Journal of Information Systems Engineering and Management
2026, 11(2s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Modern Data Store Selection: A Quantitative Framework for Enterprise Architecture

Akanksha Mishra
Independent Researcher, USA

ARTICLE INFO

Received: 18 Feb 2026

Accepted: 22 Feb 2026

ABSTRACT

The contemporary enterprise architecture is experiencing challenges in the selection of data stores as the global data is growing exponentially, and the organizations are keeping high availability needs. The economic impact of storage infrastructure failures has grown to be very high, and of the many enterprises, most have incurred expensive downtimes which directly affect not only revenue but also customer satisfaction. Storage technology choices entail basic trade-offs in architecture between row-based and column-based forms of execution, where columnar systems have proven significantly superior to analytical workloads through late materialization, block iteration optimization and invisible join methods that scale up to provide exponential performance benefit. Distributed systems have to compromise consistency assurances and latency needs because even slight, measurably, decreases in response time have a negative impact on user interactions. The performance characteristics of the various categories of technologies are radically different, with relational and document databases displaying different performance with respect to connection loads as well as workload constituencies and memory resident caching systems demonstrating less than millisecond latencies with extraordinary throughput rates. At service level, the goals are converted into measurable error budgets limiting acceptable downtime levels, compelling architects to consider storage technologies in a worst-case recovery scenario instead of in an ideal performance. Modern business ventures are more prone to implementing polyglot persistence models where different special purpose data stores are used depending on the workload properties. Nonetheless, the pattern of architecture adds significant complexity to integration, with the organizations organizing data flow with many systems operating with high frequency rates, and reliability of pipelines and synchronization are the main elements of operation instead of the peculiarities of individual database performance.

Keywords: Data Store Architecture, Columnar Versus Row-Oriented Databases, Consistency Latency Trade-Offs, Polyglot Persistence Patterns, Service-Level Objective Frameworks

1. Introduction: Why Data-Store Choice is Now a Measurable Business Risk

In a world that is characterized by data explosion and relentless demands of high availability, the current system design exists in the context of reality where storage choices must cease to be based on a purely technical basis, but must be based on a critical economic decision. This transformation is illustrated in a very impressive way by the course of the world's volume of data. It is projected that the world data will grow between 33 zettabytes and 175 zettabytes by 2025, which is a compound annual growth rate of 61 percent. In this growing data environment, devices of the IoT will yield 90 zettabytes of data, and 49 percent of total data will be stored in the public clouds. The most impressive is the time aspect: by the year 2025, close to 30 percent of information will be in real-time consumption [1].
The economic implications of a poor storage system have also been equally measurable. Recent research indicates that 54 percent of the surveyed organizations reported that there were outages that led to costs of over 100,000, and 16 percent of the organizations reported over 100,000. These occurrences have been systematic in nature, as 55 percent of the respondents had one or more outages

Journal of Information Systems Engineering and Management
2026, 11(2s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

in the last three years. It is interesting to note that the only ones that were categorized as serious or severe were about 10 percent of such incidents [2]. These results prove beyond a reasonable doubt that the reliability of data-stores, recovery behavior, and performance traits directly translate into financial exposure and customer impact, and therefore storage technology selection process is a measurable risk management activity and not an imaginary technical choice.

2. Non-Negotiable Architectural Trade-Offs: Row Versus Column Orientation and Consistency Versus Latency

2.1 Row-Oriented Versus Column-Oriented Execution Models

The performance features that distinguish between the row-oriented and column-oriented database systems are based not only on the difference in simple storage layout but also on the differences in the nature of the execution model. Experiments on these architectural differences give objective performance data that clarifies why columnar engines are the most popular for analytical work, despite indexing choices or tuning work done on row-oriented options.

The method of late materialization, which stores the values in columns in compressed form as long as possible when a query is being executed, enhances the performance of a query by about 3X on the observed workloads. The fundamental nature of this optimization is the transformation in the nature of data flowing along the execution pipeline. Block iteration and associated optimizations add another level of improvement of the order of 1.5 times on top of late materialization. In queries with joins, the performance improvement of the invisible join techniques is about 50 percent on join-intensive analytical queries because the construction of a tuple is not done until it is required to do so [3].

