The Solana Story: Building the Internet Capital Markets with Pragmatic Engineering

Increase Bandwidth, Reduce Latency

Eray Acikgoz
@erayajk

Aug 2, 2025

Abstract

This paper chronicles and analyses Solana's evolution from a telecommunications-inspired prototype to a production blockchain capable of Web-scale latencies. The analysis first shows how applying time-division multiple access concepts and the Proof-of-History cryptographic clock replaced collision-based consensus with deterministic leader scheduling, raising throughput by an order of magnitude. The paper then examines Solana's virtual-machine choice—an extended Berkeley Packet Filter (eBPF) runtime executed in the Rust BPF (rBPF) sandbox—and discusses the technical rationale for selecting an eBPF-derived runtime over WebAssembly (WASM). A case study of the Agave validator client demonstrates how Solana's parallel execution and an account-locking model achieve sustained throughput in the low-thousands transactions per second (TPS) range in production. Building on these foundations, the paper documents Alpenglow, a forthcoming consensus upgrade that combines Votor's dual-round voting and Rotor's single-hop dissemination to deliver median finality below 150 ms and tolerate 20% adversarial plus 20% non-responsive stake. Finally, the analysis discusses Multiple Concurrent Proposers, selective-friction developer strategy, and organisational spin-outs (Anza, Anza Research), distilling a broader lesson: user-visible performance gains outweigh purely theoretical innovations in driving long-term adoption.

1 Introduction

The blockchain sector has witnessed extensive efforts to mitigate the inherent scalability and throughput constraints characteristic of first-generation networks such as Bitcoin and Ethereum. In this landscape, Solana has emerged as a potential alternative, diverging from approaches centered on novel cryptography by adopting instead a pragmatic engineering methodology prioritizing iterative enhancement, high transaction throughput, and practical performance over theoretical optimality.

This paper delineates the origin, technical principles, and strategic choices underlying Solana's evolution. The network's foundational architecture was influenced by co-founder Anatoly Yakovenko's transition from traditional telecommunications to blockchain technology. Following 14 years at Qualcomm, during which he specialised in performance optimisations for mobile and embedded systems, Yakovenko cultivated a profound understanding of low-level system optimisation.

His professional background included contributions to Voice over IP technologies, mobile operating systems, and specifically, time division multiple access (TDMA) systems, subsequently informing Solana's core innovation.

While employed at Qualcomm, Yakovenko collaborated with colleague Steve Akridge on an ancillary project involving the development of a deep-learning API. During this undertaking, they first encountered cryptocurrency while GPU-mining an early altcoin, an experience that highlighted the computational bottlenecks inherent in existing chains. This direct exposure to blockchain performance constraints served as the impetus for re-evaluating consensus mechanisms through a telecommunications perspective, ultimately leading to the conception of Solana.

2 Technical Architecture and Design Philosophy

2.1 Focus on Trading and Performance

From the project's inception, the Solana team recognized that their competitive advantage resided in performance optimisation, rather than pioneering blockchain research or cryptographic primitives. Anatoly Yakovenko's background in mobile optimisation offered a clear conceptual framework, viewing blockchain as a constrained computing environment analogous to early mobile devices—characterized by limited memory and CPU, yet subject to specific performance requirements.

The initial strategic decision to focus on trading applications was also personally informed. Founder's prior experience trading options and forex across multiple platforms enabled him to directly observe the disadvantages faced by smaller traders competing against participants with enhanced access to execution infrastructure. This firsthand experience motivated Solana's fundamental design objective is to provide fair, high-speed access to financial markets.

2.2 Time Division Multiple Access (TDMA)

Solana's consensus mechanism's breakthrough was achieved through the application of telecommunications principles to blockchain architecture. Yakovenko observed that Bitcoin's consensus mechanism resembled the throughput-limited ALOHA protocol utilized in early radio communications [1, 2], wherein multiple transmitters would attempt simultaneous broadcasting, causing interference and requiring random backoff periods to resolve collisions [3, 4]. This fundamental limitation of contention-based random access protocols, where nodes compete for channel access without coordination, results in throughput degradation as network size increases [5].

The central concept was that with a reliable source of time—facilitated by verifiable delay functions (VDF's)—a blockchain could implement time division multiple access (TDMA), thereby enabling higher channel utilisation [6]. Yakovenko explained that a network's time source facilitates the construction of time division multiple access, allowing participants, each equipped with a clock, to alternate transmissions according to time and confine broadcasts to designated time slots [7].

This technical insight formed the basis for Solana's strategy regarding blockchain scalability: rather than accommodating the inefficiencies of random backoff mechanisms inherent in ALOHA-type protocols, the network could assign specific time slots to validators, thereby improving throughput and eliminating the collision-prone nature of traditional blockchain consensus mechanisms.

