How I built a high-performance Cosmos indexer

Fabien Penso

@fabienpenso
https://pen.so

DEVMOS 2024

Constellations

01	Time Spent writing indexers	2 years full-time, 3500h
02	Project size	77,000 lines of Rust
03	Performance	Indexing time from genesis: Kujira in 1h, Osmosis in 6h, Stargaze in 3h

Constellations

04	Usage	Stargaze: > 15M requests per day
05	Indexed chains	Stargaze, Osmosis, Neutron, Noble, dYdX, Kujira, ...
06	Infrastructure	Stargaze: 4 redundant servers Kujira, Osmosis, Neutron, Noble, dYdX: 1 single server

Feedback

Thank you @fabienpenso for Constellations❤️ @IBCMuffins

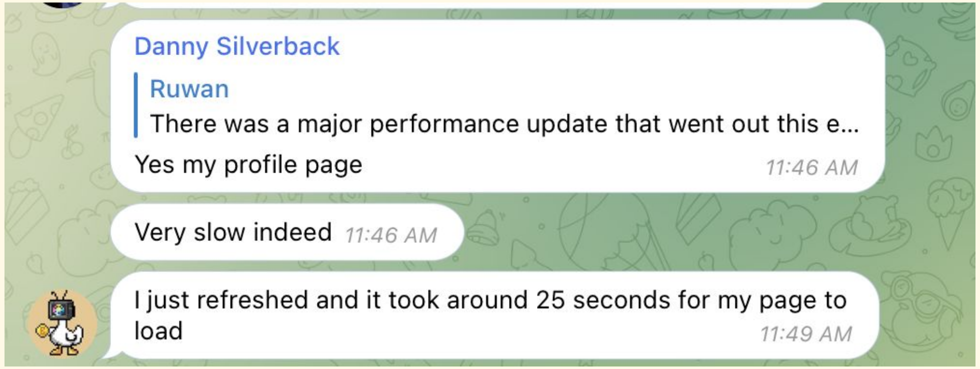
Constellations indexer [...] singlehandedly sped up development [...] threefold. @josefleventon

... and many using the public API

Stargaze: Benefits from no indexer to using an indexer?

No Indexer

- Front-end fetch lots of data directly from the chain
- Some pages took > 30 seconds. Stargaze launchpad failed when too much content was loaded. Not sustainable.



Danny Silverback

Ruwan

There was a major performance update that went out this e...

Yes my profile page 11:46 AM

Very slow indeed 11:46 AM

I just refreshed and it took around 25 seconds for my page to load 11:49 AM

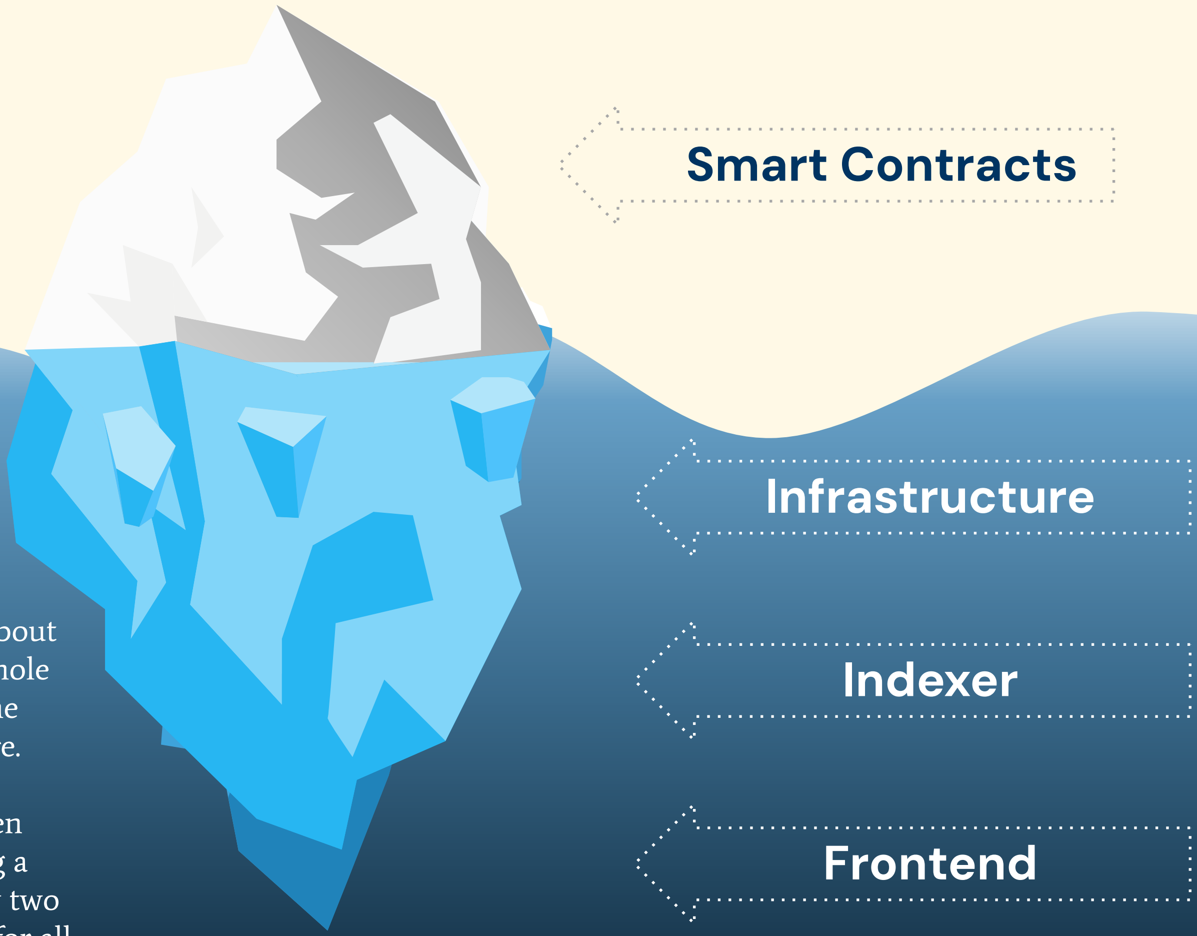
With Indexer

Page loaded < 500ms

Every chain should have an indexer

Users expect web3 products with web2 performance

Blockchain launch



A lot of blockchain launches are all and only about smart contracts, but that's a tiny 10% of the whole shebang. The remaining 90% comes from the frontend, the backend, and the infrastructure.

The significance of these components is often overlooked, yet they're crucial for developing a high-quality product. It requires approximately two years to reach the level of Stargaze, accounting for all these elements.

Prepare for Chaos

How to build an indexer?

Language

Rust

- ✓ Memory safety (multi-thread)
- ✓ Most loved language 8 years in a row (Stack Overflow)
- ✓ Performance
- ✓ Functional programming

But that's not enough, lots of time spent tweaking performance, making things faster, adding caching layers and instrumentation

