Stellar
Development
Foundation

Digital Currency

How Architecture Enables Innovation

David Mazières

Chief scientist & Co-founder, Stellar Development Foundation
(Faculty co-director, Stanford Future of Digital Currency Initiative)

Thursday, September 7, 2023

Introduction

Public blockchains unleashed a massive wave of fintech innovation since 2008

- A lot of misunderstanding, overclaiming, unhelpful religious fervor, and even fraud

What does this new technology really give us?

How to solve some challenges of leveraging blockchain at scale?

What properties should we strive for in next-gen financial infrastructure?

What blockchain is not

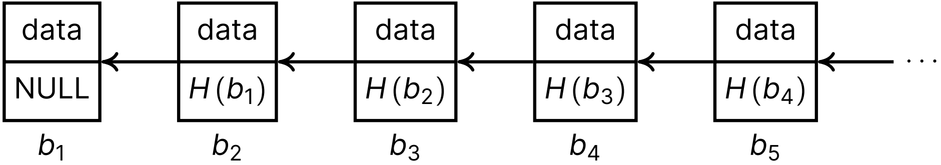
Myths of blockchain vs. traditional financial infrastructure

- Blockchains are cheaper, more scalable, more private
- Blockchains more secure and reliable
- Blockchains can solve inflation

Other common misconceptions

- Blockchains are just for cryptocurrency
- Blockchains are good for money laundering

What is a blockchain?



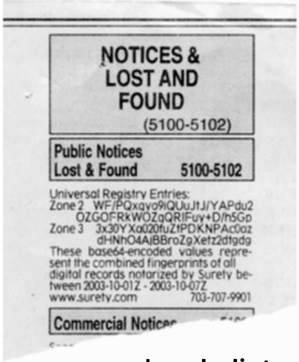
A data structure representing an append-only list of data blocks

- Each block contains a collision-resistant cryptographic hash (H) of the previous block
- If you agree on contents of a block, you agree on all prior blocks

Fist blockchain, Surety, used NYT classified ads to timestamp documents (1995)

So what’s the big deal with public blockchains?

What is a blockchain?



NOTICES & LOST AND FOUND (5100-5102)

Public Notices
Lost & Found 5100-5102

Universal Registry Entries:
Zone 2 WF/PQxqvo9iQUuJtJ/YAPdu2
OZGOFRkWOZqQRIFuy+D/h5Gp
Zone 3 3x30YXa020fuZtPDKNPAc0az
dHNhO4AjBBroZgXetz2dtgdg
These base64-encoded values represent the combined fingerprints of all digital records notarized by Surety between 2003-10-01Z - 2003-10-07Z
www.surety.com 703-707-9901

Commercial Notice

A data structure representing an append-only list of data blocks

- Each block contains a collision-resistant cryptographic hash (*H*) of the previous block
- If you agree on contents of a block, you agree on all prior blocks

Fist blockchain, Surety, used NYT classified ads to timestamp documents (1995)

So what's the big deal with public blockchains?

Big deal 1: open, self-serve access

It’s easy to underestimate the value of self-serve infrastructure

- Listing an item on ebay vs. holding a garage sale (or Christie’s auction)
- Creating a web site vs. putting books/periodicals/CD-ROMs on store shelves

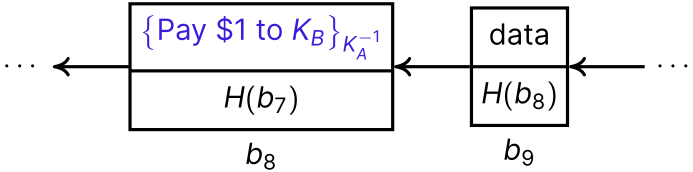
Difference even more salient when it comes to software development

- LiveScript (JavaScript) vs. native browser features
- App store vs. feature phone built-in apps
- PayPal/Stripe vs. becoming a Visa merchant acquirer

