MongoDB 3.4: What's New

January 2017

Table of Contents

Introduction 1

Multimodel Done Right 2
- Graph Processing 3
- Faceted Navigation 3
- Multi-Language Collations 4
- Aggregation Pipeline Enhancements 4
- Decimal Data Type 4
- MongoDB Connector for BI 5
- MongoDB Connector for Apache Spark 5

Mission-Critical Applications 6
- MongoDB Zones: Sophisticated Data Distribution 6
- Elastic Scalability 8
- Tunable Consistency Control 8
- Expanded Platform Support 9
- Enterprise-Grade Security 10

Modernized Tooling to Ship Applications Faster 11
- MongoDB Compass 11
- Operational Management for DevOps Teams 12

MongoDB Atlas: VPC Peering 13

Conclusion 14

We Can Help 14

Introduction

In the age of digital transformation and disruption, your ability to thrive depends on how quickly you adapt to a constantly changing market environment. MongoDB 3.4 is the latest release of the industry's fastest growing database. It offers a major evolution in capabilities and enhancements that enable you to address emerging opportunities and use cases:

Multimodel Done Right

- MongoDB 3.4 brings graph processing natively within the database, enabling efficient traversals across graphs, trees, and hierarchical data to uncover patterns and surface previously unidentified connections.
- New faceted navigation allows developers to create rich, intuitive search and analytics experiences for users. In addition, local language collations – the rules governing text comparisons and sorting – are now available for over 100 different languages and locales, enabling developers to extend global reach.
- Supporting the growing demand for real-time analytics against live, operational data, MongoDB enriches the aggregation pipeline with over 25 enhancements including new stages, expressions, array manipulation, and performance optimizations.
- The decimal data type brings new precision to the processing of financial and scientific data.
- The MongoDB Connector for BI has been re-engineered to push more SQL computation down into the database, accelerating time to insight for complex visualizations generated in industry leading analytics tools such as Tableau and BusinessObjects. The MongoDB Connector for Apache Spark has been updated to support the latest Spark 2.0 release, enabling data engineers to apply the latest innovations to sophisticated analytics pipelines.

Mission-Critical Applications

- MongoDB zones allow administrators to precisely control data placement within distributed clusters. Shards can be provisioned to to specific geographic regions for data sovereignty, or to support advanced deployment patterns such as tiered storage. Zones can

be configured visually with the MongoDB Ops Manager and Cloud Manager GUIs, or programmatically via their rich management APIs.

- Faster auto-balancing of data across nodes coupled with intra-cluster network compression allows seamless and elastic database scalability in response to dynamic application demands.
- New tunable consistency controls allow developers fine-grained optimization of data access patterns based on application requirements, including the ability to configure linearizable reads. MongoDB 3.4 offers among the strongest data consistency guarantees of any distributed database.
- Extended platform support means you can now run MongoDB anywhere – from new generations of power-efficient servers equipped with ARM processors, through to commodity x86 CPUs, all the way up to IBM Power and zSeries platforms.
- Enhanced security protection makes it even easier to deploy MongoDB into regulated industries and applications. Building upon existing LDAP authentication, MongoDB now supports LDAP authorization, allowing administrators to centrally define and manage user access control. In addition, new read-only views make it easy to implement field-level security through filtering and masking, reducing risks of data exposure.
- Support for AWS VPC peering has been added to the MongoDB Atlas hosted database service. Now users can create an extended, private network that connects the VPC housing their application servers with the AWS VPC containing their databases. VPC peering achieves this connectivity without using public IP addresses, and without needing to whitelist every client in your MongoDB Atlas group.

Modernized Tooling

- Compass, the GUI for MongoDB that makes it easy to explore and manipulate your data, is an incredibly powerful tool for developers and DBAs. It now includes full CRUD capabilities to edit documents, the ability to intuitively create and apply document validation rules, visual explain plans to explore query performance and real-time index usage statistics to help optimize performance.
- Simplified private cloud deployments for database as a service. MongoDB Ops Manager introduces Server Pools and native Cloud Foundry integration, making it easy to provision and manage database resources within cloud-native infrastructure.

MongoDB 3.4 is Generally Available and ready for production deployment now. MongoDB Atlas has been updated simultaneously with MongoDB 3.4 general availability, allowing you to quickly spin up and evaluate the new features on offer. To evaluate MongoDB 3.4 in your own environment, you can get it from the MongoDB download center. The Major Version Upgrade service from MongoDB global consulting is designed to accelerate the transition to the latest version of MongoDB. You will receive guidance from a consulting engineer on the necessary steps to upgrade, get a walk through of the upgrade process and get help on testing the upgraded application.

Each of the enhancements delivered by MongoDB 3.4 is covered in more detail through the rest of this whitepaper.

Multimodel Done Right

Rather than the monolithic codebases of the past, today's applications are increasingly being decomposed into loosely coupled suites of microservices, each implementing specific functionality within an application. Different services can place very different demands on the database used – from simple key-value lookups to complex analytics, aggregations, and graph traversals, through to rich search queries. Some data may need to be stored only in-memory for predictable low latency, while other data sets may need to be encrypted on disk for regulatory compliance. Data sets may vary from billions of small records, each just several KBs in size, to the management of large, multi-MB objects.

To try and tame the complexity that would come from using a multitude of storage technologies, the industry is moving towards the concept of "multimodel" databases. Such designs are based on the premise of presenting multiple data models within the same platform, thereby serving

diverse application requirements. However, many self-described multimodel database are little more than a collection of discrete technologies for data storage, search, and analytics, each with its own domain specific language, API and deployment requirements, and working on its own copy of the data. This approach to multimodel fails to offer much of an improvement over just running multiple independent databases, imposing high complexity, overhead, friction, and cost for developers and operations teams.

