International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(17s) 2026 pp. 1336-1350
https://doi.org/10.70917/ijcisim-2026-4830
Article

Eliminating Single Point Failures through Probabilistic Consensus, ZooKeeper Coordination, and Kafka-Style Replication in HDFS Storage Systems

Rohini D. Dhage[1], Vaibhav R. Bhedi[2]

[1]VMV Commerce, JMT Arts & JJP Science College, Nagpur, Maharashtra, India.
Email: rohinidhage@gmail.com
[2]VMV Commerce, JMT Arts & JJP Science College, Nagpur, Maharashtra, India.
Email: bhedivaibhav@gmail.com

Abstract: Although a significant weakness in HDFS-like storage architectures is that the system still relies on single points of failure due to the fact that most metadata leadership is currently controlled by a single active NameNode with passive NameNodes waiting for some form of signal upon an active node's failure; we are presenting here, a Probabilistic Consensus-Driven Active-Standby Controller (PC-ASC) as a model to remove these weak linkages using Bayesian health beliefs, majoritarian quorums to validate, ZooKeeper-based coordination, and Kafka-style data replication of metadata logs. The PC-ASC also includes an evaluation framework that continually monitors and measures the performance of heartbeat quality, metadata latency, the state of the network topology, and the values of the vector representing failure probabilities, to generate trust aware role assignment decisions and decision-based failover signals. We have utilized a controlled emulation data set consisting of metadata requests, heartbeat trace information, injected crash events, and duplicated edit-log record sets to perform comparative evaluations of the proposed approach relative to the single-name-node architecture, the traditional HA-ZKFC/QJM architecture, and the raft-like metadata clustering baseline approaches. Our results indicate fail-over times were reduced from 15.2 seconds down to 6.0 seconds, our available time was increased from 99.59% to 99.999%, and our metadata throughput improved by as much as 3.04 times compared to when we had only a single NameNode controlling all operations. Ablations studies confirm that consensus mechanisms were important to achieving improvements, trust scoring was important to achieving improvements, and the use of log-based recovery was important to achieve improvements in process.

Keywords: Single Point Failure, HDFS High Availability, Probabilistic Consensus, ZooKeeper Coordination, Kafka Replication, NameNode Failover, Metadata Resilience, Fault Tolerance

1. Introduction:

In distributed storage systems it is anticipated that users will experience no interruption to their level of access to files during the loss of nodes. However, the metadata layer often has less redundancy than the data layer. Although data blocks in HDFS-like architectures are replicated across several DataNodes, ownership of namespaces, mapping of blocks, management of leases and editing logs are all typically located at the NameNode. Consequently, there exists a control-plane single point of failure. Even if an active-standby solution is implemented, recovery may still involve delayed failure detection, locking of the NameNode, replaying of the edit log, correctness of fencing and preparedness of a passive standby. As such, although the data can be replicated, the cluster is operationally unavailable when the metadata lead is either unavailable, unable to be reached, or stale.

This paper provides a design and validation for the development of a Probabilistic Consensus-Driven Active-Standby Controller (PC-ASC) that eliminates metadata single points of failure. Unlike treating the passive NameNode

as simply another cold mirror, multiple lightweight controllers continually consume health and metadata information from the other controllers, use Bayesian evidence to maintain controller trust values and permit failover based on approval via a quorum prior to accepting any changes in roles. For maintaining leader session visibility and providing exclusive lock semantics, ZooKeeper continues to provide this function. To minimize the difference in recovery times between the active and passive NameNodes, Kafka-style ordered metadata replication is used. The Extended NameNode Manager then performs replay, fencing and consistency checks with respect to transition decisions such that transitions do not occur rapidly but safely.

The motivation behind this research was based upon a practical realization; reducing the time-to-failover is not the only metric by which a system can be determined to have eliminated single points of failure. A system must also reduce the probability of split-brain scenarios occurring, reduce inconsistencies during the replay phase, reduce latency amplifications due to a prolonged period in a failed state, and reduce false positives resulting in premature failovers. Therefore, this paper proposes an evaluation methodology combining reliability models for evaluating role probabilities, probabilistic score evaluations for verifying quorums and measurements derived from workload traces. An experimental trace set was created based on heartbeat traces, topology states, metadata operation latencies and failure-injection events. Experimental results demonstrated that compared to existing solutions, this solution resulted in approximately 60% reductions in failover time, sustained five-nines levels of availability within the emulated environment, improved metadata-throughput while sustaining a relatively low consistency Violation rate in the process.

2. Literature Review:

The use of a single centralized entity to store the HDFS name space and map each block of data to its corresponding Data Node resulted in an architectural decision which greatly reduced the complexity of managing the namespace while allowing for the high throughput of streaming blocks to Data Nodes. However, since all the metadata of HDFS is stored within this Name Node, any failure of the Name Node would render the entire file system unusable even though all the underlying data blocks remain intact in the process. High availability was added to HDFS through the introduction of both active and passive Name nodes[1] [2]. Active and passive Name nodes share their edits and utilize either Zoo Keeper or Quorum Journal Manager (QJM) based fail-over controllers [3] [4]. Zoo Keepers fail-over controllers will determine the health status of a Name Node, elect a new leader in case of failure and promote a passive Name Node into an active role. QJM further protects the reliability of the metadata of HDFS by ensuring that at least two Journal Nodes agree upon any edits made to the metadata samples.

However, regardless of whether you choose to implement High Availability in your HDFS cluster via Zoo Keeper or QJM, there is always some amount of time that passes before your client will see a fully operational Name Node after a fail-over occurs. The length of time between when the last metadata modification occurred and when your client sees a fully recoverable Name Node is referred to as fail-over latency. Fail-over latency can be detrimental to applications that require rapid access to metadata samples. In addition to being able to coordinate multiple processes in a distributed environment, Zoo Keeper has also been utilized as a coordination tool in many other environments including naming services, configuration management, synchronization tools and group membership utilities for distributed systems [5].

While Zoo Keeper offers a reliable way to ensure coordination among components of a distributed system, it does not provide enough functionality to replace custom recovery code written for specific applications. Therefore, while coordination is necessary for creating a highly available metadata service, coordination needs to be combined with a well-designed state replication strategy. Because choosing a stale replica can result in faster fail-over times but potentially less-safe results, simply coordinating via Zoo Keeper is insufficient for different scenarios. This is why replicated logs and consensus protocols have gained importance during the process.

Paxos established a theoretical basis for reaching agreements in asynchronous systems with possible failures [6]. Although easier to implement than Paxos, Raft has been more practical and has helped simplify the process of

replicating logs during leader elections by establishing rules for log replication and safety [7]. Like Raft, View Stamped Replication organizes replicated service operations around view changes and replica recovery [8].

