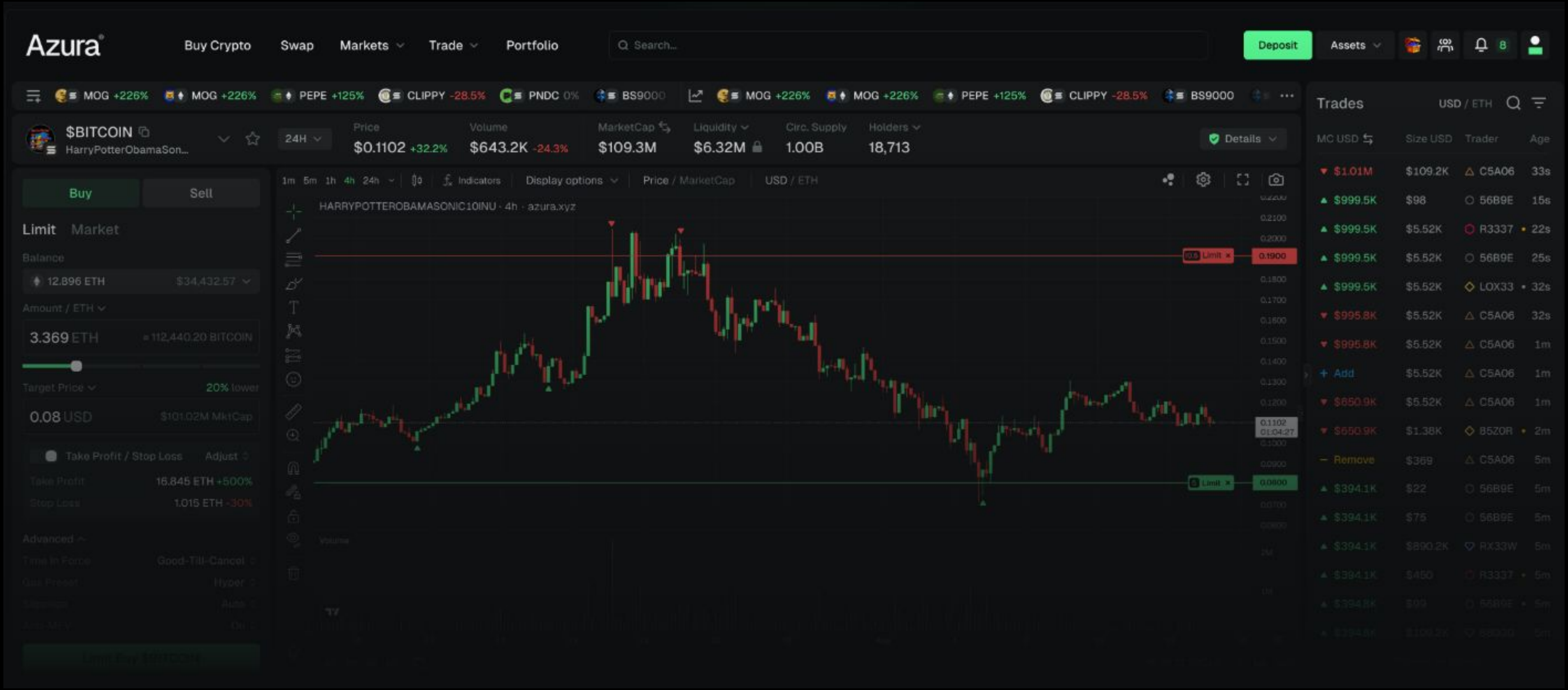
Azura

Real-Time Cross-Chain Trading Platform

March 26 2025



$BITCOIN
HarryPotterObamaSon...

Price $0.1102 +32.2%
Volume $643.2K -24.3%
MarketCap $109.3M
Liquidity $6.32M
Circ. Supply 1.00B
Holders 18,713

Buy Sell

Limit Market

Balance: 12.896 ETH $34,432.57

Amount / ETH: 3.369 ETH ≈ 112,440.20 BITCOIN

Target Price: 0.08 USD $101.02M MktCap 20% lower

Take Profit / Stop Loss Adjust
Take Profit 16.845 ETH +500%
Stop Loss 1.015 ETH -30%