MongoDB takes a different approach:

- MongoDB's flexible document data model presents a superset of other database models. It allows data be represented as simple key-value pairs and flat, table-like structures, through to rich documents and objects with deeply nested arrays and sub-documents.
- With an expressive query language, documents can be queried in many ways – from simple lookups to creating sophisticated processing pipelines for data analytics and transformations, through to faceted search, JOINs and graph traversals.
- With a flexible storage architecture, application owners can deploy storage engines optimized for different workload and operational requirements.

MongoDB's approach to delivering multimodel significantly reduces developer and operational complexity, compared to running multiple, separate technologies to satisfy diverse application requirements. Users can leverage the same MongoDB query language, data model, scaling, security, and operational tooling across different applications, all within a single, integrated database platform.

MongoDB 3.4 introduces new native graph processing, faceted navigation, multi-language collations, additional aggregation pipeline operators, a new decimal data type, along with enhanced connectors for BI and Apache Spark integration.

Graph Processing

Applications storing data in MongoDB frequently contain data that represents graph or tree type hierarchies. These connections can be as simple as a management reporting chain in a HR application, or as complex as multi-directional, deeply nested relationships maintained by social networks, master data management, recommendation engines, disease taxonomy, fraud detection, and more. While special purpose graph databases are effective at storing and querying graph data, it's often desirable to store and traverse graph data directly in MongoDB. Here it can be processed, queried, and analyzed alongside all other operational data in real time, without the complexity of duplicating data across two separate databases.

Graph and hierarchical data is commonly queried to uncover indirect or transitive relationships. For example, if company "A" is owned by company "B", and "B" is owned by company "C", then "C" indirectly owns company "A". Some relational databases implement recursive Common Table Expressions (CTEs) to support these types of hierarchical queries. MongoDB 3.4 offers this functionality via a new aggregation stage called $graphLookup to recursively lookup a set of documents with a specific defined relationship to a starting document. Developers can specify the maximum depth for the recursion, and apply additional filters to only search nodes that meet specific query predicates. $graphLookup can recursively query within a single collection, or across multiple collections.

Review the documentation to learn more about the MongoDB $graphLookup operator for graph processing.

Faceted Navigation

Faceting is a popular analytics and search capability that allows an application to group information into related categories by applying multiple filters to query results. Facets allow users to narrow their search results by selecting a facet value as a filter criteria. Facets also provide an intuitive interface to exploring a data set, and allow convenient navigation to data that is of most interest.

A common use of faceting is the navigation of product catalogs. For example, a travel site can present vacation options by destination region, trip type (i.e. hotel, self-catering, beach, ski, city break), price band, season, and more, enabling users to quickly navigate to relevant categories. Facets also enable rapid analytics – extending our travel site example, business analysts can use facets to quickly compare sales by bucketing the number of trips sold by region and season.

Most databases need to execute multiple GROUP_BY statements to render facets, resulting in long running queries and poor user experience. MongoDB 3.4 introduces new aggregation pipeline stages for the bucketing, grouping and counting of one or more facets in a single round trip to the database. As a result, developers can generate richer and intuitive experiences to help users navigate complex data sets. Review the documentation to learn more about MongoDB faceted navigation.

Multi-Language Collations

Applications addressing global audiences require handling content that spans many languages. Each language has different rules governing the comparison and sorting of data. In order to create intuitive, localized user experiences, applications must handle non-English text with the appropriate rules for that language. For example, French has detailed rules for sorting names with accents on them. German phonebooks order words differently than the German dictionary.

MongoDB 3.4 significantly expands language support capabilities to allow users to build applications that adhere to language-specific comparison rules. Support for collations – the rules governing text comparisons and sorting – has been added throughout the MongoDB Query Language and indexes for over 100 different languages and locales. Each collation can also be further customized to provide precise control over case sensitivity, numeric ordering, whitespace handling, and more.

Developers can specify collation for a collection, an index, a view, or for specific operations that support collation (i.e. find, aggregate, update). You can learn more about collation in MongoDB from the documentation.

Aggregation Pipeline Enhancements

MongoDB developers and data engineers rely on the aggregation pipeline due to its power and flexibility in enabling sophisticated processing and manipulation demanded by real-time analytics and data transformations. MongoDB 3.4 continues to extend the aggregation pipeline by adding new capabilities within the database that simplify application-side code, as well as optimizer enhancements that improve performance.

In addition to the graph and facet features described earlier, many other expressions are added in MongoDB 3.4. These expressions address string manipulation, array handling, type handling, and schema detection and transformation:

- String handling includes expressions for splitting and manipulating strings either in bytes or code points (a code point can represent a single component of the string, e.g, a character, emoji, or formatting instruction).
- Array expressions allow more sophisticated manipulation and computations on arrays, including parallel array processing.
- New expressions allow determining types of fields
- Case/switch expressions for branching
- Support for ISO week expressions

MongoDB 3.4 also brings additional performance optimizations to the aggregation pipeline. Where possible, the query optimizer automatically moves the $match stage earlier in the pipeline, and combines it with other stages, to increase cases where indexes can be used to filter results sets. In most cases, no modifications of existing queries need to be made.

You can learn more about the many MongoDB 3.4 aggregation pipeline enhancements from the documentation.

Decimal Data Type

Decimal128 is a 16 byte decimal floating-point number format. It is intended for calculations on decimal numbers where high levels of precision are required, such as financial (i.e. tax calculations, currency conversions) and scientific computations. Decimal128 supports 34 decimal digits of significance and an exponent range of −6143 to +6144.

