Scaling DLT's

A short introduction for the *Blockchain Technology* Seminar SS22

7/6/2022

Outline

- **Motivation**
- **Approaches**
- **Layer-1 vs. Layer-2**
- ⚡ **Lightning Network**
  - Bitcoin Scaling Problem
  - tl;dr
  - Importance for Bitcoin
- **Scaling Ethereum**
  - Primer - short introduction/recap
  - Sharding - ETH's L1 approach to scaling
  - Rollups - sharding isn't enough
- ❄ **Avalanche** - a different scaling approach based on consensus
- **Enter the rabbit hole** 🕳 🐇

Motivation

- only adjacent edges of one side can be fulfilled
- solving the trilemma could lead to new levels of adoption
- multiple approaches to solve it or at least improve the current state

read more

Approaches

Any ideas?

- increasing block size
- nesting blockchains
- increase consensus efficiency
- state channels
- sidechains
- bundeling / rollups
- parallelization / sharding

Layer-1

- increasing block size
- increase consensus efficiency
- parallelization / sharding

Layer-2

- nesting blockchains
- state channels
- sidechains
- bundeling / rollups

Layer-1

- main DLT network protocol itself
- scaling solutions are changes to code or network structure itself
- e.g. Bitcoin, Ethereum, Avalanche,...

Layer-2

- protocols sitting on top of the L1 to increase scalability or add functionality
- it comes down to **offloading effort from the chain**
- e.g. Optimism, Boba, Loopring...

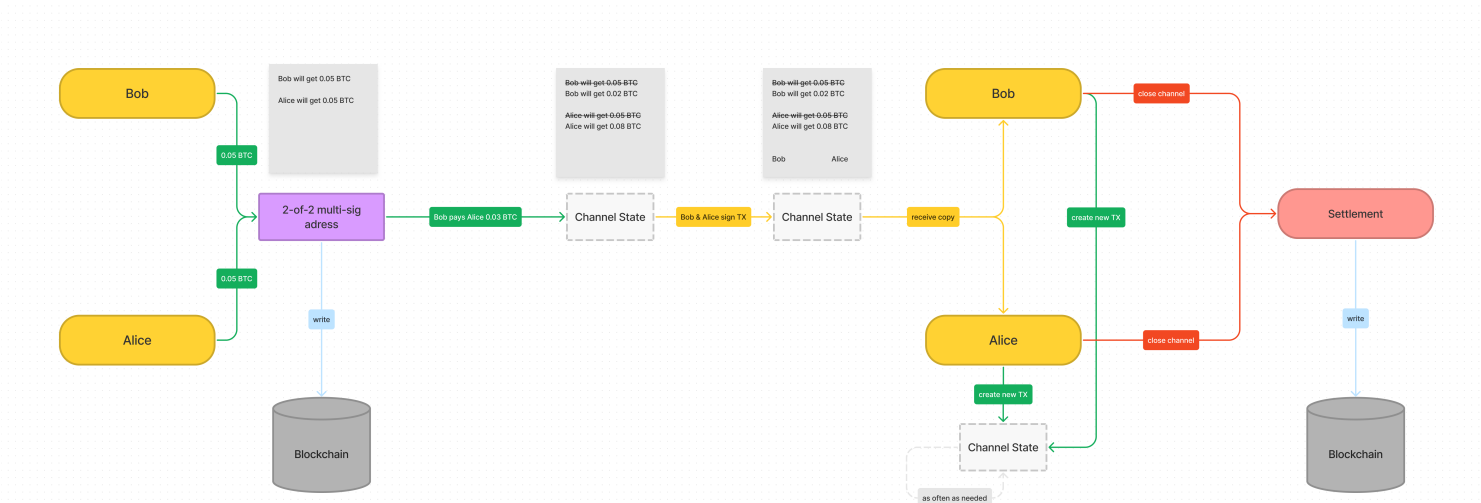
Scaling Bitcoin

Scaling Problem

- huge impact on the network if every node must know about every single transaction
- Visa's 47,000 peak tps vs. < 7 tps
- 1MB block limit
- increasing block size?
- principal-agent problem

Deal with transactions off the Bitcoin blockchain itself!

