MystenLabs

# Sui Lutris: Combining Broadcast and Consensus in a Production Blockchain System

**George Danezis**
Professor, University College London, g.danezis@ucl.ac.uk
Chief Scientist, Mysten Labs, george@mystenlabs.com

UCL 2023 | NOVEMBER | 2023

# Introductions

- **Microsoft Research,**
  Researcher, 2007-2013

- **University College London,**
  Prof. of Security and Privacy Engineering, 2013 - Now

- **Chainspace, Co-founder,**
  Head of Research, 2018

- **Facebook Novi, Libra / Diem Blockchain**
  Principal Researcher, 2019 - 2021

- **Mysten Labs, Co-founder, Sui Blockchain**
  Chief Scientist, 2021 - Now

**Involved in:**
Vega Protocol
Nym Technologies
Celestia
Linera

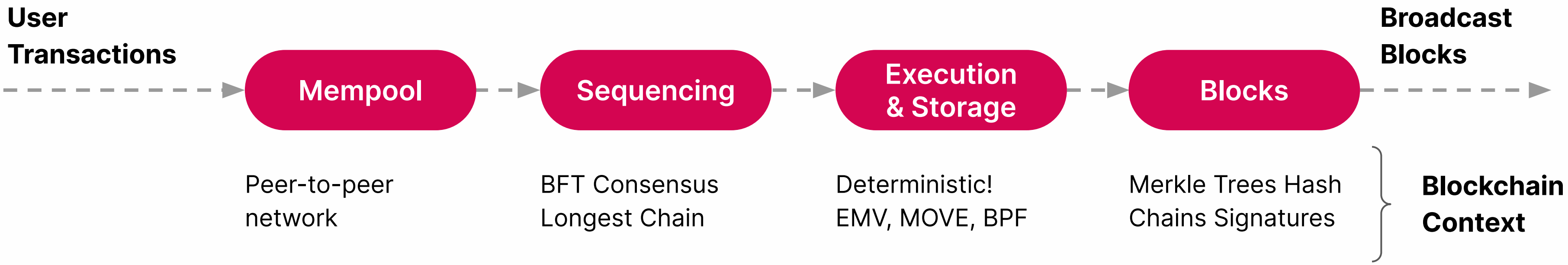
# What Makes a Blockchain?

**Distributed / Replicated Transaction Processing** — Today we talk about this

- Sybil Resistance / permission-less-ness
- Tokenomics / incentives / gas
- High-integrity Data Structures
- Privacy

Lots to say, another time

# Replicated Transaction Processing *ala* State Machine Replication (SMR)




**Vega Protocol.** Danezis, G., Hrycyszyn, D., Mannerings, B., Rudolph, T., & Šiška, D. (2019).
**State machine replication in the libra blockchain.** Baudet, Mathieu, Avery Ching, Andrey Chursin, George Danezis, François Garillot, Zekun Li, Dahlia Malkhi, Oded Naor, Dmitri Perelman, and Alberto Sonnino. The Libra Assn., Tech. Rep 7 (2019).

# State Machine Replication (SMR) and its Discontents

**User Transactions** → **Mempool** → **Sequencing** → **Execution & Storage** → **Blocks** → **Broadcast Blocks**

**Mempool**

- Chaos Reigns
- Little Research BFT troubles,
- Latency, scaling

**Sequencing**

- Seconds to Minutes latency
- Tolerance to malicious/failed validators
- Low throughput
- Hard to build

**Execution & Storage**

- Single Thread execution
- VMs upon VMs
- Storage Limits
- Slow

**Blocks**

- Expensive to maintain MT
- Interplay between CPU / store
- Batching = latency

# Solutions within the SMR Architecture

User Transactions → Mempool → Sequencing → Execution & Storage → Blocks → Broadcast Blocks

Narwhal Mempool / Tusk (Bullshark)

Parallel Execution Block STM

Chainspace

Leveraging multiple hosts/cores per validator to scale?

Lower Latency?

