White Paper

AEROSPIKE

A Graph Database Designed for Scalability

Author: George Anadiotis
Founder, Linked Data Orchestration

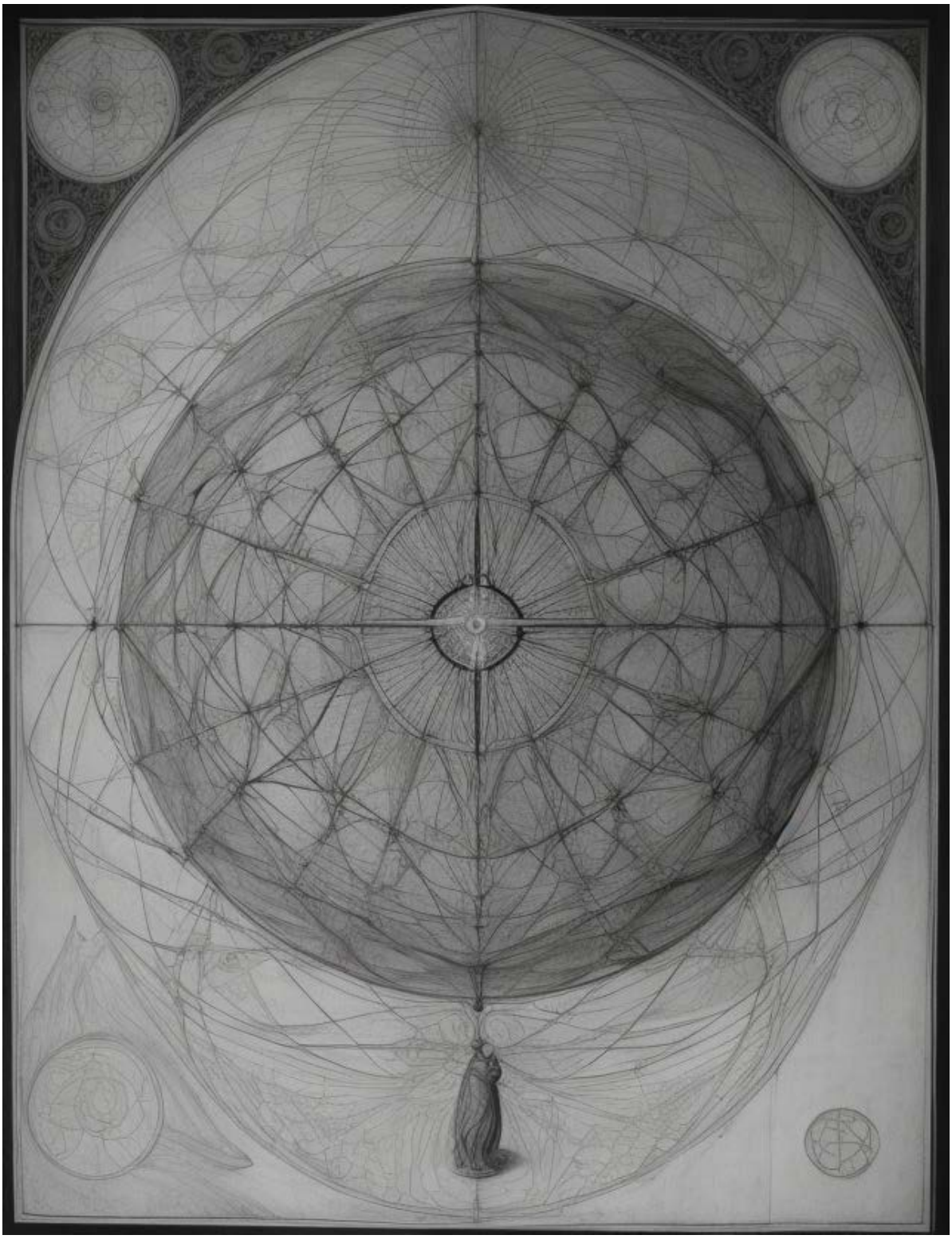
Introducing Aerospike Graph

"These are the principles for the development of a complete mind: Study the science of art. Study the art of science… Realize that everything connects to everything else."

Leonardo da Vinci

Contents

From Connections to Value



Everything is connected

There's a thin line separating deep insight from platitude. The interconnected nature of all things in nature and beyond falls under this category. The notion probably occurred in the mind of a primordial philosopher, and has been expressed countless times since.

Interconnectedness is considered a pillar of ecology, political economy, art, spirituality and complexity theory, among others. Beyond philosophical foundations, however, studying and applying the properties of interconnected networks can be of great practical significance to every organization today.

> ***"Networks are at the heart of some of the most revolutionary technologies of the 21st century, empowering everything from Google to Facebook, Cisco, and Twitter. At the end, networks permeate science, technology, business and nature to a much higher degree than it may be evident upon a casual inspection. Consequently, we will never understand complex systems unless we develop a deep understanding of the networks behind them"***
>
> ***writes Albert-László Barabási in his 2016 book Network Science.***

Graph theory and graph algorithms represent a theoretical framework and a practical set of tools, respectively, that enable a deep understanding of networks. This is the reason why interest in both the theory and the tools is peaking, and technologies that empower them are exploding.

