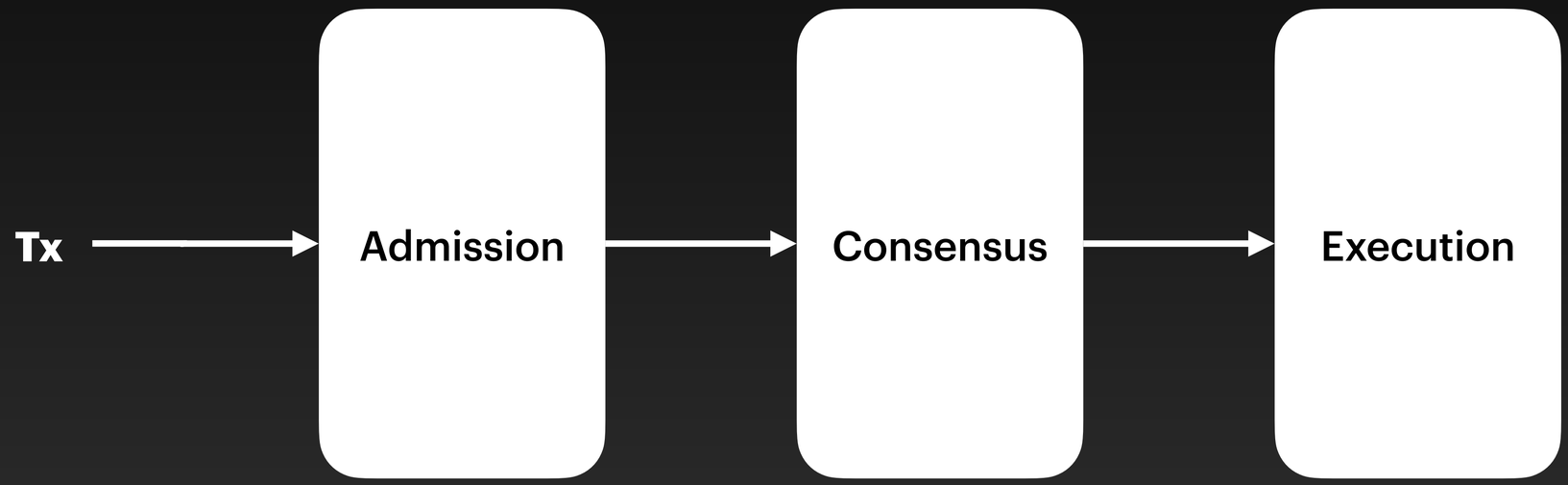
Remora

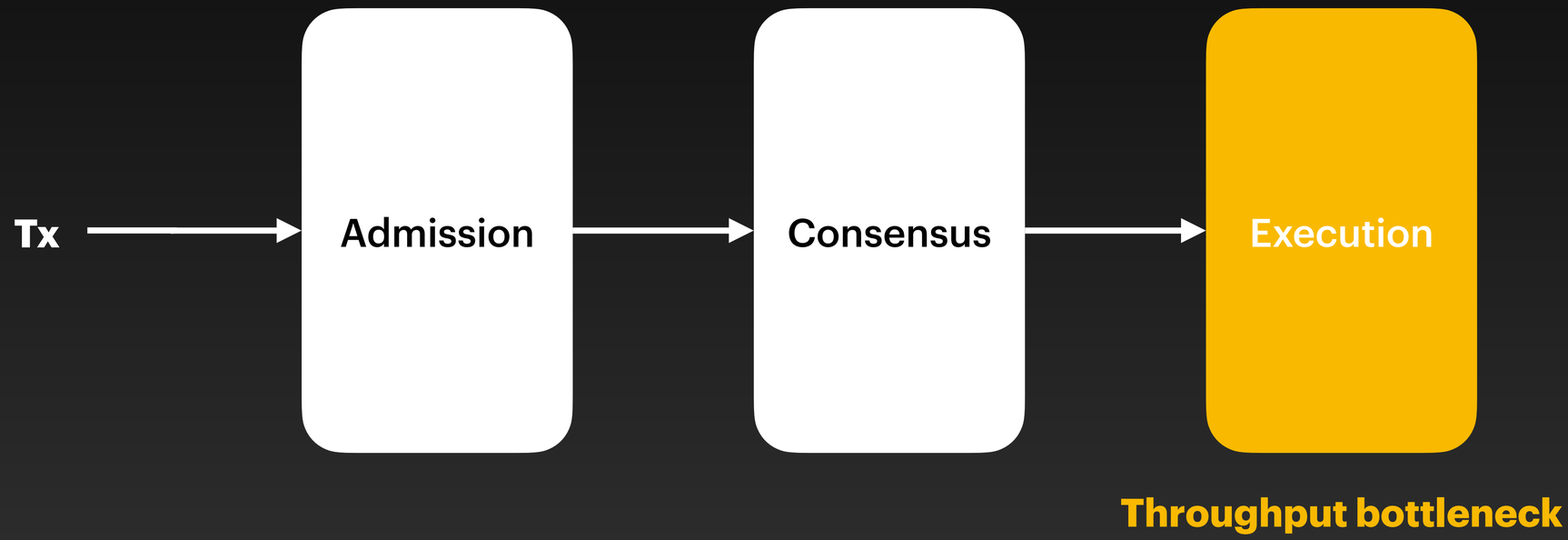
Elastic Asynchronous Distributed Execution

