Redesigning the ordering layer of Hyperledger Fabric with Trusted Execution Environments

Nicolas Strobl
Advisor: Dimitra Giantsidi
Chair of Decentralized Systems Engineering
https://dse.in.tum.de/

15th of May – 15th of September

Motivation

Blockchains

- distributed consensus and data storage
- data is permanently committed
- capable of making progress despite malicious nodes

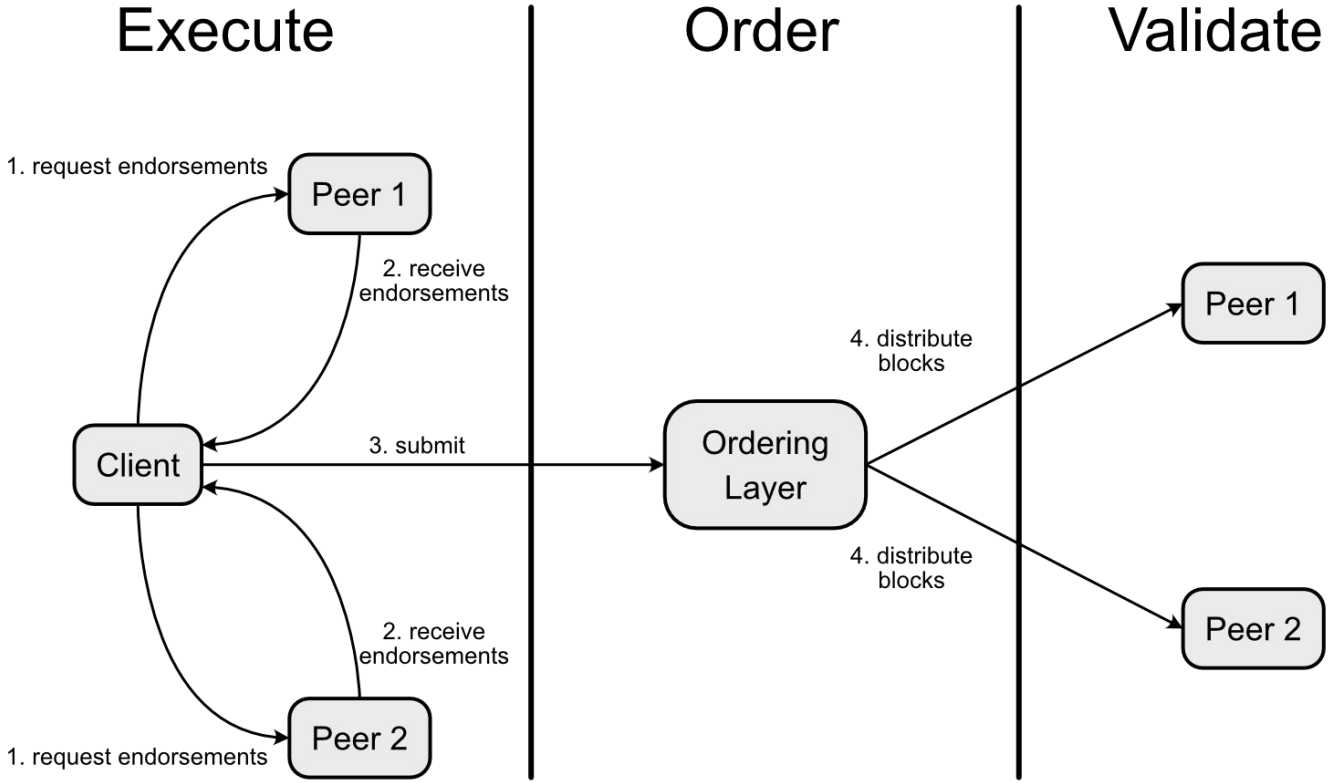
Tool to establish trust to a network of untrusted machines

Trusted Execution Environments

- hardware-based isolated execution
- not accessible or modifiable by the host system
- remote attestation: parties can verify the executed code

Tool to establish trust to an untrusted machine

Hyperledger Fabric



System: Orceval

- Design an ordering layer for Hyperledger Fabric
  - similar guarantees to BFT ordering layers
  - comparable throughput
  - combining Order and Validate phases