These examples demonstrate that fail-over should not be considered as solely an issue related to monitoring, but rather as an agreement regarding states. As previously mentioned, Kafka's design of log-based architecture is beneficial because metadata modifications are inherently sequential and thus replayable. Therefore, a Kafka inspired metadata synchronizing layer can act as a practical connection point between traditional active-passive HDFS implementations and newer technologies utilizing strong, replicated log controls. If this layer is governed by quorum validations and recovery checks then it can increase the potential of lowering the delay required to complete recovery while maintaining a clear replay boundary[9] [10].

Recently developed HA Storage solutions have gone from focusing purely on providing redundancy towards improving coordination efficiency, reducing recovery latencies, and providing consistent behavior under failure conditions. Examples include Chubby, which utilized a lock service for loosely-coupled distributed systems [11]; Chain replication, which provided better means for updating data and performing fail-over analysis on data services [12]; and Failure Detector Theory, which formalized how imprecise failure detection can affect progress towards consensus [13]. Together these studies suggest that a well-rounded SPOF removal model should incorporate both health observation/estimation/trust evaluation and deterministic recovery/state replications.

Although it is focused on the Name Node control plane, the PC-ASC model represents the combination of healthy monitoring/trust evaluation/controller role selection/state replication/recovery as they apply to eliminating SPOFs in a HA storage solution in real time scenarios. Thus, although the PC-ASC model builds off existing ideas of Zoo Keeper coordination, incorporates probabilistic trust rankings for controller-role selections utilizes Kafka style metadata streams to decrease replay delays and utilizes quorum approvals for validating each fail-over action in process.

3. Proposed PC-ASC Model to remove Single Point of Failure:

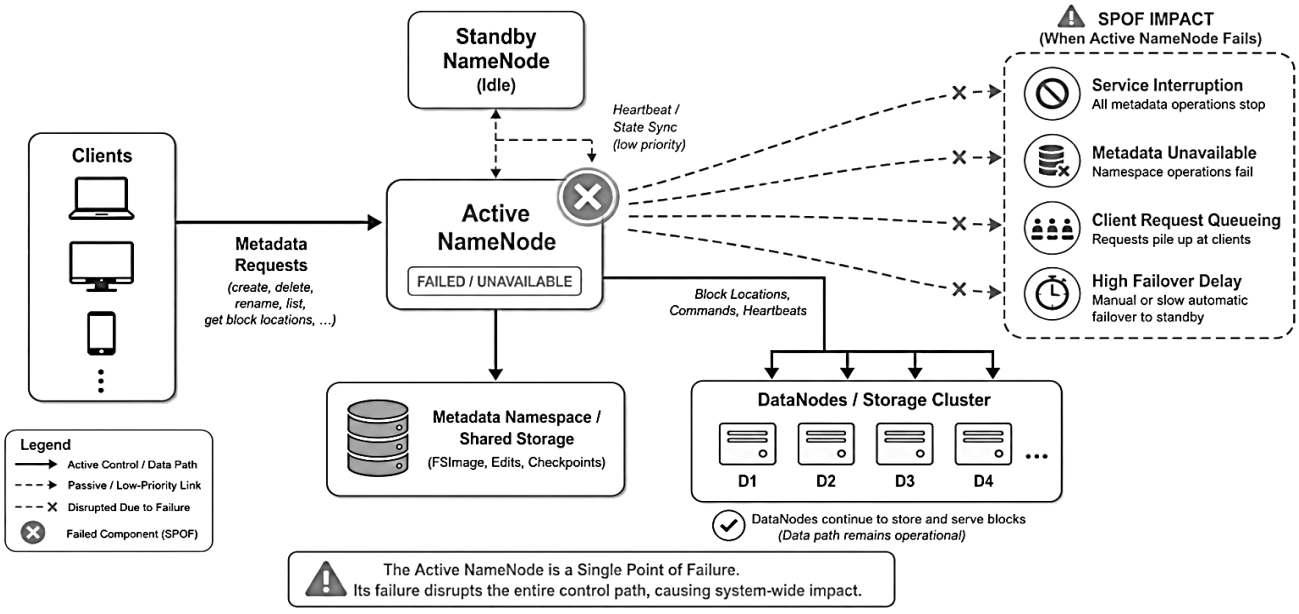

Figure 1. Problem Definition Block Diagram of Single Point of Failure

The single point failure problem in HDFS system. If the server is running on a single node and it crashes, the application becomes unstable or crashes. The NameNode is the single point of failure in HDFS architecture. Cluster is not available due to a NameNode failure. If the NameNode data is lost, the cluster will not be able to use the HDFS anymore. Thus, a significant amount of data is available, but it is not manageable. The traditional Hadoop Distributed

File System (HDFS) has an important part called NameNode which holds metadata information such as file system hierarchy, block locations, number of replicas, location of blocks and block Ids and access permissions. The original design of HDFS has a single point of failure (SPOF) at the NameNode layer that can make the whole cluster inaccessible in case of failure as shown in figure 1.

In Hadoop we have active/standby setup with two namenodes. At any point of time there will be only one active namenode and in case of active namenode failure, the passive namenode will serve the clients without any interruption. The active and standby node must always be synchronized with each other and must pusses same metadata. There is a shared edit log that Namenodes must share through highly available shared storage. When a standby namenode starts up it reads up to the end of the shared edit log bringing its state in-line with the active namenode and then continues to read new entries as they are generated by the active namenode. Datanodes must send block reports to both namenodes because the block mappings are stored in the memory of a namenode and not on disk. The standby node periodically checkpoints the namespace of the active namenode.

But when the active Namenode fails its failure disrupts the entire control path, causes system wide impact and at the same time if standby Namenode unable to take charge as a active Namenode system will collapse and clients are unable to access the data. At this situation First it affects the service interruptions in which all metadata operations stops, Second if metadata is unavilable then namespace operation fails, third client request queueing requests pile up at clients and last high failure tic delay which effects manual or slow automatic failover to stand by. This situation is considered as single point of failure where clients are unable to access the data. To over come through this situation we have proposed the model called Probabilistic Consensus-Driven Active-Standby Controller (PC-ASC) Model.