Another aspect of the architectural benefit of columnar systems is shown by the compression efficiency that they can obtain. In controlled experiments, a column with 2,405 distinct values attained an average run length of about 25000, and with less than 64 kilobytes of storage. Such an extreme compression ratio is naturally the result of columnar organization, whereby repetitions within the same column can be coded much more efficiently than in row-based layouts, whereby disparate types of attributes are mixed together [3].

These measured outcomes are the reason why columnar engines are always preferred by analytical workloads. The performance benefits are multiplied in the various optimization layers, which form execution efficiency unable to be achieved by the row-oriented systems by use of indexing and caching only. In the cases where organizations are processing big analytical datasets, the differences in the execution models yield quantifiable throughput benefits that directly influence query completion times and the use of resources.

2.2 Consistency Versus Latency in Distributed System Design

In addition to the much-debated constraints of the CAP theorem, latency inherently affects the behaviour of the distributed system with effects that have direct business implications. Consistency trade-off studies show that when latency is increased by 100 milliseconds, the probability of interaction with the user can be significantly lowered and the rate of return visits decreased. This operational observation is the reason why most distributed databases knowingly compromise the guarantees of consistency to ensure sub-second response times under the stress load [4].

There is no binary latency-consistency trade-off, but a spectrum. There has to be a trade-off between systems maintaining a high level of consistency and the degradation of the user experience due to slower response times. In cases where the distributed databases replicate state data in the different nodes in order to guarantee consistency, network delay is added in every round of coordination, which gradually builds up to visible latency. The quantifiable threshold of the measured effects of 100 milliseconds on user behavior can be employed by an architect when analyzing consistency models.

This trade-off is acute, especially in geographically dispersed systems where physical delays are inevitable due to network distances. Organizations that have global user bases need to choose between immediate response times, with eventual consistency, or high latency to ensure high consistency. The

Journal of Information Systems Engineering and Management
2026, 11(2s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

business ramifications of this choice do not just end with the user experience but also with the requirements of data correctness, regulatory compliance factors, and operational complexity. Financial transaction systems might need high consistency in the face of latency costs, and content delivery systems might need to be aggressive in response time with weak consistency guarantees [4].

Optimization Technique	Performance Improvement	Description
Late Materialization	3×	Keeps column values in compressed form during query execution
Block Iteration	1.5×	Additional optimization beyond late materialization
Invisible Joins	50%	Performance gains in join-heavy analytical queries
Compression Efficiency	<64 KB storage	Column with 2,405 unique values, average run length ~25,000
Latency Impact	100 ms	Additional latency significantly reduces user interaction probability

Table 1: Non-Negotiable Architectural Trade-Offs [3, 4]

3. Technology Categories in Practice: Benchmark-Backed Performance Numbers

3.1 Relational Versus Document Database Performance Characteristics

Comparative empirical performance results between relational and document-oriented databases have shown that the control of connection processing and workload structuring prevail over the real throughput results. Comparison benchmarks of these architectural methods under controlled circumstances give actual figures, which guide the decisions of technology choices.

With a 50/50 read/write load and 50 connections per load, the measured throughput shows that database types vary significantly. PostgreSQL attained 2,569 operations per second, whereas MongoDB had 924 operations per second. Once connection pooling was added to PostgreSQL, the throughput rose to 2,779 operations per second. There is a drastic change in the performance profile with the increase in the number of connections. The relational database, which was not pooled, sustained 300 connections and was able to support 121 operations per second, whereas the document database supported 828 operations per second. Nevertheless, the relational database that used connection pooling had reached 2,860 operations per second, which shows that connection management architecture is a core determinant regarding scalability behavior [5].

Even greater workload performance differentials can be observed with in-memory testing of 50 connections. At 95/5 read heavy workload, the relational database was 2,433 operations per second faster than the document database, 96 operations per second, which was 25.3 times faster. The relational database could process 2,211 operations per second, and the document database could process 51 operations per second under a balanced work load of 50/50, which gives a speedup of 43.4x [5].

