Growth Budget Recommendations

The general scope of this project is to define *safe* growth recommendations for V0 of Pocket Network. The recommendations are determined as of November 20th, 2021 with Pocket Core release RC-0.6.4.

Disclaimer: *The guidance below is subject to drastic change given future releases and new developments.*

1) Data

This section outlines the raw data used in the analysis. It does not attempt to provide any illustration of the recommended budget.

1.1) Merkle Sum Index Tree

```
Test_Name:
  Merkle_Tree
Pocket Core Version:
  RC-0.6.4
Environment:
  Name: Andrew's Macbook Pro 2017
  Specs: 2.9 GHz Intel Core i7 - 16GB RAM
  Cloud Provider: N/A
Methodology:
  Generate Relay Structures
  Use the Relays as leafs to the tree
  Generate the Merkle Root (Method 1 'to build the tree')
  Generate the Merkle Proof (Method 2 'to build the tree')
  Save the maximum time/memory benchmark
Data:
  1 Million Relays;  100MB of memory ~3 seconds to compute
  3 Million Relays;  285MB of memory ~14 seconds to compute
  5 Million Relays;  500MB of memory ~25 seconds to compute
  10 Million Relays; 1GB of memory but ~60 seconds to compute
Profiling:
  Go Tool PPROF; ZIP shared in discord
```

1.2) Block Size

Test_Name:
Block_Size
Pocket Core Version:
RC-0.6.4
Methodology:
Read the Tendermint Block Implementation
Apply Pocket Network parameters to code findings
Environment:
N/A
Data:
MaxTxBytes = 4MB (Block Size) -
(
653 Bytes (Header Size)
+
11 Bytes (Amino Overhead)
+
*NumOfValidators * 223 Bytes (Max_Vote Size)*
+
*NumOfEvidence * 484 Bytes (Max_Evidence Size)*
)
Profiling:
N/A

1.3) Relay RPC

Test_Name:
Relay_RPC
Pocket Core Version:
RC-0.6.4
Environment:
Name: Recommended Lab 2
Specs: 4vCPU - Intel(R) Xeon(R) CPU @ 2.80GHz - 8GB RAM
Cloud Provider: GCP
Methodology:
Create a Golang Relay Generator Client
Run a Single Node / Single App Network
Send relays at varying rates to determine ‘stability’

Data:

No Artificial Response Processing -> GanacheCLI:

695235 relays in 30m0.000715745s

= 386.241513 Relays Per Second

= 1.39 Million Relays Per Session

0 failed relays out of 695235 total = 100% success rate

CPU = ~55% Stable & Memory = ~42% Growing Slightly

Artificial Response Processing -> 5 Seconds Per Relay:

344735 relays in 30m0.000628043s

= 191.519378 Relays Per Second

= 689.5 Thousand Relays Per Session

85 failed relays out of 344735 total = 0.999753 success rate

Mainnet Validator

Artificial Response Processing -> 5 Seconds Per Relay:

62069 relays in 30m0.000628043s

= 34.482777 Relays Per Second

= 124,138 Thousand Relays Per Session

0 failed relays out of 62069 total = 100% success rate

Profiling:

htop

1.4) Claim/Proof Per Block

Test_Name:

Claim_Proof_Per_Block

Pocket Core Version:

RC-0.6.4

Environment:

N/A

Methodology:

Generated ‘Ceiling’ Structures

Marshal into bytes using proto

Measure bytes with Len()

Apply Block_Size & Max_Validators

Data:

1 Million Relay Proof = 1884 bytes

Largest Possible Claim = 401 bytes

Max_Validators = 1K (Predetermined)
Block_Size = 4MB (Predetermined)
Block_Time = 15 Min

= 3.77700MB space for Txs Per Block
= 1,652 Claim + Proof Per Block
= 6,611 Per Hour
= 6,611,000,000 Relays Per Hour

Profiling:
N/A

1.5) Session Generation

Test_Name:
Session_Generation
Pocket Core Version:
RC-0.6.4
Environment:
Name: Recommended Lab 2
Specs: 4vCPU - Intel(R) Xeon(R) CPU @ 2.80GHz - 8GB RAM
Cloud Provider: GCP
Methodology:
Using IAVL ONLY (Disabled ALL Caches)
Create IAVL Stores (Goleveldb)
Set Y Applications
Set Servicers
Run X Dispatch Requests in a row
Observe htop for resources
TimeTrack individual dispatch calls
Data:
Trial 1: 100K Servicers 5 Nodes Per Session; 100 Dispatches
AvgDuration: 120 ms
Resources: 136% CPU 4% memory

Trial 2: 100K Servicers 10 Nodes Per Session; 100 Dispatches
AvgDuration: 122 ms
Resources: 142% CPU 6% memory

Trial 3: 100K Servicers 50 Nodes Per Session; 100 Dispatches

AvgDuration: 123 ms
Resources: 142% CPU 6% memory

Trial 4: 50K Servicers 5 Nodes Per Session: 200 Dispatches
AvgDuration 57 ms
Resources: 135% CPU 3% Memory

