Empowering Light Nodes in Blockchains with Block Summarization

Asutosh Palai[1*], Meet Vora[2*], Aashaka Shah[3*]

***Abstract*—Blockchains have revolutionized the storage of data in an immutable, transparent and non-centralized way. However, public blockchain systems like the Bitcoin system face a problem of scalability, primarily due to the significant and growing size of its blockchain. This paper introduces a method, termed block summarization, which reduces blockchain storage overhead for systems having transferable transactions. The proposed method allows resource-restricted light nodes to store a form of the blockchain such that it can validate the transactions independently which ultimately reduce dependency on full nodes. This way, we can achieve a middle ground between Simplified Payment Verification (SPV) nodes which can only verify the membership of a transaction in the blockchain, and full nodes with pruning enabled which can only support pruning provided they have an infrastructure of full nodes. We implemented our algorithm for a custom blockchain using Bitcoin blocks and were able to achieve a compression ratio of 0.54.**

Keywords: Blockchain, bitcoin, light nodes, block summarization.

I. Introduction

The blockchain is immutable, cryptographically secure and a distributed ledger with high Byzantine fault tolerance. It is growing in popularity as a means to record transactions (financial or otherwise) permanently without relying on a centralized entity. Through the use of a peer-to-peer network and a distributed timestamping server, a blockchain database is managed autonomously.

The current blockchain architecture and concepts owe their origin to the Bitcoin [10], which made use of blockchain to solve the problem of double spending and distributed consensus, without the use of a trusted authority. It consists of blocks of transactions which are linked to one another through hash pointers. The ledger property of a blockchain is satisfied if it can enable valid immutable transactions and maintain their integrity throughout. The content of the transactions can vary from application to application.

One of the significant challenges in blockchains is ensuring scalability [2], followed by the challenge of preventing privacy leakage and selfish mining [13]. The scalability of blockchains is hampered by their ever-growing size due to new transactions along with slow transactions validation rate. It is particularly daunting for public blockchains like Bitcoin blockchain which currently exceeds 100 GB [1] in size to scale efficiently. The enormity in size creates a set of problems including initialization of new nodes, verification of new transactions by non-miner nodes and functioning of light nodes. This storage overhead of blockchains has motivated a variety of approaches. Some of the existing solutions to the issue involve the use of *light nodes* and *pruning* [3]. Light nodes are those entities that prefer to store only a subset of the blocks and rely on full nodes for operations requiring the complete blockchain. Light nodes implement Simplified Payment Verification where block headers are downloaded during initial sync and later request transactions from full nodes as needed. In SPV, the lightweight clients can only verify that a transaction is included in the Bitcoin blockchain. Another idea is that of pruning, in which the blocks are validated and used to make a block database. As newer blocks are generated, the older blocks are dropped from the memory, whereas the index and block database entry created for these blocks remains. Pruning was introduced in the 0.11.0 version of Bitcoin Core.

In this paper, we propose a possible solution to the blockchain size problem, specifically to allow transaction validation on light nodes. We are targeting a particular subset of applications where the transactions deal with transferable (partially or fully) entities, and a net change can be computed for a series of transactions. For example, our approach would be applicable for transactions dealing with bank account [11] or land agreements but not for storing identities [8]. We propose that instead of storing all the blocks in all the nodes, nodes can store the change in a sequence of blocks (called block summary). This summary can store all the input resources for the given blocks and the total change that was introduced by these blocks. Since we expect a lot of transactions in the block to transfer further (in a transitive manner), block summarization can reduce the storage space required for those transactions which have no further effect (except for verification of the resource inheritance chain).

The organization of the paper is as follows: Section 2 presents background and related works. Section 3 describes in detail our proposed framework. Section 4 describes the results and implementation. Finally, Section 5 describes our challenges, conclusions and future works.