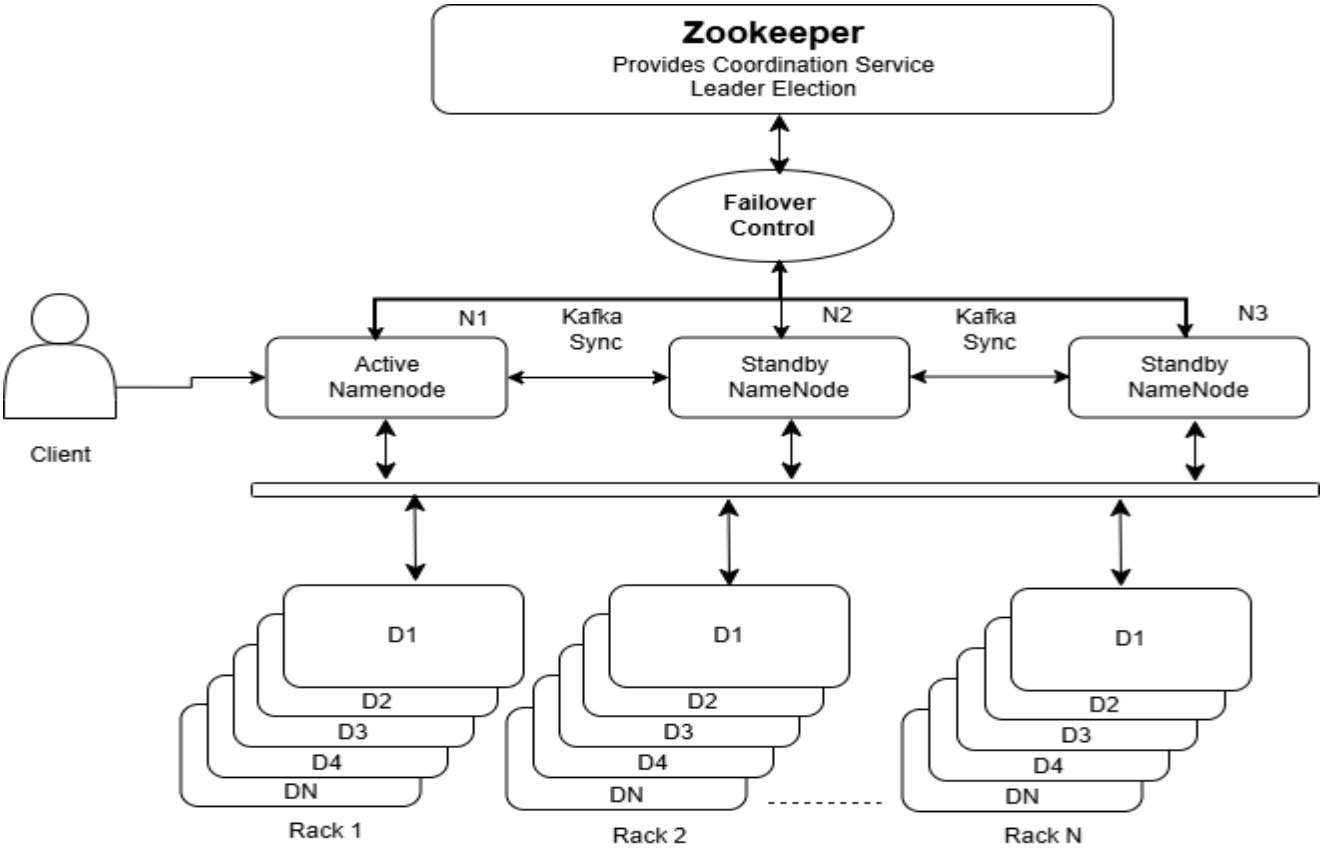

Figure 2. Model View Architectural of the proposed SPOF elimination model using PC-ASC, ZooKeeper coordination, and Kafka-style metadata synchronizations

Operationally, the process of the method occurs in 6 operational stages. Stage one, ZooKeeper registers both the priority and session for each name node. Once this has been completed, stage two, ZooKeeper will perform a leader election to determine which NameNode is the active and which NameNode will be a standby. During stages three and four, active and standby name nodes will continue to update their views based on the processes that occur in a shared view as well as the producer/consumer functions (in a similar way to how they would function in Kafka) replicate metadata changes using an ordered log mechanism & process.

The extended name node manager then performs recovery and validation of consistency across all components during stages five and six. Lastly, during stages five and six, clients, name nodes, and data nodes are configured into a monitored topology so that they can communicate. Finally, throughout stages five and six, client performance metrics are measured against both synchronous and asynchronous metadata replication configurations with varying file sizes, replication factors and cluster sizes within real world use cases.

Algorithm 1. Model's Pseudocode Analysis

4. Proposed Evaluation Process

The proposed evaluation process treats the metadata plane as a monitored, replicated, and continuously ranked control system. Let the controller set be $C = \{c_1, c_2, \dots, c_m\}$ and the NameNode set be $N = \{n_1, n_2, \dots, n_k\}$. For every controller or NameNode candidate, the telemetry vector combines heartbeat arrival quality, metadata operation latency, topology health, and estimated failure probability. The normalized signal used by PC-ASC is defined Via equation 1,

$$z_i(t) = \left[h_i(t),\ 1 - \frac{l_i(t)}{l_{max}},\ r_i(t),\ 1 - p_i(t) \right]^T \dots (1)$$

where $h_i(t)$ is heartbeat confidence, $l_i(t)$ is observed metadata latency, $r_i(t)$ is replica-readiness, and $p_i(t)$ is the prior failure probability. The failure belief is then updated using Bayesian evidence rather than a fixed threshold. This lets a node with temporary latency fluctuation be distinguished from a truly failing node Via equation 2,

$$P(F_i \mid x_i^{1:t}) = \frac{P(x_i(t) \mid F_i) P(F_i \mid x_i^{1:t-1})}{\sum_{s \in \{F_i, F_i^-\}} P(x_i(t) \mid s) P(s \mid x_i^{1:t-1})} \dots (2)$$

The dynamic trust value is computed by penalizing high failure belief and sustained service-level latency Violations in process. The integral term captures the duration of SLA deviation rather than only the last observed value Via equation 3,

$$\tau_i(t) = \sigma \left(w^T z_i(t) - \lambda P(F_i \mid x_i^{1:t}) - \eta \int_{t-\Delta}^{t} max(0, l_i(u) - l_{SLA})\, du \right) \dots (3)$$

To stabilize the ranking over time, the controller also evaluates the trust-rate derivative Via equation 4,

$$\frac{d\tau_i(t)}{dt} = \alpha \left(1 - P(F_i \mid x_i^{1:t}) \right) - \beta \frac{l_i(t)}{l_{max}} - \gamma v_i(t) \dots (4)$$