MongoDB 3.4 adds support for the decimal data type which represents decimal128 values. Unlike the double data type, which only stores approximations of decimal values, the decimal data type stores the exact value. For example, a decimal type ("9.99") has a precise value of 9.99, while 9.99 represented as a double would have an actual value of 9.9900000000000002131628207280 3,

thus creating the potential for rounding errors when it is used in calculations.

Decimal type values are treated like any other numeric type, and compare and sort correctly with other types based on actual numeric value. Operations on decimals are implemented in accordance with the decimal128 standard, so a value of 0.10 will retain its trailing zeros while comparing equal to 0.1, 0.10000 and so on.

Review the documentation to learn more about the new MongoDB decimal data type.

MongoDB Connector for BI

The MongoDB Connector for BI was introduced in November 2015. For the first time analysts, data scientists, and business users were able to seamlessly visualize semi-structured and unstructured data managed in MongoDB, alongside traditional data from their SQL databases, using the same BI tools deployed within millions of enterprises.

Building on its initial release, the Connector for BI has been reengineered to improve performance, simplify installation and configuration, and support Windows.

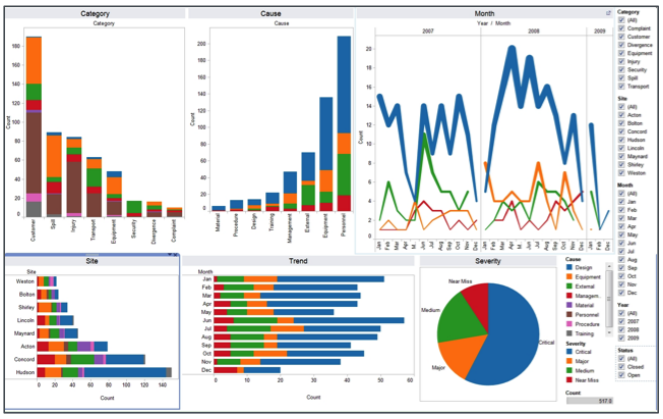

Figure 1: Uncover new insights with powerful visualizations generated from MongoDB

Performance and scalability has been improved by moving more query execution down to the MongoDB processes themselves. Queries and complex aggregations are executed natively within the database, thus reducing latency and bandwidth consumption. In addition, installation and authentication has been simplified. Users now authenticate as an existing user already declared within MongoDB, no longer needing to create separate username and password credentials within the connector. Further enhancing ease of use, the Connector for BI package is now reduced from four to two components:

- **mongodrdl** This tool connects to MongoDB and generates a document-relational definition language (DRDL) file, which maps the fields of a given MongoDB collection to a relational schema expected by the BI platform
- **mongosqld** Once installed and run as a daemon, mongosqld responds to SQL queries from the BI platform using the ubiquitous MySQL wire protocol supported by all of the major tools.

The Connector for BI is part of the Advanced Analytics suite available with MongoDB Enterprise Advanced. Review the MongoDB Connector for BI documentation to learn more.

MongoDB Connector for Apache Spark

Following general availability in June 2016, the MongoDB Connector for Apache Spark has been updated to support the latest Spark 2.0 release. Spark 2.0 support in the connector provides access to the new SparkSession entry point, the unified DataFrame and Dataset API, enhanced SparkSQL and SparkR functionality, and the experimental Structured Streaming feature. The connector exposes all of Spark's libraries, including Scala, Java, Python, and R. MongoDB data is materialized as DataFrames and Datasets for analysis through machine learning, graph, streaming, and SQL APIs.

Already powering sophisticated analytics at organizations including China Eastern Airlines, Black Swan, and x.ai, the MongoDB Connector for Apache Spark takes advantage of MongoDB's aggregation framework, rich queries, and secondary indexes to extract, filter, and process only the range of data it needs – for example, all customers located in a specific geography. This is very different from simple NoSQL datastores that do not offer secondary indexes or in-database aggregations. In these cases, Spark would need to extract all data based on a simple primary key, even if only a subset of that data is useful for the Spark process. This means more processing overhead, more

hardware, and longer time-to-insight for data scientists and engineers.

To maximize performance across large, distributed data sets, the connector can co-locate Resilient Distributed Datasets (RDDs) with the source MongoDB node, thereby minimizing data movement across the cluster and reducing latency.

You can download the MongoDB Connector for Apache Spark from GitHub, and sign up for a free Spark course from MongoDB University.

Mission-Critical Applications

Digital transformation imposes application demands that cannot be met easily or economically with traditional relational databases. Organizations are increasingly choosing non-relational technologies to accommodate a range of new requirements including the need to support growth in complex and rapidly changing data, geographically distributed applications, and agile development supported by Continuous Integration / Continuous Delivery (CI/CD) pipelines. Over 50% of the Fortune 100 are already MongoDB customers, using the database to modernize existing applications and innovate with new digital services. MongoDB 3.4 further enhances the ability of development teams to harness the latest innovations in database technology for their most mission-critical applications.

MongoDB Zones: Sophisticated Data Distribution

MongoDB provides horizontal scale-out for databases by partitioning data across low cost, commodity hardware using a technique called sharding. While many NoSQL databases also offer scale-out designs, MongoDB uniquely supports multiple sharding policies that give administrators precise control over how data is distributed across a cluster. As a result, data can be sharded according to query patterns or environmental considerations, providing higher scalability over diverse workloads and deployment architectures:

- **Range Sharding.** Documents are partitioned across shards according to the shard key value. Documents with shard key values close to one another are likely to be co-located on the same shard. This approach is well suited for applications that need to optimize range based queries.
- **Hash Sharding.** Documents are distributed according to an MD5 hash of the shard key value. This approach guarantees a uniform distribution of writes across shards, but is less optimal for range-based queries.
- **Zone Sharding.** Provides the the ability for DBAs and operations teams to define specific rules governing data placement in a sharded cluster.