⚡ tl;dr



- opening and closing of payment channel is written on blockchain
- timelocks (HTLC's) are used to ensure communication in the channels
- enables fast and cheap Bitcoin payments down to the satoshi

Is the Lightning Network a L1 or L2 scaling solution?🤔

L2

Implications for Bitcoin

The Lightning Network enables the following functionality for Bitcoin

- Instant Transactions
- Exchange Arbitrage
- Micropayments
- Financial Smart Contracts
- Cross-Chain Payments

Scaling does not "just" improve the speed of a network!

Scaling Ethereum

> "Ethereum is a technology that lets you send cryptocurrency to anyone for a small fee. It also powers applications that everyone can use and no one can take down."

It’s the world’s programmable blockchain.

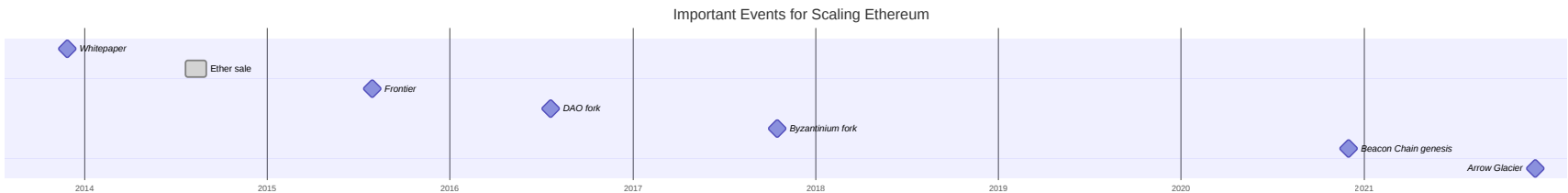
Primer

- public & permissionless
- decentralized
- no private transactions
- consensus mechanism: **Proof-Of-Work**
- enables **Smart Contracts** using
  - Solidity
  - Vyper
- used for
  - ETH (native asset)
  - DeFi & Dapps
  - NFT
  - Decentralized autonomous organizations

Primer

- inspired by Bitcoins *"underlying blockchain technology as a tool of distributed consensus"*
- the Ethereum virtual machine (EVM) is turing-complete
- inherently designed to force implementation of scaling solutions by the **"difficulty bomb"**
  - transition to POS
  - reduce chances of fork

