# Building a Solana Geyser plugin with MongoDB

## OBJECTIVE

Build a Solana Geyser plugin with MongoDB to integrate Solana's real-time data streaming capabilities with MongoDB's flexible document-oriented database and perform benchmark tests against existing solutions.

As a geyser plugin is available for PostGresSQL, this is being taken as a reference for the implementation.

## Why MongoDB?

PostgreSQL faces challenges with horizontal scaling and efficiently handling diverse data types, and it can lead to performance bottlenecks as a result of its architecture. MongoDB is a more flexible, general-purpose database built for mission-critical use cases that require robust security, durability, availability, and performance at global scale. Its scale-out architecture seamlessly handles sharding across data centers and cloud regions, ensuring low-latency access worldwide. MongoDB’s document data model allows users to work with and manage dynamic, multi-structured data on the fly, while still enabling schema constraints when necessary. Its document model also handles semantic search, vector search, geospatial, graph, and time series data which can be optimal for parallel and concurrent high-volume reads . Moreover, MongoDB offers sharding, columnar compression, densification, deletes, and gap-filling for time series collections.

## The Work Done So Far

1. Sorting out dependencies and libraries from MongoDB, Solana Geyser plugin interface and core rust, for data serialization and asynchronous communication.
2. In parallel, taking a deeper understanding of the AccountsDB model.

## The Next Steps

1. Implement the MongoDB client with geyser plugin interface
2. Benchmark test its read performance against existing implementations.
3. Benchmark test its write performance against existing implementations.

## Challenges

* Understanding geyser plugin interface
* Figuring out all required dependencies for implementation of MongoDB client for our use case
* Identify application workload. Map schema relationships (whether to link or embed related data)
* Apply schema design patterns to optimize reads and writes
* Create indexes to support queries
* Serializing solana data in BSON
* Streaming data through the plugin

## Required resources, clarifications

1. A live demonstration of geyser plugin functionality in real-world/simulation, will enable us to better understand its behaviour..
2. Do we know of validators that are already using geyser plugins, and any direction from them? The API’s that they use for connections
3. Does geyser provide API’s relevant to account replication, or does it move away from this? Need details of which API’s are relevant, which aren't
4. What data s required to be streamed & stored via Geyser: Transactions? acc data? Slot or block info?

## Reading Sources

https://blog.syndica.io/sig-engineering-part-3-solanas-accountsdb/

https://www.anza.xyz/blog/a-deep-dive-into-solana-s-accountsdb

https://solana.com/news/sealevel---parallel-processing-thousands-of-smart-contracts

<https://solana.com/news/sealevel---parallel-processing-thousands-of-smart-contracts>

https://github.com/anza-xyz/agave/blob/master/geyser-plugin-interface/README.md

Geyser Plugin -<https://docs.solanalabs.com/validator/geyser>

Velos: Light weight version of Geyser - <https://github.com/hexishq/velos>

New:

<https://www.helius.dev/blog/solana-geyser-plugins-streaming-data-at-the-speed-of-light>

<https://dev.to/valentinmadrid/a-guide-to-solana-geyser-plugins-452k>

<https://github.com/JamesMDunn/solana-geyser-plugin-scaffold/blob/main/README.md>

## Additional Notes and Rough Work

Geyser plugins allow developers/infra providers to stream real-time data such as accounts(state), blocks, slots, and transactions. Solana is an eventually consistent poorly indexed database with complicated write semantics that aims at high throughput read and write on multiple transaction isolation levels. GP extends Solana’s functions by enabling applications to consume on-chain data directly instead of repeatedly querying rpc nodes.

**Main goal of the geyser (geyser was invented to rebuild the accountsdb outside of the validator):**

serve(write), read and store outside of a validator

Consideration:

Ensure each and every account update is processed accurately. There cannot be any undetectable drift from the internal solana accounts state.

Maintain all the logic that the validator does to resolve forks- optional if we want to deal with only “confirmed” or “finalized”

Solana Geysers plugin that currently exist:

* **PostgreSQL Plugin**: for managing and querying data using PostgreSQL
* **gRPC Service Streaming Plugin**: for streaming Solana account updates to a gRPC service
* **RabbitMQ Producer Plugin**: for facilitating message queuing with RabbitMQ
* **Kafka Producer Plugin**: for streaming data using Kafka
* **Amazon SQS Plugin**: for message queuing that leverages Amazon’s Simple Queue Service
* **Google BigTable Plugin**: for managing and querying data using Google BigTable

**Overall research Plan:**

1. **Geyser plugin** is available for PostGresSQL -> But Mongodb’s model and design is more suitable for highly critical distributed systems. Therefore we aim to reimplement the geyser plugin for MongoDB and benchmark tests against existing implementation.
2. **Optimize key-value pairs:** How to optimize key-value pairs, for faster read writes?
3. **Latency:** Ideas to reduce latency? Higher data read/write rates?
4. Check which components of AccountsDB are needed on-chain, which could be off-chain, assess the access prioritization
5. **Archival Strategy:** Check Archival Strategy (could be low prio)
   1. Categorize data based on age, and prioritize access accordingly??
   2. CPU cache - has most freq accessed info
   3. RAM index
   4. Disk index

Info: Metaplex (indexing), helius (), issue affecting photon light client, question of verifying validity of a data, product from triton () - old faithful? ([github.com/rpcpool/yellowstone-faithful](http://github.com/rpcpool/yellowstone-faithful)), question of trusting snapshots

1. **Load balancing:** Any ideas of Virtual Load Balancing
   1. Opinion from Jessica, Nate - traffic distribution to RPC is not equal, clearly indicating that Node balancing is not happening/is nonoptimal
2. **Drawbacks of Current Architecture:** 
   1. Check the necessity having replicas of AccountsDA
   2. Un-optimal indexes
   3. Is there a workaround that goes away from storing empty blocks
3. **Concurrency mechanisms -** Optimistic concurrency locking & time stamp ordering

**Focus Area 1 - Geyser Plugin: Core functionality:**

* Geyser plugins acts as a bridge between Solana and external data storage solutions
* Why? - Provide low latency access to solana- GP addresses this issue by redirecting information about accounts(state), blocks, slots, and transactions (ACID). This enables RPC services to offer more flexible and targeted optimizations.
* With GP, devs can offload a significant portion of data management tasks from validators while ensuring validators remain synchronized with the network, irrespective of RPC traffic volume

**Focus Area 2 - Improving Indexing?**

* Fragmentation, Lossless joins?
* Focus is mostly on optimizing transmission/storage egress data-
* Keep in mind possibilities of skipped slot and manage update to data storage system accordingly

**Question:** Set Objective - Improve the AccountsDB or rethink the whole thing and why?

→ improve on high availability - verify its HA architecture

→ faster read and write

→ optimize replication of data within the network

→ reduce high hardware requirements?

→ think 15 yrs from now, would the current architecture stand the test of time?

**Open Questions:**

1. Does the rate of block time generation have an impact / relation to performance? I.e. faster block time = better performance ??

Ans: may not be on performance, but enquire

It could have an influence on the network, as the faster a block is produced, the faster the process moves to the next work at hand.

1. Does ZK compression help in improving performance?

**What from the current architecture could be challenged?**

1. Solana’s clock? POH?
2. Leader Mechanism?
3. Need for slot replication? - Geyser plugin attempts to avoid this

—-----------------------------------------------------------------------------------------------------