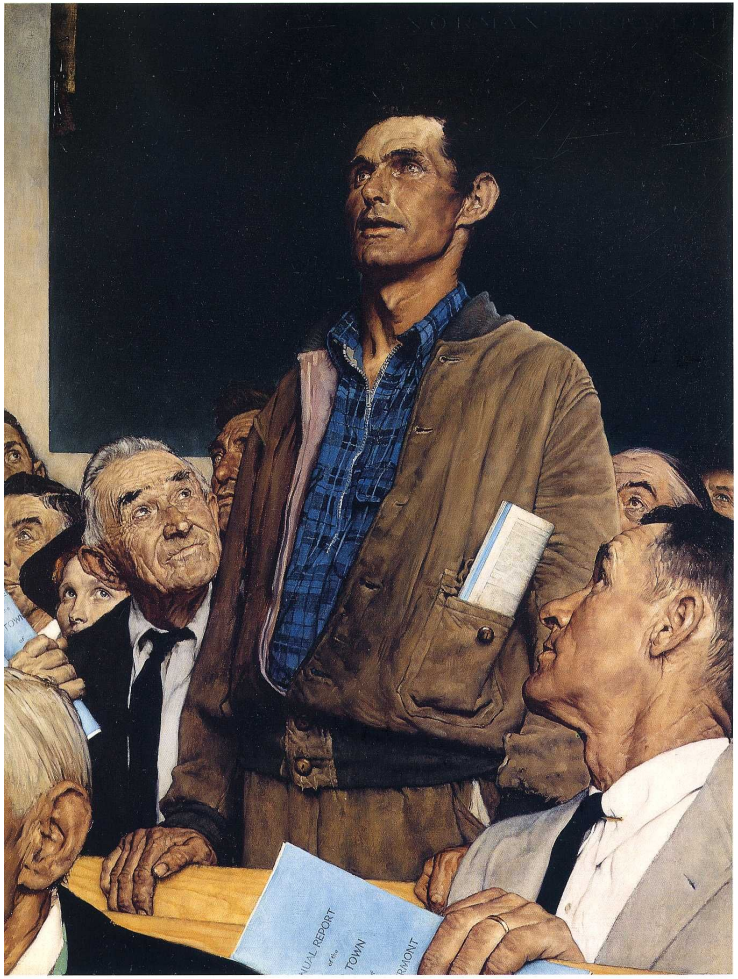
blocksize

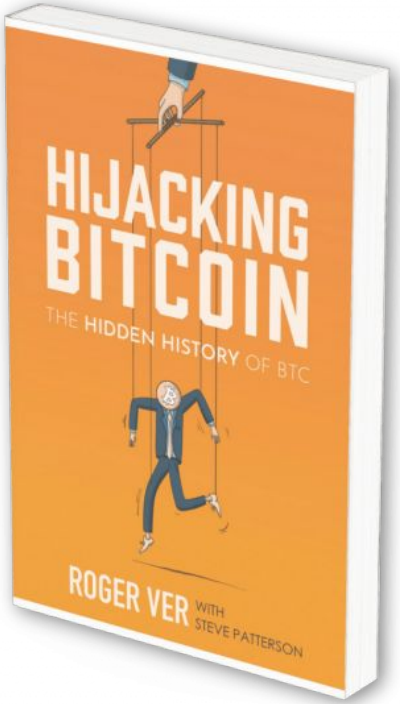
2025-04-12

tysons



Freedom of Speech, 1943

1. Why it’s worth thinking about
2. Measurements and demythologizing


HIJACKING
BITCOIN
THE HIDDEN HISTORY OF BTC
ROGER VER WITH STEVE PATTERSON

My goals

1. Deliver on bitcoin’s promise of value transmission/storage without intermediaries
    a. L1 “checking acct volume” for most people (1 txn/month)
2. Avoid (de facto) state capture

My goals

1. Deliver on bitcoin’s promise of value transmission/storage without intermediaries
    a. L1 “checking acct volume” for most people (1 txn/month)
2. Avoid (de facto) state capture

This is not about payments. This is about store of value.

My goals

1. Deliver on bitcoin’s promise of value transmission/storage without intermediaries
    a. L1 “checking acct volume” for most people (1 txn/month)
2. Avoid (de facto) state capture

This is not about payments. This is about store of value.

I think 3 is mostly achievable with tech we have (or know about) today

Why care?


~0 sat/vB
1 - 350 sat/vB
0.011 BTC
720 transactions
In ~10 minutes
891648
~0 sat/vB
1 - 125 sat/vB
0.012 BTC
831 transactions
2 minutes ago
AntPool
891647
~0 sat/vB
1 - 180 sat/vB
0.008 BTC
748 transactions
5 minutes ago
Foundry USA
891646
~2 sat/vB
1 - 200 sat/vB
0.048 BTC
3,177 transactions
7 minutes ago
AntPool
891645
~3 sat/vB
1 - 526 sat/vB
0.031 BTC
2,153 transactions
21 minutes ago
F2Pool
891644
~1 sat/vB
1 - 181 sat/vB
0.032 BTC
2,447 transactions
30 minutes ago
ViaBTC
891643
~1 sat/vB
1 - 289 sat/vB
0.03 BTC
2,154 transactions
36 minutes ago
Foundry USA
891642
~2 sat/vB
1 - 300 sat/vB
0.054 BTC
2,954 transactions
44 minutes ago
Luxor
891641
~3 sat/vB
1 - 301 sat/vB
0.072 BTC
3,574 transactions
57 minutes ago
Luxor
891640
~0 sat/vB
1 - 200 sat/vB
0.011 BTC
362 transactions
78 minutes ago
SECPOOL
891639
~0 sat/vB
1 - 529 sat/vB
0.009 BTC
910 transactions
79 minutes ago
F2Pool

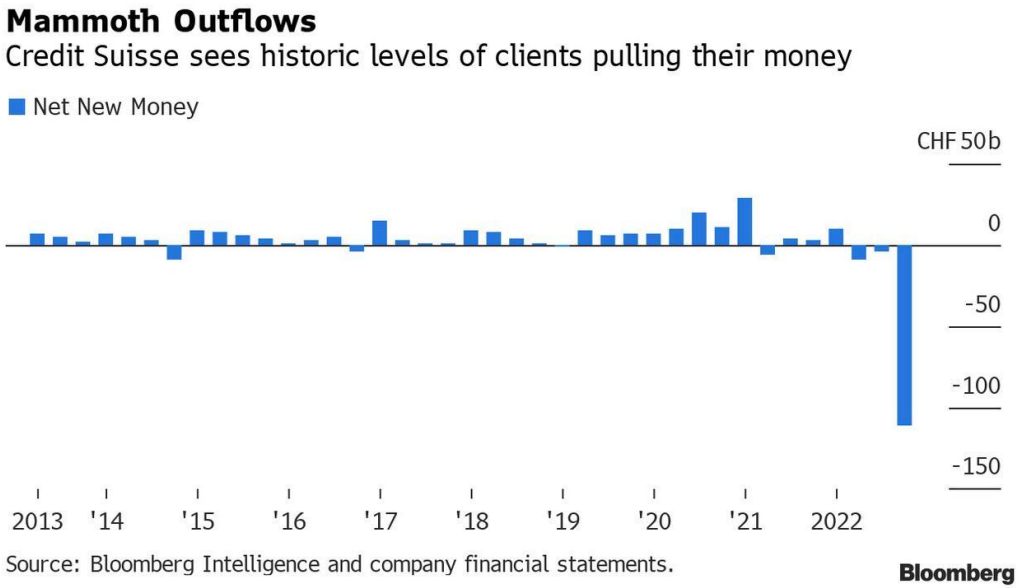
Why care?


~0 sat/vB
1 - 350 sat/vB
0.011 BTC
720 transactions
In ~10 minutes
891648
~0 sat/vB
1 - 125 sat/vB
0.012 BTC
831 transactions
2 minutes ago
AntPool
891647
~0 sat/vB
1 - 180 sat/vB
0.008 BTC
748 transactions
5 minutes ago
Foundry USA
Bitcoin / U.S. Dollar · 1W · COINBASE
O78,370.15 H81,223.67 L74,420.69 C77,203.10 −1,167.65 (−1.49%)
77,209.49
SELL
0.01
77,209.50
BUY
Vol · BTC 51.84 K
120,000.00
110,000.00
100,000.00
90,000.00
80,000.00
70,000.00
60,000.00
50,000.00
40,000.00
30,000.00
20,000.00
10,000.00
77,203.10
4d 10h
51.84 K
TradingView
Jul
2021
Jul
2022
Jul
2023
Jul
2024
Jul
2025
891640
~0 sat/vB
1 - 200 sat/vB
0.011 BTC
362 transactions
8 minutes ago
SECPOOL
891639
~0 sat/vB
1 - 529 sat/vB
0.009 BTC
910 transactions
79 minutes ago
F2Pool

Why care?