2.3 Proof of History (PoH)

The technical innovation termed "Proof of History" (PoH) originated from Yakovenko's concept for establishing a verifiable temporal source within distributed systems [7]. In contrast to conventional blockchain consensus mechanisms which rely on probabilistic finality and random leader selection [8, 9], Proof of History provided a cryptographic clock that orders events prior to consensus, thus facilitating higher throughput. This approach fundamentally differs from traditional verifiable delay functions (VDFs) [10, 11] by creating a continuous, append-only sequence of cryptographic proofs that establishes a canonical ordering of events without requiring global coordination [12].

The core innovation addresses a fundamental limitation in distributed consensus: the absence of a reliable global clock. Conventional proof-of-stake systems suffer from leader election ambiguity and the "nothing-at-stake" problem, requiring quadratic consistency guarantees [8, 13]. By implementing a verifiable delay function that produces a sequential, tamper-evident record of time passage, Solana's Proof of History enables deterministic leader scheduling and reduces settlement finality requirements from probabilistic to near-deterministic bounds [7, 14].

2.4 Virtual Machine and Programming Model

A crucial early determination for the Solana platform involved its virtual machine architecture. Rather than implementing WebAssembly (WASM) or devising a custom high-level virtual machine such as Ethereum's Virtual Machine (EVM) [15], the development team elected to utilise eBPF (extended Berkeley Packet Filter), a technology originally developed for the Linux kernel. This strategic decision was driven by multiple technical and philosophical considerations that align with Solana's performance-first engineering philosophy.

2.4.1 The eBPF Foundation

eBPF represents an evolution from the original Berkeley Packet Filter (BPF), which was developed in the early 1990s for efficient network packet filtering in Unix-like operating systems [16]. The extended version fundamentally transforms eBPF from a simple packet filter into a general-purpose bytecode execution environment capable of running sandboxed programs safely within kernel space [17]. This transformation enabled eBPF to support a broader expanded range of applications, from security monitoring and performance analysis to blockchain smart contract execution.

The architectural foundation of eBPF offers several advantages for blockchain applications. eBPF's kernel-space execution model with JIT compilation delivers near-native performance characteristics, as eBPF bytecode is compiled to native machine code at runtime, providing significant performance advantages over traditional interpreted virtual machines [17]. This performance differential proved particularly valuable for Solana's high-throughput requirements, where even marginal execution improvements translate to measurable gains in overall network capacity.

2.4.2 Technical Rationale and Performance Characteristics

Solana's adoption of eBPF was motivated by several main factors that distinguish it from competing virtual-machine technologies; Table 1 summarises these differentiators.

Table 1: eBPF versus WASM: key technical differentiators for high-throughput blockchains

Dimension	**eBPF advantage over WASM**
Stability & Backwards Compatibility	Kernel ABI guarantees and conservative evolution ensure long-term support, whereas WebAssembly continues to evolve rapidly with frequent specification updates and new feature additions [17,18].
Performance Optimisation	Register-based VM plus in-kernel JIT delivers near-native throughput for compute-bound code paths [16].
Security & Verification	On-load byte-code verifier enforces bounded loops, memory safety and control-flow integrity, providing stronger static guarantees than WASM's runtime sandboxing [19,20].
Resource Efficiency	Empirical studies on resource-constrained systems embeddeb systems show $< 10\%$ additional memory overhead for eBPF, whereas WASM can double memory footprint for comparable workloads [21].

2.4.3 Architectural Implementation

Solana's implementation utilizes rBPF (Rust BPF), a userspace implementation of the eBPF virtual machine specifically adapted for blockchain applications [22]. This adaptation maintains eBPF's core security and performance characteristics while providing the flexibility required for smart contract execution outside the Linux kernel context. The rBPF implementation incorporates Solana-specific modifications, including custom system calls (syscalls) and modified memory management to support blockchain-specific operations.

The adoption of eBPF had ramifications for Solana's developer ecosystem, establishing considerable barriers to entry that required developers to operate at a low level. However, this apparent impediment functioned as a beneficial filter, attracting dedicated developers prepared to invest considerable effort in mastering the platform. This selective friction mechanism proved strategically valuable during periods when competing blockchain platforms experienced influxes of low-effort projects that simply copied existing protocols.

2.4.4 Comparative Analysis with Alternative Virtual Machines

Research comparing eBPF with WASM reveals fundamental trade-offs between security, performance, and programmability [18]. While WASM provides Turing completeness and broader language support, eBPF's restricted instruction set and verification requirements ensure deterministic execution and superior performance characteristics for blockchain applications.