History

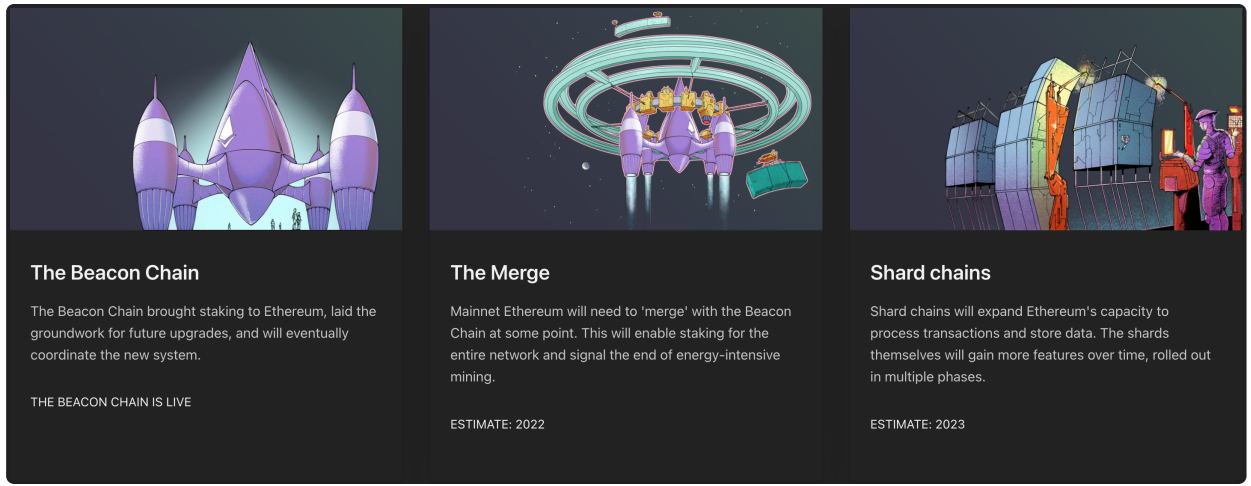


Summary

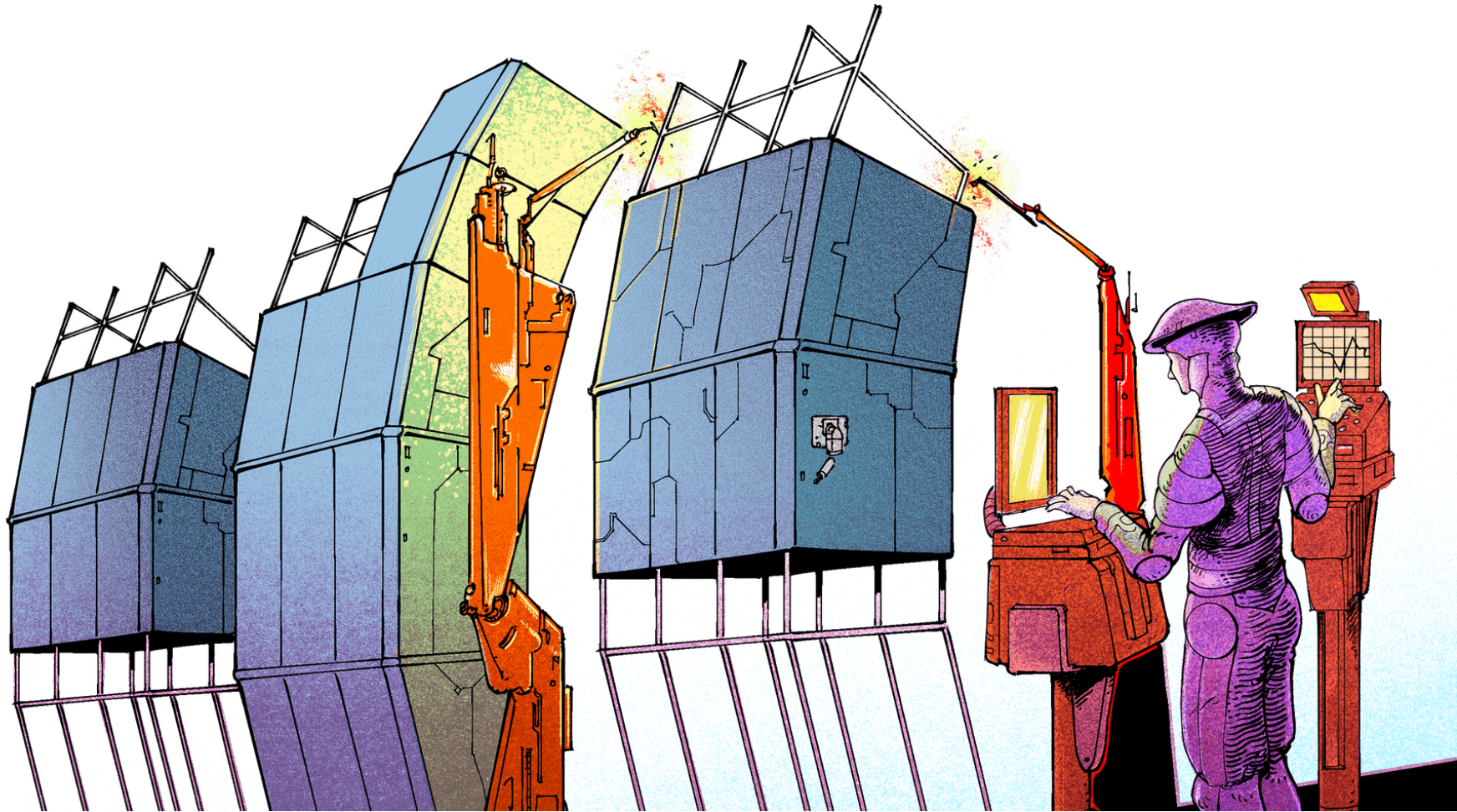
- turing-complete EVM enables programmable money
- scaling is forced by design
- after multiple pushbacks the first step of the scaling vision went live (*Beacon Chain*)

learn more about the Ethereum History

🔭 The Ethereum Vision



- Beacon Chain coordinates shards & stakers
- The Merge will connect the Ethereum mainnet with the Beacon Chain POS system
- Shard Chains will complete current scaling efforts
  - multi-phase upgrade
  - enable Layer-2 solutions to offer low fees with the security of Ethereum
- main goals: **Scalability**, Security & Sustainability