Bockchain + smart contracts let you program your money

- Increasing the number of possible developers increases the pace of innovation

Big deal 2: secure transactions in new contexts



$\{\text{Pay \$1 to } K_B\}_{K_A^{-1}}$

$H(b_7)$

b_8

data

$H(b_8)$

b_9

Anyone can verify that the block history hasn't been tampered with

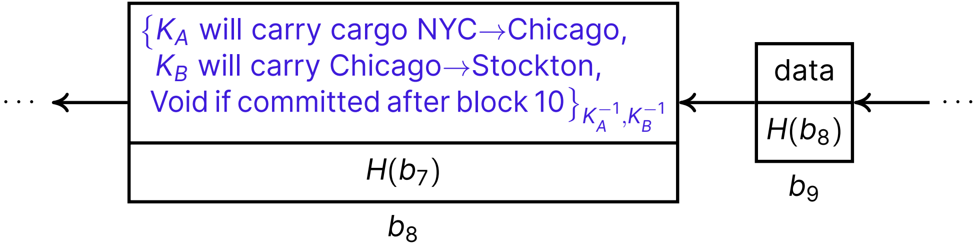
Mutually distrustful parties can commit secure atomic transactions

- Use digital signatures to authenticate transfer of assets
- Blockchain history prevents double-spending previously transferred asset

Commit distributed transactions across any two databases

- Can build products across services that weren't intended to interact
- Blockchain can acts as a reliable *two-phase commit coordinator*

Big deal 2: secure transactions in new contexts



$\{K_A$ will carry cargo NYC→Chicago,
K_B will carry Chicago→Stockton,
Void if committed after block 10$\}_{K_A^{-1},K_B^{-1}}$

$H(b_7)$

b_8

data

$H(b_8)$

b_9

Anyone can verify that the block history hasn't been tampered with

Mutually distrustful parties can commit secure atomic transactions

- Use digital signatures to authenticate transfer of assets
- Blockchain history prevents double-spending previously transferred asset

Commit distributed transactions across any two databases

- Can build products across services that weren't intended to interact
- Blockchain can acts as a reliable *two-phase commit coordinator*

Two big challenges for public blockchains

1. How can everyone agree on the blockchain history?
   - Most blockchains use incentive-based consensus:
     Reward people with cyptocurrency for making blockchain agreement more secure
   - Proof of Work (PoW) – solve easy-to-verify computational puzzle based on block
   - Proof of Stake (PoS) – blocks chosen by people holding cryptocurrency
   - Great mechanism for distributing newly created cryptocurrency
   - But what if you care about non-cryptocurrency *issued assets*?
2. How can blockchains scale to arbitrarily many transactions per second?
   - Transactions in a block must be executed deterministically to replicate ledger state
   - Speeding up with multiple CPU cores can introduce non-determinism

Challenge 1: Consensus incentives vs. issued assets

Issued assets derive value from a real-world commitment by the issuer

- CBDCs, deposit-backed stablecoins, tokenized securities, carbon credits, . . .

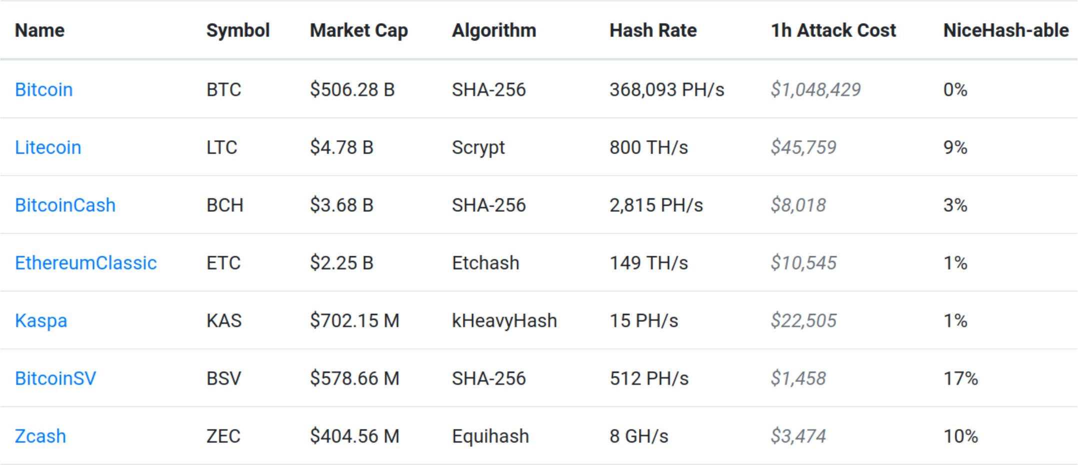
Incentive-based consensus requires belief in future value of cryptocurrency

- Main source of crypto value today is increasing cryptocurrency prices
- Only intrinsic value of cryptocurrency is paying transaction fees