**Narwhal and Tusk: a DAG-based mempool and efficient BFT consensus.** George Danezis, Lefteris Kokoris-Kogias, Alberto Sonnino, Alexander Spiegelman: EuroSys 2022: 34-50. **Block-STM: Scaling Blockchain Execution by Turning Ordering Curse to a Performance Blessing.** Rati Gelashvili, Alexander Spiegelman, Zhuolun Xiang, George Danezis, Zekun Li, Dahlia Malkhi, Yu Xia, Runtian Zhou: PPoPP 2023: 232-244
**Chainspace: A Sharded Smart Contracts Platform.** Mustafa Al-Bassam, Alberto Sonnino, Shehar Bano, Dave Hrycyszyn, George Danezis: NDSS 2018

# Consensus-less Agreement based Cryptocurrencies

**You do not need consensus to have a cryptocurrency (Guerraoui et al)**

**BUT No liveness for incorrect initiator / many uncoordinated initiatiators**

**FastPay: High-Performance Byzantine Fault Tolerant Settlement.**
Mathieu Baudet, George Danezis, Alberto Sonnino:
AFT 2020: 163-177

## Use weaker primitive: Consistent / Reliable Broadcast

- One channel initiator (broadcast)
- Many replicas (decide broadcast value) < ⅓ byzantine

## Informal properties:

- Safety: if two replicas reach a decision on a broadcast value its the same!
- Liveness: a correct initiator can always drive to reaching a decision

## One channel per coin, broadcast value is the new owner / channel initiator

# Consistent Broadcast

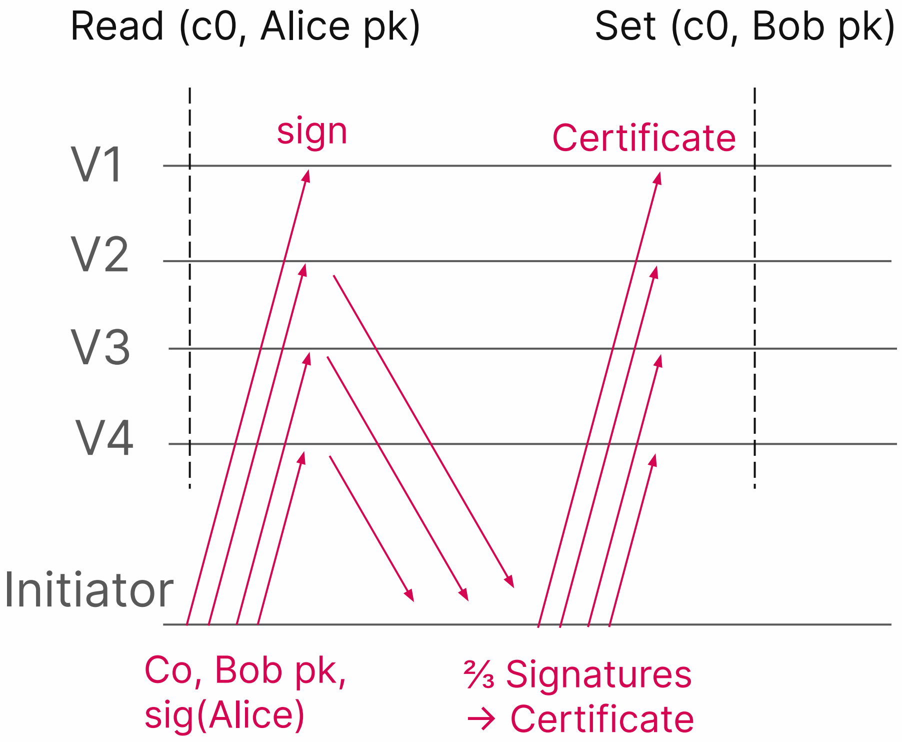
**Alice has a coin c0.**
**She wants to send it to Bob**

Initially all read (c0, Alice pk).

Correct Replica signs first authenticated request.

After all set (c0, Bob pk)

**What happens if 1 corrupt?**
**What happens if sender corrupt?**



# Fastpay and its Discontents

| | | | |
|---|---|---|---|
| Fastpay | 40K - 160K payments/s<br>45-75 1-core shards | 200ms-300ms<br>finality | Not bad! |

**Account associated with address, sequence number and balance.**
**A signed sequenced transaction transfers some of the balance to another / new account, update seq.**

**But:**

- How to extend to generic smart contracts?
- How to generate a canonical history of the replicated system?
- How to allow multi-owner objects?
- How to allow committee reconfiguration?
- Privacy? (Zef)
- How to unlock locked objects?