Make it work
Make it robust
Make it fast

Naive way to write indexers

1

Fetch

Fetch all blocks and block results from RPC nodes

2

Store

Store all blocks, transactions, messages, events into SQL

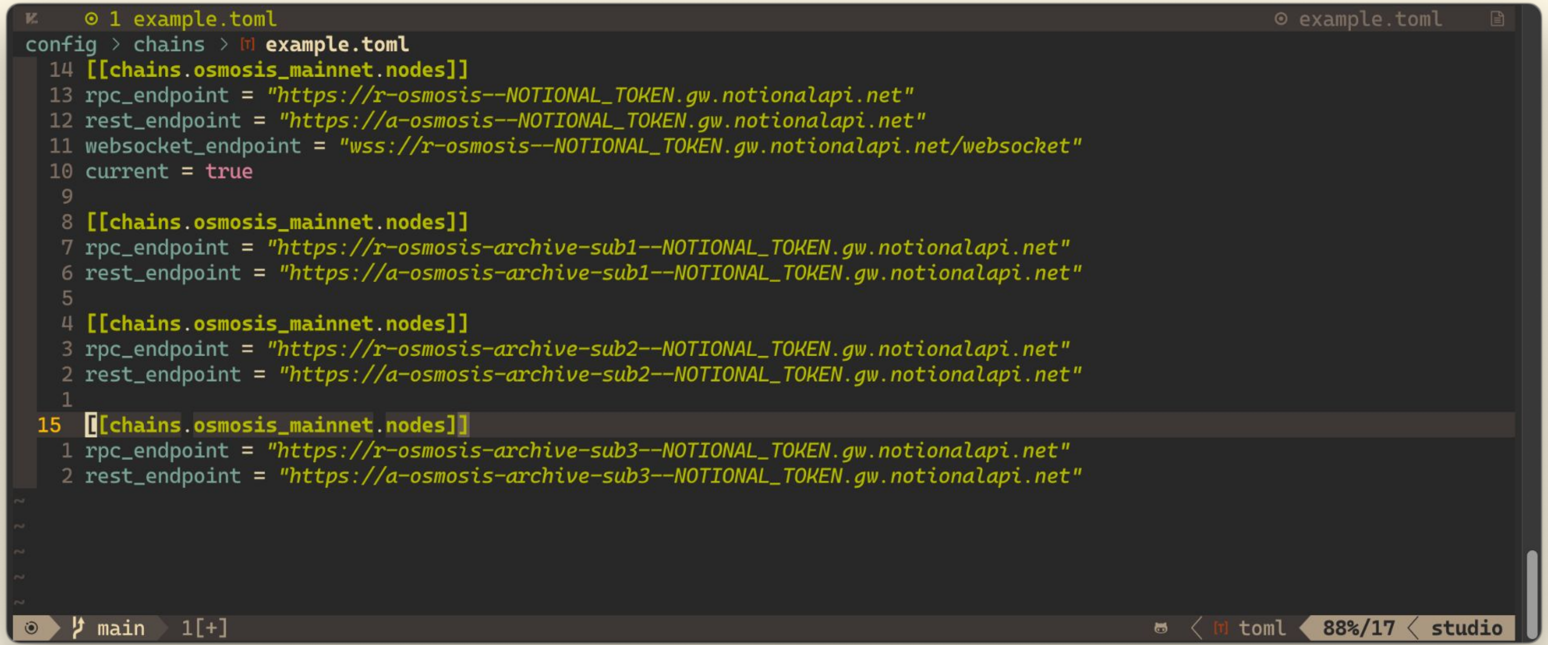
3

Parse

Go through messages and events, apply state changes

Fetching blocks

Configuration

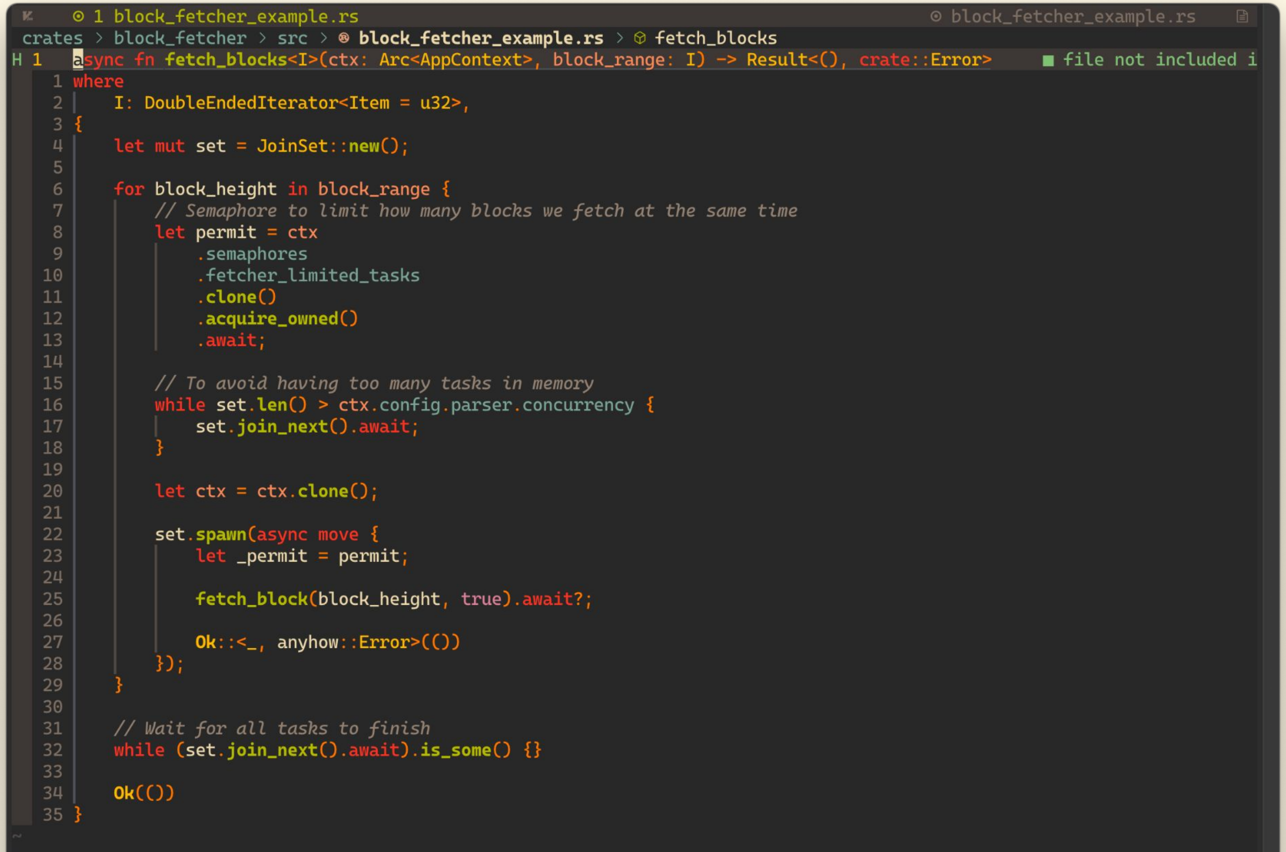


```
⊙ 1 example.toml                                                    ⊙ example.toml
config > chains > example.toml
[[chains.osmosis_mainnet.nodes]]
rpc_endpoint = "https://r-osmosis--NOTIONAL_TOKEN.gw.notionalapi.net"
rest_endpoint = "https://a-osmosis--NOTIONAL_TOKEN.gw.notionalapi.net"
websocket_endpoint = "wss://r-osmosis--NOTIONAL_TOKEN.gw.notionalapi.net/websocket"
current = true

[[chains.osmosis_mainnet.nodes]]
rpc_endpoint = "https://r-osmosis-archive-sub1--NOTIONAL_TOKEN.gw.notionalapi.net"
rest_endpoint = "https://a-osmosis-archive-sub1--NOTIONAL_TOKEN.gw.notionalapi.net"

[[chains.osmosis_mainnet.nodes]]
rpc_endpoint = "https://r-osmosis-archive-sub2--NOTIONAL_TOKEN.gw.notionalapi.net"
rest_endpoint = "https://a-osmosis-archive-sub2--NOTIONAL_TOKEN.gw.notionalapi.net"

[[chains.osmosis_mainnet.nodes]]
rpc_endpoint = "https://r-osmosis-archive-sub3--NOTIONAL_TOKEN.gw.notionalapi.net"
rest_endpoint = "https://a-osmosis-archive-sub3--NOTIONAL_TOKEN.gw.notionalapi.net"

main  1[+]                                                  toml  88%/17  studio
```

Using RPC nodes from **Rhino** (https://rhinostake.com/), **Notional** (github.com/notional-labs) or provided by the chain. Managing nodes can be a full-time work... Waiting for weeks for an archive node from a chain.

Fetching blocks

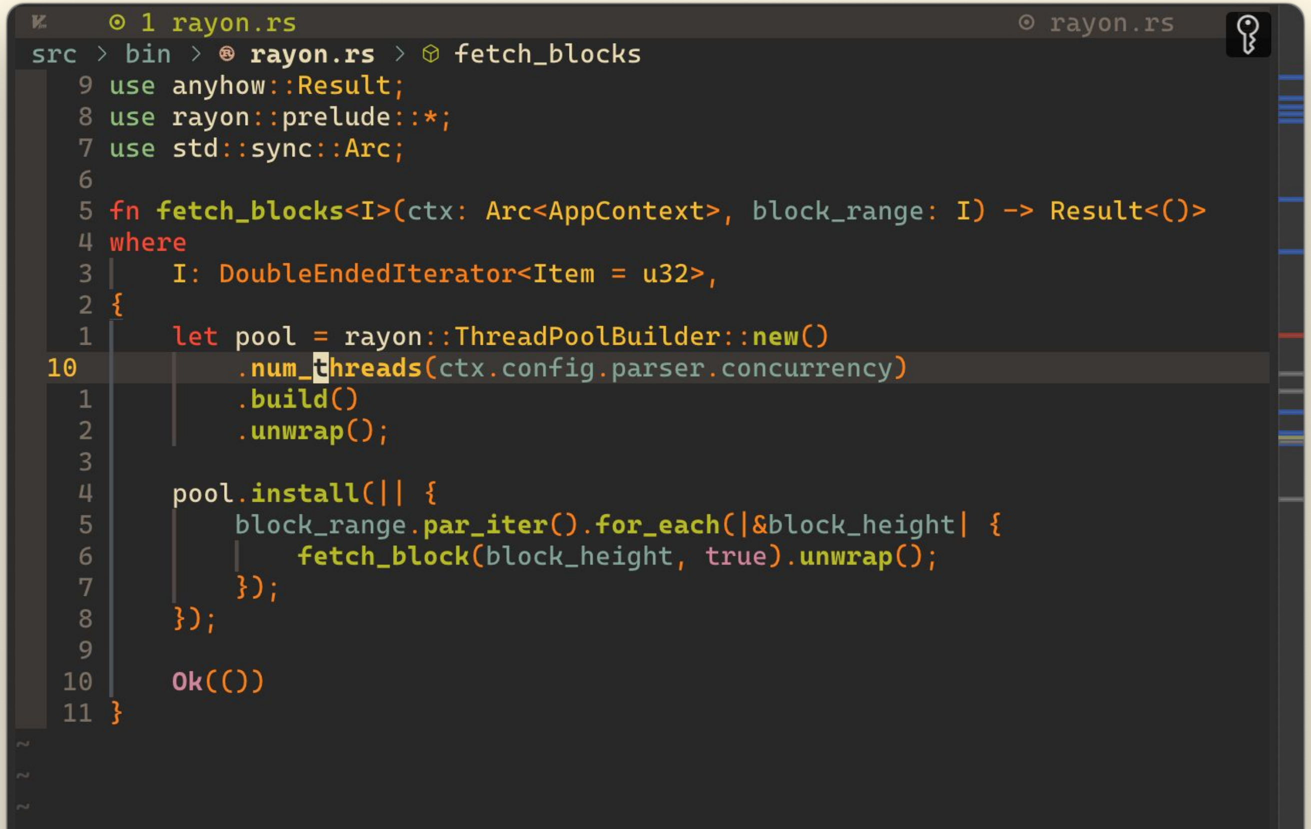
Async with Tokio



```
⊙ 1 block_fetcher_example.rs                                        ⊙ block_fetcher_example.rs
crates > block_fetcher > src > block_fetcher_example.rs > fetch_blocks
H 1 async fn fetch_blocks<I>(ctx: Arc<AppContext>, block_range: I) -> Result<(), crate::Error>     ■ file not included i
where
    I: DoubleEndedIterator<Item = u32>,
{
    let mut set = JoinSet::new();

    for block_height in block_range {
        // Semaphore to limit how many blocks we fetch at the same time
        let permit = ctx
            .semaphores
            .fetcher_limited_tasks
            .clone()
            .acquire_owned()
            .await;

        // To avoid having too many tasks in memory
        while set.len() > ctx.config.parser.concurrency {
            set.join_next().await;
        }

        let ctx = ctx.clone();

        set.spawn(async move {
            let _permit = permit;

            fetch_block(block_height, true).await?;

            Ok::<_, anyhow::Error>(())
        });
    }

    // Wait for all tasks to finish
    while (set.join_next().await).is_some() {}

    Ok(())
}
```

Fetching blocks

Multi-thread with Rayon



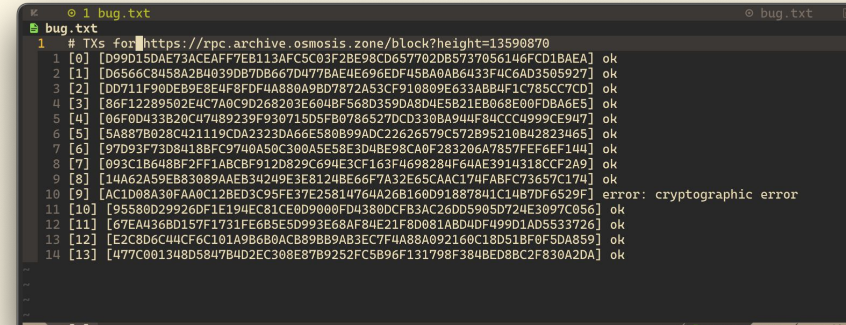
⊙ 1 rayon.rs ⊙ rayon.rs

src > bin > rayon.rs > fetch_blocks

```rust
use anyhow::Result;
use rayon::prelude::*;
use std::sync::Arc;

fn fetch_blocks<I>(ctx: Arc<AppContext>, block_range: I) -> Result<()>
where
    I: DoubleEndedIterator<Item = u32>,
{
    let pool = rayon::ThreadPoolBuilder::new()
        .num_threads(ctx.config.parser.concurrency)
        .build()
        .unwrap();

    pool.install(|| {
        block_range.par_iter().for_each(|&block_height| {
            fetch_block(block_height, true).unwrap();
        });
    });

    Ok(())
}
```