Security analysis demonstrates that eBPF prioritizes kernel integrity protection through its verifier-based approach, while WASM emphasizes sandboxing for untrusted code execution [20]. For blockchain applications requiring deterministic execution and high throughput, eBPF's security model proves more suitable than WASM's general-purpose approach.

The performance implications of this architectural choice extend beyond simple execution speed. eBPF's register-based virtual machine architecture delivers significant performance advantages over stack-based alternatives. Research demonstrates that register-based virtual machines require approximately 32% less execution time compared to stack-based designs [23], while eBPF's JIT compilation enables near-native performance characteristics [24]. This performance profile aligns directly with Solana's requirements for low-latency financial applications.

2.4.5 Long-term Strategic Implications

Solana's choice of eBPF reflects its engineering philosophy prioritizing proven technologies over novel implementations. Rather than developing a custom virtual machine, the platform adopted a mature, existing runtime with established toolchains.

This approach enabled rapid iteration by utilizing mature LLVM-based compilation toolchains, allowing developers to leverage decades of compiler optimisation research without requiring internal virtual machine development.

Furthermore, the evolution of eBPF within the Linux kernel ecosystem yields continuous performance enhancements as the wider development community refines the runtime for advancing hardware capabilities.

2.5 Agave Client/Validator Implementation

The contemporary instantiation of Solana's technical architecture finds its implementation in the Agave validator client, developed by Anza as the successor to the original Solana Labs validator implementation. Agave represents a production-grade implementation of Solana's theoretical design principles, incorporating multiple iterations based on real-world network operation.

2.5.1 Runtime Architecture and Execution Environment

The Agave implementation provides a runtime environment that differs from traditional blockchain architectures through its account-based model and parallel transaction execution capabilities. Unlike Ethereum's global state machine approach, Solana's architecture treats accounts as independent data containers that can be accessed and modified in parallel, provided there are no conflicting writes to the same account.

The runtime architecture incorporates several components:

- **Program Runtime**: Executes eBPF bytecode under fixed resource limits
- **Account Database**: Manages persistent storage optimised for parallel access
- **Transaction Processing Unit (TPU)**: Handles ingest, signature-verify, and execution pipelines

- **Transaction Validation**: Checks transaction validity, account permissions, and program constraints

2.5.2 Consensus Implementation and Network Protocols

Agave implements Solana's consensus protocol through a combination of Proof of History (PoH) for temporal ordering and Practical Byzantine Fault Tolerance (pBFT) for finalization. The implementation integrates several network protocols designed for high-throughput operation:

Turbine Protocol: Implements efficient block propagation using a tree-structured gossip network that enables rapid dissemination of blocks across the validator network. The protocol optimises for both latency and bandwidth efficiency by organizing validators into hierarchical clusters based on network topology and stake distribution.

Gulf Stream: Provides mempool-less transaction forwarding that enables validators to begin processing transactions before block finalization. This approach reduces confirmation latency by overlapping transaction validation with consensus operations.

Sealevel: Implements the parallel smart contract runtime that enables concurrent execution of non-conflicting transactions. The implementation includes dependency analysis and resource management to ensure deterministic execution across all validators.

Cloudbreak: Provides the account database implementation optimised for the high-throughput, parallel access patterns required by Solana's architecture. The system uses memory-mapped files and horizontal scaling techniques to manage the growing state while maintaining fast random access performance.

Table 2: Key Agave components and their performance contributions

Component	**Primary function**	**Performance impact**
Program Runtime (eBPF)	Executes smart-contract bytecode under deterministic resource limits	Near-native execution; predictable latency
Account Database (Cloudbreak)	Persists account state via sharded, memory-mapped storage	Eliminates global lock; $> 50\,\mathrm{k}$ random I/O ops s^{-1} on commodity SSDs
Transaction Processing Unit (TPU)	Pipelines transaction ingest, signature verification, and scheduling	Masks network latency; maximises CPU utilisation
Sealevel Runtime	Schedules non-conflicting transactions across cores	Enables thousands of parallel executions per slot
Turbine Protocol	Disseminates blocks through tree-structured gossip	Sub-second propagation with $O(\log n)$ bandwidth per node
Gulf Stream	Forwards transactions to prospective leaders before block-making	Removes mempool gossip; shortens confirmation latency
Custom Memory Allocators	Pools frequent allocations within the runtime	Cuts allocation overhead by $\sim$20 %
UDP-based Networking	Uses QUIC/UDP for consensus traffic	Avoids TCP back-off; lowers per-packet latency

Table 2 summarises how each subsystem contributes to throughput and latency. The parallel execution model enables Agave to process thousands of transactions simultaneously by analyzing transaction dependencies at ingestion time and scheduling non-conflicting transactions across multiple execution cores. This design pattern overcomes the inherent throughput constraints of sequential execution models that characterize most blockchain systems.

