

THORChain TSS Benchmark Report

thorchain.org

Abstract

In this report, we benchmark the THORChain TSS in the following two aspects. Firstly, we deploy the internal 4 nodes that running THORChain TSS keygen/keysign and evaluate the time and memory consumption in each module. We then deploy the THORChain TSS in the AWS and Digital Ocean across 4 regions. According to our benchmark result, it has a relatively low memory requirement to run the THORChain TSS and it is practical to deploy the THORChain TSS with 30 nodes that among different regions.

July 10, 2020

Contents

1	Introduction	2
2	Thorchain Tss benchmark with internal modules	3
2.1	THORChain TSS memory consumption for each module	3
2.1.1	Memory usage in standby nodes	3
2.1.2	KeyGen memory benchmark	3
2.1.3	KeyGen running time benchmark	3
2.2	KeySign memory benchmark	4
2.3	KeySign running time benchmark	4
3	THORChain TSS running time benchmark across data centres	9
4	Conclusion	10

Introduction

In this report, we conduct the intensive benchmark of the thorchain Tss (<https://gitlab.com/thorchain/tss/go-tss> on July 10, 2020) which employs the threshold signature scheme described in paper [Fast Multiparty Threshold ECDSA with Fast TrustlessSetup](#) with different scenarios. In this report, we describe the performance of THORChain TSS in terms of the time and memory consumption. For the THORChain TSS, in general, it has 4 communication rounds for the TSS key generation and 9 rounds for TSS signature generation process.

In the time consumption benchmark, we firstly benchmark the time consumption of THORChain TSS for each THORChain TSS function modules within 4 nodes machine, then, we benchmark the THORChain TSS with different number of nodes cross different data centres.

In the memory consumption benchmark, we deploy the test in 4 nodes environment, we evaluate the memory consumption each module in THORChain TSS and also the memory consumption for a standby THORChain TSS node.

According to our benchmark, the THORChain TSS can be deploy in the hybrid environment that has the more than 30 nodes with a reasonable keygen/keysign time.

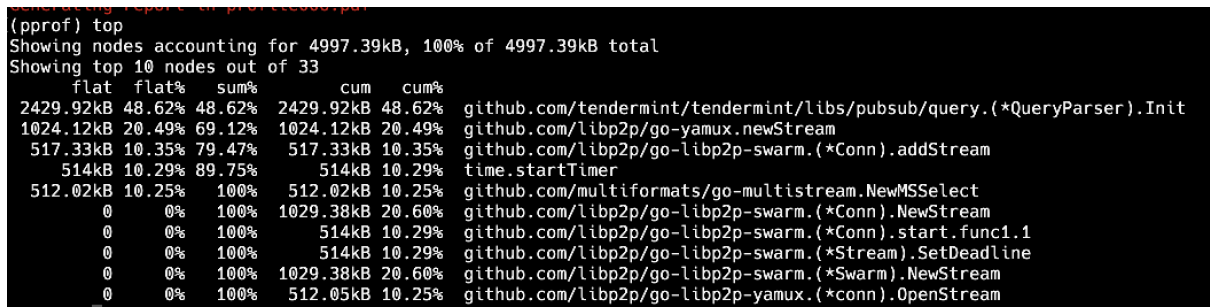
Thorchain Tss benchmark with internal modules

2.1 THORChain TSS memory consumption for each module

Setup: We set up the benchmark environment with 4 different THORChain TSS nodes in Digital Ocean data centre in the same region. For each node, we have 1GB memory and 1 CPU core. For the benchmark, we run 400 rounds of keygen and keysign respectively.

2.1.1 Memory usage in standby nodes

After running the keygen and keysign for 800 rounds, the memory usage of the THORChain TSS is shown in fig 1. It can be observed that when the THORChain TSS node finishes its task, it releases the memory that used for THORChain TSS communication and the intermediate results. The memory that needs to have the node in standby is about 5000KB. 48% of the memory is occupied for storing the cryptographic data, and the rest of the 52% memory are occupied for maintaining the underlying p2p network.



```

(pprof) top
Showing nodes accounting for 4997.39kB, 100% of 4997.39kB total
Showing top 10 nodes out of 33