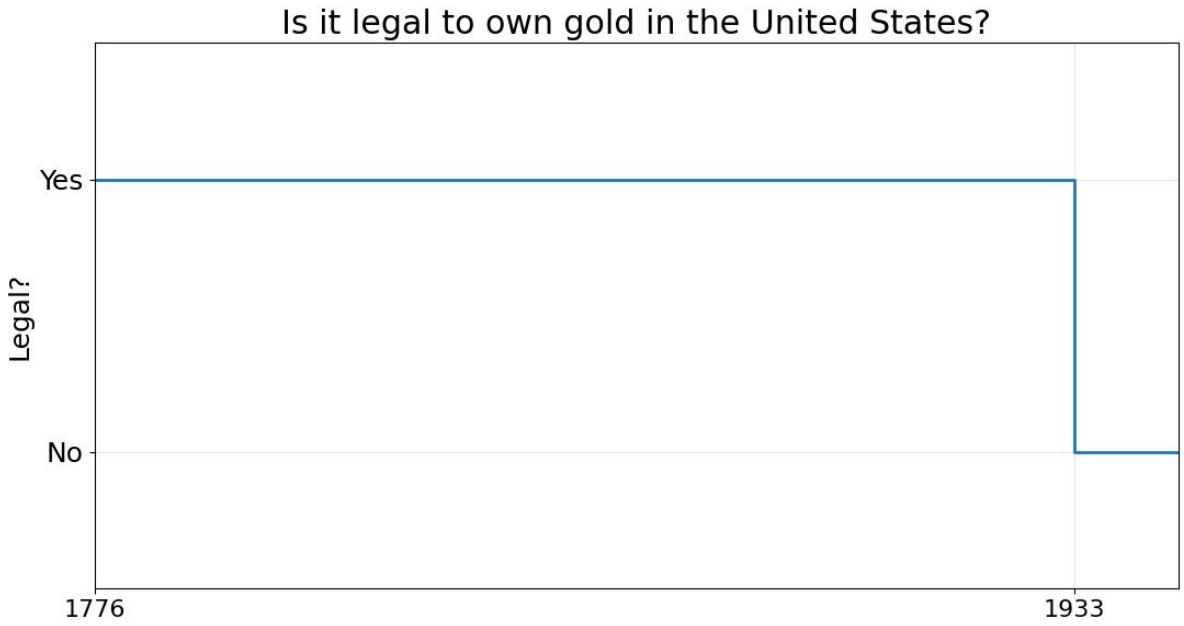
“well, I can still transact”

Finance is discontinuous



regulation

~~Finance~~ is discontinuous



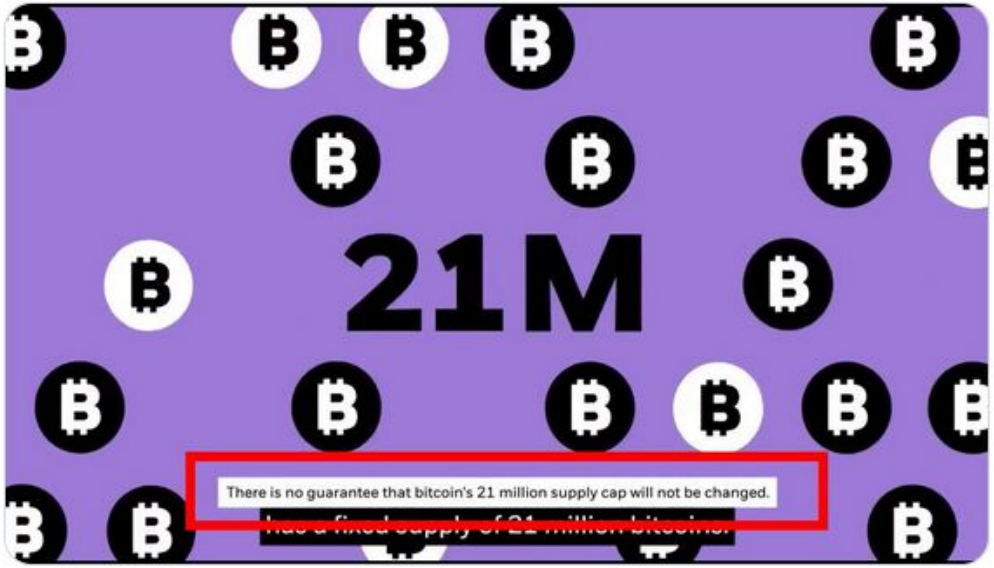
“On the afternoon of the Wednesday before Thanksgiving, something unexpected will happen to the turkey. It will incur a revision of belief.”

Nassim Taleb (ardent bitcoin supporter)

not your keys,
not your ~~coins~~ consensus

jamesob
@jamesob

Bitcoin™!



5:50 PM · Dec 18, 2024 · **136.9K** Views

View post engagements

14 34 156 11

all of the sudden
everyone wants to be a self-custodian

now what?

chain limitations on exit

[10]:
```
# Per https://contenthub-static.crypto.com/wp_media/2024/08/Crypto.com-Crypto-Market-Sizing-H1-2024-1.pdf
est_bitcoin_users = 314_000_000

# Assume that 5% of Bitcoin holders are in self-custody (probably generous)
est_self_custody_users = 0.05 * est_bitcoin_users
est_exch_users = est_bitcoin_users - est_self_custody_users

print(f"there are {est_exch_users / 1e6:_}M users holding coins on exchange")
```

```
there are 298.3M users holding coins on exchange
```

chain limitations on exit

[2]:
```
vb_per_block = 1_000_000

blocks_per_month = 6 * 24 * 30

txsize_2in2out_vb = 208.5  # per https://bitcoinops.org/en/tools/calc-size/

txns_per_block = int(vb_per_block / txsize_2in2out_vb)

txns_per_month = txns_per_block * blocks_per_month

print(f"capacity: {txns_per_month:,} (2-in-2-out) txns per month")
```

```
capacity: 20,718,720 (2-in-2-out) txns per month
```

chain limitations on exit

[10]:
```
# Per https://contenthub-static.crypto.com/wp_media/2024/08/Crypto.com-Crypto-Market-Sizing-H1-2024-1.pdf
est_bitcoin_users = 314_000_000

# Assume that 5% of Bitcoin holders are in self-custody (probably generous)
est_self_custody_users = 0.05 * est_bitcoin_users
est_exch_users = est_bitcoin_users - est_self_custody_users

print(f"there are {est_exch_users / 1e6:_}M users holding coins on exchange")
```

```
there are 298.3M users holding coins on exchange
```

[12]:
```
custodial_move_months = est_exch_users / txns_per_month
print(f"it would take {custodial_move_months:.2f} "
      f"months of uninterrupted use chain for the {est_exch_users / 1e6:_}M custodial users to withdraw")
```

```
it would take 14.40 months of uninterrupted use chain for the 298.3M custodial users to withdraw
```

1. US gov (or G7 countries) 6102 an impl. w/ forks
2. “Exit window” given
3. Blackrock has fork choice in their S1, dumps other side
4. Profit-seeking (and regulated) miners must follow
5. “Real” bitcoin PoW cooked

1. US gov (or G7 countries) 6102 an impl. w/ forks
2. “Exit window” given
3. Blackrock has fork choice in their S1, dumps other side
4. Profit-seeking (and regulated) miners must follow
5. “Real” bitcoin PoW cooked

deployed L2s don’t solve this

[14]:
```
# A 1-in-2-out LN channel. Bare minimum, probably larger on average.
txsize_ln_open_vb = 154  # e.g. https://mempool.space/tx/f9b47a6c2137c0adf0cbb44155b46b27b690018fc16eb282abb85e791ca90715

time_to_US_channels = (est_exch_users * txsize_ln_open_vb) / (vb_per_block * blocks_per_month)

print(f"it would take {time_to_US_channels:.2f} months of uninterrupted chain space "
      f"to open an LN channel for each custodial user")
```

```
it would take 10.63 months of uninterrupted chain space to open an LN channel for each custodial user
```

best-case response

- some kind of batched withdrawal to “other” L2s
  - Ark, Lightning, cashu, fedimint
  - some TBD shared UTXO scheme
- you don’t have real bitcoin, but you escaped gov?
- exchanges need preexisting batch connections

best-case response

- some kind of batched withdrawal to “other” L2s
  - Ark, Lightning, cashu, fedimint
  - some TBD shared UTXO scheme
- you don’t have real bitcoin, but you escaped gov?
- exchanges need preexisting batch connections