Research offsite 2024

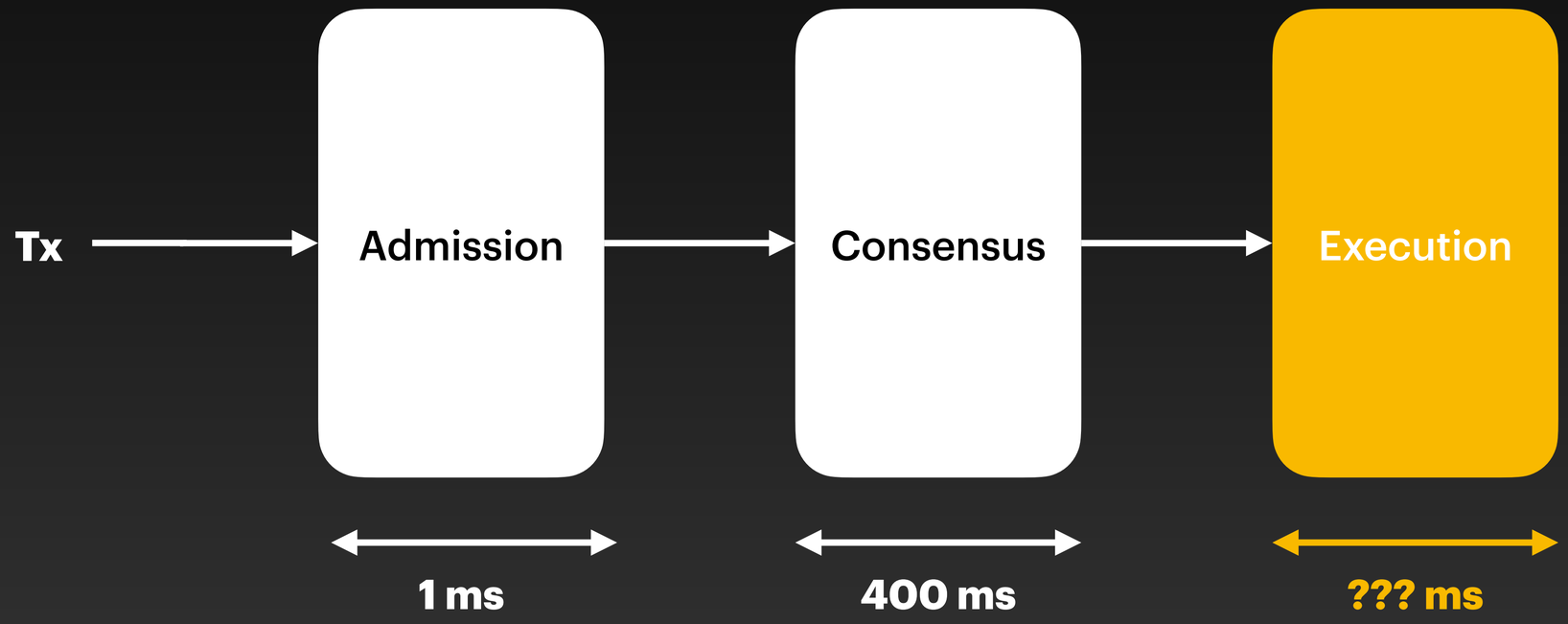
Current State



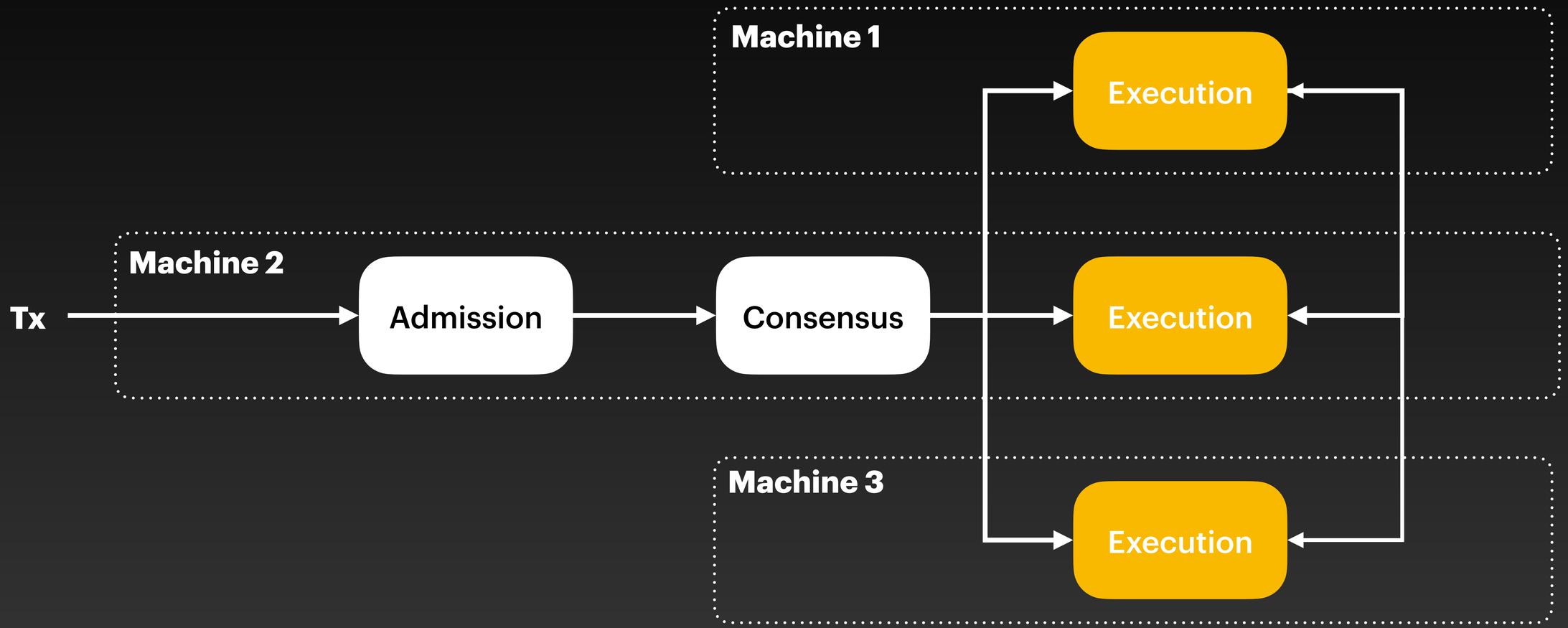
Current State



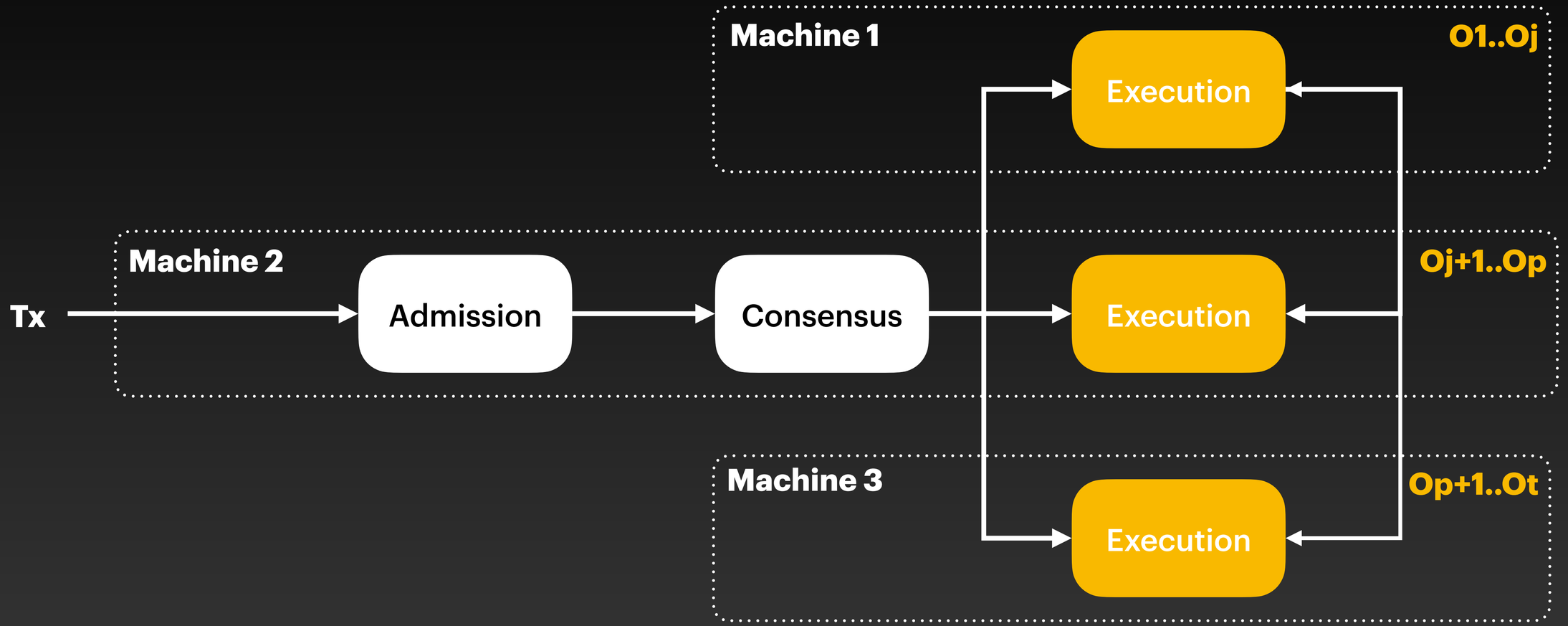
Current State



Pilotfish