2.5.3 Performance Optimisation and Resource Management

The Agave implementation incorporates multiple optimisations derived from production profiling under real-world load conditions. Key optimisation areas include:

- **Memory Management**: Custom allocators and pooling reduce allocation overhead
- **Network Optimisation**: UDP-based protocols minimise networking overhead for consensus and data propagation
- **CPU Utilisation**: Multi-threaded pipelines maximise CPU use while preserving deterministic ordering
- **Storage Efficiency**: Compressed account storage and indexing reduce disk I/O bottlenecks

2.5.4 Validator Client Architecture

The Agave validator client implements a modular architecture that separates concerns across distinct functional components:

RPC Interface: Exposes Solana RPC methods and subscription streams for client applications.

Banking Stage: Validates, executes, and builds blocks with throughput-oriented scheduling algorithms.

Broadcast Stage: Propagates blocks and transactions with low-latency networking protocols.

Replay Stage: Processes received blocks and maintains vote and fork state.

Storage Management: Manages ledger storage for archival and pruned-state nodes.

The modular design enables independent optimisation of each component while maintaining strict interfaces that ensure consistent behavior across different validator configurations.

2.5.5 Consensus Safety and Liveness Guarantees

Agave implements comprehensive safety and liveness mechanisms that ensure network stability even under adverse conditions:

Fork Selection: Implements Tower Byzantine Fault Tolerance (Tower BFT) with fork selection rules that balance safety and liveness requirements. The algorithm considers both vote weight and lock-out periods when selecting the canonical fork.

Optimistic Confirmation: Provides fast confirmation semantics for applications that can tolerate minimal rollback risk, enabling user interfaces to provide immediate feedback while maintaining eventual safety guarantees.

Vote Credits: Implements economic incentives for validators that encourage timely and accurate consensus participation while penalizing malicious or offline behavior.

2.5.6 Integration with Solana Virtual Machine (SVM)

Agave serves as the reference implementation for the Solana Virtual Machine (SVM), which has emerged as a portable execution environment that can be integrated into other blockchain systems. The SVM implementation within Agave includes:

- **Program Execution**: Deterministic eBPF bytecode execution with strict resource limits and security guarantees
- **Cross-Program Invocation**: Enables composable program interactions while maintaining security isolation
- **Account Model**: Implements Solana's unique account-based state management with rent collection and program-derived addresses
- **System Programs**: Provides core system functionality including token management, stake delegation, and governance operations

The SVM's portability enables other blockchain projects to adopt Solana's execution model while potentially using different consensus mechanisms or network architectures, expanding the ecosystem of compatible tools and applications.

2.6 Alpenglow

Alpenglow is represents a major modification to be implemented within Solana's core protocol to date [25]. This novel consensus protocol resulted researchers from Anza Research, who previously identified fundamental limitations in Solana's current consensus mechanisms [26]. Unlike incremental enhancements, Alpenglow constitutes a complete architectural transformation, deprecating legacy elements such as TowerBFT and Proof-of-History in favor of a dedicated consensus system optimised for global, high-performance proof-of-stake operations [27].

The protocol introduces two primary innovations that fundamentally transform Solana's consensus architecture:

Votor: Manages voting and block finalisation logic through integrated one and two-round voting modes operating concurrently. Should 80% of stake participate, blocks finalize in a single voting round; if only 60% of stake is responsive, the system transitions to two-round finalization. This dual-mode approach ensures finalization occurs as soon as the faster path terminates, achieving enhanced performance in distributed consensus.

Rotor: Rotor constitutes a data dissemination protocol that refines the approach of Turbine. It mitigates the fundamental constraint of network latency, acknowledging the inherent physical limits on signal propagation speed. Rotor minimizes network hops via a single-layer relay architecture, in contrast to Turbine's multi-layer tree structure. This design optimisation acknowledges that information dispersal delays are predominantly influenced by network latency rather than transmission or computation overhead. Akin to Turbine, Rotor utilizes participant bandwidth proportional to stake, thus ensuring asymptotically optimal bandwidth utilization while alleviating leader bottlenecks.

Alpenglow attains median finality times of approximately 150 milliseconds, with optimal conditions allowing finalization within 100 milliseconds [27]. These latency attributes represent a reduction from TowerBFT's 12.8-second finality times and surpass the performance of Solana's optimistic confirmation mechanisms. A 150-millisecond median latency enables Solana to directly compete with the responsiveness of conventional Web2 infrastructure, potentially facilitating the deployment of blockchain technology in new categories of real-time applications.

The protocol's "20+20" resilience model provides advanced fault tolerance, operating effectively under challenging network conditions while tolerating up to 20% adversarial stake and 20% non-responsive stake. This framework ensures robust operation during network partitions, coordinated attacks, or extensive validator failures.

