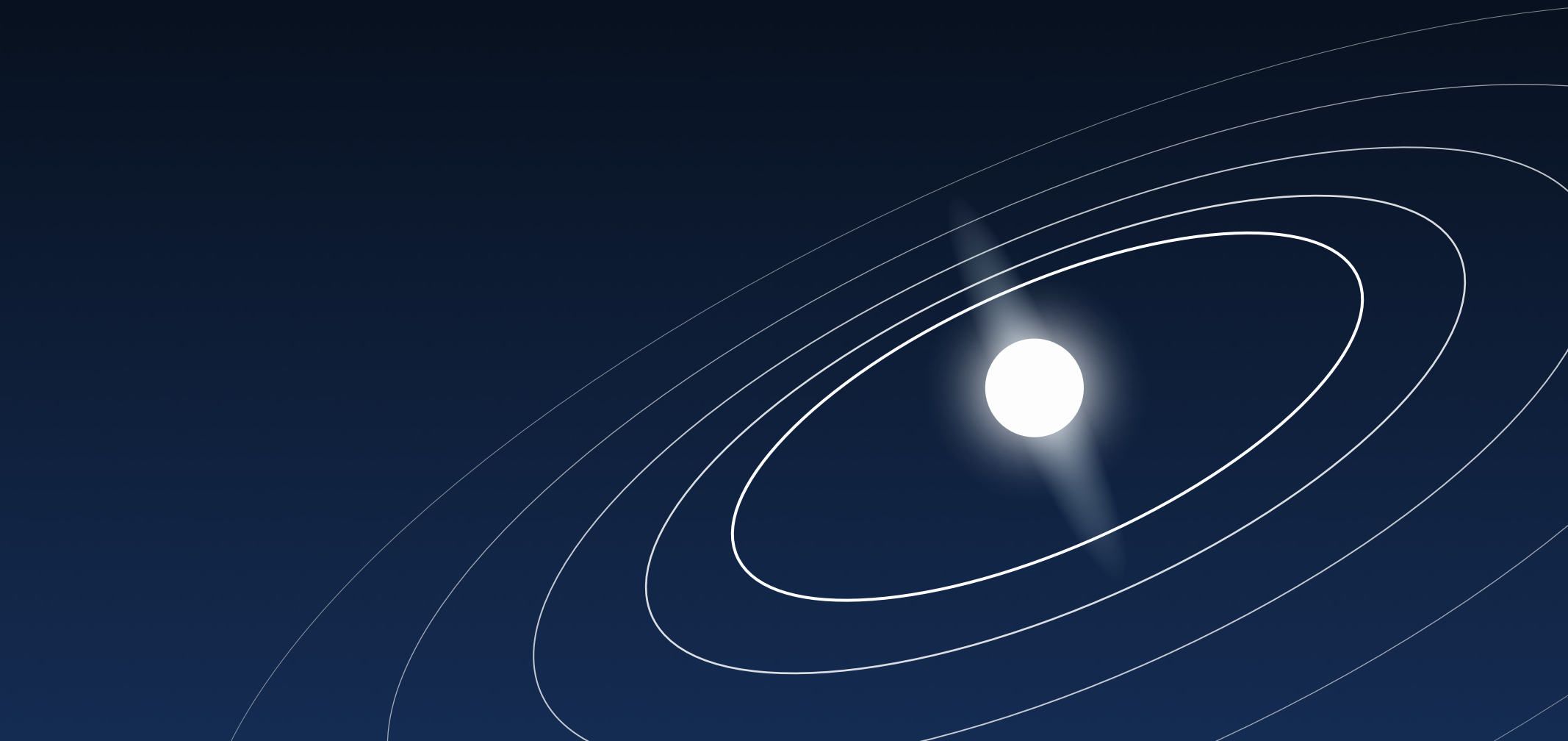
Aerospike | E-BOOK

Get ahead of the pack

Building AdTech for speed and scale



Introduction

Act fast or fall behind. In AdTech, every decision counts. Platforms process thousands of decisions per second, each in milliseconds, all while handling massive data volumes, and every transaction is a race against competitors.

AdTech engineers operate under intense pressure. The right database architecture determines whether your platform scales or stalls. Leading AdTech companies rely on Aerospike to overcome data bottlenecks and achieve true hyperscale.