According to Gartner, graph technologies will be used in 80% of data and analytics innovations by 2025, up from 10% in 2021[1], facilitating rapid decision-making across the enterprise.

Graph technologies can be broadly classified in 4 categories: Graph Analytics, Graph Databases, Knowledge Graphs, and Graph AI.

Graph theory was initially formulated by Swiss scientist and engineer Leonhard Euler in the 18th century in an effort to model and solve an optimization problem known as the bridges of Königsberg[2], where he modeled bridges as edges and land masses as vertices. Since then, several graph algorithms have been developed that can aid in addressing a wide range of problems.

Path finding, centrality, community detection and similarity are some of the main classes of graph algorithms. Graph algorithms have many applications in data analytics. Some of the most common

[1] https://www.gartner.com/en/newsroom/press-releases/2021-03-16-gartner-identifies-top-10-data-and-analytics-technologies-trends-for-2021
[2] https://en.wikipedia.org/wiki/Seven_Bridges_of_Königsberg

use cases are fraud detection and investigation, anti-money laundering and financial crime, customer 360, supply chain management, master data management, and recommendations. Graph algorithms can bring utility to applications both in offline, batch analytics scenarios as well as by providing near-real time, operational data insights. Use cases that make a great match for these scenarios have one thing in common: they can benefit from heuristics based on the network structure of their data, if modeled as graphs.

Graph algorithms can be executed on any back end, as they only require reading graph-shaped data. However, exporting data to graph libraries is not the most efficient way to work with graphs. Graph databases have the ability to model data as a graph, fully supporting CRUD (Create, Read, Update, Delete) operations with an API and query language.

Using a graph database when dealing with graph-shaped data makes lots of sense as it minimizes the effort to ETL data on CRUD operations. In addition, databases optimized to work with graph-shaped data have better performance when applied to graph-shaped tasks.

Many models, one objective

The continuing rise of graph databases is part of a broader phenomenon in the evolution of databases and infrastructure, the so-called NoSQL movement. Up until the early 2010s, relational databases were practically the only game in town. From this point on, everything changed.

Many different types of data models and database architectures were implemented. They were driven by the need for horizontal scalability and data distribution, much higher performance, reliability and flexibility, as well as the realization that not all application workloads can be served equally well by one type of database.

In parallel, both the infrastructure landscape as well as applications and their architecture have been evolving in multiple directions. In terms of infrastructure, the predominant model evolved from scale up hardware in data centers that organizations owned to scale out hardware in the cloud that organizations use on demand.

Often, the tradeoff of having one specialized database per application vs. having to manage those databases will tip the scale in favor of simplicity and reduced Total Cost of Ownership. This becomes even more pronounced in times of economic downturn, when organizations are looking to do more with less.

That's why more organizations today are turning to multi-model, multi-cloud databases that can be effective and efficient when dealing with different scenarios. Aerospike has been around for long enough to have lived through this evolution, enabling it to digest many lessons and respond in kind.

The multi-model, NoSQL Aerospike Database enables organizations to cost-effectively process gigabytes-to-petabytes of data to make real-time decisions at scale. Aerospike delivers

predictable performance, is strongly consistent, and offers cross-datacenter replication for a true globally distributed real-time database.
Aerospike is a flash memory and in-memory open source distributed NoSQL database management system, with its roots going back all the way to 2010. It features a distributed architecture with three key objectives:

- Create a flexible, scalable, real-time platform for web-scale applications
- Provide the robustness and reliability expected from traditional databases
- Provide system elasticity and operational efficiency with minimal manual involvement

Aerospike provides full support for key-value and document data models. SQL queries are supported via a tightly integrated, distributed SQL analytics engine powered by Starburst (Presto/Trino). Aerospike has now taken the next step in its evolution, offering full support for graph operations, battle-tested for use in some of the most demanding applications in the world.

Graph Analysis for real-time applications at global scale

Leveraging connections brings business value

Aerospike's foray into graph technology has been evolving for years, driven by a number of use cases brought forward by its customers. Let's revisit some of the most prominent among Aerospike's graph use cases: Adobe, Signal (acquired by TransUnion), and PayPal.

Founded 40 years ago on the idea of creating innovative products that change the world, Adobe offers groundbreaking technology that empowers everyone, everywhere to imagine, create, and bring any digital experience to life. Industry analysts have named Adobe a leader in 40+ analyst reports in categories such as digital experience platforms, content management systems, customer analytics, B2B marketing automation platforms, digital commerce, enterprise marketing suites, and more.

The Adobe Experience Cloud is a comprehensive solution for analytics, advertising, marketing, and commerce designed to deliver exceptional experiences from creation all the way through monetization and acquisition through renewal. At its core, the Adobe Experience Platform creates

a dynamic real-time identity graph of customer profiles with high volume, high-velocity data coming in from multiple disparate data sources.