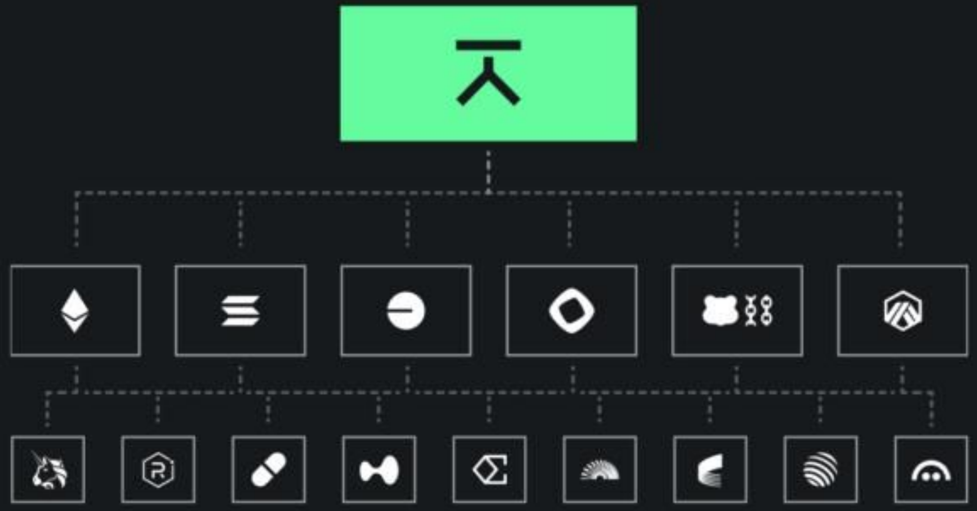
Advanced
Time In Force Good-Till-Cancel
Gas Preset Hyper

Trades USD / ETH

MC USD	Size USD	Trader	Age
▼ $1.01M	$109.2K	△ C5A06	33s
▲ $999.5K	$98	○ 56B9E	15s
▲ $999.5K	$5.52K	○ R3337 •	22s
▲ $999.5K	$5.52K	○ 56B9E	25s
▲ $999.5K	$5.52K	◇ LOX33 •	32s
▼ $995.8K	$5.52K	△ C5A06	32s
▼ $995.8K	$5.52K	△ C5A06	1m
+ Add	$5.52K	△ C5A06	1m
▼ $650.9K	$5.52K	△ C5A06	1m
▼ $650.9K	$1.38K	◇ B5ZOR •	2m
− Remove	$369	△ C5A06	5m
▲ $394.1K	$22	○ 56B9E	5m
▲ $394.1K	$75	○ 56B9E	5m
▲ $394.1K	$890.2K	♡ RX33W	5m
▲ $394.1K	$450	○ R3337 •	5m
▲ $394.8K	$99	○ 56B9E •	5m
[illegible]	$109.2K	△ C5A06	5m

The Application & Interfacing Standard

The Azura application bundles together and abstracts away DeFi's complexities through hundreds of bespoke integrations with only the most battle hardened and trusted protocols.



Agenda

01 Challenge & Context

02 Data Stack, Architecture

03 Use Cases

04 Materialized Views

05 Materialized Views: OHLC

06 Questions

The Cross-Chain Challenge

Multi-Chain Support

Handling Solana and 7 Ethereum-based chains simultaneously creates a firehose of diverse data such as:

- ALL On-chain Trades
    - OHLC data for charts
    - “Trending” and “New” assets
- Add/Remove Liquidity Events
- Token Transfers etc

Scale

8TB production database with 11GB read/sec.
Minimum vertical scaling (per node): 64 GiB, 16 vCPU
Maximum vertical scaling (per node): 236 GiB, 59 vCPU | Number of replicas - 4

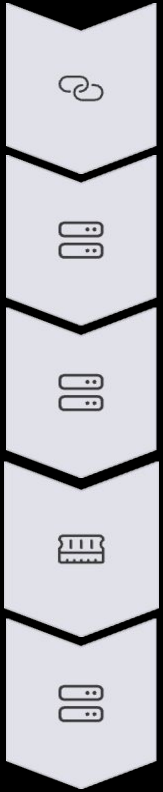
Performance Demands

75% of operations are selects requiring near-instant response times.
25% of operations are inserts from our live indexer that are batched.

Data Stack

1. Custom Rust Binaries
    a. Consume raw data from blockchain RPCs
    b. Transform as required, enrich each struct (e.g: trades)
2. Transport into clickhouse, realtime message queues
3. Materialized views on clickhouse further enrich our use cases.
4. Data Engine (Rust Binary) that consumes all of the above and provides low latency access to market data.

