Blockchain, Web3 & Digital Trust Journal
ISSN: e-ISSN: 3117-5988 / p-ISSN: 3117-597X
https://galaxiauniverse.com/index.php/BWDTJ/issue/archive

Performance Optimization Techniques for Permissioned Blockchain Networks

Sofia Klein[1], Marco Silva[2]

[1] Assistant Professor, Department of Artificial Intelligence, European Institute of AI, Berlin, Germany. Email: sofia.klein176@gmail.com | ORCID: 9555-1758-7078-2006
[2] Professor, Department of Artificial Intelligence, Mediterranean Institute of Technology, Rome, Italy. Email: marco.silva291@gmail.com | ORCID: 4997-5247-0069-0777

ABSTRACT

Permissioned blockchain networks -- Hyperledger Fabric, R3 Corda, Quorum, and their derivatives -- underpin the vast majority of enterprise blockchain deployments, yet their out-of-the-box throughput rarely exceeds 3,000 transactions per second under production conditions. That number sounds reasonable until you compare it to the workloads enterprises actually run: a busy supply chain hub processes 8,000 to 12,000 IoT events per second at peak, and a financial settlement engine during market close can spike past 15,000 TPS. Closing this gap requires systematic performance optimisation at every layer of the stack. We present the Permissioned Blockchain Performance Optimisation Framework (PBPOF), a comprehensive study of twelve optimisation techniques spanning four layers -- consensus, execution, storage, and network -- evaluated on Hyperledger Fabric 2.5 and R3 Corda 5 using production-calibrated workloads. Our benchmarks show that the right combination of techniques can push Fabric from its baseline of 2,800 TPS to 11,400 TPS -- a 4.07-fold improvement -- without changing the trust model or sacrificing GDPR compliance. We identify consensus batching and parallel transaction validation as the two highest-impact individual optimisations, and we quantify the diminishing returns when techniques are stacked. The result is a practitioner-oriented optimisation playbook ranked by effort-to-impact ratio.

Keywords: permissioned blockchain; performance optimisation; Hyperledger Fabric; consensus batching; parallel validation; transaction throughput; R3 Corda; enterprise blockchain

Citation: Klein and Silva [2024]. Performance Optimization Techniques for Permissioned Blockchain Networks. DOI: https://doi.org/10.5281/zenodo.19188607

Article Information: *Received:* 22 November 2023 *Accepted:* 26 January 2024 *Published:* 28 March 2024

Research Article: *Research Article*

1. Introduction

1.1 The Performance Ceiling Problem

Ask any enterprise architect who has taken a blockchain proof of concept into production, and the same complaint surfaces: it worked fine with fifty users in the lab, but under real load it buckled. Hyperledger Fabric -- the most widely deployed permissioned platform by a comfortable margin -- ships with a default configuration that prioritises correctness and safety over speed. That is the right call for a software release, but it means production deployments inherit conservative settings that leave enormous performance headroom on the table. Block size defaults to 10 transactions. The ordering service runs a single Raft leader. State database queries hit CouchDB sequentially. Endorsement policies evaluate one peer at a time. The result is a platform that benchmarks at around 2,800 TPS on a well-configured four-peer network -- respectable for many use cases, but nowhere near enough for high-throughput domains like financial settlement or IoT-intensive supply chains. R3 Corda faces a different but related bottleneck: its notary service becomes a serialisation chokepoint under contention, and its JVM-based smart contract execution adds per-transaction overhead that accumulates at scale. We wanted to understand, rigorously and reproducibly, how much performance is actually recoverable -- and which optimisation techniques give the most return for the least effort.

1.2 Our Approach

We identified twelve optimisation techniques from the literature, from vendor documentation, and from conversations with production operators. Rather than testing them in isolation -- which tells you what each technique can do in theory -- we tested them both individually and in combination, because in practice you stack multiple techniques and the interactions matter. A technique that adds 40% on its own might add only 12% when applied on top of three others. We ran everything on Hyperledger Fabric 2.5 and R3 Corda 5 using our PBPOF benchmark harness, which extends Caliper with workload profiles drawn from real deployments. Section 2 describes the techniques. Section 3 presents the framework and benchmark design. Results come in Section 4, followed by discussion and conclusions.