Issued assets pose different trade-offs from cryptocurrency

- Utility is cross-asset interoperability & bootstrapping markets, not crypto valuation
- Can’t subject issuers to the whims of anonymous miners/stakers
- Fees are strictly negative (asset users suffer from friction on issued assets)

PoW: Precarious Balance of Incentives & Security[1]



Name	Symbol	Market Cap	Algorithm	Hash Rate	1h Attack Cost	NiceHash-able
Bitcoin	BTC	$506.28 B	SHA-256	368,093 PH/s	*$1,048,429*	0%
Litecoin	LTC	$4.78 B	Scrypt	800 TH/s	*$45,759*	9%
BitcoinCash	BCH	$3.68 B	SHA-256	2,815 PH/s	*$8,018*	3%
EthereumClassic	ETC	$2.25 B	Etchash	149 TH/s	*$10,545*	1%
Kaspa	KAS	$702.15 M	kHeavyHash	15 PH/s	*$22,505*	1%
BitcoinSV	BSV	$578.66 M	SHA-256	512 PH/s	*$1,458*	17%
Zcash	ZEC	$404.56 M	Equihash	8 GH/s	*$3,474*	10%

[1] Source: crypto51.app

PoS: Precarious Balance of Incentives & Security[2]

Chain	Avg. TPS	Staked	Incentives/yr	Fees/yr	Slashing
Algorand	7	$0.2B	$16M	$29M	N
Avalanche	10*	$2.8B	$205M	$8M	N
Ethereum	13	$40.0B	$1,750M	$1,910M	Y
Polygon	31	$2.0B	$120M	$19M	Y
Solana	452	$8.0B	$559M	$17M	Y
Stellar	60	reputation	n/a	$41k	n/a

Ethereum fees too high ($∼5/tx) for many applications

Algorand security too low ($235M) for many applications

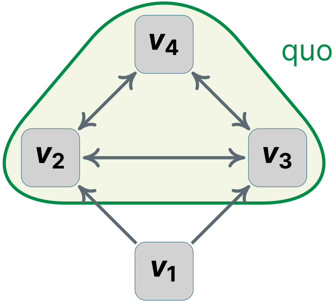
- Liquid staking, shorting, derivatives further reduce security

For most PoS, crypto speculators unsustainably subsidize validators (6–32×)

Stellar not PoS or incentive-based, enjoys lowest fees

[2] Sources: realtps.net, stakingrewards.com, tokenterminal.com

Stellar’s alternative: Proof of Agreement



quorum for v_2, v_3, v_4

$$\text{slices}(v_1) = \{\{v_1, v_2, v_3\}\}$$

$$\text{slices}(v_2) = \text{slices}(v_3) = \text{slices}(v_4) = \{\{v_2, v_3, v_4\}\}$$

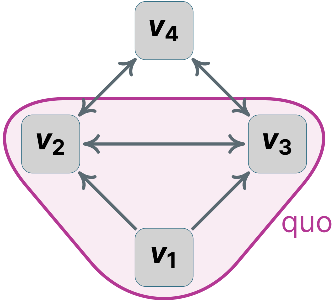
Use the value of inter-organizational agreement as the basis of consensus[3]

- Network run by non-anonymous *validators* (e.g., asset issuers, wallet publishers)
- Specify which other validators they must agree with—quorum slices
- Only reach consensus when validators mutually satisfy every validator’s requirements

For details, see paper in SOSP 2019 – `https://stellar.org/sosp19`

[3] if you don’t need cryptocurrency distribution

Stellar's alternative: Proof of Agreement



$\text{slices}(v_1) = \{\{v_1, v_2, v_3\}\}$

$\text{slices}(v_2) = \text{slices}(v_3) = \text{slices}(v_4) = \{\{v_2, v_3, v_4\}\}$

quorum slice for v_1, but not a quorum

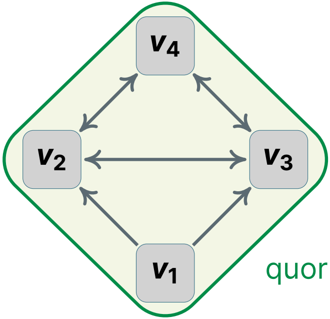
Use the value of inter-organizational agreement as the basis of consensus[3]

- Network run by non-anonymous *validators* (e.g., asset issuers, wallet publishers)
- Specify which other validators they must agree with—quorum slices
- Only reach consensus when validators mutually satisfy every validator's requirements

