AEROSPIKE

Exploring data consistency in Aerospike Enterprise Edition

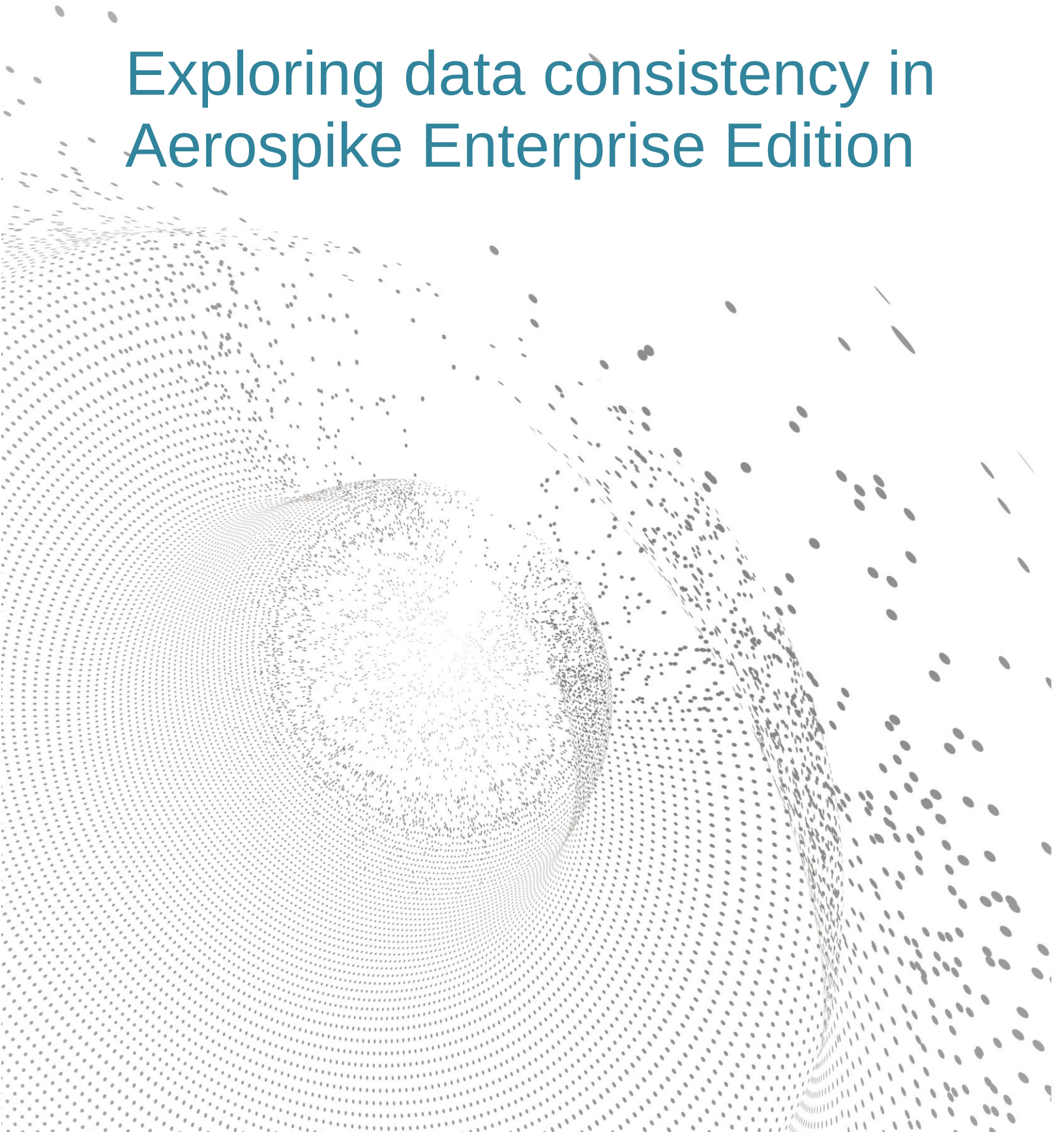

Table of Contents

Executive overview

It's something that's eluded the Information Technology (IT) industry for years – a scalable, reliable operational database platform that combines extremely fast read/write speeds with strong data consistency at an attractive price point. Until recently, commercial offerings fell short of this demanding goal, forcing firms to implement costly workarounds in their business processes, applications, and infrastructures. Some organizations reluctantly accepted slow performance to ensure the strong and immediate consistency of their data, while others tolerated weaker forms of consistency to achieve aggressive read/write data access speeds.

