AIP[1]-014: Critical Resources of Blockchain Services

platfowner 2021-03-06

Goals

- Define AIN Blockchain's critical resources available for the blockchain service users
- Define how to track them in transparent manners
- And find and set reasonable limits for them to keep the blockchain services healthy

Problem Definition

In blockchains, there is a hard limit on the block size (see Bitcoin Scalability Problem). Ethereum for example has a block gas limit, which is a hard cap on the block size in gas. Here block size is an example of critical resources of blockchain services. In this document, we define critical resources of AIN Blockchain, the ways to track them in transparent manners, and find and set reasonable limits for them from the system stability perspective.

We have several major questions:

- What are the critical resources of blockchain services in terms of the service stability and performance?
- How to measure the maximum allowable values of the critical resources?
- How to track them and set limits for them?

Requirements

Critical resources have the following properties:

- Containing their use should help the service stability and performance
- Should be well-defined, deterministic, and reproducible with public data, i.e., depending only on public data (e.g. blocks, DB states)
- Fewer would be better for simplicity

[1] AI Network Improvement Proposal. Visit https://docs.ainetwork.ai for the full list.

Proposed Design

Key Ideas

Candidates of AIN Blockchain's critical resources are:

- Data size -> disk
    - Transaction size
    - Block size
- Bandwidth (qps) -> computation, network
    - Write operations per block
    - Read operations per second
- State DB -> memory
    - Depth of states
    - Number of states
- Others
    - Tx pool size

Design Details

Critical Resources

Type	Resource	How to track?	Where to store?	How to limit?	Limit type (unit)
Data	Transaction bytes	Client API	-	In client API handling code	Independent (bytes)
	Block size	Block proposal	In block	In block creation code	**Dependent (# of write ops / block)**
Bandwidth	**Write operations**	Transaction execution	In transaction, In block	In block creation code	**Dependent (# of write ops / block)**
	Read operations	Client API	-	In client API handling code	Independent (qps)
State DB	Depth of states	Transaction execution	In state	In transaction execution code	Independent (path length)
	Number of states	Transaction execution	In state	In transaction execution code	**Dependent (# states)**
Others	Transaction pool size	Transaction pool	In transaction pool	In transaction execution code	Independent (# of txs)

Number Tracking

DB Write Operation

The DB write operation, which is used as a basic unit of the bandwidth and the block size, is measured as follows:

- Bandwidth = # of DB write operations per second where the write operation here includes all those executed by the native functions triggered by the transaction
- The number of the write operations of a transaction is measured when it's actually executed in DB and saved in the Transaction object
- The number of all the write operations of the transactions in a block is counted when the block is created and is saved in Block object for block validation

The measured write operation metric is saved as an extra field of each transaction, and is used to control the maximum bandwidth and the maximum block size as follows:

- The block size is contained by posing a hard limit on the maximum DB write operations of the transactions in a block
- With a consensus algorithm of a constant block creation rate (usually 1 block in N seconds), the bandwidth is contained accordingly

State Depth and State Size

The state node's depth or subtree size is automatically updated whenever a write operation is executed for those nodes affected by the operation. So the depth or the tree size of the whole DB is easily obtained from the root node of the DB. Similarly app level depth or tree size is also easily obtained from the app's root node.

The number of states are controlled as follows:

- Any transactions that make the size of the state database exceed the limit is rejected to be added to the transaction pool

Measuring Maximum Allowable Values

How to measure maximum allowable values?

- Independent measuring:
    - While other resources are kept to be normal, a specific resource is increase until the blockchain service turns unstable
- Dependent measuring:
    - Given the range of values obtained from the independent measuring,
    - Sets most independent resource values (e.g. data, state DB) first and the less independent resources values (e.g. bandwidth) later
    - Use more advanced optimization techniques like the Monte Carlo method.

Milestones

- Step 1: Implement tracking features
- Step 2: Extend the load testing tool and do the independent measuring
- Step 3: Extend the load testing tool and do the dependent measuring
- Step 4: Set limits with the measured values

Conclusion

- Provided a candidate list of AIN Blockchain's critical resources
- Provided high-level ideas of how to track them
- Provided high-level ideas of how to measure their maximum allowable values
- Provided milestones for phased execution

Links

- Bitcoin Scalability Problem (wiki)
- Ethereum Block Gas Limit (link)
- Monte Carlo method (wiki)

Document History

Date	Who	Change	Notes
2021-03-06	platfowner	Initial draft	
2021-03-08	platfowner, cshcomcom	Synced with csh@	
2021-03-15	minsulee2, liayoo, cshcomcom, platfowner	Internal review	
2021-03-25	platfowner	Number Tracking section	
2021-05-12	platfowner	Github IDs Link to full list	
2021-05-12	platfowner	Published	