MongoDB 7.0 General Changes

Cache Refresh Time Fields

Starting in MongoDB 7.0, log messages for slow queries includes a new cache refresh time field.

Concurrent Storage Engine Transactions (Read and Write Tickets)

Starting in version 7.0, MongoDB uses a default algorithm to dynamically adjust the maximum number of concurrent storage engine transactions (including both read and write tickets) to optimize database throughput during overload.

The following table summarizes how to identify overload scenarios for MongoDB 7.0 and prior releases:

Version	Diagnosing Overload Scenarios
7.0	A large number of queued operations that persists for a prolonged period of time likely indicates an overload. A concurrent storage engine transaction (ticket) availibility of 0 for a prolonged period of time **does not** indicate an overload.
6.0 and prior	A large number of queued operations that persists for a prolonged period of time likely indicates an overload. A concurrent storage engine transaction (ticket) availibility of 0 for a prolonged period of time likely indicates an overload.

To learn more, see:

- `wiredTiger.concurrentTransactions`
- `storageEngineConcurrentReadTransactions`
- `storageEngineConcurrentWriteTransactions`

`currentOp` Metrics

Starting in MongoDB 7.0, the `currentOp` command and the `db.currentOp()` method include these new fields:

- `currentOp.admissionPriority`
- `currentOp.collUuid`
- `currentOp.startTime`
- `currentOp.samplesPerSecond`
- `currentOp.sampledReadsCount`
- `currentOp.sampledWritesCount`
- `currentOp.sampledReadsBytes`
- `currentOp.sampledWritesBytes`

`$currentOp` (aggregation) Metrics

Starting in MongoDB 7.0, the currentOp aggregation stage includes these new fields:

- `$currentOp.collUuid`
- `$currentOp.startTime`
- `$currentOp.samplesPerSecond`
- `$currentOp.sampledReadsCount`
- `$currentOp.sampledWritesCount`
- `$currentOp.sampledReadsBytes`
- `$currentOp.sampledWritesBytes`

Compound Wildcard Indexes

Starting in MongoDB 7.0, you can create compound wildcard indexes. A compound wildcard index has one wildcard term and one or more additional index terms.

Use compound wildcard indexes to support queries on known patterns and to limit the total number of indexes in a collection.

Large Change Stream Events

Starting in MongoDB 7.0, if you have change stream events larger than 16 MB, you can use the new `$changeStreamSplitLargeEvent` stage to split the events into smaller fragments.

The following new metrics report information about large change stream events:

- `metrics.changeStreams`
- `metrics.changeStreams.largeEventsFailed`
- `metrics.changeStreams.largeEventsSplit`

serverStatus Output Change

`serverStatus` includes the following new fields in its output:

Plan Cache Metrics

- `planCache`

queryAnalyzer Metrics

- `queryAnalyzers.activeCollections`
- `queryAnalyzers.totalCollections`
- `queryAnalyzers.totalSampledReadsCount`
- `queryAnalyzers.totalSampledWritesCount`
- `queryAnalyzers.totalSampledReadsBytes`
- `queryAnalyzers.totalSampledWritesBytes`

Slot-Based Query Execution Engine

Starting in MongoDB 7.0, the slot-based query execution engine improves performance for a wider range of find and aggregation queries.

Slow query log messages now include a `queryFramework` field that indicates which query engine completed the query:

- `queryFramework: "classic"` indicates that the classic engine completed the query.
- `queryFramework: "sbe"` indicates that the slot-based query execution engine completed the query.

User Roles System Variable

Starting in MongoDB 7.0, you can use the new `USER_ROLES` system variable to return the roles of the current user.

For use cases that include `USER_ROLES`, see the find, aggregation, view, updateOne, updateMany, and findAndModify examples.

New Sharding Statistics for Chunk Migrations

Available starting in MongoDB 7.0 (and 6.3.2, 6.0.6, and 5.0.18).

MongoDB includes the following new sharding statistics for chunk migrations:

- `shardingStatistics.countDonorMoveChunkCommitted`
- `shardingStatistics.countDonorMoveChunkAborted`
- `shardingStatistics.totalDonorMoveChunkTimeMillis`
- `shardingStatistics.countBytesClonedOnRecipient`

- `shardingStatistics.countDocsClonedOnCatchUpOnRecipient`
- `shardingStatistics.countBytesClonedOnCatchUpOnRecipient`

New Slow Query Log Message

Starting in MongoDB 7.0 (and 6.0.13, 5.0.24),
the `totalOplogSlotDurationMicros` in the slow query log message shows the time between a write operation getting a commit timestamp to commit the storage engine writes and actually committing. `mongod` supports parallel writes. However, it commits write operations with commit timestamps in any order.

