Upcoming in Bitcoin Core 0.15

2017-08-28

Greg Maxwell

DE47 BC9E 6D2D A6B0 2DC6 10B1 AC85 9362 B041 3BFA

Upcoming in 0.15

- Activity by the numbers
- Main themes and major improvements
  - Performance, Wallet features
- Service bit disconnection
- Ongoing further work

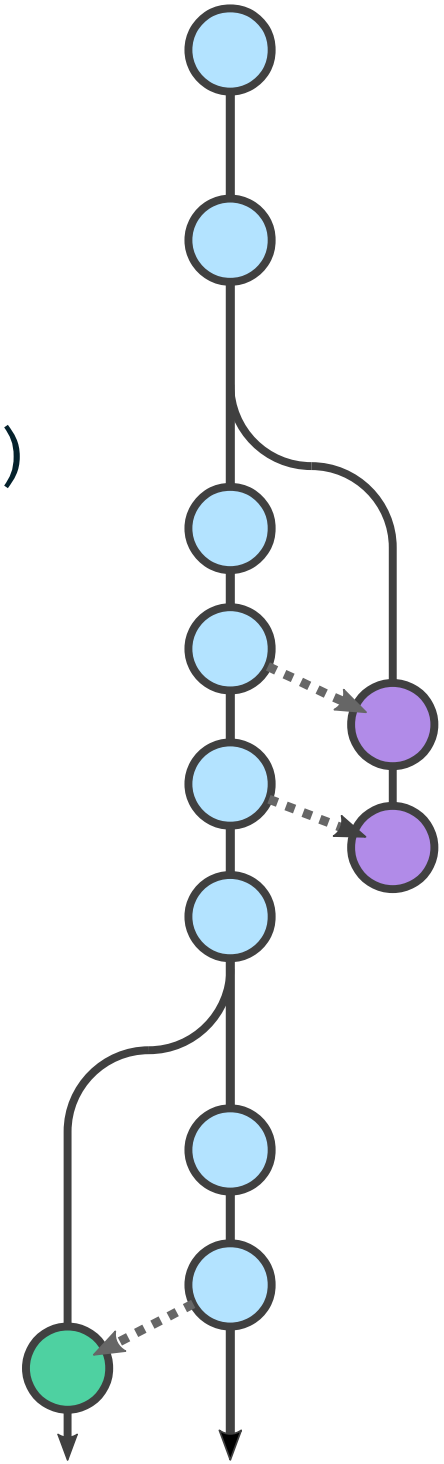
Greg Maxwell

DE47 BC9E 6D2D A6B0 2DC6 10B1 AC85 9362 B041 3BFA

Bitcoin Core releases

- Major releases per six months
  - Fixes backported in point releases (e.g. 0.14.1)
- Feb 17th 0.14 branched off
  - March 8th 0.14.0, April 22nd 0.14.1, June 17th 0.14.2
- Aug 14th 0.15 branched off
  - Aug 16th $0.15.0_{rc1}$, Aug 22nd $0.15.0_{rc2}$
  - Sept 14th or 15th 0.15.0 release (target)

    (original target 9/1; slip due to travel and minor issues)



Activity by the numbers

- ~185 days, 627 pull requests merged
- 1,081 non-merge commits by 95 authors, ~6 commits a day
  - 20% are adding or updating tests
  - Top contributors by commits:
    - 14% John Newbery *(Chaincode)* – 1st commit Nov 2016, now half of all test related commits!
    - 9% Pieter Wuille *(Blockstream)*
    - 8% Matt Corallo *(Chaincode)*
    - 7% Alex Morcos *(Chaincode)*
    - 7% Wladimir van der Laan *(DCI)*
  - 55% everyone else including 7% from 62 people with one or two commits
- 698 files changed, 52373 insertions(+), 19984 deletions(-)
  - About 3k lines changed to review per week

