Kadena

The first scalable, high performance private blockchain

Will Martino <will@kadena.io>

Revision v1.0

August 2016

Abstract

This paper introduces Kadena, the first private/permissioned blockchain technology to achieve high performance at scale. Kadena is an implementation of the novel ScalableBFT consensus protocol, which draws inspiration from the Tangaroa protocol as well as practical engineering realities. Until now, private blockchain technologies have been able to provide either high performance or scalability but not both. Rarely deployed into production, BFT-Consensus algorithms achieve high performance initially, but exhibit drastic performance degradation as cluster size increases. Mining (or "proof of work") is the only workable alternative: it has practically no scaling limit. However, being a probabilistic mechanism, mining is necessarily slow and thus unable to attain the performance demanded by enterprise use-cases.

The ability to perform at scale, without sacrificing the guarantees that blockchains provide, has been a major unresolved problem in the implementation of private blockchains. Scaling is critical for industrial adoption. Blockchain applications must be able to maintain acceptable performance for a successful deployment to sustain wide and growing participation.

Common Features of Private Blockchains

The term “blockchain” is imprecise, often deployed as an umbrella term to describe a wide array of blockchain-like applications. It is even less well-defined for *private* or *permissioned* blockchains: industrial applications, which unlike Bitcoin or Ethereum, have identified participants operating nodes in the cluster. To address this imprecision, we enumerate a core feature-set that any private blockchain solution must provide.

1) *Distributed*: participating entities must be able to host their participating nodes from whichever locality or data-center they choose.

2) *Full Replication*: if a node is lost or goes offline, the cluster must be able to fully recover that node and/or replay to new nodes, restoring full functionality and data.

3) *Immutability*: once a transaction is committed to the blockchain it can neither be changed nor removed. This by extension requires all transactions to be fully verifiable via cryptographic techniques such as Merkle trees or incremental hashes.

4) *Privacy*: transactions must be able to be hidden from all but the intended counterparties as required by the use-case.

5) *Byzantine-Fault-Tolerant (BFT) Consensus*: the blockchain must employ a deterministic mechanism for achieving fault-tolerant consensus, instead of probabilistic and incentive-based approaches like mining.

6) *High Performance*: able to achieve superior throughput and latency (by orders of magnitude) than what current mining-based approaches offer.

7) *Scalability*: able to maintain high performance as the number of nodes increases, well into the hundreds if not thousands.

8) *Strong, Durable Cryptographic Identification*: the author of every submitted transaction must be cryptographically verifiable in a storable and reproducible way.

Kadena: High Performance at Scale

Kadena, our proprietary implementation of the ScalableBFT protocol, has achieved an industry first: a deterministic blockchain consensus that provides high performance and low latency, scales flat when customers need it to, and is uncompromising on the guarantees that blockchains are meant to provide. The chart below demonstrates Kadena's expected and real-world performance as it scales. Kadena executes over 7000 transactions per second, with latency in the low milliseconds, with consistent performance on cluster sizes of up to 256 nodes. We forecast steady performance well into the 1000s of nodes.