Fortunately, such compromises are no longer necessary for real-time applications that use single-record transactions. Recent advances in database management technology from Aerospike have eliminated the need to trade off high performance for strong consistency (or vice versa) for these applications. And Aerospike has delivered this support without requiring a separate caching layer, additional hardware, or complex tuning efforts, all of which would drive up ownership costs. Today, Aerospike's database system combines strong, immediate data consistency with speed at scale for a fraction of the cost of alternative solutions. As a result, firms in banking, telecommunications, and other industries are deploying Aerospike as a system of record and as a system of engagement, effectively transforming their IT infrastructures to meet challenges of contemporary business demands.

Perhaps that sounds hard to believe. After all, database researchers and vendors – including Aerospike – have been struggling to combine exceptional performance with strong data consistency for years. But the facts prove otherwise. For example, Aerospike's strong consistency support was recently confirmed by independent Jepsen tests. Furthermore, Aerospike benchmarks show that its runtime speeds haven't been slowed by its support for strong, immediate consistency. Finally, Aerospike clients frequently attest to the system's exceptional cost efficiency. Firms using Aerospike in production for mission-critical applications have cut their server footprints up to 90% and often enjoy total cost of ownership (TCO) savings of $1 to $10 million per application. To date, no other system can provide proof points that match Aerospike's speed, consistency, and cost efficiency.

If you're not already familiar with Aerospike, a separate white paper introduces its architecture and describes key features that distinguish Aerospike from other solutions. This paper will help you understand Aerospike's support for data consistency, including what behavior you can expect under normal and exceptional conditions. You'll also explore sample customer applications, so you can learn why it's critical – and cost effective – to use an operational database system that delivers strong consistency and low data access latencies. But first, let's quickly review a few basics about data consistency.

About consistency

Data consistency for transactional workloads has been an area of innovation for decades, fueled in part by the power and popularity of distributed computing environments. Node and network failures are unavoidable, and both increase as clusters grow. Consequently, it's important for IT organizations to understand how different database offerings cope with the inevitable challenges of data consistency and availability during normal operations and failure scenarios. Regrettably, that's no easy task: the subject itself is complex, and some terms aren't even used consistently in the IT industry.

What can customers do? For a start, ask the right questions. Here's a short list to consider:

- What's the scope of a transaction or unit of work? A single operation on a single record? Multiple operations spanning multiple records? The latter is more flexible but carries greater performance overhead.

- When is consistency enforced? Consider that most operational database systems running on clusters of commodity servers store multiple copies of user data for resiliency. Is consistency enforced immediately (which implies synchronous updates to all replicas in a cluster)? Or eventually (which implies asynchronous updates to replicas)?
- What types of data inconsistencies can arise during normal operations when the system is fully healthy and during various failure scenarios? A few common problems include:
    - Stale reads (reading data that isn't the most recently committed value)
    - Dirty reads (reading data that hasn't yet been committed)
    - Data loss
- If varying levels of data consistency are supported, how does each impact runtime performance?

In the next section, you'll learn the answers to these questions for Aerospike. But first, let's provide some context by reviewing general approaches in the industry.

Relational DBMSs deployed on commodity server clusters typically take a strict approach to consistency. They enable programmers to group multiple operations over various records into a single transaction that will either be committed or rolled back immediately. Through locks and other mechanisms, they enforce ACID properties (atomicity, consistency, isolation, and durability) and provide strong guarantees against loss of data, reads of uncommitted data, reads of stale data, and so on. But such mechanisms can be quite heavy-handed, rendering the systems too slow to accommodate the real-time read/write workloads of contemporary online transaction processing (OLTP) applications. In addition to performance limitations, these systems generally maintain one copy of user data, which reduces data availability during certain failure scenarios. For many modern applications, including those supporting systems of engagement, such characteristics simply aren't suitable.

Some commercial systems, including various NoSQL (not-only-SQL) systems, relax aspects of traditional data consistency implementations to achieve lower data access latencies, higher transaction throughput, greater data availability, or other goals. For example, some NoSQL systems automatically maintain three replicas of data to increase data availability and help improve response times. These systems often implement *eventual consistency*, guaranteeing that eventually all access to a given record will return the same value, assuming no other updates are made to that record before copies converge. But this comes at a cost. During normal operations – not just failure scenarios – readers may see stale data. Furthermore, if the same record is updated more than once before all copies converge, the conflict must be resolved in some manner, which can slow performance and result in data loss.