II. Background

Several approaches have been used to deal with the large size of the blockchain. There is research which focuses on reducing the size of the blockchains by changing implementation or tweaking with the data structures. As mentioned earlier, pruning is an implementation approach in which a node drops past data while storing its important part in a table data structure locally. However, pruning comes out to

* All authors contributed equally to this work.
[1] A. Palai is with Indian Institute of Technology Roorkee, Roorkee, Uttarakhand 247667, India. Email: asupl.ucs2014@iitr.ac.in
[2] M. Vora is with Indian Institute of Technology Roorkee, Roorkee, Uttarakhand 247667, India. Email: meetv.uee2014@iitr.ac.in
[3] A. Shah is with Indian Institute of Technology Roorkee, Roorkee, Uttarakhand 247667, India. Email: ash96.ucs2014@iitr.ac.in

978-1-5386-3662-6/18/$31.00 ©2018 IEEE

be too complex. A cryptocurrency, cryptonite [5], deals with blockchain size using the concept of mini-blockchain [4]. It involves pruning out the old transactions, maintaining account balances in a hash tree structure called account tree and maintaining the complete list of blockchain headers [9].

There is also research which aims to make blockchains more accessible for light nodes or thin clients. These are systems which are neither able to store the entire blockchain nor implement the necessary checks for transaction validation. They prefer to store a subset of the blocks and rely on peers for exhaustive operations. Usually constrained by space or processing powers or both, they are primarily used for transaction creation. As a result, light nodes are unable to validate the incoming transactions or mine new blocks. Such nodes employ Simplified Payment Verification to verify that a transaction has made it into the blockchain, without storing the entire blockchain. Instead, it stores the hashes of the block header of the longest chain as queried from the network. The transaction can be verified to exist on the blockchain if a Merkle branch from a block leads to the transaction hash. The SPV approach is quite popular and is also used by Tomescue et al. in their system Catena [12] to enable thin clients to act as Bitcoin witnesses and agree on a log of application-specific statements managed by an adversarial server. However, in SPV, the entire blockchain needs to be downloaded again to access a full block which had been pruned earlier. Further, the current implementation does not support the rescanning of the blockchain. Hooff et al. proposed a different way for the light node to exist in their paper VerSum [7] which allowed light nodes to securely outsource expensive computations over Bitcoin blockchain to multiple servers. However, not all type of computations can be securely performed and only include wallet queries like determining the address balance, all incoming or outgoing transactions to an address, and other global statistics.

The major advantage of thin clients is that they consume less amount of resources which is a crucial constraint for nodes running on mobile or embedded devices which are low on storage and processing power. Such nodes are important in a given blockchain system as they serve as actual usage endpoints. For example, a shop owner might use an embedded device to validate transactions before accepting a bitcoin payment. Some powerful systems may also resort to thin clients because of the large size of the blockchain. They are able to perform the transactions and verify their inclusion in blockchain without expending huge disk-space or network bandwidth required for bootstrapping the chain and maintain the integrity of the chain by checking all the transactions. Hence, our solution tries to maintain a summarized version of the complete blockchain which can be stored by the light nodes.

Input : l consecutive blocks B
Output: S, summary of B blocks

Function *SummarizeBlocks(B: Block[], l: int) : SummaryBlock* **is**
  $T \leftarrow Map < Hash, Transaction >$
  $O \leftarrow Set < Transaction >$
  $S \leftarrow newSummaryBlock$
  for $i = 0$ **to** $l - 1$ **do**
    foreach t *in* $B[i]$ **do** $T.insert(trx.hash, trx)$
  end
  for $i = 0$ **to** $l - 1$ **do**
    foreach trx *in* $B[i]$ **do**
      foreach $input$ *in* $trx.inputs$ **do**
        if $input.hash$ *in* T **then**
          $ptrx \leftarrow T[input.hash]$
          $ptrx.markSpentOutput(input.num)$
        else
          $O.insert(trx)$
        end
      end
    end
    $S.addHash(B[i].hash)$
  end
  foreach trx *in* $T.values$ **do**
    if $trx.unspentOutputs \neq 0 \wedge trx \notin O$ **then**
      $S.insert(trx)$
    end
  end
  foreach trx *in* O **do** S.insert(trx)
  return S
end

Algorithm 1: Summarizing a sequence of blocks

III. Proposed Solution

The inability of light nodes to validate transactions is due to the lack of access to the complete blockchain. Even through SPV, light nodes can only verify if a transaction belongs to a given block, which in itself is coupled with a couple of security issues. Firstly, it is unable to validate the actual transaction resulting in consequences of forged transactions if the majority of its connected network is an attacker. Secondly, a full node can pinpoint the identity of light nodes which query and interact with them [6]. Our proposed method of *block summarization* for light nodes aims to maintain the lightness of thin clients while at the same time providing some of the powers belonging to a full node. Through block summarization, we intend to replace the actual blocks with their corresponding *summary blocks* which contain the essential details such as spent transactions and unspent outputs across the original block summary. This would enable the thin clients to verify the incoming transactions. Moreover, in situations where the nodes are computationally powerful, one might also be able to mine next blocks. This would greatly help to reduce the dependency of light nodes on the peers.