MongoDB zones (superceding tag-aware sharding in earlier MongoDB releases) allow precise control over where data is physically stored, accommodating a range of deployment scenarios – for example by geographic region, by hardware configuration, or by application feature. Administrators can continuously refine data placement rules by modifying shard key ranges, and MongoDB will automatically migrate the data to its new zone.

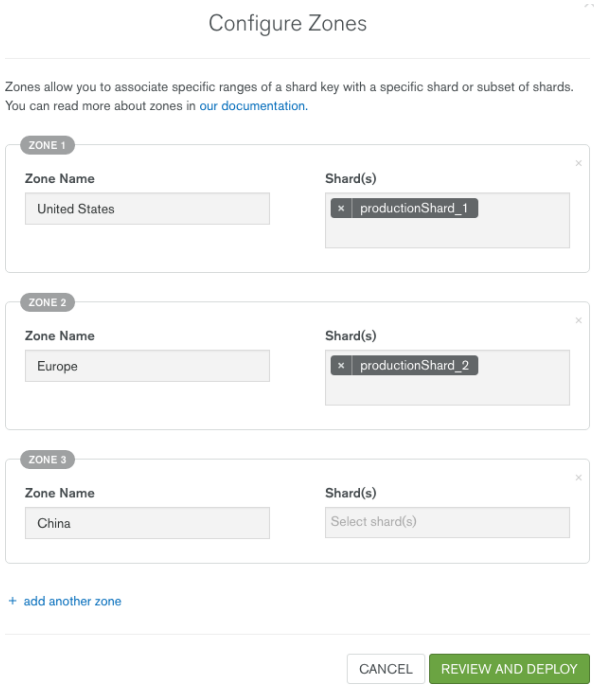

Figure 2: Easy and intuitive configuration of geographically distributed MongoDB zones via Ops Manager GUI

MongoDB 3.4 adds new helper functions and additional options in Ops Manager and Cloud Manager to configure zones – essential for managing large deployments.

The most popular use cases for MongoDB zones include the following:

Geographically Distributed Clusters

MongoDB 3.4 gives users the ability to create zones in multiple geographic regions. Each Zone is part of the same, single cluster and can be queried globally, but data resides in the correct location based on sovereignty and local access requirements. By associating data to shards based on user location, administrators are able to maintain low latency access.

To illustrate further, an application may have users in North America, Europe, and China. The application owner can assign each shard to a zone representing the physical location (North America, Europe, or China) of that shard's servers, and then map all documents to the correct zone based on its region field. Any number of shards can be associated with each zone, and each zone can be scaled independently of the others – for instance, accommodating faster user growth in China than North America.

Learn more by reviewing our tutorial on creating geographically distributed clusters with MongoDB zones.

Localized Writes in a Distributed Cluster

Zones provide a solution for continuous availability of insert-only workloads such as the ingestion of sensor data in IoT applications. Zones can be used to create configurations specifically for localized writes in a distributed cluster, ensuring there is always a node available to accept inserts, even during a data center failure. As demonstrated in Figure 3, the topology for distributed local writes will maintain nodes from both shards in each data center. In case a datacenter is unavailable, application side logic can automatically modify the shard key to redirect the write to the alternative data center.

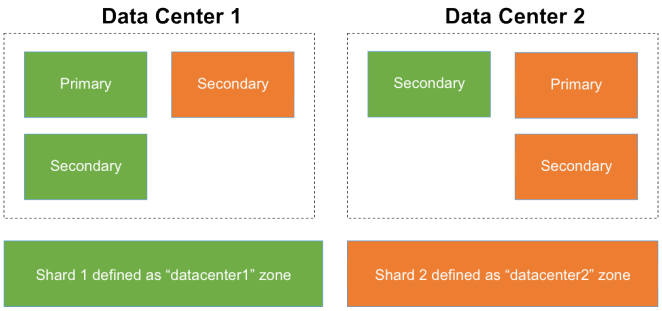

Figure 3: Maintaining continuous availability and local writes with MongoDB zones

Learn more by reviewing our tutorial on configuring localized writes with MongoDB zones.

Tiered Storage

Different subsets of the database may have different response time requirements, usually based on access frequency and age of the data. For example, IoT applications or social media services handling time-series data will demand users experience the lowest latency when accessing the latest data. However aged data sets that are read less frequently from an "active archive" have relaxed latency SLAs.

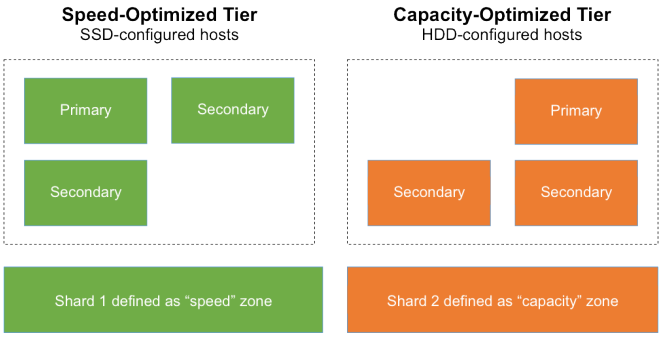

Figure 4: Implementing tiered storage with MongoDB zones

Asymmetric hardware configurations within a sharded cluster can be created and managed with zones. By implementing a tiered storage pattern:

- The most recent data can be located on the highest performance hardware with fast CPUs and SSDs
- Aged data can be moved onto slower, but less expensive hardware based on conventional, high capacity spinning disks.