**Zef: Low-latency, Scalable, Private Payments.** Mathieu Baudet, Alberto Sonnino, Mahimna Kelkar, George Danezis: CoRR abs/2201.05671 (2022). **Linera** start-up

# How to Combine a Fast Path & Consensus Path?

- General smart contract platform (MoveVM + Objects)
- Fast path / low latency / simple scaling for owned objects
- Consensus path to support shared objects
- Parallel execution / early finality
- Chekpoints & reconfiguration

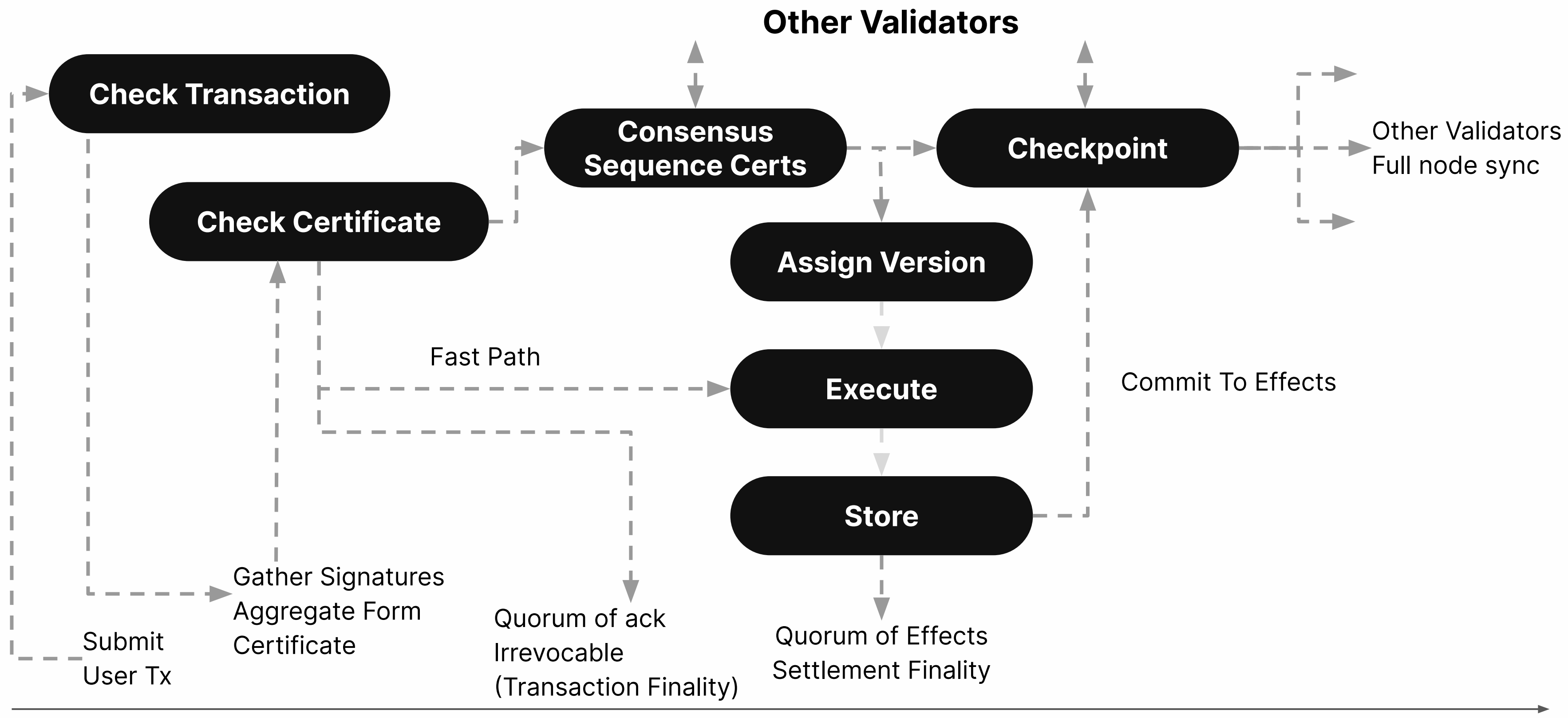
**Scale via validators using many core / hosts**