This e-book covers:

- Technical challenges of building and maintaining AdTech at scale
- How data gravity sets a trap that kills AdTech performance
- Lessons learned from building solutions for top industry players
- Aerospike for AdTech: Real-time, petabyte-scale data infrastructure

1

Core technical challenges in modern AdTech architecture

AdTech platforms operate at hyperscale (billions of requests per day) and hyperspeed (sub-millisecond response requirements), where engineering decisions directly impact revenue and market position.

Developers face numerous technical challenges:

Hyperscale data ingestion and storage

AdTech platforms operate at petabyte-scale throughput, ingesting billions of daily ad requests: The Trade Desk processes over 10 million queries per second. Systems ingest millions of bid requests, impressions, clicks, and conversions every second, peaking unpredictably during events like Black Friday or the Super Bowl. There's zero tolerance for dropped events. Pipelines have to absorb traffic spikes in real time, across regions, with no lag and no excuses.

Developers must tune ingestion flows like high-frequency trading systems—fast, precise, and lossless. Schemas are chaos. Every supply-side platform (SSP), demand-side platform (DSP), or tracker sends a slightly different flavor of JSON, Avro, or Protobuf. Fields disappear, mutate, or show up undocumented. Ingestion must be fault-tolerant, schema-aware, and flexible enough to survive constant change without breaking downstream jobs.

Storage is a ticking cost bomb. Petabytes of logs pile up fast. Teams want raw logs for audits, summaries for dashboards, and clean samples for modeling. You can't store everything forever, so developers have to pick their battles: compress, aggregate, archive, or drop. Every decision affects what insights you can get or what data you'll never recover.

Low-latency execution constraints

Slow code means lost bids and lost revenue. Development teams never stop measuring execution time and optimizing code.

Real-time bidding (RTB) systems demand sub-millisecond decision windows, with <10ms P99 latency thresholds to stay competitive in auctions. RTB auctions are typically won in 100-120 milliseconds, but the actual response window (set by tmax) may be longer or shorter depending on the SSP. When a user lands on a webpage or opens an app, the publisher's site triggers an ad call, which the SSP forwards to multiple DSPs as a bid request, usually in JSON format. The DSP's backend must deserialize this request, run targeting logic, apply brand safety and fraud filters, look up pricing rules, select the appropriate creative, and return a bid response, all without exceeding the time budget.

Developers try many different techniques to meet this narrow window, including complex in-memory caching schemes, high-performance NoSQL data structures, and non-blocking asynchronous code. Disk reads, garbage collection pauses, or external API calls can easily cause timeouts and lost bids. They deploy services close to major data centers or edge locations to reduce network latency and use horizontal scaling to handle high request volumes. Any code that introduces a delay directly reduces win rate and revenue.

Regulatory-driven system complexity

AdTech platforms constantly juggle the demands of GDPR, CCPA, and LGPD compliance while trying to keep response times under five milliseconds; a real headache for engineers. Instead of clean, maintainable architectures, teams often end up bolting region-specific conditional logic into their ingestion pipelines just to stay compliant, which adds latency and turns data flows into fragile, overcomplicated systems. The real pain shows up at query time, where applying opt-out flags and data retention policies mid-query can't slow things down or break SLAs, but traditional application-layer filters and post-processing tack on extra milliseconds. Privacy-preserving techniques like differential privacy further tax the system, slowing attribution models by as much as 30–40%.

Distributed system anti-patterns

Legacy architectures trap engineers in reactive maintenance. For example, when you stack Memcached for caching on top of Couchbase for persistence, you introduce cache/persistence drift that can lead to production outages. Partitioning by user ID or region just makes things worse, creating unbalanced clusters and driving hardware usage up. Manual, ad hoc replication between data centers can't keep up with regional traffic spikes; consistency degrades, and the system loses reliability. These anti-patterns turn your engineering team into infrastructure janitors instead of feature builders.

Integration debt

Dated platforms built on MongoDB or HBase force engineers to build expensive translation layers, adding 15-40ms of latency to every transaction, a killer for real-time AdTech. Protocol fragmentation only makes things worse. Supporting OpenRTB 2.6, IAB TCF 3.0, and a mess of proprietary SSP APIs turns integrations into a tangle of unmaintainable connector code. As supply path optimization (SPO) continues to advance, getting closer to the supply or demand continues to be in tight focus. Creating and maintaining a competitive edge now requires a robust tech stack capable of operationalizing data with speed and scale.

