The Limits to Blockchain Scalability

Special thanks to Felix Lange, Martin Swende, Marius van der Wijden and Mark Tyneway for feedback and review.

Just how far can you push the scalability of a blockchain? Can you really, asElon Musk wishes, "speed up block time 10X, increase block size 10X & drop fee 100X" without leading to extreme centralization and compromising the fundamental properties that make a blockchain what it is? If not, how far can you go? What if you change the consensus algorithm? Even more importantly, what if you change the technology to introduce features such as ZK-SNARKs or sharding? A sharded blockchain can theoretically just keep adding more shards; is there such a thing as adding too many?

As it turns out, there are important and quite subtle technical factors that limit blockchain scaling, both with sharding and without. In many cases there are solutions, but even with the solutions there are limits. This post will go through what many of these issues are.

It's crucial for blockchain decentralization for regular users to be able to run a node

At 2:35 AM, you receive an emergency call from your partner on the opposite side of the world who helps run your mining pool (or it could be a staking pool). Since about 14 minutes ago, your partner tells you, your pool and a few others split off from the chain which still carries 79% of the network. According to your node, the majority chain's blocks are invalid. There's a balance error: the key block appeared to erroneously assign 4.5 million extra coins to an unknown address.

An hour later, you're in a telegram chat with the other two small pools who were caught blindsided just as you were, as well as some block explorers and exchanges. You finally see someone paste a link to a tweet, containing a published message. "Announcing new on-chain sustainable protocol development fund", the tweet begins.

By the morning, arguments on Twitter, and on the one community forum that was not censoring the discussion, discussions are everywhere. But by then a significant part of the 4.5 million coins had been converted on-chain to other assets, and billions of dollars of defi transactions had taken place. 79% of the consensus nodes, and all the major block explorers and endpoints for light wallets, were following this new chain. Perhaps the new dev fund will fund some development, or perhaps it will just all be embezzled by the leading pools and exchanges and their cronies. But regardless of how it turns out, the fund is for all intents and purposes a fait accompli, and regular users have no way to fight back.

Can this happen on your blockchain? The elites of your blockchain community, including pools, block explorers and hosted nodes, are probably quite well-coordinated; quite likely they're all in the same telegram channels and wechat groups. If they really want

to organize a sudden change to the protocol rules to further their own interests, then they probably can. The Ethereum blockchain has fully resolved consensus failures in ten hours; if your blockchain has only one client implementation, and you only need to deploy a code change to a few dozen nodes, coordinating a change to client code can be done much faster. The only reliable way to make this kind of coordinated social attack not

effective is through passive defense from the one constituency that actually is decentralized: the users

.

Imagine how the story would have played out if the users were running nodes that verify the chain (whether directly or through more advanced indirect techniques), and automatically reject blocks that break the protocol rules even if

over 90% of the miners or stakers support those blocks. If every

user ran a verifying node, then the attack would have quickly failed: a few mining pools and exchanges would have forked off and looked quite foolish in the process. But even if some

users ran verifying nodes, the attack would not have led to a clean victory

for the attacker; rather, it would have led to chaos

, with different users seeing different views of the chain. At the very least, the ensuing market panic and likely persistent chain split would greatly reduce the attackers' profits. The thought of navigating such a protracted conflict would itself deter most attacks.

If you have a community of 37 node runners and 80000 passive listeners that check signatures and block headers, the attacker wins. If you have a community where everyone runs a node, the attacker loses. We don't know what the exact threshold is at which herd immunity against coordinated attacks kicks in, but there is one thing that's absolutely clear: more nodes good, fewer nodes bad, and we definitely need more than a few dozen or few hundred.

So, what are the limits to how much work we can require full nodes to do?

To maximize the number of users who can run a node, we'll focus on regular consumer hardware. There are some increases to capacity that can be achieved by demanding some specialized hardware purchases that are easy to obtain (eg. from Amazon), but they actually don't increase scalability by that much.

There are three key limitations to a full node's ability to process a large number of transactions:

- Computing power

: what % of the CPU can we safely demand to run a node?

- Bandwidth

: given the realities of current internet connections, how many bytes

can a block contain?