**Many things happen at the same time.**

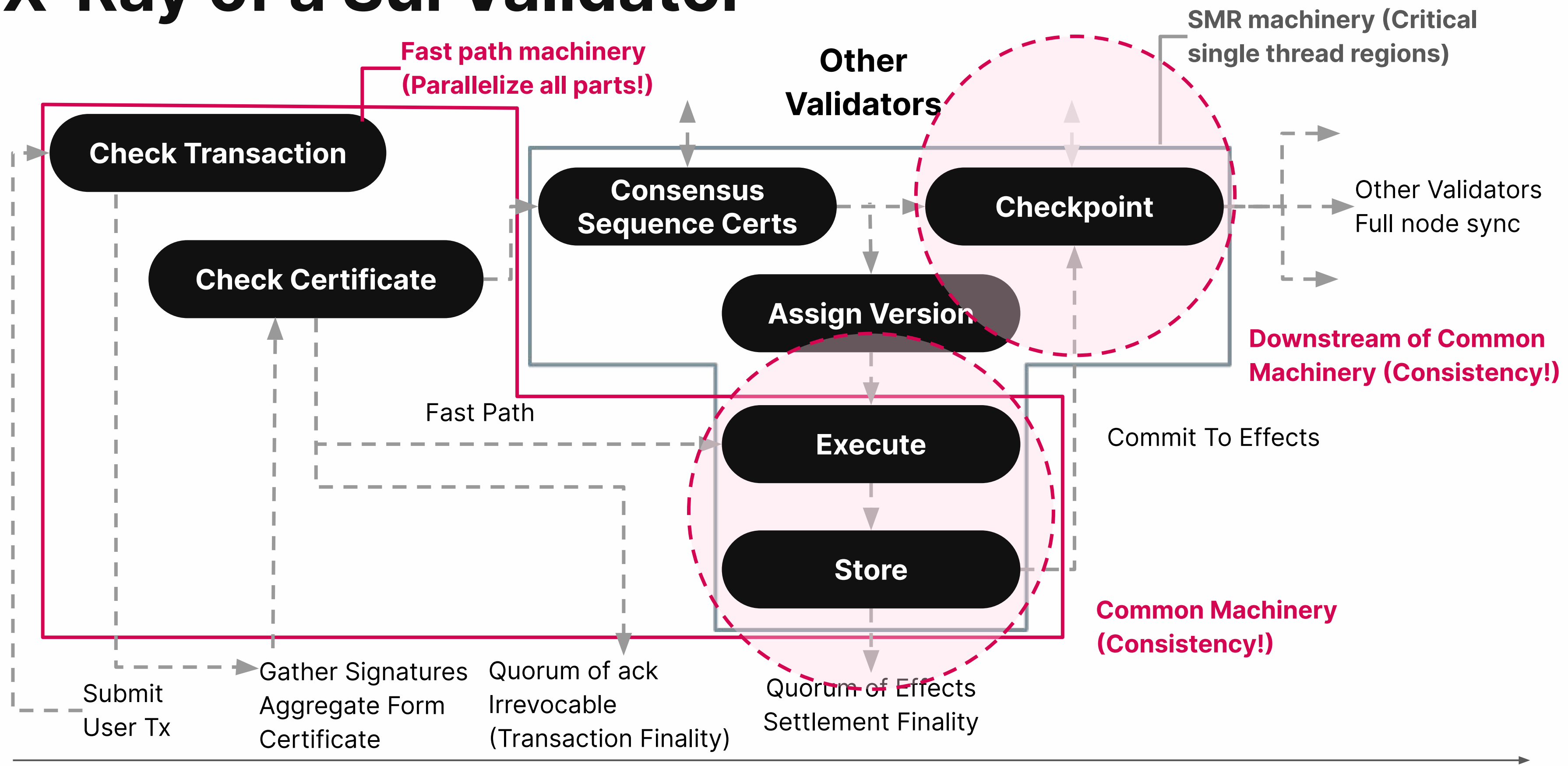
**Safety / consistency despite this!**

**Integrated as the base mechanism in the Sui Blockchain!**

# X-Ray of a Sui Validator



# X-Ray of a Sui Validator



# Simplified Data Model

## User Transaction

**Input Objects:**
(ObjID, Version) → Owned Objects (current version) / Shared Objects (initial version)

**Command:**
(pkg, name, args)

**Signature**

## Object Store

**(ObjID, Version)** → **Owner** (owned<Address> / shared), **Move Type**, **Data (Move Struct)**

## Owned Object Locks

**(ObjID, Version)** → **Option<TxID>**

**Atomic: check on Empty key & update**

**State machine: Authenticated Transactions consume object versions, and create new object versions**

# Fast Path: Validator View

Send User Tx
To all Validators

## Check Transaction

- Signature?
- Objects exist?
- Owner correct?
- Test & set this Tx in owned object locks (Atomic!)