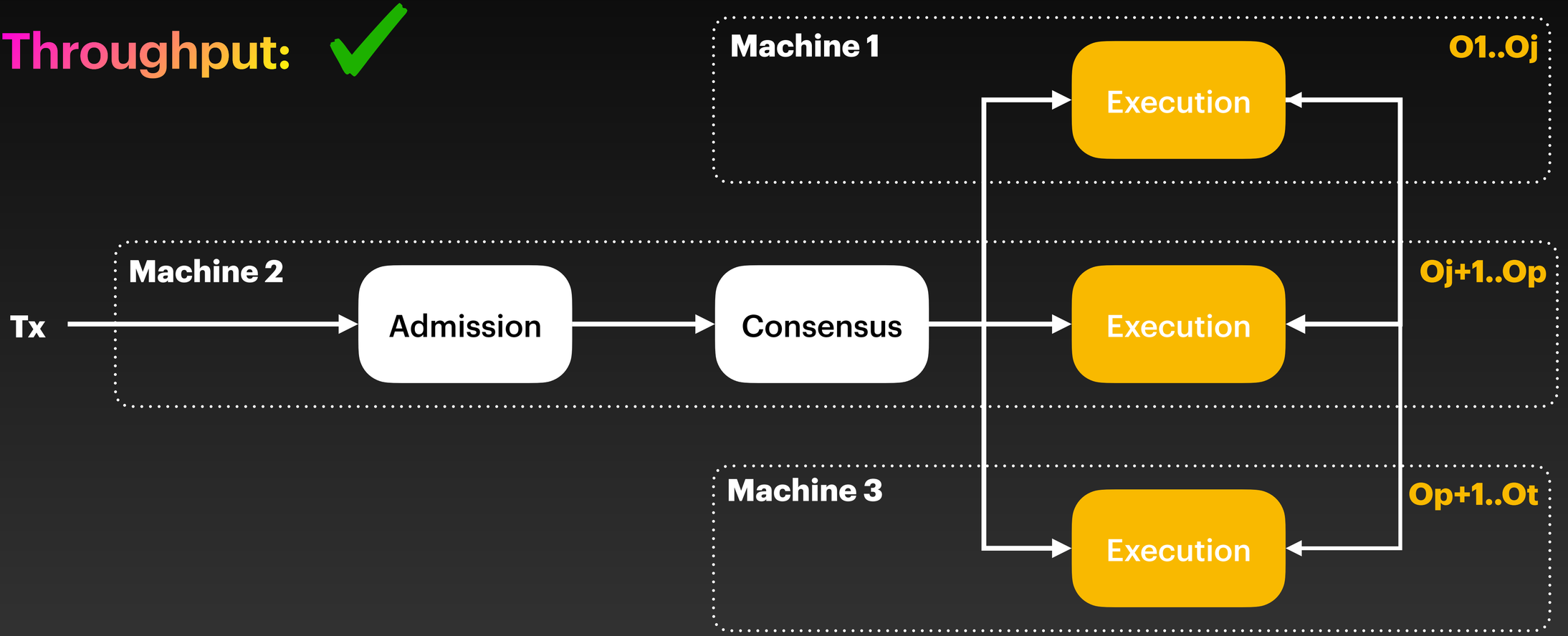

Machine 1
Execution
Machine 2
Tx
Admission
Consensus
Execution
Machine 3
Execution

Pilotfish


Machine 1
O1..Oj
Execution
Machine 2
Oj+1..Op
Tx
Admission
Consensus
Execution
Machine 3
Op+1..Ot
Execution