A. Simplified transactions

Summarization can be applied on those blockchains where transactions perform generation, transfer or destruction of entities which can be added or subtracted from one other. Further, the entity being transacted should no longer be

Input : C blockchain; blocks to be summarized in l chunks in each depth; o blocks to be left at head of chain; m blocks are to be left at each depth
Output: nCnew blockchain with summarized blocks

Function *SummarizeChain(C: BlockChain, l: int, o: int, m: int) : BlockChain* **is**
  $B \leftarrow newBlock[\ ]$
  $B.insert(C.blocks)$
  $end \leftarrow C.maxHeight - o$
  $depth \leftarrow 1$
  while $end \geq l$ **do**
    $BS \leftarrow newBlock[\ ]$
    $i \leftarrow 0$
    while $i + l \leq end$ **do**
      $S \leftarrow SummarizeBlocks(B[i : i + l - 1], l)$
      $S.setDepth(depth)$
      $BS.insert(S)$
      $i = i + l$
    end
    $end \leftarrow BS.length - m$
    $BS.insert(B[i : B.length - 1])$
    $B.swap(BS)$
    $depth = depth + 1$
  end
  $nC \leftarrow newBlockChain$
  $nC.blocks \leftarrow B$
  return nC
end

Algorithm 2: Summarizing an entire blockchain

available with its previous owner after the transaction, i.e. it is neither shared nor replicated. Simply stated, our approach targets those specific systems where a sequence of transaction can be stated in terms of net change, i.e. intermediate inputs and outputs can be canceled out.

B. Summary of blocks

A summary of a sequence of blocks (not necessarily the entire blockchain) represents the net change introduced by them. A *summary block* is a new block that represents the net difference or addition in value introduced by the blocks it summarizes. When a given sequence of blocks is replaced by its summary block, the net effect in the blockchain remains the same.

A summary block can be viewed as a single big transaction whose inputs are the inputs of all the transactions of its constituent blocks and whose output is all the unspent outputs of the same. The outputs which are inputs to any transaction in the given block sequence would end being canceled out and replaced by their net effect. Further, the outputs of the summary block are tagged with the transaction IDs and output index so that any future transaction referring them can be validated. Summary blocks also contain the hashes of the included blocks and, if required by the protocol, bounding heights, so that the summary block containing a given original block can be located. Algorithm 1 describes this step in detail.

C. Summarizing the blocks

Due to the frequent forking of the blockchain near the tip, re-summarizing the chain for every such incident would lead to massive overhead. Since there is no fixed depth to guarantee the avoidance of fork, we aim to find a trade-off between the quantity of blocks left untouched and the ones which have been summarized. To minimize the overhead in case of a fork and also facilitate easy verification of the summary blocks, we summarize consecutive blocks in chunks of fixed length l leaving a chain of original blocks of length o from the tip. Thus, on the addition of a new block to the original blockchain, we look for $o + l$ unsummarized blocks from the tip. If true, we replace the l-long sequence of blocks (after o blocks) with a summary block.

An equivalence can be shown between the chain of summary of blocks and original blockchain. Until now, even a chain of summary blocks would pose the same size problems as the original blockchain when the summarized chain too becomes huge. This can be solved by taking a summary of summary blocks in a similar chunked and offset method.

D. Summarization tree

Given the previous definition, each summary block is a summary of l consecutive blocks. Here, we propose to *recursively* summarize the l summary blocks after m summary blocks from the tip. We define *summary depth* as the number of summaries taken to obtain a summary block. Thus, summary depth for the original blocks is 0 while the summary depth of direct summaries of original blocks is 1. Similarly, the summary depth for a summary of depth 1 summary blocks is 2. Thus, summary of depth x summary blocks is of $x + 1$ depth.