As a result, administrators can optimize the hardware spread across the two tiers, providing a great user experience at an optimized cost point. By including a timestamp in the shard key, the MongoDB cluster balancer can migrate data based on age from the high performance tier to the active archive tier. Learn more by reviewing our tutorial on configuring tiered storage with MongoDB zones.

Application Affinity

Data for a specific application feature or customer can be associated to specific zones. For instance, a company offering Software-as-a-Service (SaaS) may assign users on its free usage tier to shards provisioned onto lower specified hardware, while paying customers are allocated to richly configured, premium infrastructure. The SaaS provider has the flexibility to scale parts of the cluster differently for free users and paying customers. For example, the free tier can be provisioned to just several shards, while paying customers can be assigned to dozens of shards.

Learn more by reviewing our tutorial on configuring application affinity with MongoDB Zones.

Elastic Scalability

Many modern workloads have unpredictable performance and capacity demands. Loads can quickly spike in response to specific events, and then stabilize to regular levels. For example, MongoDB is used by some of the largest online gaming companies in the world. Rave reviews of newly published titles drive a sudden surge in usage, which then decline once the initial launch hype has passed. Time-based promotions also drive regular spikes in demand for the game, temporarily increasing load on the platform.

Responding to such dynamic workloads requires elastic database clusters that allow for capacity to be seamlessly added and removed on demand. This allows the business to get the scalability it needs, when it needs it, and can then reduce resources when demand drops, thus avoiding over provisioning and containing costs. MongoDB 3.4 adds a range of enhancements to further support elastic clusters.

Faster Cluster Balancing & Node Synchronization

The MongoDB sharded cluster balancer, responsible for evenly distributing data across the nodes of a cluster, has been improved, allowing users to scale capacity quickly and easily with minimal operational overhead.

The balancer process now supports parallel data migrations. Multiple node pairs can perform balancing migrations simultaneously, significantly improving balancing throughput as nodes are added or removed from the cluster, or as data is redistributed across nodes. An individual node can be involved in at most one migration at a time, so the benefits of parallelized balancing will be observed in clusters with four or more shards. In addition, with WiredTiger as the default MongoDB storage engine, balancer throttling that was necessary with the earlier MMAP engine is now off by default, which dramatically speeds migrations, by as much as 10x in some deployments.

Adding new replica set members to a MongoDB cluster has also been improved with an optimized “initial sync” process. Initial sync is implemented by copying all data from an existing replica to the newly added replica member. Initial sync is typically used when adding nodes to a cluster, migrating to a new MongoDB storage engine, or restoring a node that has fallen too far behind the replication process. MongoDB 3.4 offers the following enhancements to initial sync:

- Indexes are now created as data is copied, rather than after copying is complete, therefore reducing IO overhead and improving overall initial sync times, especially when synchronizing large data sets between nodes.
- The initial sync retry logic has been updated to be highly resistant to transient network issues, which significantly reduces the need to restart the copying process.

Intra-Cluster Network Compression

As a distributed database, MongoDB relies on efficient network transport during query routing and inter-node replication. MongoDB 3.4 introduces a new option to compress the wire protocol used for intra-cluster communications. Based on the snappy compression

algorithm, network traffic can be compressed by up to 70%, providing major performance benefits in bandwidth-constrained environments, and reduced networking costs. One early access tester was able to reduce its networking bill by over $20,000 a month after configuring network compression.

Compressing and decompressing network traffic requires CPU resources – typically imposing a low single digit percentage overhead. Compression is ideal for those environments where performance is bottlenecked by bandwidth, and sufficient CPU capacity is available.

Tunable Consistency Control for Application Flexibility

MongoDB is strongly consistent by default, enabling applications to immediately read what has been written to the database, thus avoiding the developer complexity imposed by eventually consistent systems.

Secondary Consistency Control

In some scenarios, it is acceptable to trade consistency against specific performance goals. Examples include serving read traffic from replicas in a remote data center to reduce geographic latency, or distributing reporting queries across secondary replica set members. With Secondary Consistency Control, MongoDB 3.4 allows application owners to dial up or dial down consistency levels, based on application needs.

Through read preferences, MongoDB gives users control over how and when to route queries to secondary replica set members. As secondary replica members replicate the primary member's writes asynchronously, each can lag the primary's live state by different amounts of time. With MongoDB 3.4, a request to process a query by a secondary replica can now be configured to indicate how much lag the request is willing to accept when choosing which secondary node to service the query. If there are no nodes available that satisfy the consistency window, an error is returned to the application.

Unlike other databases, being able to configure acceptable data consistency levels within MongoDB allows the user to improve data quality, while maintaining the ability to scale read traffic across secondary replicas. Consider a media company using MongoDB to power a live dashboard reporting the results of A/B testing against different headlines and copy. The editorial team may want to ensure they are reporting against near real-time data that is less than, for example, 15 seconds old. Secondary Consistency Control gives them the ability to configure for data freshness, thus improving accuracy and decision making.

Linearizable Reads

Dialing consistency levels up, MongoDB 3.4 adds a new readConcern level of "linearizable". This option confirms the primary replica is still connected to a quorum (majority) of replica nodes before returning results to the client. When used to perform reads against a single document, linearizable read concern provides two guarantees:

- First, it guarantees that the returned data reflects only writes that are committed to a majority of nodes in the replica set, and therefore will not roll back in the future as a result of a replica set election.
- Second, it guarantees that the read is not stale. This means that the returned data reflects the last write operation to the document that successfully replicated to a majority of nodes. If a new primary replica is elected and a client writes to a document using that new primary – and that write propagates to a majority of nodes – a subsequent read by any client using linearizable read concern will be guaranteed to reflect that write or return an error, regardless of which node is used to service the read.