Local checks

**All mutable owned object must belong to the user that signs!**

Collect > ⅔ stake Signatures Aggregate into certificate

## Check Certificate

- >⅔ stake signed
- all local checks
- Owned only?
- Sequence

Send to Fast Path Execution

Seq, Exec, Store

Ack

> ⅔ stake Transaction Finality

Effects

> ⅔ stake Settlement Finality

# Fast Path: Validator View

**Assume < ⅓ stake is byzantine, asynchronous network, crypto works.**

**If a certificate on a Tx exists:**

- No other certificate exists containing one or more input owned objects at the same (ObjId, version).
- Certificates exist on correct validators to generate all inputs versions and execute the transaction certified.

**The world of transactions is potentially inconsistent.**
**The world of certificates is consistent with respect to owned objects.**

# Finality: “Irrevocable and Unconditional”

**Transaction Finality:** a transaction will execute and cannot be cancelled

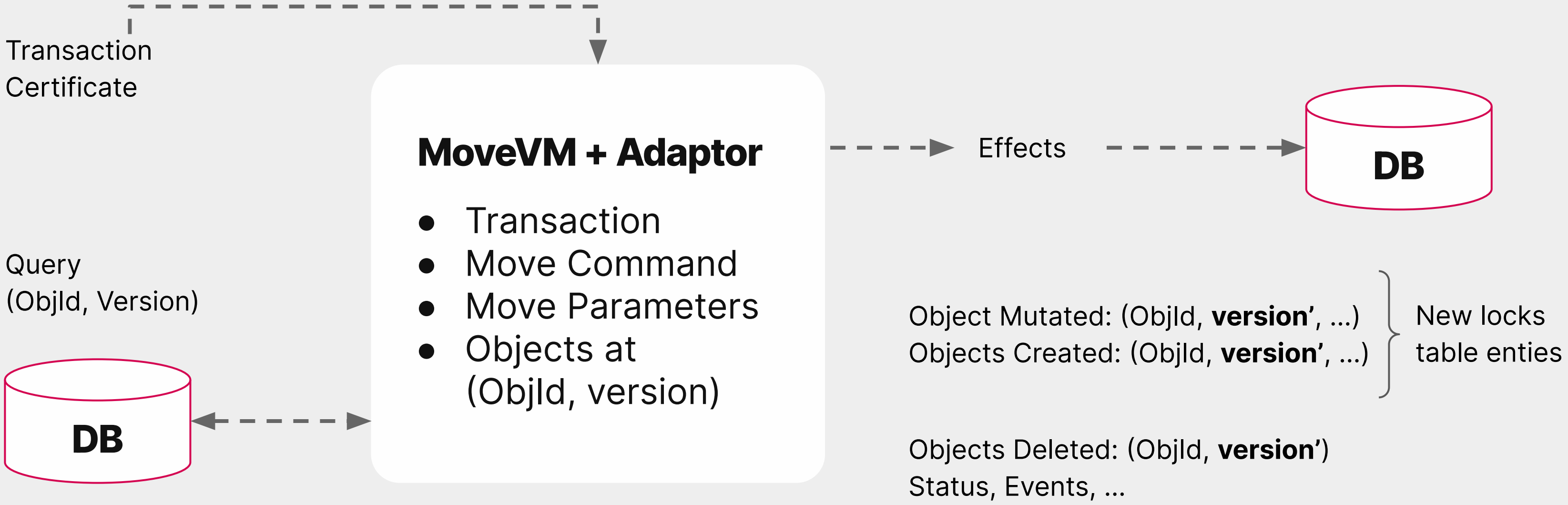
- 2 round trips + processing
- > ⅔ stake Acks after checking certificate
- Guarantee despite failures, malice, epoch change, and concurrent processing

**Settlement Finality:** effects are known and ready to use (assets changed hands)

- > ⅔ same effects after execution
- Before blocks / checkpoints are formed

**Checkpoints and Reconfiguration must respect finality guarantees.**

# Execution: Parallel on all cores




**Use eventually consistent stores, ready to extend to multiple hosts.**

**Lamport Timestamp (max(v_in) +1)**
**Fresh ObjIDs using hash of TxID**

# Shared Objects: What is the challenge?

→ **Disparate users may include the same object as an input to their transaction.**