These measurements show that workload properties and connection architecture generate performance differences that are many times bigger than those that can be attributed to differences in the core database engine. The high and low connection counts reversal in the dramatic performance testifies to the fact that the operational factors tend to prevail over intrinsic architectural benefits. Organizations that choose between such technologies should profile their workload patterns, specifically, and not based on the general architecture preferences.

Journal of Information Systems Engineering and Management
2026, 11(2s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

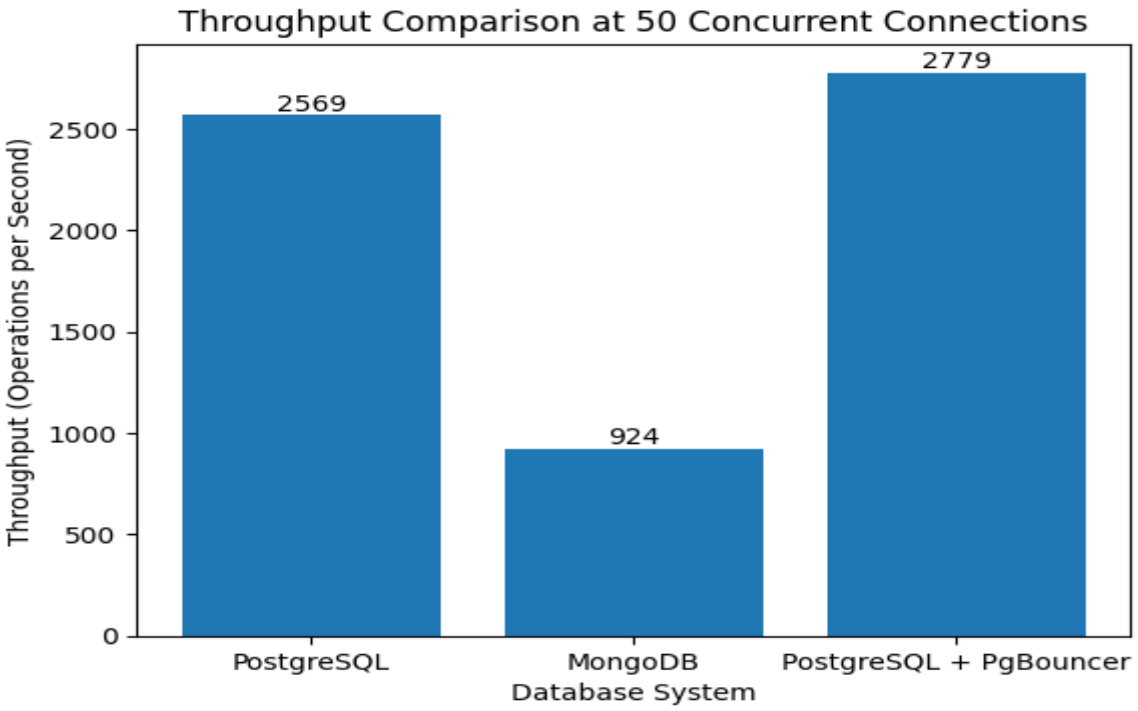

Fig. 1. Throughput Comparison of PostgreSQL and MongoDB at 50 Concurrent Connections [5]

3.2 In-Memory Caching System Performance Characteristics

Structural latency and throughput benefits offered by memory-resident data stores make particular architectural patterns possible. In-memory caching system benchmark measurements are a set of performance attributes of the system, measured under controlled standard test conditions.

In the case of SET operations, a measured performance of 180,180 requests/second and a median of 0.143 milliseconds gave the impression of mediocre performance. LPUSH calls had a rate of 188,323 requests per second with a median latency of 0.135 milliseconds. These tests were performed on 50 concurrent clients in 5,000,000 total requests, which gives statistically significant performance on-bases [6].

Sub-milliseconds of latency measurements would indicate the architectural benefit of removing disk I/O from the critical path. The access time on all-memory systems is many orders of magnitude lower than that of disk-based systems. The throughput figures show the magnitude with which system resources resident in memory can be used. This level of performance allows architectural designs in which there are caching layers, which absorb read traffic that would otherwise congest the backend databases.