Outline

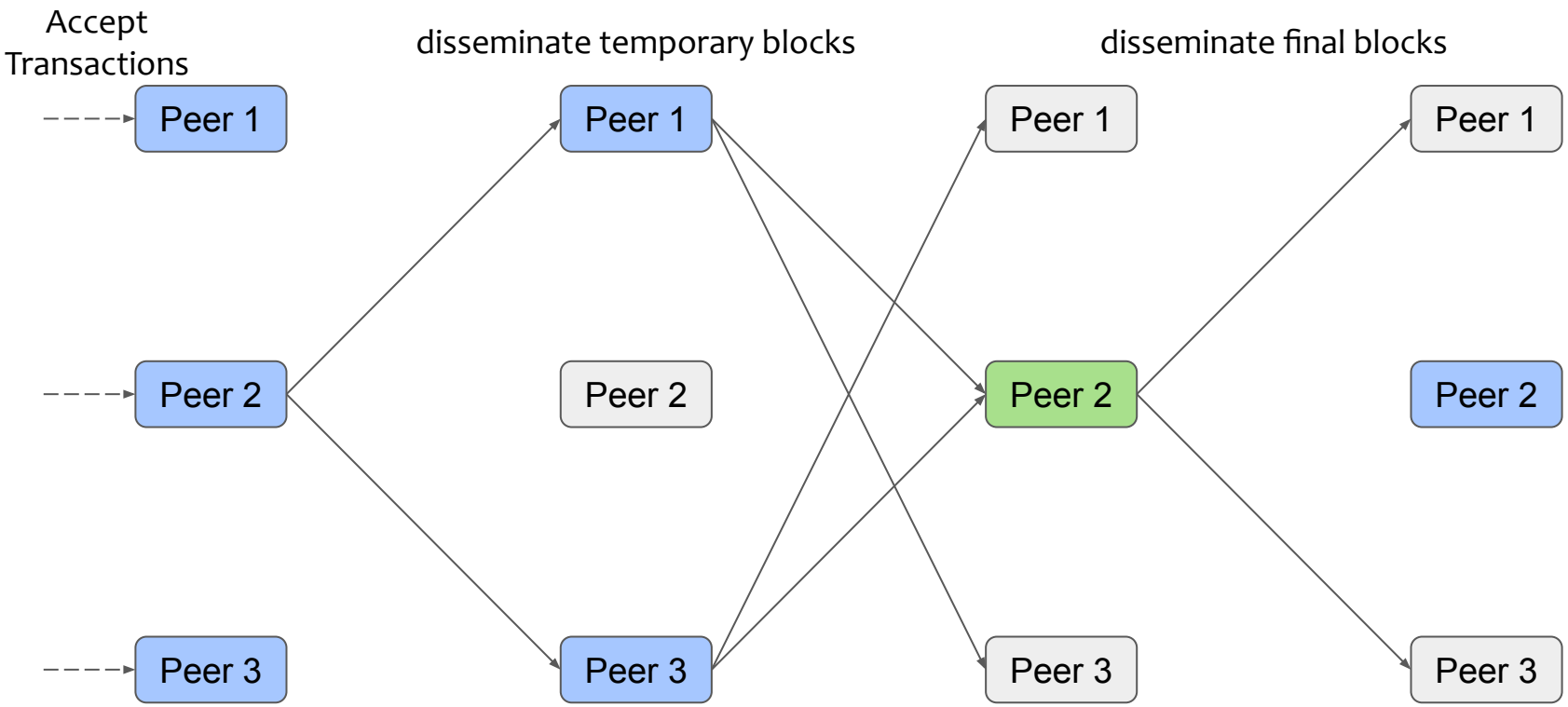
- ~~Motivation~~
- Design
  - Design overview
  - Message pattern
- Implementation
- Evaluation

Design overview

- **every peer** accepts new transactions
- validates endorsement policy
- independently validates read and write sets (from its perspective)

- **one peer** resolves read and write conflicts between blocks
- peer for a given round is deterministically chosen

Message pattern



Design details

- crashed ordering peers are detected with missing TCP acknowledgements or timeouts
  - dedicated coordination messages are not required in the regular workflow
  - A single active ordering peer can make progress
- Ordering and validation is executed in TEEs
  - any machine can use remote attestation to verify code execution