Alpenglow exemplifies Solana's iterative engineering philosophy and its willingness to supersede foundational innovations with superior solutions, even while operating at scale. The protocol represents a successful synthesis of academic research and practical blockchain development, demonstrating the network's commitment to fundamental architectural improvements.

3 Engineering Culture and Development Philosophy

3.1 Iteration Over Perfection

A core principle guiding Solana's development prioritizes functional software delivery over theoretical perfection. This philosophy emerged from founders' experience in building multiple systems, where developing multiple systems exposed the pattern of second-system syndrome; by his fourth or fifth iteration, prioritizing rapid deployment was deemed essential. The team accepted accumulating technical debt to address immediate user needs, anticipating that successful systems would later provide the resources to resolve it. A strict hierarchy is followed, prioritizing end-user experience, then developer convenience, and finally, internal engineering preferences. Instead of planning comprehensive upgrades, the team focuses on incremental improvements enabling specific user behaviors or removing specific blockers.

3.2 Small Team Advantages

Throughout its development, Solana maintained a deliberately small core team. This structural characteristic conferred several advantages during periods of crisis, as a limited team size coupled with substantial workload minimized internal conflicts and political maneuvering. Each member possessed clear visibility regarding the direct impact of their contributions on user outcomes, while fewer stakeholders facilitated rapid decision-making in response to emergent challenges. Furthermore, a collective sense of ownership fostered shared accountability for both positive and negative outcomes.

3.3 User-Centric Priority Hierarchy

Solana's development process follows a strict prioritization hierarchy that places end-user experience above all other considerations:

1. **End users**: The people actually using applications on the network
2. **Developers**: The teams building applications and infrastructure
3. **Validators**: The operators maintaining network infrastructure
4. **Core engineers**: The client teams building the protocol

This hierarchy governs decision-making during resource conflicts. Inconvenience to developers is deemed acceptable if it facilitates superior user experiences, and internal engineering preferences are subordinate to the requirements of both users and developers.

3.4 Rapid Iteration and Low-Level Changes

A distinctive characteristic of the Solana network is its capacity to undertake fundamental modifications to core protocol components even while operating at scale. The current initiative to entirely replace Solana's consensus algorithm serves as a prime illustration of this methodology.

Counter-intuitively, implementing these low-level changes is frequently simpler than altering higher-level APIs. This was because such changes involved deep, low-level code affecting only a limited number of engineers, whereas breaking an RPC API impacted numerous applications and users dependent upon it.

This principle permeates all facets of the system. The development team regards software as intrinsically dynamic, necessitating continuous adaptation to evolving hardware capabilities and user requirements. Static systems are invariably rendered obsolete as the underlying computing substrate progresses.

3.5 The Formation of Anza and Anza Research

In January 2024 Solana established Anza, a dedicated validator-client development organisation founded by former Solana Labs executives and core engineers, to focus on building Agave, the successor to the original Solana Labs validator client [28]. This separation permits specialised teams to iterate more rapidly, decentralises client development and allows Solana Labs to pursue complementary initiatives, such as Solana Mobile.

In December 2024 Solana launched Anza Research, a collaboration with Professor Roger Wattenhofer and ETH Zurich researchers Kobi Sliwinski and Quentin Kniep, to develop a next-generation consensus algorithm, Alpenglow [29]. The partnership originated from the researchers' public critique of Solana's consensus protocol which prompted discussions with Solana team; they elected to devote their research efforts exclusively to Solana. Anza Research aims to achieve lower latency, enhanced resilience under adverse network conditions, and incentive alignment sufficient to sustain performance at global scale.

4 Developer Ecosystem and Platform Strategy

Solana intentionally maintains a non-trivial entry barrier for developers. Writing programs in eBPF requires explicit memory management, deterministic control flow and compile-time resource budgeting. These constraints discourage superficial code forks and favour engineering teams that optimise for latency and throughput.

4.1 Standardisation and Composability

eBPF's restricted syscall interface, combined with the single-account write-lock model, channels development toward a narrow set of well-audited primitives. Token-2022 has gained adoption with over 700,000 tokens deployed by 2024 [30], while the Metaplex metadata standard accounts for over 99.9% of the Solana non-fungible token (NFT) market, encompassing more than 20 million NFTs [31]. The resulting homogeneity confers three technical advantages:

- **Reduced audit overhead**: A single reference implementation shortens security review cycles.
- **Predictable composability**: Uniform interfaces allow automatic indexing and listing across protocols.
- **Lower integration cost**: New applications interact with existing programmes without bespoke adapters.