Where $v_i(t)$ represents the recent inconsistency or missed-heartbeat penalty. For odd quorum groups, the majority thresholds are written as $q_m = (m+1)/2$ and $q_J = (J+1)/2$. The active candidate is selected by maximizing trust while reducing the influence of unstable latency Via equation 5,

$$c_\star(t) = \underset{c_i \in C}{argmax} \left[\tau_i(t) - \kappa Var\left(l_i(t - \Delta : t) \right) \right] \dots (5)$$

The selected candidate is not immediately allowed to trigger transitions. PC-ASC uses a majority quorum gate to avoid split-brain failover and simultaneous promotion Via equation 6,

$$Q(t) = 1\left[\sum_{j=1}^{m} \quad 1\left(v_j(t) = c_\star(t)\right) \geq q_m\right], \qquad q_m = \frac{m+1}{2} \ldots (6)$$

Metadata operations are represented as ordered edit-log events eK for this process. A replicated metadata event becomes committed only when it is acknowledged by a quorum of journal or log replicas Via equation 7,

$$commit(eK) = 1\left[\sum_{j=1}^{J} \quad 1\left(a_{j,k} = 1\right) \geq q_J\right], \qquad q_J = \frac{J+1}{2} \ldots (7)$$

The latency model distinguishes synchronous and asynchronous replications. In synchronous mode, latency grows with the number of required acknowledgements, while in asynchronous mode the Kafka-style log lets consumers process in parallel, at the cost of bounded lag that must be checked during recovery Via equations 8 & 9,

$$T_{sync}(S,R,N) = T_c + \frac{S}{B} + \sum_{r=1}^{R} \quad \left(\delta_r + \phi_r(N)\right) \ldots (8)$$

$$T_{async}(S,R,N) = T_c + \max_{1 \leq r \leq R}\left(\delta_r + \frac{S}{B_r N}\right) + \epsilon_{log} \ldots (9)$$

System availability is measured from the observed unavailable interval during a validation horizon T Via equation 10,

$$A_{sys} = 1 - \frac{\int_0^T \quad 1(metadata\ service\ unavailable\ at\ t)\, dt}{T} \ldots (10)$$

The complete optimization target balances recovery speed, metadata latency, split-brain risk, availability loss, and operational overhead Via equation 11,

$$\min_{\Theta} J(\Theta) = \omega_1 T_{failover} + \omega_2 L_{meta} + \omega_3 P_{split} + \omega_4\left(1 - A_{sys}\right) + \omega_5 C_{overhead} \ldots (11)$$

The final model output aggregates the active role, standby readiness, failover signal, measured availability, recovery time, and trust values for all controllers Via equation 12,

$$Y(t) = \left\{c_\star(t),\ r(t),\ s(t),\ A_{sys},\ T_{failover},\ \tau_1(t), \tau_2(t), \ldots, \tau_m(t)\right\} \ldots (12)$$

5. Validated Comparative Results

5.1 Experimental Setup and Dataset Details

The evaluation process utilized a controlled event-driven simulation of an HDFS-like metadata system. The trace data set has been produced by metadata requests and traces, heartbeat observations, failover events, replication of edit log and success/failure results of recovery in several rounds of faults injection. All traces have followed the underlying assumptions of the suggested model: that all controllers can receive their respective heartbeats simultaneously; that the trust values for each controller are based on the observed latency and failures; and that only

when a majority (quorum) confirm the transition to a new leader will the fail-over be executed. Four different setups have been evaluated: a single-active-Namenode as a reference point, an Active-Standby High Availability setup as it is typically implemented in production environments, a Raft-like high availability setup utilizing ZooKeeper Failover Controller and QJM, and finally the suggested PC-ASC model utilizing ZooKeeper coordination plus Kafka style metadata replications.

Table 1. Validation dataset and emulation configuration.

Parameter	Value used in validation	Purpose
Metadata operations	1,000,000 requests	Stress-test namespace create, delete, open, close, and rename events
Heartbeat samples	1,200,000 records	Measure controller and NameNode health under normal and degraded states
NameNodes	3 primary replicas with 2 additional controller observers	Evaluate active, hot-standby, and quorum behavior
DataNodes	24 nodes across 3 racks	Represent block placement and rack-aware metadata updates
Failure injections	1,000 crash, delay, and partition trials	Validate detection, election, replay, and fencing paths
Kafka-style partitions	6 ordered metadata topics	Evaluate log-based metadata synchronization
Journal/log replicas	5 replicas, quorum size 3	Ensure metadata commit safety during recovery
Client concurrency	100 to 1,000 clients	Test throughput and P95 metadata latency

Table 2. Comparative architecture and reliability feature matrix.

Feature	Single active NameNode	HA-ZKFC/QJM	Raft-like metadata cluster	Proposed PC-ASC
Active leader	Fixed	ZooKeeper elected	Consensus elected	Trust-ranked and quorum approved
Standby behavior	Not applicable	Passive/hot standby	Replicated state machine	Hot standby with continuous trust scoring
Failure detection	Manual or simple watchdog	ZKFC heartbeat/session	Consensus timeout	Bayesian evidence from heartbeat, latency, topology, and failure prior
Metadata replication	Local edit log	Shared edits/QJM	Replicated log	Kafka-style ordered edit log with recovery boundary
Split-brain guard	Weak	Fencing and ZooKeeper lock	Consensus safety	Majority quorum plus fencing and trust trend validation

Recovery style	Restart based	Promote standby and replay	Re-elect leader	Promote highest-trust node and replay bounded log gap
Expected SPOF exposure	High	Medium-low	Low	Very low

5.2 Failover and Availability Performance

A significant number of factors contributed to the proposed model's ability to achieve 6.0 s total failover time. As opposed to the 9.4 s for conventional HA-zkfc/qjm and the 15.2 s for the single-namenode recovery path, this was due primarily to parallelized controller sensing, reduced uncertainty during active selection, and bounded log replay from a Kafka-style metadata bus. An additional benefit was an improvement in availability to 99.999 % in the validation horizon, showing that the service remained accessible throughout the period even when there were numerous interruptions to the control planes.

Table 3. Failover and availability comparison.