The operational implications of the memory-resident storage should be considered by organizations that utilize these performance characteristics. The durability of data, the cost-effectiveness of the capacity planning (in terms of memory versus disk storage), and the problem of the cold-start (when a new instance of the cache is started) should be considered in the context of failover strategies. These complexities of operation are worth the measured performance benefits when the workload requires architecture-level sub-milliseconds latency and extreme throughput [6].

Journal of Information Systems Engineering and Management
2026, 11(2s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

4. A Decision Framework with Quantitative Service-Level Objective Targets

4.1 Availability and Error-Budget Mathematics

Service-level objectives are used to convert abstract reliability goals into specific operational targets by using clear downtime allowances. The percentages of availability are directly proportional to the allowed outage times, which form the quantitative models of assessing the data-store reliability needs. A service with a 99.99 percent availability allows 52.56 minutes of unavailability a year. This yearly budget should be able to cover all the planned maintenance, unplanned outages and periods of poor performance. At a reduced availability capacity, 99.9 percent availability will translate to around 43 minutes of unavailability per month [7], [8].

These calculations give the architects tangible measurements to determine whether the workings of a data-store meets the aims of the service levels. Structurally, systems that have several such events each year cannot achieve high levels of availability despite their normal operation performance.

The cascading failure compounded effect can also be seen in the error-budget framework. Once applications-layer timeouts, cache invalidations, and client retries are caused by a failure in a storage system, the actual downtime can be even greater than the recovery time of the storage system. To compute the compatibility of storage technology options with their availability goals, organizations have to consider these propagation effects [7], [8].

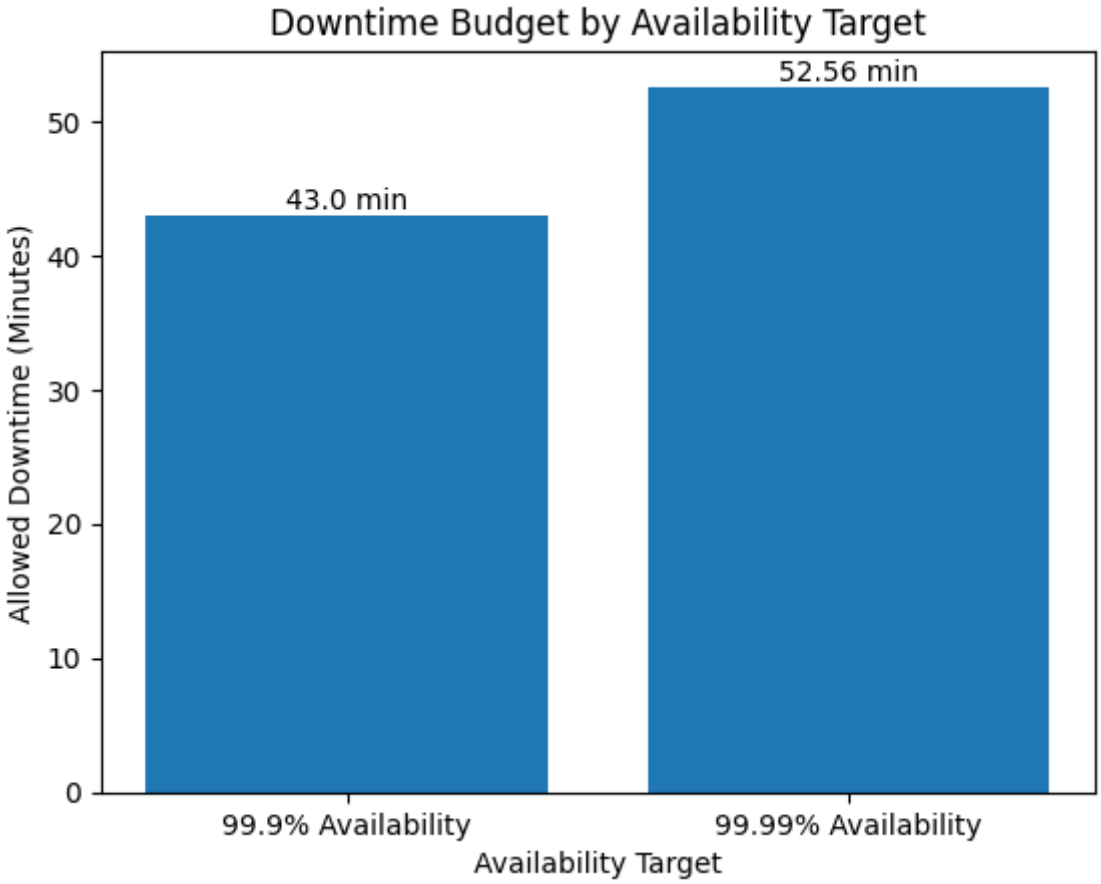

Fig. 2. Downtime Budget as a Function of Availability Target [7, 8]