Strong eventual consistency refines this approach to avoid data loss, and *conflict-free replicated data types* (CRDTs) are one way to implement strong eventual consistency within a single cluster or across clusters. In effect, by exploiting commutative operations for certain data types, systems that support CRDTs can concurrently process updates to the same data on different nodes. After propagating the operations to all replicas (in any order), the data will eventually become consistent across all nodes without loss. If that sounds confusing, consider a numeric value replicated three times. If Application A adds 10 to the original value of 100 on Node 1 and Application B subtracts 20 from the same value replicated on Node 2, eventually all replicas will converge to a value of 90. Arithmetically, it doesn't matter if the addition or subtraction operation occurs first. As long as both are applied to the original value, the end result will (eventually) be the same. Of course, until the replicas converge, readers may see inconsistent data.

As you might imagine, there's no easy way to balance the often-conflicting demands of consistency, availability and high performance. Indeed, many Aerospike clients discovered this firsthand after testing or deploying other offerings only to find them too slow, too unpredictable, and/or too expensive.

Aerospike offers a compelling alternative: a cost-efficient, high-performance operational database system that supports strong, immediate consistency and high levels of availability.

Aerospike's approach to consistency

If you're not already familiar with Aerospike, it's a distributed NoSQL system that provides extremely fast – and predictable – read/write access to operational data sets that span billions of records in databases of 10s – 100s TB. Its patented Hybrid Memory Architecture™ (HMA) delivers exceptional performance using a much smaller server footprint than competing solutions. As shown in Fig. 1, Aerospike uses dynamic random-access memory (DRAM) for index and user data. Optionally, applications can commit each write directly to fast, non-volatile memory (solid state disks or SSDs), which Aerospike treats as raw devices for speed and efficiency. Sophisticated (and automatic) data distribution techniques, a "smart client" layer, and other features further drive Aerospike's speed and cost efficiency. For more about Aerospike's overall architecture and capabilities, see this white paper.

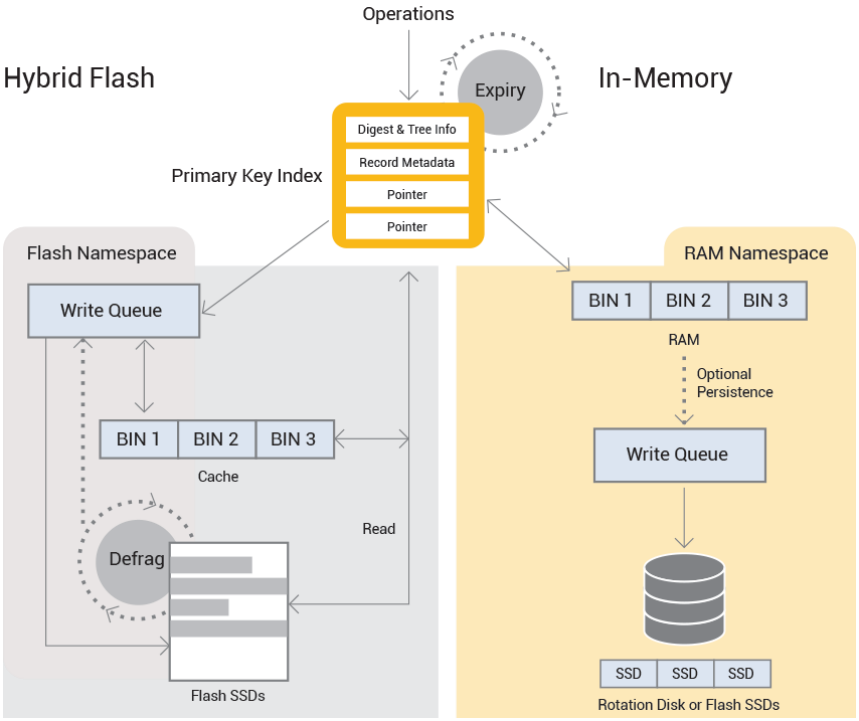

Figure 1: Overview of Aerospike storage architecture

To meet the demanding needs of real-time enterprise applications, Aerospike namespaces (databases) can be configured to operate with strong consistency (SC), which prevents stale reads, dirty reads, and data loss. Firms can also configure Aerospike to relax its strong consistency support and operate in available / partition tolerant (AP) mode, which offers backward compatibility with early Aerospike releases. However, given that performance is comparable in both modes and that strong consistency simplifies application logic, Aerospike expects most firms to prefer to configure their namespaces to operate with strong consistency.