For details, see paper in SOSP 2019 – `https://stellar.org/sosp19`

[3] if you don't need cryptocurrency distribution

Stellar's alternative: Proof of Agreement



$$\text{slices}(v_1) = \{\{v_1, v_2, v_3\}\}$$

$$\text{slices}(v_2) = \text{slices}(v_3) = \text{slices}(v_4) = \{\{v_2, v_3, v_4\}\}$$

quorum for $v_1, \ldots, v_4$

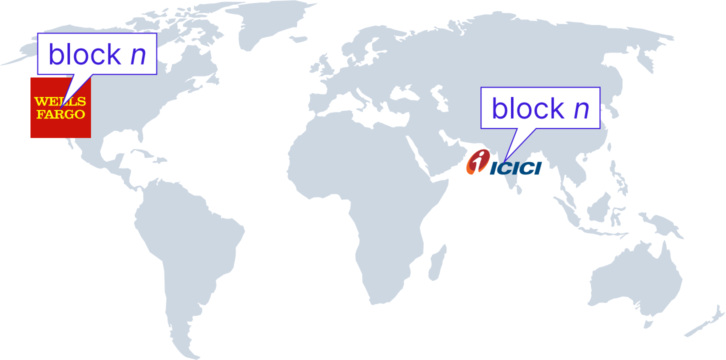
Use the value of inter-organizational agreement as the basis of consensus[3]

- Network run by non-anonymous *validators* (e.g., asset issuers, wallet publishers)
- Specify which other validators they must agree with—quorum slices
- Only reach consensus when validators mutually satisfy every validator's requirements

For details, see paper in SOSP 2019 – `https://stellar.org/sosp19`

[3] if you don't need cryptocurrency distribution

The Internet hypothesis



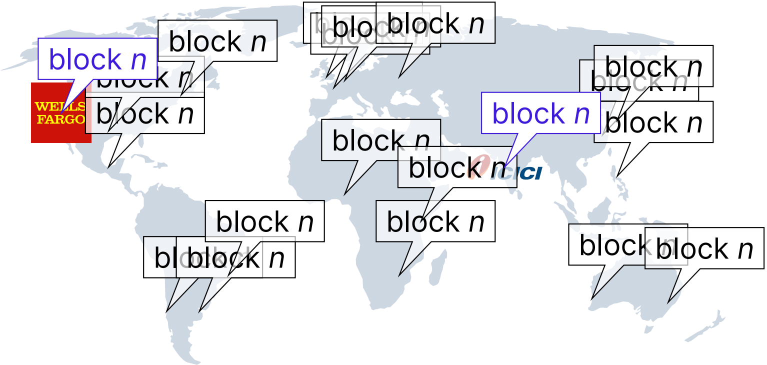
Will two organizations that don't include each other agree on blockchain state?

- Yes if the quorum slice graph transitively converges

Hypothesis: any two validators you'd care about share a transitive dependence

- True of Internet (e.g., China⟷Stanford⟷Google) and correspondent banking
- If they don't, maybe it doesn't matter (risk limited to in-flight transactions)

The Internet hypothesis



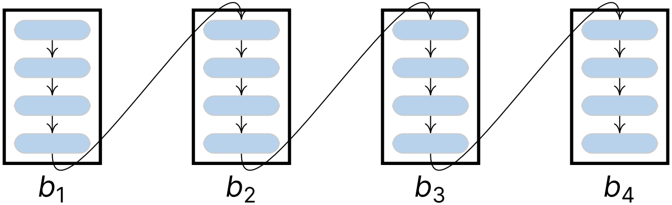
Will two organizations that don't include each other agree on blockchain state?

- Yes if the quorum slice graph transitively converges

Hypothesis: any two validators you'd care about share a transitive dependence

- True of Internet (e.g., China ⟷ Stanford ⟷ Google) and correspondent banking
- If they don't, maybe it doesn't matter (risk limited to in-flight transactions)

Challenge 2: Scaling transaction throughput



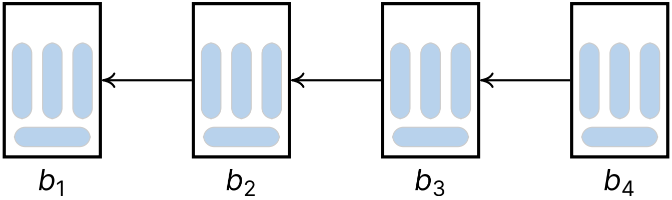
Traditionally transactions totally ordered, each tx sees effects of previous ones