➔ Endorsement peers do not run validation

Outline

- ~~Motivation~~
- ~~Design~~
- Implementation
- Evaluation

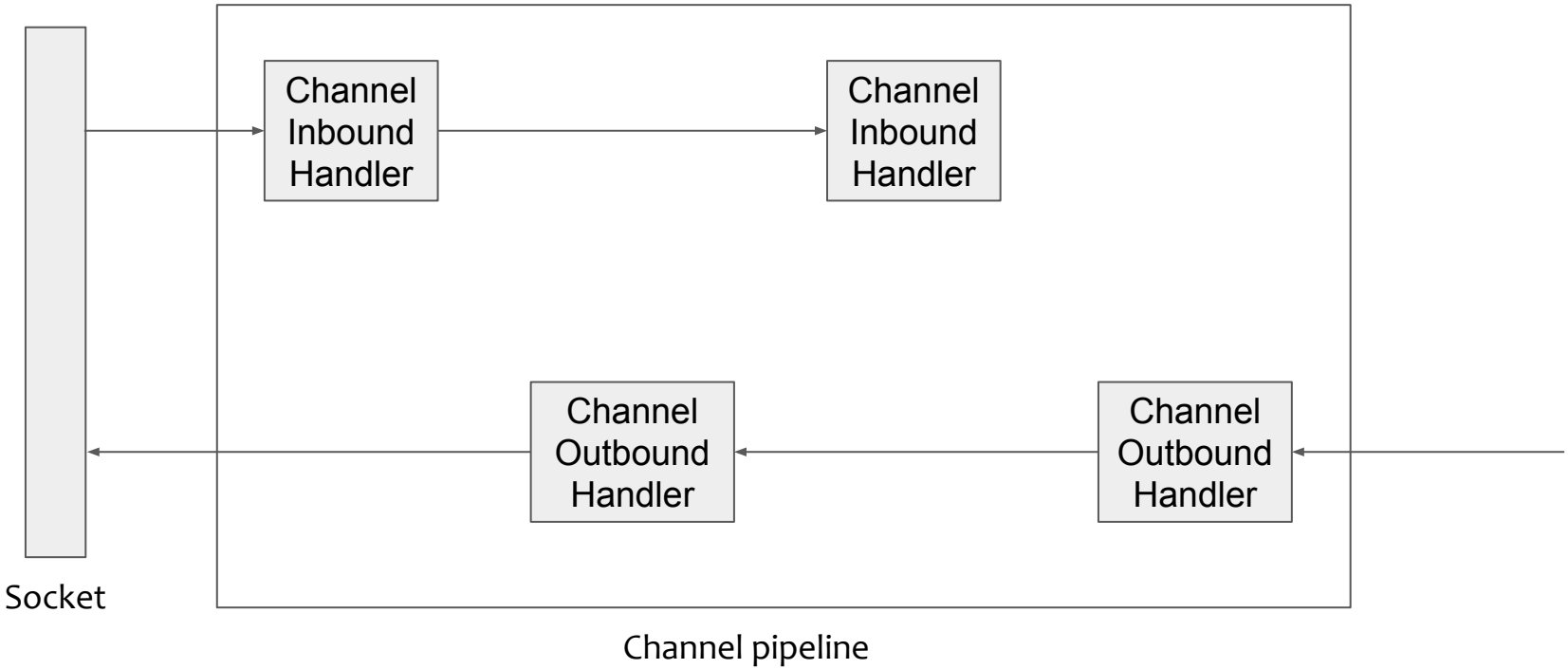
Implementation

Jackson API[1]: Java API for JSON serialization and deserialization

Netty[2]: Java library for non-blocking networking using channels and pipelines

[1]Jackson Project: https://github.com/FasterXML/jackson
[2]Netty Project: https://netty.io/

Implementation



Outline

- ~~Motivation~~
- ~~Design~~
- ~~Implementation~~
- Evaluation

Evaluation

- Experimental setup (Cloudlab):
  - 2x Intel Xeon Silver 4114 CPU (2.20 GHz base, 10 cores, 20 threads)
  - 196608 MB memory
  - 10 Gbps network bandwidth
- Systems:
  - Orceval
  - BFT-SMaRt[1]