```

flat	flat%	sum	cum	cum%	module
2429.92kB	48.62%	48.62%	2429.92kB	48.62%	github.com/tendermint/tendermint/libs/pubsub/query.(*QueryParser).Init
1024.12kB	20.49%	69.12%	1024.12kB	20.49%	github.com/libp2p/go-yamux.newStream
517.33kB	10.35%	79.47%	517.33kB	10.35%	github.com/libp2p/go-libp2p-swarm.(*Conn).addStream
514kB	10.29%	89.75%	514kB	10.29%	time.startTimer
512.02kB	10.25%	100%	512.02kB	10.25%	github.com/multiformats/go-multistream.NewMSSelect
0	0%	100%	1029.38kB	20.60%	github.com/libp2p/go-libp2p-swarm.(*Conn).NewStream
0	0%	100%	514kB	10.29%	github.com/libp2p/go-libp2p-swarm.(*Conn).start.func1.1
0	0%	100%	514kB	10.29%	github.com/libp2p/go-libp2p-swarm.(*Stream).SetDeadline
0	0%	100%	1029.38kB	20.60%	github.com/libp2p/go-libp2p-swarm.(*Swarm).NewStream
0	0%	100%	512.05kB	10.25%	github.com/libp2p/go-libp2p-yamux.(*conn).OpenStream

Figure 1: THORChain TSS memory consumption of standby nodes

2.1.2 KeyGen memory benchmark

The snapshot of the memory of the running Tss keygen is shown in fig.2. It can be observed the total memory consumption for the keygen is 8798KB. THORChain TSS intermediate results occupies most of the memory(46.03%) and these memory will be released once the process is done. Additionally, sending the Tss shares to the peers consumes 16.08% of the total memory. To process the received message, THORChain TSS also reserves 11.7% of the memory for storing the temporary data read from the p2p network.

2.1.3 KeyGen running time benchmark

The snapshot of the running time of the keygen is shown in fig.3. It can be observed that keygen takes 12.21 seconds to finish. It can be observed that for the four nodes THORChain TSS, most of the time is spent on cryptographic calculation (big.AddMul takes 75.87% of total time). The time spent for each round is shown in table 1. It can be observed that round 2 takes 66.53% of the total keygen time. Round 3 consumes the least time as it account for 1.71% of the total keygen time.

	Time spent (second)	The percentage of total time
Round 1	3.83	25.18%
Round 2a	5.05	33.20%
Round 2b	5.07	33.33%
Round 3	0.26	1.71%
Round 4	0.6	3.94%

Table 1: *keygen time consumption for each round*

2.2 KeySign memory benchmark

The snapshot of the memory of the running Tss keysign is shown in fig.4. It can be observed the total memory consumption for the keysign is 5504KB. THORChain TSS intermediate results occupies most of the memory(44.15%) and these memory will be released. Additionally, sending the Tss shares to the peers consumes 16.08% of the total memory. To process the received message, THORChain TSS also reserves 11.7% of the memory for storing the temporary data read from the p2p network.

2.3 KeySign running time benchmark

The snapshot of the running time of the Tss keysign is shown in fig.5. It can be observed that keysign takes 13.95 seconds to finish. It can be observed that for the four nodes THORChain TSS, most of the time is spent on cryptographic calculation (`big.AddMul` takes 80.57% of total time). The time spent for each round is shown in table 2. It can be observed that round 2 takes 55.42% of the total keysign time. Time spend in round 4 to round 7 is not recorded in the benchmark as the time is too short to be ignored. Since we only have 4 nodes in the group and all the nodes are deployed in the same network, time spent on transmitting the data accounts for less then 2% of the total time.

	Time spent (second)	The percentage of total time
Round 1	1.42	10.18%
Round 2a	3.83	27.46%
Round 2b	3.90	27.96%
Round 3a	1.98	14.19%
Round 3b	1.98	14.19%
Round 4	NA	NA
Round 5	NA	NA
Round 6	NA	NA
Round 7	NA	NA

