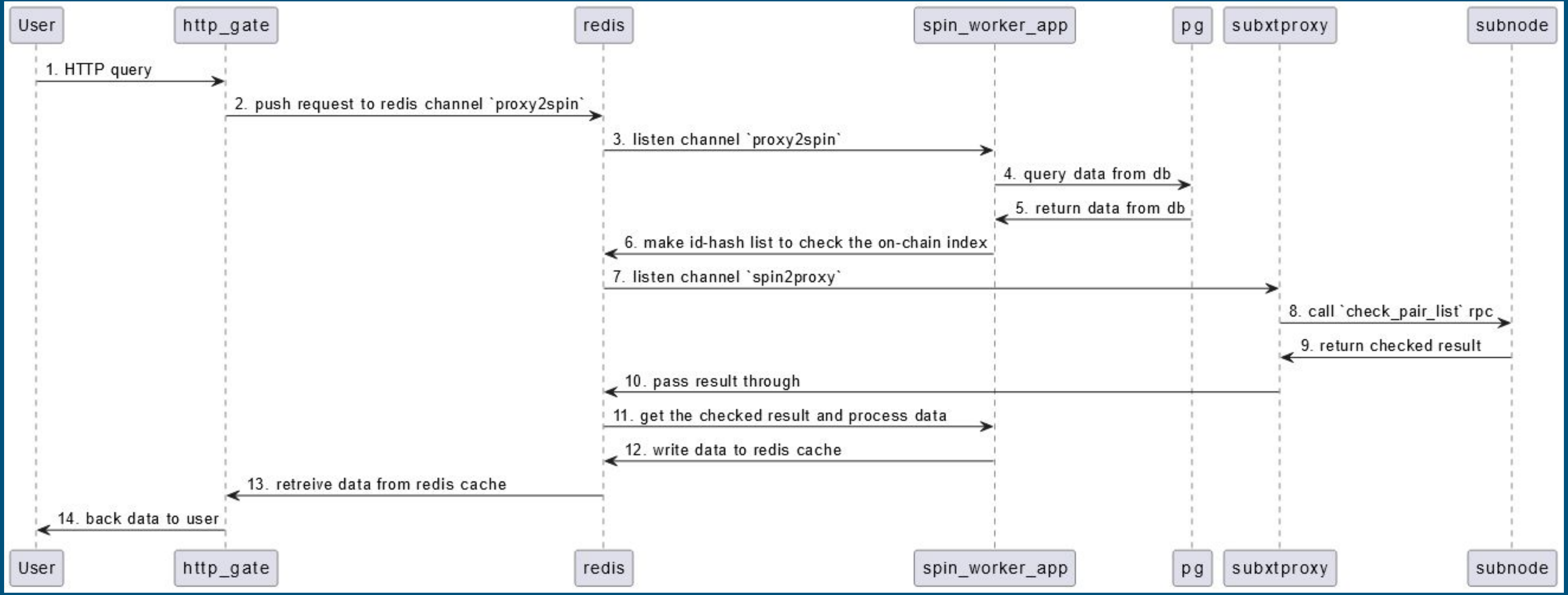
- EightFish actually makes the on-chain index process standardized
- Underlying is the theory of the relational database.