Main areas of focus

- Major performance boosts in IBD, tip relay, and wallet
- Lots of little pieces of polish in various areas, improved reliability and better handling of various corner cases
- With SegWit outstanding some areas have received less attention in 0.15:
  - SegWit dependent wallet features
  - Other new consensus rules
- ... a lot of interesting things there in the future (more at end)

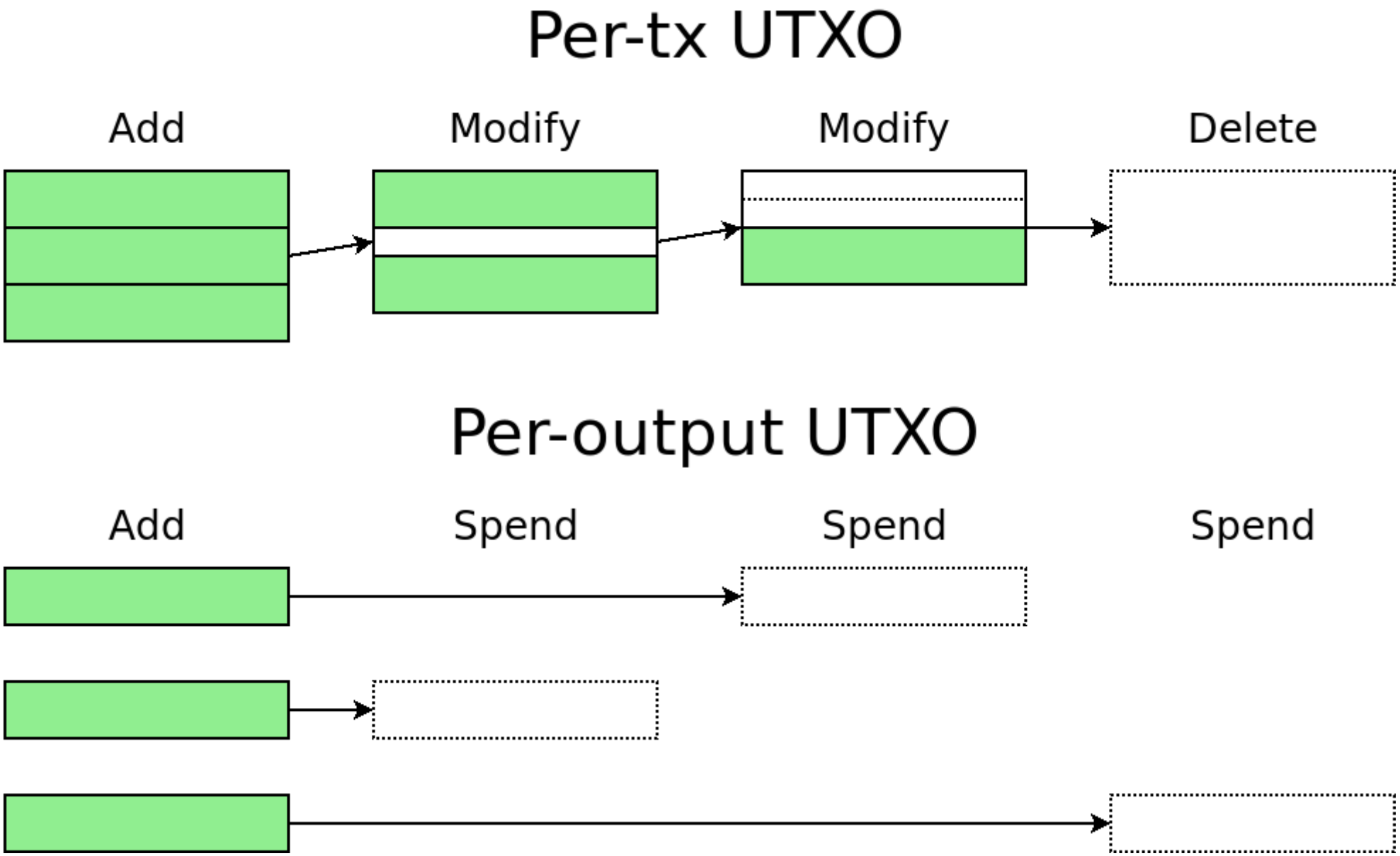
New chainstate (UTXO) database *#10195*

- UTXO database is a compact database with all the blockchain data required to verify a new block, introduced in 0.8.0
- Previously all the txouts for a single tx were in a single record
  - Saves space because of shared key, height, coinbase flag
  - Spending outputs one at a time from a sendmany means a quasi-quadratic IO blowup
  - Combined data means that the software must support modifying records in the database, not just inserting and removing them
  - Memory overhead from reading in whole entries