The recursive strategy to summarize summary blocks as described in Algorithm 2, can be explained as:

- The first o blocks (from the tip) of original chain remain untouched.
- The next m blocks will be summary blocks (of depth 1), each of l original blocks.
- The next m blocks will be summary blocks (of depth 2), each of l blocks of depth 1, i.e. of $l * l = l^2$ original blocks.
- In general, there will be m blocks of depth k covering l^k original blocks and so on.
- There may be a block sequence of length p near the genesis block which is either left untouched or is summarized up to a certain depth less than the max height achieved before these blocks.

As shown in Fig. 1, this arrangement evidently forms a geometric sequence where the number of blocks n can be related as:

$$n = o + m \times l + m \times l^2 + .. + m \times l^k + p \qquad (1)$$

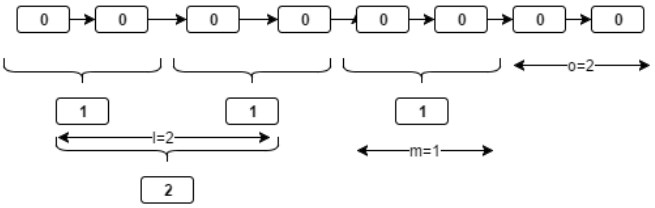

Fig. 1: Recursive summarization tree

l	**overall**	**depth 1**	**depth 2**	**depth 3**
10	0.62	0.87	0.76	0.68
20	0.59	0.82	0.68	0.59
25	0.57	0.80	0.65	0.57
40	0.60	0.75	0.60	-
50	0.57	0.74	0.57	-
75	0.54	0.69	0.53	-

TABLE I: Overall compression and the average compression at different depths

E. Handling of forks (maintaining consensus)

We assume that the blocks which are being summarized are already verified and thus, the corresponding summary blocks can be treated as verified blocks. The immutability property of blockchain ensures that the summarized blocks need not be undone. But if the blockchain gets diverged (forked) and the forked chain becomes longer, as per longest chain rule, we have to accept the version of blockchain with the highest amount of work to achieve consensus.

If the fork takes place in the first o blocks, then it can be handled normally as with a regular blockchain. In case a fork takes place at a block summarized in some summary blocks, we need to locate the corresponding summary block by traversing backwards and comparing the hash values. A given summary block stores the height range and the hashes of the blocks it summarizes. Thus, on an encounter with a summary block with modified hashes, we replace the following blocks with new ones and re-summarize. To facilitate this, we fetch the new blocks from the full node peers.

IV. Experiment And Results

We tried to implement the above algorithm for our custom blockchain implementation without features like smart contract and seeded them with Bitcoin blocks from height 200000 to 250000. We choose Bitcoins because on ignoring the scripts, the Bitcoin transactions can be simplified as a transfer of resource (Bitcoins) from one entity to another. We downloaded the blocks by running a full Bitcoin node. To measure the compression, for different values of l, l consecutive blocks were summarized by removing all the transactions which are created and were completely spent within these blocks. All other transactions, the ones with input from outside of these l blocks and the ones with some unspent transaction were included in the summary. No modifications were done on transaction level since that would have modified the *transaction hash*, its only identity as per the protocol, but for other blockchains inputs in the same set and the spent outputs could also be removed from partially spent transactions. Then we were able to verify new transactions successfully, without the scripts[1]. The compression achieved for different values of l are given in table I.

As it can be observed from Fig. 2, the compression ratio (the ratio of sum of sizes of compressed blocks to that of the original blocks) improves on increasing l and the best compression is achieved at $l = 75$. The anomaly between $l = 25$ and $l = 50$ is due to decrease in maximum depth of

[1]As the scripts are left intact in this experiment, they could also be verified