Encountered issues

- ✓ Very slow archive nodes
- ✓ 502 timeouts, 429 rate limiting, nodes down for hours
- ✓ Secp256k1 public keys with invalid SEC1 tags are accepted (cosmos-sdk issue #20406) by the go SDK, refused by CosmRS
- ✓ Invalid `txs_results` returned for legacy ABCI responses (CometBFT issue #3002), preventing fetching some dYdX/Sei block results
- ✓ Blank validator keys
- ✓ Some Osmosis block results can be > 280MB, and node is failing

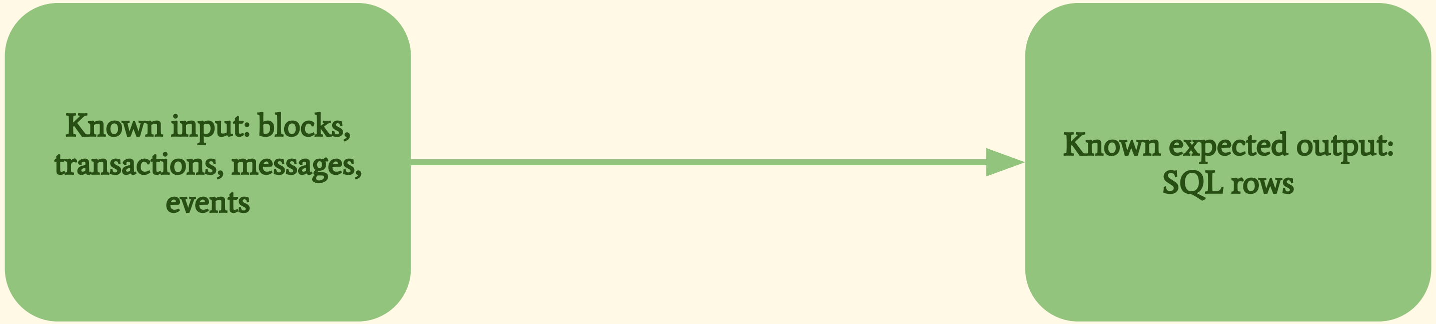


```
# TXs for https://rpc.archive.osmosis.zone/block?height=13590870
[0] [D99D15DAE73ACEAFF7EB113AFC5C03F2BE98CD657702DB5737056146FCD1BAEA] ok
[1] [D6566C8458A2B4039DB7DB667D477BAE4E696EDF45BA0AB6433F4C6AD3505927] ok
[2] [DD711F90DEB9E8E4F8FDF4A880A9BD7872A53CF910809E633ABB4F1C785CC7CD] ok
[3] [86F12289502E4C7A0C9D268203E604BF568D359DA8D4E5B21EB068E00FDBA6E5] ok
[4] [06F0D433B20C47489239F930715D5FB0786527DCD330BA944F84CCC4999CE947] ok
[5] [5A887B028C421119CDA2323DA66E580B99ADC22626579C572B95210B42823465] ok
[6] [97D93F73D8418BFC9740A50C300A5E58E3D4BE98CA0F283206A7857FEF6EF144] ok
[7] [093C1B648BF2FF1ABCBF912D829C694E3CF163F4698284F64AE3914318CCF2A9] ok
[8] [14A62A59EB83089AAEB34249E3E8124BE66F7A32E65CAAC174FABFC73657C174] ok
[9] [AC1D08A30FAA0C12BED3C95FE37E25814764A26B160D91887841C14B7DF6529F] error: cryptographic error
[10] [95580D29926DF1E194EC81CE0D9000FD4380DCFB3AC26DD5905D724E3097C056] ok
[11] [67EA436BD157F1731FE6B5E5D993E68AF84E21F8D081ABD4DF499D1AD5533726] ok
[12] [E2C8D6C44CF6C101A9B6B0ACB89BB9AB3EC7F4A88A092160C18D51BF0F5DA859] ok
[13] [477C001348D5847B4D2EC308E87B9252FC5B96F131798F384BED8BC2F830A2DA] ok
```

You're dealing with on-chain and off-chain data, you can't trust any of it. Malformed user submitted UTF8 strings, renamed smart contracts events, null bytes breaking Postgres, wrong smart contract address, invalid base64. It's the wild west.

Must build for errors and resilience.

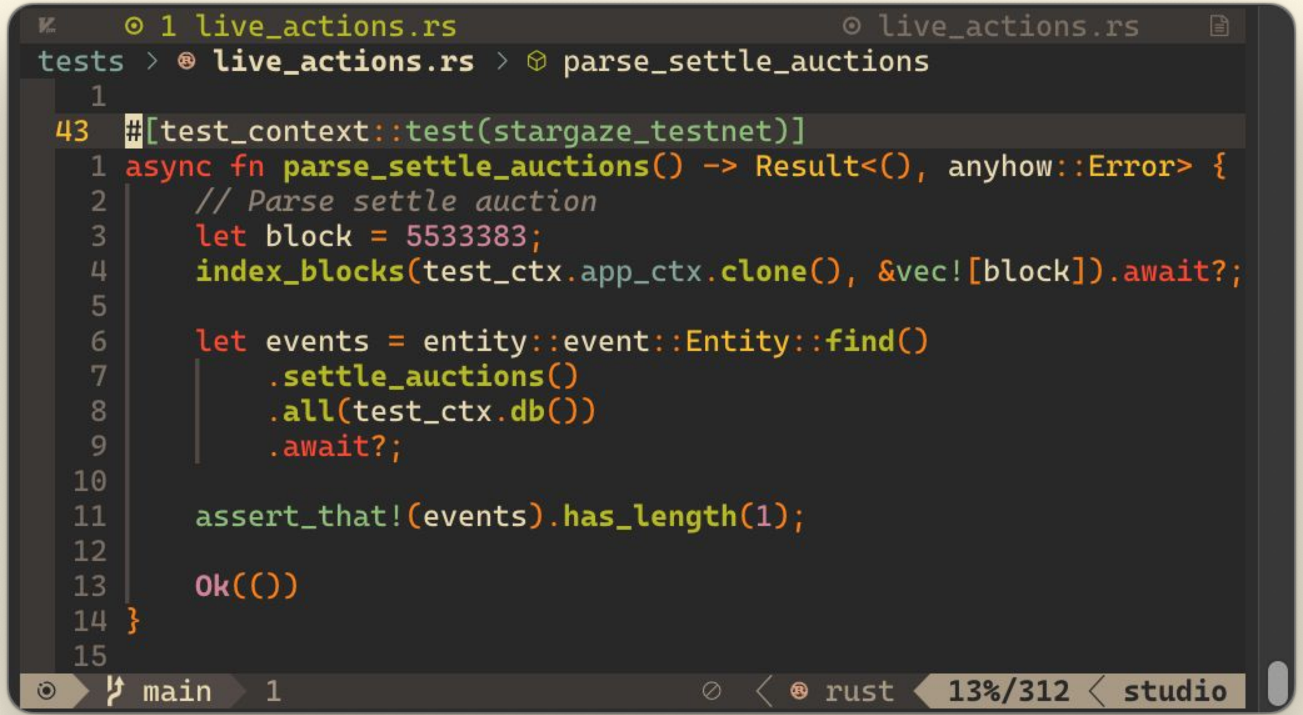
Test Driven Development


Known input: blocks, transactions, messages, events
Known expected output: SQL rows

Indexers are the perfect use-case for TDD

Storing blocks

Test Driven Development

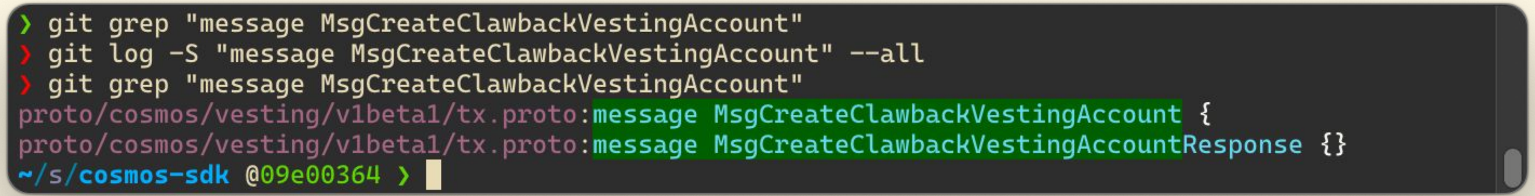


```rust
#[test_context::test(stargaze_testnet)]
async fn parse_settle_auctions() -> Result<(), anyhow::Error> {
    // Parse settle auction
    let block = 5533383;
    index_blocks(test_ctx.app_ctx.clone(), &vec![block]).await?;

    let events = entity::event::Entity::find()
        .settle_auctions()
        .all(test_ctx.db())
        .await?;

    assert_that!(events).has_length(1);

    Ok(())
}
```

Protobuf files

Cosmos-sdk chains are using protobuf for on-chain messages. But parsing historical messages isn't as easy as you'd think.

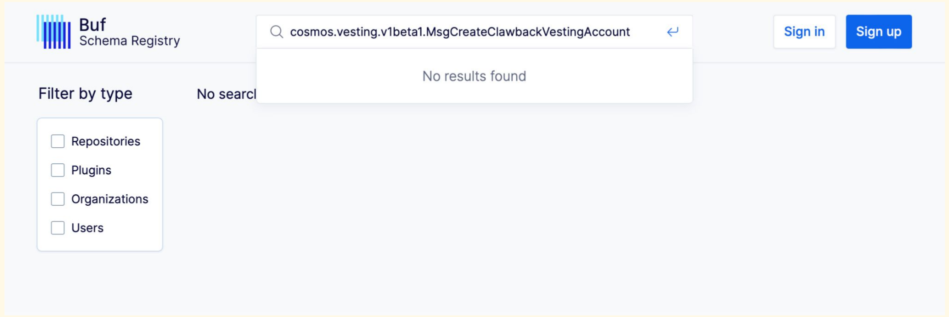
I had to dig in full git history to retrieve deleted protobuf files and fields, and merge all needed within my private repository.



```
❯ git grep "message MsgCreateClawbackVestingAccount"
❯ git log -S "message MsgCreateClawbackVestingAccount" --all
❯ git grep "message MsgCreateClawbackVestingAccount"
proto/cosmos/vesting/v1beta1/tx.proto:message MsgCreateClawbackVestingAccount {
proto/cosmos/vesting/v1beta1/tx.proto:message MsgCreateClawbackVestingAccountResponse {}
~/s/cosmos-sdk @09e00364 ❯
```

Storing blocks

Protobuf files



cosmos.vesting.v1beta1.MsgCreateClawbackVestingAccount

No results found

No searcl

- ✓ Missing/removed fields
- ✓ Missing files
- ✓ Linked to a buf.build project in buf.yaml, but not pushed and not available