New chainstate (UTXO) database *#10195*

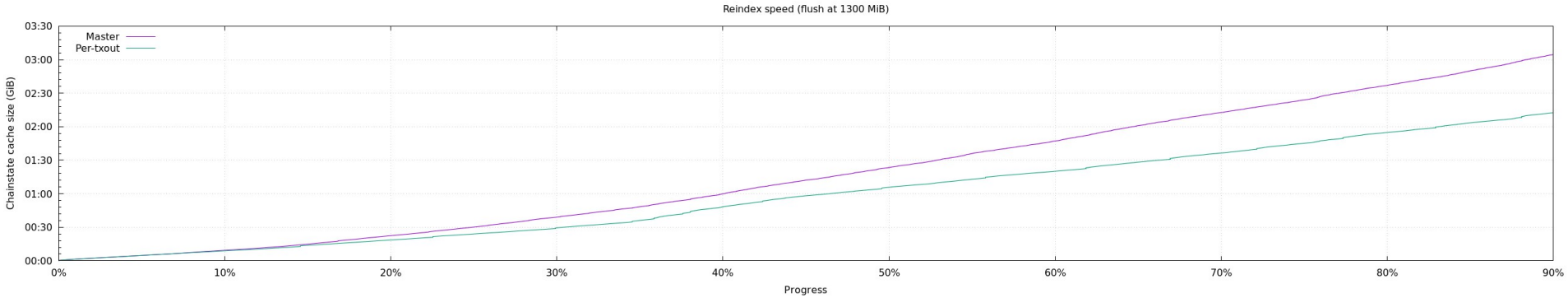
- New structure is one record per txout
  - 40% faster sync
  - 10% less memory used by dbcache (larger caches with the same ram)
  - Flushes less often (faster on hosts with slow IO)
  - Implementation is much simpler
- 15% chainstate size increase (2.8GB now)

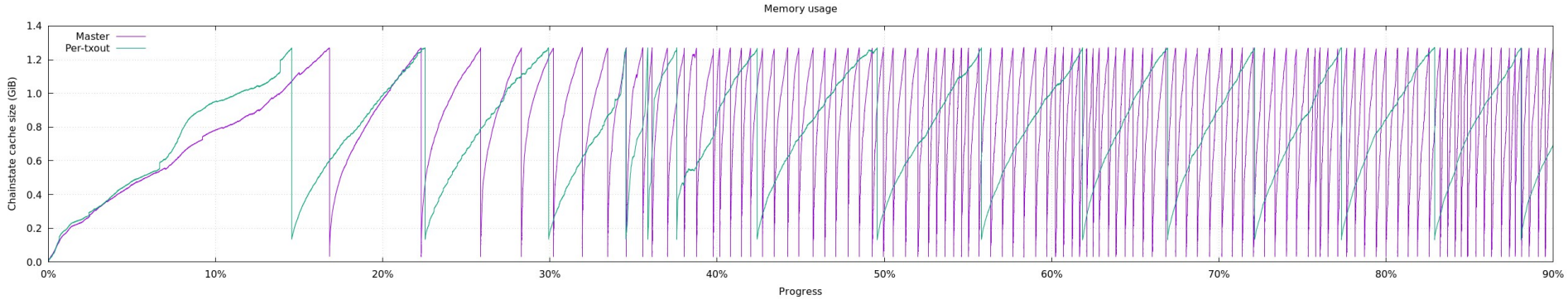
New chainstate (UTXO) database #10195



New chainstate (UTXO) database *#10195*

- Migrates to the new format on startup, takes a few minutes
  (to a half an hour on very slow devices)
  - No going back except through chainstate reindex
- If you ran master and now have a 5.8GB chainstate:
  - Use the hidden -forcecompactdb option