Adobe needed to manage high-volume, high-velocity data from various sources like CRM, Adobe Analytics, transactions, and attributes. This data had to be updated in milliseconds while considering privacy constraints. Each data source used different IDs, creating separate silos with fragmented customer information. An identity resolution service was employed to stitch these sources together, forming a unified, data-enriched 360-degree customer view facilitated by an identity graph.

Adobe has harnessed Aerospike to develop a custom identity graph solution for the Adobe Experience Cloud, focusing on crucial advertising and identity resolution services. Aerospike now serves as a robust system of record for handling approximately 100 billion key elements. This strategic migration to Aerospike has yielded substantial benefits for Adobe Experience Cloud, including a remarkable 7x increase in overall throughput and a significant 3x reduction in operational costs.

Signal, now part of TransUnion's TruAudience marketing solutions, powers data integration for brands and agencies globally. TransUnion is a global information and insights company that makes trust possible in the modern economy by providing a comprehensive picture of each person so they can be reliably and safely represented in the marketplace. TransUnion has a presence in more than 30 countries, with a global customer base of over 65,000 businesses and organizations and 166 Million consumers.

Signal sits at the intersection of the adtech and martech ecosystems and enables brands and agencies to build low-latency, large-scale identity graphs for media activation and identity resolution. Signal was a long-time Cassandra user, however they made the decision to simplify their data model and build a new identity graph solution.

Signal, after evaluating various options for their identity graph solution (including some popular graph databases), chose to build it on Aerospike because it was the only solution that could handle their scalability requirements. This decision resulted in significant benefits: a 66% reduction in Opex, a projected 68% total cost of ownership reduction over the first three years, a server count reduction from 450 to 60, and a tenfold improvement in the execution time of critical business processes. This led to enhanced core operations, stronger customer relationships, and allowed staff to focus on strategic, forward-looking projects.

PayPal is one of the biggest and most impactful financial technology organizations in the world. It supports over 400 million active consumers and merchants worldwide, processing several thousand payment transactions per minute amounting to $282 billion annually. PayPal is also a long-time Aerospike adopter, putting data at the heart of its fraud strategy with Aerospike.

PayPal's fraud rate is between 0.17% and 0.18% of revenue. While this number is well below the industry average of 1.86%, it still represents over 1 billion dollars a year for the firm. Needing to be able to more quickly process and analyze its data to identify emerging fraud patterns in real-time, PayPal sought to quickly build a real-time decisioning platform that was highly effective while minimizing end-user friction.

As PayPal is constantly innovating to keep ahead in the race against fraud, Aerospike continues to help accelerate the evolution of PayPal's solution. PayPal's analysts and engineers understand the potential of graph analytics to provide insights for their use case. The relationships in PayPal's payment network are strong predictors of behaviors and great risk indicators that can reveal fraud.

PayPal's graph platform consists of real-time, interactive, and analytics graph technology stacks. The analytics graph platform is where the modeling happens. The interactive graph platform serves use cases where the query latency can be within a few seconds or minutes. The real-time graph platform serves use cases where the graph query results are needed in under a second.

Graph analysis at global scale

The common thread among all these customers was that they had requirements around performance and scalability from a graph solution. In each of these instances, they turned to Aerospike after failing to find a commercial solution that could check all the items in their graph data scalability and query performance list:

- Real-time query latency and optimization
- Near real time updates for data freshness
- Horizontal scalability with fault tolerance
- Million QPS (query per second) throughput
- Compliant to data security and privacy requirements

Seeing the success customers had with custom solutions based on Aerospike, the Aerospike team set out to deliver an official implementation that would benefit others based on two pillars: Aerospike and Apache Tinkerpop.

Apache TinkerPop is an open source computing framework for graph databases and graph analytic systems. Designed to appeal to software developers, TinkerPop lets developers add graph computing capabilities to their applications without worrying about developing APIs, graph processing engines, or graph algorithms.

Gremlin is the graph traversal language of Apache TinkerPop. Gremlin is a functional, data-flow language that enables users to succinctly express complex traversals on their application's graph. This is the Gremlin way of expressing graph queries and implementing graph algorithms. Every Gremlin traversal is composed of a sequence of (potentially nested) steps.

TinkerPop comes complete with documentation, a console, and a set of examples or "recipes" showing how to perform common graph-oriented tasks using Gremlin queries. All of that weighed in on the decision to adopt TinkerPop. Pairing TinkerPop and Aerospike's scalability, robustness and operational efficiency was a natural choice.

TinkerPop has a modular architecture and offers a SPI (Service Provider Interface) that makes it easy to plugin any backend into it. In the words of TinkerPop founder, Marko Rodriguez:

"Gremlin is not only a language for processing graphs, but also a virtual machine similar in many respects to the relationship between Java and the Java virtual machine. Microsoft's CosmosDB and Amazon's Neptune realized the benefit of this model and have become adopters of Apache TinkerPop."

