The Nuts & Bolts of Blockchain

Thomas Catalano

www.CONCEPTSLEARNINGMACHINE.com

@tomsnode

Blockchain record state and history state is needed by transactions.

Transactions are signed, integrity resigned, integrity protected by the chain.

Double Spending Problem

Solved by the central authority

Bank

HashCash

Bmoney- server trust

A double spend is an attack where the given set of coins is spent in more than one transactions.

Send two conflicting transactions in rapid succession on the network. This is a race attack.

Mining solues the distrabution and Double Spending Problem longest chain wins.

Calculate the hash solution in N time.

UTXO

Upsent transaction output

Responsible for beginning and ending each transaction.

Confirmations of transaction results in the removal of spent tokens from the UTXO database.

Ledger

Ledger replaces UTXO

secure hash chain

Contains transactions

Contains metadata

Binary Formats

Look out in certain programming langs binary conversions do not convert back to the same value

JSON – binary – JSON
does not convert back with the same value

Block of Chains

A block of one more new transactions is collected into the transaction data part of a block.

A merkel root is stored in the block header. Each block also stores the hash of the previous blocks header, chaining the blocks together. This ensures a transaction cannot be modified without modifying the block that records it and all folowing blocks.

Consensus

First on agreement, second on correctness.

Changed by social consensus.

Public blockchain must be a fortress

Code is public

Public API

Makes development 10x slower

Byzantine General Problem

If each sides message isnt read by the other and is required to respond before action which one will respond first.

Performance

Some tasks are parallel

Consensus is a BG problem

Blockchains do not scale horizontally

Public/Private key parallelism

Isolation

Transactions must be determinstic

Deterministic by isolation

Move semantics

Transfer ownership of resources from one object to another.

Lambdas

Allows to preserve layering

Defered & Dispatched work

This is to be able to set the lock and release the lock in the database to prevent data corruption.

Lambda expression

Is a convenient way to define an anonymous function

{=} () mutable throw() - > int
{
Int n = x + y:
x = y
y = x
return n:
}

Code Isolation

Namespaces

Seperation of Implantation analysis.

API for user, API for deviation.

The Slicing Problem

Assigning a derived class object to a base class variable, assigning a derived class object to a base class object slices off data.

TransactionA vtransactionA;
TransactionB vtransactionB;
vtransactionA = vtransactionB

VtransactionB cannot be calling objects for a member function from TransactionA unless the function is also a member function of TransactionB and all member variables of vtransactionB, and all the member variables of vtransactionA that are not inherited from the class TransactionB are lost

Slicing Problem

Simply making a member function virtual does not defeated the slicing problem.

In order to defeat the slicing problem the function must be virtual and you must use pointers and dynamic variables.

Not great solutions

Raw Pointers

Unique Pointers

Shared Pointers

Clone Idioms

Use of strong & weak pointers

Cache holds strong and weak pointers.

Time denotes a strong pointer and frequently used objects lock the cache.

Algorthmic complexity attack

Salted Hash

A cryptographic salt is made up of random bits added to each password instance before the hashing. Salts help us mitigate rainbow table attacks by forcing atackers to re-compute them using a salt.

Key / Value

Hash Trees

Find by a key

Bucket count dynamic

template

Template

```
Template < class Codec, class Function >
Bool
Visit (
path_type const& path,
std::size_t read_size,
Function&& f)
```

Templates give you concepts and they can‘t slice

Set of rules for a concept to run

Compile time polymorphism, templates

std::variant c++17

class template represents a type-safe union

allocates a fixed portion of memory and reuses it to hold a value of one of several predefined alternatives types at a time.

Compile time polymorphism

Overloading

Polymorphic code fully optimized and inlines

When more than one method share the same name with different parameters.

Class template optional is a wrapper representing optional or nullable objects who may not yet contain a valid value.

Optional objects offer full value semantics; they are good for passing by value and usage inside STL container.

Compile time polymorphsim

```
template < class Th, class TOut >

class TofferStreamBase
{
Protected:
Toffer <Tout> offer_;
boost::optional<TOut> overFunds;
```

The function signature means that the functions can either return a value of type int or a flag indicating that no value of int is available. This does not indicate a error. It is like one additional value of int.