Protobuf files

Do's

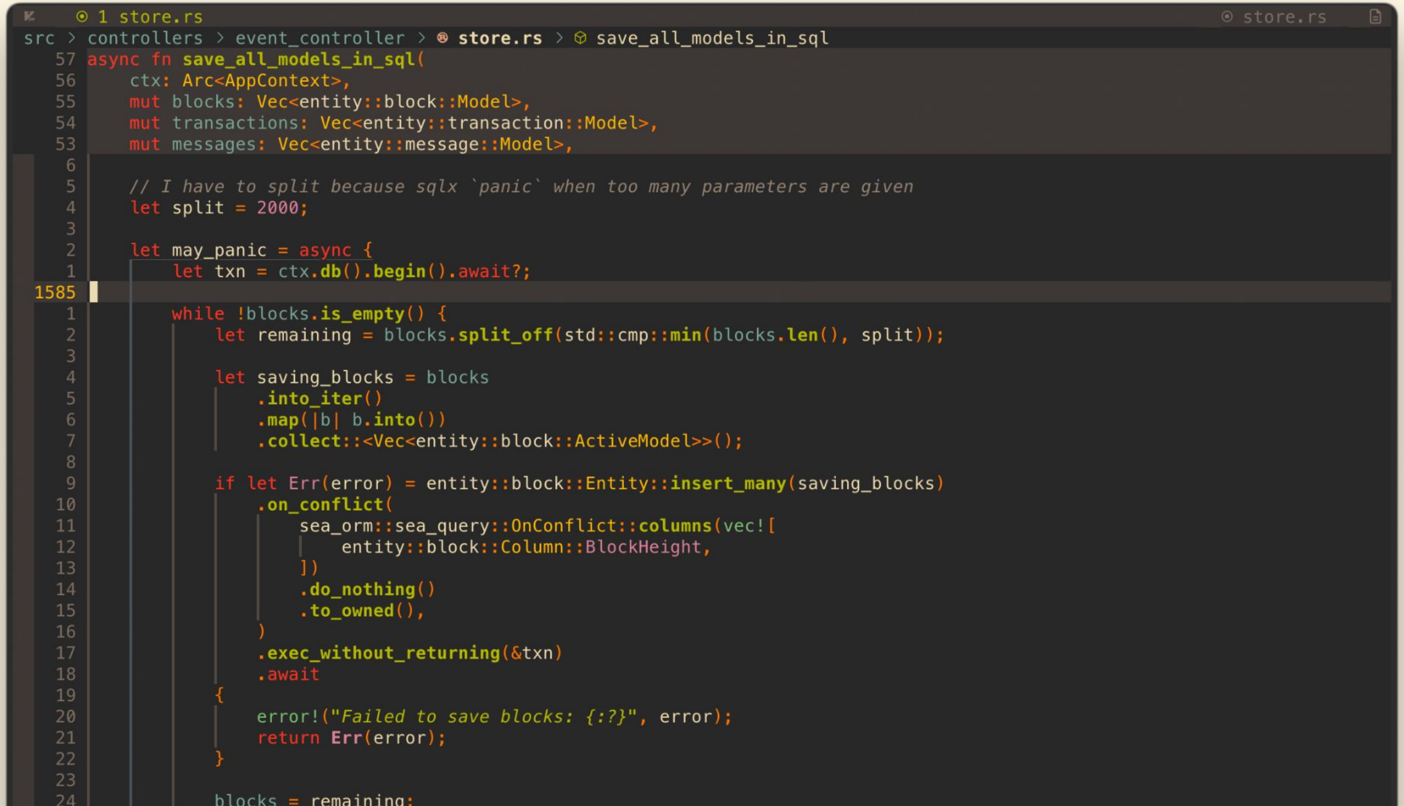
- Copy and save proto files into your repository
- Write your own protos to Rust structs into a specific crate (using prost, prost-build)
- Might need to search older deleted fields from proto files

Don'ts

- Don't link to existing repo via git submodule
- Don't rely on buf.build, or only to copy current existing files
- Don't think using existing proto files is fine, fields get deleted and replaced with *reserved* ***later***

Storing blocks

Storing in SQL



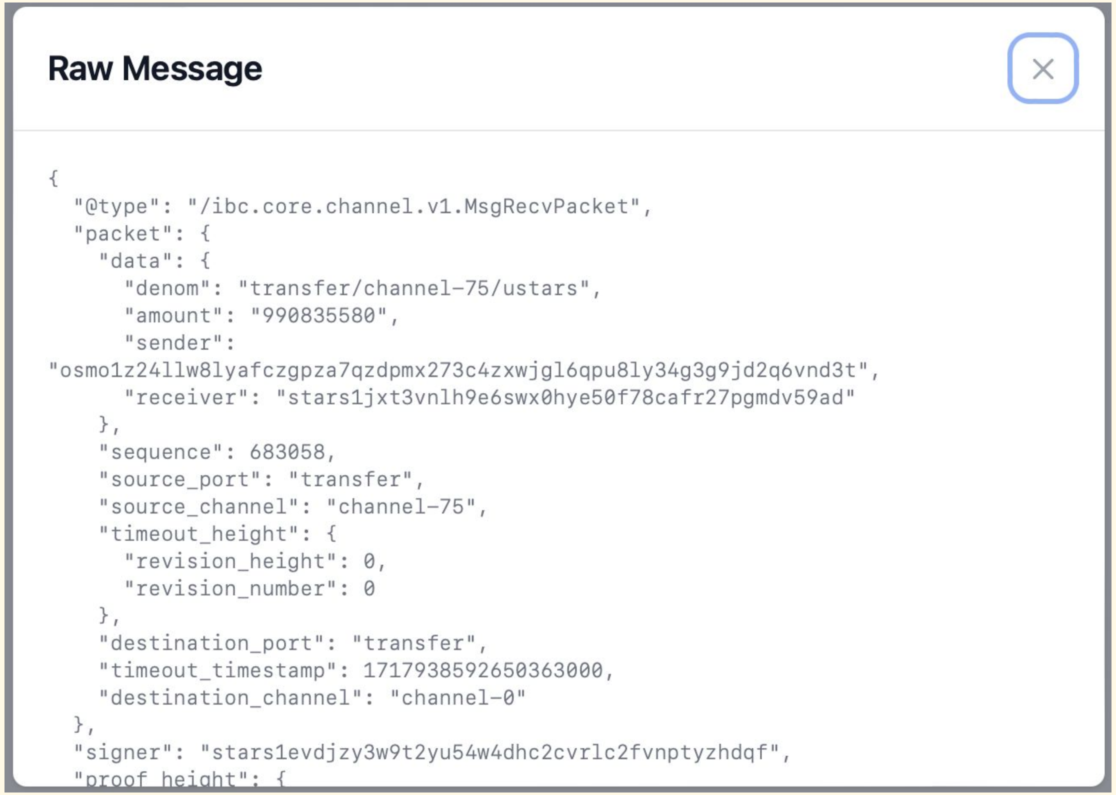
⊙ 1 store.rs ⊙ store.rs

src › controllers › event_controller › store.rs › save_all_models_in_sql

```
async fn save_all_models_in_sql(
    ctx: Arc<AppContext>,
    mut blocks: Vec<entity::block::Model>,
    mut transactions: Vec<entity::transaction::Model>,
    mut messages: Vec<entity::message::Model>,

    // I have to split because sqlx `panic` when too many parameters are given
    let split = 2000;

    let may_panic = async {
        let txn = ctx.db().begin().await?;

        while !blocks.is_empty() {
            let remaining = blocks.split_off(std::cmp::min(blocks.len(), split));

            let saving_blocks = blocks
                .into_iter()
                .map(|b| b.into())
                .collect::<Vec<entity::block::ActiveModel>>();

            if let Err(error) = entity::block::Entity::insert_many(saving_blocks)
                .on_conflict(
                    sea_orm::sea_query::OnConflict::columns(vec![
                        entity::block::Column::BlockHeight,
                    ])
                    .do_nothing()
                    .to_owned(),
                )
                .exec_without_returning(&txn)
                .await
            {
                error!("Failed to save blocks: {:?}", error);
                return Err(error);
            }

            blocks = remaining;
```

Storing blocks

Decoding IBC packets


Raw Message

```
{
  "@type": "/ibc.core.channel.v1.MsgRecvPacket",
  "packet": {
    "data": {
      "denom": "transfer/channel-75/ustars",
      "amount": "990835580",
      "sender":
"osmo1z24llw8lyafczgpza7qzdpmx273c4zxwjgl6qpu8ly34g3g9jd2q6vnd3t",
      "receiver": "stars1jxt3vnlh9e6swx0hye50f78cafr27pgmdv59ad"
    },
    "sequence": 683058,
    "source_port": "transfer",
    "source_channel": "channel-75",
    "timeout_height": {
      "revision_height": 0,
      "revision_number": 0
    },
    "destination_port": "transfer",
    "timeout_timestamp": 1717938592650363000,
    "destination_channel": "channel-0"
  },
  "signer": "stars1evdjzy3w9t2yu54w4dhc2cvrlc2fvnptyzhdqf",
  "proof_height": {
```

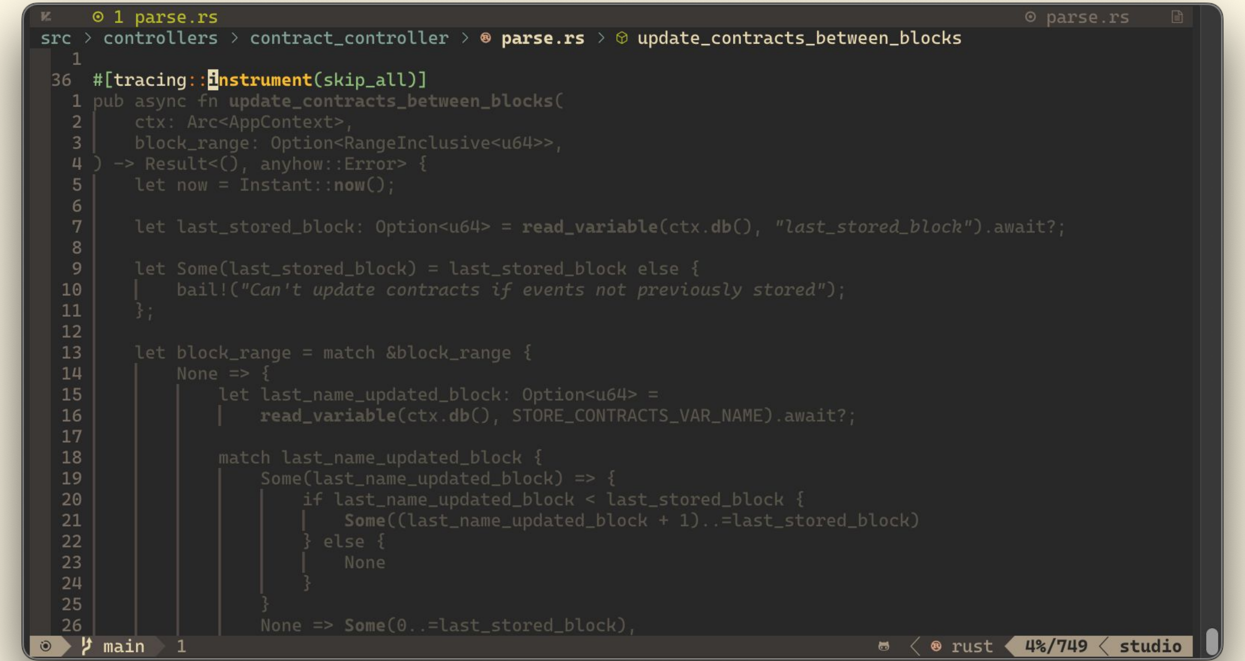
Parsing blocks

Processing stored txs

- Create NFT models (collections, nfts) and apply historical messages (owners, sales, ...)
- Create IBC related models (clients, connections, channels, denoms)
- Create Stargaze Names models
- Set invalid events, invalidated by later new events (bid invalidated by a sale)
- Create validator related models
- Fetch off-chain data