4.2 Hackathons and Grants

Solana's hackathon ecosystem provides quantitative evidence of the platform's ability to attract and retain high-quality developers despite its technical complexity. Since 2020, eight major hackathons have generated entrepreneurial activity, with over 70,000 participants competing globally and producing more than 5,000 launched products[1]. The economic impact has been considerable: hackathon winners have collectively raised over $650 million in venture capital funding, with 80% of all VC-backed startups in the Solana ecosystem originating from these competitive events[2].

Recent hackathons demonstrate increasing scale and sophistication. The Hyperdrive hackathon attracted 907 project submissions from over 120 countries, while Grizzlython offered $5 million in prizes and engaged 18,000 participants who submitted 750 projects[3]. This growth trajectory reflects the platform's expanding global developer community and the maturation of supporting infrastructure, including the Rust SDK, Anchor framework, and local SVM testing environments.

The ecosystem's commitment to post-hackathon support has institutionalized through Colosseum, an independent organisation that raised a $60 million venture fund in 2024 specifically

[1]Colosseum. (2024). Welcome to Colosseum. Available: https://www.colosseum.org/hackathon

[2]Solana Compass. (2023). Breakpoint 2023 Highlights Hyperdrive Hackathon Winners on Solana Blockchain. Available: https://solanacompass.com/learn/breakpoint-23/breakpoint-2023-winners-of-the-hyperdrive-hackathon

[3]Solana Foundation. (2023). Welcome to Grizzlython, the next Solana Hackathon. Available: https://solana.com/news/grizzlython-solana-hackathon

to invest in hackathon winners. Colosseum provides $250,000 in pre-seed funding to selected teams through a six-week accelerator program, having already deployed $2.75 million across 11 companies[4]. This structured pathway from hackathon participation to venture funding demonstrates the platform's evolution from experimental developer outreach to systematic startup formation.

Complementary grant programs extend support beyond hackathon winners. Superteam's Instagrants program offers equity-free grants ranging from $1,000 to $20,000 with rapid 72-hour decision cycles, while maintaining active chapters across multiple regions including India, Germany, Vietnam, and the United Kingdom[5]. The distributed nature of these programs enables localized developer support while maintaining technical standards aligned with Solana's performance-first philosophy.

5 Future Vision and Strategic Direction

5.1 Multiple Concurrent Proposers

Solana's forthcoming Multiple Concurrent Proposers (MCP) upgrade introduces *simultaneous block production* to resolve the adverse-selection and cancellation-race problems that hinder on-chain central-limit order books or other DeFi protocols. Under the current single-proposer design, market-makers win the cancel race only 13 % of the time [32]; MCP seeks to restore price competitiveness without sacrificing decentralisation.

Dual-fee ordering model. MCP separates fees into (i) *inclusion fees*, which compensate validators, and (ii) *ordering fees*, which are burned. Programs access the latter via the syscall `get_transaction_metadata`, enabling user-defined policies—e.g. "cancels before takes"—that give market-makers deterministic priority and tighten bid–ask spreads.

Consensus workflow.

1. *Proposal* — Each concurrent leader erasure-codes its block into shards.
2. *Dissemination* — Shards are broadcast across the first Turbine layer on a fixed schedule.
3. *Attestation* — Relays issue signed `IHAVE` messages confirming shard receipt.
4. *Finalisation* — A consensus leader aggregates attestations and builds the canonical block.

Binding/blinding constraints prevent leaders from reacting to competitors' blocks, and *wall-clock fairness* enforces near-simultaneous commitment.

Conflict resolution. If concurrent blocks contain conflicting transactions, the protocol unions the transaction sets and orders them by descending ordering-fee, guaranteeing censorship-resistance and predictable sequencing.

Expected impact. Benchmarks indicate MCP maintains Solana's 2200-validator decentralisation while delivering low latency. By letting applications pay for ordering determinism

[4]Ashraf, A. (2024). Solana-Focused Startup Accelerator Colosseum Raises $60M to Invest in Early-Stage Projects. CoinDesk. Available: https://www.coindesk.com/business/2024/06/25/solana-focused-startup-accelerator-colosseum-raises-60m-to-invest-in-early-stage-projects

[5]Superteam India. (2022). Apply for an Instagrant! Available: https://in.superteam.fun/instagrants

rather than bidding in private auctions, MCP provides a protocol-native revenue stream and advances Solana's goal of supporting global financial throughput and latency requirements.

5.2 Application-Driven Development

Solana's future development strategy prioritizes applications with proven product-market fit over abstract technical advancements. This stance reflects the observation that, despite extensive blockchain development over several years, only a limited number of smart contract patterns have attained adoption.