- Storage

: how many gigabytes on disk can we require users to store? Also, how quickly must it be readable? (ie. is HDD okay or do we need SSD)

Many erroneous takes on how far a blockchain can scale using "simple" techniques stem from overly optimistic estimates for each of these numbers

. We can go through these three factors one by one:

Computing power

- Bad answer

: 100% of CPU power can be spent on block verification

- Correct answer

: ~5-10% of CPU power can be spent on block verification

There are four key reasons why the limit is so low:

- We need a safety margin to cover the possibility of DoS attacks (transactions crafted by an attacker to take advantage of weaknesses in code to take longer to process than regular transactions)
- Nodes need to be able to sync the chain after being offline. If I drop off the network for a minute, I should be able to catch up in a few seconds
- Running a node should not drain your battery very quickly and make all your other apps very slow
- There are other non-block-production tasks that nodes need to do as well, mostly around verifying and responding to incoming transactions and requests on the p2p network

Note that up until recently, most explanations for "why only 5-10%?" focused on a different problem: that because PoW

blocks come at random times, it taking a long time to verify blocks increases the risk that multiple blocks get created at the same time. There are many fixes to this problem (eg. Bitcoin NG, or just using proof of stake). But these fixes do NOT solve the other four problems, and so they don't enable large gains in scalability as many had initially thought.

Parallelism is also not a magic bullet. Often, even clients of seemingly single-threaded blockchains are parallelized already: signatures can be verified by one thread while execution is done by other threads, and there's a separate thread that's handling transaction pool logic in the background. And the closer you get to 100% usage across all

threads, the more energy-draining running a node becomes and the lower your safety margin against DoS.

Bandwidth

- Bad answer

: if we have 10 MB blocks every 2-3 seconds, then most users have a >10 MB/sec network, so of course they can handle it

- Correct answer

: maybe

we can handle 1-5 MB blocks every 12 seconds. It's hard though.

Nowadays we frequently hear very high advertised statistics for how much bandwidth internet connections can offer: numbers of 100 Mbps and even 1 Gbps are common to hear. However, there is a large difference between advertised bandwidth and the expected actual bandwidth of a connection for several reasons:

1. "Mbps" refers to "millions of bits

per second"; a bit is 1/8 of a byte, so you need to divide advertised bit numbers by 8 to get the advertised byte numbers.

1. Internet providers, just like all companies, often lie.
2. There's always multiple applications using the same internet connection, so a node can't hog the entire bandwidth.
3. p2p networks inevitably introduce their own overhead: nodes often end up downloading and re-uploading the same block multiple times (not to mention transactions being broadcasted through the mempool before

being included in a block).

When Starkware did an experiment in 2019 where they published 500 kB blocks after the transaction data gas cost decrease made that possible for the first time, a few nodes were actually unable to handle blocks of that size. Ability to handle large blocks has since been improved and will continue to be improved. But no matter what we do, we'll still be very

far from being able to naively take the average bandwidth in MB/sec, convince ourselves that we're okay with 1s latency, and be able to have blocks that are that size.

Storage

- Bad answer

: 10 terabytes

- Correct answer

: 512 gigabytes

The main argument here is, as you might guess, the same as elsewhere: the difference between theory and practice. In theory

, there are 8 TB solid state drives that you can buy on Amazon (you do

need SSDs or NVME; HDDs are too slow for storing the blockchain state). In practice

, the laptop that was used to write this blog post has 512 GB, and if you make people go buy their own hardware, many of them will just get lazy (or they can't afford $800 for an 8 TB SSD) and use a centralized provider. And even if you can fit a blockchain onto some storage, a high level of activity can easily quickly burn through the disk and force you to keep getting a new one.

Additionally, storage size determines the time needed for a new node to be able to come online and start participating in the network. Any data that existing nodes have to store is data that a new node has to download. This initial sync time (and bandwidth) is also a major barrier to users being able to run nodes. While writing this blog post, syncing a new geth node took me ~15 hours. If Ethereum had 10x more usage, syncing a new geth node would take at least a week, and it would be much more likely to just lead to your internet connection getting throttled. This is all even more important during an attack, when a successful response to the attack will likely involve many users spinning up new nodes when they were not running nodes before.