Sharding

- related to horizontal data partition
- shared-nothing architecture
  - shards are autonomous
  - memory & storage is not shared between nodes
- adds complexity & potential failure points ⚠

When to shard ? 🧐

- application data outgrows capacity of single database node
- slowed response times due to volume of read/writes to single node
- network bandwidth required is greater than the bandwidth available to a single node

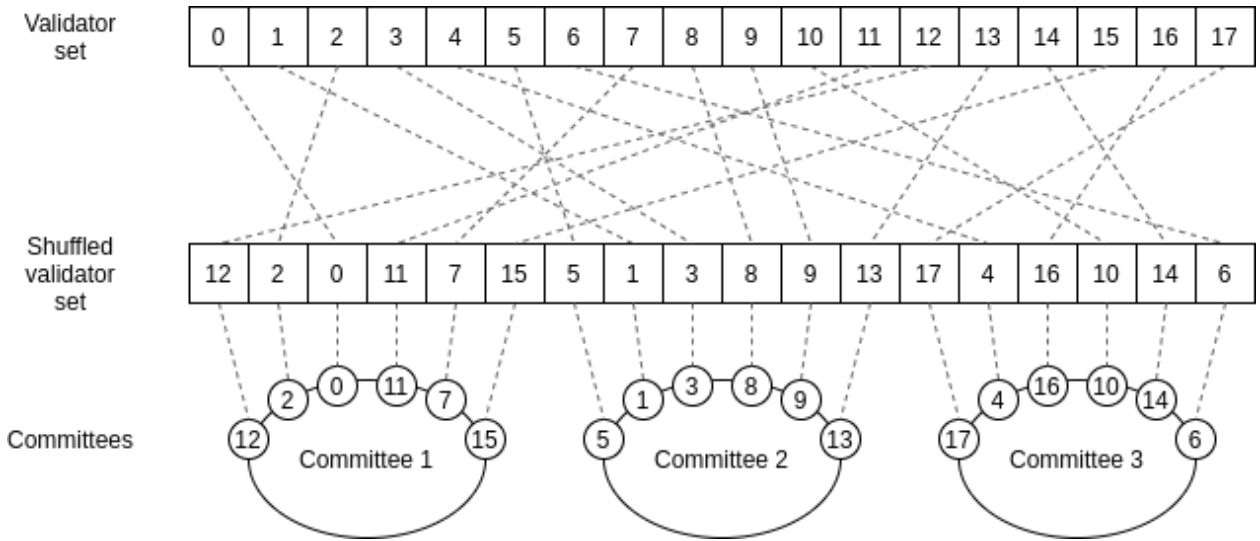
commonly used:

Implementations [edit]

- Altibase provides combined (client-side and server-side) sharding architecture transparent to client applications.
- Apache HBase can shard automatically.[4]
- Azure SQL Database Elastic Database tools shards to scale out and in the data-tier of an application.[5]
- ClickHouse, a fast open-source OLAP database management system, shards.
- Couchbase shards automatically and transparently.
- CUBRID shards since version 9.0
- Db2 Data Partitioning Feature (MPP) which is a shared-nothing database partitions running on separate nodes.
- DRDS (Distributed Relational Database Service) of Alibaba Cloud does database/table sharding,[6] and supports Singles' Day.[7]
- Elasticsearch enterprise search server shards.[8]
- eXtreme Scale is a cross-process in-memory key/value data store (a NoSQL data store). It uses sharding to achieve scalability across processes for both data and MapReduce-style parallel processing.[9]
- Hibernate shards, but has had little development since 2007.[10][11]
- IBM Informix shards since version 12.1 xC1 as part of the MACH11 technology. Informix 12.10 xC2 added full compatibility with MongoDB drivers, allowing the mix of regular relational tables with NoSQL collections, while still allowing sharding, fail-over and ACID properties.[12][13]
- Kdb+ shards since version 2.0.
- MonetDB, an open-source column-store, does read-only sharding in its July 2015 release.[14]
- MongoDB shards since version 1.6.
- MySQL Cluster automatically and transparently shards across low-cost commodity nodes, allowing scale-out of read and write queries, without requiring changes to the application.[15]
- MySQL Fabric (part of MySQL utilities) shards.[16]
- Oracle Database shards since 12c Release 2 and in one liner: Combination of sharding advantages with well-known capabilities of enterprise ready multi-model Oracle Database.[17]
- Oracle NoSQL Database has automatic sharding and elastic, online expansion of the cluster (adding more shards).
- OrientDB shards since version 1.7
- Solr enterprise search server shards.[18]
- Spanner, Google's global-scale distributed database, shards across multiple Paxos state machines to scale to "millions of machines across hundreds of data centers and trillions of database rows".[19]
- SQLAlchemy ORM, a data-mapper for the Python programming language shards.[20]
- SQL Server, since SQL Server 2005 shards with help of 3rd party tools.[21]
- Teradata markets a massive parallel database management system as a "data warehouse"
- Vault, a cryptocurrency, shards to drastically reduce the data that users need to join the network and verify transactions. This allows the network to scale much more.[22]
- Vitess open-source database clustering system shards MySQL. It is a Cloud Native Computing Foundation project.[23]
- ShardingSphere related to a database clustering system providing data sharding, distributed transactions, and distributed database management . It is a Apache Software Foundation (ASF) project.[24]