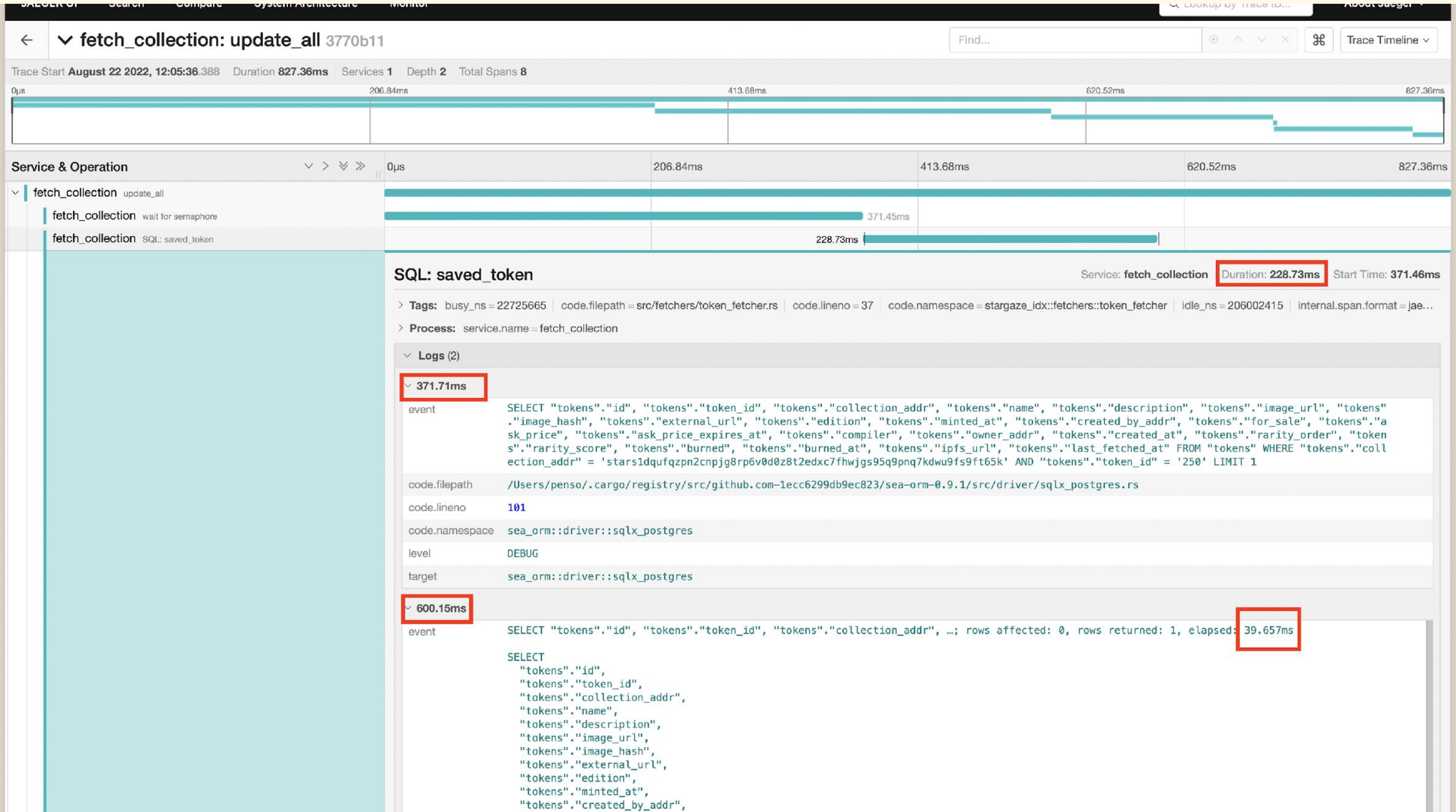
Instrumentation

tracing + opentelemetry


1 parse.rs
parse.rs
src > controllers > contract_controller > parse.rs > update_contracts_between_blocks
main 1
rust 4%/749 studio

```rust
#[tracing::instrument(skip_all)]
pub async fn update_contracts_between_blocks(
    ctx: Arc<AppContext>,
    block_range: Option<RangeInclusive<u64>>,
) -> Result<(), anyhow::Error> {
    let now = Instant::now();

    let last_stored_block: Option<u64> = read_variable(ctx.db(), "last_stored_block").await?;

    let Some(last_stored_block) = last_stored_block else {
        bail!("Can't update contracts if events not previously stored");
    };

    let block_range = match &block_range {
        None => {
            let last_name_updated_block: Option<u64> =
                read_variable(ctx.db(), STORE_CONTRACTS_VAR_NAME).await?;

            match last_name_updated_block {
                Some(last_name_updated_block) => {
                    if last_name_updated_block < last_stored_block {
                        Some((last_name_updated_block + 1)..=last_stored_block)
                    } else {
                        None
                    }
                }
                None => Some(0..=last_stored_block),
```

Jaeger dashboard



Trace Start **August 22 2022, 12:05:36**.388 Duration **827.36ms** Services **1** Depth **2** Total Spans **8**

fetch_collection update_all
fetch_collection wait for semaphore 371.45ms
fetch_collection SQL: saved_token 228.73ms

SQL: saved_token

Service: **fetch_collection** Duration: **228.73ms** Start Time: **371.46ms**

Tags: busy_ns = 22725665 | code.filepath = src/fetchers/token_fetcher.rs | code.lineno = 37 | code.namespace = stargaze_idx::fetchers::token_fetcher | idle_ns = 206002415 | internal.span.format = jae…

Process: service.name = fetch_collection

Logs (2)

371.71ms

event	`SELECT "tokens"."id", "tokens"."token_id", "tokens"."collection_addr", "tokens"."name", "tokens"."description", "tokens"."image_url", "tokens"."image_hash", "tokens"."external_url", "tokens"."edition", "tokens"."minted_at", "tokens"."created_by_addr", "tokens"."for_sale", "tokens"."ask_price", "tokens"."ask_price_expires_at", "tokens"."compiler", "tokens"."owner_addr", "tokens"."created_at", "tokens"."rarity_order", "tokens"."rarity_score", "tokens"."burned", "tokens"."burned_at", "tokens"."ipfs_url", "tokens"."last_fetched_at" FROM "tokens" WHERE "tokens"."collection_addr" = 'stars1dqufqzpn2cnpjg8rp6v0d0z8t2edxc7fhwjgs95q9pnq7kdwu9fs9ft65k' AND "tokens"."token_id" = '250' LIMIT 1`
code.filepath	`/Users/penso/.cargo/registry/src/github.com-1ecc6299db9ec823/sea-orm-0.9.1/src/driver/sqlx_postgres.rs`
code.lineno	101
code.namespace	sea_orm::driver::sqlx_postgres
level	DEBUG
target	sea_orm::driver::sqlx_postgres

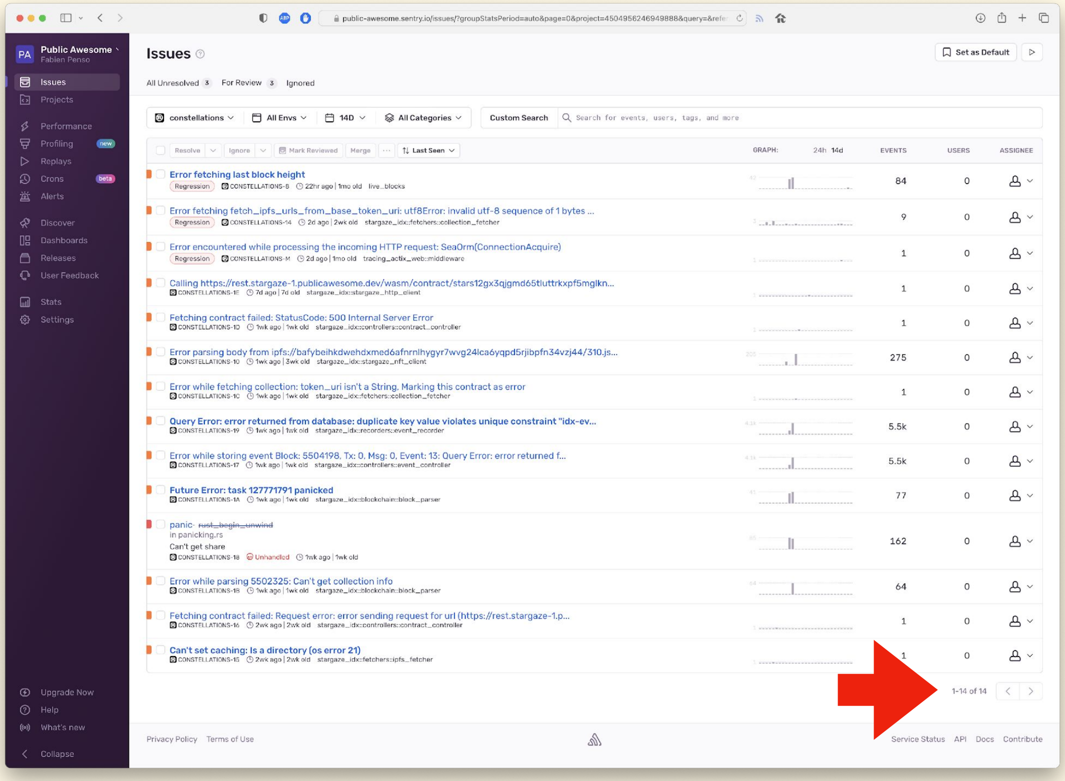
600.15ms

event

```
SELECT "tokens"."id", "tokens"."token_id", "tokens"."collection_addr", …; rows affected: 0, rows returned: 1, elapsed: 39.657ms

SELECT
  "tokens"."id",
  "tokens"."token_id",
  "tokens"."collection_addr",
  "tokens"."name",
  "tokens"."description",
  "tokens"."image_url",
  "tokens"."image_hash",
  "tokens"."external_url",
  "tokens"."edition",
  "tokens"."minted_at",
  "tokens"."created_by_addr",
```

Instrumentation

Sentry dashboard


public-awesome.sentry.io/issues/?groupStatsPeriod=auto&page=0&project=4504956246949888&query=&refe