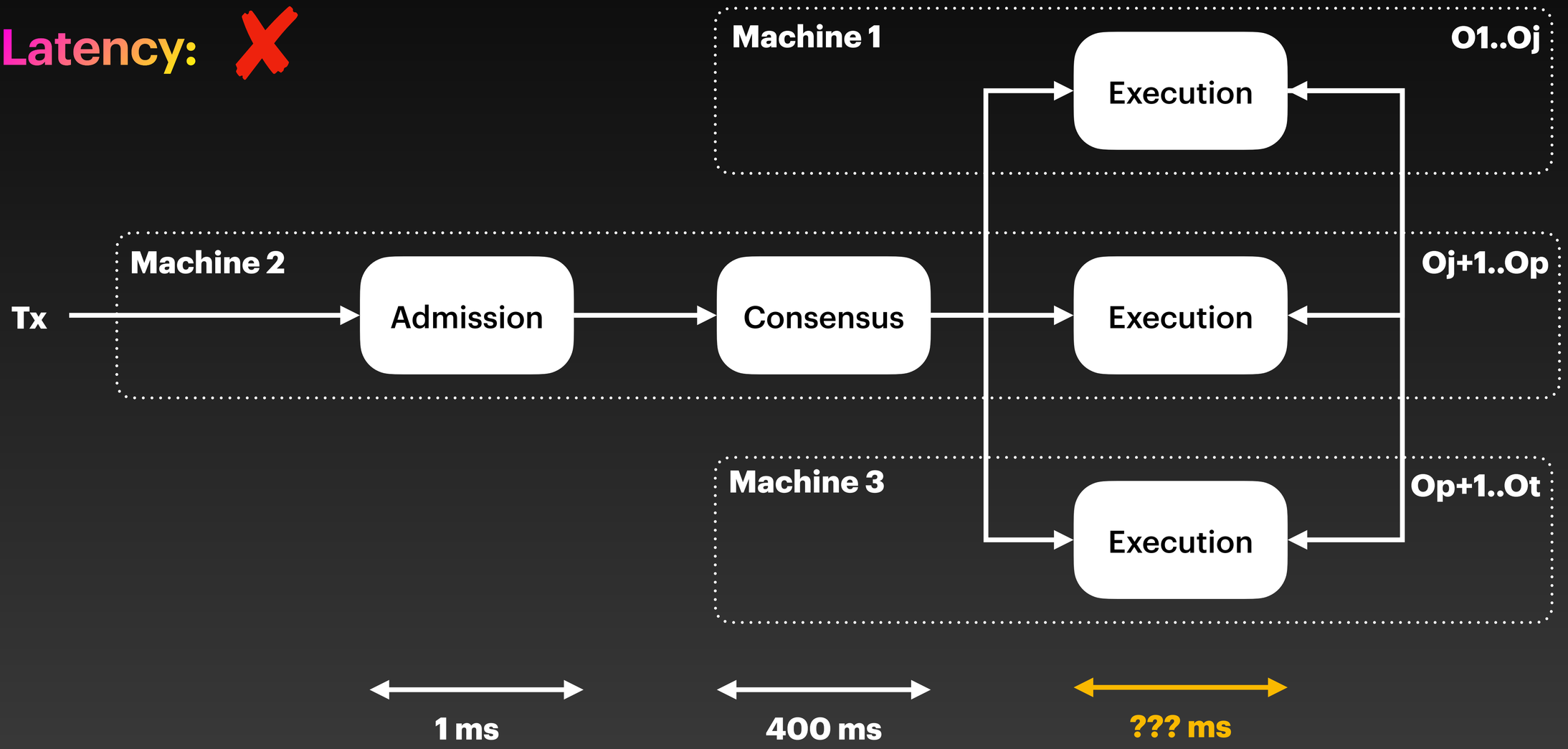
Pilotfish

Throughput: ✓



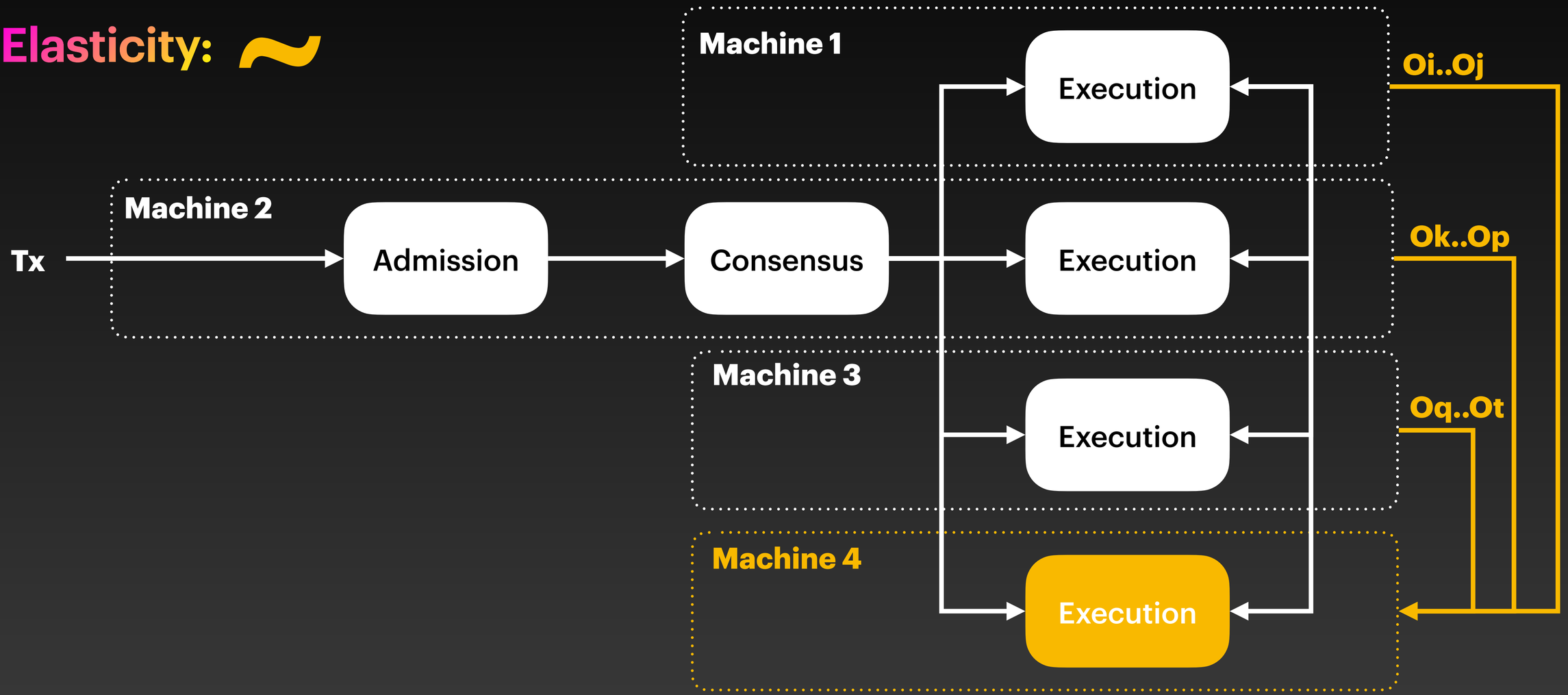
Pilotfish

Latency: ✗

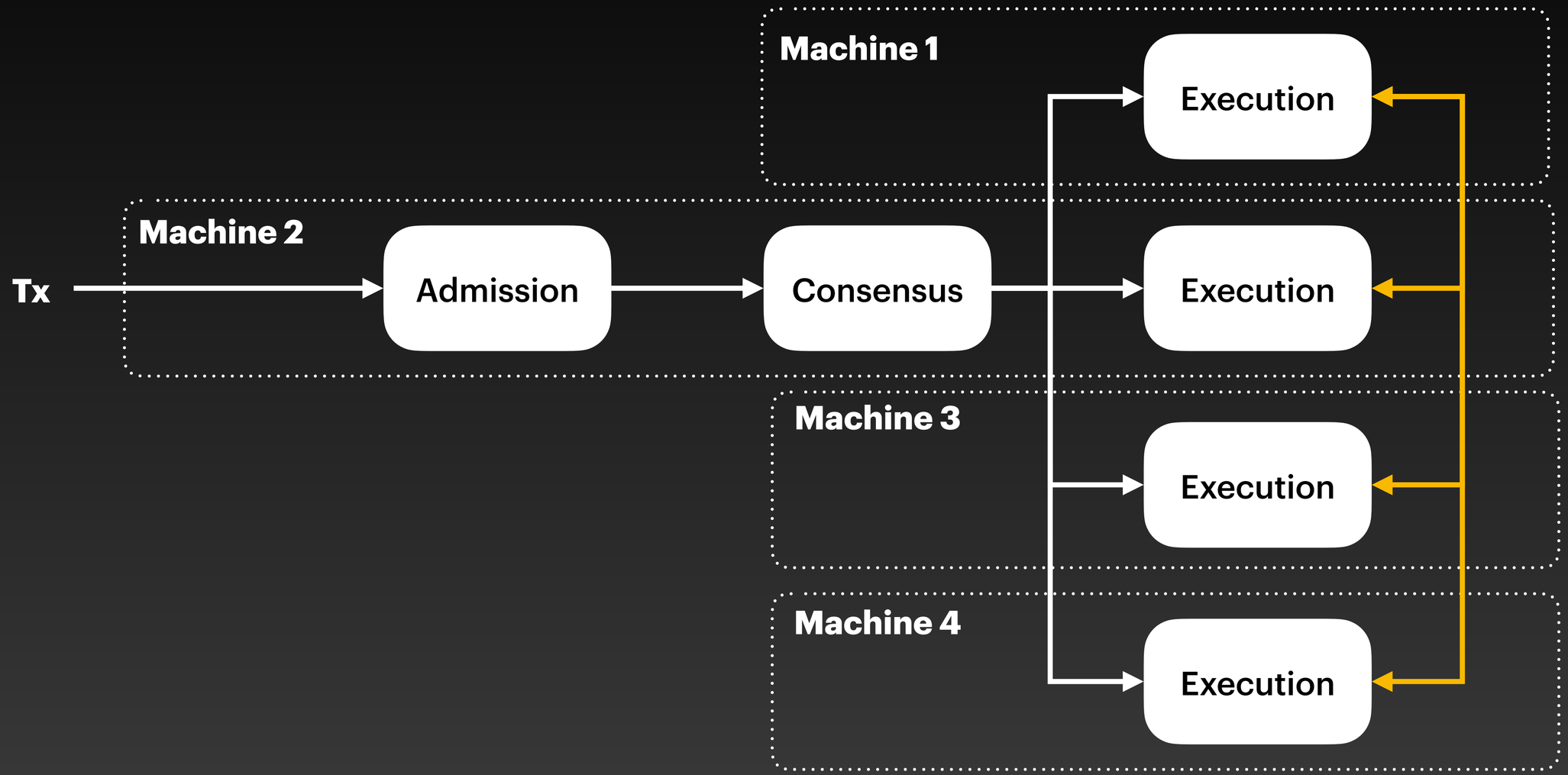


Pilotfish

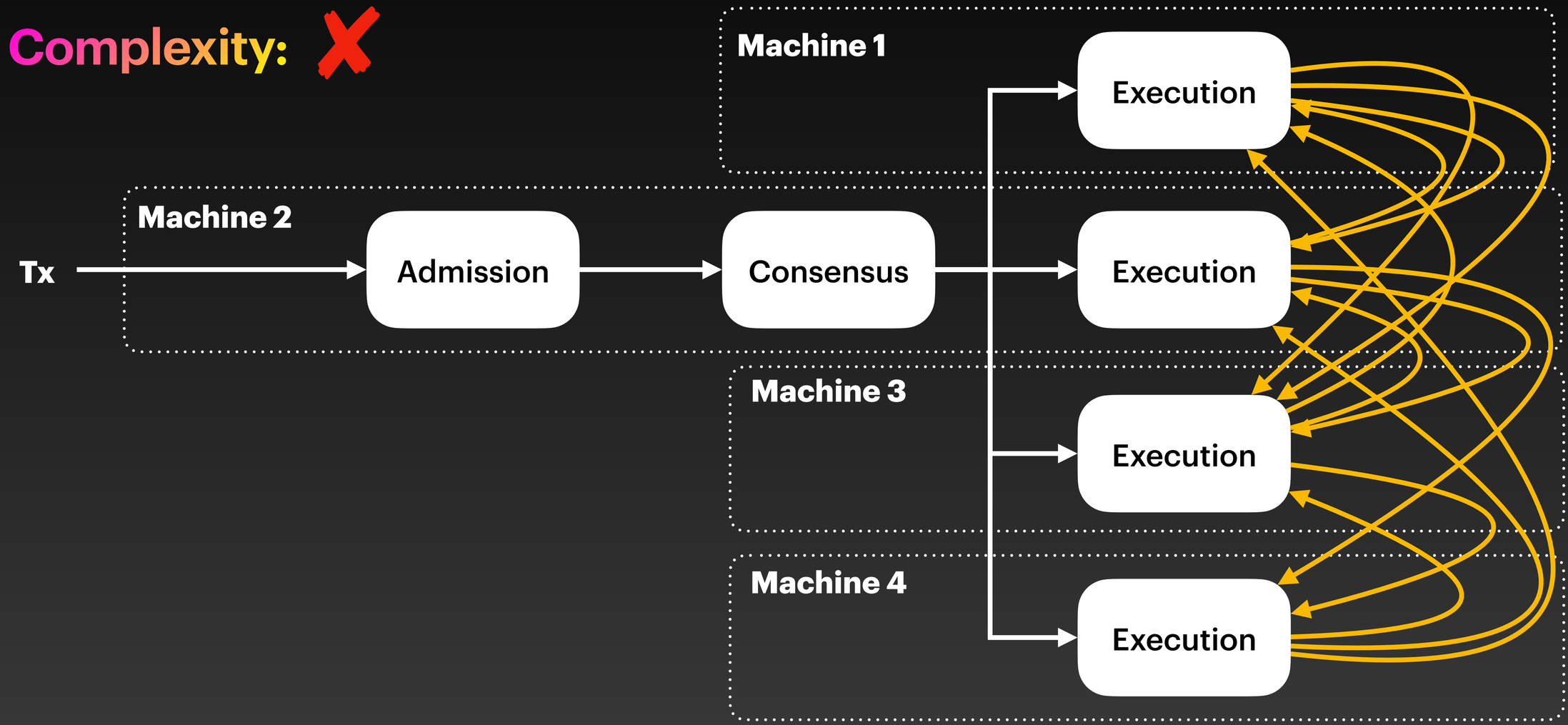
Elasticity: ~



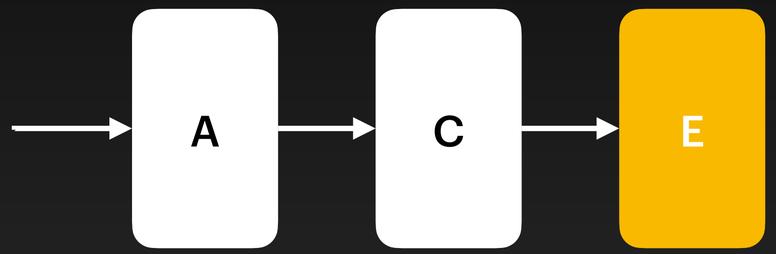