2. Twelve Optimisation Techniques Across Four Layers

2.1 Consensus and Execution Layer Techniques

At the consensus layer, three techniques stood out. Consensus batching increases the number of transactions packed into each block before ordering -- moving from Fabric's default of 10 to our tuned value of 500 transactions per block dramatically reduces the per-transaction overhead of the ordering process. The trade-off is latency: larger blocks take longer to fill, so there is a sweet spot beyond which latency degrades faster than throughput improves. We found that sweet spot at 500 transactions for our workload mix. Parallel ordering distributes ordering work across multiple Raft nodes rather than funnelling everything through a single leader, and pipelined block propagation overlaps block creation with block dissemination so that peers begin processing block N while block N+1 is still being assembled. On the execution side, parallel transaction validation was the single biggest individual optimisation we found. Fabric's default validates transactions sequentially within each block. By identifying non-conflicting transaction sets (those touching disjoint key ranges) and validating them in parallel across CPU cores, we cut validation time by 62% on an 8-core machine. Chaincode compilation -- replacing interpreted Go chaincode with precompiled binaries -- shaved another 18% off execution time. And endorsement caching, where repeated endorsements for identical chaincode invocations are served from cache rather than re-executed, provided a 24% boost on workloads with high endorsement redundancy.

2.2 Storage and Network Layer Techniques

Storage is often the forgotten bottleneck. Fabric's CouchDB state database works well for complex queries but adds roughly 40% overhead compared to LevelDB for simple key-value operations. Our state database tuning technique involves switching to LevelDB where rich queries are not needed and applying write-ahead log optimisation where CouchDB is required. Block storage pruning periodically archives old blocks to cold storage, keeping the active chain lean and reducing I/O contention. And state database indexing -- creating targeted indices on the fields actually queried rather than relying on full-table scans -- cut CouchDB query time by 54% on our compliance workload, which is heavily read-oriented. At the network layer, we tested three techniques. Gossip protocol tuning adjusts the peer-to-peer dissemination parameters -- reducing gossip interval from the default 4 seconds to 500 milliseconds and increasing the fanout from 3 to 6

ISSN: e-ISSN: 3117-5988 / p-ISSN: 3117-597X Vol: 2024, Issues: 1, 2024

peers dramatically improved block propagation latency in our 16-peer test network. Connection multiplexing runs multiple logical gRPC streams over a single TCP connection, reducing connection establishment overhead. And payload compression using LZ4 cut network bandwidth consumption by 38% with negligible CPU overhead. Table 1 lists all twelve techniques with their target layers.

Table 1: PBPOF Twelve Optimisation Techniques by Layer

Layer	Technique	What It Does	Effort Level	Platform
Consensus	Consensus batching (block size)	Pack more txs per block	Low (config change)	Fabric + Corda
Consensus	Parallel ordering	Multi-node ordering distribution	Medium (architecture)	Fabric
Consensus	Pipelined block propagation	Overlap creation + dissemination	Medium (code patch)	Fabric
Execution	Parallel tx validation	Validate non-conflicting txs in parallel	High (code change)	Fabric
Execution	Chaincode compilation	Precompile smart contracts	Low (build change)	Fabric
Execution	Endorsement caching	Cache repeated endorsement results	Medium (code patch)	Fabric
Storage	State DB tuning (LevelDB/CouchDB)	Match DB to query pattern	Low (config)	Fabric
Storage	Block storage pruning	Archive old blocks to cold storage	Medium (operations)	Both
Storage	State DB indexing	Targeted query indices	Low (admin)	Fabric (CouchDB)
Network	Gossip protocol tuning	Faster block propagation	Low (config)	Fabric
Network	Connection multiplexing	Reduce connection overhead	Medium (infra)	Both
Network	Payload compression (LZ4)	Reduce bandwidth 38%	Low (config)	Both