New chainstate database: testing

- Major consensus critical component of the system
- PR had two months of review
  - 245 comments (by github's activity metric)
- In addition to extensive pre-existing and new automated tests we used mutation testing: Walking line by line through the changed code and inserting a bug at each opportunity then verifying that the tests failed with the bug inserted
- Testing turned up a pre-existing non-exploitable crash bug, and several shortcomings in the tests but no issues in the reviewed code

Non-atomic flushing

- Dbcache in Bitcoin Core is more of a buffer than a cache:
  - The performance improvement it provides is primarily from preventing rapidly spend outputs from ever making it to the database
- Consistency requires flushing this buffer completely or not at all.
- Preparing the write demands a temporary copy of the data
- Smarter cache management strategies interact poorly with the need to flush the whole thing at once, all we could do is expire non-dirty data which isn't much of a win post per-txo.

Non-atomic flushing (cont)

- However, the blockchain itself is a write-ahead log!
  - The chainstate doesn't need to be consistent
- Small database change to store pointers indicating the span of blocks that may be partially flushed. Replay on start.
- Far fewer requirements for the chainstate database, no 2x memory usage peak so we can use twice the cache sizes and run on lower memory systems
- Much easier with the per-txout change, since there are no modify operations anymore
- Opens the door to fancier cache management and lower latency flushing in the future

Platform acceleration

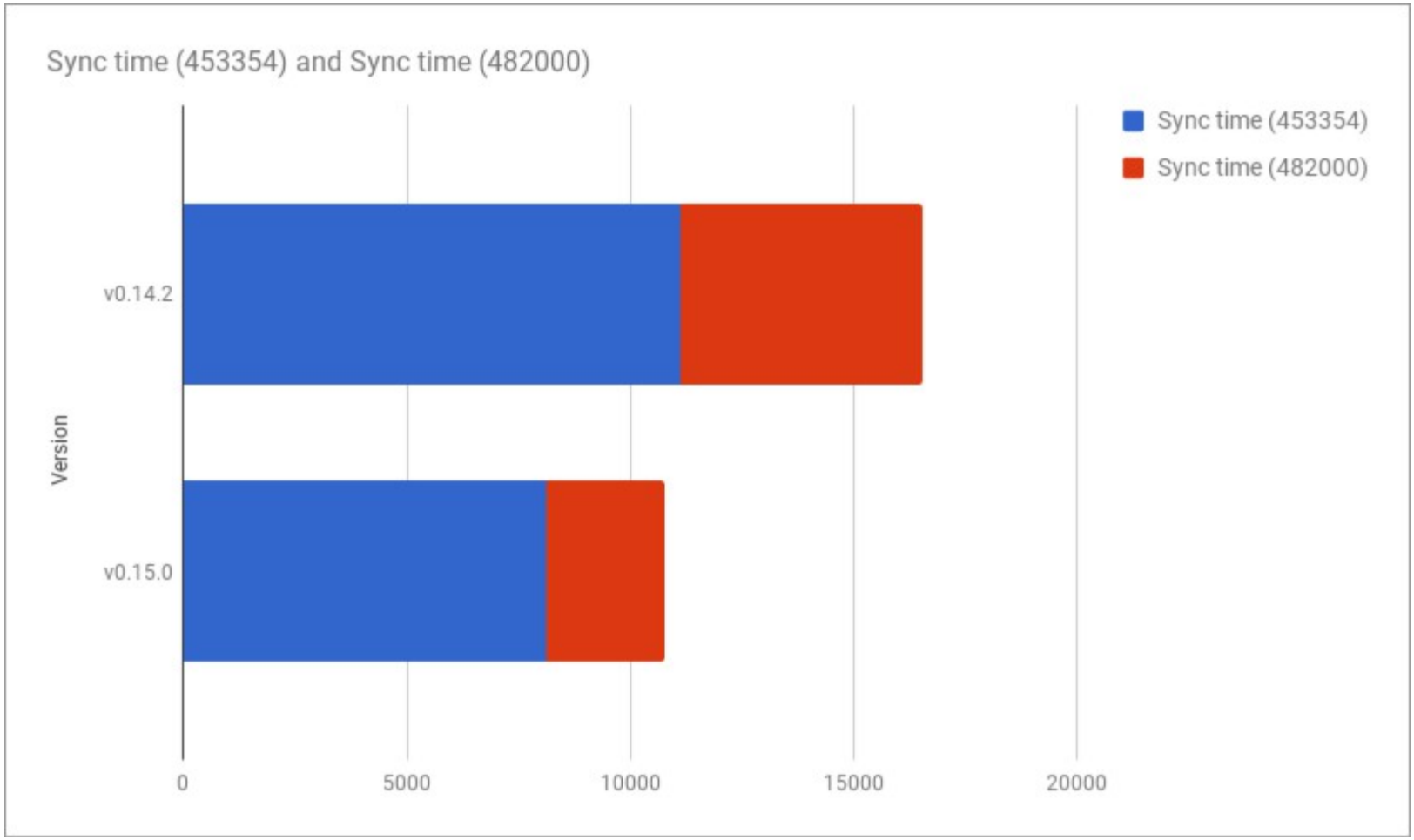
- Use SSE4 assembly implementation of SHA256
  - 5% speedup in IBD, similar for tip connect
  - Currently behind a –enable-experimental-asm flag
    - Finished right before feature freeze and then we spent three days trying to figure out why it crashed on OSX (linker was optimizing it out)
  - Support for SHA-NI is implemented but not merged (additional 10% speedup on ryzen)
- Databases' CRC32 uses the SSE4.2 instruction for CRC