Pilotfish


Machine 1
Execution
Machine 2
Tx
Admission
Consensus
Execution
Machine 3
Execution
Machine 4
Execution

Pilotfish

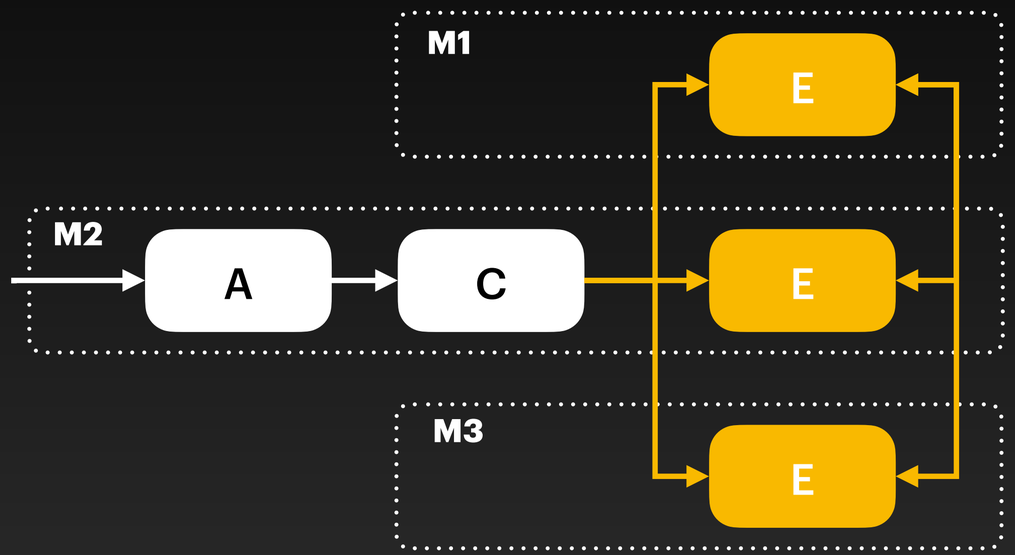

Complexity:
Machine 1
Machine 2
Tx
Admission
Consensus
Execution
Execution
Machine 3
Execution
Machine 4
Execution