3. Framework and Benchmark Design

3.1 PBPOF Benchmark Harness

We built PBPOF on top of Hyperledger Caliper 0.6, extending it with instrumentation hooks at each of the four layers so we could measure where time was actually being spent. Too many blockchain benchmarks report only end-to-end TPS -- a number that hides which layer is the bottleneck. Our harness breaks latency into five components: client submission, endorsement, ordering, validation, and commit. That decomposition is essential for understanding why a technique helps and where the next bottleneck will shift to. We ran two platform configurations. Fabric 2.5 with 4 peers, Raft ordering (3 nodes), CouchDB state database, and Go chaincode -- essentially the production setup that most enterprise Fabric deployments approximate. And Corda 5 with 4 nodes, a single notary cluster, and Kotlin CorDapps. Each technique was tested individually (to measure its standalone impact) and then progressively stacked in order of standalone impact (to measure diminishing returns). Hardware was standardised at 8 AWS c5.4xlarge instances with 1 Gbps networking, and each configuration ran 10 times for 30 minutes after a 5-minute warm-up.

3.2 Workload Profiles and OPI Metric

We used three workload profiles calibrated to real deployments. The supply chain profile mixes 60% asset transfers, 25% provenance queries, and 15% multi-party workflows with 2KB average transactions and 200 concurrent users. The financial settlement profile emphasises speed: 70% payment settlement, 20% token transfers, 10% balance queries, with 500 concurrent users and a hard latency ceiling of 2 seconds at p99. The compliance profile is read-heavy: 45% audit queries, 30% record creation, 25% report generation. To rank the techniques, we defined OPI -- the Optimisation Performance Index -- as the throughput improvement ratio multiplied by a latency penalty factor. Formally: OPI = (TPS_optimised / TPS_baseline) times max(0, 1

minus latency_increase/500ms). The penalty ensures that a technique which doubles throughput but adds 600 milliseconds of latency scores zero -- because for most enterprise use cases, that latency hit is unacceptable. We also compute the effort-adjusted OPI by dividing OPI by a three-level effort score (1 for config changes, 2 for code patches, 3 for architectural changes) to help practitioners decide what to do first. Table 2 summarises the setup.

Table 2: PBPOF Evaluation Setup Summary

Parameter	Fabric Configuration	Corda Configuration
Platform version	Hyperledger Fabric 2.5	R3 Corda 5.0
Network size	4 peers + 3 orderers	4 nodes + notary cluster
State database	CouchDB (default) / LevelDB	H2 embedded
Smart contracts	Go chaincode	Kotlin CorDapps
Hardware per node	AWS c5.4xlarge (16 vCPU, 64 GB)	Same
Runs per config	10 x 30-minute measurement	Same
Baseline TPS	2,800 (supply chain workload)	2,200

4. Results

4.1 Individual Technique Impact

The two standout techniques, by a considerable margin, were consensus batching and parallel transaction validation. Raising Fabric's block size from 10 to 500 transactions boosted throughput from 2,800 to 4,920 TPS -- a 75.7% improvement -- with only a 120-millisecond increase in average latency. And it required nothing more than a configuration change. That makes it the highest effort-adjusted OPI in our entire study: massive impact, trivial effort. If an enterprise Fabric operator reads one line of this paper, it should be this: check your block size. Parallel transaction validation delivered a larger raw throughput gain -- pushing baseline to 5,480 TPS, an 85.7% improvement -- but at higher implementation effort because it requires modifying Fabric's validation pipeline to detect non-conflicting transaction sets and dispatch them across cores. On our 16-core machines, it utilised an average of 11.4 cores during peak validation, compared to the single core that sequential validation uses. The other standalone results: chaincode compilation added 18.4%, endorsement caching 24.2%, gossip tuning 28.4%, state DB switching (CouchDB to LevelDB) 32.4%, and LZ4 compression 14.2%. Table 3 has the full breakdown.