Script validation caching

- Bitcoin has had signature validation caching since 0.7
  - Cache entry per EC signature, stops DOS and a speedup
- Still doing script validation which includes signature hashing, which is slow esp for large non-segwit transactions
- Can't use presence in mempool as validity proxy
  - mempool validity rules are different, would massively increase consensus critical code size plus past bugs
- Cache keyed on H(salt|txid|flags) change in validation flags results in harmless cache misses
- 50% speedup accepting blocks at the tip

Other speedups

- DisconnectTip (reorgs) made much faster-- perhaps tens of times faster for large reorgs-- by deferring mempool processing
- Cache compact block messages instead of reconstructing them for each peer
- Wallet key generation made 20x faster by deferring flushing
- Minor secp256k1 crypto speedups on the order of 1%


Sync time (453354) and Sync time (482000)
Sync time (453354)
Sync time (482000)
Version
v0.14.2
v0.15.0
0
5000
10000
15000
20000

Multiwallet

- CLI / RPC only right now (GUI likely in 0.16)
- Run multiple wallets at once, specified with -wallet conf or command-line arguments
- Each wallet is completely separate
- Select wallet with -rpcwallet in bitcoin CLI or /wallet/<name/ endpoint in the RPC
- Very useful with pruned nodes to keep many wallets in-sync
- Consider it somewhat experimental: The interface is explicitly unstable (we had a hard time agreeing on what it should be) and may change in the next version.

Improved fee estimator

- Tracks estimates on three distinct time horizons to make estimates more responsive to faster changes
- Supports two different estimates 'conservative' and 'economical' – economical responds faster but is more likely to get caught off-guard; used for BIP125 replaceable tx.
- Outputs detailed information for use by external estimators
- Can produce estimates for up to 1008 blocks and produces results which are more precise and reliable

Improved fee handling

- Support for opting into replacability and bumping fees in the GUI (previously RPC only)
- Last remaining known corner cases where automatic coin selection could overpay fees are fixed
  - Could show up in 'dusty' wallets where an initial coin selection picked many coins and then it picked few coins when it realized it needed more fee, if there was no change in the first pass or substract fee from amount was used it could overpay.
- Wallet is smarter about not producing change which would be uneconomical to spend and will slightly overpay fees to avoid doing so, with a maximum controllable via -discardfee