State of the Art



Throughput: ✗

Latency: ✗



Throughput: ✓

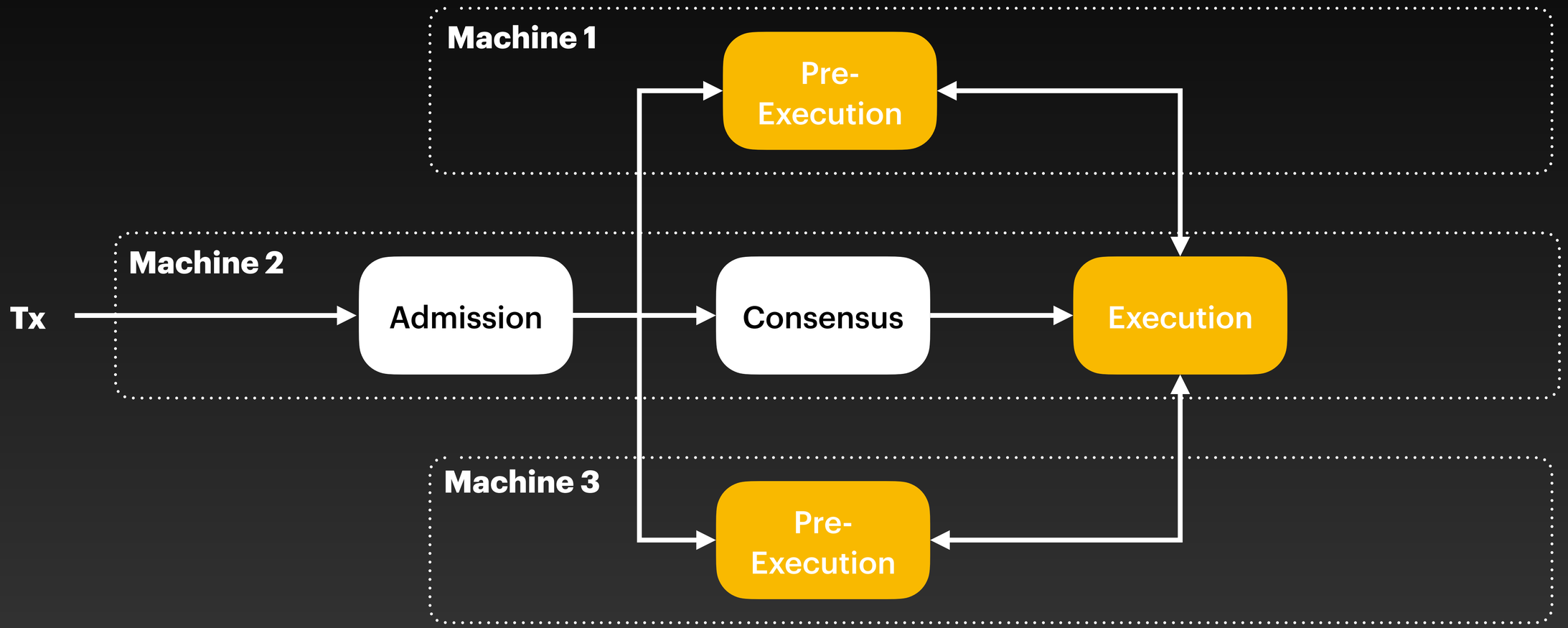
Latency: ✗

Elasticity: ~

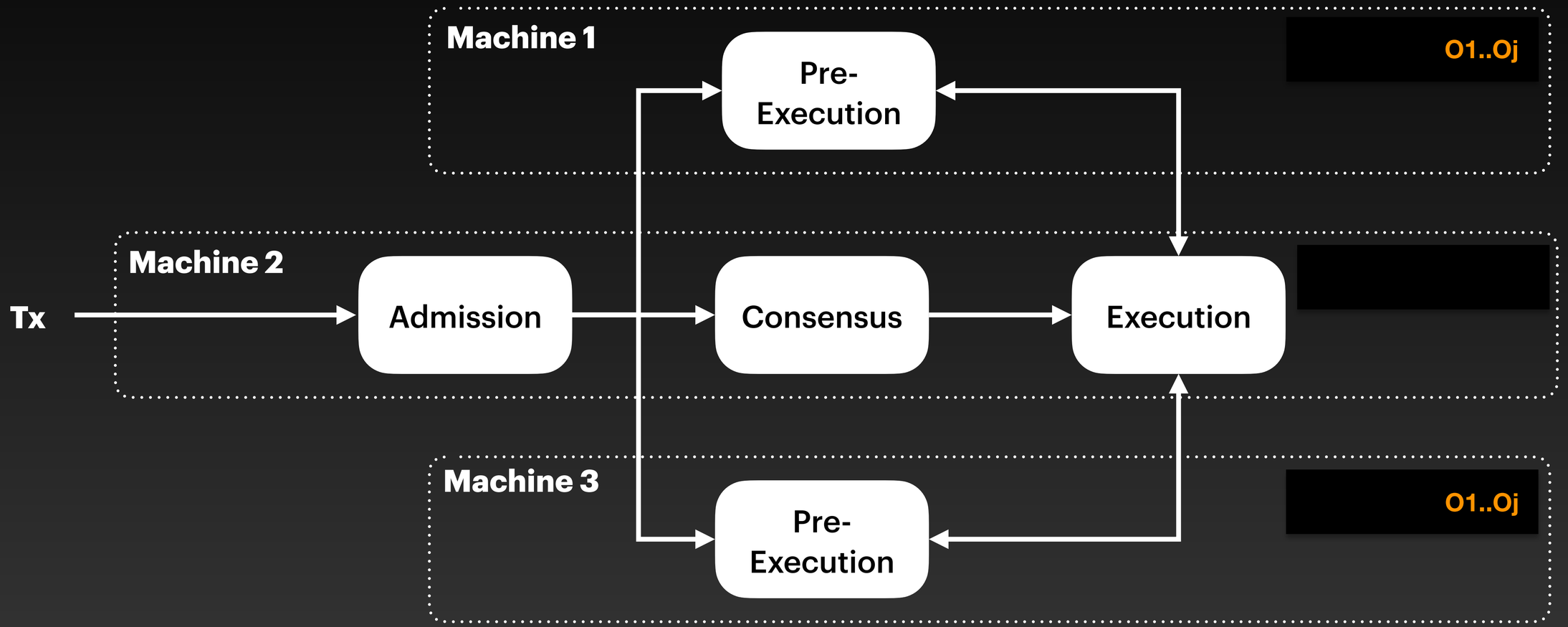
Complexity: ✗

Remora

Remora

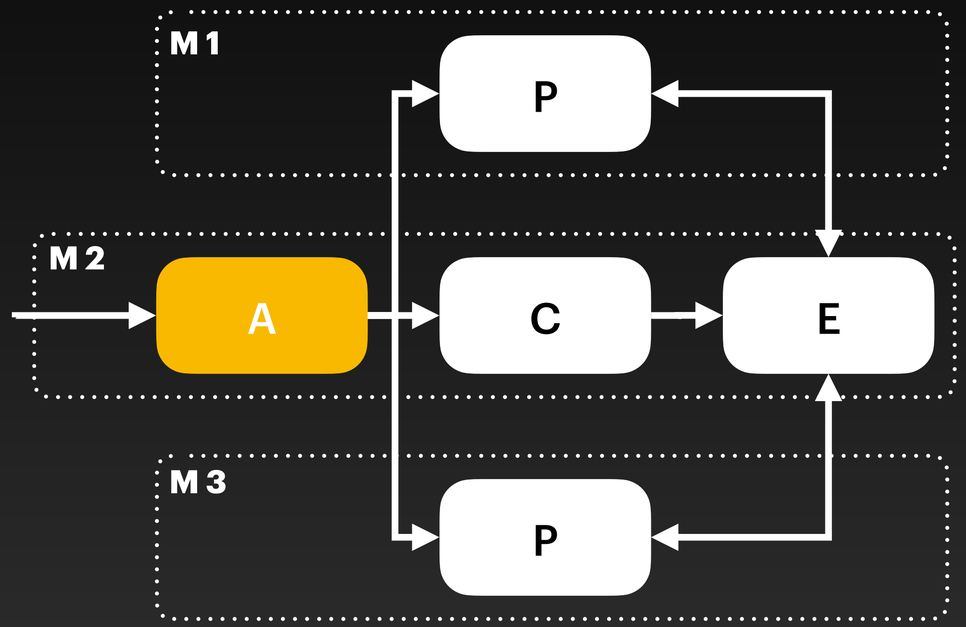

Machine 1
Pre-
Execution
Machine 2
Tx
Admission
Consensus
Execution
Machine 3
Pre-
Execution

