Section 1

Background
This Workshop and Me

- Worked in the space since 2018
- SC Engineer at Labrys and TracerDAO/Mycelium
- Team Lead Engineer at Mycelium delivering on-chain monitoring of oracle systems
- Spent 2 years working on Indexing and event processing pipeline solutions

Please ask questions as we go along!
Etherscan: Lots to see

- Clicking through and exploring is hugely beneficial, you’ll always discover something new/interesting
- Bot attempting to quickly fanout eth to multiple addresses (not a contract but an EOA!)
Observability and Transparency: Pros and Cons
Solana DeFi Trading Platform Mango Markets Loses $100M in Hack

The latest hack comes less than a week after BNB Chain lost $100 million.
What is an Indexer?
Extract: Request data from a Node
Transform: Change its form to be easily human/machine readable
Load: Insert it to have a better retrieval strategy or pass it to another service
Why Make an Indexer?

**Retrieval Time**
- Creating a unique index allows quick retrieval
- Reduce node requests

**Example:**
Index to fetch the transaction history of an address

**Non-Permanent Data**
- Store mempool history for developing arbitrage strategy
- Store and simulate transactions and their results
- Gas estimation history of local development environment
Breaking Down an Example: Chainlink

- Even decoded it's difficult to tell what's going on
- Some points are obvious: answer
- Some are very confusing (i.e. observers)
Step 1: Understand the System and the Contracts
Chainlink: Off-Chain-Reporting

- Instead of having each node individually send a transaction with their “answer” one “transmitter” will send everyone’s data on their behalf.
- This reduces gas usage; Chainlink used to utilize a majority of Mainnet’s bandwidth by itself!
- The transmitter also aggregates (takes a median) of everyone’s answer and that is the price of the feed for that “round”
Here’s the timeline.

**Price Deviation**
Chainlink oracles detect a price deviation and a round is initiated. All oracles assigned to the feed retrieve a price from their subscribed APIs.

**Leader Collects**
The round leader collects all of the different answers from each oracle.

**Submit On-Chain**
The answer is submitted on-chain by the transmitter as well as every oracles individual answers.
More Legwork:

- Observers tell us who is who
- Observations are too large to make sense: a multiplier is being used.
Adding Complexity: Block by Block Changes

24. transmitters

The list will match the order used to specify the transmitter during setConfig

Return:
list of addresses permitted to transmit reports to this contract

Multipliers and transmitters can change every block!
### 24. transmitters

The list will match the order used to specify the transmitter during setConfig

```
0x8b1d49a93a84b5d0917a1ed42d8a3e191c28524,0x0312ea121df0a323ff535bf753172736cc9d53d13,0x218b5a7861d8f368d09a84ed0b1f6c6dddbf99db8,0x9cfa1b513fffa293e7023159b3c7a4c984b6a3480,0xc4b732fd121f2f3783a9ac2a6c62fd535fd1f3fda,0xfa0e4f48ae369bb3ecbc8e60b511937ea81bfc29,0xc29be4ca92d4ec43c8451fa94c200b839916,0xc774e67bfcd623c803d48afce74a09e6ff6ba599003,0xf16e77a999529a4c58318aee8a1548df3fccc1,0x88782222f9a2e3c30cde77d28191f8d298b3,0xf07131f578ae57f08a2e2ccbc9fa98458099e0fffb4,0xca859e48f6ce9834a119ba04fdcc53c1d4f1082a7,0xe3ed596ac55ae6044b75bab27426f7d9e018d4,0xd22cc87d7a3f12dcb75cebda2e96f6768ae114f,0xb6f7078a848974623a968d8bdc8e0a35e657c26,0xe3cd128883f2934d78923467b67ea7c4f257c48
```

**Return:**

list of addresses permitted to transmit reports to this contract
Database vs Node Retrieval

- Getting this data from a node is very complex and expensive
- Instead of all of this processing, eth calls, and fetching we could instead request this data through SQL
- Imagine trying to get the average oracle accuracy for a specific oracle on Wednesdays through node requests!
Blocks, Transaction, Receipts, Addresses
Dissecting A Log

- Topics 1-3, Indexed data (searchable by the node)
- Data: Unlimited space
- Block Number
- Tx Index
- Log Index
- Removed
- This defines the log

```
Topic[0] = keccak256(event_name(type1, type 2, ...))
```

Careful: This is only unique for each contract!
Storage: Private Variables are not as Private as you Think