- Smart contract transactions may contain arbitrary code
- Observable execution order must be the same at all replicas
- Worst case: all transactions potentially conflict, can only use one CPU

Groundhog [Ramseyer'23]: Ordered batches of unordered transactions

- Every transaction in a block sees the same initial ledger state
- Transactions are commutative—output is independent of execution order
- Contracts that need serialization can implement it themselves

Challenge 2: Scaling transaction throughput



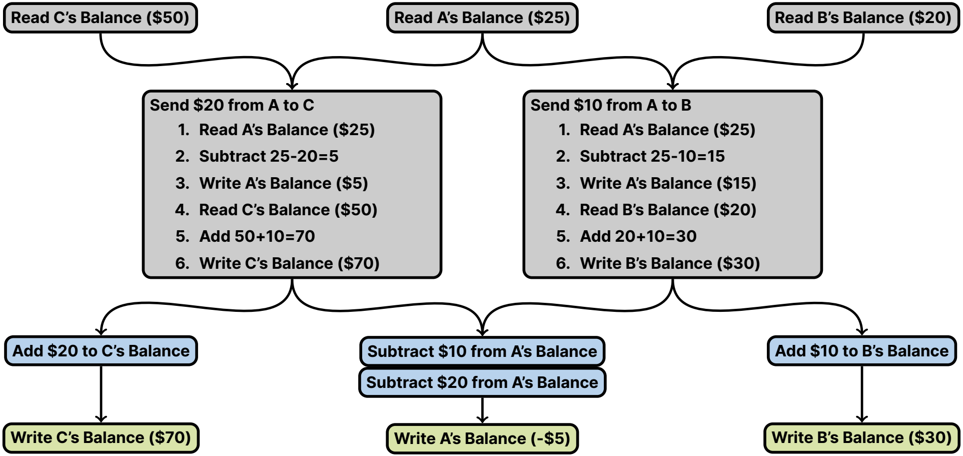
Traditionally transactions totally ordered, each tx sees effects of previous ones

- Smart contract transactions may contain arbitrary code
- Observable execution order must be the same at all replicas
- Worst case: all transactions potentially conflict, can only use one CPU

Groundhog [Ramseyer'23]: Ordered batches of unordered transactions

- Every transaction in a block sees the same initial ledger state
- Transactions are commutative—output is independent of execution order
- Contracts that need serialization can implement it themselves

Groundhog: commutative smart contracts



Read C's Balance ($50)

Read A's Balance ($25)

Read B's Balance ($20)

Send $20 from A to C

1. Read A's Balance ($25)
2. Subtract 25-20=5
3. Write A's Balance ($5)
4. Read C's Balance ($50)
5. Add 50+10=70
6. Write C's Balance ($70)

Send $10 from A to B

1. Read A's Balance ($25)
2. Subtract 25-10=15
3. Write A's Balance ($15)
4. Read B's Balance ($20)
5. Add 20+10=30
6. Write B's Balance ($30)

Add $20 to C's Balance

Subtract $10 from A's Balance

Subtract $20 from A's Balance

Add $10 to B's Balance

Write C's Balance ($70)

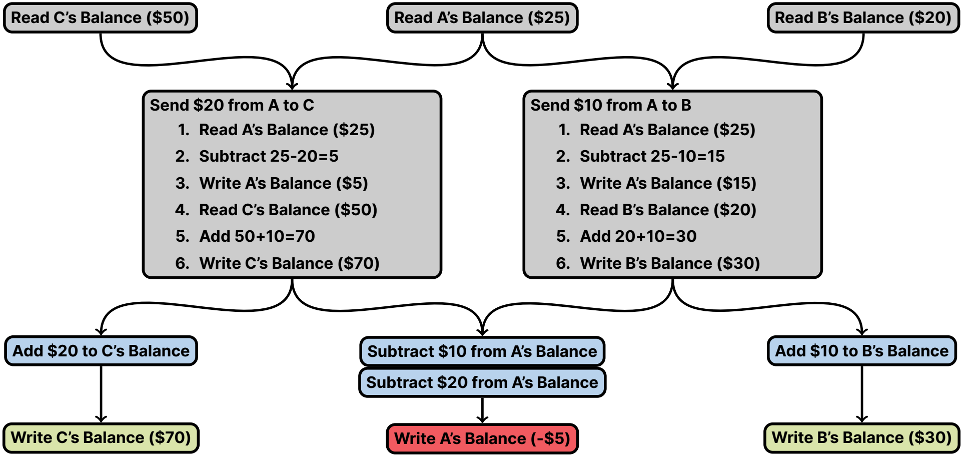
Write A's Balance (-$5)