It's important to note that Aerospike's support for consistency is *immediate* regardless of its operational mode. In other words, any write to a record is applied synchronously to all replicas in a local cluster. Furthermore, Aerospike applies each record operation atomically. Restricting the transaction scope to a single record enables Aerospike to deliver strong consistency and exceptional runtime performance not seen with other approaches. Many real-time, mission critical applications can be implemented with this model. A subsequent section discusses sample customer application scenarios that require the extremely low data access latencies that Aerospike provides.

Strong consistency technology

Aerospike supports strong, immediate consistency to prevent conflicting writes and ensure that reads see the most recently committed data values. Aerospike processes all writes for a given record sequentially so that writes won't be re-ordered or skipped. Independent tests of the Jepsen workload revealed no errors when operating with Aerospike's recommended configuration settings. And Aerospike benchmarks revealed no measurable performance impact with strong consistency, as you'll see in a subsequent section. To date, no other NoSQL system has delivered comparable levels of consistency and runtime performance for operational workloads at scale.

To ensure strong consistency, Aerospike uses an internal roster and the "heartbeats" of individual nodes to assess the cluster's current state. The roster contains a full list of nodes that comprise a healthy cluster and identifies the data owned by each node, noting if the data is considered a master or replica copy. This roster, which is stored on every node in the cluster, enables Aerospike to determine what operations are valid during various failure scenarios, such as a network failure that causes some nodes to be unable to communicate with others. (This is sometimes called a "split brain" scenario.) To preserve appropriate ordering of events across the cluster, Aerospike implemented a logical Lamport clock that combines timestamps with additional information, including a counter that increments when certain events occur. This approach is superior to a simple timestamp-based architecture, which can suffer from clock synchronization problems that lead to inadvertent data inconsistencies.

Although full details of Aerospike's internal logic are beyond the scope of this paper, we'll summarize a few aspects, so you can better understand how Aerospike preserves strong consistency. Further information about data consistency in Aerospike is available publicly through Aerospike's product documentation. However, before we begin exploring some operational scenarios, it's important to note that **unlike many systems, Aerospike doesn't require three or more copies of data to provide strong, immediate consistency**. Instead, it operates by default with a replication factor of two – a feature that enables Aerospike to reduce ownership costs and simplify operations compared with other offerings.

Operational scenarios

Let's explore how Aerospike operates under normal and failure scenarios. Fig. 2 illustrates four different operational states involving a 5-node system. Node 5 is designated in the roster as the master for a given data record, and Node 4 is designated in the roster as the record's replica. The top row shows a healthy cluster with all nodes operational and all data available. The next row depicts a situation in which Nodes 1-3 are split from Nodes 4-5. In such cases, read/write requests for the target data are permitted only on Nodes 4-5, which contain both the master and replica copies as designated in the roster.

The third and fourth rows depict slightly more complex scenarios. In the third row, we see Nodes 1-4 split from Node 5. In this case, because Node 4 contains a designated replica (or "roster-replica") of the data and is part of a majority sub-cluster, it will assume the "master" role for this data and process read/write requests. Furthermore, Aerospike will automatically replicate this data to another available node (Node 3, in this example) to further promote data availability and system resiliency. Note that Node 5 will not accept read/write requests in this situation because it is part of a minority sub-cluster (in this case, a sub-cluster of one). Preventing Node 5 from processing data access requests in this situation blocks potential loss of data or stale reads. Finally, the last row depicts what would happen if the Node 4 were to subsequently split from Nodes 1-3. In such a case, Aerospike would not process read/write requests for this record because Nodes 4 and 5 – designated in the roster as the replica and master owners of the data – don't form their own sub-cluster and aren't part of a majority sub-cluster of nodes listed in the roster.