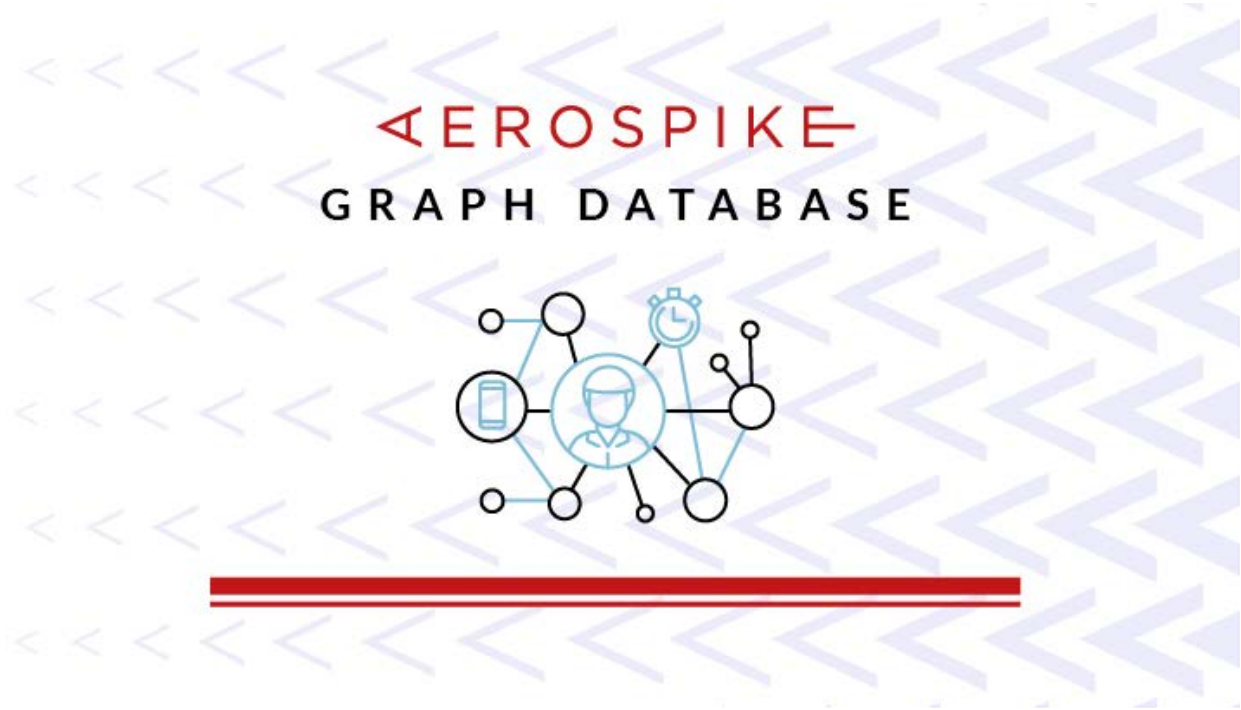
PayPal paved the way, as they were the first to implement a TinkerPop - Aerospike connector and reap the benefits. That implementation has proven to be very effective in fraud detection and prevention. Today, almost all PayPal risk models are using graph features.

Having proven itself in this environment, the combination of Aerospike and Tinkerpop was also a natural choice for Aerospike's investment in the graph data model. PayPal's use case served to validate the concept and inspire the Aerospike team.

Working closely with Rodriguez and a group of bonafide graph experts, the goal was to deliver a product that combines the strengths of both platforms. Rodriguez himself was heavily involved in architecting and implementing Aerospike Graph, endorsing the use of TinkerPop by Aerospike.

Today, Aerospike Graph checks all the items in the graph data scalability and query performance list.

Inside Aerospike Graph



Graph databases have been around for almost 20 years, but the attention they have been getting since 2017 is only accelerating. With AWS and Microsoft having moved into the domain, with Neptune and Cosmos DB they ended up exposing graph databases to a wider audience.

Based on the increasing usage, popularity and maturity of graph, Gartner estimates that the market for graph technologies, including graph database management systems (DBMSs), will grow to $3.2 billion by 2025 with a CAGR of 28.1%.

This hitherto niche domain has been the hottest in data management since then. Besides trends, however, there are real reasons why graph databases are interesting, and real use cases they can help with.

Investing in graph was a decision that Aerospike made for a number of reasons. The Aerospike team identifies a confluence of indicators shaping the direction the product is taking.

Integration

Organizations today are way past the single source of truth stage to drive decisions. They have multiple internal sources coupled with multiple external sources. They're adding new data sources continuously. Some Aerospike clients claim to add hundreds of data sources every quarter, some others even thousands.

But this is just the first step. The value comes from making sense of the relationships inherent in the data. Aerospike sees several types of use cases being served. Whether it's global payment systems, national or global retailers or supply chains, there is a common thread in all this.