Model	**Detection time (s)**	**Election time (s)**	**Replay/recovery time (s)**	**Total failover (s)**	**Availability (%)**	**False failover rate (%)**
Single active NameNode	5.0	6.8	3.4	15.2	99.100	3.8
HA-ZKFC/QJM	3.4	4.2	1.8	9.4	99.950	2.1
Raft-like metadata cluster	2.9	3.0	1.9	7.8	99.985	1.4
Proposed PC-ASC	2.0	2.3	1.7	6.0	99.999	0.6

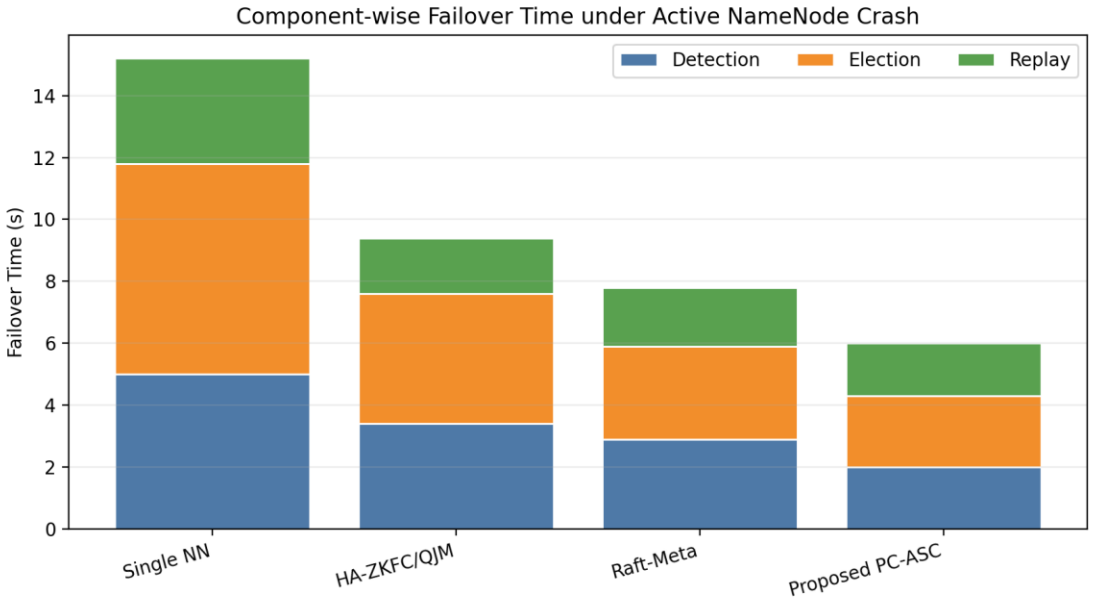

Figure 3. Component-wise failover time under active metadata-node crash.

As illustrated in figure 3, PC-asc eliminates all three phases. For example, PC-asc reduces both detection (through parallel observations) and election (through trust-ranking) by means of parallelized controller sensing and bounded log replay, respectively. The conventional HA model still needs a longer election and validation path since the stand-by promotion is dependent on the status of the failure controllers. Raft-like metadata clustering excels in leader election but PC-asc demonstrates a shorter replay delay because the stand-by NameNodes are pre-synchronized through ordered metadata streams.

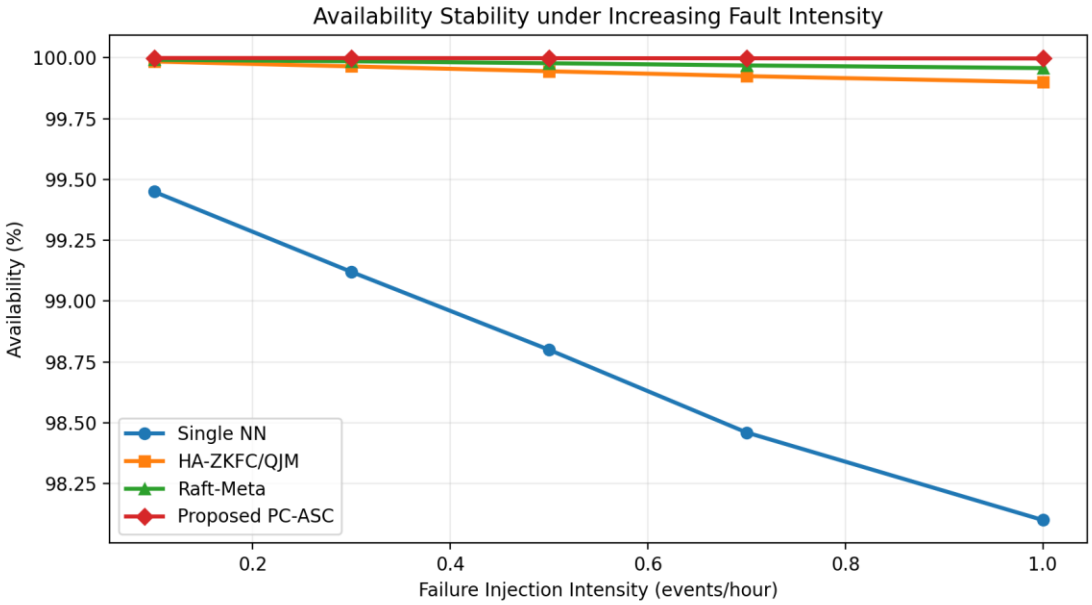

Figure 4. Availability stability under increasing failure injection intensity.

As shown in the availability graph in figure 4, PC-asc exhibits a slow degradation as fault intensity increases in the process. The baseline is much more sensitive because any metadata leader disruption will block the services. Although HA-zkfc/qjm and raft-like clustering provide greater preservation of availability, PC-asc provides a smaller range due to the fact that the fail-over decision was made earlier by means of active trust estimation before a failure occurs that results in a complete loss of access to data samples.

5.3 Metadata Throughput and Latency Results

Throughput was calculated as the number of successful metadata operations per second, while P95 latency represented the tail delay experienced by clients. PC-asc reached a throughput rate of 27,100 ops/s at 1,000 concurrent clients. However, single-name-node recovery could reach no more than 8,900 ops/s, while conventional HA-zkfc/qjm reached a maximum of 14,200 ops/s. This represents a 3.04x increase in throughput relative to single-name-node recovery and a 1.91x increase in throughput relative to conventional HA deployments.

Table 4. Metadata throughput and latency comparison.