4.2 Translating Service-Level Objectives into Storage Acceptance Criteria

Storage technologies should be able to meet the specific recovery-time and latency criteria to ensure that they do not eat excessive shares of available error budgets. A system that aims at 99.99 percent availability with an annual downtime of 52.56 minutes cannot support storage technologies with

Journal of Information Systems Engineering and Management
2026, 11(2s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

rebuild times of hours in the event of the failure of a node. Only one such accident is enough to drain the whole annual error budget.

This limitation compels architects to consider storage systems in the worst-case scenario instead of the best-path performance. Structural reliability is posed by technologies that have a high throughput in normal operation and have a long recovery period in the event of failures. These trade-offs are explicitly defined in the quantitative error-budget framework, and organizations are able to dismiss technologies that fail to comply with the recovery-time requirements despite their other benefits [7], [8].

5. Polyglot Persistence and Operational Scaling Considerations

5.1 Evidence of Widespread Multi-Database Usage Patterns

However, modern software systems are becoming more dependent on various specialized data stores instead of making a single database technology standard. This polyglot persistence trend is measured in survey data that analyzes technology adoption among thousands of survey participants.

The relational databases are higher in the surveyed developers, with one used by 55.6 percent of respondents and the other by 40.5 percent. The application of embedded databases is 37.5 percent, with in-memory cache at 28.0 percent. The usage of document databases is found to be 24.0 percent, search engines 16.7 percent, and managed NoSQL services 9.8 percent. More focused technologies, such as cloud data warehouses at 4.1 percent, columnar analytics databases at 2.4 percent, and graph databases at 2.6 percent, are used in particular applications [9].

These adoption patterns affirm that the adoption of databases in an organization is based on the workload features and not the architectural homogeneity. The workloads related to analytics are being drawn towards columnar systems, search capability is based on specialised search engines, caching layers utilise in-memory stores, and transactional systems are being based on relational databases. The varied adoption rates indicate the sophistication of data architecture, which is where various data access patterns are given storage technologies with the purpose in mind.

5.2 Integration and Data-Movement Cost Considerations

Polyglot persistence helps in selecting the most optimized storage, but it adds complexity of integration, which tends to dominate the total cost of ownership. Surveys conducted on the data integration practices demonstrate the extent of multi-system coordination in the current architectures.

It has been decided that 59 percent of the organizations are using 11 or more different data sources, and almost 25-percent are using more than 50 different data sources. The time aspect of integration is also very challenging, as 72 percent of organizations transfer data more than a single time per day across systems [10].

These numbers point to the fact that reliability, governance, and synchronization of pipelines are often the highest cost in polyglot architectures, instead of the performance of each database or database licenses. Organizations are required to sustain data movement infrastructure, ensure a coherent schema evolution between systems, pipeline failures, data freshness, and synchronization conflicts. The operational load increases with the systems of different systems and the frequency of data moving.

This is complicated by the fact that data consistency between systems is also an issue. Propagation of changes in many databases should have referential integrity, partial failure handling, and eventual consistency windows. To coordinate these movements, organizations deploy change data capture systems, message queues, and orchestration platforms. This integration infrastructure is usually more expensive and complex than the cost of the separate databases [10].