Public Awesome
Fabien Penso

Issues · Projects · Performance · Profiling (new) · Replays · Crons (beta) · Alerts · Discover · Dashboards · Releases · User Feedback · Stats · Settings · Upgrade Now · Help · What's new · Collapse

Issues

All Unresolved 3 · For Review 3 · Ignored

constellations · All Envs · 14D · All Categories · Custom Search · Search for events, users, tags, and more

Resolve · Ignore · Mark Reviewed · Merge · ··· · Last Seen · Set as Default

	GRAPH: 24h 14d	EVENTS	USERS	ASSIGNEE
Error fetching last block height Regression · CONSTELLATIONS-8 · 22hr ago | 1mo old · live_blocks		84	0	
Error fetching fetch_ipfs_urls_from_base_token_uri: utf8Error: invalid utf-8 sequence of 1 bytes ... Regression · CONSTELLATIONS-14 · 2d ago | 2wk old · stargaze_idx::fetchers::collection_fetcher		9	0	
Error encountered while processing the incoming HTTP request: SeaOrm(ConnectionAcquire) Regression · CONSTELLATIONS-M · 2d ago | 1mo old · tracing_actix_web::middleware		1	0	
Calling https://rest.stargaze-1.publicawesome.dev/wasm/contract/stars12gx3qjgmd65tluttrkxpf5mglkn... CONSTELLATIONS-1E · 7d ago | 7d old · stargaze_idx::stargaze_http_client		1	0	
Fetching contract failed: StatusCode: 500 Internal Server Error CONSTELLATIONS-1D · 1wk ago | 1wk old · stargaze_idx::controllers::contract_controller		1	0	
Error parsing body from ipfs://bafybeihkdwehdxmed6afnrnlhygyr7wvg24lca6yqpd5rjibpfn34vzj44/310.js... CONSTELLATIONS-10 · 1wk ago | 3wk old · stargaze_idx::stargaze_nft_client		275	0	
Error while fetching collection: token_uri isn't a String. Marking this contract as error CONSTELLATIONS-1C · 1wk ago | 1wk old · stargaze_idx::fetchers::collection_fetcher		1	0	
Query Error: error returned from database: duplicate key value violates unique constraint "idx-ev... CONSTELLATIONS-19 · 1wk ago | 1wk old · stargaze_idx::recorders::event_recorder		5.5k	0	
Error while storing event Block: 5504198, Tx: 0, Msg: 0, Event: 13: Query Error: error returned f... CONSTELLATIONS-17 · 1wk ago | 1wk old · stargaze_idx::controllers::event_controller		5.5k	0	
Future Error: task 127771791 panicked CONSTELLATIONS-1A · 1wk ago | 1wk old · stargaze_idx::blockchain::block_parser		77	0	
panic- ~~rust_begin_unwind~~ in panicking.rs Can't get share CONSTELLATIONS-18 · Unhandled · 1wk ago | 1wk old		162	0	
Error while parsing 5502325: Can't get collection info CONSTELLATIONS-1B · 1wk ago | 1wk old · stargaze_idx::blockchain::block_parser		64	0	
Fetching contract failed: Request error: error sending request for url (https://rest.stargaze-1.p... CONSTELLATIONS-16 · 2wk ago | 2wk old · stargaze_idx::controllers::contract_controller		1	0	
Can't set caching: Is a directory (os error 21) CONSTELLATIONS-15 · 2wk ago | 2wk old · stargaze_idx::fetchers::ipfs_fetcher		1	0	