In order to provide the extra guarantees, using linearizable read concern level will have a significant impact on read latency.

With the linearizable read concern, MongoDB offers among the strongest data consistency guarantees of any modern, distributed database. You can learn more by reviewing the linearizable read concern documentation

Expanded Platform Support

As MongoDB adoption accelerates, there has been growing demand to run the database on a more diverse range of platforms to support a broader set of use-cases:

- MongoDB 3.4 has been ported to the ARM v8-64 bit platform, supporting new generations of power-efficient servers being deployed into ultra-dense data center racks.
- MongoDB 3.4 has been ported to IBM's POWER8 and zSeries platforms, providing a seamless migration for large enterprises modernizing legacy workloads as part of digital transformation initiatives. The port is available for the MongoDB Enterprise Server, available as part of MongoDB Enterprise Advanced.

All of these new ports are available from the MongoDB download center

Enterprise-Grade Security for Regulatory Compliance

With widespread usage across financial services, healthcare, retail, and government, MongoDB offers some of the most extensive security controls available in modern databases. Robust access control, end-to-end encryption, and auditing for forensic analysis enable organizations to build regulatory compliant apps. MongoDB 3.4 further extends security protection with new LDAP authorization and read-only views.

LDAP Authorization

LDAP is widely used by organizations to standardize and simplify the way large user populations are managed across internal systems and applications. In many cases, LDAP is also used as the centralized authority for user access control to ensure that internal security policies are compliant with corporate and regulatory guidelines.

MongoDB 3.4 extends existing support for authenticating users via LDAP to now include LDAP authorization as well. This enables existing user privileges stored in the LDAP server to be mapped to MongoDB roles, without users having to be recreated in MongoDB itself. When configured with an LDAP server for authorization, MongoDB 3.4 will allow user authentication via LDAP, Active Directory, Kerberos, or X.509 without requiring local user documents in the $external database. When a user successfully authenticates, MongoDB will perform a query against the LDAP server to retrieve all groups the LDAP user is a member of, and will transform those groups into their equivalent MongoDB roles.

MongoDB 3.4 now leverages native platform libraries to integrate with LDAP. This removes the need for the external sasld dependencies and configuration required in earlier releases, while also adding support for LDAP when running MongoDB on Windows. In addition, LDAP authentication and authorization can now be configured in Ops Manager, rather than separately via the command line on each MongoDB node.

The LDAP enhancements in MongoDB 3.4 significantly reduce administrative overhead and TCO, while allowing seamless MongoDB integration into centralized enterprise access management infrastructure. You can learn more about the LDAP enhancements in MongoDB 3.4 from the documentation. MongoDB Enterprise Advanced is required to take advantage of LDAP integration.

Read-Only Views

New in MongoDB 3.4, DBAs can define non-materialized views that expose only a subset of data from an underlying collection, i.e. a view that filters out specific fields, such as Personally Identifiable Information (PII) from sales data or health records, or filter out entire documents, such as customers who have opted out of marketing communications. As a result, risks of data exposure are dramatically reduced. DBAs can define a view of a collection that's generated from an aggregation over another collection(s) or view. Permissions granted against the view are specified separately from permissions granted to the underlying collection(s). This capability allows organizations to more easily meet compliance standards in regulated industries by restricting access to sensitive data, without creating the silos that emerge when data has to be broken apart to reflect different access privileges.

Views can also contain computed fields – for example summarizing total and average order value per region, without exposing underlying customer data. All of this can be done without impacting the structure or content of the original source collections. Developers and DBAs can modify the underlying collection's schema without impacting applications using the view.

As views are non-materialized, the view data is generated dynamically by reading from the underlying collections when a user queries the view. This reduces data duplication in the database, and eliminates inconsistencies between the base data and view.

Views are defined using the standard MongoDB Query Language and aggregation pipeline. They allow the inclusion or exclusion of fields, masking of field values, filtering, schema transformation, grouping, sorting, limiting, and joining of data using $lookup and $graphLookup to another collection.

You can learn more about MongoDB read-only views from the documentation.

Modernized Tooling to Ship Applications Faster

To preserve existing investments in skills and enterprise infrastructure standards, MongoDB is provided with the extensive tooling for data modeling and systems management expected by DBAs and operations teams. MongoDB 3.4 includes enhancements to Compass and Ops Manager, enabling IT teams to ship new applications faster with less effort and cost, while gaining greater oversight and control over their entire IT estate.

MongoDB Compass

MongoDB Compass is the easiest way for developers and DBAs to explore and manage MongoDB data. As the GUI for MongoDB, Compass enables users to visually explore their data, and run ad-hoc queries in seconds – all with zero knowledge of MongoDB's query language.

The latest Compass release expands functionality to allow users to manipulate documents directly from the GUI, optimize performance, and create data governance controls.

Developers and DBAs can interact with and manipulate MongoDB data from Compass. They can edit, insert, delete, or clone existing documents to fix data quality or schema issues in individual documents identified during data exploration. If a batch of documents need to be updated, the query string generated by Compass can be used in an update command within the mongo shell.

Trying to parse text output can significantly increase the time to resolve query performance issues. Visualization is core to Compass, and has now been extended to generating real-time performance statistics, and presenting indexes and explain plans.