State Replication



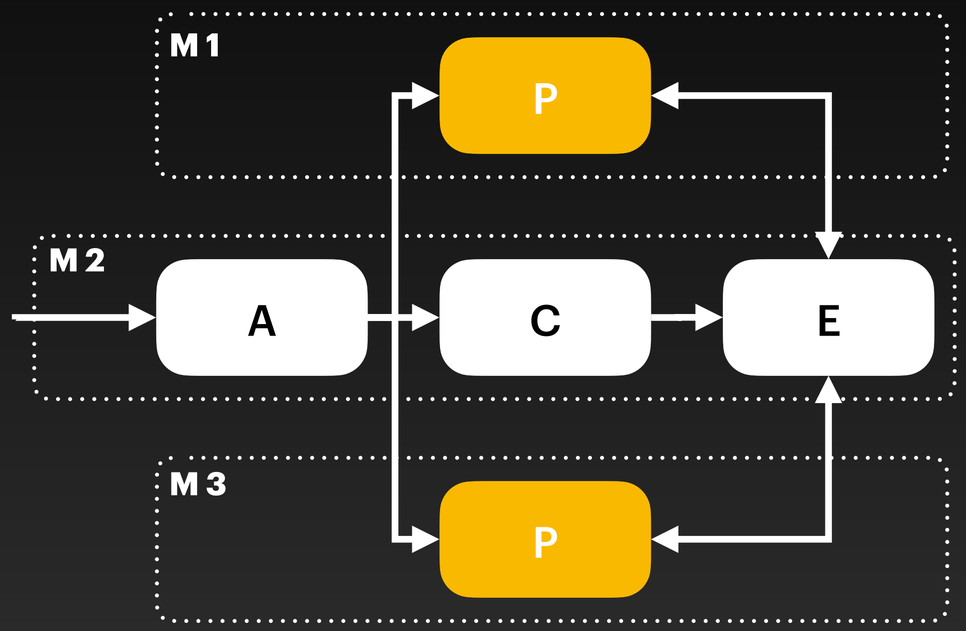
Admission

- Forward the transaction to
  - One pre-executor
  - Consensus



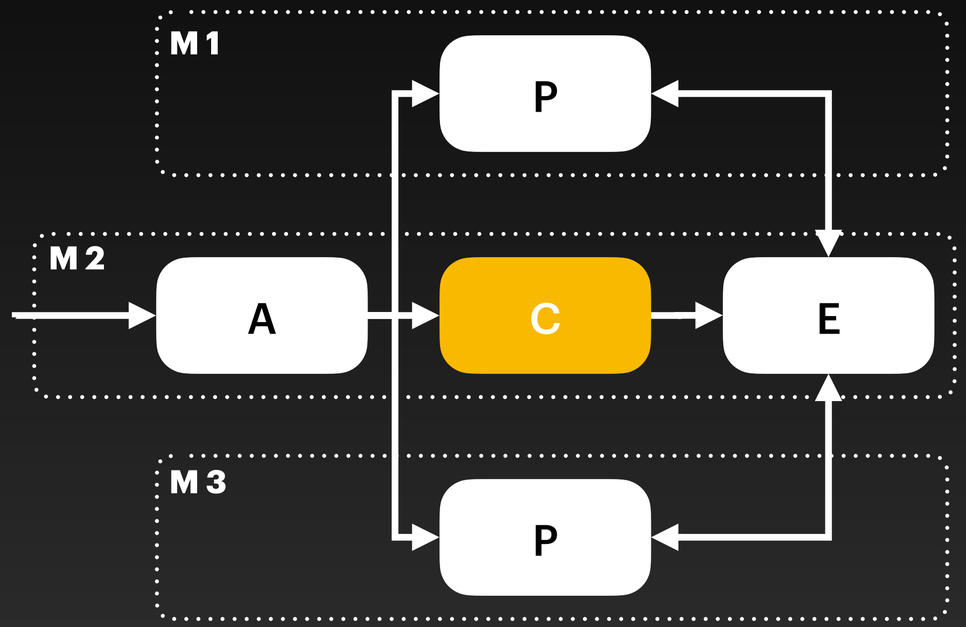
Pre-Executors

- Execute transactions
- Compute lookup table with:
  - Calls to pre-compiled functions
  - Status of cryptographic checks
- Forward results and table to primary executor
- Lazy update objects state



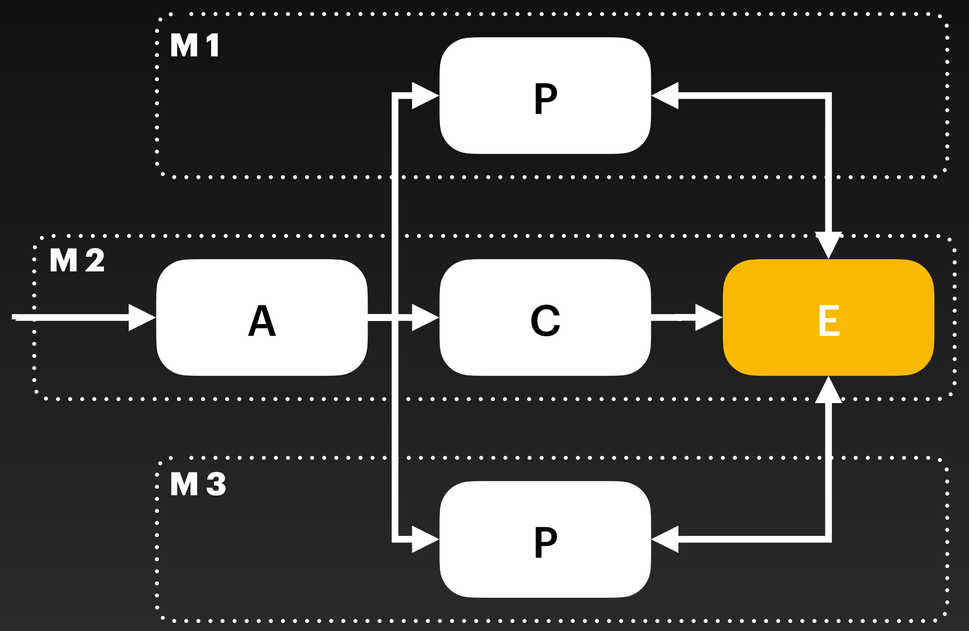
Consensus

- Sequence transactions (as usual)
- Forward commits to primary executor



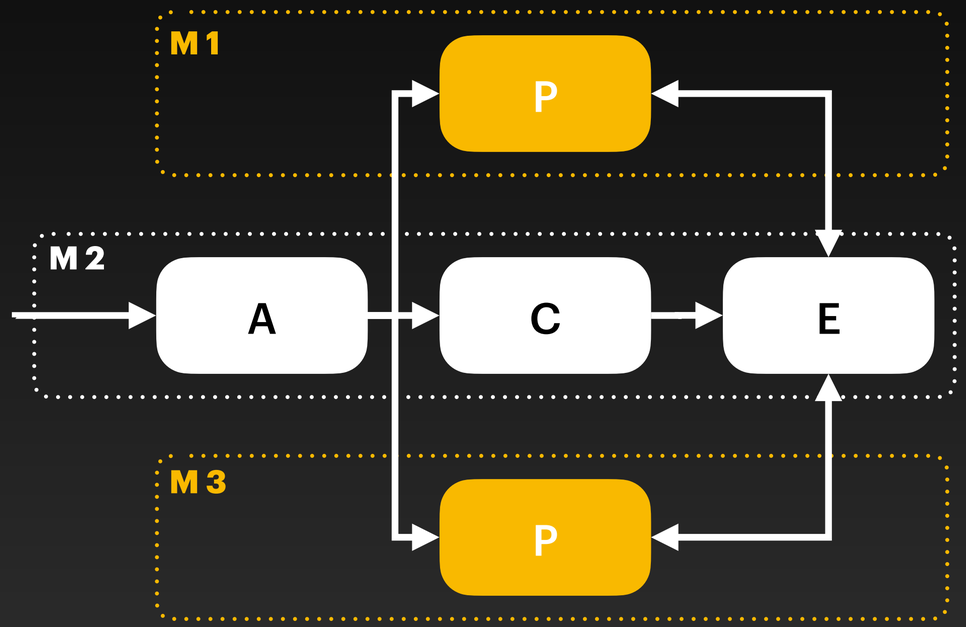
Primary Executor

- Merge pre-executor results
- If conflicts, re-execute using lookup table
- Feedback to pre-executor (avoid hot objects)