It has been noted that over the preceding five years, approximately five smart contract patterns have demonstrated significance: implementations for tokens, NFTs, automated market makers (AMMs), lending protocols (encompassing various risk engines), and central limit order book markets. These were characterised as a small number of asset types, market types, and leverage/risk systems, collectively comprising five or six distinct areas.

Consequently, rather than focusing on developing increasingly sophisticated virtual machines or programming languages, the emphasis is placed on enabling these core use cases to operate with greater efficiency and provide enhanced user experiences.

5.3 Hardware Co-evolution

Solana's long-term vision is distinguished by the recognition that blockchain software must evolve concurrently with advancements in hardware capabilities. In contrast to many blockchain projects with fixed technical architectures, Solana embraces continuous adaptation. This perspective posits that software development is inherently continuous; it becomes non-viable when static, given that the evolution of fundamental hardware components, such as changes in instruction set architectures (e.g., from AVX1 to AVX2), alters performance characteristics and operational costs. These hardware shifts necessitate corresponding software adjustments. This principle enables Solana to leverage enhancements in processing units, memory systems, and network hardware for consistent performance improvement without major architectural revisions.

6 Conclusion

Solana's history shows how pragmatic, performance-oriented engineering can push blockchain infrastructure toward Web-scale latency and throughput. Beginning with telecom–inspired ideas such as TDMA and the Proof-of-History clock, the network moved from a prototype to production by favouring incremental, measurable wins over large rewrites. This ethos is visible in the choice of an eBPF-based virtual machine (implemented as rBPF), the Agave validator that operationalises parallel execution, and the coming Alpenglow consensus and Multiple Concurrent Proposers upgrades that target sub-second finality and fair, high-bandwidth block production.

Three principles consistently explain Solana's trajectory:

1. **Iterate for user-visible performance**. Sustained reductions in latency and increases in throughput translate directly into higher utilisation and fee revenue, whereas purely theoretical novelty rarely delivers durable adoption.

2. **Selective friction strengthens ecosystems**. Low-level programming, deterministic resource budgeting and a narrow account model deter superficial forks while rewarding teams that optimise for speed and composability. Hackathon and grant programmes then convert this filtered talent into production-grade applications.

3. **Software must co-evolve with hardware**. By treating the network as living infrastructure—ready to swap out consensus, networking or execution layers when better algorithms or faster silicon appear—Solana avoids ossification and captures hardware-driven efficiency gains.

Looking forward, the combination of Alpenglow's $\sim$150 ms median finality, MCP's deterministic ordering, and ever-improving validator hardware positions Solana to service real-time markets that demand both decentralisation and millisecond-level responsiveness. For the wider blockchain sector, the lesson is clear: success hinges less on research novelty than on relentless, data-driven iteration that solves concrete user problems and scales with the computing substrate itself.

References

[1] Kuo, F. F. (1981). Computer networks—the ALOHA system. *Journal of Research of the National Bureau of Standards*, 86(6), 591–627.

[2] Abramson, N. (1970). The ALOHA system: Another alternative for computer communications. *Proceedings of the Fall Joint Computer Conference*, 281–285.

[3] Farhadi, F., & Ashtiani, F. (2014). Stability region of a slotted ALOHA network with K-exponential backoff. *arXiv preprint arXiv:1406.4448.*

[4] Lee, T. T., & Dai, L. (2009). Buffered ALOHA with K-exponential backoff part I: Stability and throughput analysis. *arXiv preprint arXiv:0907.4251.*

[5] Barcelo, J., Faridi, A., Bellalta, B., Martorell, G., & Malone, D. (2013). On the distributed construction of a collision-free schedule in WLANs. *arXiv preprint arXiv:1311.0787.*

[6] Vinel, A. (2012). Multiple access communications in future-generation wireless networks. *EURASIP Journal on Wireless Communications and Networking*, 2012(1), 45.

[7] Yakovenko, A. (2020). Solana: A New Architecture for a High Performance Blockchain. *arXiv preprint arXiv:2008.03678.*

[8] Blum, E., Kiayias, A., Moore, C., Quader, S., & Russell, A. (2019). Linear consistency for proof-of-stake blockchains. *arXiv preprint arXiv:1911.10187.*

[9] Azouvi, S., & Cappelletti, D. (2021). Private attacks in longest chain proof-of-stake protocols with single secret leader elections. *3rd ACM Conference on Advances in Financial Technologies (AFT '21).*

[10] Boneh, D., Bonneau, J., Bünz, B., & Fisch, B. (2018). Verifiable delay functions. *Advances in Cryptology—CRYPTO 2018*, 757–788.

[11] Wu, Q., Xi, L., Wang, S., Ji, S., Wang, S., & Ren, Y. (2022). Verifiable delay function and its blockchain-related application: A survey. *Sensors*, 22(19), 7524.