- Can access any storage of a contract even if the variable is marked private
- Extremely useful for finding contracts using the EIP-1967 Proxy pattern
Infrastructure Design

- Node 1
- Node 1
- Node 1

- AI Analysis
- Events Subscription Service
- EPNS
- Database
Section 3

Code Walkthrough
Golang and Geth

- Fast
- Extremely well maintained
- Safe parallelisation
- Geth calls are portable to most EVM chains

This will allow us to deploy our program across multiple chains since they will adhere to the RPC specification in the yellowpaper.
Creating a Client

```go
client, err := ethclient.Dial(rpc)
if err != nil {
    log.Fatal("Failed to connect to the websocket of the Node (RPC)", err)
} else {
    fmt.Println("successfully connected to the RPC endpoint!")
}
```
Websockets vs Http

WebSocket Connection

Client → Request → Hand Shake → Web Socket → Server

VS

HTTP Connection

Client → Request → Response → Connection Terminated → Server
Creating a Query

```javascript
contractAddress := common.HexToAddress("0x60Ae865ee4C725cd04353b5AAb364553f56ceF82")
query := ethereum.FilterQuery{
    Addresses: []common.Address{contractAddress},
    Topics: [][]common.Hash{[common.HexToHash("0x44403e38baed5e40df7f64ff8708b076c75a0dfda8380e75df5c36f11a476743")],
    }
```
FilterLogs vs Subscribe Filter Logs

```go
sub, err := client.SubscribeFilterLogs(context.Background(), query, logs)
if err != nil {
    log.Fatal(err)
} else {
    fmt.Println("successfully subscribed to the contract events!")
}

historiclogs, err := clientH.FilterLogs(context.Background(), historicQuery)
```
Creating a Channel

```go
logs1 := make(chan types.Log)

for {
    select {
        case err := <-sub.Err():
            log.Fatal(err)
        case vLog := <-logs1:
            //Do Processing
    }
}
```
Generating an ABI

colc --abi events.sol

====== events.sol:Events =======
Contract JSON ABI
[
  {"anonymous":false,"inputs":[{"indexed":false,"internalType":"string","name":"name","type":"string"},{"indexed":false,"internalType":"string","name":"symbol","type":"string"},{"indexed":false,"internalType":"uint256","name":"timestamp","type":"uint256"}],"name":"BaseInitialized","type":"event"},
  {"anonymous":false,"inputs":[{"indexed":true,"internalType":...}]}
]
Making an ABI Object and Unpacking

```go
contractABI, err := abi.JSON(strings.NewReader(ABI_String))
if err != nil {
    log.Fatal("could not convert JSON ABI string to ABI object")
}

Interfaces, err := contractABI.Unpack("MY_EVENT_NAME", my_data)

MyBigInt := Interfaces[0].(*big.Int)
```
Working with a DB: Making Tables

type User struct {
    ID     string `gorm:"primaryKey"`
    Address string
}

type CommentMessage struct {
    MessageID                uuid.UUID `gorm:"primaryKey"`
    Sent                     bool
    ProfileId                decimal.Decimal
    PubId                    decimal.Decimal
    ContentURI               string
    ProfileIdPointed         decimal.Decimal
    PubIdPointed             decimal.Decimal
    CollectModule            string
    CollectModuleReturnData  string
    ReferenceModule          string
    ReferenceModuleReturnData string
    Timestamp                decimal.Decimal
}
db, err := gorm.Open(postgres.Open(dsn), &gorm.Config{})

// panic if we cannot connect to the database
if err != nil {
    panic("failed to connect database")
} else {
    // or else we are good to go
    fmt.Println("Connected to database")
    fmt.Println(db)
}

db.AutoMigrate(&database.User{})

db.AutoMigrate(&database.CommentMessage{})
```go
myvar := database.FollowMessage{
    MessageID: uuid.New(),
    Sent: false,
    ProfileId: profileID,
    FollowNFT: followNFT,
    Timestamp: TimestampDecimal,
}

ProfileIDBI := ProfileIdInterface[0].(*big.Int)
profileID := decimal.NewFromBigInt(ProfileIDBI, 0)
followNFT := common.HexToAddress((Topics[2].Hex()).Hex()}
```

```sql
db.Clauses(clause.OnConflict{
    UpdateAll: true,
}).Create(&myvar)
```
Thank you!

Benjamin Memisevic

@MemiHack_eth