Source

Sharding through Random Sampling



- randomly split verification work
- shuffle validator list and assign **committees** of size n to verify a block
- each validator publishes a signature upon block validation

Sharding Consensus

Important Definitions

- beacon chain
- **slot:** 12 second time-frame in which a block is expected to be added to the chain
- **epoch:** comprised of 32 slots
- **attestation** consisting of
  - vote for current beacon chain head
  - vote on which beacon block should be justified/finalized
  - vote on the current state of the shard chain
  - signatures of all validators agreeing with that vote

How can we use the properties of attestations within a committee in a clever way?🤔

Sharding Consensus

- shard & beacon chain votes should be identical across validators within the same committee
- if we just had a way to combine all signatures... 🤷‍♂️
  - short-midterm: BLS signatures enable us to store & check signatures `A, B` by only storing `C = A + B`
  - long-term: zkSNARKs

Only 1 signature is stored for a **whole committee**

What about bad actors? 🦹

- general assumption: committees represent the overall validator set (more or less 😄)
- we try to prevent malicious validators from ending up in the same committee by...
  - ensuring random committee assignments
  - requiring a minimum number of validators per committee
- it is quite unlikely for a bad actor to gain control over a committee
  - e.g. using 128 randomly sampled validator per committee leads to the probability of an attacker with $\frac{1}{3}$ of all validators getting a $> \frac{2}{3}$ committee of less than 2^{-40} src

Sharding Roadmap

Ethereum 2.0

- **Phase 0:** POS beacon chain without shards (shipped in 2020 🚀)
  - beacon chain contains logic to secure and sync shards
  - also coordinates stakers in the network; assignment of shards
- **Phase 1:** Basic sharding without EVM (planned for 2023 📅)
- *Phase 2: EVM state transition function*
- *Phase 3: Light client state protocol*
- *Phase 4: Cross-shard transactions*
- *Phase 5: Tight coupling with main chain security*
- *Phase 6: Super-quadratic or exponential sharding* **Recursively, shards within shards within shards...**
- **Ethereum 3.0**