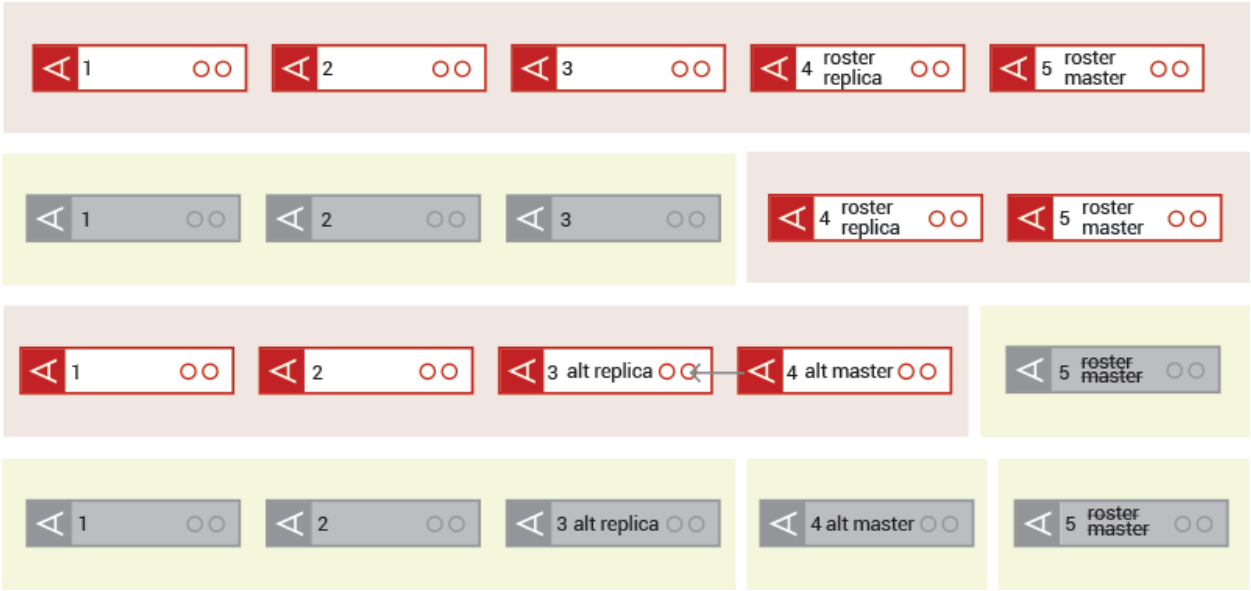

Figure 2: Aerospike's strong consistency under different cluster states

Aerospike's strong consistency model addresses a variety of subtle situations to ensure appropriate behavior. For example, consider a failure scenario in which the original master node for a record hasn't yet detected a network problem impacting the cluster's state and, therefore, hasn't relinquished master ownership of its data even though the new master (in a separate sub-cluster) has detected the cluster's changed state and taken over. During this transitional period (sometimes called the "master overhang" period), clients might be writing to both the old and new masters, which could introduce data inconsistencies if only timestamps were used to order events. To prevent such inconsistencies, Aerospike associates "regime" information with each user record. This information, coupled with the last update time and record generation data, comprises Aerospike's customized Lamport clock, which enables Aerospike to properly order events across the cluster and avoid data loss.

By default, Aerospike directs write operations to DRAM buffers. When a buffer is full, Aerospike flushes the buffer's contents to storage in a single block write for maximum efficiency. Each node manages its own write buffers and disk flushes, so maintaining two or more replicas of user data provides considerable durability during node failures. For additional durability, programmers can instruct Aerospike to commit each write to the disk on a per-transaction basis; to achieve fast speeds, this approach requires drives that have very low latency write performance (typically, certain types of Non-Volative Memory express or NVMe drives).

Linearized Access and Session Consistency

Applications can choose between *linearized* access or *session consistency* for each read operation. The former is most restrictive, imposing a linear view of the data for all clients that access it. Essentially, when one application completes its write operation, all later reads by any application will see the value of that write or a later write. By contrast, session consistency guarantees that the same application will read its own writes. Other applications may or may not, depending on the timing of operations. Note that linearized access carries a greater performance overhead and that session consistency is the default.

If you'd like to understand about data consistency in Aerospike, refer to the consistency section of Aerospike's online documentation.