[12] Ephraim, N., Freitag, C., Komargodski, I., & Pass, R. (2020). Continuous verifiable delay functions. *Advances in Cryptology—EUROCRYPT 2020*, 125–154.

[13] Quader, S., Kiayias, A., & Russell, A. (2020). Consistency of proof-of-stake blockchains with concurrent honest slot leaders. *arXiv preprint arXiv:2001.06403.*

[14] Motepalli, S., & Jacobsen, H. A. (2021). Decentralizing permissioned blockchain with delay towers. *arXiv preprint arXiv:2203.09714.*

[15] Wood, G. (2014). Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Yellow Paper.* [Online]. Available: `https://ethereum.github.io/yellowpaper/paper.pdf`

[16] Scholz, D., Raumer, D., Emmerich, P., Kurtz, A., Lesiak, K., & Carle, G. (2018). Performance implications of packet filtering with Linux eBPF. *Proceedings of the 30th International Teletraffic Congress*, 209–217.

[17] Gbadamosi, B., Leonardi, L., Pulls, T., Høiland-Jørgensen, T., Ferlin-Reiter, S., Sorce, S., & Brunström, A. (2024). The eBPF runtime in the Linux kernel. *arXiv preprint arXiv:2410.00026.*

[18] Huang, W., & Paradies, M. (2021). An evaluation of WebAssembly and eBPF as offloading mechanisms in the context of computational storage. *arXiv preprint arXiv:2111.01947.*

[19] Xu, T., et al. (2023). Programmable system call security with eBPF. *Proceedings of the 18th European Conference on Computer Systems*, 154–168.

[20] Dejaeghere, J., Gbadamosi, B., Pulls, T., & Rochet, F. (2023). Comparing Security in eBPF and WebAssembly. *Proceedings of the 1st Workshop on EBPF and Kernel Extensions*, 35–41.

[21] Zandberg, K., & Baccelli, E. (2020). Minimal virtual machines on IoT microcontrollers: The case of Berkeley packet filters with rBPF. *9th IFIP/IEEE International Conference on Performance Evaluation and Modeling in Wired and Wireless Networks (PEMWN 2020).*

[22] Anza. (2024, December 16). The Solana eBPF Virtual Machine. *Anza Blog.* [Online]. Available: `https://www.anza.xyz/blog/the-solana-ebpf-virtual-machine`

[23] Shi, Y., Gregg, D., Beatty, A., & Ertl, M. A. (2005). Virtual machine showdown: Stack versus registers. *Proceedings of the 2nd International Conference on Virtual Execution Environments*, 153–163.

[24] Mao, J., Ding, H., Zhai, J., & Ma, S. (2024). Merlin: Multi-tier optimization of eBPF code for performance and compactness. *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, 864–878.

[25] Anza. (2025, May 19). Alpenglow: A New Consensus for Solana. *Anza Blog.* [Online]. Available: `https://www.anza.xyz/blog/alpenglow-a-new-consensus-for-solana`

[26] Sliwinski, J., Kniep, Q., Wattenhofer, R., & Schaich, F. (2024). Halting the Solana blockchain with epsilon stake. *ICDCN '24: Proceedings of the 25th International Conference on Distributed Computing and Networking*, 45–54.

[27] Kniep, Q., Sliwinski, J., & Wattenhofer, R. (2025, May 19). Solana Alpenglow Consensus: Increased Bandwidth, Reduced Latency. *Anza White Paper v1.* [Online]. Available: `https://drive.google.com/file/d/1y_7ddr8oNOknTQYHzXeeMD2ProQ0WjMs/view`

[28] Washington, J. (2024, January 30). Meet Anza, a New Solana-Focused Dev Shop. *Anza Blog.* [Online]. Available: `https://www.anza.xyz/blog/meet-anza-a-new-solana-focused-dev-shop`

[29] Anza. (2024, December 16). Anza Research: A New Chapter for Solana's Protocol. *Anza Blog.* [Online]. Available: `https://www.anza.xyz/blog/anza-research-a-new-chapter-for-solanas-protocol`

[30] Certora. (2024, August 5). Reviewing Token Extensions on Solana Using Formal Verification. *Certora Blog.* [Online]. Available: `https://www.certora.com/blog/token-extensions-audit`

[31] The Metaplex Foundation. (2022, September 19). Metaplex: A Decentralized Protocol for Digital Assets. *Metaplex Whitepaper v0.1.* [Online]. Available: `https://whitepaper.metaplex.com/whitepaper.pdf`

[32] Resnick, M., & Yakovenko, A. (2025, May 8). The Path to Decentralized Nasdaq. *Anza Blog.* [Online]. Available: `https://www.anza.xyz/blog/the-path-to-decentralized-nasdaq`