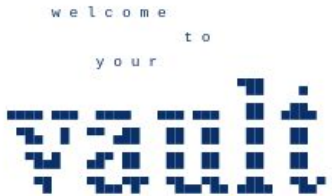
- **without L1 access, you can always get rugged**

Bitcoin: A Peer-to-Peer Electronic Cash System

Satoshi Nakamoto
satoshin@gmx.com
www.bitcoin.org

Abstract. A purely peer-to-peer version of electronic cash would allow online payments to be sent directly from one party to another without going through a financial institution. Digital signatures provide part of the solution, but the main benefits are lost if a trusted third party is still required to prevent double-spending. We propose a solution to the double-spending problem using a peer-to-peer network.

a more optimistic case

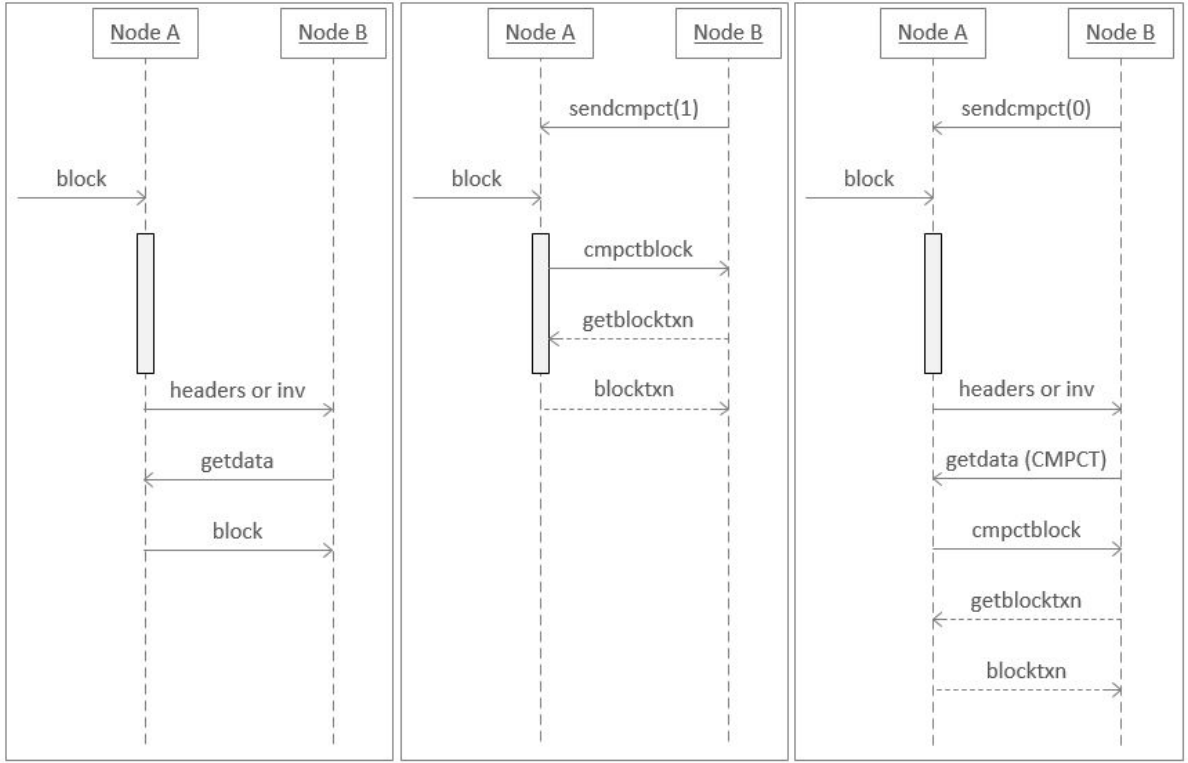


objections

miner centralization

block propagation

compact blocks



miner centralization

block propagation

good hardware, internet connection is basically the last thing limiting pools

mempool diffs

“too much in the mempool; upon new block I’ll have to retrieve too much”

mempool diffs

- Weak blocks fix this
  - Like pool shares for the network
  - PoW backed mempool syncing
  - Miner-incentive compatible
- P2P relay was always sort of doomed anyway
  - Relies on bandwidth donation
  - Useful policy always lags
- Would need this anyway with high fee rates

Second Look at Weak Blocks

Implementation

instagibbs 2 Apr 2024

Weak blocks 28, or “near blocks” are not a new idea.

In short, have miners propagate what amounts to mining shares over the p2p network, which allows PoW-backed sharing of data.

Historical discussions 8 of weak blocks centered around the blocksize and scaling debate 4, which means there was intense focus on reducing the marginal bytes sent per weak block to aid “gigameg blocks”. There was seemingly a lot of focus on creating DAGs, extra-consensus chains, and similar mechanisms for increasing the blocksize safely.

Almost 10 years have passed, communities have split, basically everyone is a small blocker of some kind. Ignoring blocksize increases as a motivation, *is there value in reconsidering this type of proposal?*

miner centralization

selfish mining

- mostly about share of hash rather than blocksize
- worth thinking about

fee revenue for miners

“miners will need high fees and full blocks to get paid”

fee revenue for miners

Assuming a price of $250,000/BTC (i.e. $0.0025/sat)

Fee subsidy of 1.5625 BTC

- 4MvB block
  - feerate required: 156 sats/vB
  - total payments: 4,796
  - cost per payment: $81.45
- 1 sat/vB feerate
  - full blocksize 156.25 MvB required
  - total payments: 749,400
  - cost per payment: $0.52

Fee subsidy of 3.125 BTC

- 4MvB block
  - feerate required: 312 sats/vB
  - total payments: 4,796
  - cost per payment: $162.89
- 1 sat/vB feerate
  - full blocksize 312.5 MvB required
  - total payments: 1,498,800
  - cost per payment: $0.52

Fee subsidy of 6.25 BTC

- 4MvB block
  - feerate required: 625 sats/vB
  - total payments: 4,796
  - cost per payment: $325.78
- 1 sat/vB feerate
  - full blocksize 625.0 MvB required
  - total payments: 2,997,601
  - cost per payment: $0.52


Subsidy
Fees
Subsidy (USD)
Fees (USD)
Subsidy (%)
Fees (%)
60 BTC
50 BTC
40 BTC
30 BTC
20 BTC
10 BTC
0 BTC
January 3, 2009
November 30, 2010
October 21, 2012
September 12, 2014
August 3, 2016
June 25, 2018
May 16, 2020
April 7, 2022
February 27, 2024

makes IBD harder

“how will I sync?”