Write B's Balance ($30)

Each transaction outputs a list of (mostly) commutative deltas

- E.g., Subtract $10 from this account, not set account to $15
- Deltas can be efficiently combined locally to minimize cache contention

Handling conflicts



Read C's Balance ($50)

Read A's Balance ($25)

Read B's Balance ($20)

Send $20 from A to C

1. Read A's Balance ($25)
2. Subtract 25-20=5
3. Write A's Balance ($5)
4. Read C's Balance ($50)
5. Add 50+10=70
6. Write C's Balance ($70)

Send $10 from A to B

1. Read A's Balance ($25)
2. Subtract 25-10=15
3. Write A's Balance ($15)
4. Read B's Balance ($20)
5. Add 20+10=30
6. Write B's Balance ($30)

Add $20 to C's Balance

Subtract $10 from A's Balance

Subtract $20 from A's Balance

Add $10 to B's Balance

Write C's Balance ($70)

Write A's Balance (-$5)

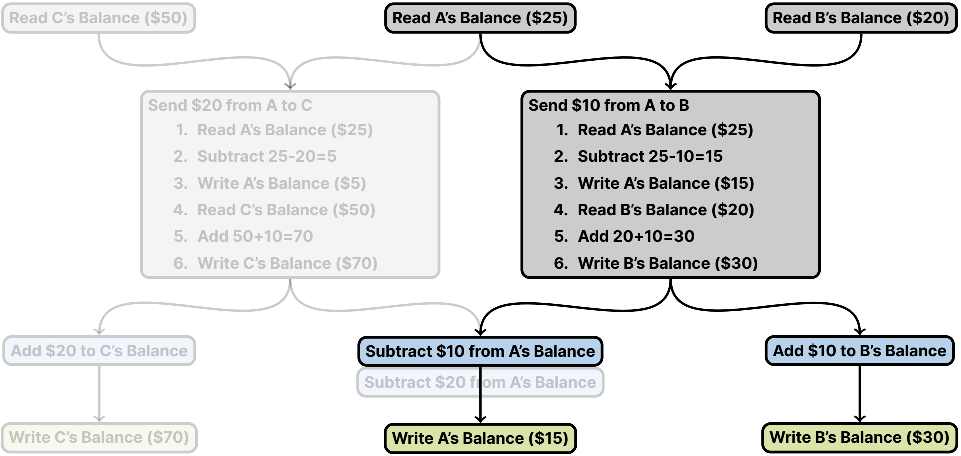
Write B's Balance ($30)

Some operations still conflict or violate invariants

- Can't create and use account in same block, can't set balance below $0

Conservatively exclude conflicting transactions when proposing block

Handling conflicts



Read C's Balance ($50)

Read A's Balance ($25)

Read B's Balance ($20)

Send $20 from A to C

1. Read A's Balance ($25)
2. Subtract 25-20=5
3. Write A's Balance ($5)
4. Read C's Balance ($50)
5. Add 50+10=70
6. Write C's Balance ($70)

Send $10 from A to B

1. Read A's Balance ($25)
2. Subtract 25-10=15
3. Write A's Balance ($15)
4. Read B's Balance ($20)
5. Add 20+10=30
6. Write B's Balance ($30)

Add $20 to C's Balance

Subtract $10 from A's Balance

Subtract $20 from A's Balance

Add $10 to B's Balance

Write C's Balance ($70)

Write A's Balance ($15)

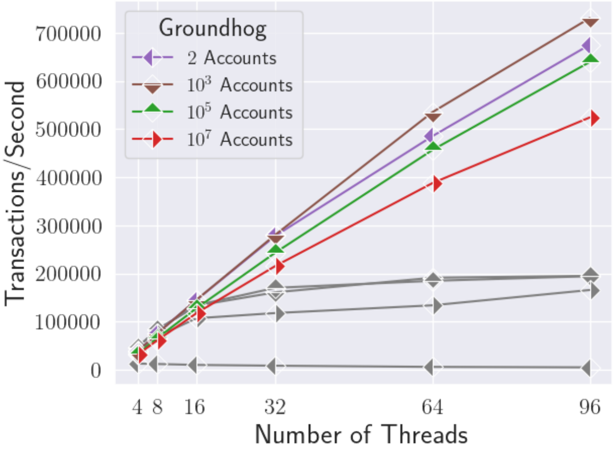
Write B's Balance ($30)

Some operations still conflict or violate invariants

- Can't create and use account in same block, can't set balance below $0

Conservatively exclude conflicting transactions when proposing block

Groundhog results



A few simple primitives suffice to implement many popular smart contracts

- E.g., non-negative integers (balances, semaphores), key-value maps (queues)

Far more scalable than BlockSTM (prior state of the art, shown in gray)

Can we scale trading?