How data gravity can lead to vendor lock-in

Data gravity occurs when a platform accumulates such a large volume of data and associated tooling that moving data out becomes technically complex and financially impractical, leading to vendor lock-in. As more data and services become concentrated in one system, it becomes increasingly difficult to extract and repurpose that data in other environments. This lack of portability restricts integration options, reduces transparency, and makes switching providers costly or even unfeasible. While centralizing data can enhance efficiency, such as modeling user journeys or powering real-time decisions, it often comes at the expense of flexibility, effectively trapping teams within a single ecosystem.

Please see the next section, "What is data gravity," for a deeper discussion of the topic.

Engineering impact

Most AdTech teams burn over 70% of their time keeping legacy systems running instead of building new features or improving the product. When you spend your days patching up brittle integrations, chasing outages, and tuning outdated infrastructure, you lose momentum and stall innovation. To break out of this cycle, you need to overhaul your architecture: consolidate fragmented systems, automate away operational grunt work, and align your stack with the demands of hyperscale AdTech. If you ignore these problems, data gravity will keep draining your resources and slow your growth to a crawl, while your competitors race ahead.

2

What is data gravity?

Data gravity, introduced in the previous section, can be a double-edged sword. When you manage data efficiently, data gravity works in your favor. Systems stay responsive, and infrastructure scales cleanly. However, once complexity and inefficiency take over, data gravity becomes a trap. It slows growth, eats resources, and turns your architecture into a maintenance burden instead of a growth engine.

How data gravity becomes a trap

A single factor rarely causes data gravity to spiral out of control and become a trap. The problem usually emerges from a combination of technical decisions and compounding issues over time.

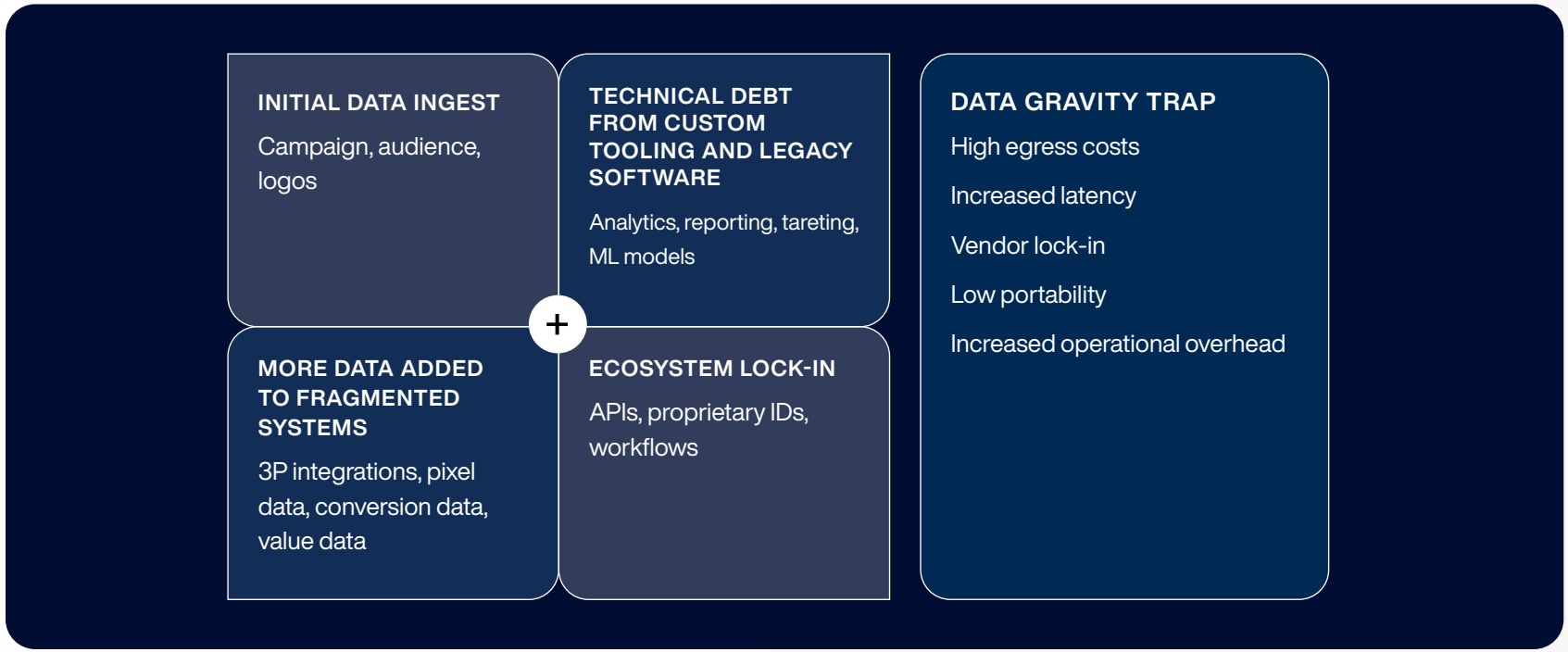
Common causes include:

- **Technical debt** piles up when you leave inefficiencies unaddressed. Each workaround or shortcut you take multiplies instability and unpredictability, especially as your data and traffic grow. When you bolt on complexity, like split-cache and persistence layers, you might solve a short-term problem, but you introduce drift and operational headaches that scale with your data.

- **Legacy systems** that weren't built for the cloud or distributed scale can be forced to fit, but they end up draining time and resources. You spend more effort keeping them alive than building new features or improving performance.
- **Fragmented data architectures** add to the pain. If your data isn't unified, every transaction requires extra normalization and standardization, which increases I/O latency and slows everything down.

Don't feed data gravity—rethink, don't scale blindly

If you ignore these issues, total cost of ownership (TCO) grows unpredictably as your data grows. Operational overhead spirals, SLAs slip, and the business case for change becomes critical. Throwing more hardware or money at the problem might buy you time, but it won't fix the underlying issues. The only way out is to re-architect for scale, leveraging unified data while avoiding data gravity.



3

Case studies in accelerating AdTech

Switching databases is a lot like performing a heart transplant. The process can feel risky and uncomfortable, but it often marks the difference between breaking through scale barriers and falling behind.

Leading AdTech companies have made the jump to Aerospike's real-time database to break free from the constraints of data gravity and accelerate their growth. Here's how they recognized the need for change, why they chose Aerospike, and how they delivered real results.

Criteo

Criteo's ad platform matches content to users over 950 billion times a day, powered by more than 1.2 trillion records and 300+ TB of data. Nearly 200 applications depend on real-time access, with many running under strict sub-50ms SLAs. As demand grew, the system started to buckle.

Instead of building new features, engineers spent their days managing two core data systems: Memcached for low-latency caching and a community version of Couchbase for persistence and backup, spread across 3,200 physical servers in six data centers. Latency spikes and outages became more frequent, burning out the team. Minor inefficiencies scaled with both data volume and app count, turning into systemic problems.

The system worked...right up until it didn't. Minor inefficiencies became systemic and scaled alongside application count and data volume.

The core architectural issue was clear: bolting a cache onto a persistence layer works until it doesn't. Memcached required huge RAM, offered no persistence, and scaled poorly. Couchbase brought complicated failover, variable latency, and growing ops overhead. With 175+ apps running mixed batch and streaming workloads, instability increased. Incidents hit six per year, SLAs slipped, and keeping the platform alive became a full-time job.

Migrating, strategically

When Couchbase support ended, the team saw an opportunity to rethink the stack. Rather than swap out one component (Couchbase), they rebuilt the data platform as a single unit by combining cache and persistence. Migrating both layers simultaneously made the transition more complex in the short term, but it ultimately simplified the system, eliminated cache-persistence drift, and created a more future-proof architecture.

The migration ran in parallel: new Aerospike clusters launched alongside the old systems to minimize switchover time. During migration, data was dual-written to both stacks using a duplication pipeline. Standardizing keys ensured sub-millisecond reads and consistent hashing. Aerospike's Kubernetes Operator took over scaling, storage orchestration, and zone-aware replication, reducing manual ops.

Migration execution:

Before	After
Memcached and Couchbase	Aerospike single-tier system
Dual clusters per workload	Mixed workloads on a single cluster
DRAM-hevy caching	Hybrid Memory Architecture (HMA), combining in-memory keys and values stored on NVMe SSDs
Manual scaling and tuning	Automated orchestration via Kubernetes Operator

Architectures for scale

The new platform did more than handle scale; it enabled growth. Aerospike's low-latency architecture lets Criteo run high-throughput/low-storage and low-throughput/high-storage workloads on shared infrastructure. Cross Datacenter Replication (XDR) enabled active-active clustering, removing the need for regional failover logic and supporting live failovers at AdTech speeds. Aerospike kept P99 read latency under 1ms at peak (>250K/s TPS), cutting read and write latencies at P99 by 75% and 63%.