~~makes IBD harder~~

- easy problem to solve
- assumeutxo is already sufficient
  - to tip in under ~2 hours on most computers
  - assumevalid already in use by default
- maybe someday
  - swiftsync
  - zerosync

makes tip maintenance harder

“how will my node stay current?”

brief measurement digression

- Measurements performed on “pretty good” consumer desktop
- Directionally true to anything with SSD, >32GB RAM

let’s make it interesting

[6]:
```
ppl = 7_000_000_000
target_txsperpersonmonth = 2
target_txspermonth = ppl * target_txsperpersonmonth

txs_per_block = int(target_txspermonth / blocks_per_month)

mvb_per_block = (txs_per_block * txsize_2in2out_vb) // (1000 ** 2)
sig_checks_per_block = txs_per_block * 2

print(f"MvB per block: {mvb_per_block}")
print(f"sig checks per block: {sig_checks_per_block:_}")
```

```
MvB per block: 675.0
sig checks per block: 6_481_480
```

checking sigs

File: secpbench.cpp

```cpp
#include <iostream>
#include <format>
#include <cstring>
#include <vector>
#include <thread>
#include <random>
#include <chrono>
#include <secp256k1.h>

constexpr size_t NUM_KEYS = 6'500'000;
constexpr size_t NUM_THREADS = 30;

void generate_keys(std::vector<std::vector<unsigned char>>& keys, size_t start, size_t end) {
    std::random_device rd;
    std::mt19937_64 gen(rd()); // 64-bit generator
    std::uniform_int_distribution<uint64_t> dist(0, UINT64_MAX);

    for (size_t i = start; i < end; ++i) {
        std::vector<unsigned char> key(32);
        for (size_t j = 0; j < 4; ++j) { // Generate 4 blocks of 64 bits (8 bytes each)
            uint64_t rand_block = dist(gen);
            std::memcpy(key.data() + j * 8, &rand_block, 8);
        }
        keys[i] = std::move(key);
        if (i % 10'000 == 0) {
            std::cout << "Generated " << i << " keys" << std::endl;
        }
    }
}
```

checking sigs

```
void benchmark_sign(
        const std::vector<std::vector<unsigned char>>& keys, size_t thread_id, size_t& sign_count) {
    secp256k1_context* ctx = secp256k1_context_create(SECP256K1_CONTEXT_SIGN);
    unsigned char msg[32] = {0};
    unsigned char sig[64];
    size_t count = 0;
    size_t keys_per_thread = NUM_KEYS / NUM_THREADS;
    size_t start = thread_id * keys_per_thread;
    size_t end = (thread_id + 1) * keys_per_thread;

    for (size_t i = start; i < end; ++i) {
        secp256k1_ecdsa_signature signature;
        if (secp256k1_ecdsa_sign(ctx, &signature, msg, keys[i].data(), nullptr, nullptr)) {
            ++count;
        }
        if (count % 10'000 == 0) {
            std::cout << std::format("signer thread {} signed {}\n", thread_id, count);
        }
    }

    secp256k1_context_destroy(ctx);
    sign_count = count;
}
```

```
        }
        keys[i] = std::move(key);
        if (i % 10'000 == 0) {
            std::cout << "Generated " << i << " keys" << std::endl;
        }
    }
}
```

checking sigs

checking sigs

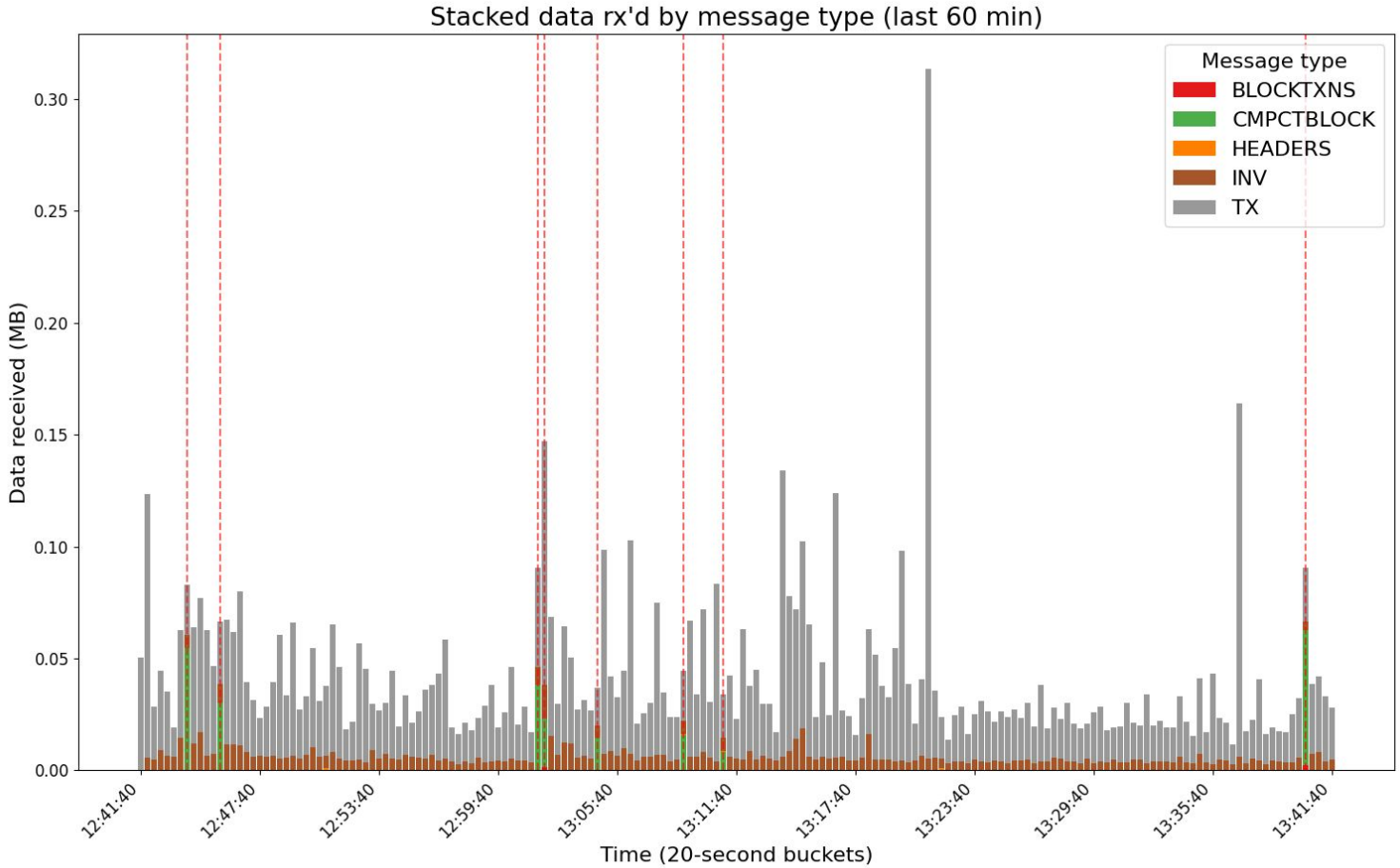
```
12:55:42 james@fido james/tmp % clang++ -lsecp256k1 -std=c++23 secpbench.cpp && ./a.out | tail -n 6
signer thread 14 signed 210000
signer thread 16 signed 210000
signer thread 10 signed 210000
signer thread 0 signed 210000
Total signatures: 6499980
Time taken: 8859 ms
12:55:56 james@fido james/tmp %
```