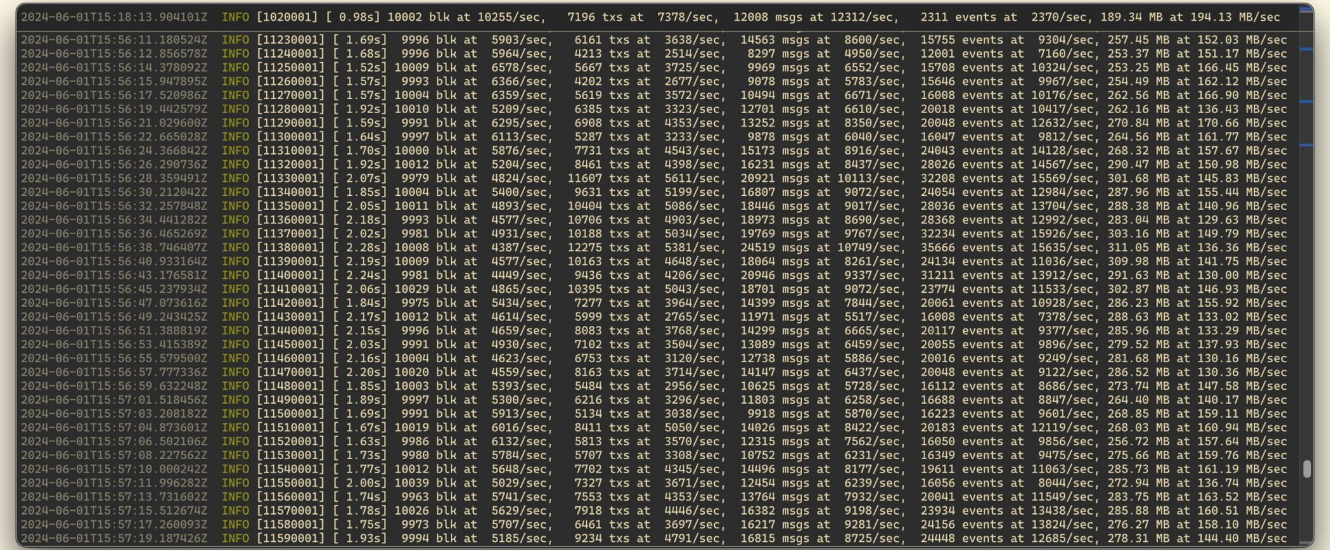
1-14 of 14

Privacy Policy · Terms of Use · Service Status · API · Docs · Contribute

Benchmarking

Indexing speed

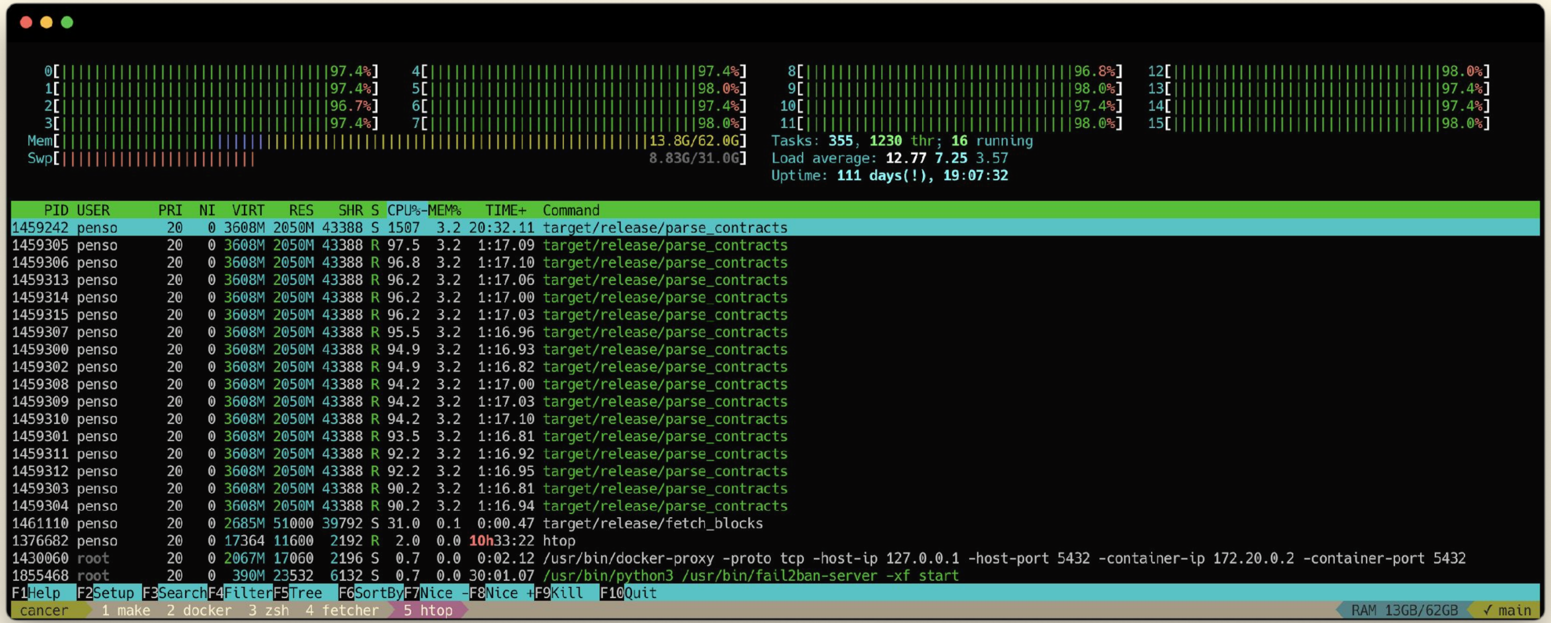
Look at parsing performance based on data throughput, not block count. I had different speed for different chains and found out I had the same data throughput.



```
2024-06-01T15:18:13.904101Z  INFO [1020001] [ 0.98s] 10002 blk at 10255/sec,   7196 txs at  7378/sec,  12008 msgs at 12312/sec,   2311 events at  2370/sec, 189.34 MB at 194.13 MB/sec
2024-06-01T15:56:11.180524Z  INFO [11230001] [ 1.69s]  9996 blk at  5903/sec,   6161 txs at  3638/sec,  14563 msgs at  8600/sec,  15755 events at  9304/sec, 257.45 MB at 152.03 MB/sec
2024-06-01T15:56:12.856578Z  INFO [11240001] [ 1.68s]  9996 blk at  5964/sec,   4213 txs at  2514/sec,   8297 msgs at  4950/sec,  12001 events at  7160/sec, 253.37 MB at 151.17 MB/sec
2024-06-01T15:56:14.378092Z  INFO [11250001] [ 1.52s] 10009 blk at  6578/sec,   5667 txs at  3725/sec,   9969 msgs at  6552/sec,  15708 events at 10324/sec, 253.25 MB at 166.45 MB/sec
2024-06-01T15:56:15.947895Z  INFO [11260001] [ 1.57s]  9993 blk at  6366/sec,   4202 txs at  2677/sec,   9078 msgs at  5783/sec,  15646 events at  9967/sec, 254.49 MB at 162.12 MB/sec
2024-06-01T15:56:17.520986Z  INFO [11270001] [ 1.57s] 10004 blk at  6359/sec,   5619 txs at  3572/sec,  10494 msgs at  6671/sec,  16008 events at 10176/sec, 262.56 MB at 166.90 MB/sec
2024-06-01T15:56:19.442579Z  INFO [11280001] [ 1.92s] 10010 blk at  5209/sec,   6385 txs at  3323/sec,  12701 msgs at  6610/sec,  20018 events at 10417/sec, 262.16 MB at 136.43 MB/sec
2024-06-01T15:56:21.029600Z  INFO [11290001] [ 1.59s]  9991 blk at  6295/sec,   6908 txs at  4353/sec,  13252 msgs at  8350/sec,  20048 events at 12632/sec, 270.84 MB at 170.66 MB/sec
2024-06-01T15:56:22.665028Z  INFO [11300001] [ 1.64s]  9997 blk at  6113/sec,   5287 txs at  3233/sec,   9878 msgs at  6040/sec,  16047 events at  9812/sec, 264.56 MB at 161.77 MB/sec
2024-06-01T15:56:24.366842Z  INFO [11310001] [ 1.70s] 10000 blk at  5876/sec,   7731 txs at  4543/sec,  15173 msgs at  8916/sec,  24043 events at 14128/sec, 268.32 MB at 157.67 MB/sec
2024-06-01T15:56:26.290736Z  INFO [11320001] [ 1.92s] 10012 blk at  5204/sec,   8461 txs at  4398/sec,  16231 msgs at  8437/sec,  28026 events at 14567/sec, 290.47 MB at 150.98 MB/sec
2024-06-01T15:56:28.359491Z  INFO [11330001] [ 2.07s]  9979 blk at  4824/sec,  11607 txs at  5611/sec,  20921 msgs at 10113/sec,  32208 events at 15569/sec, 301.68 MB at 145.83 MB/sec
2024-06-01T15:56:30.212042Z  INFO [11340001] [ 1.85s] 10004 blk at  5400/sec,   9631 txs at  5199/sec,  16807 msgs at  9072/sec,  24054 events at 12984/sec, 287.96 MB at 155.44 MB/sec
2024-06-01T15:56:32.257848Z  INFO [11350001] [ 2.05s] 10011 blk at  4893/sec,  10404 txs at  5086/sec,  18446 msgs at  9017/sec,  28036 events at 13704/sec, 288.38 MB at 140.96 MB/sec
2024-06-01T15:56:34.441282Z  INFO [11360001] [ 2.18s]  9993 blk at  4577/sec,  10706 txs at  4903/sec,  18973 msgs at  8690/sec,  28368 events at 12992/sec, 283.04 MB at 129.63 MB/sec
2024-06-01T15:56:36.465269Z  INFO [11370001] [ 2.02s]  9981 blk at  4931/sec,  10188 txs at  5034/sec,  19769 msgs at  9767/sec,  32234 events at 15926/sec, 303.16 MB at 149.79 MB/sec
2024-06-01T15:56:38.746407Z  INFO [11380001] [ 2.28s] 10008 blk at  4387/sec,  12275 txs at  5381/sec,  24519 msgs at 10749/sec,  35666 events at 15635/sec, 311.05 MB at 136.36 MB/sec
2024-06-01T15:56:40.933164Z  INFO [11390001] [ 2.19s] 10009 blk at  4577/sec,  10163 txs at  4648/sec,  18064 msgs at  8261/sec,  24134 events at 11036/sec, 309.98 MB at 141.75 MB/sec
2024-06-01T15:56:43.176581Z  INFO [11400001] [ 2.24s]  9981 blk at  4449/sec,   9436 txs at  4206/sec,  20946 msgs at  9337/sec,  31211 events at 13912/sec, 291.63 MB at 130.00 MB/sec
2024-06-01T15:56:45.237934Z  INFO [11410001] [ 2.06s] 10029 blk at  4865/sec,  10395 txs at  5043/sec,  18701 msgs at  9072/sec,  23774 events at 11533/sec, 302.87 MB at 146.93 MB/sec
2024-06-01T15:56:47.073616Z  INFO [11420001] [ 1.84s]  9975 blk at  5434/sec,   7277 txs at  3964/sec,  14399 msgs at  7844/sec,  20061 events at 10928/sec, 286.23 MB at 155.92 MB/sec
2024-06-01T15:56:49.243425Z  INFO [11430001] [ 2.17s] 10012 blk at  4614/sec,   5999 txs at  2765/sec,  11971 msgs at  5517/sec,  16008 events at  7378/sec, 288.63 MB at 133.02 MB/sec
2024-06-01T15:56:51.388819Z  INFO [11440001] [ 2.15s]  9996 blk at  4659/sec,   8083 txs at  3768/sec,  14299 msgs at  6665/sec,  20117 events at  9377/sec, 285.96 MB at 133.29 MB/sec
2024-06-01T15:56:53.415389Z  INFO [11450001] [ 2.03s]  9991 blk at  4930/sec,   7102 txs at  3504/sec,  13089 msgs at  6459/sec,  20055 events at  9896/sec, 279.52 MB at 137.93 MB/sec
2024-06-01T15:56:55.579500Z  INFO [11460001] [ 2.16s] 10004 blk at  4623/sec,   6753 txs at  3120/sec,  12738 msgs at  5886/sec,  20016 events at  9249/sec, 281.68 MB at 130.16 MB/sec
2024-06-01T15:56:57.777336Z  INFO [11470001] [ 2.20s] 10020 blk at  4559/sec,   8163 txs at  3714/sec,  14147 msgs at  6437/sec,  20048 events at  9122/sec, 286.52 MB at 130.36 MB/sec
2024-06-01T15:56:59.632248Z  INFO [11480001] [ 1.85s] 10003 blk at  5393/sec,   5484 txs at  2956/sec,  10625 msgs at  5728/sec,  16112 events at  8686/sec, 273.74 MB at 147.58 MB/sec
2024-06-01T15:57:01.518456Z  INFO [11490001] [ 1.89s]  9997 blk at  5300/sec,   6216 txs at  3296/sec,  11803 msgs at  6258/sec,  16688 events at  8847/sec, 264.40 MB at 140.17 MB/sec
2024-06-01T15:57:03.208182Z  INFO [11500001] [ 1.69s]  9991 blk at  5913/sec,   5134 txs at  3038/sec,   9918 msgs at  5870/sec,  16223 events at  9601/sec, 268.85 MB at 159.11 MB/sec
2024-06-01T15:57:04.873601Z  INFO [11510001] [ 1.67s] 10019 blk at  6016/sec,   8411 txs at  5050/sec,  14026 msgs at  8422/sec,  20183 events at 12119/sec, 268.03 MB at 160.94 MB/sec
2024-06-01T15:57:06.502106Z  INFO [11520001] [ 1.63s]  9986 blk at  6132/sec,   5813 txs at  3570/sec,  12315 msgs at  7562/sec,  16050 events at  9856/sec, 256.72 MB at 157.64 MB/sec
2024-06-01T15:57:08.227562Z  INFO [11530001] [ 1.73s]  9980 blk at  5784/sec,   5707 txs at  3308/sec,  10752 msgs at  6231/sec,  16349 events at  9475/sec, 275.66 MB at 159.76 MB/sec
2024-06-01T15:57:10.000242Z  INFO [11540001] [ 1.77s] 10012 blk at  5648/sec,   7702 txs at  4345/sec,  14496 msgs at  8177/sec,  19611 events at 11063/sec, 285.73 MB at 161.19 MB/sec
2024-06-01T15:57:11.996282Z  INFO [11550001] [ 2.00s] 10039 blk at  5029/sec,   7327 txs at  3671/sec,  12454 msgs at  6239/sec,  16056 events at  8044/sec, 272.94 MB at 136.74 MB/sec
2024-06-01T15:57:13.731602Z  INFO [11560001] [ 1.74s]  9963 blk at  5741/sec,   7553 txs at  4353/sec,  13764 msgs at  7932/sec,  20041 events at 11549/sec, 283.75 MB at 163.52 MB/sec
2024-06-01T15:57:15.512674Z  INFO [11570001] [ 1.78s] 10026 blk at  5629/sec,   7918 txs at  4446/sec,  16382 msgs at  9198/sec,  23934 events at 13438/sec, 285.88 MB at 160.51 MB/sec
2024-06-01T15:57:17.260093Z  INFO [11580001] [ 1.75s]  9973 blk at  5707/sec,   6461 txs at  3697/sec,  16217 msgs at  9281/sec,  24156 events at 13824/sec, 276.27 MB at 158.10 MB/sec
2024-06-01T15:57:19.187426Z  INFO [11590001] [ 1.93s]  9994 blk at  5185/sec,   9234 txs at  4791/sec,  16815 msgs at  8725/sec,  24448 events at 12685/sec, 278.31 MB at 144.40 MB/sec
```