The results speak for themselves: server count dropped 75% (from 3,200 to 800), incidents fell from six per year to one in four years, P99 read latency dropped from ~4ms to under 1ms, and operational costs and carbon footprint shrank by millions and 80%, respectively. Every match request now lands under 50ms.

Outcomes:

Metric	Result/Impact
Server count	75% — from 3,200 to 800
Incidents	From 6 per year to 1 in 4 years
Latency (P99 reads)	From ~4ms to <1ms
Operational cost	Millions annually
Carbon footprint	80%
Response time	<50ms for all match requests

Takeaways for platform architects

Split-cache architectures create fragility at scale: Drift between caching and persistence layers amplifies with load, hurting consistency and lengthening recovery time.

Databases aren't just infrastructure; they are a critical component of your tech stack: Consolidating onto a robust, unified data layer lets engineers focus on shipping features, not tuning infrastructure.

Run migrations as controlled rollouts: Use parallel builds, dual-writing, and automated testing with Kubernetes to protect data integrity and ensure a smooth cutover with minimal downtime.

Adobe

Adobe built its reputation on strong in-house tech and smart acquisitions, but this approach created real headaches in the Experience Platform. Data silos, fragmentation, and inconsistent standards made it hard to deliver a unified customer experience. Lookup times spiked and regularly blew past SLAs during peak loads. The operational footprint ballooned, and developers had no unified API for personalization, segmentation, or campaign orchestration.

To fix this, Adobe needed to support four to five million transactions per second at the edge, manage over 20TB of edge data and more than 10PB in the core, and keep data flowing bidirectionally with strong consistency. Personalizations had to load in under 5ms, with strict P95 latency targets.

The legacy stack of Cassandra, HBase, and in-memory PostgreSQL couldn't keep up. Each data pipeline had evolved for a specific use case, so refactoring everything at once looked risky. Instead of a single, smooth edge for customer 360, Adobe ended up with multiple, conflicting edges that made scaling a true hassle.

Identity fragmentation was a core problem. Multiple user identities existed across channels with no unifying structure, making it impossible to resolve a single customer view in real time. Data stream complexity added to the challenge, with identity data coming in from many sources and models. The team needed to clean, structure, and unify logical profiles without physically merging records. Lookups weren't simple get-by-ID queries. They had to resolve identities from a graph, collect fragments, filter by role, consent, and privacy, and aggregate everything into a dynamic view. This required millisecond-level read latency for millions of profiles, regardless of query complexity.

High-volume segmentation pushed the limits further. Millions of segmentation queries per second came in from streaming events, way beyond what general-purpose NoSQL databases could handle. The solution required highly optimized indices, smart data locality, and real-time profile projection to edge clusters. Data governance was also critical: access controls had to adapt dynamically based on RBAC, opt-in/out, and custom filters, with enforcement at query time, not just at the field level.

A new beginning rooted in experience

Adobe needed a fresh start without losing years of engineering investment. Aerospike delivered a single, real-time unified database that powered both the core hub and the edge. The migration was structured: Aerospike clusters ran in parallel with the old stack, moving hot partitions first and using consistency checks for cutover. The team rolled out on Azure for elastic scaling and on bare metal for performance-critical edge regions. Edge clusters achieved sub-2ms lookups, and Aerospike exposed a unified API for all developer use cases.

With this architecture, Adobe built a distributed personalization engine at the edge. Aerospike's high-performance profile store enabled edge nodes to instantly access session data, customer traits, and event histories, enabling seamless segmentation and personalization. Bidirectional XDR kept global data in sync. Hybrid Memory Architecture (HMA) empowers Adobe to process streaming updates with low latency and lower costs. Integrated AI/ML pipelines made personalization and segmentation smarter and faster, all with P95 latencies of 2ms or less.

Migration execution:

Before	After
Cassandra, HBase, and in-memory PostgreSQL	Aerospike real-time data platform
Fragmented edge and core systems	Unified system of record for edge and core, Aerospike nodes colocated with edge compute
Complex governance	Conditional reads with attribute-based access filters in custom application layers on top of Aerospike's fast key-value store
Conflicting developer systems	Unified API
Ad hoc data syncing	Bidirectional sync with Cross Datacenter Replication (XDR)

Results: Faster, easier, better experiences

Infrastructure and ops costs dropped by 3x, storage use was cut in half, and P95 latency held steady at 2ms even at peak. Throughput hit 5 million TPS at the edge, and the system scaled linearly - 20TB at the edge, 10PB in the hub. Developers finally had a unified API, making their work easier and more consistent, and customers saw faster load times and better experiences.