brief measurement digression

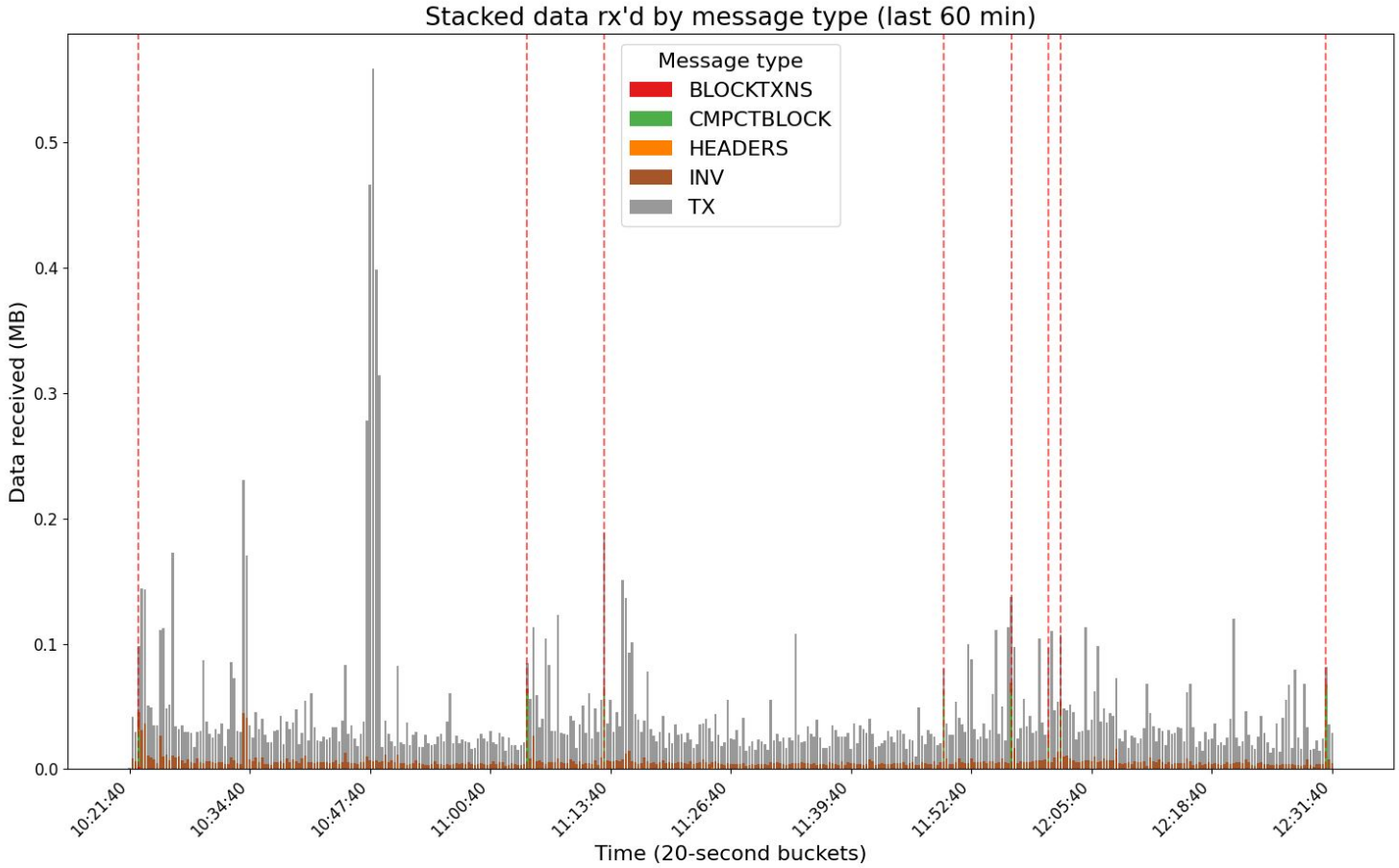
- P2P as the event loop: TX, CMPCTBLOCK, BLOCKTXN, ...
- There are basically two files in Core for perf:
  - validation.cpp
  - net_processing.cpp
- Script caching: cached on mempool acceptance
- UTXO set cache warming