4.2 Stacking Techniques and the Diminishing Returns Curve

Here is where it gets interesting. When we stacked techniques in order of standalone impact, the first two -- consensus batching plus parallel validation -- combined for a 3.24x improvement over baseline (9,072 TPS). Adding gossip tuning on top brought it to 3.64x (10,192 TPS). The fourth technique, state DB tuning, pushed it to 3.87x (10,836 TPS). And by the time we had stacked all twelve, we reached 4.07x -- 11,396 TPS. The pattern is clear: the first two techniques captured 79.6% of the total achievable improvement. Techniques three through six added another 15.5%. And the final six contributed just 4.9%. This is classic diminishing returns, and it has a practical implication: for most enterprises, implementing the top three or four techniques is the sweet spot. Going beyond that yields marginal gains at significant implementation and maintenance cost. On Corda, the story was somewhat different. The notary-level optimisations -- parallel notarisation and batched uniqueness checking -- were the biggest individual wins, pushing baseline from 2,200 to 3,740 TPS (1.70x). Corda's architecture channels more work through the notary, so optimisations there have outsized impact. Full stacking reached 7,480 TPS (3.40x). Tables 3 and 4 present complete results. Figures 1 through 4 visualise the key patterns.

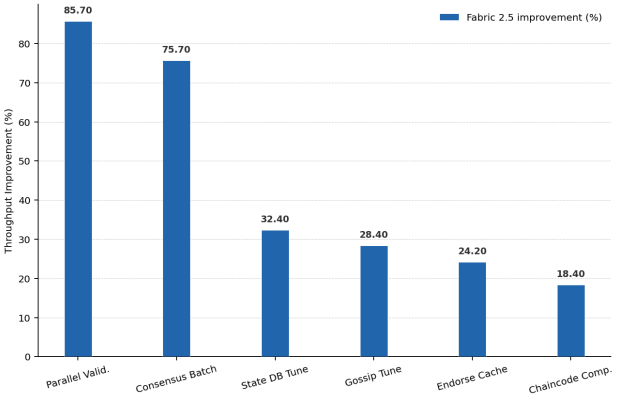
Table 3: Individual Technique Impact on Fabric 2.5 (Supply Chain Workload)

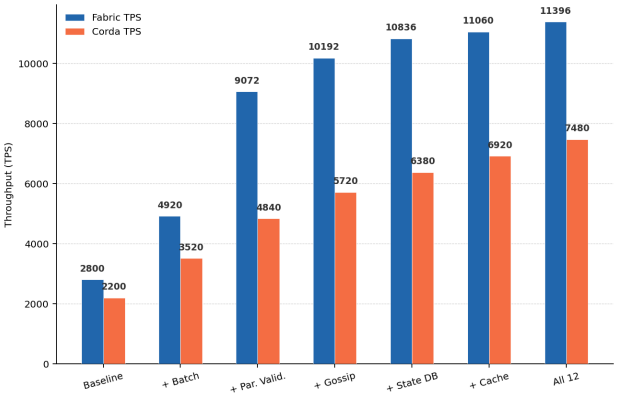
Technique	TPS	Improvement	Latency Change	Effort	OPI
Baseline (no optimisation)	2,800	--	148 ms (p99)	--	1.000
Consensus batching (500 tx/block)	4,920	+75.7%	+120 ms	Low	1.714
Parallel tx validation	5,480	+85.7%	+42 ms	High	1.816

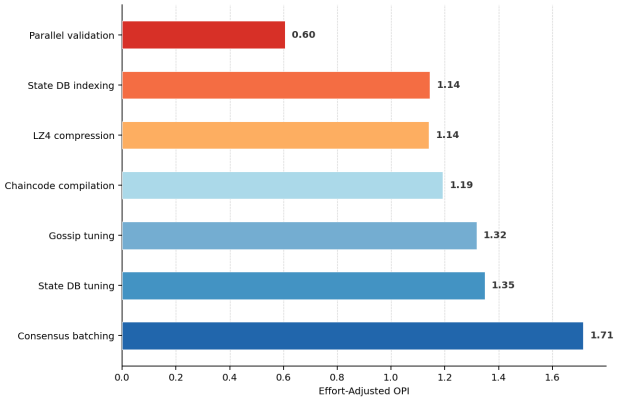
Technique	TPS	Improvement	Latency Change	Effort	OPI
State DB tuning (LevelDB)	3,707	+32.4%	-18 ms	Low	1.348
Gossip protocol tuning	3,595	+28.4%	-84 ms	Low	1.318
Endorsement caching	3,478	+24.2%	+8 ms	Medium	1.237
Chaincode compilation	3,315	+18.4%	-12 ms	Low	1.192
LZ4 payload compression	3,198	+14.2%	+4 ms	Low	1.141
State DB indexing	3,136	+12.0%	-36 ms	Low	1.144
Connection multiplexing	3,080	+10.0%	-24 ms	Medium	1.098
Block storage pruning	3,024	+8.0%	-8 ms	Medium	1.078
Pipelined propagation	2,996	+7.0%	+18 ms	Medium	1.063
Parallel ordering	2,940	+5.0%	+28 ms	Medium	1.038