Table 2: *keysign time consumption for each round*

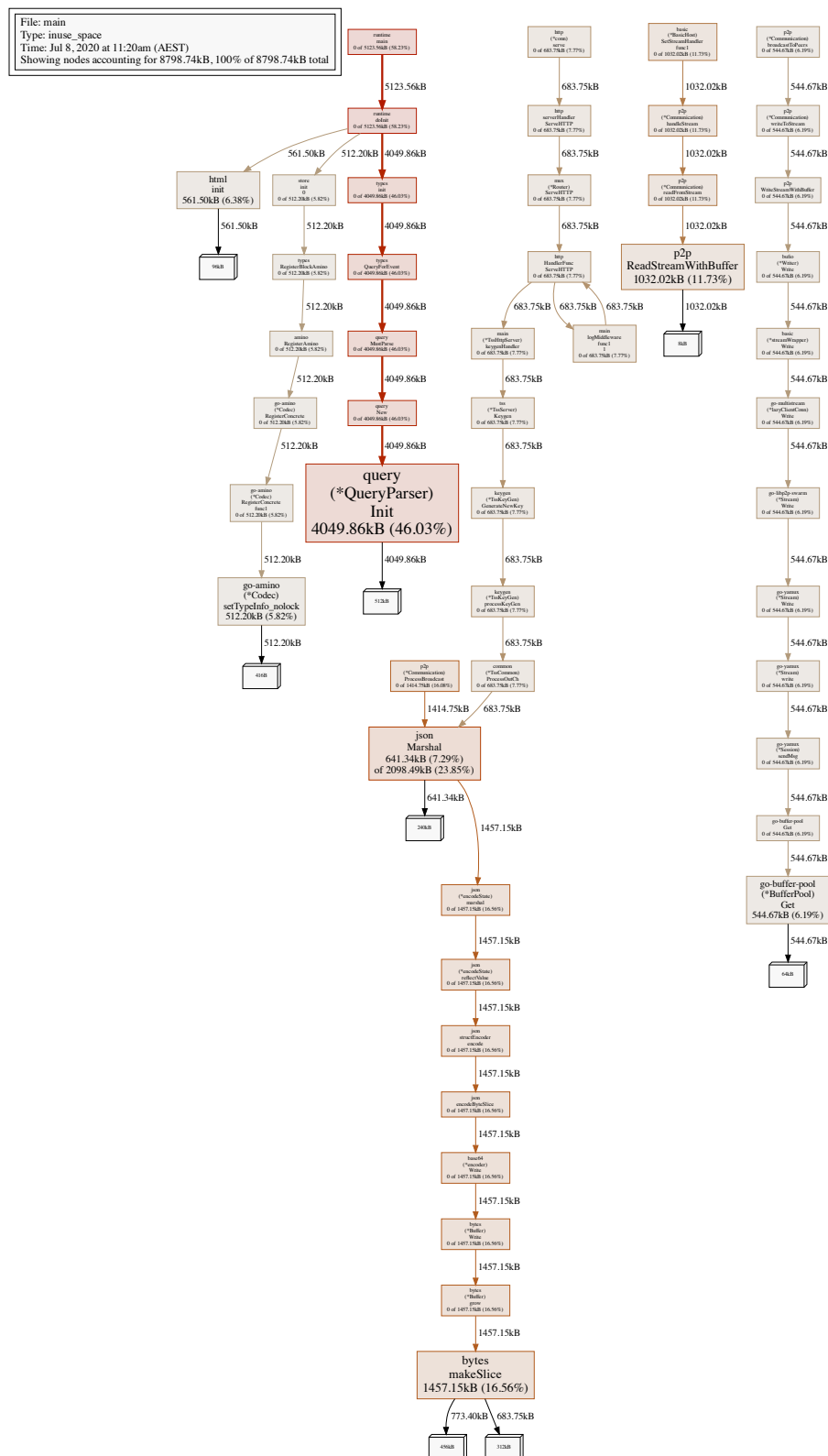


Figure 2: *Thorchain Tss memory consumption for keygen*

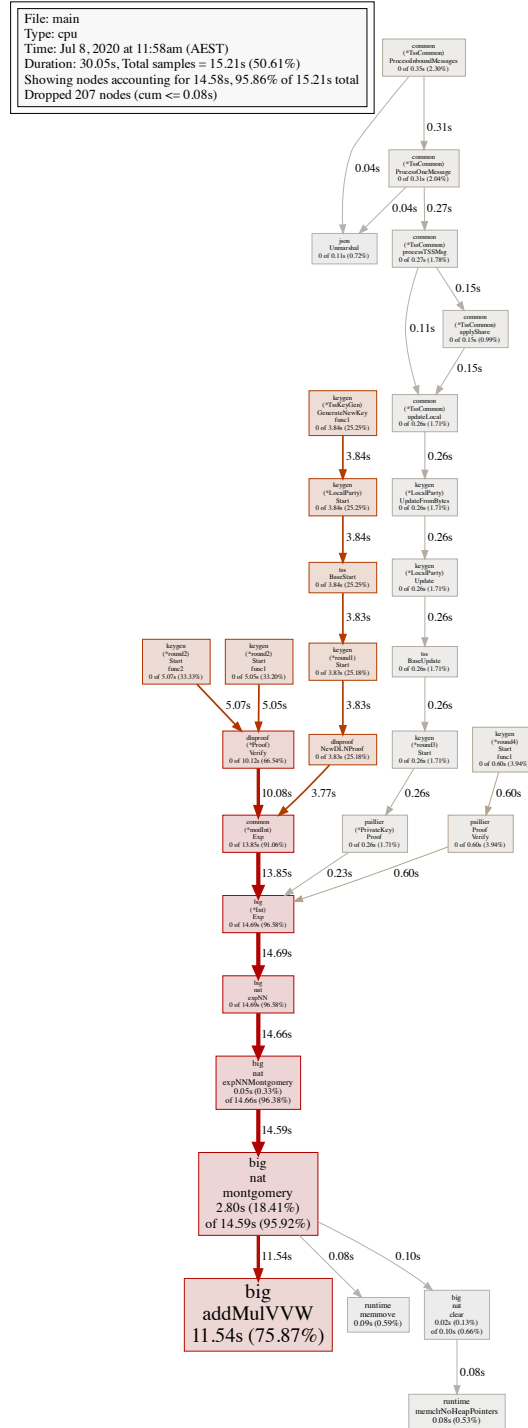


Figure 3: THORChain TSS time consumption for keygen

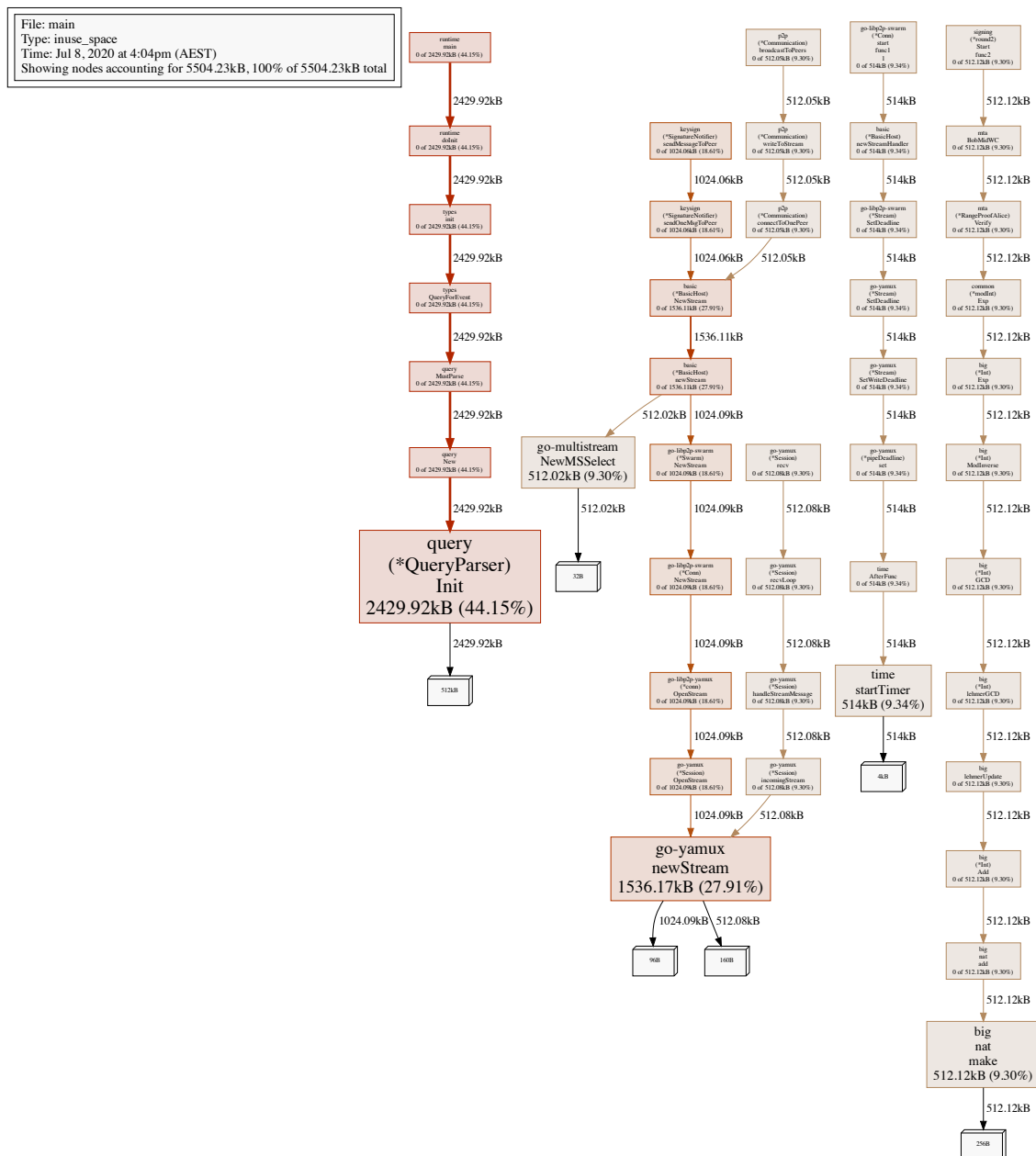


Figure 4: *THORChain TSS memory consumption for keysign*



THORChain TSS running time benchmark across data centres

In this section, we deploy THORChain TSS nodes from 3 nodes up to 30 nodes in Digital Ocean data centre and Amazon Web Service(AWS), all the virtual machines have 1G memory and 1 CPU.

To deploy the benchmark environment, We deploy 10 virtual machines in AWS and 22 virtual machines in Digital Ocean machines. To make it fair, we deploy the 22 Digital Ocean virtual machines across 4 regions. For each benchmark, we randomly pick up some nodes to form the THORChain TSS party and run the test for 5 times to get the average running time.