Efficiency

We see graphs that are hand implemented. i.e. custom coded for that data model. When organizations like Adobe and Playtika take the step of assigning resources, expertise and time to implement a custom solution, that is a very strong signal which tells us two things.

First, this tells us that graph analysis is really important. Second, it also tells us that none of the existing solutions satisfy their needs. Aerospike felt that this is an opportunity to do better. Organizations will do better by leveraging domain expertise to move their business forward, letting Aerospike do the heavy lifting.

Scalability

To apply graph queries in practice and at scale, organizations have to handle thousands to millions of queries per second against very large graphs. The people who use Aerospike in the financial services world, e.g. large banks and brokerages, are all after scale.

When we talk about scale, this is not just about the large size of graphs. It's also about the number of queries going against that graph, anywhere from thousands per second to millions per second. That's the kind of performance that Aerospike routinely supports for key-value and document data models, and now that's available for graph data, too.

Aerospike Team

In 2023, Aerospike Graph was officially unveiled. Based on internal benchmarks, Aerospike Graph delivers millisecond multi-hop graph queries at extreme throughput across trillions of vertices and edges: more than 100,000 queries per second with sub-5ms latency achieved on modest infrastructure. Let's embark on a tour through the foundational principles underpinning Aerospike Graph. To get a basic understanding of Aerospike's underlying architecture and data model, please refer to the Appendix.

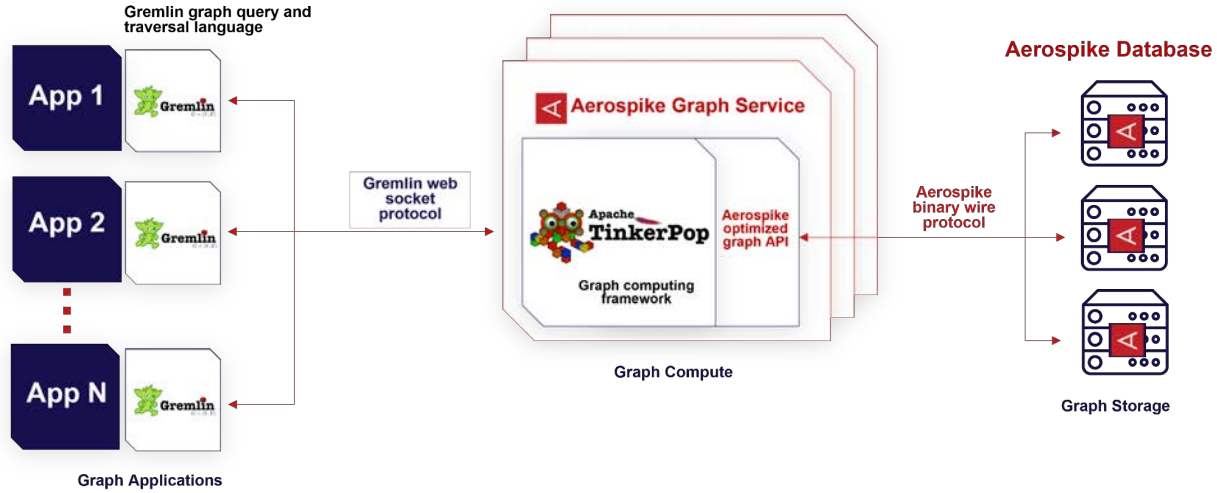
Aerospike Graph Technical Overview

Aerospike Graph leverages Apache TinkerPop, an open source graph computing framework for online transaction processing and online analytical processing graph queries. It's a mature

codebase that has been developed and tested since 2009. In Aerospike Graph, Aerospike Database is the underlying persistence layer for TinkerPop, the graph computing engine.

Aerospike Graph Service. This is the Graph API implementation that provides a deep integration between TinkerPop and Aerospike. This is where a highly optimized data model was created to represent graph elements.

Vertices and edges. Vertices and edges are mapped to the Aerospike data model using records (akin to rows for an RDBMS), bins (akin to columns), and other Aerospike features. Numerous traversal strategies and step optimizations were implemented by taking advantage of secondary indexes, expressions and other performance and scalability features of the Aerospike server.



Aerospike Graph Architecture - Gremlin query language and separate scaling of compute and storage

Gremlin. Aerospike Graph provides developers with a Gremlin interface. Gremlin is a graph query language. Aerospike Graph supports Gremlin out of the box as a first-class citizen in the TinkerPop ecosystem.

Optimization. The Gremlin virtual machine is the execution engine and Gremlin bytecode is used to program graph applications in Aerospike. Aerospike Graph makes use of more than 10 custom Gremlin steps, over 10 compiler optimization strategies, and extends Gremlin's profiling infrastructure.

Indexing. Aerospike Graph supports common indexes such as range queries and full-text queries with compiler strategies that leverage cardinality statistics effectively. While common in RDBMS, this type of optimized indexing is not common in NoSQL databases and much less so in the graph database space.