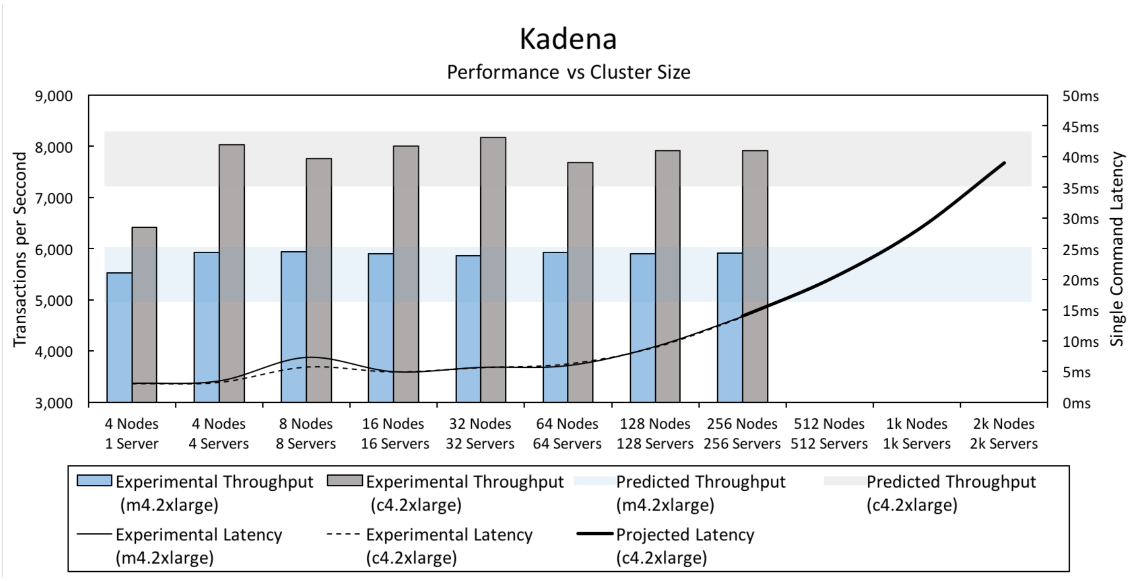

Figure 1: Kadena performance. Nodes are Amazon m4.2xlarge and c4.2xlarge, deployed to the same AWS region.

To design ScalableBFT we returned to the first principles of the *Tangaroa* (Copeland/Zhong) and *Juno* (Martino/Popejoy) BFT-Consensus protocols to develop a predictive model of how Kadena, our implementation of ScalableBFT, should perform. The prediction was as surprising as it was accurate: well before we finished our first cut of Kadena, our model predicted it would support flat scaling over very large node counts. This model provided the groundwork for upgrading Juno/Tangaroa into what is now the *ScalableBFT protocol*, while also guiding the design of the Kadena application.

The scalability of Kadena is, of course, not infinite. Our model predicts that that between 5,000 and 15,000 nodes (the limit is hardware and network dependent) Kadena will begin to exhibit linear performance degradation to some limit of acceptable performance, after which nodes can no longer be added.

We maintain that even for widely-adopted private applications, node counts under 10,000 are sufficient. Assuming that every entity participating in a private blockchain hosts 3 nodes for resiliency, Kadena would enable the cluster to scale up to three hundred participants (900 nodes) with no performance degradation.

Subject to the aforementioned scaling constraints, Kadena's performance is dictated by hardware and network capabilities. Cluster latency is a function of the median network latency, and cluster throughput is a function of both median bandwidth and median host-server CPU capacity. These are "good problems to have," resolved by purchasing better capabilities, instead of facing fundamental algorithmic limitations, as we'll discuss later.

It is also worth noting that Kadena uses available bandwidth very efficiently. All private blockchains must replicate new transactions before they come to consensus, which requires some amount of initial bandwidth to complete. Kadena only requires an additional 0.1% of this initial bandwidth to then reach consensus.

An Overview of BFT-Consensus in Blockchain

The private blockchain scalability problem is subtle and easily misunderstood. It behooves us to briefly cover the current set of mechanisms that are under consideration for implementing consensus for private blockchains.

Mining and Private Blockchains

Mining represents a master-stroke of design as a solution to BFT Consensus for an anonymous, public blockchain. Though a full description of how and why mining works is outside the purview of this whitepaper we would like to point out some of the advantages, limitations and unresolved issues of using mining for private blockchains.

Mining's biggest advantage is its ability to provide BFT in a trustless environment. This advantage, however, is useless in a private setting, as by definition all participant identities are known ahead of time. Mining's biggest applicable advantage is its ability to massively scale: the throughput and latency of a mining-based system is not related to the number of participating entities.

