Reducing the Disk Requirements of Blockchains

Nicholas Gordon
04/23/2020

Blockchains: They’re great

- If the hype is to be believed, blockchains can solve all problems
- Secure, trustless, verifiable, incentivized!
- *“Why isn’t everyone using them?*” you might ask
- Because I don’t have two weeks to bootstrap my phone!

The problem: blockchain size is out of control

- As I have said many times this term, Bitcoin and Ethereum are enormous.
- For today's numbers:
  - Bitcoin: ~273GB
  - Ethereum: ~332GB (full node this time)

But why is that a problem?

- Research indicates it takes anywhere from *days* to *weeks* to sync a full Bitcoin node
- Ain’t nobody got time for that
- Except maybe large corporations
  - What was that about distributed, trustless ledger?
- Also Bitcoin on a raspberry pi could be cool

The plan

- Survey – can’t do something new until you know what isn’t new anymore.
- After a month or so of recursively following related works and keyword searches, all papers were found.
- 21 made the cut, and all were read (mostly)

The results

- Broadly speaking, works fell into three categories:
  - Summarization
  - Pruning
  - Structural change
- We'll go over each category and talk about it and pick out some interesting examples

Are the results black-and-white?

- No. Most of these fit at least partially into two categories.
- I decided which category they fit most into
- Also, if the work was primarily about something else, I disregarded that other part and focused on the storage requirements part.
  - Remember what I said earlier about time?

Summarization

- This could have been called compression, abbreviation, or condensing.
- All works here try to more efficiently represent the same information.
- Always you lose some information, such as full transaction traceability.

Mini-blockchain

- We already talked about this, but it is an archetypal entry.
- First well-known attempt at this sort of thing (2014)
- Account tree is the summarization part
- Remove old block bodies to lose redundant information
- Keep headers to retain security

Empowering Light Nodes in blockchains with block summarization

- A long title, ironically.
- Proposes to summarize every *m* blocks into a summary block, repeated in a way that eventally the chain is *g* guard blocks and then an arbitrary number of summary blocks.
- Then, every *m* summary blocks, summarize them again!
  - A “summarization tree”
- An interesting middle ground between SPV where you can only verify a transaction is in a block and a full node

Blockchain Abbreviation

- Pretty straightforward idea: compress all blocks into a new genesis block.
- Have to adjust block difficulty, change forking protocol to ensure abbreviation adoption
- Security problems? What about contracts?

Pruning

- All techniques aim to delete unneeded information. Sometimes radically, by losing whole blocks.
- In some cases, overlaps with “summarization” depending on the angle you look at it from

Downsampling Blockchain Algorithm

- Borrows a technique from digital signal processing world, downsampling
  - A downsampling factor of 1/*M*, where you keep only *M* samples
- Use *statistics (!)* to find a probability function for how many UTXO a block has
- Then, only keep the *M* highest scoring blocks.

Selective Blockchain Transaction Pruning and State Derivability

- Also talked about this one
- Everyone cares about a different set of transactions in the network, and few people care about all transactions
- Allow an arbitrary predicate to decide what transactions to prune
- Check blocks, apply predicate to transactions, delete matching transactions

Temporal Blockchain: A Formal Analysis

- Use a formal language to build a specification of blockchain
- Also, half an attempt to solve the size problem
- Only keep blocks that are younger than *n* days old, say 30 days old.
- Use a “checkpoint” to make deletion safe, then delete them. No mention of how to checkpoint, though.

Structural Change

- All techniques in this category take a more “radical” approach to solving the problem
- Often feature proof-of-work changes, sometimes even depart from the block*chain* idea.
- Perhaps the most promise, but requires the most work

Proof-of-property

- Why do nodes have to keep state? To verify transactions.
  - Again: why do *nodes* have to keep state? Why not make the transaction submitters do it?
- Extend Ethereum's per-block system state to include a "validation path" with transactions that contains all the necessary state for a node to verify that the transaction is valid and can complete, i.e. has enough gas

Section/Segment Blockchain

- A bizarre duo of papers that are very similar and only a little different by the same author
- Utilize a “proof of storage” that forces miners to retain segments of the blockchain, which contain consecutive blocks.
- Distributed by “occupations” that nodes apply for and they can get fired by not showing their proof of storage.
  - This incentivizes nodes to store blocks instead of relying on altruistic full nodes to do it

Rollerchain

- Probably the inspiration for “proof of storage” concepts
- Extend the Bitcoin POW to require “tickets” that you obtain by storing state snapshots, in this case a snapshot of UTXOs at the time of that block
  - Required for mining
- Again, not trying to “invent the problem away,” but give nodes a reason to actually retain blocks.

What do I think?

- After the mini-blockchain presentation I began to suspect we could not have our cake and eat it, too
- There may be irreconcilable tensions between some aspects of blockchain that we must rank by their importance to us
- That will dictate which of the techniques we use and the results we can get from them

Questions, Comments?