Small and Large nodes/vertices. Aerospike Graph makes a distinction between "small nodes" (few connections) and "large nodes" (super nodes). Small nodes maintain inline record edge lists

that reduce the number of database lookups during traversal. Large nodes maintain multi-record edge lists that allow for the lazy composition of edges as needed for traversal.

Partitioning. Aerospike Graph's encoding boils down to a standard distributed key-value adjacency list representation. A record's key is a node id and a record's value contains the node's properties and edges. The record value is partitioned such that properties, in-edges and out-edges can be accessed in O(1). That means it's a constant time algorithm that will always take the same amount of time to be executed.

Aerospike Graph Deployment Model

The Aerospike Graph deployment model consists of three main components: an application layer, a query language layer, and a data storage layer.

The Aerospike Graph Service transparently acts as a client on behalf of the user.

- The application layer may consist of multiple applications or application threads, using drivers in one or more programming languages. See Gremlin Drivers and Variants for more information.
- In the query layer, applications communicate with Aerospike Graph Service instances through a websocket protocol.
- In the data storage layer, Aerospike clusters may reside on bare metal, in on-premises servers or in cloud-based installations.

Aerospike Graph Present and Future

Aerospike foresees growing adoption of Aerospike Graph in organizations using it in use cases for AdTech, Customer 360, fraud detection and prevention, and more.

As highlighted by the PayPal use case, Aerospike Graph is currently optimized for OLTP-like use cases requiring real-time response. In parallel, the implementation is being expanded to include optimization of OLAP-like use cases.

Aerospike Graph supports a dedicated development and visualization environment via its partnership with G.V(), and comes with a built-in mechanism for importing data.

Aerospike invested in Graph as part of its long-term strategy to become the most powerful real-time, multi-model database. Aerospike supports graph, key-value and document data models; we are committed to onboarding existing users as well as new adopters of graph technology. In parallel, several new capabilities such as Graph Analytics and support for OpenCypher are being explored to enable users to extract even more value from their data.

Evaluating Graph Databases

In order to evaluate graph database offerings, a holistic mindset and framework should be adopted. A minimum of familiarity with the different graph data model options is required, as well as careful consideration of functional and non-functional criteria.

Graph data models

Graph databases are created to support data modeled as a graph as opposed to data modeled as tables, documents, or other data formats.

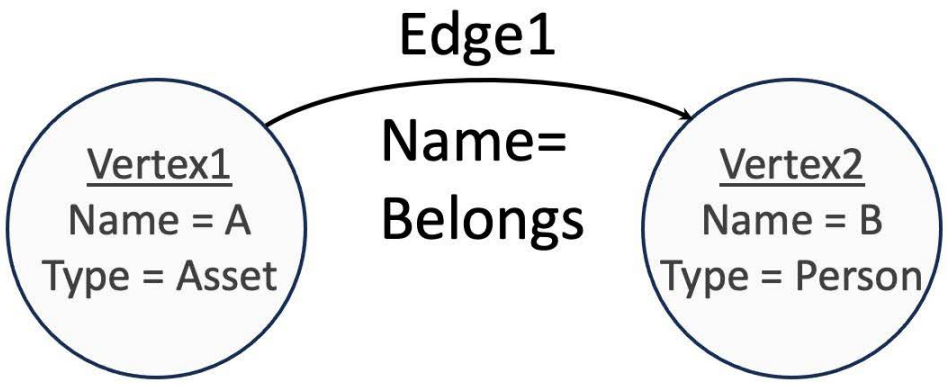
There are two common ways to model graph data: the labeled property graph (LPG) and the resource description framework (RDF). We describe each of them here, as the type of data model chosen is in close relationship to the types of use cases an organization is looking to implement.

LPG: Labeled Property Graphs

Labeled Property Graphs (LPG) model the world as a set of vertices and edges, where vertices represent entities and edges represent the relationships between them. Each vertex and edge may have a number of properties, hence the name "property graphs". In LPGs, vertices and edges may also have labels attached to them, representing names. This feature gives rise to the name "labeled property graph".

LPGs can be thought of as an abstraction that offers a graph API on top of a big key-value store - although that does not necessarily mean that all graph databases are implemented that way. However, unlike key-value stores, where only primitives such as put and get are available, LPGs group key-value pairs and associate them with a vertex or an edge. This grouping allows for more efficient querying and traversal of the graph.

Edges in LPGs can be directional, meaning that relationships can be defined in a specific way. For example, if we have a pair of vertices, A and B, and an edge with the label "Belongs" with direction from A to B, that would be interpreted as A belonging to B and not the other way around.



Labeled Property Graph (LPG) data model

It is important to note that not all LPG databases support explicit schemas. Without a schema, rules such as cardinality, property types, and relationships to other classes must be enforced by the application. Labels can be used to group vertices or edges together as members of the same class by using the same label for them.

In summary, Labeled Property Graphs provide a powerful and flexible way to model data as a graph, allowing for efficient querying and traversal. However, the lack of explicit schemas requires additional effort from the application to enforce rules and relationships.