Mining is a probabilistic process that necessarily depends on time for its BFT guarantees. One cannot simply decrease mining difficulty to increase performance, as confirmation of a transaction is probabilistic. Waiting the customary "6 blocks" for confirmation in Bitcoin is based on the probability that a longer fork could be created approaching zero. For a public blockchain like Bitcoin, these are excellent tradeoffs, as in return we obtain a BFT consensus that is capable of public, anonymous deployment.

When used for private blockchains, mining's tradeoffs are no longer acceptable, as a public, anonymous system is no longer desired. Indeed, mining's very consensus mechanism becomes questionable in private settings. One of the foundational pillars of mining is providing meaningful *incentives* to non-byzantine miners, usually in the form of fees and/or the issuance of new crypto-coins. Behaving in a byzantine ("traitorous") way must not be as financially viable as behaving in a non-byzantine ("loyal") way for miners. This pillar of incentivization crumbles when mining is used for private blockchains: the reward of coins/fees cannot hope to compete with significant business advantage that arises from the ability to retroactively invalidate transactions by mining a longer chain.

Proposed solutions to the incentivization problem usually revolve around registering all transactions with a third party oracle, and/or legal recourse for invalidating transactions. Such solutions are unsatisfactory, as they significantly detract from the promise of private blockchains: they are not *distributed*, instead relying on a central 3rd party oracle, and are not *robust,* relying on legal recourse.

BFT-Consensus Algorithms, Past and Present

BFT Consensus Algorithms (BFT-Consensus) have been a subject of academic and industrial research for decades. Historically BFT-Consensus has found production use in air- and spacecraft-control systems, where a small set of computers, connected via fast and reliable networks, come to consensus about control decisions, as a critical safety feature. In contrast, private blockchain applications demand large, globally-distributed clusters capable of maintaining high performance over real-world internet infrastructure.

General-purpose BFT-Consensus algorithms[1] include Practical Byzantine Fault Tolerance (PBFT) and SmartBFT. They provide the same core guarantees and basic features:

- A consensus mechanism robust against malicious and faulty nodes
- High performance (both throughput and latency)
- Deterministic (as opposed to probabilistic) confirmation

Unfortunately, this family of consensus algorithms were never designed for performance at scale yet performance at scale is precisely what is needed to take private blockchains into production. Pilot projects using non-scalable algorithms can demonstrate performance that trounces mining. As solutions are rolled out, however, scalability comes to the forefront: performance degradation spikes as node count climbs. This issue is fundamental to this family of algorithms, and thus unavoidable: they were simply not designed with +1k node clusters in mind.

[1] An interesting aside, when discussing this family of algorithms, is that they clearly demonstrate the difference between a blockchain and a distributed ledger. A distributed ledger is not necessarily a blockchain. The key difference is the utilization of a cryptographic data structure for storing transactions. Many algorithms in this family achieve a BFT consensus surrounding transactions without needing a cryptographic data structure (e.g. Merkle tree). That is not to say that these algorithms cannot use cryptographic data structures, as the addition of one is generally trivial given that BFT is the hard part, just that they aren't always necessary. In this way one can create a distributed ledger that is not technically a blockchain. This distinction makes the approach neither "better" nor "worse," merely different with respect exactly what type of security is provided for historical transactions and how it is provided.

Challenges Facing Private Blockchains

We are witnessing the birth of a paradigm-shifting technology, akin to the shift from mainframes to databases. Currently we are in the “whatever works” discovery phase due to the immaturity of the technology: low throughput, ill-designed smart-contract layers, and compromises of cryptographic security in the name of “performance” are the norm. Performance at scale is often left undiscussed.

As the market matures and adoption of this technology increases the focus of performance, scalability, security, and usability will increase. Though a perfect forecast of this evolution is impossible, new problems will arise that will demand solutions.

Parameters of Performance

All blockchain applications must, at some point:

1) Replicate new transaction received from clients.
2) Confirm the successful replication of the new transactions to the cluster (i.e. come to consensus).
3) Validate the authorship of each transaction.
4) Execute each transaction.