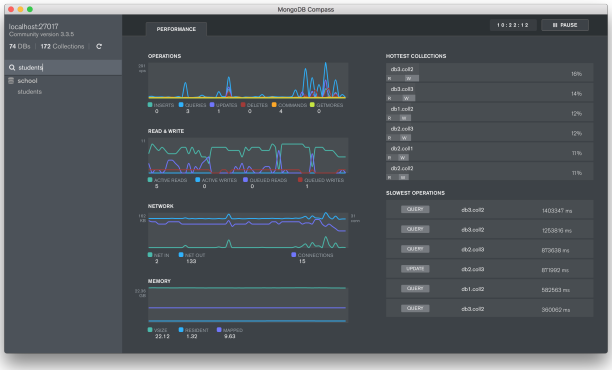

Figure 5: Real-time performance statistics now available from MongoDB Compass

- The visualization of the same real-time server statistics generated by the mongotop and mongostat commands directly within the Compass GUI allows DBAs to gain an immediate snapshot of server status and query performance.
- If performance issues are identified, DBAs can visualize index coverage, enabling them to determine which specific fields are indexed, their type, size, and how often they are used.
- Compass also provides the ability to visualize explain plans, presenting key information on how a query performed – for example the number of documents returned, execution time, index usage, and more. Each stage of the execution pipeline is represented as a node in a tree, making it simple to view explain plans from queries distributed across multiple nodes.

If specific actions, such as adding a new index, need to be taken, DBAs can use MongoDB's management tools to automate index builds across the cluster.

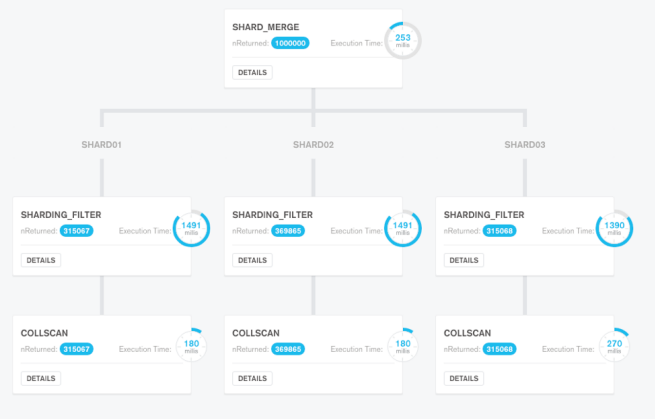

Figure 6: MongoDB Compass visual query plan for performance optimization across distributed clusters

Document validation allows DBAs to enforce data governance by applying checks on document structure, data types, data ranges, and the presence of mandatory fields. Validation rules can now be managed from the Compass GUI. Rules can be created and modified directly using a simple point and click interface, and any documents violating the rules can be clearly presented. DBAs can then use Compass's CRUD support to fix data quality issues in individual documents.

MongoDB Compass is included with both MongoDB Professional and MongoDB Enterprise Advanced subscriptions used with your self-managed instances, or hosted MongoDB Atlas instances. MongoDB Compass is free to use for evaluation and in development environments. You can get MongoDB Compass from the download center, and read about it in the documentation.

Operational Management for DevOps Teams

Ops Manager is the simplest way to run MongoDB on your own infrastructure, making it easy for operations teams to deploy, monitor, backup, and scale MongoDB. Ops Manager is available as part of MongoDB Enterprise Advanced, and its capabilities are also available in Cloud Manager, a tool hosted by MongoDB in the cloud. Ops Manager and Cloud Manager provide an integrated suite of applications that manage the complete lifecycle of the database:

- Automated deployment and management with a single click and zero-downtime upgrades
- Proactive monitoring providing visibility into the performance of MongoDB, history, and automated alerting on 100+ system metrics
- Disaster recovery with continuous, incremental backup and point-in-time recovery, including the restoration of complete running clusters from your backup files

Ops Manager has been enhanced as part of the MongoDB 3.4 release, now offering:

- Finer-grained monitoring telemetry
- Configuration of MongoDB zones and LDAP security
- Richer private cloud integration with server pools and Cloud Foundry
- Encrypted backups
- Support for Amazon S3 as a location for backups

Ops Manager Monitoring

For modern applications and distributed systems, it should be possible to configure the granularity of telemetry data collected by system monitoring agents. Finer-grained telemetry can help administrators more accurately pinpoint and root-cause specific events, but comes at the cost of increased data storage overhead. Ops Manager now allows telemetry data to be collected every 10 seconds, up from the previous minimum 60 seconds interval. By default, telemetry data at the 10-second interval is available for 24 hours. 60-second telemetry is retained for 7 days, up from the previous 48-hour period. These retention policies are now fully configurable, so administrators can tune the timelines available for trend analysis, capacity planning, and troubleshooting.

Generating telemetry views synthesized from hardware and software statistics helps administrators gain a complete view of each instance to better monitor and maintain database health. Ops Manager has always displayed hardware monitoring telemetry alongside metrics collected from the database, but required a third party agent to collect the raw hardware data. The agent increased the number of system components to manage, and was only available for Linux hosts. The Ops Manager agent has now been extended to collect hardware statistics, such as disk

utilization and CPU usage, alongside existing MongoDB telemetry. In addition, platform support has been extended to include Windows and OS X.

Private Cloud Integration

Many organizations are seeking to replicate benefits of the public cloud into their own infrastructure through the build-out of private clouds. A number of organizations are using MongoDB Enterprise Advanced to deliver an on-premise Database-as-a-Service (DBaaS). This allows them to standardize the way in which internal business units and project teams consume MongoDB, improving business agility, corporate governance, cost allocation, and operational efficiency.