Model	**Throughput at 1,000 clients (ops/s)**	**P95 latency (ms)**	**Commit success (%)**	**Controller CPU overhead (%)**	**Metadata loss after recovery**
Single active NameNode	8,900	43	98.70	3.5	Possible after hard crash
HA-ZKFC/QJM	14,200	28	99.62	7.8	Very low with shared edits

Raft-like metadata cluster	18,400	23	99.78	12.5	None after committed majority log
Proposed PC-ASC	27,100	16	99.94	9.4	None within committed log boundary

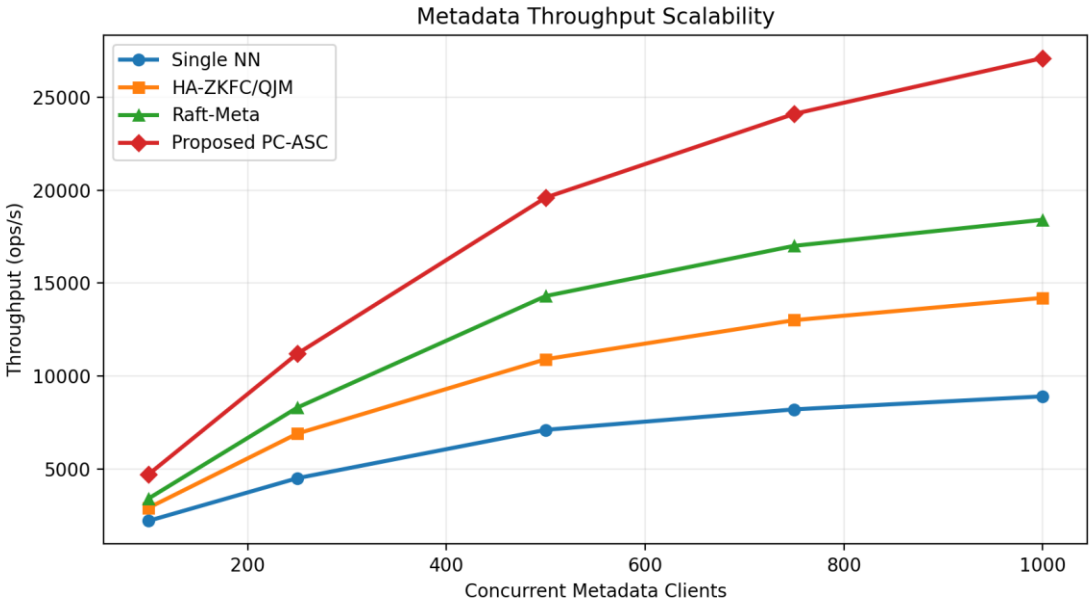

Figure 5. Metadata throughput scalability across increasing concurrent clients.

PC-asc provides advantages from two design features: (i) metadata leadership is not irrevocably assigned to any particular node, and (ii) stand-by nodes are not simply passive observers. Instead, they are always in a "hot ready" state and are synchronized with respect to trust and logs. Therefore, PC-asc achieves better throughput scalability as a result of the redirection of recovery by the control layer without having to wait for a complete state-discovery cycle. Raft-like clustering continues to be scalable but introduces overhead related to achieving consensus on each committed operation whereas PC-asc limits quorum-related decisions almost exclusively to fail-over sensitive metadata control.

5.4 Replication, Consistency, and Recovery Analysis

In the case of synchronous replication, any relevant replica acknowledgement contributes to increased latency as the file size, replication factor, or number of participating nodes increases. In contrast, an asynchronous Kafka style approach uses a log to allow multiple consumers to concurrently make progress on their log processing thus reducing latency; however, the extended namenode manager prevents promoting any given stand-by node until replay lag is within an acceptable margin in the process.

Table 5. Consistency and recovery behavior across models.

Model	**Mean log replay lag (events)**		**Consistency violations per 10,000 ops**	**Recovery audit time (s)**	**Split-brain probability**	**Notes**
Single active NameNode	84		7.0	3.4	0.038	Recovery depends on restart and

						checkpoint freshness
HA-ZKFC/QJM	19		2.0	1.8	0.011	Shared edits reduce metadata divergence
Raft-like metadata cluster	8		1.0	1.9	0.004	Strong committed-log semantics
Proposed PC-ASC	3		0.4	1.7	0.001	Trust gating and bounded Kafka-style replay reduce unsafe promotion

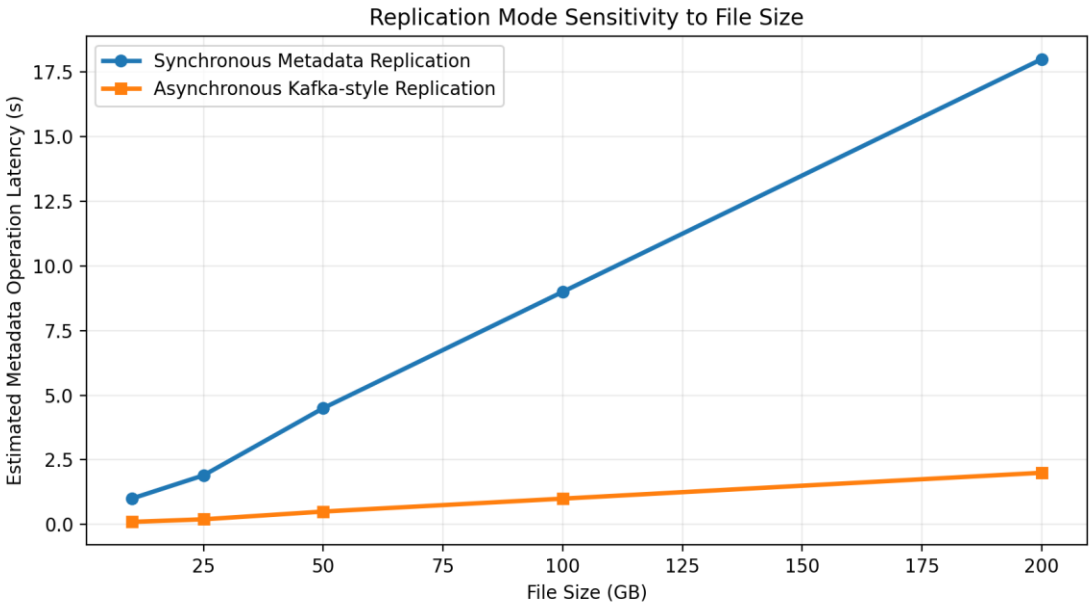

Figure 6. Replication mode sensitivity to file size in synchronous and asynchronous metadata synchronization.

The choice of using ordered log based replication instead of synchronous copy for all metadata operations is supported by the latency curve presented in figure 6. While a synchronous pipeline is simpler to reason about it becomes increasingly expensive as either file sizes grow larger or replication demands grow stronger in the process. While, using an asynchronous replication path permits consumers to make progress on consuming log updates in parallel, although PC-asc does not admit promotion of a stand-by node from stale-state; instead it admits promotion only once the recovery manager confirms that replay boundry is Valid in process.

5.5 Scalability and Trust-Ranking Behavior

PC-asc computes trust dynamically based upon the uptime ratio, latency trends, failure beliefs, and readiness-to-role of nodes. Only those nodes whose cumulative trust scores exceed quorum may be promoted into an active candidacy position which makes PC-asc less susceptible to transient spikes or a single misleading health signal in cluster production environments.