Properties

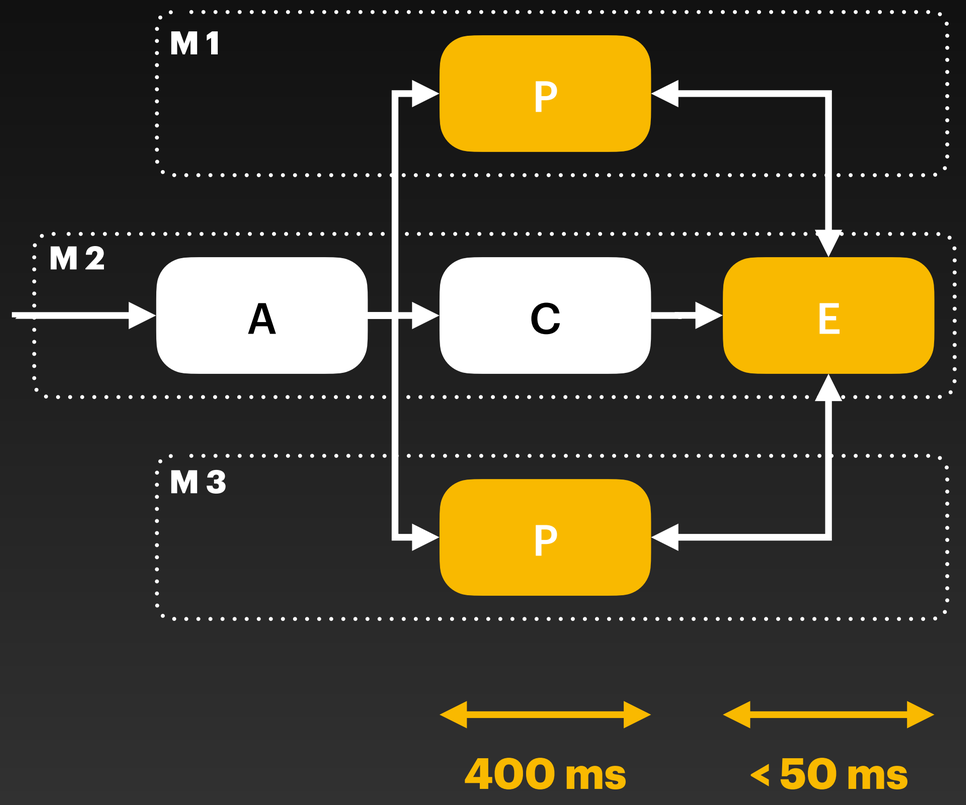
Throughput: ✓



Properties

Throughput: ✓

Latency: ✓

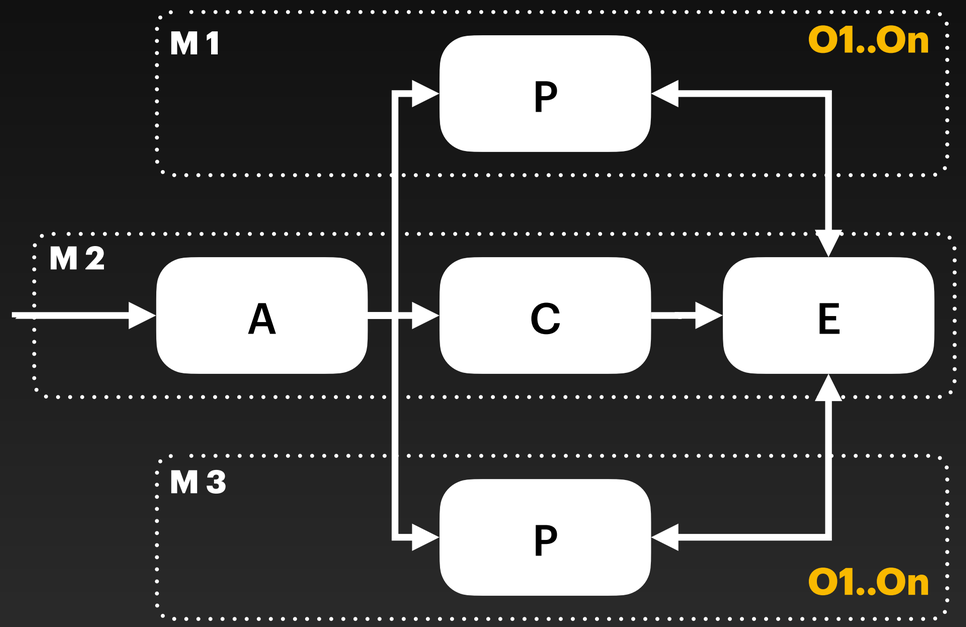


Properties

Throughput: ✓

Latency: ✓

Elasticity: ✓



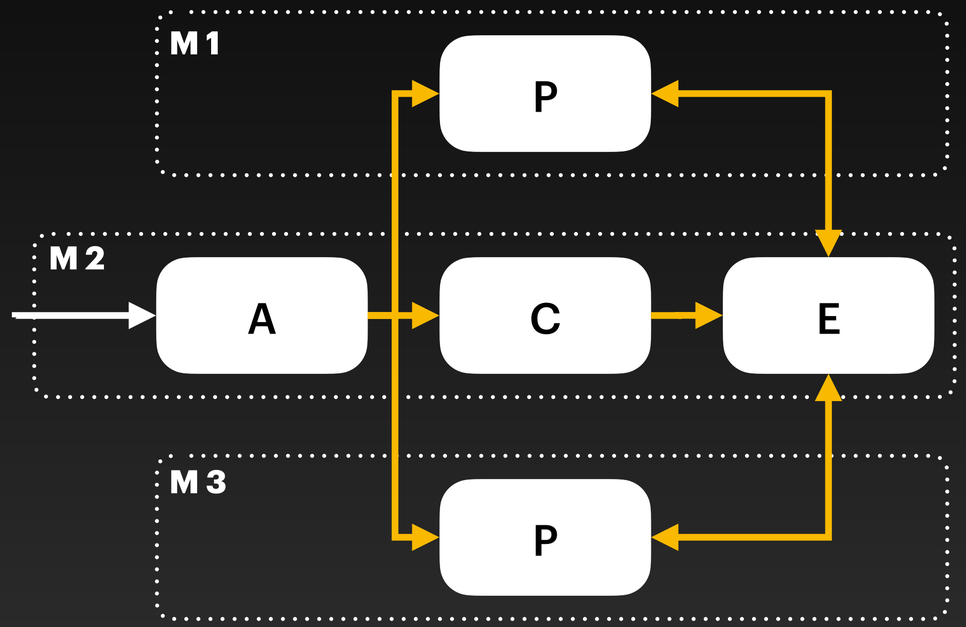
Properties

Throughput: ✓

Latency: ✓

Elasticity: ✓

Complexity: ✓



Hard Questions

Pre-Executor Selection

- Every pre-executor has the entire state
- Forward transaction to executor based on
  - Total load on the pre-executor
  - Target each pre-executor with a subset of the state (best effort)
- Eventually each pre-executor will have a subset of the state in memory / cache

State Update

- Primary executor keeps stats of pre-execution misses
- Push state update when misses exceed a threshold

Multi-Core Execution

Merge Operation

- Adopt pre-execution if lookup table contains versioned inputs
- Skip authenticators verified by pre-executors
- Read from lookup table all dynamic objects
- Adopt from lookup table results to pre-compiled functions if in lookup table

Overwhelmed Primary

- Select a subset of the state that is problematic
- Select a free pre-executor
  - Give a read lock over that subset of the state to the pre-executor
  - Forward all transactions to that pre-executor
- Upon a single (TBD), get back to normal operations