bandwidth distribution

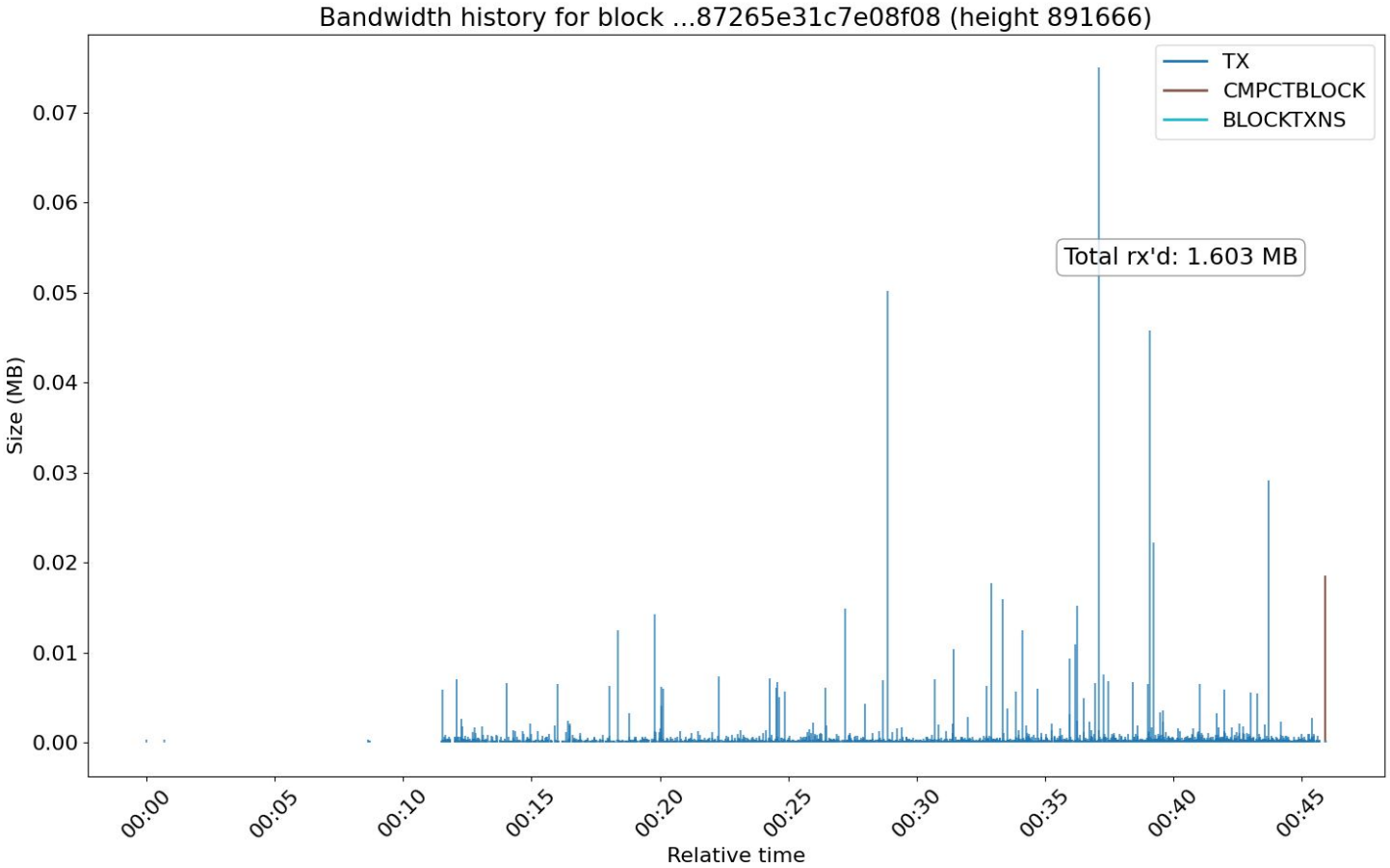
measurement: distribution of data rx'd



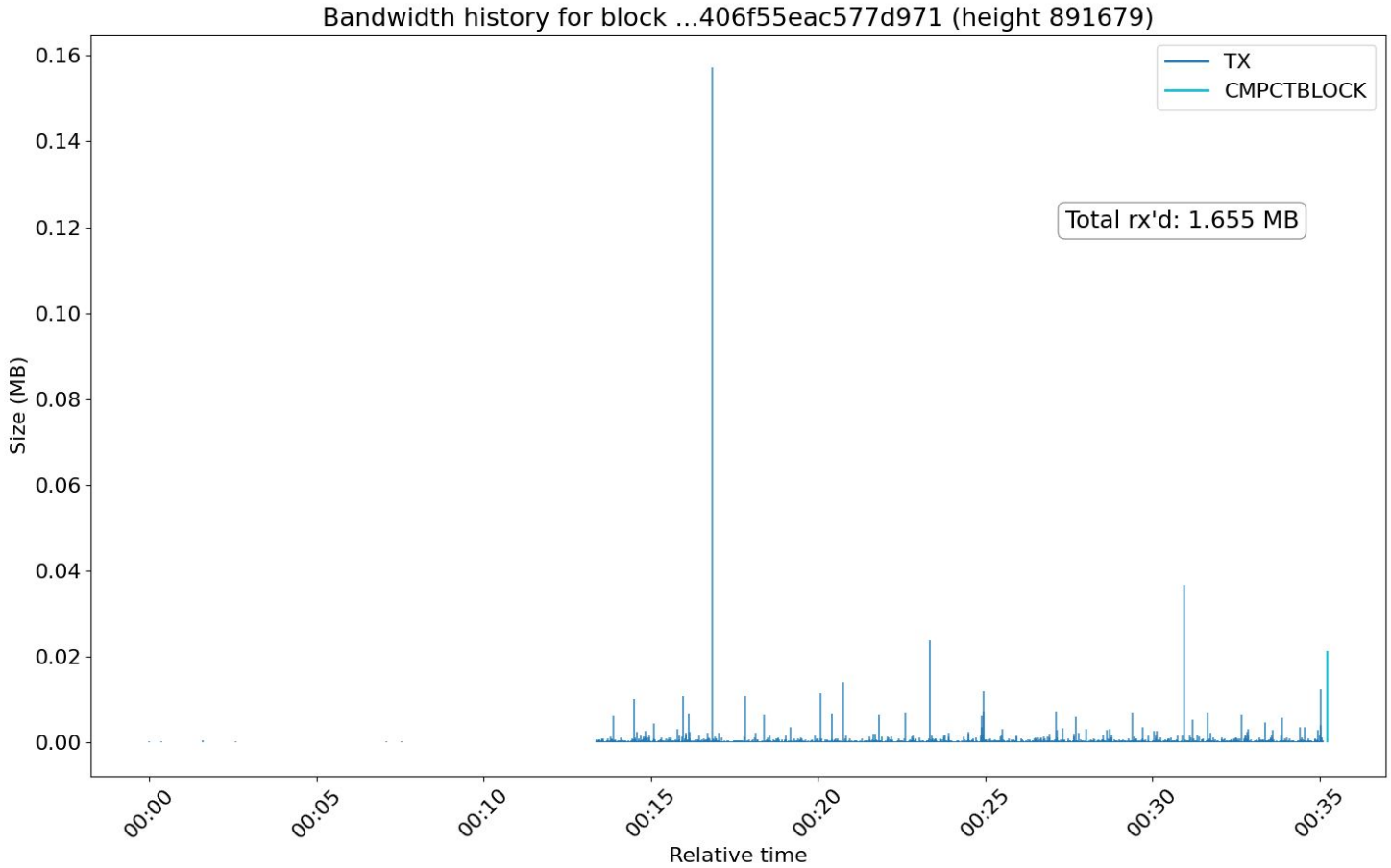
measurement: distribution of data rx'd



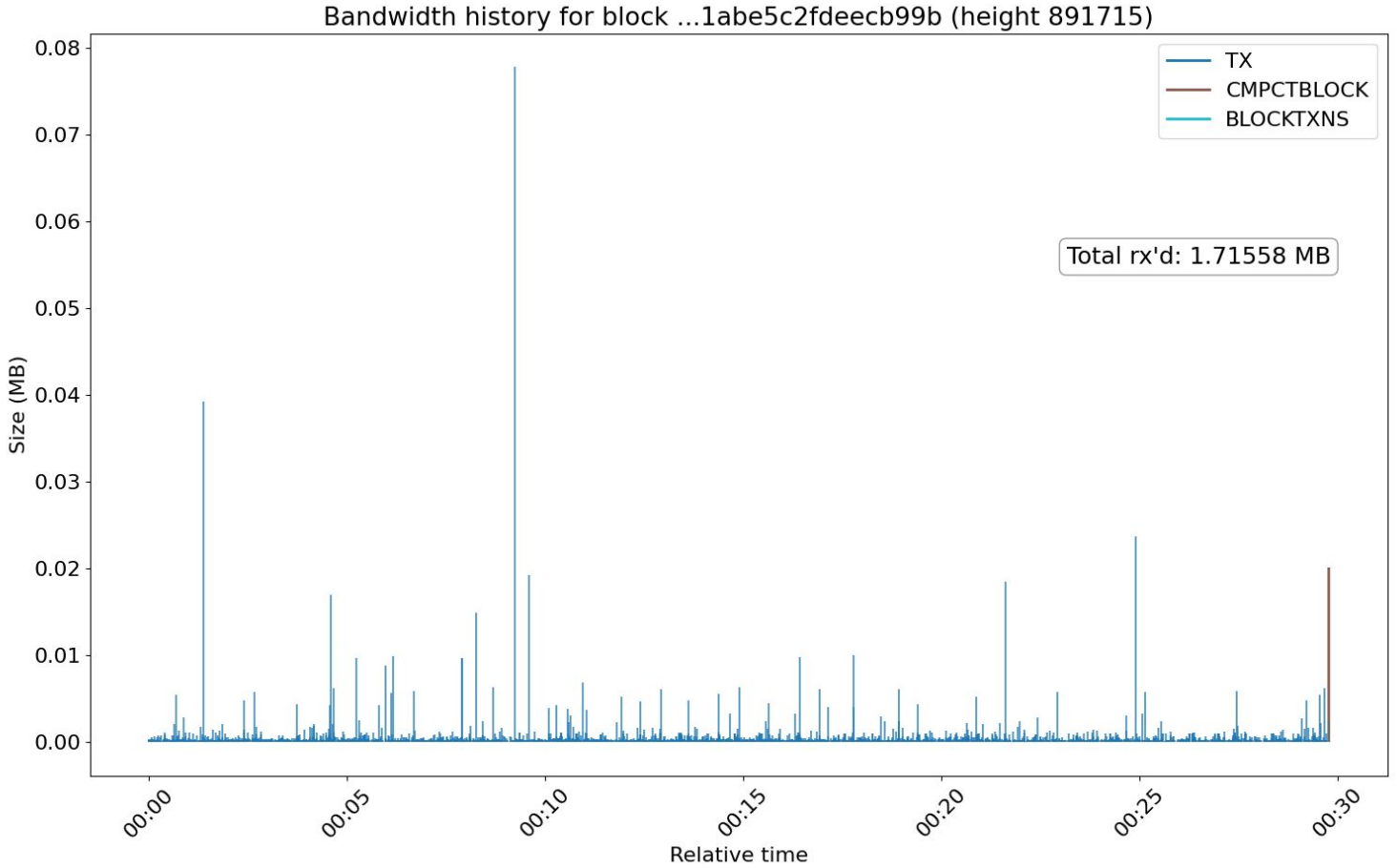
measurement: block bandwidth history



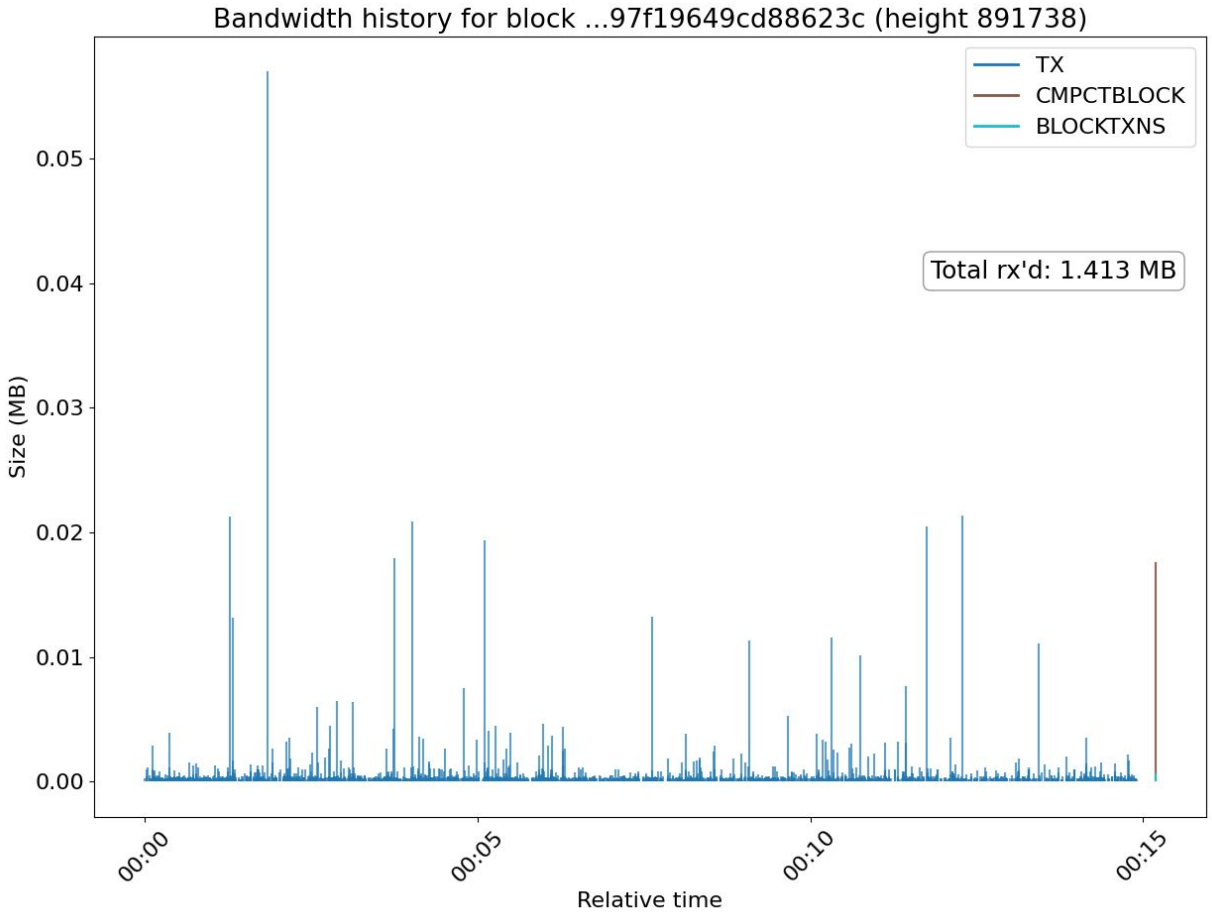
measurement: block bandwidth history



measurement: block bandwidth history



measurement: block bandwidth history

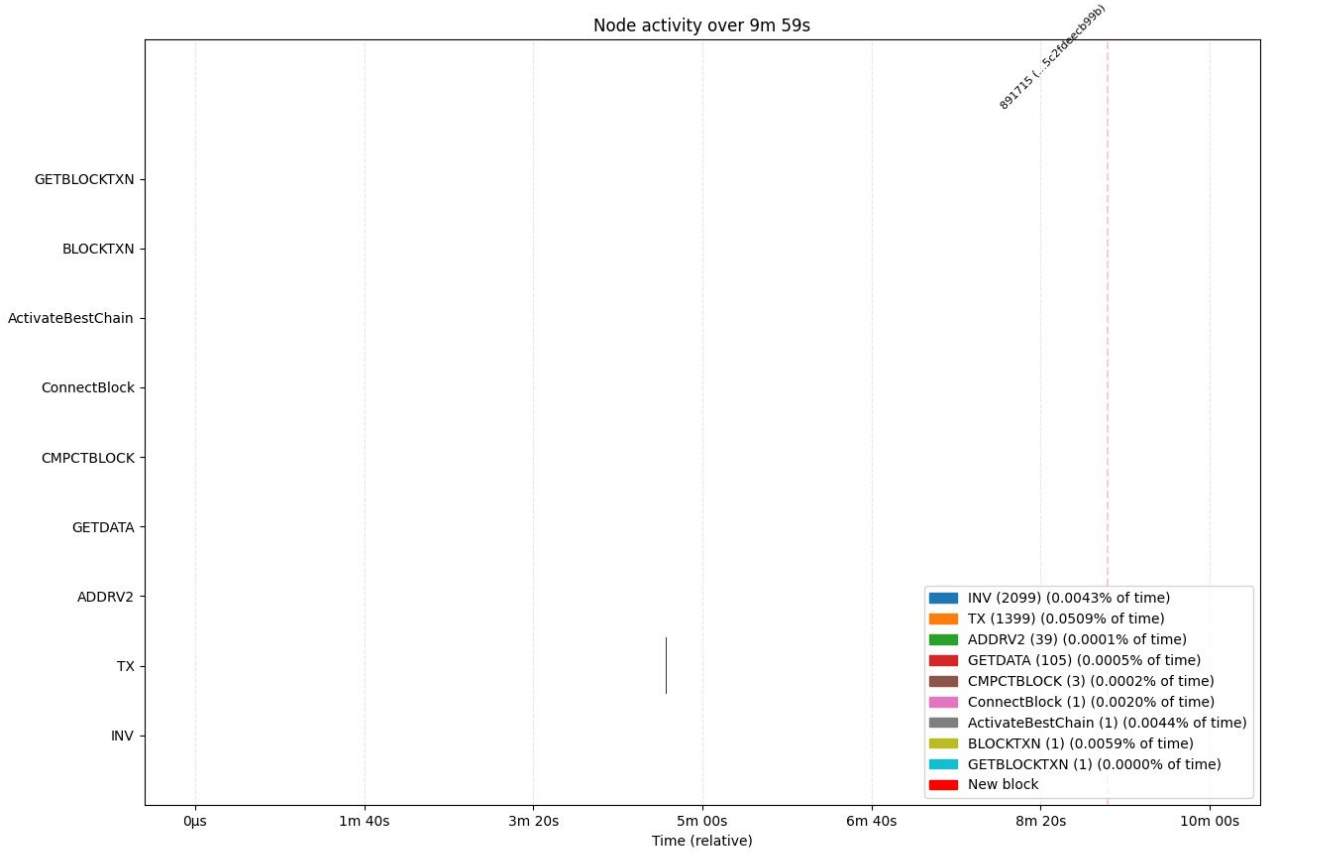


work distribution

```
@@ -2443,10 +2444,14 @@ bool Chainstate::ConnectBlock(const CBlock& block, BlockValidationState& state,
2443  2444      AssertLockHeld(cs_main);
2444  2445      assert(pindex);
2445  2446
      2447  +   auto event = g_event_logger->time_event("ConnectBlock");
      2448  +
2446  2449      uint256 block_hash{block.GetHash()};
2447  2450      assert(*pindex->phashBlock == block_hash);
2448  2451      const bool parallel_script_checks{m_chainman.GetCheckQueue().HasThreads()};
2449  2452
      2453  +   event.add_metadata("blockhash="+block_hash.ToString());
      2454  +
2450  2455      const auto time_start{SteadyClock::now()};
2451  2456      const CChainParams& params{m_chainman.GetParams()};
2452  2457
@@ -2612,6 +2617,16 @@ bool Chainstate::ConnectBlock(const CBlock& block, BlockValidationState& state,