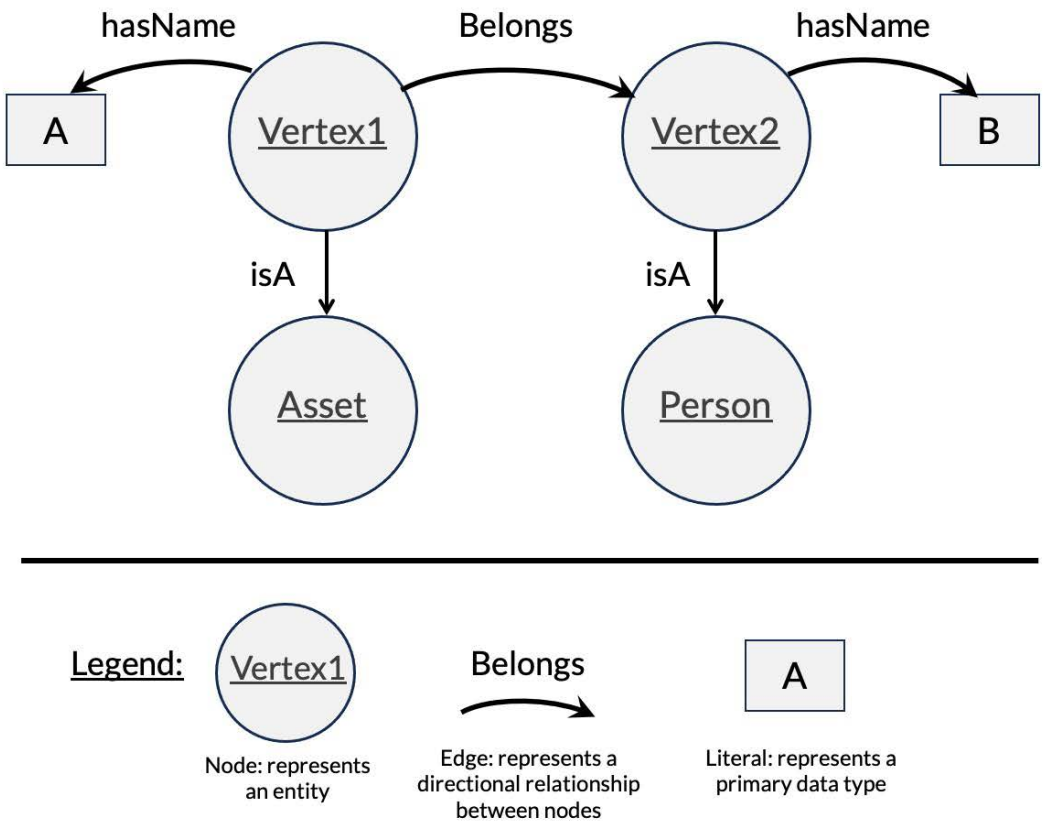
RDF: Resource Description Framework

Resource Description Framework (RDF) databases also have vertices, edges, and properties, but differ in their model and implementation details. RDF databases are also called triple stores, as they express the RDF model as subject-predicate-object triples.

RDF can express everything that Labeled Property Graphs (LPG) can, but requires more vertices and edges to describe the same graph model. However, RDF has the ability to express statements about edges, called *reification*, which makes it similar to LPG. This ability has become part of the RDF standard with the introduction of RDF-star, providing a standard way of expressing statements about edges in RDF.

RDF has several alternatives such as RDFS, OWL, and SHACL, which vary in their schema support and expressivity. These variations range from schema-free to complex rules and description logic. The simpler schema variation, SHACL, is gaining popularity and is now supported by RDF vendors.

All specifications in the RDF stack have formal semantics, which allows for unambiguous interpretation of data for inference or validation purposes. This is important when integrating datasets developed independently, as the schema acts as a contract between data developers and consumers.



Resource Description Framework (RDF) graph data model

Criteria for evaluating graph databases

Based on the different graph data models, use cases where schema and data integration are the main focus are typically better served by RDF graph databases, while use cases where fast traversals, analytics and algorithms are more important are typically better served by LPG graph databases.

Data integration use cases are assisted by RDF's strong support for semantics and federation, while LPG's typically more compact representation can be an advantage in traversals needed for graph algorithm execution. However, the boundaries are blurred and there is ongoing convergence. That, and the fact that knowledge graphs are emerging as a solution to ground generative AI models, makes it hard to opine on which segment is more likely to grow faster.

Evaluating what solution is most appropriate for the use cases of interest to an organization is a multi-factorial equation. For most organizations, performance, scalability and total cost of ownership are top of mind, and rightly so. However, evaluating performance is always complicated, and the performance of graph databases is no exception.

While benchmarks may offer indications, the devil is usually in the details: dataset, configuration, hardware, loading process, and fine-tuning. They all have an impact on how a graph database platform will perform. The best performance evaluation is always done using real data and representative conditions.

If this is not possible, however, we advise reviewing the work done by the Linked Data Benchmark Council (LDBC). LDBC brings industry and users together to develop benchmarks whereby advances in graph data management technologies can be assessed and directed.