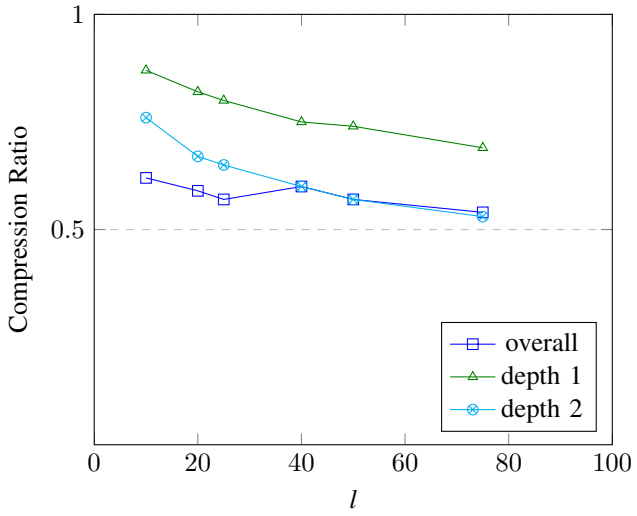

Fig. 2: Plot of compression ratio vs l for 50,000 bocks

compression, 3 and 2 respectively. Due to setup constraints, we couldn't verify beyond $l = 75$.

V. Conclusion

In this paper, we solve the problem of large blockchain size using block summarization. Our proposed solution reduces the blockchain size and makes computational activities feasible for nodes with low memory capacity or processing power and make them less dependent on peers.

Block summarization still faces a few challenges. One of them is dealing with huge block summary size. The amount of saving in block size is closely related to the fragmentation of transactions. If the transaction increases the fragmentation of the resource, the gain will decrease. A future work could be in determining an adaptive value of l so as to maintain a trade-off between having a high compression ratio in one summary block and moderating the size of the summary block. Another challenge is providing an incentive to full nodes to maintain the availability of original blocks. More the percentage of the light nodes among the nodes, less is the availability of original blocks of the blocks. The original blocks are an essential part and are required for verification of block summaries, record keeping and to resolve forks. For a new node to be able to trust the current state, it may wish to verify the integrity of the chain for which original blocks are required. The nodes with sufficient resource should continue to function as a full node and for this, they should be provided with an incentive in some form (the actual incentive will depend on the domain in which

blockchain is being applied). For private blockchain systems, this problem can be solved by maintaining some full nodes. However summarization is an independent concept, it can also be used in tandem with other blockchain size management techniques like pruning for more efficiency.

Our main reason for block summarization was to empower the light nodes by providing them with enough weight to participate in blockchain system. We wished to facilitate the interaction of different types of clients in the blockchain, which is extremely important as Blockchains have started pervading all domains of data storage and transactions like cryptocurrency, finance, healthcare, and music.

VI. Acknowledgment

This paper has been accepted for publication in Blockchains and Smart Contracts workshop (BSC'2018). The authors would like to thank Sugata Gangopadhyay & Nishant Sinha from IIT Roorkee for their valuable contribution in form of guidance, support and writing assistance.

References

[1] Bitcoin blockchain size over time, 2017. http://blockchain.info/charts/blocks-size.
[2] Bitcoin Scalability. https://en.bitcoin.it/wiki/Scalability
[3] Block file pruning. https://github.com/bitcoin/bitcoin/blob/v0.11.0/doc/release-notes.md#block-file-pruning
[4] J. D. Bruce. The mini-blockchain scheme, 2017. http://cryptonite.info/files/mbc-scheme-rev3.pdf.
[5] Cryptonite cryptonite.info/
[6] A. Gervais, G. O. Karame, D. Gruber, S. Capkun. On the privacy provisions of bloom filters in lightweight bitcoin clients. Annual Computer Security Applications Conference, ACM, 2014. https://eprint.iacr.org/2014/763.pdf.
[7] J. v. D. Hooff, M. F. Kaashoek, N. Zeldovich. Versum: Verifiable computations over large public logs. ACM SIGSAC Conference on Computer and Communications Security. ACM, 2014.
[8] O. Jacobovitz. Blockchain for Identity Management: Technical Report. http://cs.bgu.ac.il/~frankel/TechnicalReports/2016/16-02.pdf
[9] Mini-blockchain scheme. http://cryptonite.info/wiki/index.php?title=Mini-blockchain_scheme
[10] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2014. https://bitcoin.org/bitcoin.pdf.
[11] R3, https://www.r3.com/
[12] A. Tomescu and S. Devadas. Catena: Preventing Lies with Bitcoin. IACR Cryptology ePrint Archive, Report 2016/1062, 2016. https://eprint.iacr.org/2016/1062.
[13] Z. Zheng, S. Xie, H. N. Dai, H. Wang. Blockchain challenges and opportunities: A survey. International Journal of Web and Grid Services, 2016.