Outcomes:

Metric	Result/Impact
Cost reduction	3x overall decrease in infrastructure and operations cost
Storage efficiency	50% less storage used across system
Latency (P95)	Consistent ~2ms at peak load
Throughput	7x increase, 5 million TPS at the edge
Scale	20 TB edge, 10 PB hub, linear scaling
Developer experience	Unified API that can handle any use case, simplifying work for developers
Customer impact	Faster response and loading, higher satisfaction

Takeaways for platform architects:

Edge compute is only as fast as the data it can access. Optimize data structures and query handling; colocate data where it will be used.

Pulling data into a single, real-time database cuts complexity and enables scale, but requires intentional design. Complexity is not a product feature.

Data governance must be built in from the start. Modern personalization demands dynamic, query-level enforcement of roles, privacy, and regulations. Modern personalization systems have to respect roles, classifications, regulations, and consumer choices at the query level.

AppsFlyer

AppsFlyer tracks 90% of all first-time mobile app installs worldwide, and their User Installs Database is mission-critical. For years, the team ran this workload on a Couchbase cluster with over 2 billion active records (replicated to 6 billion) and more than 180,000 IOPS. But as scale increased, the system became unmanageable: node failures triggered long rebalancing, daily backups took more than 24 hours, write amplification overwhelmed available RAM, and latency spikes made SLA compliance nearly impossible.

Reaching for stability

Running on 45 AWS r3.4xlarge instances should have been enough, but, as the team discovered, simply throwing powerful hardware at a performance-constrained NoSQL database doesn't solve the problem. Furthermore, AppsFlyer needed a zero-downtime migration, active-active availability across two AZs, high-volume write support, accurate TTL handling, and a way to reduce both node count and engineering overhead.

Building a solid foundation

Keeping the old system running took all the team's time. Aerospike's solution focused on stability and performance, freeing engineers to build instead of fighting fires.
The migration replaced 45 Couchbase nodes with just 10 i3en.2xlarge Aerospike nodes. Aerospike's Hybrid Memory Architecture eliminated the need for a separate cache tier and cut RAM usage, while XDR enabled multi-AZ replication for disaster recovery.

Migration execution:

Before	After
Couchbase (45 r3.4xlarge nodes)	Aerospike real-time database (10 i3en.2xlarge nodes)
Volatile cluster, frequent rebalance storms	Stable, low latency under load
Manual backups, over 24 hours long	Automated backups with verified restore
String-based keys, unbound key sizes	20-byte fixed key hashes
No migration tooling	App-driven dual-write with fallback read
No disaster recovery	XDR replication over multiple AZs

The team built live migration logic into the application layer. They dual wrote to Couchbase and Aerospike, then switched to dual-read with fallback, tracking hit/miss metrics in real time. For long-tail data, a Couchbase view emitted the full keyspace with TTL metadata, streamed through Kafka to a Go loader that wrote to Aerospike only if the record was missing. This approach preserved data integrity and allowed a zero-downtime cutover.

Once Aerospike stabilized the core workload, AppsFlyer engineers quickly started building new features, including a new DeviceRank product that updates over 10 million new devices daily. Deploying on the Aerospike Kubernetes Operator enabled automated scaling and lifecycle management, while Karpenter handled burst scaling for high I/O periods without overprovisioning.

Results were immediate: latency dropped below 1ms under load, backups became fast and reliable, and monthly infrastructure costs fell from over $30k to around $10k. The team cut node count by more than 75%, improved reliability, and unblocked the feature backlog.

Outcomes:

Metric	Couchbase	Aerospike
Record volume	2B records, 3x replication	2B records, 4x replication (2AZs x 2 replicas)
IOPS	180k+ read IOPS, 2k write IOPS	Higher volume, better stability
Instance count	45 r3.4xlarge nodes	10 i3en.2xlarge, significantly reduced
Latency (P99)	3-5ms with frequent spikes	<1ms under load
Backups	24+ hours, frequent failures	Verified and restorable
Product velocity	Infra ops taking majority of developer time	Immediate return to feature releases, feature backlog unlocked
Monthly infrastructure cost	Over $30k	Approximately $10k

Takeaways for platform architects

Build migration logic into your apps for safer, more auditable, and portable migrations. Avoid lock-in and simplify future cloud migrations.