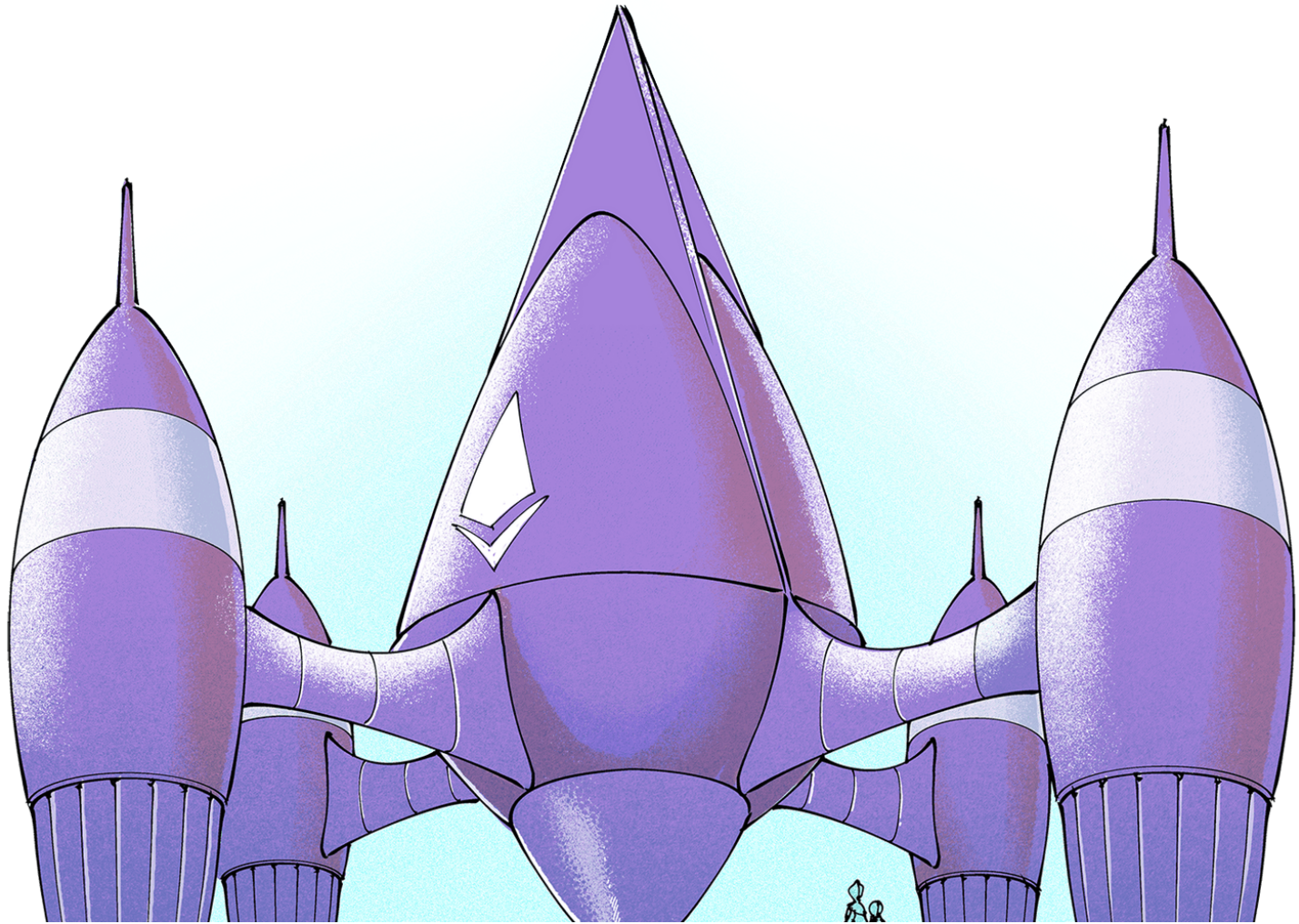
learn more

Quick Recap

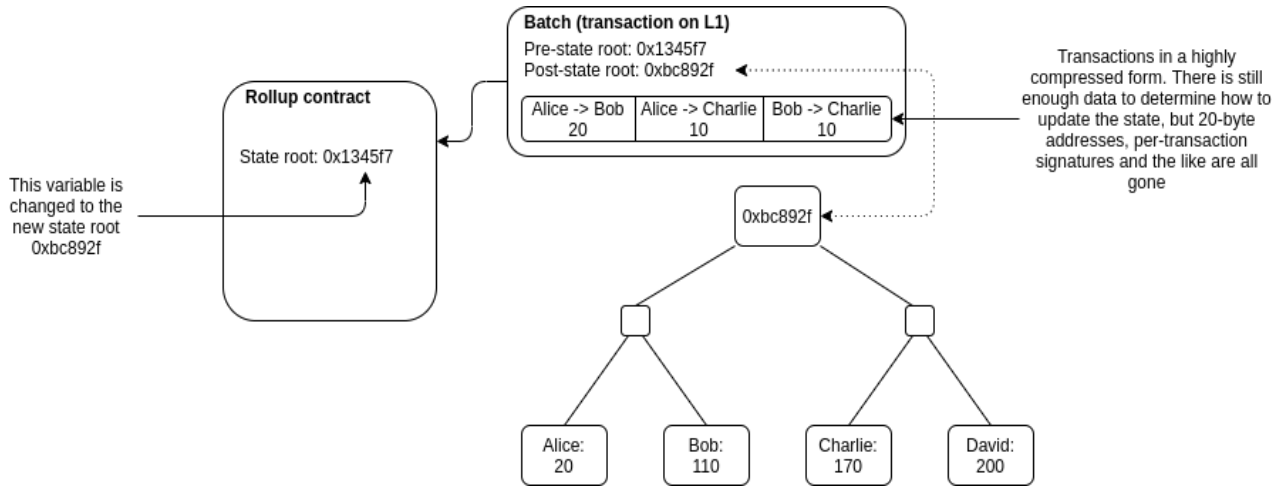
i.g.: splitting a database horizontally

- part of multi-phase upgrade to improve scalability and capacity
- for now shard chains provide *just* storage layers - *you will learn how to use that in a clever way in a second* 😉
- committees of random validators approve blocks
- validation results of several committees are combined in a way that only requires checking the signatures of all validators
  - attestation aggregation
  - BLS signatures / zkSNARKs
- **not only** a matter of **scaling** ⇢ improves security & decentralization of the network

Any questions or comments?



Rollups


Batch (transaction on L1)
Pre-state root: 0x1345f7
Post-state root: 0xbc892f
Alice -> Bob 20
Alice -> Charlie 10
Bob -> Charlie 10
Transactions in a highly compressed form. There is still enough data to determine how to update the state, but 20-byte addresses, per-transaction signatures and the like are all gone
Rollup contract
State root: 0x1345f7
This variable is changed to the new state root 0xbc892f
0xbc892f
Alice: 20
Bob: 110
Charlie: 170
David: 200

Rollups

perform transactions on Layer-2 and post data to Layer-1 once settled.

How do they work?

- on-chain smart contract maintains **state root**: *merkle root* of the rollup state
- anyone can publish a transaction **batch**
  - transactions highly compressed
  - ***previous*** state root
  - ***next*** state root

There are two distinctly different types of rollups:

- optimistic ✨
- zero-knowledge ⛓

✨ Optimistic Rollups

- sequencer certifies that computing `C` with input `X` leads to output `Y` ⇢ *"the transactions I validated are correct"*
  - provides a bond
- **any** other party (**challenger**) can submit a **fraud proof** during a *dispute period* 🚨
  - a bond also needs to be provided
  - gas is paid for with bond of the other party when correct
- suspicious transaction is executed by all nodes on the Ethereum chain 👮
- the loosing party's bond is slashed
- if no one submits a **fraud proof** transactions are considered final after the *dispute period* is over