It can be observed in fig 6 that the time spend on keygen/keysign increases quadratically with the increase of the nodes involved in the THORChain TSS. In general, keygen operation consumes much more time compared with keysign operation. The time spent on keygen is predicted by the equation $y = 67.672x^2 + 916.8x + 11572$ with $R^2 = 0.9562$ where x is the number of the nodes involved in the THORChain TSS. The time spent on keysign is predicted by the equation $y = 50.252x^2 - 373.93x + 6198.1$ with $R^2 = 0.8958$ where x is the number of the nodes involved in the THORChain TSS.

According to the table 3, for 6 nodes THORChain TSS, it takes about 18 seconds to run the keygen and it takes about 15 seconds to run the keysign. For the 30 nodes keygen, it needs about 109 seconds to run the keygen and it needs about 47 seconds to run the keysign.

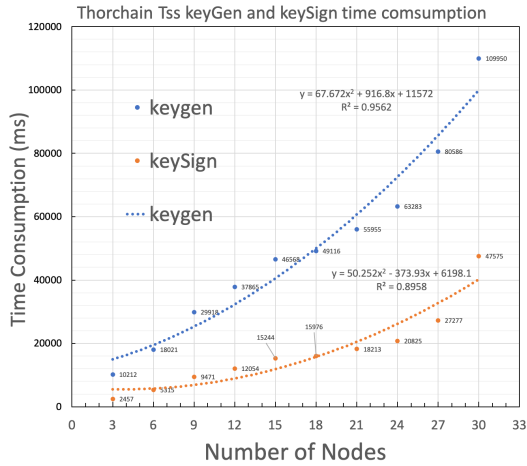


Figure 6: THORChain TSS time consumption for nodes across data centres

nodes	keygen(second)	keysign(second)
3	10.212	2.457
6	18.021	5.315
9	29.918	9.471
12	37.865	12.054
15	46.568	15.244
18	49.116	15.976
21	55.955	18.213
24	63.283	20.825
27	80.586	27.277
30	109.95	47.575

Table 3: keygen/keysign running time on machines across data centres

Conclusion

Though our intensive benchmark, it can be concluded for a small number of nodes, the cryptographic computation dominates the time spent on THORChain TSS processing, while with the growth of the number of the nodes in the network, the time spent on sending/receiving the data from the peers overwhelms the time spent on calculating the THORChain TSS intermediate results. According to the result of the benchmark, the time growth of THORChain TSS processing is quadratic. THORChain TSS is viable for key-generating and key-signing committees up to 30 nodes.