Automate scaling from the start using Kubernetes and cloud-native tools. Infrastructure-as-code is essential for operating AdTech at scale.

Optimize key design and topology early to avoid memory bloat and unpredictable costs. Aerospike's HMA, combined with optimal key design, creates efficiencies that have a direct effect on TCO.

A robust, unified data platform lets your team focus on shipping features, not just keeping the lights on.

Resources

AppsFlyer: Large Scale NoSQL DB Migration Under Fire

AppNexus/Xandr

Xandr (formerly AppNexus) runs one of the world's largest digital advertising platforms, handling both demand-side and supply-side operations, auctions, data management, and ad serving. Their engineering mandate is simple: "AppNexus does not close. Ever." As the business grew and data complexity exploded, the team needed a data platform that could scale without sacrificing uptime or performance.

Aerospike delivered a system that scaled with Xandr's growth and feature set, while maintaining rock-solid reliability. The platform handled a 40x increase in database size without a single outage in over seven years. Xandr's system processes more than a million reads per second every day. Even at peak, Aerospike kept 99th percentile read and write latency below 8ms. When hardware failed, the cluster rebalanced instantly and automatically, so engineers didn't waste time on manual intervention or recovery.

With Aerospike, Xandr's engineering team met their zero-downtime goal and kept pace with rapid business and data growth. The platform's flexibility and dynamic routing allowed them to keep innovating without worrying about stability or scale.

Outcomes:

Metric	Result/Impact
Database size growth	40x increase handled with no downtime
Uptim	Zero downtime for over seven years
Read throughput	Sustained 1 million reads per second
Latency (P99.8)	Below eight milliseconds for reads and writes
Cluster recovery	Automatic, instant rebalancing during hardware failures
Scalability	Flexible, dynamic routing and horizontal scaling as business grew

The results were impressive: a 40x increase in data volume, a sustained 1 million reads per second, with zero downtime.These results highlight Xandr's ability to scale its platform and feature set without sacrificing reliability or speed.

Resources

When five nines is not enough: What 100% uptime looks like

Nielsen

Nielsen built its reputation on delivering fast, reliable analytics to the world's largest advertisers and retailers. As digital channels grew and real-time requirements became the norm, Nielsen needed a platform that could guarantee sub-millisecond reads and writes, scale globally, and stay online even during unpredictable natural disasters.

Aerospike became the backbone of Nielsen Marketing Cloud's data infrastructure. The engineering team used Aerospike to store and retrieve user history based on cookie or mobile ID at sub-millisecond speeds, powering real-time decisions and analytics. Every transaction, including network hops and analytics, had to finish in under 250 milliseconds. Aerospike's architecture handled hot keys and high write loads without bottlenecks, and cross-region replication provided low-latency failover and true global availability.

The platform's efficiency lets Nielsen fully take advantage of modern hardware SSDs, NVMe storage, and high-performance RAM while keeping operational costs down. The system anticipates hardware failures and rebalances automatically, so engineers don't have to scramble during outages. Even during major events like Hurricane Sandy, Nielsen stays online and delivers uninterrupted service.

Outcomes:

Metric	Result/Impact
Read/write latency	Sub-millisecond, even under high load
End-to-end transaction time	<250 ms, including analytics and network time
Disaster resilience	Zero downtime during major events (e.g., Hurricane Sandy)
Global availability	Cross-region replication, low-latency failover
Hot key handling	Efficient write-hot key management, no bottlenecks
Hardware utilization	Maximized use of SSD, NVMe, RAM; significant cost savings
Scalability	Fully redundant, anticipates and rebalances before hardware issues
Real-time analytics	Enabled real-time user tracking and identity resolution

Nielsen's engineers now focus on building new features and analytics, not fighting infrastructure fires. The platform's reliability and speed keep Nielsen at the top of the market analytics game.

The Trade Desk

The Trade Desk runs the world's largest demand-side platform, processing over 10 billion records daily, each with tens of thousands of data points. The platform must match advertisers, customers, publishers, and context in real time, millions of times per second, all while keeping costs razor-thin. Even a small delay or missed match can turn a profitable day into a loss.

