Celo Mainnet Stall, July 13-14, 2022

cLabs Blockchain Team

Start, End	7/13/2022 - 11:20 pm UTC 7/14/2022 - 11:42 pm UTC
Impact	• Full cessation of all block production during the affected period. • No application deployed on Celo was able to have transactions successfully committed, although the entire blockchain state remained available for read/view access • Based on the average TPS at that time, approx. 400k transactions were delayed because of the stall • Since blockchain time “stop” for 24hrs, validators were unable to earn 1 day of rewards (~16,800 cUSD for the whole validator set)
Root cause summary	• High Transaction Load induced the network to delay consensus and generate a big enough block (in bytes) that produced network to stall • Stall was not a consensus protocol bug, but a hardcoded limit on network messages byte size

Timeline and Impact

7/13/2022 - 11:20 pm UTC	Network stalled at block 14035019
7/13/2022 - 4:45 am UTC	**First attempt to resume.** Plan to restart every validator with a reset of roundState DB started
7/13/2022 between 5:15 am UTC and 5:45 am UTC	**First attempt failed**. Quorum of validators restarted, but network did not resume block production
7/14/2022 - 8:39 am UTC	**Second attempt to resume.** Proposer of the first attempt performed the restart & reset of roundState DB again, and network resumed briefly; and stalled again on block 14035045

7/14/2022 - 12:08 pm UTC	**First attempt to fix.** Release 1.5.7. With an increase in the p2p protocol message limit. After a quorum of validators upgraded, network did not resume
7/14/2022 - 10:24 pm UTC	**Second attempt to fix.** Release 1.5.8. Which refactors `preprepare` message encode/decode logic to reduce block duplication on it.
7/14/2022 - 11:42 pm UTC	**Second attempt succeeded.** Quorum of validators upgraded. Network resumed after proposer for current round reset roundStateDB (triggering a `preprepare` message).

Related docs:

- Round State Shareout (After Reset)
- Celo Network Resetting Instructions
- https://github.com/celo-org/celo-blockchain/releases/tag/v1.5.7
- https://github.com/celo-org/celo-blockchain/releases/tag/v1.5.8

Root Causes

- Network stalled because the `preprepare` message for the current block height was bigger than `16mb`; which was a limit enforced by the network layer
- Preprepare size was due to:
    a. Block was big in bytes (~1mb); due to considerable amount of transaction calldata use.
    b. `Preprepare` contained multiple `prepareCertificates` from previous rounds and each contained another copy of the block
- **Condition (a)** was made possible by the increase of block gas limit, as a result of governance proposal 57
- **Condition (b)** is part of the consensus protocol, still “odd” as it suggests that the network was able to reach a quorum of `prepare` messages, but unable to reach a `quorum` of commit. Which, by protocol, can only happen due to a timeout.
    a. Could be argued, that timeout was induced by the size of the block, and validators being unable to process the block in time.
- On network layer limitations:
    a. The p2p protocol layer, defines a 10mb limit on message size (see code)
    b. The RLPx transport protocol, defines a 16mb limit on message size (see code)

- On first & second attempts:
    a. First attempt. The problem was correctly identified: message size; but this was not enough, because only the p2p protocol limit was increased. Only after lifting this limit was that the RLPx limit appeared in scene
    b. Second attempt. Increasing the RLPx limit was tricky and potentially not feasible. Decision to refactor serialization logic on the `preprepare` message. This was not decided before as it was deemed “dangerous” as a change.

Reflection

Previous similar incidents

- None

Things that went well

- Validator community responsiveness was remarkable. Validators were quick to respond, either to execute on the fixing attempts, but also to provide logs, debugging data, or jump into a call with the core developers to inspect the state of the validator.
- The Core developers detected the problem, and fixed it; in a setup where they had little observability of the internal state of validators.
    a. Even if the first attempt failed, the problem was already correctly identified. The solution was only partial

Things that didn’t go well

- There was no “incident playbook” for this type of network stall already in place .
- Observability of the consensus protocol is quite poor. Log messages or RPC endpoints provide little information about the internal state of the consensus protocol.
- Even if round changes exponential timeout guarantee eventual restart (after the fix); they are also impractical. After consecutive failed rounds, the network will have to wait hours or days to do a new consensus round.
- Due to the decentralized nature of the protocol. Difficulty to access logs or metrics from validators, introduces complexity to the debugging exercise. To access logs, or state, it was required to contact validators one by one; it worked, but could be improved.
- Stress testing for CGP-56 didn’t include scenarios with `prepareCertificate` or big blocks (in bytes); thus it was unable to catch this scenario

Questions

- Should automated testing have caught this (e.g. testing DoS attacks by sending massive data transactions)?
    - Not for “data” attacks, yes for “execution” attacks. Type of issue that will require a solution.
- How better to coordinate response among validators?
    - [technical] how do we intentionally halt the network, if necessary?
    - Increased engagement for responsiveness / Disaster recovery simulation?
    - Given the decentralization and diversity of validators, how best to contact all validators?
- Could reduce time to recovery by just rolling back versions rather than trying to fix?
    - No, the problem was not due to some previous change in code. Encoding logic was there since mainnet.
- It was predictable that eventually there could be an issue like this that required large scale validator coordination and participation, and the system didn’t have the capacity to think through and prepare for this. Are there other predictable issues that can be prepared for now?

Actions

1. Mitigate immediate risks of surpassing network limits by other means.
    a. A hotfix to decrease the Block Gas Limit to 20M was approved, to avoid this case.
2. Modify consensus or consensus messages, to avoid large messages during consensus.
3. Modify network layer, to increase max message size allowed.
4. Coordinate alongside with validator community to create an Incident Playbook to improve the response in the case of a future event