Performance

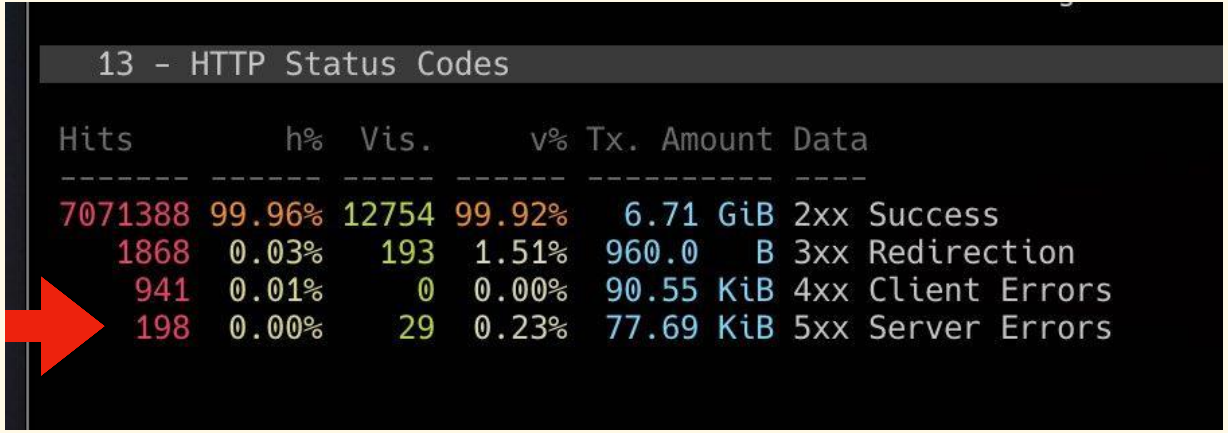
CPU Usage while indexing



```
    0[|||||||||||||||||||||||||||||||||97.4%]   4[|||||||||||||||||||||||||||||||||97.4%]    8[|||||||||||||||||||||||||||||||||96.8%]   12[|||||||||||||||||||||||||||||||||98.0%]
    1[|||||||||||||||||||||||||||||||||97.4%]   5[|||||||||||||||||||||||||||||||||98.0%]    9[|||||||||||||||||||||||||||||||||98.0%]   13[|||||||||||||||||||||||||||||||||97.4%]
    2[|||||||||||||||||||||||||||||||||96.7%]   6[|||||||||||||||||||||||||||||||||97.4%]   10[|||||||||||||||||||||||||||||||||97.4%]   14[|||||||||||||||||||||||||||||||||97.4%]
    3[|||||||||||||||||||||||||||||||||97.4%]   7[|||||||||||||||||||||||||||||||||98.0%]   11[|||||||||||||||||||||||||||||||||98.0%]   15[|||||||||||||||||||||||||||||||||98.0%]
  Mem[||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||13.8G/62.0G]  Tasks: 355, 1230 thr; 16 running
  Swp[||||||||||||||||||||||||                                               8.83G/31.0G]  Load average: 12.77 7.25 3.57
                                                                                           Uptime: 111 days(!), 19:07:32

    PID USER       PRI  NI  VIRT   RES   SHR S CPU%-MEM%   TIME+  Command
1459242 penso       20   0 3608M 2050M 43388 S 1507  3.2 20:32.11 target/release/parse_contracts
1459305 penso       20   0 3608M 2050M 43388 R 97.5  3.2  1:17.09 target/release/parse_contracts
1459306 penso       20   0 3608M 2050M 43388 R 96.8  3.2  1:17.10 target/release/parse_contracts
1459313 penso       20   0 3608M 2050M 43388 R 96.2  3.2  1:17.06 target/release/parse_contracts
1459314 penso       20   0 3608M 2050M 43388 R 96.2  3.2  1:17.00 target/release/parse_contracts
1459315 penso       20   0 3608M 2050M 43388 R 96.2  3.2  1:17.03 target/release/parse_contracts
1459307 penso       20   0 3608M 2050M 43388 R 95.5  3.2  1:16.96 target/release/parse_contracts
1459300 penso       20   0 3608M 2050M 43388 R 94.9  3.2  1:16.93 target/release/parse_contracts
1459302 penso       20   0 3608M 2050M 43388 R 94.9  3.2  1:16.82 target/release/parse_contracts
1459308 penso       20   0 3608M 2050M 43388 R 94.2  3.2  1:17.00 target/release/parse_contracts
1459309 penso       20   0 3608M 2050M 43388 R 94.2  3.2  1:17.03 target/release/parse_contracts
1459310 penso       20   0 3608M 2050M 43388 R 94.2  3.2  1:17.10 target/release/parse_contracts
1459301 penso       20   0 3608M 2050M 43388 R 93.5  3.2  1:16.81 target/release/parse_contracts
1459311 penso       20   0 3608M 2050M 43388 R 92.2  3.2  1:16.92 target/release/parse_contracts
1459312 penso       20   0 3608M 2050M 43388 R 92.2  3.2  1:16.95 target/release/parse_contracts
1459303 penso       20   0 3608M 2050M 43388 R 90.2  3.2  1:16.81 target/release/parse_contracts
1459304 penso       20   0 3608M 2050M 43388 R 90.2  3.2  1:16.94 target/release/parse_contracts
1461110 penso       20   0 2685M 51000 39792 S 31.0  0.1  0:00.47 target/release/fetch_blocks
1376682 penso       20   0 17364 11600  2192 R  2.0  0.0 10h33:22 htop
1430060 root        20   0 2067M 17060  2196 S  0.7  0.0  0:02.12 /usr/bin/docker-proxy -proto tcp -host-ip 127.0.0.1 -host-port 5432 -container-ip 172.20.0.2 -container-port 5432
1855468 root        20   0  390M 23532  6132 S  0.7  0.0 30:01.07 /usr/bin/python3 /usr/bin/fail2ban-server -xf start
F1Help  F2Setup F3SearchF4FilterF5Tree  F6SortByF7Nice -F8Nice +F9Kill  F10Quit
 cancer    1 make   2 docker   3 zsh   4 fetcher   5 htop                                                                                                         RAM 13GB/62GB   ✓ main
```

Error rate

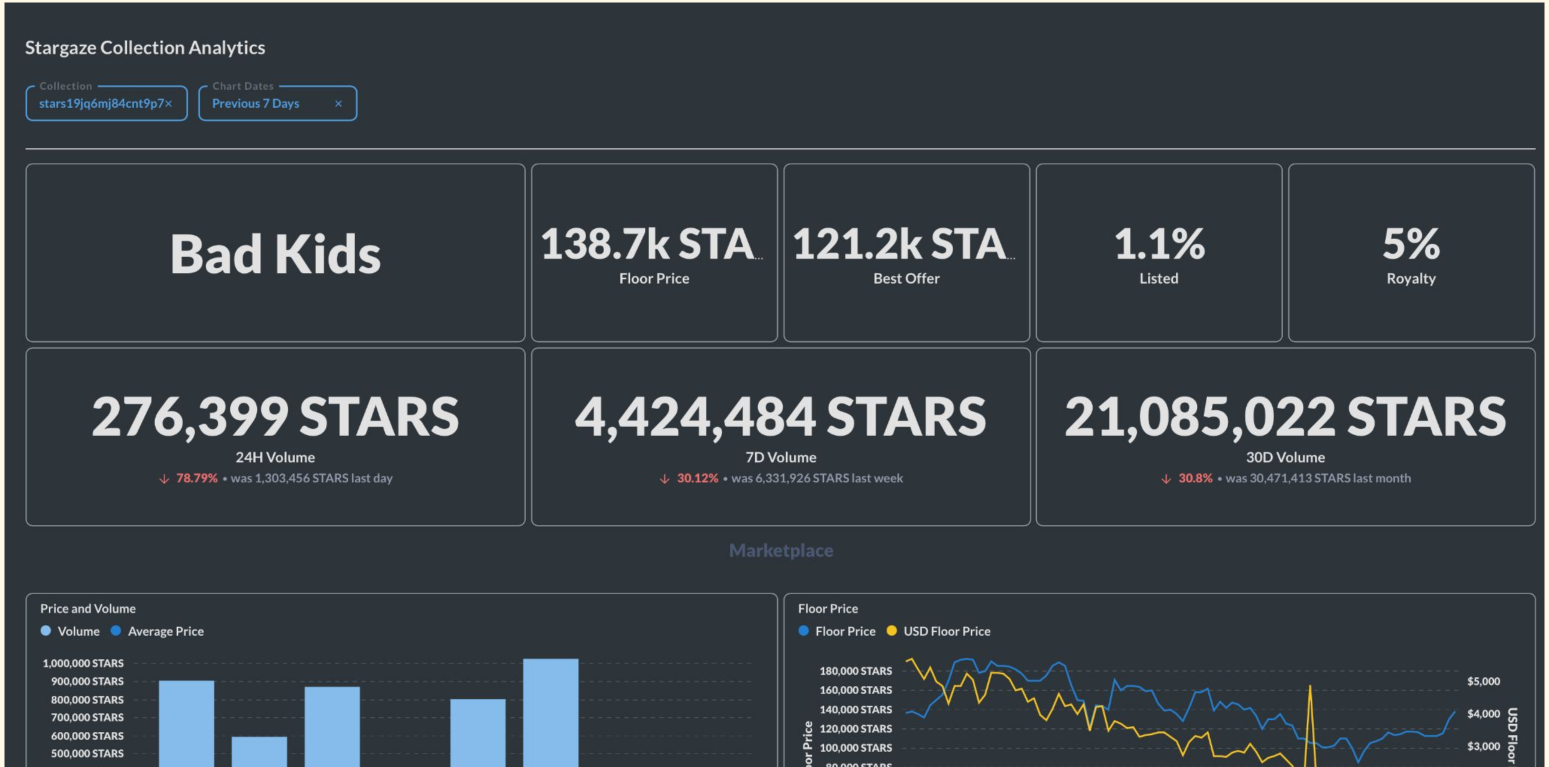
198 errors on 7M requests



```
  13 - HTTP Status Codes

Hits          h%   Vis.      v%  Tx. Amount Data
-------- ------ ----- ------ ---------- ----
7071388 99.96% 12754 99.92%   6.71 GiB 2xx Success
   1868  0.03%   193  1.51%  960.0   B 3xx Redirection
    941  0.01%     0  0.00%  90.55 KiB 4xx Client Errors
    198  0.00%    29  0.23%  77.69 KiB 5xx Server Errors
```

BI Tools

Metabase


Stargaze Collection Analytics

Collection: stars19jq6mj84cnt9p7×

Chart Dates: Previous 7 Days ×

Bad Kids

138.7k STA… Floor Price

121.2k STA… Best Offer

1.1% Listed

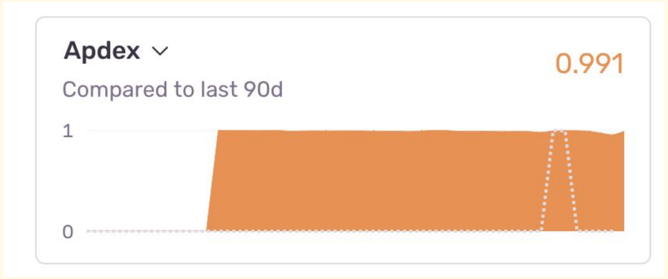
5% Royalty

276,399 STARS
24H Volume
↓ 78.79% • was 1,303,456 STARS last day

4,424,484 STARS
7D Volume
↓ 30.12% • was 6,331,926 STARS last week

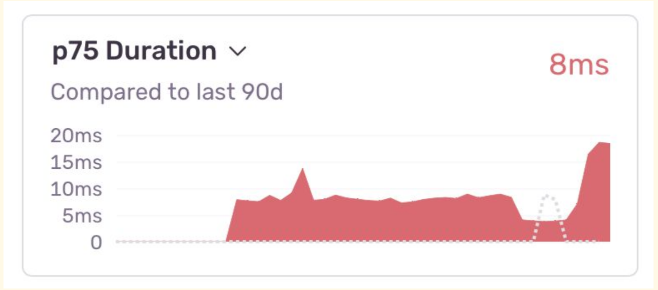
21,085,022 STARS
30D Volume
↓ 30.8% • was 30,471,413 STARS last month

Marketplace

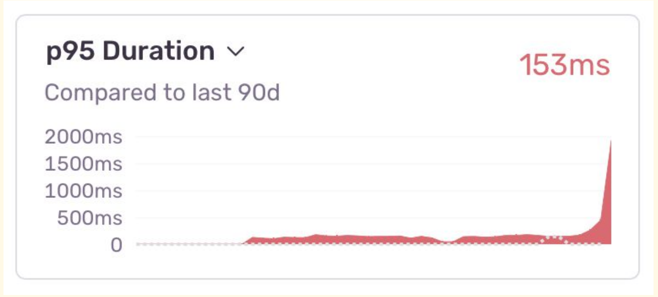


An excellent score falls in 1.00-0.94, a good score ranks from 0.93-0.85, a fair score hits 0.84-0.70 and a poor one between 0.69 and 0.49. Any lower number is unacceptable.

Source: TechTarget



The p75 threshold is the value at which 25% of transaction durations are greater than the threshold



✨Stargaze: DEVMOS 2024 After-Party✨

https://lu.ma/Stargaze_DEVMOS-2024

Thank you

For further discussions, reach out to @fabienpenso or devmos@pen.so