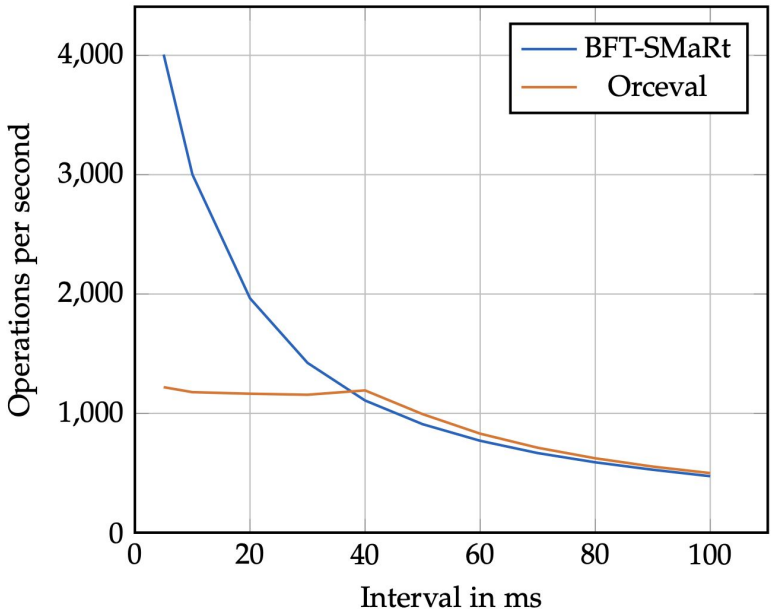
[1]BFT-SMaRt: https://github.com/bft-smart/library

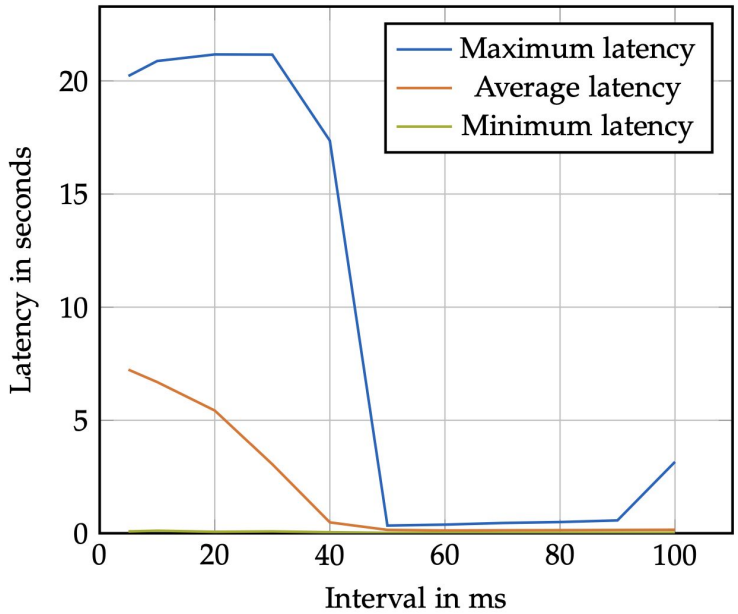
Evaluation

Defaults

- 4 peers
- 50 clients each submitting 500 transactions/operations at an interval of 10 ms
- each operation both reads and writes
- Orceval: 25 transactions per block