Journal of Information Systems Engineering and Management
2026, 11(2s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Technology Category	Adoption Rate	Usage Context
Relational Database (Type 1)	55.6%	Transactional workloads
Relational Database (Type 2)	40.5%	Transactional workloads
Embedded Database	37.5%	Local storage
In-Memory Cache	28.0%	High-performance caching
Document Database	24.0%	Flexible schema workloads
Search Engine	16.7%	Full-text search
Managed NoSQL Service	9.8%	Cloud-native applications
Cloud Data Warehouse	4.1%	Analytics at scale
Columnar Analytics Database	2.4%	Specialized analytics
Graph Database	2.6%	Relationship-oriented data

Table 5: Polyglot Persistence and Operational Scaling [9, 10]

Conclusion

The selection of an enterprise data store has changed over time since the technical preference to store is no longer assumed, as the risk management is quantifiable, where the storage decision has a direct financial cost because of its effect on system availability and performance. The architectural difference between the columnar and row-oriented systems causes the actual performance differences that are quantifiable and render the columnar engine practically required by the analytical workload, and the transactional systems are supportive of the row-oriented systems. The distributed database architectures need to work around the natural conflict between consistency guarantees and low-latency responses, and even minor latency increments are proven to be jeopardizing user behavior and business results. Comparison of performance empirically shows that operational aspects such as connection management and workload composition can significantly override intrinsic architectural benefits, whereas the fact that the memory-resident systems support a completely different architectural pattern with their extraordinary throughput and sub-millisecond response times. Service-level objectives are used to give the quantitative reference point of determining whether storage technologies can deliver the availability targets, where the percentage uptime targets are converted to tangible downtime resources, limiting acceptable failure recovery times. Polyglot persistence is an aspect that is ubiquitous in architectural maturity when organizations align specialized storage technologies with particular workload needs instead of achieving monolithic consistency. This specialization provides the optimal execution of a variety of access patterns but creates significant complexity of operation due to the need to move data, coordinate pipes, and the challenges of cross-system synchronization. Total cost of ownership in polyglot architectures is progressively clustering in integration infrastructure, as opposed to individual database licensing or performance, and data movement reliability and governance are the key success factors in the current data architecture strategy. Organizations implementing these quantitative frameworks should establish regular architectural review cycles that reassess storage technology choices against evolving workload patterns and measured performance metrics, ensuring that initial technology selections remain aligned with current operational requirements and business objectives.

References

[1] Andy Patrizio, "IDC: Expect 175 zettabytes of data worldwide by 2025," Network World, 2018. https://www.networkworld.com/article/966746/idc-expect-175-zettabytes-of-data-worldwide-by-2025.html

[2] Uptime Institute, "Annual Outage Analysis 2024 – Executive Summary," 2024. https://datacenter.uptimeinstitute.com/rs/711-RIA-145/images/2024.Resiliency.Survey.ExecSum.pdf
[3] Daniel J. Abadi et al., "Column-Stores vs. Row-Stores: How Different Are They Really?" ACM SIGMOD, 2008. https://www.cs.umd.edu/~abadi/papers/abadi-sigmod08.pdf
[4] Daniel J. Abadi, "Consistency Tradeoffs in Modern Distributed Database System Design," IEEE Computer Society, 2012. [Online]. Available: https://jmromer.org/papers/2010-consistency-tradeoffs-in-modern-distributed-database-system-design.pdf
[5] EnterpriseDB, "Performance Benchmark Postgresql / Mongodb". [Online]. Available: https://info.enterprisedb.com/rs/069-ALB-339/images/PostgreSQL_MongoDB_Benchmark-WhitepaperFinal.pdf
[6] Redis, "Redis Benchmarks," [Online]. Available: https://redis.io/docs/latest/operate/oss_and_stack/management/optimization/benchmarks/
[7] Marc Alvidrez et al., "Embracing Risk," Google. 2015. [Online]. Available: https://sre.google/sre-book/embracing-risk/
[8] Adrian Hilton and Alec Warner, "Understanding error budget overspend—CRE life lessons," Google Cloud, 2018. [Online]. Available: https://cloud.google.com/blog/products/gcp/understanding-error-budget-overspend-cre-life-lessons
[9] Stack Overflow, "2025 Developer Survey," 2025. [Online]. Available: https://survey.stackoverflow.co/2025/technology/
[10] Fivetran, "Modern infrastructure helps data engineers deliver maximum value," 2021. [Online]. Available: https://www.fivetran.com/blog/modern-infrastructure-offers-value-to-data-engineers