To learn more, see Logging Slow Operations.

New Parameters

analyzeShardKey-related Parameters

MongoDB 7.0 adds the following parameters related to
the `analyzeShardKey` command:

- `analyzeShardKeyCharacteristicsDefaultSampleSize`
- `analyzeShardKeyNumMostCommonValues`
- `analyzeShardKeyNumRanges`
- `analyzeShardKeyMonotonicityCorrelationCoefficientThreshold`

autoMergerIntervalSecs Parameter

MongoDB 7.0 adds the `autoMergerIntervalSecs` parameter which, when AutoMerger is enabled, specifies the amount of time between automerging rounds, in seconds. `autoMergerIntervalSecs` can only be set on config servers of sharded clusters.

autoMergerThrottlingMS Parameter

MongoDB 7.0 adds the `autoMergerThrottlingMS` which, when AutoMerger is enabled, specifies the minimum amount time between merges initiated by the AutoMerger on the same collection, in milliseconds. `autoMergerThrottlingMS` can only be set on config servers of sharded clusters.

balancerMigrationsThrottlingMs Parameter

MongoDB 7.0 adds the `balancerMigrationsThrottlingMs` parameter which allows you to throttle the balancing rate.

gEnableDetailedConnectionHealthMetricLogLines Parameter

MongoDB 7.0 adds the `gEnableDetailedConnectionHealthMetricLogLines` parameter which lets you specify whether or not a set of log messages related to cluster connection health metrics appears in the log.

`oidcIdentityProviders` Parameter

MongoDB 7.0 adds the `oidcIdentityProviders` parameter which allows you to specify identity provider (IDP) configurations when using OpenID Connect authentication.

configureQueryAnalysis-related Parameters

MongoDB 7.0 adds the following parameters related to the `configureQueryAnalysis` command:

- `queryAnalysisSamplerConfigurationRefreshSecs`
- `queryAnalysisWriterIntervalSecs`

- `queryAnalysisWriterMaxMemoryUsageBytes`
- `queryAnalysisWriterMaxBatchSize`
- `queryAnalysisSampleExpirationSecs`

Security

Queryable Encryption General Availability

Starting in MongoDB 7.0, Queryable Encryption with equality queries is generally available (GA). Improvements in the GA make it incompatible with the Queryable Encryption Public Preview, which should not be used now that the feature is GA. For details, see Compatibility Changes in MongoDB 7.0.

KMIP 1.0 and 1.1 Support

MongoDB 7.0 (and 6.0.6) adds the `useLegacyProtocol` setting. This setting allows MongoDB servers to connect to KMIP servers that use KMIP protocol version 1.0 or 1.1.

OpenSSL and FIPS Support

Starting in MongoDB 7.0 and 6.0.7, MongoDB supports OpenSSL 3.0 and the OpenSSL FIPS provider with these operating systems:

- Red Hat Enterprise Linux 9
- Amazon Linux 2023
- Ubuntu Linux 22.04

For details, see TLS/SSL (Transport Encryption).

OpenID Connect

Starting in 7.0, MongoDB Enterprise provides support for OpenID Connect authentication. OpenID Connect is an authentication layer built on top of OAuth2. You can use OpenID Connect to configure single sign-on between your MongoDB database and a third-party identity provider.

Aggregation

New operators:

Name	Description
`$median`	Returns an approximation of the median, the 50th percentile, as a scalar value. This operator can be used as an accumulator and as an aggregation expression.
`$percentile`	Returns an array of scalar values that correspond to specified percentile values. This operator can be used as an accumulator and as an aggregation expression.

Month Name Specifier for $dateToString

MongoDB 7.0 adds the following format specifiers to use with the `$dateToString` operator:

Specifiers	Description	Possible Values
`%b`	Abbreviated month name (3 letters)	`jan-dec`
`%B`	Full month name	`january-december`

Time Series

MongoDB 7.0 removes most of the time series limitations from these operations that are based on the `delete` command:

- `delete`
- `deleteOne()`
- `deleteMany()`
- `Bulk.find.delete()`
- `Bulk.find.deleteOne()`

Sharding

Sharding Metadata Diagnostics

Starting in MongoDB 7.0, the `checkMetadataConsistency` command is available to check sharding metadata at the cluster, database, and collection levels for inconsistencies. These inconsistencies can originate in cases such as:

- Upgrades in cases where the cluster encountered a bug while running previous releases of MongoDB
- Manual interventions that corrupt the cluster catalog

The following helper methods are now available through `mongosh`:

- `db.checkMetadataConsistency()`
- `db.collection.checkMetadataConsistency()`
- `sh.checkMetadataConsistency()`

For more information on the inconsistencies the command checks for, see Inconsistency Types.

mergeAllChunksOnShard Command

Starting in MongoDB 7.0, the `mergeAllChunksOnShard` command finds and merges all mergeable chunks that a shard owns for a given collection.