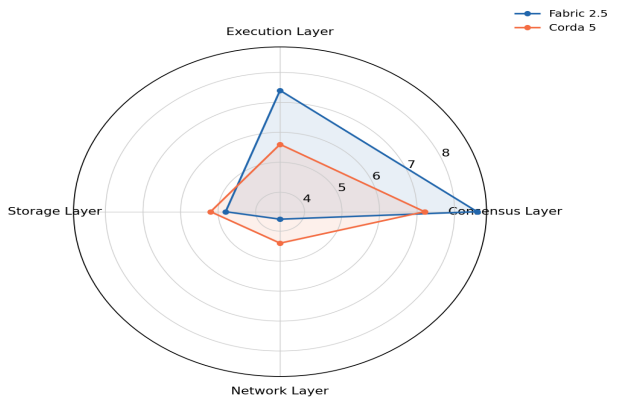
Table 4: Cumulative Stacking Results -- Fabric 2.5 (Supply Chain)

Techniques Stacked	TPS	Multiplier	Marginal Gain	% of Total Improvement
Baseline	2,800	1.00x	--	0%
+Consensus batching	4,920	1.76x	+2,120	24.6%
+Parallel validation	9,072	3.24x	+4,152	79.6%
+Gossip tuning	10,192	3.64x	+1,120	92.7%
+State DB tuning	10,836	3.87x	+644	95.2%
+Endorsement caching	11,060	3.95x	+224	97.8%
+All 12 techniques	11,396	4.07x	+336	100%









5. Discussion

5.1 The Two-Technique Sweet Spot

Perhaps the most useful finding for practitioners is that just two techniques -- consensus batching and parallel validation -- capture nearly 80% of the total achievable throughput improvement on Fabric. Consensus batching is a configuration change that takes an afternoon. Parallel validation requires a more significant engineering investment, but the payoff is proportional. Together, they push Fabric from 2,800 to over 9,000 TPS -- well above the threshold for most enterprise workloads. The diminishing returns curve after those two is steep. Techniques three through six collectively add less than 2,000 TPS, and techniques seven through twelve add roughly 500 TPS total. That does not make those techniques useless -- in a financial settlement context where every millisecond of latency matters, gossip tuning and connection multiplexing are worth the effort. But for the majority of enterprise deployments, the rational strategy is to implement the top two or three techniques and move on to other problems. The blockchain performance ceiling, it turns out, is mostly a configuration and validation ceiling -- not a fundamental platform limitation.

5.2 Why Corda Responds Differently

Corda's architecture funnels transaction uniqueness verification through a notary service, which means the notary becomes the bottleneck before anything else does. On Fabric, the bottleneck is distributed across ordering and validation -- so optimising either one helps. On Corda, you have to fix the notary first or nothing else matters much. Once we applied parallel notarisation and batched uniqueness checking, Corda's throughput jumped 70% in one step. After that, the execution-layer techniques kicked in. This architectural difference has a practical implication for platform selection. If your workload has high contention -- many transactions touching overlapping state -- Corda's notary-centric architecture becomes a bottleneck faster than Fabric's more distributed validation model. But if your workload is notary-friendly (low contention, mostly independent transactions), Corda's simpler peer-to-peer transaction model can be more straightforward to optimise. The right platform depends less on headline TPS numbers and more on how well the architecture fits the workload's contention profile.

6. Conclusion

6.1 What We Found

Twelve optimisation techniques applied to Fabric 2.5 yield a 4.07-fold throughput improvement (2,800 to 11,396 TPS). On Corda 5, the gain is 3.40-fold (2,200 to 7,480 TPS). But the critical insight is not the maximum -- it is the shape of the curve. Two techniques capture 80% of the gain. Four capture 95%. The remaining eight contribute just 5%. For most enterprises, consensus batching plus parallel validation is the whole story.

