Narwhal and Tusk
A DAG-based Mempool and Efficient BFT Consensus

Alberto Sonnino
Acknowledgements

George Danezis
Lefteris Kokoris-Kogias
Alexander Spiegelman
Alberto Sonnino

Facebook Novi
Byzantine Fault Tolerance

> 2/3
How to build (really) high performance blockchains

The goal of this project
Current Designs

• Monolithic protocol sharing transaction data as part of the consensus
• Optimize overall message complexity of the consensus protocol
Current Designs
Typical leader-based protocols
Current Designs
Typical leader-based solutions
The mempool is the key

Reaching consensus on metadata is cheap
Narwhal

Dag-based mempool
Narwhal
The workers and the primary

Client transactions

Narwhal mempool

Worker 1
Worker 2
Worker n

Primary
Narwhal
The workers and the primary

Client transactions

Transactions

Worker 1

Transactions

Worker 2

Narwhal mempool

Transactions

Worker n

Primary
Narwhal
The workers and the primary
Narwhal
The workers and the primary

Client transactions

Transactions

Worker 1

Worker 2

Worker n

Digest

Digest

Digest

Narwhal mempool

Transactions

Transactions

Transactions

Batch

Batch

Batch
Narwhal
The workers and the primary

Client transactions

Transactions

Worker 1

Worker 2

Worker n

Narwhal mempool

Digest

Batch

Transactions

Primary

Digest

Digest

Digest

Transactions

Transactions

Transactions

'Brien protocol'
Narwhal
The primary machine

block header

G1

H

G2

H

G3

H
Narwhal
The primary machine

block header

G1

G2

G3

certificate

H

V

C

H

V

C

C

C
Narwhal
The primary machine

Round 1
Narwhal
The primary machine

Round 1
Byzantine 'Reliable' Broadcast
Narwhal
The primary machine

r1  r2  r3  r4  r5
Tusk
Just interpret the DAG

r1  r2  r3

Diagram showing a directed acyclic graph with nodes connected by directed edges.
Tusk
The random coin elects the leader of r-2
Tusk
The leader needs $f+1$ links from round $r-1$

Not enough support!
(Nothing is committed at this stage)
Tusk

Nothing is committed and we keep build the DAG
Tusk
Elect the leader of r3

L1
L2

coin
Tusk
Leader L2 has enough support
Tusk
Leader L2 has links to leader L1

First commit L1
Then commit L2
Tusk
Commit all the sub-DAG of the leader
HotStuff on Steroids

Just by replacing the mempool
HotStuff on Narwhal

Overview

Client transactions → Narwhal mempool → Certificates → Partially Synchronous Consensus (HotStuff) → Ordered transactions → State machine replication execution

- Garbage collection
HotStuff on Narwhal
Enhanced commit rule
HotStuff on Narwhal
Enhanced commit rule
HotStuff on Narwhal
Enhanced commit rule
Implementation

• Written in Rust
• Networking: Tokio (TCP)
• Storage: RocksDB
• Cryptography: ed25519-dalek

https://github.com/asonnino/narwhal
Evaluation
Experimental setup on AWS

m5d.8xlarge
Evaluation

Throughput latency graph
Evaluation
Scalability
Evaluation
Performance under faults
Narwhal & Tusk

- Separate consensus and data dissemination for high performance
- Scalable design, egalitarian resource utilizations

• **Code**: https://github.com/facebookresearch/narwhal
Future Works

Interested?

• Performance under DDoS attack?
• Can we embed a partially synchronous consensus into the DAG?
• How to implement scalable execution?