The engineering team hit a wall with their Cassandra stack. Latency was too high, and cold storage access was slow-making it impossible to meet the demands of real-time bidding (RTB) and advanced analytics. TTD needed to analyze complex, high-volume data instantly, but the old system couldn't keep up.

Aerospike delivered a new architecture that redefined hot cache and cold storage. By condensing records and using dynamic keys, the system could return the right data for any context or use case. Cold storage thaw time dropped to just eight milliseconds, making it usable for RTB. The new platform handled 12 million queries per second (800 billion per day), with peak writes to cold storage topping 20 million per second. Server count dropped from 500 to just 60, with better geo-distribution for faster responses. Consolidating all data into a single record freed up CPU resources, letting engineers optimize bids and run more advanced analytics.

The result: faster data access, lower costs, and a better customer experience; no more latency bottlenecks on the front end.

Outcomes:

Metric	Before	After (Aerospike)
Cold storage thaw time	Too slow for RTB	8ms
Query throughput	Limited by latency	12M QPS (800B daily)
Peak write throughput	Not specified	20M writes/sec to cold
Server count	500	60
Data architecture	Fragmented, CPU-bound	Unified, CPU-efficient
Customer experience	Latency bottlenecks	Real-time, seamless
Analytics/bidding capacity	Constrained	Expanded, more optimized

3

Aerospike for AdTech: Real-time, petabyte-scale data infrastructure

Aerospike delivers sub-millisecond, high-throughput performance for real-time AdTech systems to handle mixed workloads (reads/writes/updates), enforce compliance policies inline, and scale to billions of events per day with minimal infrastructure.

Performance at scale

- 4–5M transactions per second (TPS) on a 20-node cluster.
- Sub-millisecond P99 latencies, even at peak load.
- Exactly-once semantics for clickstream, sessionization, and bid data.
- Automatic partitioning and hotkey-aware distribution prevent I/O bottlenecks.
- Built for petabyte-scale datasets with real-time access across all tiers.

Hybrid Memory Architecture (HMA)

- Indexes in RAM, data on NVMe SSDs for fast, cost-efficient storage.
- Up to 80% lower infrastructure costs vs. all-RAM or cache-persistent systems.
- Removes the need for external cache layers and avoids cache-persistence drift.
- Supports in-memory secondary indexes for instant lookups on any field.
- Keeps large historical BLOBs instantly accessible with no S3 thaw delays.

Availability and replication

- Active-active clustering for HA and zero downtime.
- Cross Datacenter Replication (XDR) with geo-filtering and real-time sync.
- Automatic data rebalancing and failure recovery without operator intervention.
- Prevents split-brain scenarios with consistent hashing and cluster-aware clients.

Built-in compliance and data governance

- Enforces GDPR, CCPA, and regional privacy rules at query time.
- Role-based (RBAC) and attribute-based (ABAC) access built into the database layer.
- Policy enforcement without impacting latency or adding pipeline complexity.
- Consent-aware queries return only compliant user data with no external validation required.

Developer and ops enablement

- Unified data plane: Store raw bidstream data, processed events, and ML models in one system.
- Protocol-agnostic API layer normalizes ingest, removing format-specific logic.
- Schemaless JSON support with shared indexes for flexible data modeling.
- Kubernetes Operator automates scaling, failover, upgrades, and topology management.
- Replaces batch reconciliation jobs with real-time metric synchronization.

Aerospike is ideal for:

- Real-time bidding (RTB) engines
- Identity graphs and frequency capping
- Attribution and clickstream
- Consent management and privacy enforcement
- Large-scale user profile stores

Aerospike eliminates the speed, cost, and complexity tradeoffs of legacy systems. Purpose-built for AdTech, it enables developers to focus on optimization and product innovation, not on infrastructure maintenance.

About Aerospike

Aerospike is the real-time database built for infinite scale, speed, and savings. Our customers are ready for what's next with the lowest latency and the highest throughput data platform. Cloud and AI-forward, we empower leading organizations like Adobe, Airtel, Criteo, DBS Bank, Experian, PayPal, Snap, and Sony Interactive Entertainment. Headquartered in Mountain View, California, our offices include London, Bangalore, and Tel Aviv.

For more information, please visit https://www.aerospike.com.

2440 W. El Camino Real, Suite 100, Mountain View, CA 94040 | (408) 462-2376