2612  2617          }
2613  2618      }
2614  2619
      2620  +   std::string txids{};
      2621  +
      2622  +   for (const auto& tx : block.vtx) {
      2623  +       if (txids.size() > 0) {
      2624  +           txids += ",";
      2625  +       }
      2626  +       txids += tx->GetHash().ToString();
      2627  +   }
      2628  +   event.add_metadata("txids=" + txids);
      2629  +
2615  2630      // Enforce BIP68 (sequence locks)
```

work distribution

measurement: critical-path work timing



Note: 3521 events shorter than 356µs have been enlarged for visibility

measurement: critical-path work timing

Total time: 9h 20m 47s

event type	count	total time	
TX	133684	30.537s total	0.0908% of total
ActivateBestChain	54	3.934s total	0.0117% of total
BLOCKTXN	41	3.599s total	0.0107% of total
INV	147322	1.545s total	0.0046% of total
CMPCTBLOCK	152	1.001s total	0.0030% of total
ConnectBlock	53	571ms total	0.0017% of total
GD	10946	329ms total	0.0010% of total
ADDRV2	2576	72ms total	0.0002% of total
VERSION	80	9ms total	0.0000% of total
HEADERS	212	4ms total	0.0000% of total
VERACK	62	3ms total	0.0000% of total
GETHEADERS	52	1ms total	0.0000% of total
GETBLOCKTXN	4	106µs total	0.0000% of total
ADDR	1	5µs total	0.0000% of total