HD Rescan and auto-topup

- Default keypool (lookahead) increased to 1000 from 100
- Initial HD wallet support didn't advance the keypool as keys were used sparsely, could lead to missed payments when restoring a HD wallet from a backup
- Now it automatically marks each key it sees on the blockchain as used and all keys earlier than it and will keep refilling the pool
- Handling of encrypted wallets is still not perfect: reliable restore with an encrypted wallet requires unlocking and starting a rescan.
- There is now an abortrescan rpc to stop rescans

RNG improvements

- Many version long effort to eliminate any use of OpenSSL
  - All that remains is the RNG and use in HTTPS in QT
- Most RNG use replaced with a ChaCha20 stream cipher, including all the prior uses of a non-cryptographic RNG (which was hardly faster)
- Several incidents with OS RNGs (freebsd and netbsd) in recent years are a cause for concern, along with VM state restores and other corner cases
- Long term keys and seeding of the ChaCha20 streams uses

  state[32], rand[32] = SHA512(OpenSSL|OSrng|HWrng|state|counter)

Disconnecting service bits 6/8

- Worth mentioning only due to widespread misunderstanding
- Bitcoin's partitioning resistance is heavily driven by being very greedy about keeping helpful and long uptime connections
- Network topology cannot suddenly change without causing disruption, with nodes isolated & stuck hunting for peers and finding good ones have all their connections full
- SegWit front loaded topology changes with 0.13.1 causing them to preferentially connect out to other segwit peers, so any problems would happen slowly and while the node operator was around, rather than in a flash when segwit activated

Disconnecting service bits 6/8 (cont)

- 0.15 will disconnect service bits 6 or 8 until Aug 1, 2018
  - Corresponding to S2X and Bcash
  - Incompatible consensus rules which unfortunately use the same P2P port and magic, pretends to be compatible but then effectively DOS attacks by not being compatible
- S2X developer complained about premature partitioning, but that concern is unjustified:
  - Prior versions will still happily accept their connections. Past upgrade rates indicate they would be okay for years (if not for their rule changes two months after release).

Future work in-flight

Keep in mind: Bitcoin Core is an open collaboration

"*Instead of a roadmap, there are technical guidelines. Instead of a central resource allocation, there are persons and companies who all have a stake in the further development of the Linux kernel, quite independently from one another: People like Linus Torvalds and I don't plan the kernel evolution. We don't sit there and think up the roadmap for the next two years, then assign resources to the various new features. That's because we don't have any resources. The resources are all owned by the various corporations who use and contribute to Linux, as well as by the various independent contributors out there. It's those people who own the resources who decide.*" – Andrew Morton on the Linux kernel

What happens in the future depends critically on what people want to contribute.

No single person can tell you what the future will be, but I can tell you some of the things I know people are working on and think we could see (mostly in 0.16.0 – 0.16.1)