Interaction effects

Additionally, there are interaction effects between these three types of costs. Because databases use tree structures internally to store and retrieve data, the cost of fetching data from a database increases with the logarithm of the size of the database. In fact, because the top level (or top few levels) can be cached in RAM, the disk access cost is proportional to the size of the database as a multiple of the size of the data cached in RAM

.

For example, if the cache is 4 GB, and we assume that each layer of the database is 4x bigger than the previous, then Ethereum's current ~64 GB state would require ~2 accesses. But if the state size increases by 4x to ~256 GB, then this would increase to ~3 accesses (so 1.5x more accesses per read). Hence, a 4x increase in the gas limit, which would increase both the state size and the number of reads, could actually translate into a ~6x increase in block verification time. The effect may be even stronger: hard disks often take longer to read and write when they are full than when they are near-empty.

So what does this mean for Ethereum?

Today in the Ethereum blockchain, running a node already is challenging for many users, though it is still at least possible

on regular hardware (I just synced a node on my laptop while writing this post!). Hence, we are close to hitting bottlenecks. The issue that core developers are most concerned with is storage size.

Thus, at present, valiant efforts at solving bottlenecks in computation and data, and even changes to the consensus algorithm, are unlikely to lead to large gas limit increases being accepted. Even solving Ethereum's largest outstanding DoS vulnerability only led to a gas limit increase of 20%.

The only solution to storage size problems isstatelessness and state expiry. Statelessness

allows for a class of nodes that verify the chain without

maintaining permanent storage. State expiry

pushes out state that has not been recently accessed, forcing users to manually provide proofs to renew it. Both of these paths have been worked at for a long time, and proof-of-concept implementation on statelessness has already started. These two improvements combined can greatly alleviate these concerns and open up room for a significant gas limit increase. But even after statelessness and state expiry are implemented, gas limits may only increase safely by perhaps ~3x until the other limitations start to dominate.

Another possible medium-term solution is using ZK-SNARKs to verify transactions. ZK-SNARKs would ensure that regular users do not have to personally store the state or

verify blocks, though they still would need to download all the data in blocks to protect againstdata unavailability attacks. Additionally, even if attackers cannot force invalid

blocks through, if capacity is increased to the point where running a consensus node is too

difficult, there is still the risk of coordinated censorship attacks. Hence, ZK-SNARKs cannot increase capacity infinitely, but they still can increase capacity by a significant margin (perhaps 1-2 orders of magnitude). Some chains are exploring this approach at layer 1; Ethereum is getting the benefits of this approach through layer-2 protocols (called ZK rollups) such as zksync, Loopring and Starknet.

What happens after sharding?

Sharding fundamentally gets around the above limitations, because it decouples the data contained on a blockchain from the data that a single node needs to process and store.

Instead of nodes verifying blocks by personally downloading and executing them, they useadvanced mathematical and cryptographic techniques to verify blocks indirectly.

As a result, sharded blockchains can safely have very high levels of transaction throughput that non-sharded blockchains cannot. This does require a lot of cryptographic cleverness in creating efficient substitutes for naive full validation that successfully reject invalid blocks, but it can be done: the theory is well-established and proof-of-concepts based on draft specifications are already being worked on.

Ethereum is planning to use quadratic sharding

, where total scalability is limited by the fact that a node has to be able to process both a single shard and the beacon chain which has to perform some fixed amount of management work for each shard. If shards are too big, nodes can no longer process individual shards, and if there are too many shards, nodes can no longer process the beacon chain. The product of these two constraints forms the upper bound.

Conceivably, one could go further by doing cubic sharding, or even exponential sharding. Data availability sampling would certainly become much more complex in such a design, but it can be done. But Ethereum is not going further than quadratic. The reason is that the extra scalability gains that you get by going from shards-of-transactions to shards-of-shards-of-transactions actually cannot be realized without other risks becoming unacceptably high.

So what are these risks?

Minimum user count

A non-sharded blockchain can conceivably run as long as there is even one user that cares to participate in it. Sharded blockchains are not like this: no single node can process the whole chain, and so you need enough nodes so that they can at least process the chain together

. If each node can process 50 TPS, and the chain can process 10000 TPS, then the chain needs at least 200 nodes to survive. If the chain at any point gets to less than 200 nodes, then either nodes stop being able to keep up with the chain, or nodes stop being able to detect invalid blocks, or a number of other bad things may happen, depending on how the node software is set up.