Functional criteria

Some of the functional criteria to evaluate include how solutions fare in the following areas:

Analytics: Graph algorithms for analytics—for example the PageRank algorithm that traverses paths in a graph to measure properties of vertices and extract metrics—offer a benefit over graph processing frameworks that are external to the database.

Knowledge Graphs: Arguably the best technology to go beyond managing data to managing knowledge, these graphs connect data and add semantics to information, resulting in an interconnected network with value that is greater than the sum of its parts. While knowledge graphs can be stored in any type of underlying system, graph databases are a natural match.

Visual Exploration: Visual interfaces can enable ad hoc discovery that happens by picking a starting point of interest and following connections, as well as drive intentional exploration of a specific subset of graph, retrieved via graph queries or via analytics. Navigating and exploring

large graphs can be daunting, which is why visual exploration of graphs is part and parcel of many graph databases.

Data Import/Export: Graph databases routinely need to import data from other systems in formats that are not graph native, which means data must be ETLed and then imported en masse. Even for graph data, importing in batches is essential, as it saves effort and ensures a streamlined and repeatable process. Reliability and options for logging and resuming when errors occur during the process are key.

IDE Support: Integrated development environments are a huge factor in developer productivity. At the very minimum, graph databases should support popular IDEs via connectors, plugins, and libraries. Some graph databases incorporate their own custom IDEs, signifying an investment in onboarding users to produce graph-based applications.

Non-functional criteria

Non-functional criteria to evaluate include the following:

Developer Friendliness: The degree to which each solution facilitates application development is a function of its engine and API, data model, and query language. Being easy to work with is not something that can be quantified intuitively, but is nevertheless an important factor impacting productivity and quality.

User Accessibility: As the foundation for applications, analytics, and insights, databases need to be available to more target groups than developers to deliver value. Business intelligence (BI) tools typically are leveraged to enable this. Now graph databases are finding ways to work with these BI tools, in addition to their own tools to democratize access to data and perform graph analytics.

Operational Agility: Support for multiple environments, both on-premises and in the cloud, is a requirement for modern databases. This includes support for on-premises deployment on bare metal or virtual machines running on a variety of operating systems. Cloud deployments require capabilities ranging from user-managed container or virtual machine images to turnkey database-as-a-service (DaaS) solutions that integrate with cloud vendors' core and extended services.

Appendix: Aerospike Architecture and Data Model

Aerospike Architecture Overview

Aerospike was designed to serve as a distributed, scalable database. The architecture has three key objectives:

- Create a flexible, scalable, real-time platform for web-scale applications.
- Provide the robustness and reliability expected from traditional databases.
- Provide system elasticity and operational efficiency with minimal manual involvement.

The Aerospike architecture comprises three layers:

- Client Layer: This cluster-aware layer includes open source client libraries, which implement Aerospike APIs, track nodes, and know where data resides in the cluster thereby ensuring a single hop to data
- Clustering and Data Distribution Layer: This layer manages cluster communications and automates fail-over, replication, Cross-datacenter Replication (XDR), and intelligent re-balancing and data migration.
- Data Storage Layer: This layer reliably stores data in DRAM and Flash for fast retrieval.

Aerospike Data Model Basics

As a schemaless distributed database, Aerospike has a distinct data model for organizing and storing its data. The Aerospike database does not require a traditional schema. Rather, the data model is determined through use of the system. The main concepts in the Aerospike data model are as follows:

Physical storage. Aerospike can store data on any combination of DRAM, NVMe, SSD, PMem and traditional spinning media. Different namespaces in the same cluster can use distinct types of storage.

Namespace. Similar to a *tablespace* in an RDBMS, a namespace is a collection of records that share one specific storage engine, with common policies such as replication factor, encryption and more. A database can contain multiple namespaces.

Record. A record is an object similar to a *row* in an RDBMS. It is a contiguous storage unit for all the data uniquely identified by a single key.

Set. Records can be optionally grouped into sets. Sets are similar to *tables* in an RDBMS, but without an explicit schema.

Bin. A record is subdivided into bins, which are similar to *columns* in an RDBMS. Each bin has its own data type, and those do not need to agree between records. Secondary indexes can optionally be declared on bins.

Aerospike also supports primary, secondary and set indexes for optimized data access.

About Aerospike

The Aerospike Real-time Data Platform enables organizations to act instantly across billions of transactions while reducing cloud infrastructure up to 80%. The Aerospike data platform powers real-time applications with predictable sub-millisecond performance up to petabyte scale with five-nines uptime with globally distributed, strongly consistent data. Applications built on the Aerospike Real-time Data Platform fight fraud, provide recommendations that dramatically increase shopping cart size, enable global digital payments, and deliver hyper-personalized user experiences to tens of millions of customers. Customers such as Airtel, Experian, Nielsen, PayPal, Snap and Yahoo rely on Aerospike as their data foundation for the future.

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