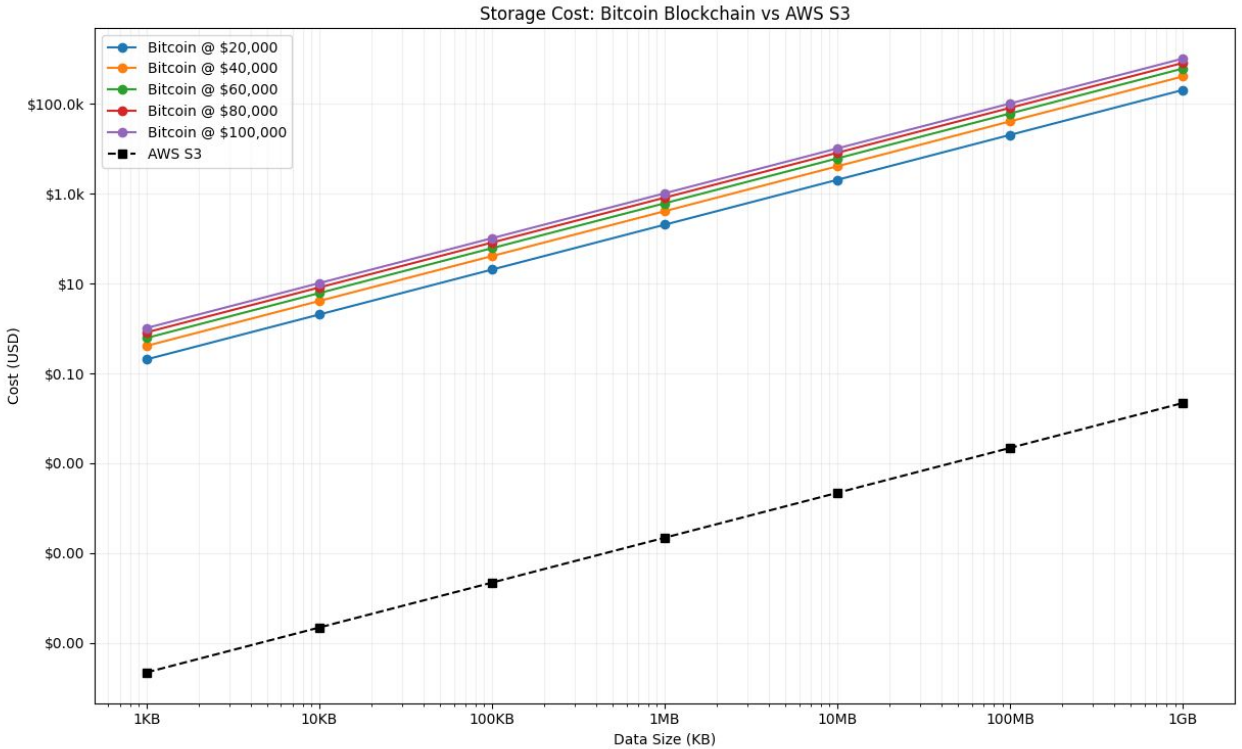
measurement: critical-path work timing

```
Total time: 12h 13m 12s

event type           count              total time

TX                   174830             40.577s total        0.0922% of total
ActivateBestChain    79                 4.123s total         0.0094% of total
BLOCKTXN             43                 3.307s total         0.0075% of total
INV                  219037             2.047s total         0.0047% of total
CMPCTBLOCK           184                1.590s total         0.0036% of total
ConnectBlock         78                 612ms total          0.0014% of total
GD                   7166               193ms total          0.0004% of total
FlushStateToDisk     1                  168ms total          0.0004% of total
HEADERS              380                91ms total           0.0002% of total
ADDRV2               3582               84ms total           0.0002% of total
GETHEADERS           70                 9ms total            0.0000% of total
VERSION              102                7ms total            0.0000% of total
VERACK               68                 3ms total            0.0000% of total
ADDR                 1                  1µs total            0.0000% of total

-------------------------------------------------

Total time spent working: 52.816s
Total time spent working (percent): 0.1201%

Scaled linearly 100x: 12.0%
Scaled linearly 1000x: 120.1%
```

bandwidth required

Definition	Recommended up/down	Total MB/min	Total MB/hr
480p/Standard	1.0 Mbps/600 Kbps	13.5	810
720p	2.6/1.8 Mbps	22.5	1350 (1.35GB)
1080p	3.8/3.0 Mbps	45	2700 (2.7GB)

spam on-chain



actual problems

[46]:

```python
# From gettxoutsetinfo
txouts_on_disk = 188194369
txout_size_on_disk_bytes = 12777162109
avg_utxo_size_bytes = txout_size_on_disk_bytes / txouts_on_disk

ppl = 7_000_000_000

print(f"average utxo size on disk: {avg_utxo_size_bytes:.2f} bytes")

total_utxo_size_gb = (avg_utxo_size_bytes * ppl * 2) / (1000 ** 3)

print(f"total UTXO set size (2 utxo per person): {total_utxo_size_gb:.1f}GB")
```

```
average utxo size on disk: 67.89 bytes
total UTXO set size (2 utxo per person): 950.5GB
```

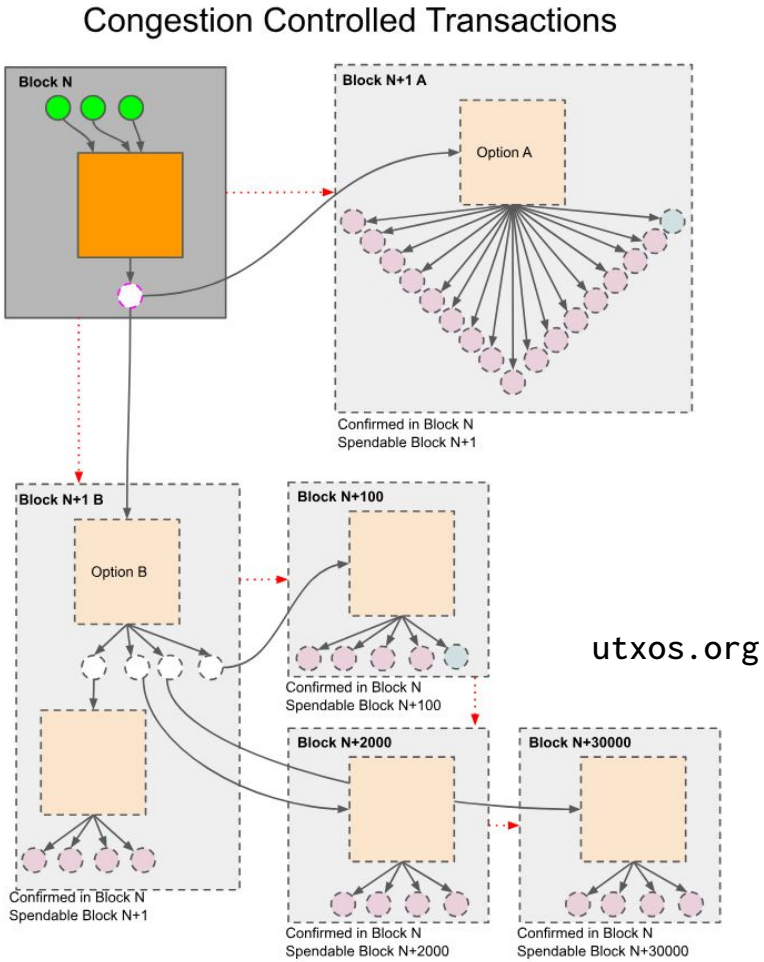
```cpp
// Test setup for benchmarking CCoinsViewCache performance.

/**
 * Benchmark for CCoinsViewCache retrieval performance.
 *
 * This benchmark tests the performance of retrieving coins from a CCoinsViewCache
 * that sits in front of a LevelDB-backed CCoinsViewDB. It creates a large number
 * of coins, stores them in the database, and then measures the time it takes to
 * perform random retrievals.
 *
 * The benchmark mimics real-world usage by:
 * 1. Setting up a CCoinsViewDB using LevelDB in a temporary directory
 * 2. Creating a CCoinsViewCache with a 10GB maximum size
 * 3. Populating the database with coins
 * 4. Performing random retrievals to measure performance
 */

#include "dbwrapper.h"
#include <bench/bench.h>
#include <coins.h>
#include <txdb.h>
#include <validation.h>
#include <util/fs.h>
#include <util/time.h>
#include <primitives/transaction.h>
#include <random.h>
#include <script/script.h>

#include <chrono>
#include <iostream>
#include <vector>
#include <random>

static const uint64_t NUM_COINS_TO_CREATE = 6'500'000'000ULL;
static const uint64_t NUM_RETRIEVALS = 6'500'000ULL;
static const uint64_t MAX_CACHE_SIZE_BYTES = 10ULL * 1024 * 1024 * 1024; // 10GB

static void CoinsViewCacheBenchmark(benchmark::Bench& bench)
{
    fs::path tempDir = fs::temp_directory_path() / "bitcoin_bench_coinsview";
    fs::create_directories(tempDir);

    fs::remove_all(tempDir);
    fs::create_directories(tempDir);

    auto dbparams = DBParams{tempDir, 1 << 24};
    auto coins_opts = CoinsViewOptions{};

```

UTXO set indexing/size

- maybe ditch leveldb?
- (remember, UTXO retrieval is distributed uniformly throughout time in the average case)
  - coins cache gets warm
- utreexo fixes this at the cost of ~2x bandwidth

mitigations

design for exit



utxos.org

“Did they get you to trade
your heroes for ghosts?

Cold comfort for change?

Did you exchange
a walk-on part in the war
for `arm/v7` as a target architecture?”

Roger Waters

Existing tech is underutilized:

cmpctblocks, assumeutxo, pruning

Prospective tech:

utreexo, weakblocks, not-leveldb

thank you

https://github.com/jamesob/blocksize-talk