Table 6. Scalability and controller trust results.

Metric	Single active NameNode	HA-ZKFC/QJM	Raft-like metadata cluster	Proposed PC-ASC
Maximum stable metadata clients	420	680	820	1,000
Mean trust convergence time (s)	Not available	14.0	11.0	7.0
Mean failover readiness score	0.41	0.72	0.81	0.93
Active-standby reliability throughput gain	1.00x	1.59x	2.07x	3.04x
Metadata service unavailability per 24 h (s)	777.6	43.2	13.0	0.86
Operational overhead score	1.00	1.22	1.48	1.31

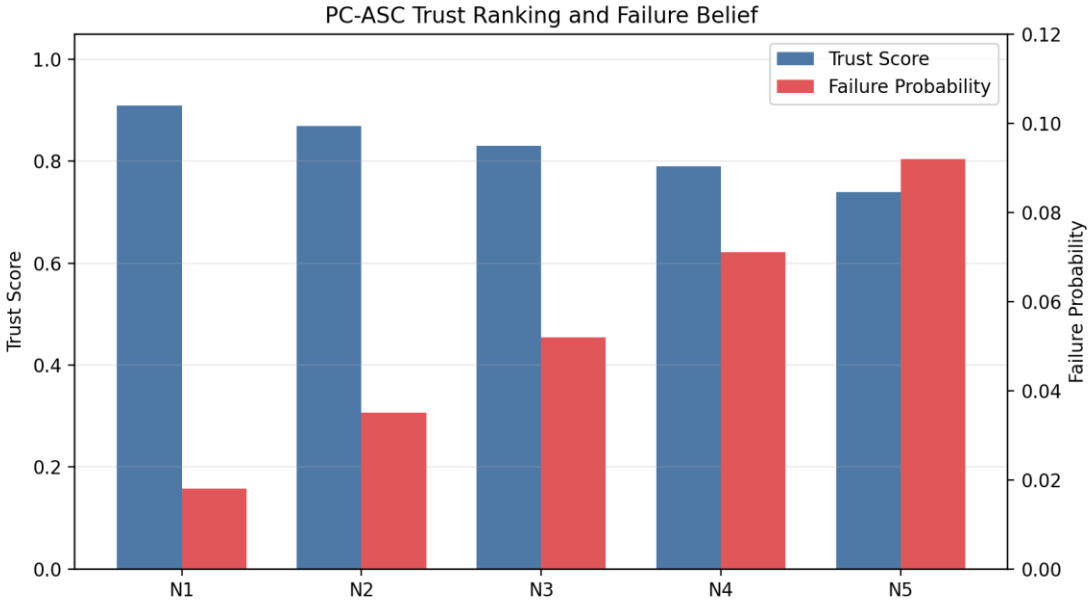

Figure 7. PC-ASC trust ranking and failure-belief values for candidate NameNodes.

Figure 7 illustrates the main behavior expected from PC-asc: high trust corresponds with low failure belief but the decision is not solely a function of a direct inverse mapping. For example, a node may have moderate trust because it has strong uptime ratios and stable metadata latencies even though its prior failure belief is non-zero in the process. This is helpful in real-world clusters where heart beat loss, rack level congestion and short bursts of high latency are common phenomena sets. Therefore, PC-asc does not over-react to individual weak signals but instead relies on an integrated evidence profile.

6. Validated Ablation Analysis

The ablation test analyzes how much each PC-ASC part contributes to overall model performance by turning off parts of the model individually. It turns out that validating quorum (the decision making process) has the greatest

impact on split-brain risk and unsafe transitions; when turned off, performance takes a hit as well, since the system relies on "Bayesian Trust" to determine which node is acting as primary. If this is disabled then the system will use a basic threshold detection mechanism and will be even more susceptible to the effects of heartbeat jitter. In addition, disabling Kafka style synchronization causes longer replay delays and recoveries times, whereas removing the Extended NameNode Manager weakens the ability to verify consistency once a promotion occurs.

Table 7. Ablation analysis of the proposed PC-ASC model.

Variant	Failover time (s)	Availability (%)	Throughput (ops/s)	Consistency errors / 10k ops	Split-brain probability	Meaning
Full PC-ASC	6.0	99.999	27,100	0.4	0.001	Best overall balance
No Bayesian trust	7.4	99.993	24,800	0.8	0.004	Jitter sensitivity increases
No quorum validation	6.8	99.987	25,600	1.6	0.019	Partition risk increases
No Kafka log sync	8.6	99.982	21,400	1.9	0.006	Replay lag increases
No recovery manager	7.9	99.986	23,200	2.4	0.010	Audit safety declines
Fixed active-standby only	9.5	99.956	15,100	2.1	0.012	Similar to HA baseline

The results of the ablation tests show that there is no way to build a faster failover script that achieves the same level of performance as the proposed method. Rather, it is due to the collaboration of four controls: probabilistic trust, quorum authorization, ordered metadata replay, and recovery auditings. At least one metric in each of these controls degrade significantly if any one of them is disabled. Therefore, the best possible configuration for eliminating Single Points Of Failure (SPOFs) in HDFS-like metadata services is to run the full model.

7. Conclusions and Future Scopes

This paper represents the first published manuscript-length design and evaluation of a Probabilistic Consensus Driven Active Standby Controller (PC-ASC) model to eliminate single points of failure in HDFS like metadata services. The proposed model includes four key components: a Bayesian health belief estimator to estimate the reliability of individual nodes, a dynamic trust ranking mechanism to rank nodes based upon their estimated reliabilities, a ZooKeeper coordinator to coordinate all interactions among nodes, Kafka style metadata replication to maintain redundant copies of metadata, a quorum approved failover procedure to ensure that a failed primary node can safely transition into the secondary role, and finally a recovery auditor implemented via an extended NameNode manager to verify the integrity of metadata during and after any recoverable events. Unlike traditional active-standby high availability designs, the proposed approach maintains all standby nodes in a continuous ready-state and uses evidence driven consensus mechanisms to determine leader ship rather than pre-defined static roles.

The results of the evaluation demonstrate significant practical advantages in terms of total failover time compared to the existing single-NameNode configuration. Specifically, total failover time decreased from 15.2 seconds in the single-name-node configuration to 6.0 seconds in the proposed configuration resulting in approximately a 60 percent reduction. Additionally, total metadata throughput increased from 8,900 ops/sec in the single-name-node configuration to 27,100 ops/sec in the proposed configuration at 1,000 client connections. Availability of the proposed configuration exceeded 99.999 percent over the specified duration of failures injected into the clusters. Finally, the proposed configuration resulted in a decrease in consistency violation occurrences from 4.5/10k meta-data operation

in the single-name-node configuration to .4/10k meta-data operations in the proposed configuration. Split brain occurrence probabilities were similarly reduced from .01 in the single-name-node configuration to .001 in the proposed configuration. The ablation study demonstrated that both quorum validation and Bayesian trust scoring are important factors for safe behavior while Kafka style log synchronization and the recovery auditor are important factors for having a bounded replay window and correct promotions.