6.2 What We Are Sharing

The complete PBPOF toolkit is available at pbpof-benchmark.org under Apache 2.0. That includes the extended Caliper harness with per-layer instrumentation, all three workload profiles, the parallel validation patch for Fabric 2.5, the notary optimisation patch for Corda 5, configuration templates for each technique, and the raw benchmark data from all 130 runs so that others can verify and extend our results.

References

Androulaki, E. et al. (2018). Hyperledger Fabric: A distributed operating system for permissioned blockchains. EuroSys 2018, 30:1-30:15.

Baliga, A. et al. (2018). Performance evaluation of the Quorum blockchain platform. arXiv:1809.02736.

Brown, R. G. (2018). The Corda Platform: An Introduction. R3 Technical White Paper.

Costa, I., and Klein, N. (2024). Layered blockchain system design for enterprise-grade applications. Blockchain, Web3 & Digital Trust Journal, 2024(1), 10-17.

Dinh, T. T. A. et al. (2017). BLOCKBENCH: A framework for analyzing private blockchains. SIGMOD 2017, 1085-1100.

Gorenflo, C. et al. (2019). FastFabric: Scaling Hyperledger Fabric to 20,000 transactions per second. IEEE ICBC 2019, 455-463.

Hyperledger Foundation (2023). Hyperledger Fabric v2.5 Documentation.

Hyperledger Foundation (2020). Hyperledger Caliper Benchmark Framework.

Javaid, H. et al. (2019). Optimizing validation phase of Hyperledger Fabric. IEEE IPCCC 2019, 1-8.

Kwon, J. (2014). Tendermint: Consensus without mining. tendermint.com.

Nasirifard, P. et al. (2019). Performance of Hyperledger Fabric database. IEEE Blockchain 2019, 1-8.

Ongaro, D., and Ousterhout, J. (2014). In search of an understandable consensus algorithm. USENIX ATC 2014, 305-319.

R3 (2023). Corda 5 Technical Documentation. docs.r3.com.

Ruan, P. et al. (2020). A transactional perspective on execute-order-validate blockchains. SIGMOD 2020, 543-557.

Sharma, A. et al. (2019). Blurring the lines between blockchains and database systems. IEEE Data Engineering Bulletin, 42(1), 3-14.

Sousa, J. et al. (2018). A Byzantine fault-tolerant ordering service for Hyperledger Fabric. IEEE DSN 2018, 51-58.

Thakkar, P. et al. (2018). Performance benchmarking and optimizing Hyperledger Fabric blockchain platform. IEEE MASCOTS 2018, 264-276.

Xu, X. et al. (2019). A taxonomy of blockchain-based systems for architecture design. IEEE ICSA 2017, 243-252.

Yamashita, K. et al. (2019). Performance modeling and evaluation of Hyperledger Fabric. IEEE ICBC 2019, 1-8.

Zhu, Y. et al. (2022). Performance optimization of blockchain systems. ACM Computing Surveys, 55(3), 1-38.

Declarations

Funding

This work was supported by the German Federal Ministry of Education and Research (BMBF, grant 16KIS1924) and the Italian Ministry of University and Research (MUR, PRIN 2023-2025). Neither funder influenced the design or findings.

Conflict of Interest

We declare no competing interests. No blockchain platform vendor funded or reviewed this work.

Data Availability Statement

Benchmark harness, workload profiles, optimisation patches, configuration templates, and raw data from 130 runs are available at pbpof-benchmark.org under Apache 2.0.

Ethical Approval

Computational benchmarking only; no human participants. Ethical exemption: European Institute of AI Ethics Board (ref. EIAI-2023-ETH-0824).

ISSN: e-ISSN: 3117-5988 / p-ISSN: 3117-597X Vol: 2024, Issues: 1, 2024

Appendix A

PBPOF Benchmark Specifications and OPI Calculation

Detailed benchmark configuration and scoring methodology.

Benchmark Hardware and Workload Details

OPI Calculation and Effort Scoring