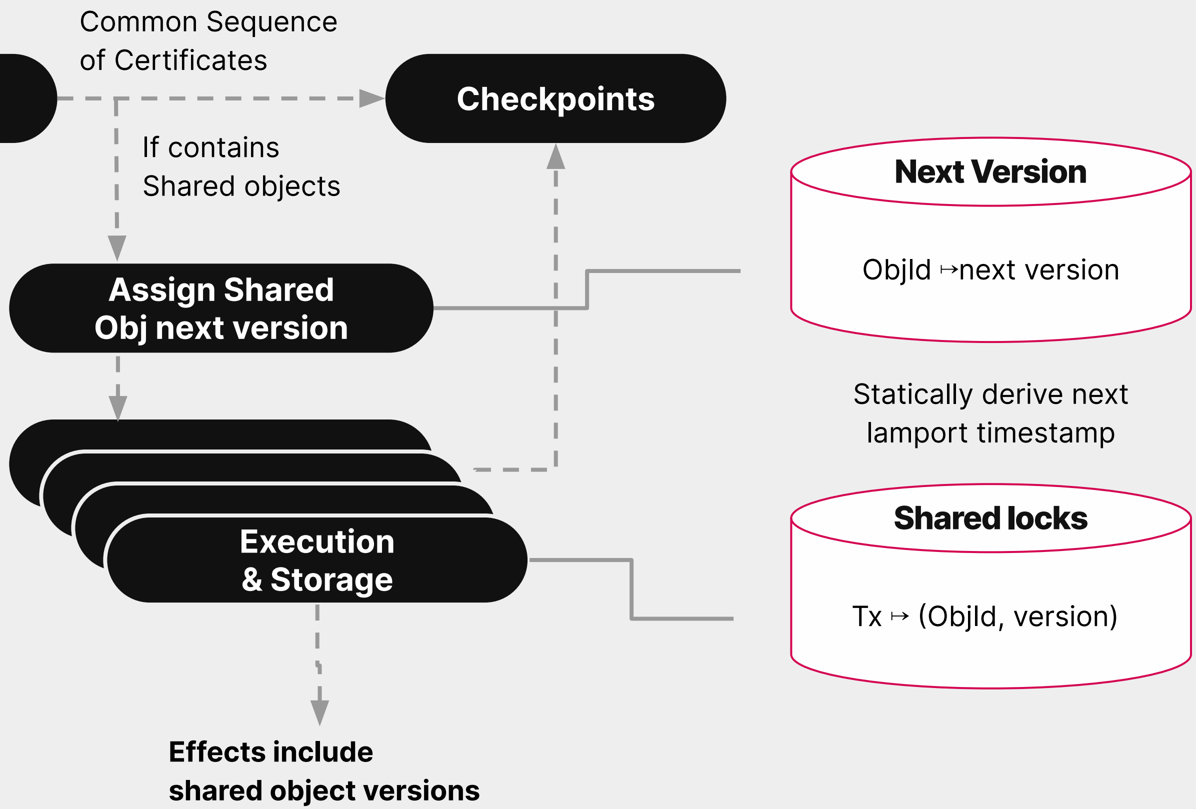
→ **Cannot coordinate to not re-use the same version or have consistent versions.**

→ **System must assign the versions!**

**Shared Object Critical Path:**

- Sequence Certificates with shared objects
- **Statically** assign shared object a version number without execution

# Shared Object Path




**Execution now knows what shared object version to use for each certificate**

**Parallel execution as for owned object transactions**

**World of Effects is consistent!**

# Checkpoints

Want a shared causal history of all executed certificates. **Finality is Earlier!**
Validator Sync, Archival, Epoch Change, Full Node, Completeness, ...

All certificates are sequenced, but may be out of causal order ⇒ need to wait for certificates to "fill in the gaps"

**When a Validator accepts a certificate it will not close the epoch until it is checkpointed.**

**Theorems:**

- If a certificate is sequenced eventually all previous certificates will be sequenced to create a full causal sequence of all final transactions.
- Eventually all final transactions will be included in an epoch checkpoint.

# Reconfiguration & Epoch Change

**2-step process:**

1. Validators stop signing new transactions, to make new certificates.
2. When all certificates received / executed locally are checkpointed,
   A validator votes to close the epoch.