Trial 5: 50K Servicers 10 Nodes Per Session: 200 Dispatches
AvgDuration 58 ms
Resources: 135% CPU 3% Memory

Trial 6: 10K Servicers 5 Nodes Per Session; 1K Dispatches
AvgDuration 8.5 ms
Resources: 128% CPU 2% Memory

Trial 7: 10K Servicers 10 Nodes Per Session: 1K Dispatches
AvgDuration 8.5 ms
Resources: 128% CPU 2% Memory

Profiling:
Htop

1.6) State Growth

Test_Name:
State_Growth
Pocket Core Version:
RC-0.6.4
Environment:
N/A
Methodology:
Analyze the impact of each Message
Generated 'Ceiling' Structures
Marshal into bytes using proto
Measure bytes with Len()
Data:
Max Validator = 1012 Bytes
- *Keys & Values for*
  - *AllVals*
  - *SigningInfo*

```
        - ValByChains (Max Chains)
        - StakedVals
        - PrevValPower
    NOTE: (Each additional chain adds 51 Bytes)
  Max Application = 273
   -  Keys & Values for
        - AllApps
        - StakedApps
     NOTE: (Each additional chain adds 6 Bytes)
Profiling:
  N/A
```

2) Analysis

This section draws conclusions and makes claims based on the raw data in section 1.

Assumptions*:*

- RC-0.6.4 Software
- MaxValidators = 1K
- 5 minute processing time for all Sessions within a block
- Session Generation will always be 50% or more of block processing
- Submission of the ClaimProof remains probabilistically uniform
- Service Node count will be between 6K and 1 Million

Relay Limits Ordered By Maximum Possible

- ClaimProof Txs Per Block
- Merkle Tree
- RPC

If RPC is the bottleneck in MaxDailyRelays, then let's start there.

- *100K Relays per Node per Hour at 30 relays per second*

What's the hourly hard limit on Claim+Proof (Active Nodes Per Session Period)?

- *6K Active Nodes per Session period*

So given these two strict limitations, what's the MaxDailyRelays possible for V0?

- *Theoretically, 14 Billion*

How much overhead does each ServiceNodes add?

- ~1.2 Microseconds for Session Processing
- ~1KB of State Data (Every block for RC-6.4)

Why do we care about Session Processing in terms of Block Time?

- Each Claim+Proof requires 1 session generation to validate the tx

# of Service Nodes are currently deemed limitless, what exactly does the # of nodes affect?

- # of ServiceNodes affects Session Processing & Blockchain Size

How quickly would the blockchain be growing at 1 Million Service Nodes

- ~1 GB per Block with RC-0.6.4

If 5 Minutes is the assumed Sessions processing time per block, what's max # of Sessions per hour?

- 1K simultaneous Sessions at 1 Million Service Nodes

Does NodesPerSession affect SessionGeneration times?

- NodesPerSession <= 100 is not statistically significant for block processing

How can NodesPerSession be leveraged for scalability?

- Less Apps & More Nodes Per Session is better for block processing & blockchain data
    - 1K Apps & 6 Nodes Per Session
        - 5 Minute Processing Per Block & 237KB of State Data
    - 500 Apps & 12 Nodes Per Session
        - 2.5 Minute Processing Per Block & 119 KB of State Data
    - 250 Apps & 24 Nodes Per Session
        - 1.25 Minute Processing Per Block & 59 KB of State Data

What is the effect to the node runner of increasing NodesPerSession and lowering apps:

- Gateway has more nodes to cherry-pick. Assuming the relays are NOT maxed, the cherry-picker will likely bias even more towards higher quality nodes compared to today.

3) Growth Budget

NumberActiveServiceNodes Per Session Hard

- 3K

NumberActiveServiceNodes Per Session Re-Evaluate

- 2K

Explanation: In RC-0.6.4, the block size is ~3.7MB for transactions. The amount of simultaneous Service Nodes active per session is limited by the block size in that each individual node must submit claim+proof transactions to receive rewards. If there are more than 3K active ServiceNodes per Session, the Block Size is unable to safely support the amount of Claim+Proof transactions the network is producing.

MaxRelays Hard

- 3 Billion Relays

MaxRelays Re-Evaluate

- 1 Billion Relays

Explanation: In RC-0.6.4, the amount of relays each individual node can do is currently limited by the RPC. The number of individual nodes submitting claim+proof transactions to receive rewards is limited by block size. Combining these two metrics, over 3 Billion relays the network will no longer be able to support the capacity either by exceeding the safe RPC rate-limit or exceeding block size.

App:Node Ratio Hard

- 1:12+ (less than or equal to 250 Active Apps)
- 1 App for 24 Nodes Per Session

Explanation: In RC-0.6.4, the validation of Claim+Proof requires 1 Session Generation. Any amount of NodesPerSession less than 100 is not statistically significant for Session Generation times, this may be leveraged for scalability. By increasing the number of NodesPerSession, a smaller amount of unique Sessions are needed to be generated during the validation. This means blocks are able to be processed more efficiently as Session Generation is the bottleneck in Claim+Proof validation.