Continuous double auctions are a key mechanism for efficient markets

- Limit orders that fill each other when the prices cross
- Worst case for parallelism—every operation modifies one of a few order books
- Until recently, people had give up on scaling in-blockchain asset exchanges

Key idea: make trading operations commutative

- Use Arrow-Debreu batch exchange market
- Each asset A gets fixed valuation p_A for a batch of trades
- Anyone paying A to buy B pays the price p_B/p_A—so trades commutative

Other benefits of Arrow-Debreu exchange markets

- Eliminates cyclic arbitrage (no need to trade through intermediary currencies)
- Eliminates in-block front running MEV (“miner extractable value”)

SPEEDEX [NSDI'23]: Batch Trading

Input: Block of Offers

1. Compute Valuations
   - New approach computes approximate valuations fast and scalably
2. Trade with SPEEDEX at Valuation Quotients
   - Meaningless units
   - No pairwise matching!

Trade 10 USD for EUR
min $\frac{9}{10}\frac{EUR}{USD}$

Trade 9 EUR for JPY
min $140\frac{JPY}{EUR}$

Trade 1350 JPY for USD
min $\frac{1}{135}\frac{USD}{JPY}$

Trade 10000 USD for EUR
min $1000\frac{EUR}{USD}$

SPEEDEX Pricing Engine

$$p_{USD} = 9$$
$$p_{EUR} = 10$$
$$p_{JPY} = \frac{1}{15}$$

"Clearing" if no surplus or debt

Theorem (Arrow and Debreu, 1954)

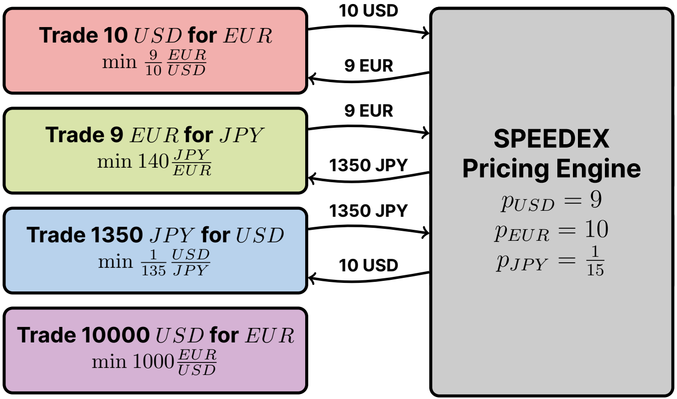
There always exists a unique* set of valuations $\{p_A\}$ that clears the market.

SPEEDEX [NSDI’23]: Batch Trading

Input: Block of Offers

1. Compute Valuations
   - New approach computes approximate valuations fast and scalably
2. Trade with SPEEDEX at Valuation Quotients
   - Meaningless units
   - No pairwise matching!

“Clearing” if no surplus or debt



Trade 10 USD for EUR
min $\frac{9}{10}\frac{EUR}{USD}$

Trade 9 EUR for JPY
min $140\frac{JPY}{EUR}$

Trade 1350 JPY for USD
min $\frac{1}{135}\frac{USD}{JPY}$

Trade 10000 USD for EUR
min $1000\frac{EUR}{USD}$

10 USD

9 EUR

9 EUR

1350 JPY

1350 JPY

10 USD

SPEEDEX Pricing Engine

$p_{USD} = 9$

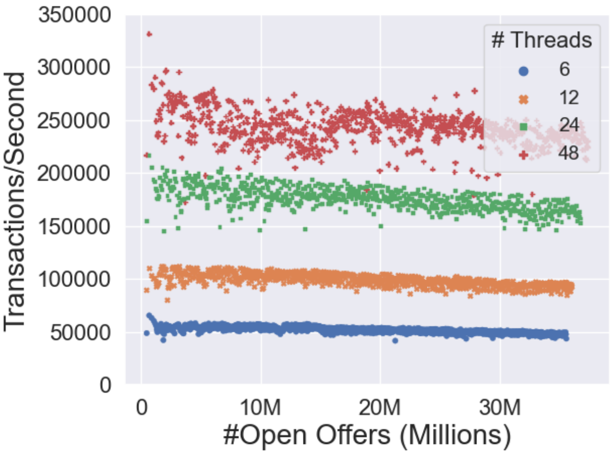
$p_{EUR} = 10$

$p_{JPY} = \frac{1}{15}$

Theorem (Arrow and Debreu, 1954)