**When > ⅔ stake validators vote to close the epoch (in the checkpoint) the epoch ends.**
**Others may have to revert executions (1-step at most).**

**Theorem:** all final transactions will be within the checkpoints by the end of the epoch.

**Reset all the owned object locks for the new epoch ⇒ Alleviate loss of liveness.**

# Integration into a Production System: Sui

**350K LOC of code (~30K subsystem we discussed)**

**~8400 commits**

**Researchers:**
+60K LOC Initial fastpay + NW/BS prototypes

**1.5 years, team of 70 Eng at the end**

**Devnet since March 2022, 3 Testnets**

**Testnet (30 Apr)**

- 42K Move packages
- 780M Objects
- 269M PTBs
- 2.5M Checkpoints
- ~300 TPS organic
- Induced 130K TPS peak (transfers / large batches in PTB)

**Today:**

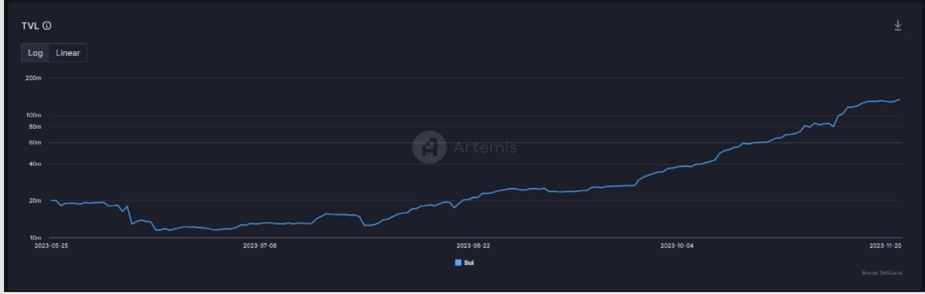
Single machine multi-core implementation. Focus on latency of owned object path.

**Future:**

More Aggressive Multi-core and multi-host. Focus on latency scaling, shared object paths.

# Key metrics May 2023 - Nov 2023







**920.16M** Transaction Blocks **8.97M** Addresses **7.9K** Packages **61M** Objects

# What we have not talked about...

- Programmable Transaction Blocks
- Wrapping / Unwrapping objects
- Objects own other objects
- Dynamic child fields / lookup
- Object deletion
- Networking, DB, Sync, ...

- Move Verifier
- Transaction Verifier
- SDKs, APIs
- Read Interfaces
- Indexing
- Crypto Econ / Gas / Stake

# What kind of performance are we looking at?

## Owned Object Transactions (Optimized path)

- ~500ms latency to transaction / settlement finality
- 200K-300K TPS for simple payments with PTB
  10K TPS for single Tx PTB

## Shared Object Transactions (Conservative for Stability)

- ~500ms to transaction finality 3s-7s p50 settlement finality (NW / Bullshark)
- 7K TPS for shared counter single Tx PTB

**Next step lower NW / BS latencies**
**Add more workers (1 now!)**
**Integrate better fast / consensus path.**

**Geo-distributed but homogenous**
**100 validator network, May 2023**

# The Cutting Edge

Kushal Babel, Andrey Chursin, George Danezis, Lefteris Kokoris-Kogias, Alberto Sonnino: **Mysticeti: Low-Latency DAG Consensus with Fast Commit Path.** CoRR abs/2310.14821 (2023)

Lefteris Kokoris-Kogias, Alberto Sonnino, George Danezis: **Cuttlefish: Expressive Fast Path Blockchains with FastUnlock.** CoRR abs/2309.12715 (2023)

Mathieu Baudet, Alberto Sonnino, Mahimna Kelkar, George Danezis: **Zef: Low-latency, Scalable, Private Payments.** CoRR abs/2201.05671 (2022)

# Conclusion

Production systems need to combine research design patterns to get the right mix of **features, robustness, scaling, performance.**

How to preserve safety and liveness in these **systems** (CPU, DB, Network, Replication, Reads, Writes) is also an exciting **research** area.

**Sui Lutris: A Blockchain Combining Broadcast and Consensus.** Sam Blackshear, Andrey Chursin, George Danezis, Anastasios Kichidis, Lefteris Kokoris-Kogias, Xun Li, Mark Logan, Ashok Menon, Todd Nowacki, Alberto Sonnino, Brandon Williams, Lu Zhang. Technical Report (2023).

MystenLabs