🤔 Why does the bond get slashed?

instead of paying out the full amount to the challenger...

⛓️ zk Rollups

includes an easily verifiable cryptographic **valididty** proof in every *batch*.

- enable a party to prove knowledge about something while having **z**ero **k**nowledge about "the something"
- rollups only inlcude validity proof
- faster finality
- currently there are two approaches to zk proofs

⛓ zk Rollups

Scalable **T**ransparent **A**rgument of **K**nowledge ⇢ **STARKs**

- rely on hash functions ⇢ quantum resistant
- don’t require trusted set-up
- larger proof size requires
  - more time to verify
  - higher gas fees (up to 24%)

Succinct **N**on-interactive **AR**gument of **K**nowledge ⇢ **SNARKs**

- based on elliptic curves
- currently the main approach of Ethereum using the PLONK protocol
- better developer support among other benefits

read more about the differences & advantages

🥜 zkSNARKs in a nutshell

- **S**uccinct
  - message sizes are really small in comparison to the actual computation
- **N**on-interactive
  - setup phase and single message from prover to verifier
  - anyone can verify without interacting anew
- **AR**guments
  - protection only against computationally limited provers
  - computational soundness vs. perfect soundness
- **K**nowledge
  - impossible for prover to construct a proof w/o a witness
  - e.g. address, path to Merkle-tree node,...

🥜 zkSNARKs in a nutshell

four main ingredients:

- encoding as polynomial problem
- succinctness by random sampling
- homomorphic encryption
- zero knowledge

🥜 zkSNARKs in a nutshell

$$t(x)h(x) = w(x)v(x)$$

encoding as polynomial problem

- program/data is compiled into quadratic equation of polynomials
- equality only holds if the program is computed correctly
- a **prover** wants to convince the **verifier** that said equality holds

🥜 zkSNARKs in a nutshell

$$t(s)h(s) = w(s)v(s)$$

succinctness by random sampling

- verifier chooses a/multiple evaluation point s
- therefore simplifying the problem fromm polynomial multiplication to
  - simple multiplication
  - equality check on numbers

