Adapting Project Hamilton OpenCBDC to Leverage Oracle Blockchain Tables

Group Members: Nate Cable, Zee Khan, Dan McClellan, Bella Wang
Advisor: Professor Andrea Smith

CSB/CSE
Capstone Project
2023

Abstract

Central Bank Digital Currencies (CBDCs) are continually growing in popularity around the world, with many countries beginning to adopt their own versions of this type of currency.

OpenCBDC, a collaboration between Federal Reserve Bank of Boston and MIT, is an open-source project exploring CBDC design, with a GitHub repository.
The original project utilizes in-memory storage for key data, leaving the possibility of improvement through the use of Oracle's wide variety of database tools and functions.

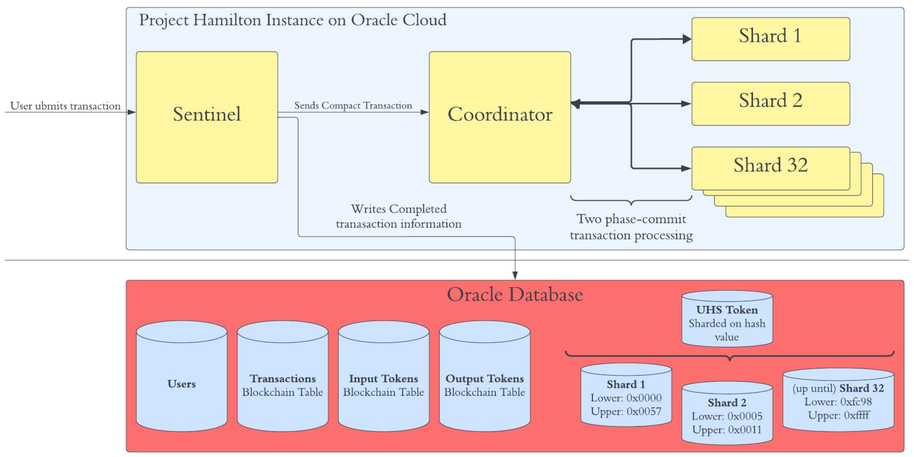
The ultimate objective is to port, test, and benchmark OpenCBDC architecture to Oracle DB, leveraging Oracle's database tools and functions for a more efficient platform.

Motivation: Project Hamilton

The goal of this project was to integrate Project Hamilton with Oracle Cloud Services. This instance of Project Hamilton on Oracle Cloud processes transactions sent by users and records the transaction into blockchain tables. To achieve this goal, we had to modify our instance to write to a backend that records transactions into a blockchain table.

The Project Hamilton architecture is broken up into several servers that communicate with one another via RPCs, the sentinels, controllers, and shards. The **Sentinel** verifies transactions submitted to it by the user. It ensures the sends owns the input tokens along, with other checks. Once verified, the sentinel passes the transaction the the **coordinator** to process. The coordinator utilizes 2PC with the **shards** to process transactions.

Our project writes to **blockchain tables** within Project Hamilton to store general and compliance iinformation about processed transactions.



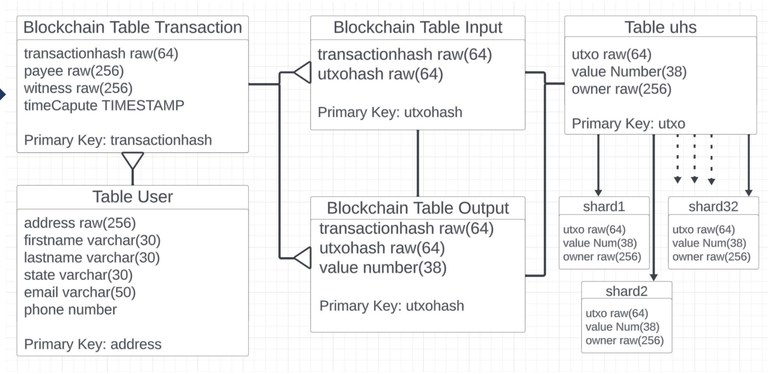
Our Solutions

Data Sending

- Important information tracked for each OpenCBDC transaction:
  - **Compact transaction hash**
  - **Payee information**
  - **List of input token hashes**
  - **List of output token hashes**
- Universal Hash System (UHS) token values associated with each hash
- Enables comprehensive tracking of each token from input to output
- Facilitates observation of cash flow within the system

Blockchain Tables and Schema

- Modified Project Hamilton instance to record completed transactions
- Transaction information is written multiple times for accurate tracking since coordinators and shards do not store token owner information
- Utilized blockchain tables for storing processed transactions to ensure security and transparency since blockchain tables are non-update and non-drop



Building to Project with OCI to connect Oracle DB

- Developed a C-Lang connector to interface with the Oracle Autonomous Database via Oracle Call Interface
- Connector enables data transmission to Oracle Database while OpenCBDC is running
- Real-time retrieval of data from the Oracle Database for the application frontend

Results and Conclusions

Achievements

1. **Building the project with OCI** to utilize Oracle Autonomous DB
2. Successfully **Sending Data from Sentinel to DB**
3. Blockchain **Table and its Schema Design**
4. **Expense Tracker UI and Server Development**

   The whole application is now deployed on Oracle Cloud Infrastructure (OCI): http://150.136.246.83:8000
5. **Admin UI Demo**

   Visualizing the total number of users, as well as the total number of transactions and payment table.

What we learned?

1. Proficiency in Oracle Tools and OCI Libraries

2. Technical Growth and Programming Skills Enhancement in C and C++

3. Enhancing Team Communication Skills Through Weekly Meetings

4. Holding in-person meetings and adopting a divide-and-conquer approach for task allocation are effective strategies to improve teamwork.

Based on what we've learned, what can we do differently?

OpenCBDC by itself is a very complex system that is created around the in-memory storage and the separate system architectures. If we were to do it again, we would likely build our own CBDC, rather than implement tools on top of an existing and specialized CBDC, as it would be a lot easier to develop with the tools we want to use in mind and not retrofit an existing project to use the tools.

Where would we extend this project?

1. **Continue Admin UI Development**: We constructed a framework for the Admin UI, primarily focusing on the structure without retrieving data from the database for diagram presentation. To further enhance this, it is crucial to address data filtering and sorting when executing the SELECT statement and fetching the data from the backend.

2. **Benchmarking** the entire implementation and comparing the metrics with those found in the OpenCBDC Whitepaper would definitely be good to implement. Discussions of this task were had many times during the course of this project, and a initial plan has been laid out and documented for future work on the project.