SegWit fully supported in wallet

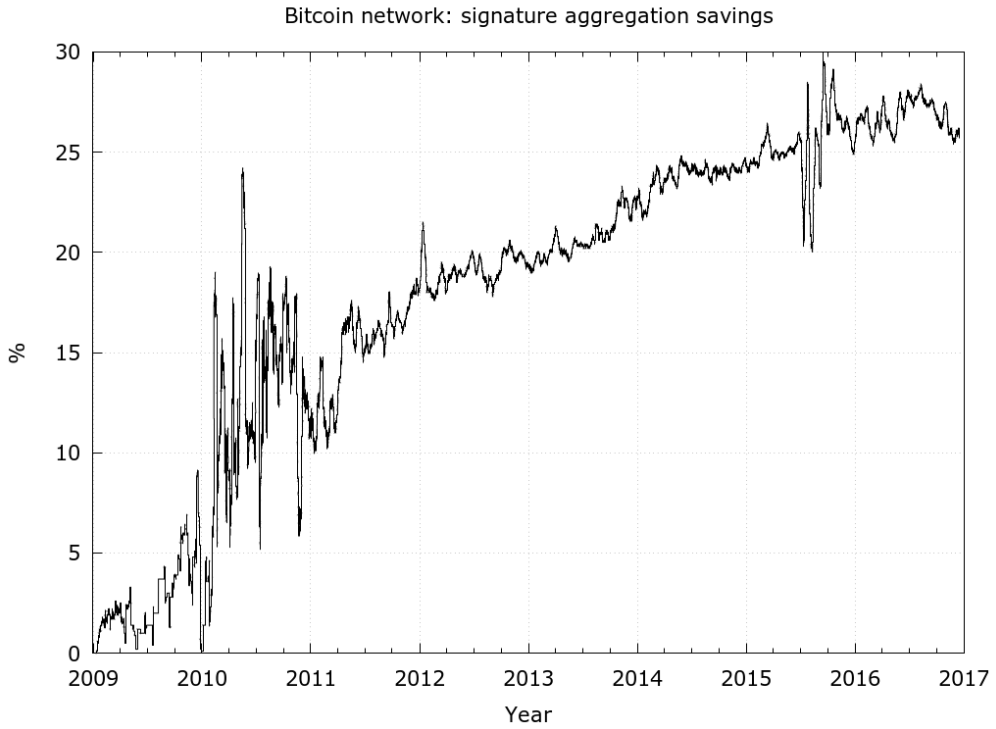
- There is RPC support for segwit, as well as most of the required wallet backend machinery.
  - This support was primarily intended for testing segwit.
  - In particular, addwitnessaddress isn't especially backup durable
- Per the segwit developer guide we held off offering segwit as a user facing feature until post activation due to complexities around the transition and reorg security right at activiation.
- Planning on a short release right after 0.15.0 with full support including sending to BIP173 (bech32) addresses