🥜 zkSNARKs in a nutshell

$$E$$

homomorphic encryption

- quasi homomorphic function E is selected
- allows the **prover** to compute $E(t(s)), \ldots$ withouth knowing s
- only $E(s)$ and some other encrypted values are known

🥜 zkSNARKs in a nutshell

$$t(s)h(s)k = w(s)v(s)k$$

zero knowledge

- **prover** permutes the encrypted values by multiplication with a number k
- **verifier** can check if their *stucture* is correct even ***withouth knowing the encoded values***

Handwavy Idea 👋🏻

- checking $t(s)h(s) = w(s)v(s)$ is identical to checking $t(s)h(s)k = w(s)v(s)k$
- with one important ***difference***:
  - checking $t(s)h(s)k = w(s)v(s)k$ only requires $t(s)h(s)k$ and $w(s)v(s)k$
  - it's impossible to derive $t(s)h(s)$ or $w(s)v(s)$

🔏 A simple zk-SNARK example

A zk-SNARK consists of three algorithms

- key `G`enerator
  - takes secrect parameter `lambda` and program `C`
  - returns publicly available *proving key* `pk` and *verification key* `vk`
- `P`rover
  - takes `pk`, public input `x` and private witness `w`
  - returns a *proof* `prf=P(pk, x, w)`
- `V`erifier
  - takes `vk`, `x` and `prf`
  - `V(vk, x, prf)` returns `true` if proof is correct; `false` otherwise

`V` returns `true` if prover knows a witness satisfying `C(x, w) == true`

🔏 A simple zk-SNARK example

premise: *Alice and Bob use a zkSNARK, since Alice wants to proof her knowledge about a secret value.*

```
function C(x, w) {
    return ( sha256(w) == x );
}

(pk, vk) = G(C, lambda);
prf = P(pk, H, w);
result = V(vk, H, prf) ? 'Alice is a lier 🚫' : 'Alice knows the secret 🔓';

// 'Alice knows the secret 🔓'
```

👨‍💻 Quick Recap

- based on elliptic curves
- enables **verifiers** to check knowledge of **prover** without the need to reveal the secret
  - $t(x)h(x) = w(x)v(x)$
  - $t(s)h(s) = w(s)v(s)$
  - $E(t(s)), \ldots$
  - checking $t(s)h(s) = w(s)v(s)$ is identical to checking $t(s)h(s)k = w(s)v(s)k$
- currently on the roadmap to be utilized by Ethereum
- there are alternatives and plenty of different implementations (STARKs, PLONK,...)

Rollups: ✨ Optimistic vs. zk ⛓️

Property	Optimistic Rollups	zk Rollups
Gas cost per batch	**~40,000**	~500,000
Withdrawal period	~1 week	next batch ⇢ **very fast**
Complexity	**low**	high
Generalizability	**easier**	harder
Per-transaction gas cost on chain	higher	**lower**
Off-chain computation cost	**lower**	higher

Source

Phase 1 Ethereum Scaling

according to the rollup-centric ethereum roadmap

- current Ethereum has ~ 15 TPS
- moving to rollups can yield ~ 3000 TPS
- utilizing sharded chains as rollup data storage will allow a theoretical max of **~ 100000 TPS**
- *later phases will improve TPS even more...*

Questions ?

Enter the rabbit hole 

- DeFi Developer Roadmap
- DLTs
- Blockchain Scaling Solutions
- Computerphile Video on Blockchain Scaling
- Lightning Network

Ethereum

- Ethereum Whitepaper
- How does Ethereum work, anyway?
- Designing Ethereum | Vitalik Buterin
- Ethereum History
- Vitalik on 10.000x scaling Ethereum
- Ethereum Upgrades
- Ethereum Docs | Scaling
- Scaling Potential

Enter the rabbit hole 

Rollups

- Rollup Centric Roadmap
- Understanding PLONK | Vitalik Buterin
- lambda design ritual
- zkSync
- Cairo
- Hardware Acceleration for zk Proofs

zkSNARKS

- Mathematics behind zkSNARKs
- ConsenSys Workshop
- Rollups on data sharded ETH2
- Why is the combination so powerful?

Thank you for your attention!

The presentation is available at `https://calwritescode.github.io/scaling-dlts-2022/`

Illustrations

- Ethereum Illustrations
- Avalanche Illustrations