The AutoMerger

Starting in MongoDB 7.0, the AutoMerger can automatically merge chunks that meet the mergeability requirements. The AutoMerger is enabled by default.

Starting in MongoDB 7.0, you can use the following methods to control the AutoMerger behavior:

- `sh.startAutoMerger()`
- `sh.stopAutoMerger()`
- `sh.enableAutoMerger()`
- `sh.disableAutoMerger()`

enableAutoMerger Parameter for configureCollectionBalancing

Starting in MongoDB 7.0, the `configureCollectionBalancing` command accepts the `enableAutoMerger` parameter. Use `enableAutoMerger` to set whether or not the AutoMerger takes this collection into account.

rangeDeleterHighPriority Parameter for Deprioritizing Range deletions

Starting in MongoDB 7.0, you can prioritize or deprioritize cleanup of orphaned documents over user operations using the `rangeDeleterHighPriority` parameter.

operationsBlockedByRefresh Metrics Removed

MongoDB 7.0 removes the `operationsBlockedByRefresh` document that contains statistics about operations blocked by catalog cache refresh activity because the `operationsBlockedByRefresh` counters increased on `mongos` for every operation that used collection routing information even if the operation wasn't blocked by a catalog refresh activity.

analyzeShardKey Command and db.collection.analyzeShardKey() Method

MongoDB 7.0 adds the `analyzeShardKey` command and the `db.collection.analyzeShardKey()` method, which let you calculate metrics for evaluating a shard key.

configureQueryAnalyzer Command and db.collection.configureQueryAnalyzer() Method

MongoDB 7.0 adds the `configureQueryAnalyzer` command, which allows you to configure query sampling for a collection. MongoDB 7.0 also adds the `db.collection.configureQueryAnalyzer()`, which wraps the `configureQueryAnalyzer` command. Sampled queries provide information to `analyzeShardKey` to calculate metrics about read and write distribution of a shard key.

Changes Introduced in 6.X-Series Rapid Releases

MongoDB 7.0 includes changes and features from the following Rapid Release versions:

- MongoDB 6.1 Release Notes
- MongoDB 6.2 Release Notes
- MongoDB 6.3 Release Notes

Platform Support

Removed Platforms

MongoDB 7.0 removes support for **RHEL 7 / CentOS 7 / Oracle 7** on the PPC64LE and s390x architectures.

Upgrade Procedures

IMPORTANT

Feature Compatibility Version

To upgrade to MongoDB 7.0 from a 6.0 deployment, the 6.0 deployment must have `featureCompatibilityVersion` set to `6.0`. To check the version:

```
db.adminCommand( { getParameter: 1, featureCompatibilityVersion: 1 } )
```

To upgrade to MongoDB 7.0, refer to the upgrade instructions specific to your MongoDB deployment:

- Upgrade a Standalone to 7.0
- Upgrade a Replica Set to 7.0
- Upgrade a Sharded Cluster to 7.0

If you need guidance on upgrading to 7.0, MongoDB professional services offer major version upgrade support to help ensure a smooth transition without interruption to your MongoDB application. To learn more, see MongoDB Consulting.

Downgrade Considerations

Only Single-Version Downgrades are Supported

MongoDB only supports single-version downgrades. You cannot downgrade to a release that is multiple versions behind your current release.

For example, you may downgrade a 7.0-series to a 6.0-series deployment. However, further downgrading that 6.0-series deployment to a 5.0-series deployment is not supported.

Downgrade Policy Changes

Starting in MongoDB 7.0:

- Binary downgrades are no longer supported for MongoDB Community Edition.
- You cannot downgrade your deployment's fCV to or from a rapid release version of MongoDB.
- The `setFeatureCompatibilityVersion` command requires an additional parameter, `confirm`, which must be set to `true` to upgrade or downgrade fCV.
- If you upgrade or downgrade your deployment's fCV, you cannot downgrade your Enterprise deployment's binary version without assistance from support.

Backward-Incompatible Features

MongoDB 7.0 includes features that are not compatible with earlier releases. Downgrading from 7.0 to an earlier release requires that you remove data that uses these features.

For more information, see Backward-Incompatible Features.