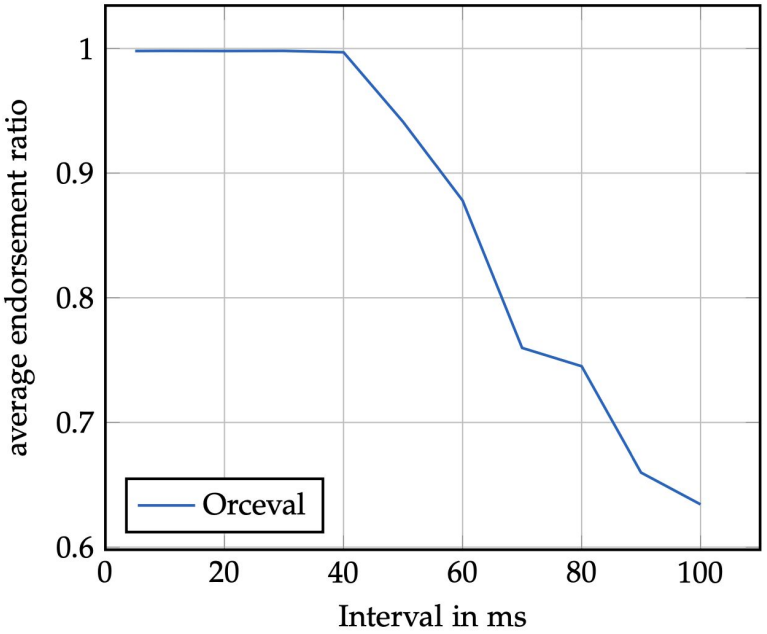
Throughput and Latency





Orceval has similar or better throughput but also a bottleneck

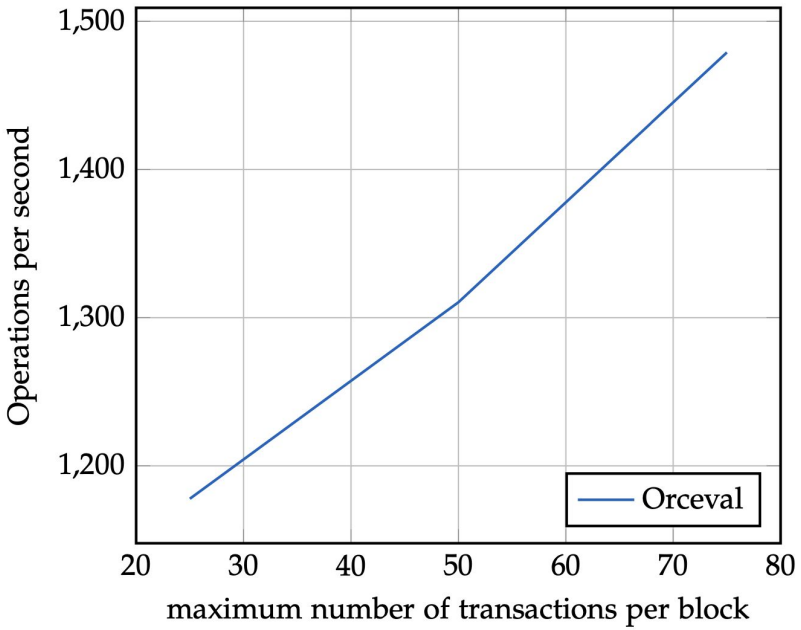
Bottleneck



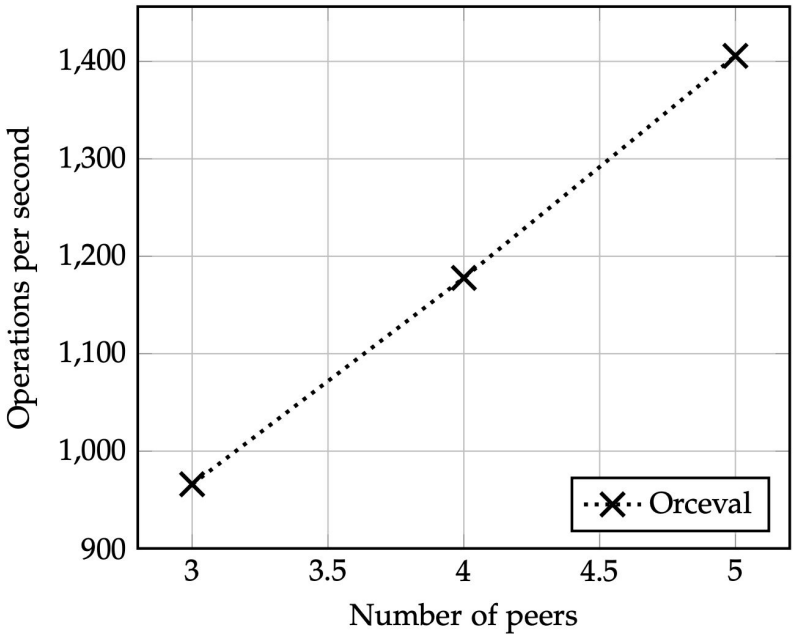
Average ratio of time in endorsement validation divided by time in Per Block Validation

Computation of the endorsement validation is the bottleneck

Improving bottleneck performance



Larger blocks increase parallelization



Performance scales well regarding the number of peers

Summary

Orceval

- similar guarantees to BFT ordering layers
- similar throughput to BFT ordering layers despite additionally performing validation
- great scalability as opposed to the adverse scalability of traditional BFT systems