Performance considerations

Exceptional performance is a hallmark of Aerospike, and its engineering staff took pains to implement strong consistency in a way that would fulfill its clients' aggressive demands. Table 1 summarizes internal benchmark results that demonstrated no measurable difference for read/write operations and throughput when using default settings for SC (strong consistency) or AP (available/partition tolerant) mode. (Both presume a replication factor of two, and SC mode uses session consistency.) Furthermore, even with linearized access in SC mode, Aerospike still delivered nearly 2 million transactions per second with sub-millisecond latency.

	SC Mode: Linearizable	SC Mode: Session Consistency	AP Mode
OPS	1.87 million	5.95 million	6 million
Read Latency	548 µs	225 µs	220 µs
Update Latency	630 µs	640 µs	640 µs

Table 1: Aerospike benchmark results of 5-node cluster operating with different consistency settings. Tested configuration featured a replication factor of 2, 500M keys with 8-byte objects, and a setting of in-memory with persistence.

Consistency across clusters

At this point, we've focused on strong consistency within a single Aerospike cluster. For firms that need to deploy Aerospike across multiple data centers, Aerospike offers two primary options. If the data centers are located within 10 miles of each other, firms can leverage Aerospike's rack-aware mechanism. This mechanism effectively supports strong consistency by allocating replicas and masters to different racks, enabling a transaction to be committed across both data centers within a couple of milliseconds.

For firms with multiple data centers located further away – perhaps even in different countries – Aerospike provides cross-data center replication (XDR). This offering enables firms to ensure continuous operations by transparently and asynchronously replicating namespaces or specific sets to one or more remote clusters installed on premises, in the cloud, or a combination of both. As of this writing, XDR supports asynchronous replication and does not preserve strong consistency across multiple clusters. Clients that need to implement strong consistency across geographically distributed data centers should contact Aerospike for implementation guidance.

Applications and use cases

Firms in banking, financial markets, telecommunications, retail, transportation, and other industries have come to rely on Aerospike to provide extremely fast response times for read/write workloads spanning very large data sets. Applications deployed on Aerospike span both systems of engagement and systems of record. The former foster collaboration and interaction; they're often driven by social media, mobile device usage, cloud computing, real-time chats, and streaming data. Systems of record support more classic data management applications by providing a single authoritative record of critical business data, such as account information or sales records.

Many firms need to support both types of applications. And increasingly, firms are recognizing that both types of applications impose some common – and aggressive – demands on their operational data systems, such as:

- Service-level agreements (SLAs) that require sub-millisecond database response times.
- High throughput for mixed workloads (e.g., 3 – 5 million operations per second).
- Support for managing billions of business records in databases of 10s – 100s TB.
- High scalability for handling unpredictable increases in data volumes and transactions.
- Low total cost of ownership (TCO).

To understand how Aerospike's strong consistency can benefit different types of applications, let's review a few sample scenarios.

Payments. Credit card transactions, electronic funds transfers, currency exchanges, and other digital banking and payments applications all demand extremely low data access latencies and strong levels of data correctness. Providers maintain systems of record to track the details of each transaction, which often must be accepted or rejected in less than 100 milliseconds. Aerospike can support such aggressive application demands. In addition, digital payments providers and various banking institutions must combat increasingly sophisticated attempts at fraud. Quite often, they maintain important historical data in Apache Hadoop or a relational DBMS warehouse, which they use to develop and refine proprietary fraud detection rules. A subset of this data is periodically transferred into Aerospike to support real-time detection of suspicious – and potentially fraudulent – business transactions. The results of this modernized approach can be stunning. Indeed, one Aerospike client realized a 30x improvement for the service level agreements (SLAs) associated with its fraud detection algorithms by moving from a 2-layer architecture involving Oracle RAC and 360 Terracotta servers to a 20-node Aerospike cluster.

Trading (financial services). Intra-day trading applications are among the most demanding in the financial services industry. Clients often initiate trades through mobile devices and expect an instant response, forcing IT organizations to provide sub-millisecond data access speeds for hundreds of millions of transactions per day. One global investment bank with $3 trillion in client assets turned to Aerospike to fulfill these aggressive demands. Its mainframe relational DBMS was already overloaded, and the firm's use of a caching layer to front-end the mainframe DBMS had proved problematic. Now a 12-node Aerospike cluster serves as the system of record for intra-day trades, replacing the cache and offloading some work previously done on the mainframe. The mainframe continues to serve existing applications, and the firm regularly transfers data between the two systems. This modernized infrastructure yielded a five-fold increase in processing speeds even as new business drove the database size from 4 to 14TB. Furthermore, Aerospike enabled the firm to accomplish this with 90% fewer servers deployed, saving an estimated $10,000 per trading day.

