Solana Runtime on a Solana Validator

The runtime is a concurrent transaction processor. Transactions specify their data dependencies upfront and dynamic memory allocation is explicit. By separating program code from the state it operates on, the runtime is able to choreograph concurrent access. Transactions accessing only read-only accounts are executed in parallel whereas transactions accessing writable accounts are serialized. The runtime interacts with the program through an entrypoint with a well-defined interface. The data stored in an account is an opaque type, an array of bytes. The program has full control over its contents.

The transaction structure specifies a list of public keys and signatures for those keys and a sequential list of instructions that will operate over the states associated with the account keys. For the transaction to be committed all the instructions must execute successfully; if any abort the whole transaction fails to commit.

Account Structure

Accounts maintain a lamport balance and program-specific memory.

Transaction Engine

The engine maps public keys to accounts and routes them to the program's entrypoint.

Execution

Transactions are batched and processed in a pipeline. The TPU and TVU follow a slightly different path. The TPU runtime ensures that PoH record occurs before memory is committed.

The TVU runtime ensures that PoH verification occurs before the runtime processes any transactions.

At theexecute stage, the loaded accounts have no data dependencies, so all the programs can be executed in parallel.

The runtime enforces the following rules:

1. Only theowner
2. program may modify the contents of an account. This means that upon assignment data vector is guaranteed to be zero.
3. Total balances on all the accounts are equal before and after execution of a transaction.
4. After the transaction is executed, balances of read-only accounts must be equal to the balances before the transaction.
5. All instructions in the transaction are executed atomically. If one fails, all account modifications are discarded.

Execution of the program involves mapping the program's public key to an entrypoint which takes a pointer to the transaction, and an array of loaded accounts.

SystemProgram Interface

The interface is best described by theInstruction::data that the user encodes.

- CreateAccount
  - This allows the user to create an account with an allocated data array and assign it to a Program.
- CreateAccountWithSeed
  - Same asCreateAccount
- , but the new account's address is derived from* the funding account's pubkey,
  - a mnemonic string (seed), and
  - the pubkey of the Program
- Assign
  - Allows the user to assign an existing account to a program.
- Transfer
  - Transfers lamports between accounts.

Program State Security

For blockchain to function correctly, the program code must be resilient to user inputs. That is why in this design the

program specific code is the only code that can change the state of the data byte array in the Accounts that are assigned to it. It is also the reason whyAssign orCreateAccount must zero out the data. Otherwise there would be no possible way for the program to distinguish the recently assigned account data from a natively generated state transition without some additional metadata from the runtime to indicate that this memory is assigned instead of natively generated.

To pass messages between programs, the receiving program must accept the message and copy the state over. But in practice a copy isn't needed and is undesirable. The receiving program can read the state belonging to other Accounts without copying it, and during the read it has a guarantee of the sender program's state.

Notes

- There is no dynamic memory allocation. Client's need to useCreateAccount
- instructions to create memory before passing it to another program. This instruction can be composed into a single transaction with the call to the program itself.
- CreateAccount
- andAssign
- guarantee that when account is assigned to the program, the Account's data is zero initialized.
- Transactions that assign an account to a program or allocate space must be signed by the Account address' private key unless the Account is being created byCreateAccountWithSeed
- , in which case there is no corresponding private key for the account's address/pubkey.
- Once assigned to program an Account cannot be reassigned.
- Runtime guarantees that a program's code is the only code that can modify Account data that the Account is assigned to.
- Runtime guarantees that the program can only spend lamports that are in accounts that are assigned to it.
- Runtime guarantees the balances belonging to accounts are balanced before and after the transaction.
- Runtime guarantees that instructions all executed successfully when a transaction is committed.

Future Work

- Continuations and Signals for long running TransactionsPrevious Validator Gossip Service Next Validator Geyser Plugins