There always exists a unique set of valuations $\{p_A\}$ that clears the market.*

SPEEDEX results



50 assets, 10M accounts, near linear scalability (shown on 48-core machine)

- (log dependence on the number of open offers)

Lessons for next-generation fintech APIs

Provide open, self-service APIs

- Developers can innovate in ways designer didn’t anticipate

Enabling general innovation will enable innovation in scams

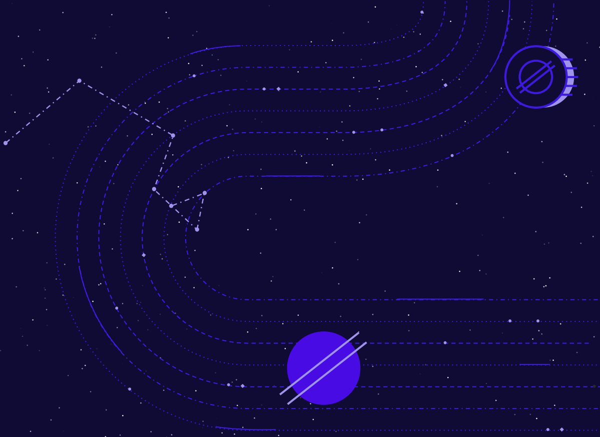
- Don’t sandbag financial infrastructure just because it can be misused
- Establish norms around difficult-to-misuse interfaces
  (e.g., expose secure names in the UI)

Use blockchains for public auditability – even in a centralized system

- Future systems should support distributed transactions with unknown/untrusted systems, including blockchains

Achieve scalability through a commutative APIs

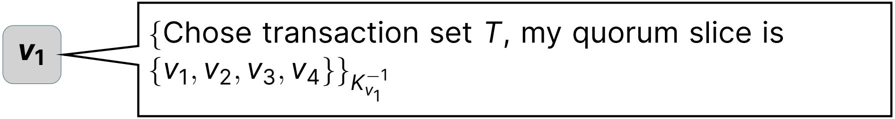
- Ordered blocks of unordered transactions: blockchain-friendly & scalable throughput



Thank You

www.stellar.org
(fdc.stanford.edu)

The Stellar Consensus Protocol (SCP)



$\{$Chose transaction set T, my quorum slice is $\{v_1, v_2, v_3, v_4\}\}_{K_{v_1}^{-1}}$

First, proof-of-agreement protocol, SCP, entered production use in 2015

Validators digitally sign all protocol messages

- Every message specifies the validator's quorum slice
- For redundancy, nodes specify multiple quorum slices

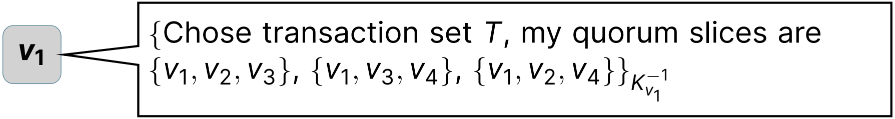
Proceed when you have a quorum: a set that includes at least one slice for each member

- Some subtleties required to ensure liveness

Additional benefits beyond proof of agreement:

- Low computational cost, low energy consumption, low latency
- Only assumes digital signatures and hashes

The Stellar Consensus Protocol (SCP)



$\{$Chose transaction set T, my quorum slices are $\{v_1, v_2, v_3\}$, $\{v_1, v_3, v_4\}$, $\{v_1, v_2, v_4\}\}_{K_{v_1}^{-1}}$

First, proof-of-agreement protocol, SCP, entered production use in 2015

Validators digitally sign all protocol messages

- Every message specifies the validator's quorum slices
- For redundancy, nodes specify multiple quorum slices

Proceed when you have a quorum: a set that includes at least one slice for each member

- Some subtleties required to ensure liveness

Additional benefits beyond proof of agreement:

- Low computational cost, low energy consumption, low latency
- Only assumes digital signatures and hashes