Post process

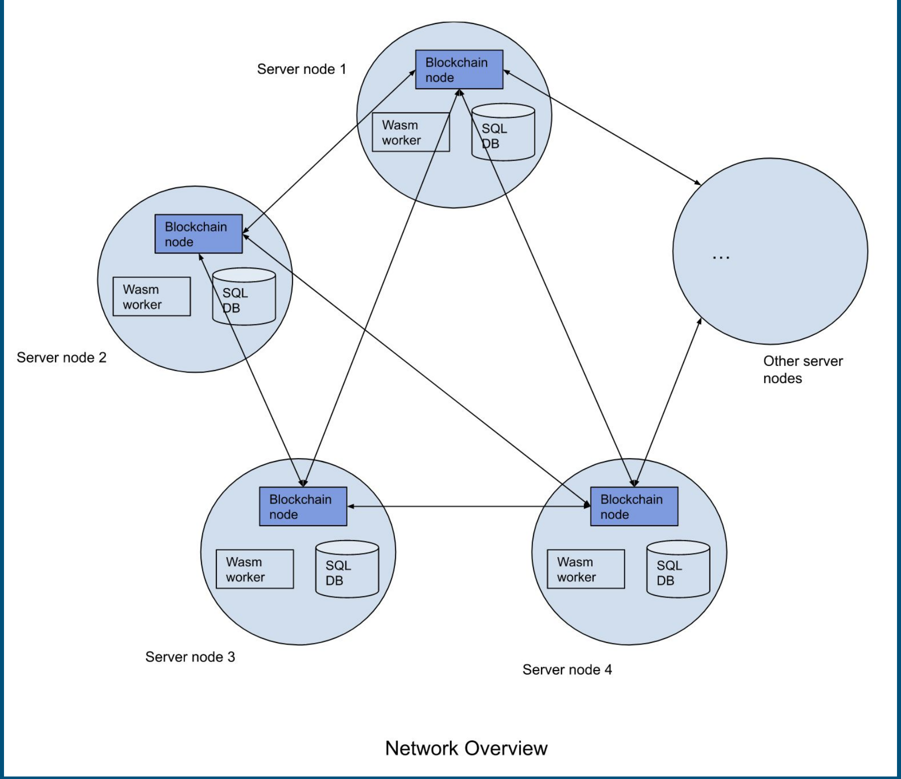


1. HTTP Request
2. redirect raw request
3. listen channel `spin2proxy`
4. call `act` dispatchable
5. emit event of `Act`
6. pass through
7. listen channel `proxy2spin`
8. main logic, write data to db
9. the result of action
10. update id-hash pair index
11. pass through
12: call `update_index` dispatchable
13: emit event of `UpdateIndex`
14. pass through
15. pass through
16. write data to redis cache
17. retreive data from redis cache
18. back data to user

Query process


User
http_gate
redis
spin_worker_app
pg
subxtproxy
subnode
1. HTTP query
2. push request to redis channel `proxy2spin`
3. listen channel `proxy2spin`
4. query data from db
5. return data from db
6. make id-hash list to check the on-chain index
7. listen channel `spin2proxy`
8. call `check_pair_list` rpc
9. return checked result
10. pass result through
11. get the checked result and process data
12. write data to redis cache
13. retreive data from redis cache
14. back data to user
User
http_gate
redis
spin_worker_app
pg
subxtproxy
subnode

EightFish app network



Network Overview

Explanation

- Sync the SQL db tables and rows by through the Substrate node
- All components except Substrate node are local, no connections to others EightFish node, so it’s yet a Substrate network
- EightFish could leverage all major outstanding databases. e.g. PostgreSQL, MySQL, Sqlite, even mongodb.

The role of Substrate in EightFish

1. Used to record the incoming raw writing requests, bake them into blocks, like a log system
2. Used to sync runtime state among all EightFish nodes (Substrate network), further to coordinate the state of the SQL db among all nodes
3. Used to store the version of wasm code and make the code of spin worker upgrade forklessly
4. Used to interoperate with other Substrate-based chains by leveraging existing pallets

Some limites of Spin Worker

- Should not use local random number, but certain type of deterministic random number from the Substrate node
- Should not use local timestamp, but the timestamp from the Substrate node
- Can not do the type of operations of SQL JOIN family, because we must check the query result from local SQL db with the on-chain index internally in EightFish, which is unaware by developers. But it supports full-fledged single SQL table query

Yet EightFish is powerful.

So what is EightFish for

- Non-assets/xFi application
- Fully on-chain application, e.g. fully on-chain game
- Fully decentrialized application
- General computation/CPU intensive computation

How to use EightFish

1. Use EightFish MVC framework to write your app service;
2. Modify a docker compose template to boot up the dependencies of 5 EightFish components as services + your app service

MVC doc:

https://github.com/eightfish-org/eightfish/blob/master/docs/eightfish_mvc_intro.md

ORM doc:

https://github.com/eightfish-org/eightfish/blob/master/eightfish-derive/src/eight_fish_model.rs#L134

docker compose file:

https://github.com/eightfish-org/eightfish/blob/master/docker/docker-compose-1node.yml