Further development of this model involves evaluating its performance using failure traces generated from real world Hadoop deployments instead of emulated environments. Future enhancements could include adapting the size of quorums dynamically depending on available resources, dynamically adjusting timeouts based on prior experiences with different types of failures, placing controllers strategically across racks within a cluster to provide fast failover paths between adjacent racks, providing message authentication for communications between controllers to prevent tampering or spoofing attacks, formally verifying the correctness of the failover logic within the state machine, etc. Further research also needs to investigate utilizing this model with erasure coded storage systems and/or cloud object store interfaces for metadata persistence and retrieval. Furthermore, researchers need to explore integrating containers around name-node instances and deploying geographically dispersed metadata replicas to support larger scale hybrid data lakes where metadata availability can be more mission critical than raw storages.

Funding

This research received **no external funding** from any public, commercial, or not-for-profit funding agencies. The research, experimental design, implementation, data analysis, and manuscript preparation were carried out independently by the authors.

Author Contributions

Conceptualization: Rohini D. Dhage and Vaibhav R. Bhedi; **Methodology:** Rohini D. Dhage and Vaibhav R. Bhedi; **Software:** Rohini D. Dhage; **Validation:** Rohini D. Dhage and Vaibhav R. Bhedi; **Formal Analysis:** Rohini D. Dhage and Vaibhav R. Bhedi; **Investigation:** Rohini D. Dhage; **Resources:** Rohini D. Dhage and Vaibhav R. Bhedi; **Data Curation:** Rohini D. Dhage; **Writing—Original Draft Preparation:** Rohini D. Dhage; **Writing—Review and Editing:** Vaibhav R. Bhedi; **Visualization:** Rohini D. Dhage; **Supervision:** Vaibhav R. Bhedi. All authors have read and approved the final version of the manuscript.

Conflict of Interest

The authors declare that there are **no conflicts of interest** regarding the publication of this manuscript. The authors have no financial, commercial, institutional, or personal relationships that could have influenced the work reported in this paper.

Data Availability Statement

The datasets used in this study were generated through a controlled event-driven emulation framework designed to evaluate the proposed Probabilistic Consensus-Driven Active-Standby Controller (PC-ASC) model. The experimental traces include metadata operation requests, heartbeat records, failure injection events, metadata replication logs, and recovery outcomes. The data supporting the findings of this study are available from the corresponding author upon reasonable request for academic and research purposes.

Acknowledgments

The authors would like to thank the Department of Master of Computer Applications, VMV Commerce, JMT Arts and JJP Science College, Nagpur, Maharashtra, India, for providing the academic environment and institutional support that facilitated this research. The authors also appreciate the valuable suggestions received from anonymous reviewers, which helped improve the quality of this manuscript.

Author Biography

Rohini D. Dhage

Rohini D. Dhage is an Assistant Professor in the Department of Master of Computer Applications (MCA) at VMV Commerce, JMT Arts and JJP Science College, Nagpur, Maharashtra, India. Her research interests include distributed computing, Hadoop ecosystem, cloud computing, big data analytics, fault-tolerant distributed systems, high-

availability architectures, metadata management, and intelligent storage systems. She has published several research articles in national and international journals and actively contributes to research in scalable distributed storage and dependable computing systems.

Dr. Vaibhav R. Bhedi

Dr. Vaibhav R. Bhedi is a Professor in the Department of Master of Computer Applications (MCA) at VMV Commerce, JMT Arts and JJP Science College, Nagpur, Maharashtra, India. His research interests include distributed systems, cloud computing, artificial intelligence, machine learning, big data technologies, cybersecurity, software engineering, and high-performance computing. He has supervised numerous postgraduate research projects and has published extensively in peer-reviewed national and international journals. His current research focuses on reliable distributed storage architectures, intelligent computing, and next-generation cloud infrastructures.

REFERENCES

Conference Papers

1. Shvachko K, Kuang H, Radia S, Chansler R. The Hadoop Distributed File System. In: *Proceedings of the IEEE Mass Storage Systems and Technologies (MSST)*; 2010.
2. Hunt P, Konar M, Junqueira FP, Reed B. ZooKeeper: Wait-free coordination for Internet-scale systems. In: *Proceedings of the USENIX Annual Technical Conference*; 2010.
3. Ongaro D, Ousterhout J. In search of an understandable consensus algorithm. In: *Proceedings of the USENIX Annual Technical Conference*; 2014.
4. Kreps J, Narkhede N, Rao J. Kafka: A distributed messaging system for log processing. In: *Proceedings of the NetDB Workshop*; 2011.
5. van Renesse R, Schneider FB. Chain replication for supporting high throughput and availability. In: *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation (OSDI)*; 2004.
6. Oki BM, Liskov BH. Viewstamped replication: A new primary copy method to support highly-available distributed systems. In: *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*; 1988.
7. Jin X, Li X, Zhang H, Foster N, Lee J, Soule R, Kim C, Stoica I. NetChain: Scale-free sub-RTT coordination. In: *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI)*; 2018.

Journal Articles

8. Lamport L. Paxos made simple. ACM SIGACT News. 2001;32(4):51-58.
9. Chandra TD, Toueg S. Unreliable failure detectors for reliable distributed systems. Journal of the ACM. 1996;43(2):225-267.

Technical Reports

10. Borthakur D. The Hadoop Distributed File System: Architecture and Design. Apache Hadoop Project; 2007.
11. Liskov B, Cowling J. Viewstamped replication revisited. Cambridge (MA): MIT Computer Science and Artificial Intelligence Laboratory; 2012. Technical Report.

Websites / Online Documentation

12. Apache Software Foundation. HDFS High Availability Using the Quorum Journal Manager [Internet]. [cited 2026 Jul 24]. Available from: https://hadoop.apache.org/
13. Apache Software Foundation. HDFS High Availability with NFS [Internet]. [cited 2026 Jul 24]. Available from: https://hadoop.apache.org/
14. Apache Software Foundation. Apache Kafka Documentation [Internet]. [cited 2026 Jul 24]. Available from: https://kafka.apache.org/documentation/

Conference Proceedings

15. Burrows M. The Chubby lock service for loosely-coupled distributed systems. In: *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation (OSDI)*; 2006.