The speed at which a private blockchain can execute these four steps dictates its fundamental *performance*, while the efficiency, specifically the number of inter-server messages needed and the bandwidth needed for these post-replication messages, dictate its *scalability*. Though there are additional steps, these four form the vast majority of the workload.

If one accepts that mining is inadequate for consensus in private settings, due to its un-fixable performance issues and the breakdown of mining incentives, then performance and scalability become inescapably intertwined for private blockchains.

Cryptography Demands on Private Blockchains

The third fundamental step, verifying authorship, places some of the most stringent bounds on the performance of private blockchains because of the cryptographic work required. There are several approaches to accomplishing this verification: Public-Private Key signatures (signed transactions), threshold encryption, and secured channels (TLS).

Public-key signatures

Signed messages, the method that Kadena and Bitcoin use, is by far the most computationally demanding approach. In return it provides *durable security* and auditability guarantees. A signed transaction is permanently secured – the transaction cannot be changed without compromising the key needed to construct it.

Threshold Encryption

Threshold encryption is a newer and potentially fast method that can provide durable security. However, it is a distributed affair, requiring cluster nodes to work together to decrypt each command. This increases the bandwidth demands post-replication, while also coupling the "threshold required" to cluster size.

TLS/Secured Channels

Secured channels is the most computationally efficient, as the very existence of the channel itself affords the authorship guarantee. However, this method only provides *ephemeral security*, which cannot be persisted to disk. It is not possible to analyze past transactions command to prove authorship, which significantly degrades the security guarantees that blockchains are promising.

This verification model has major performance implications. For example, Kadena spends the vast majority of CPU time verifying signatures. When Kadena is configured to employ ephemeral security via TLS, its performance spikes to nearly 20k transactions per second. Unfortunately, we at Kadena cannot accept the tradeoffs imposed by ephemeral security.

Questions for a Private Blockchain Implementation

Despite the hype and astronomical market size estimates concerning private blockchains the fundamental technology must (a) actually work and (b) be better than current centralized, database-oriented solutions. Discerning which solutions hold the most promise will, of course, be tricky. Though the following list is by no means complete, it contains many of the critical questions that, over the coming years, interested parties should be asking of any private blockchain solution they find:

- How many rounds of messages are required to commit a single transaction?
  - Is this number a function of cluster size?
  - Can the solution batch transactions?
  - How is the number related to the number of transactions being replicated?
- Does the solution provide *ephemeral* or *durable* security?
  - Can we audit the blockchain and prove authorship of historical transactions?
  - If we shut down the blockchain can we still audit transactions?
- Is transaction authorization *logical* or *crypto-verifiable*?
  - Would a transaction with CEO-level authorization require a special key to create or would it be like any other command except that the "from" field says "CEO"?
- Does the security model used impact bandwidth demands?
  - Does it impact messaging rounds?
  - Is it impacted by the cluster size?
- How much bandwidth, in addition to the initial replication's bandwidth, is required by each node? By the cluster?
  - How is it related to cluster size?
  - How is it related to newly replicated commands?
- What would an attacker need to do to masquerade as someone else and submit a malicious command?
- If an attacker fully compromises a single node/server participating in the private blockchain, how much damage could the attacker do without the cluster being aware something was wrong?
  - Could the attacker submit malicious transactions? Change the body of a committed transaction? Reorder the log of transactions?
  - How do these answers change if the attacker compromises 33%, 34%, 50%, 51%, 67%, or 100% of the entire cluster?[2]

[2] NB: the answers to this question should get progressively worse for any solution. BFT-consensus is fundamentally about "coming to an agreement in an environment where actors can lie." When the majority is dishonest, especially in a coordinated way, it is impossible for a virtuous node to agree to the "right" thing; at that point a "do no harm" approach is all one can ask for.