Our Hybrid Architecture



1. Blockchain Events

Live trades captured across 7 chains

2. Hermaus (Rust Indexer)

Indexes trades and other data via batch inserts

3. ClickHouse

Stores 8TB of trading data

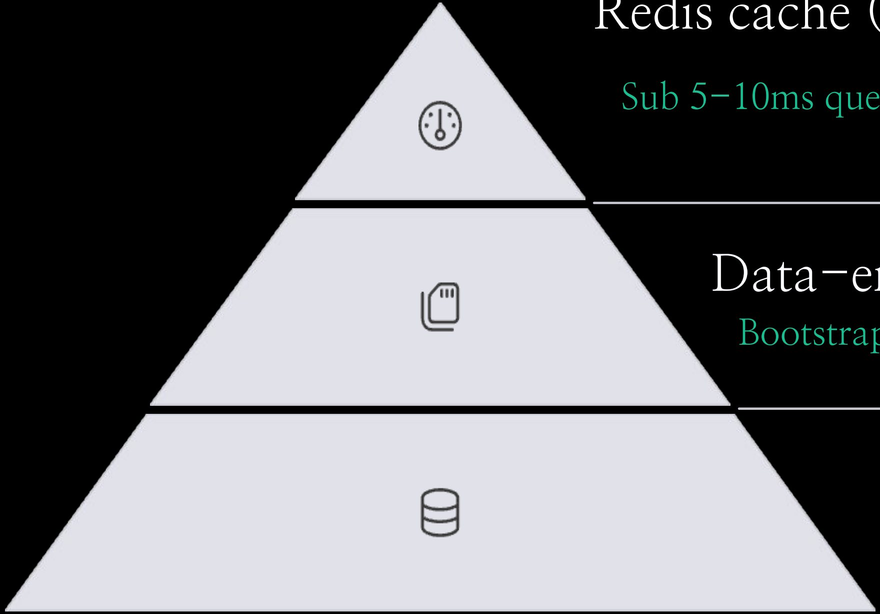
4. Data–Engine (Rust)

Custom in–memory database, WS / REST server.

5. Frontend Client

Serves real–time data to traders on our platform

Memory-First Low-Latency Strategy



Redis cache (select data)

Sub 5-10ms query times, kept fresh by the indexer that also writes to CH

Data-engine In-Memory Processing (hot data)

Bootstraps via CH MV's, tracks new events to apply state transitions

ClickHouse Fallback (complete data)

Disk queries only when necessary

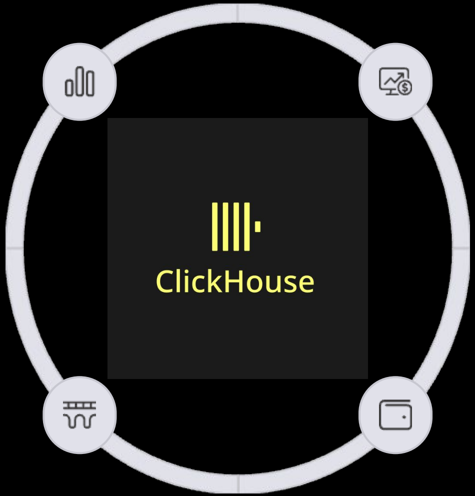
Primary Use Cases

Performant Charts

Pre–aggregated price data for responsive charting.

Bridging + Order Execution

Enable users to bridge and trade assets across several chains.



Trending Tokens

Real–time popularity metrics across all chains.

Cross–Chain Balances

Tracking net positions across multiple blockchains by indexing all transfers.

Use Case: Data Engine

1. Serves as a source of truth for market data across lots of blockchains
   1. Statistics
   2. Trade-feed
   3. User transfers
   4. Etc ..
2. Needs to be able to process 48h of data extremely quickly
3. 48 hours of data
   1. Can be 200m-250m rows, extremely time consuming to process on a per-row basis
4. Use materialized views to compress this data into 4-5m rows
5. End Result
   1. Rapid bootstrap, 60-70s to bootstrap 48h of market data

Materialized Views Pipeline

table.sql

Creates a table to store materialized view results

mv.sql

Defines materialized views for trending tokens, OHLC charts, etc

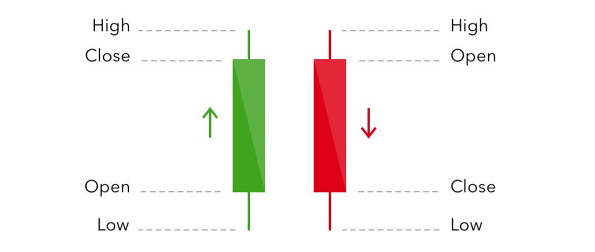
hydrate.sql

Bootstraps historical data into materialized views

view.sql

Creates the interface for querying pre-aggregated data

Materialized Views: OHLC



How We Generate Real-Time OHLC Charts

Data Source: On-chain trade events across all blockchains

Ingestion: Hermaus (Rust indexer) captures trades in real-time

Storage & Processing: ClickHouse materialized view pipeline

Step 1: Specialized Table Structure

- Create dedicated table with optimized schema for time-series financial data
- Leverage ClickHouse AggregatingMergeTree engine for efficient OHLC calculations
- Define columns for dimensions (token, pool, exchange) and time buckets

Step 2: Materialized View for Continuous Aggregation

- Transform raw trade events into time-bucketed price data (1m, 5m, 15m, etc.)
- Apply specialized aggregate functions to calculate open, high, low, close values
- Automatically update as new trades are indexed by Hermaus

Step 3: Flexible Query Interface

- Create parameterized views that support different time intervals
- Implement efficient data access patterns for chart rendering
- Optimize for frontend requirements with minimal data transfer

Thank you.

https://azura.xyz

Questions?

advaith@azura.xyz