Ops Manager now provides the ability to create pre-provisioned server pools. The Ops Manager agent can be installed across a fleet of servers (physical hardware, VMs, AWS instances, etc.) by a configuration management tool such as Chef, Puppet, or Ansible. The server pool can then be exposed to internal teams, ready for provisioning servers into their local groups, either by the programmatic Ops Manager API or the Ops Manager GUI. When users request an instance, Ops Manager will remove the server from the pool, and then provision and configure it into the local group. It can return the server to the pool when it is no longer required, all without sysadmin intervention. Administrators can track when servers are provisioned from the pool, and receive alerts when available server resources are running low. Pre-provisioned server pools allow administrators to create true, on-demand database resources for private cloud environments. You can learn more about provisioning with Ops Manager server pools from the documentation.

Building upon server pools, Ops Manager now offers certified integration with Cloud Foundry. BOSH, the Cloud Foundry configuration management tool, can install the Ops Manager agent onto the server configuration requested by the user, and then use the Ops Manager API to build the desired MongoDB configuration. Once the deployment has reached goal state, Cloud Foundry will notify the user of the URL of their MongoDB deployment. From this point, users can log in to Ops Manager to monitor, back-up, and automate upgrades of their deployment.

MongoDB Ops Manager is available for evaluation from the download center.

Amazon S3 Support

Ops Manager can now store backups in the Amazon S3 storage service, with support for deduplication, compression, and encryption. The addition of S3 provides administrators with greater choice in selecting the backup storage architecture that best meets specific organizational requirements for data protection:

- MongoDB blockstore backups
- Filesystem backups (SAN, NAS, & NFS)
- Amazon S3 backups

Whichever architecture is chosen, administrators gain all of the benefits of Ops Manager, including point-in-time recovery of replica sets, cluster-wide snapshots of sharded databases, and data encryption.

You can learn more about Ops Manager backups from the documentation.

MongoDB Atlas: VPC Peering

Since launching the MongoDB Atlas database service in June 2016, over 2,000 clusters have already been deployed. Today MongoDB Atlas is providing production services to diverse organizations and applications across the globe, including media broadcasters, gaming platforms, social media analytics, financial services, logistics, scientific research, and universities. MongoDB Atlas provides the features of MongoDB, without the operational heavy lifting required for any new application. MongoDB Atlas is available on-demand through a pay-as-you-go model and billed on an hourly basis, letting developers focus on apps, rather than ops.

MongoDB Atlas offers the latest 3.4 release (community edition) as an option. In addition, MongoDB Atlas also now offers AWS Virtual Private Cloud (VPC) peering. Each MongoDB Atlas group is provisioned into its own AWS VPC, thus isolating the customer's data and underlying systems from other MongoDB Atlas users. With the addition of VPC peering, customers can now connect their

application servers deployed to another AWS VPC directly to their MongoDB Atlas cluster using private IP addresses. Whitelisting public IP addresses is not required for servers accessing MongoDB Atlas from a peered VPC. Services such as AWS Elastic Beanstalk or AWS Lambda that use non-deterministic IP addresses can also be connected to MongoDB Atlas without having to open up wide public IP ranges that could compromise security. VPC peering allows users to create an extended, private network connecting their application servers and backend databases.

You can learn more about MongoDB Atlas from the documentation.

Conclusion

MongoDB 3.4 is a significant evolution of the industry's fastest growing database:

- Native graph processing, faceted navigation, richer real-time analytics, and powerful connectors for BI and Spark integration bring additional multimodel database support right into MongoDB.
- Geo-distributed MongoDB zones, elastic clustering, tunable consistency, and enhanced security controls bring state-of-the-art database technology to your most mission-critical applications.
- Enhanced DBA and DevOps tooling for schema management, fine-grained monitoring, and cloud-native integration allow engineering teams to ship applications faster, with less overhead and higher quality.

To get started today:

- Download MongoDB 3.4
- Alternatively, spin up your own MongoDB 3.4 cluster on the MongoDB Atlas database service
- Sign up for our free 3.4 training from the MongoDB University

We Can Help

We are the MongoDB experts. Over 2,000 organizations rely on our commercial products, including startups and more than a half of the Fortune 100. We offer software and services to make your life easier:

MongoDB Enterprise Advanced is the best way to run MongoDB in your data center. It's a finely-tuned package of advanced software, support, certifications, and other services designed for the way you do business.

MongoDB Atlas is a database as a service for MongoDB, letting you focus on apps instead of ops. With MongoDB Atlas, you only pay for what you use with a convenient hourly billing model. With the click of a button, you can scale up and down when you need to, with no downtime, full security, and high performance.

MongoDB Cloud Manager is a cloud-based tool that helps you manage MongoDB on your own infrastructure. With automated provisioning, fine-grained monitoring, and continuous backups, you get a full management suite that reduces operational overhead, while maintaining full control over your databases.

MongoDB Professional helps you manage your deployment and keep it running smoothly. It includes support from MongoDB engineers, as well as access to MongoDB Cloud Manager.

Development Support helps you get up and running quickly. It gives you a complete package of software and services for the early stages of your project.

MongoDB Consulting packages get you to production faster, help you tune performance in production, help you scale, and free you up to focus on your next release.

MongoDB Training helps you become a MongoDB expert, from design to operating mission-critical systems at scale. Whether you're a developer, DBA, or architect, we can make you better at MongoDB.

Resources

For more information, please visit mongodb.com or contact us at sales@mongodb.com.

Case Studies (mongodb.com/customers)
Presentations (mongodb.com/presentations)
Free Online Training (university.mongodb.com)
Webinars and Events (mongodb.com/events)
Documentation (docs.mongodb.com)
MongoDB Enterprise Download (mongodb.com/download)
MongoDB Atlas database as a service for MongoDB (mongodb.com/cloud)

mongoDB

New York • Palo Alto • Washington, D.C. • London • Dublin • Barcelona • Sydney • Tel Aviv
US 866-237-8815 • INTL +1-650-440-4474 • info@mongodb.com