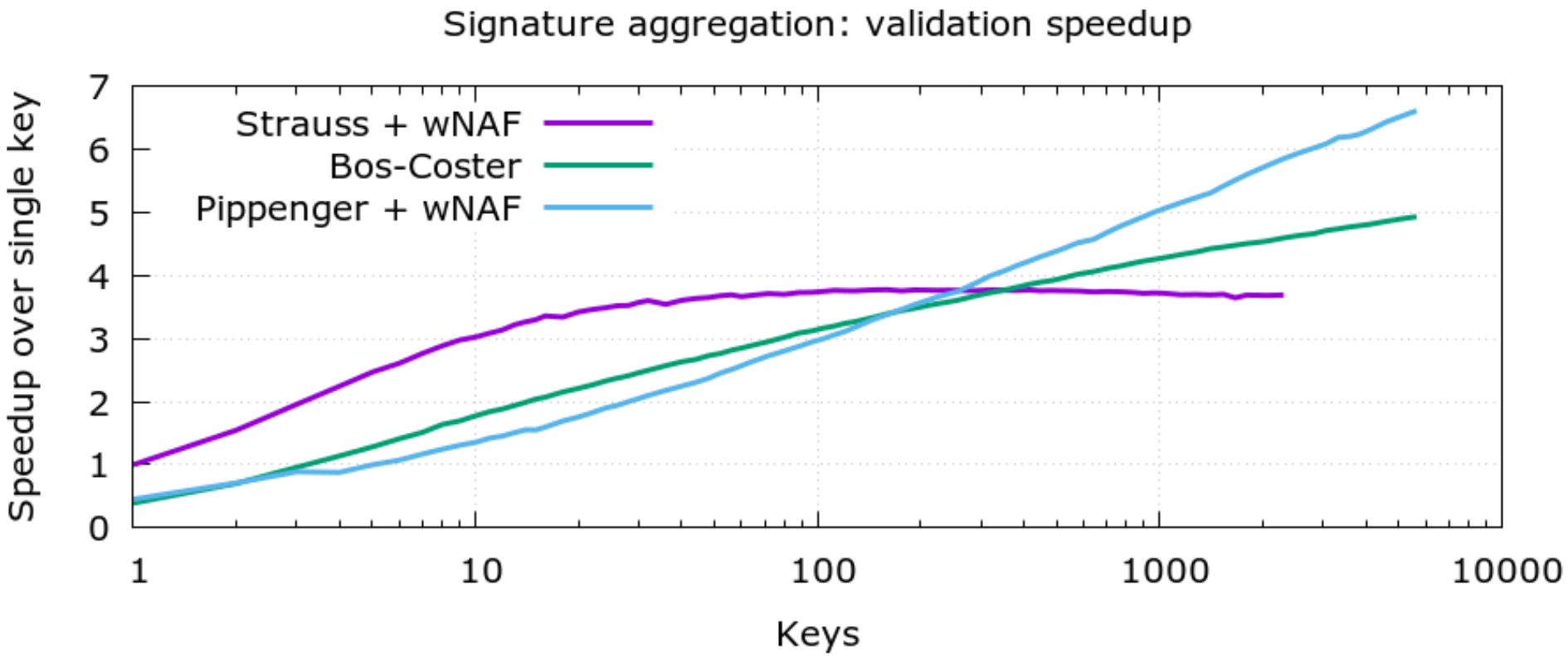
Further wallet improvements

- Automatic fee bumping, where the wallet presigns timelocked bumps and automatically merges new payments as you make them. Transaction malleability created near intractable corner cases that complicated doing this before
- Support for hardware wallets and easy offline signing
  - Andrew Chow's BIP-PSBT (partially signed bitcoin tx)
  - Much easier to do safely and efficiently with segwit
- Branch and bound coin selection that can produce changeless outputs much of the time
- GUI Multiwallet, Full block lite mode, HTLC txn *(BIP199)*, CSV txn
- GUI/Backend process separation (there are patches...)

Network and consensus

- Rolling UTXO set hashes (PR#10434)
  - Efficiently maintain a cryptographic hash of the UTXO set that can be updated at every block, allowing for easier state comparisons and new initial sync methods
- Signature Aggregation (google for my BCT post)
  - Pieter, Andrew Poelstra, and myself wrote a paper on the technique for FC'17-- a reviewer found an earlier paper with an almost identical approach, we switched to it
  - We have the crypto implemented and are playing with performance optimizations
  - Batch validation and aggregation can be combined


Bitcoin network: signature aggregation savings
%
Year
0
5
10
15
20
25
30
2009
2010
2011
2012
2013
2014
2015
2016
2017


Signature aggregation: validation speedup
Strauss + wNAF
Bos-Coster
Pippenger + wNAF
Speedup over single key
Keys
0
1
2
3
4
5
6
7
1
10
100
1000
10000

Network and consensus (cont)

- Encrypted and optionally authenticated P2P (BIP 150/151)
  - Has been waiting on networking refactorings that are almost done
- Private transaction announcement (see “Dandelion” paper)
- Peer interrogation to more rapidly ditch peers on incompatible consensus rules
- Improved block fetching robustness (e.g. multiple compact block fetches)
- Bloom “map” (gcs-lite-client BIP)

Further further...

- SegWit made script enhancements easier to make and reason about their safety, there are many things in early draft stage people are working with
  - Including whole replacement script systems, minor improvements like merkle tress in script
- POW as an additional connection priority mechanism (BIP154)
- PIR methods for privately getting transaction histories to wallets on pruned nodes
- Efficient mempool reconciliation
- Compact tx serialization (>25% reduction in history size)

In summary...

- Many important improvements in 0.15
  - including across the board 50% speedups!
- Many exciting new things being developed

  and I didn't even mention the more speculative things...
- Many new contributors joining in and ramping up, including more paid contributors from several different orgs
- More contributors are always welcome
  - Testing, GUI, and the wallet in the greatest need of more attention

Thanks for your time.

Greg Maxwell

DE47 BC9E 6D2D A6B0 2DC6 10B1 AC85 9362 B041 3BFA