In practice, the safe minimum count is several times higher than the naive "chain TPS divided by node TPS" heuristic due to the need for redundancy (including for data availability sampling); for our above example, let's call it 1000 nodes.

If a sharded blockchain's capacity increases by 10x, the minimum user count also increases by 10x. Now, you might ask: why don't we start with a little bit of capacity, and increase it only when we see lots of users so we actually need it, and decrease it if the user count goes back down?

There are a few problems with this:

1. A blockchain itself cannot reliably detect how many unique users are on it, and so this would require some kind of governance to detect and set the shard count. Governance over capacity limits can easily become a locus of division and conflict.
2. What if many users suddenly and unexpectedly drop out at the same time?
3. Increasing the minimum number of users needed for a fork to start makes it harder to defend against hostile

takeovers.

A minimum user count of under 1,000 is almost certainly fine. A minimum user count of 1 million, on the other hand, is certainly not. Even a minimum user count of 10,000 is arguably starting to get risky. Hence, it seems difficult to justify a sharded blockchain having more than a few hundred shards

.

History retrievability

An important property of a blockchain that users really value is permanence

. A digital asset stored on a server will stop existing in 10 years when the company goes bankrupt or loses interest in maintaining that ecosystem. An NFT on Ethereum, on the other hand, is forever

.

But once a blockchain's capacity gets too high, it becomes harder to store all that data, until at some point there's a large risk that some part of history will just end up being stored by... nobody.

Quantifying this risk is easy. Take the blockchain's data capacity in MB/sec, and multiply by ~30 to get the amount of data stored in terabytes per year. The current sharding plan has a data capacity of ~1.3 MB/sec, so about 40 TB/year. If that is increased by 10x, this becomes 400 TB/year. If we want the data to be not just accessible, but accessible conveniently

, we would also need metadata (eg. decompressing rollup transactions), so make that 4 petabytes per year, or 40 petabytes after a decade. The Internet Archive uses 50 petabytes. So that's a reasonable upper bound for how large a sharded blockchain can safely get.

Hence, it looks like on both of these dimensions, the Ethereum sharding design is actually already roughly targeted fairly close to reasonable maximum safe values

. The constants can be increased a little bit

, but not too much.

Summary

There are two ways to try to scale a blockchain: fundamental technical improvements

, and simply increasing the parameters

. Increasing the parameters sounds very attractive at first: if you do the math on a napkin, it is easy to convince yourself that a consumer laptop can process thousands of transactions per second, no ZK-SNARKs or rollups or sharding required. Unfortunately, there are many subtle reasons why this approach is fundamentally flawed.

Computers running blockchain nodes cannot spend 100% of CPU power validating the chain; they need a large safety margin to resist unexpected DoS attacks, they need spare capacity for tasks like processing transactions in the mempool, and you don't want running a node on a computer to make that computer unusable for any

other applications at the same time. Bandwidth similarly has overhead: a 10 MB/s connection does NOT mean you can have a 10 megabyte block every second! A 1-5 megabyte block every 12 seconds, maybe

. And it is the same with storage. Increasing hardware requirements for running a node and limiting node-running to specialized actors is not

a solution. For a blockchain to be decentralized, it's crucially important for regular users to be able to run a node, and to have a culture where running nodes is a common activity

.

Fundamental technical improvements, on the other hand, can

work. Currently, the main bottleneck in Ethereum is storage size, and statelessness and state expiry can fix this

and allow an increase of perhaps up to ~3x - but not more, as we want running a node to become easier

than it is today. Sharded blockchains can scale much further

, because no single node in a sharded blockchain needs to process every

transaction. But even there, there are limits to capacity: as capacity goes up, the minimum safe user count goes up, and the cost of archiving the chain (and the risk that data is lost if no one bothers to archive the chain) goes up. But we don't have to worry too much: those limits are high enough that we can probably process over a million transactions per second with the full security of a blockchain. But it's going to take work to do this without sacrificing the decentralization that makes blockchains so valuable.