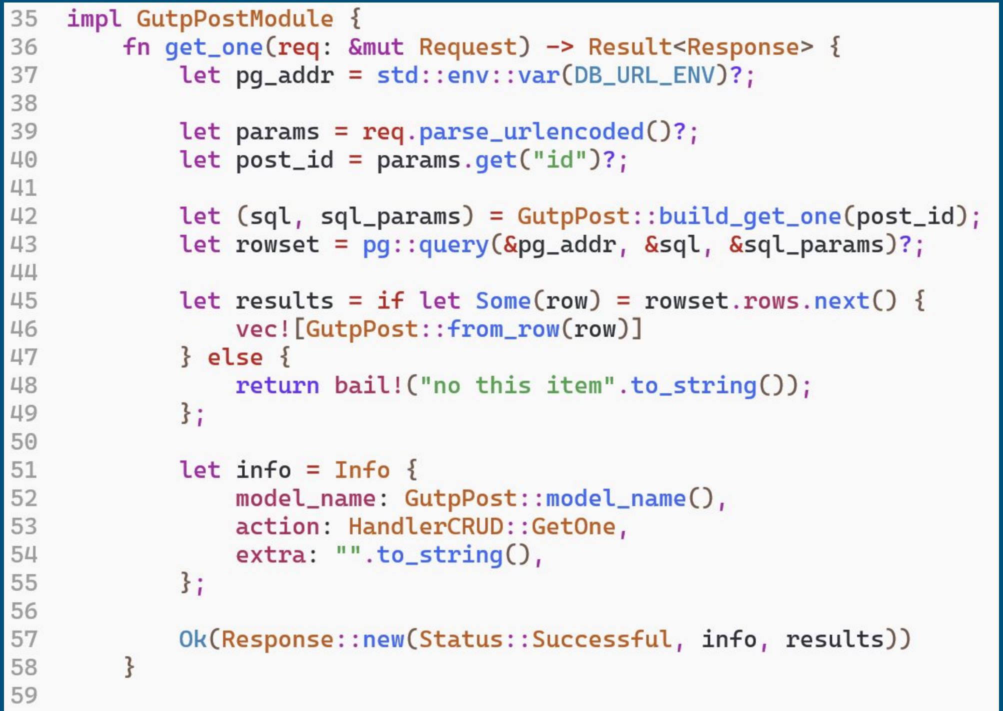
How to deploy

Use docker compose to deploy a network, example:

https://github.com/eightfish-org/eightfish/blob/master/docker/docker-compose-4node.yml

Just similar with the standard Substrate chain network.

The business code demo



```rust
impl GutpPostModule {
    fn get_one(req: &mut Request) -> Result<Response> {
        let pg_addr = std::env::var(DB_URL_ENV)?;

        let params = req.parse_urlencoded()?;
        let post_id = params.get("id")?;

        let (sql, sql_params) = GutpPost::build_get_one(post_id);
        let rowset = pg::query(&pg_addr, &sql, &sql_params)?;

        let results = if let Some(row) = rowset.rows.next() {
            vec![GutpPost::from_row(row)]
        } else {
            return bail!("no this item".to_string());
        };

        let info = Info {
            model_name: GutpPost::model_name(),
            action: HandlerCRUD::GetOne,
            extra: "".to_string(),
        };

        Ok(Response::new(Status::Successful, info, results))
    }

```

https://github.com/eightfish-org/gutp/blob/master/gutp/src/post.rs#L35

For whom?

EightFish is developed for:

- Web developers (Web2)
- Substrate developers
- Other Web3 developers

One case of EightFish app

GUTP & Meblog, work in progress

https://github.com/eightfish-org/gutp

https://github.com/miketang84/meblog

Could be publicly tested at the beginning of June.

Q&A

Email: daogangtang@gmail.com

Twitter: @daogangtang

开源社
kaiyuanshe

THANK YOU

QUESTIONS?

欢迎扫码打卡
积分可兑换对应礼品哟！

微信公众号：开源社KAIYUANSHE

视频号：开源社KAIYUANSHE

新浪微博：开源社

B站：开源社KAIYUANSHE

简书：开源社

头条：开源社

Facebook：KaiyuansheChina

Twitter：开源社KAIYUANSHE

扫码关注 开源社 公众号

扫码添加讲师联系方式