Retail and entertainment. Online retail and event ticketing applications need to process customer orders and understand the real-time status of available inventory. The ability to handle large volumes of transactions in real time is critical, particularly during holidays, promotions, and other periods of peak activity. Aerospike can provide high transaction throughput without compromising runtime speeds or data correctness. Furthermore, Aerospike can support personalization applications more typical of systems of engagement. For example, a retailer or ticketing agency may want to customize a client's web page to promote another item or event just before checkout. Aerospike's low data access latencies enable this personal form of last-minute engagement with the client. Indeed, one popular online retailer recently replaced a Cassandra cluster with Aerospike to support a personalization workload and is enjoying lower data access latencies, a smaller server footprint, and less administrative work.

Mass transit. Cities around the world have adopted transit cards for collecting fares on subways and other mass transit systems. Such cards replace tokens and coins, functioning like debit cards that transit riders can swipe at kiosks or turnstiles to purchase a ride. These systems must be backed by a real-time operational

database that can accommodate large numbers of concurrent transactions with accuracy and speed – requirements that Aerospike can readily support. In addition, the same Aerospike cluster can support real-time analytics applications, perhaps enabling the transit provider to understand peak usage periods and trends. Using the same database platform to support systems of record as well as systems of engagement reduces costs and simplifies the overall IT infrastructure.

Summary

Until recently, firms that required ultra-fast access to operational data faced some tough choices involving data consistency, data availability, data access latency, and TCO. Different architectures brought different trade-offs. Mainframe solutions are strong in the first three areas, but they're also expensive to maintain and difficult to adapt to evolving business needs. Distributed relational DBMSs offer strong consistency but rarely deliver the speed needed for real-time applications, even with extensive (and costly) tuning efforts. Two-tier solutions that couple a caching layer with an operational data store (relational or otherwise) alleviate some of the performance challenges but introduce consistency and TCO hurdles.

Aerospike's database platform introduces new techniques for managing these tough challenges. It provides extremely low data access latencies (often sub-millisecond) for read/write workloads over large volumes of operational data at a fraction of the TCO of other solutions. Furthermore, it eliminates the need for firms that require strong, immediate consistency to compromise on performance. With Aerospike, firms can use the same database platform to act as a system of engagement as well as a system of record.

Want to learn more? Contact Aerospike to arrange a technical briefing or discuss potential pilot projects. If you'd prefer to explore the technology independently, you'll find plenty of materials available on Aerospike's web site, including free on-demand webinars, benchmarks, and product documentation. You can even "get your hands dirty" by downloading the free Community Edition.

So why wait? Firms around the globe are already achieving tangible business results by modernizing their IT infrastructures with Aerospike. Why not explore what Aerospike can do for you?

Resources

Aerospike accelerates: Strong Consistency with High Performance in Aerospike 4.0, Brian Bulkowski, CTO, March 2018.

Aerospike Consistency Architecture Guide, product documentation.

Aerospike overview white paper, *Maximize the Value of Your Operational Data*.

Consistency Model (background information), Wikipedia.

Jespen report on Aerospike 3.99.0.3, Kyle Kingsbury, Jespen.io, March 2018.

AEROSPIKE

About Aerospike

Aerospike is the world's leading enterprise-grade, internet scale database whose patented Hybrid Memory Architecture™ enables digital transformation by powering real-time, mission critical applications and analysis. Only Aerospike delivers strong consistency, predictable high performance and low TCO with linear scalability. Serving the financial services, banking, telecommunications, technology, retail/ecommerce, adtech/martech and gaming industries, Aerospike has proven customer deployments with zero downtime for seven years running. Recognized by industry analysts as a visionary and leader, Aerospike customers include Nielsen, Williams Sonoma, Kayak, Neustar, Bharti Airtel, ThreatMetrix, InMobi, Applovin and AppNexus. Aerospike is based in Mountain View, CA, and is backed by New Enterprise Associates, Alsop Louie Partners, Eastward Capital Partners, CNTP and Silicon Valley Bank

2525 E. Charleston Road, Suite 201
Mountain View, CA 94043
Tel: 408 462 AERO (2376)
www.aerospike.com