Alerting & Metrics

The following sections review some selected metrics available for monitoring and alerting in Atlas. These metrics are described in detail along with their typical behaviour and any recommendations for alerting where applicable.

This is intended only as an introduction and is not an exhaustive list.

1 Opcounters

Lower typically means less load. Number of queries, updates, inserts, deletes, getmore and (other) commands per second. Gives a basic overview of what the system is doing.

Alert Threshold

An alert should be configured for each of the first four (queries, updates, inserts, deletes). Measure or estimate the amount of these occurring at peak load for each metric. Set an alert for around double the expected peaks.

Action

Excessive Opcounter levels typically indicate that load has increased unexpectedly in some dimension. Any increasing usage of the application also causes an increase to Opcounters. Opcounters alone are not a cause for concern, though any unexpected increase in activity should be investigated. The alert levels may need to be revised if an application change was expected.

2 Tickets

Higher is better. Tickets represent the number of concurrent operations allowed into the storage engine. There are separate tickets for read and write operations. When this value reaches zero new requests may queue until a ticket becomes available.

Unless the load is known to be highly concurrent, this value should remain at the default (128 at the time of writing) and never be observed to dip much, or frequently. During sudden load events, it may dip a little, though it should recover quickly (typically within seconds).

Alert Threshold

Tickets available dropping for a sustained period of time is a symptom of problems elsewhere. However, watching tickets available is important since it indicates further research is required. Different workloads will utilize tickets under pressure at different rates so it's difficult to predict exactly which thresholds are actionable. Generally, however, thresholds can include the following:

- < 100 tickets available for more than 2 minutes
- < 50 tickets available for more than 1 minute

- < 10 tickets available at any time

Action

Typically this alert indicates that operations within the database are being rate limited due to load beyond hardware capabilities. Look for causes or symptoms of a significant load increase. If this is the only alert or symptom, raise a MongoDB support request immediately.

3 Replication Lag

Lower is better. Replication lag is calculated between the last operation time on the primary and the time of the last operation received by the secondary. Excessive replication lag diminishes the worth of failover nodes and adversely affects the latency for write operations using a majority write concern or any value greater than one.

See replication-lag[1] in the MongoDB documentation for more.

Alert Threshold

Replication lag should be forgiven for short spikes caused by sudden load after long periods of quiescence. The height of such spikes is limited only by the length of quiescence, while the length of such spikes should be 1 second or less.

A replication lag alarm should be configured to trigger if lag exceeds more than a few tens of seconds persistently for more than a few minutes.

Action

Check the query load of the member that is lagged (Opcounters). Check for any network connectivity issues between members. Check the hard-drive health of the lagged member.

If there are no apparent load-related causes for the lag, raise a support request with MongoDB.

4 Replication Oplog Window

Higher is better. This is the amount of time in write operations that the oplog covers, this window represents how long a Secondary can be down before losing its ability to catch up to the Primary.

Alert Threshold

A minimum value of 72 hours for this metric is recommended. Below 24 hours is risky.

[1] https://docs.opsmanager.mongodb.com/current/reference/alerts/replication-lag/

Action

A large batch import of data, restoration, or onboarding, can be a good reason why the replication oplog window may drop dramatically, though temporarily. If a large data import has occurred recently, it may be prudent to wait until a period of normal load has occurred to re-assess if the oplog is adequate.

If the alert has no specific cause, or if it is known that write load has increased such that the oplog is no longer adequate to cover 72 hours typical write load, then it may be necessary to resize the oplog. The oplog can be resized by editing the deployment in Atlas[2] and changing the oplog size[3]. The change requires no downtime, though for older versions of MongoDB (**<3.4.9**) it does cause an election, thus it should be done at a quiet time to avoid disruption. Use the headroom graph to predict the new size needed, increase the size of the oplog by a proportional amount needed to push the graph upwards to achieve 72 hours headroom.

5 Connections

Lower is better. This represents the number of connected sockets the server is actively servicing. This includes sockets used for heartbeats and replication between members. It should never be zero.

Alert Threshold

An alert should be set to a little above double the peak observed. Thus, if 180 connections are observed during peak load, set an alert for 400.

Transient network glitches, or application restarts, can cause erroneous transient reports of around double the normal connection load. These should be ignored for short periods, thus the recommendation for alerts only beyond double the observed quantity.

Action

If the connection count has increased over a long period of time due to slowly increasing workload then it may need to re-evaluate the alert threshold.

If connections have risen suddenly then the cause can be pinpointed in time on the graph to the nearest 10 seconds. Investigate what may have changed recently.

If connections have risen in response to other effects, such as slow query response, then the number of connections may be an application response to slow queries. Such a response can trigger a connection storm which can amplify a small issue into a larger one. Application connection limits may need to be reviewed. A connection limit can be specified in the connection string[4].

[2]
https://docs.atlas.mongodb.com/scale-cluster/#scale-cluster-more-configuration-options[3]
https://docs.atlas.mongodb.com/cluster-config/additional-options/#set-oplog-size [4]
https://docs.mongodb.com/manual/reference/connection-string/#urioption.maxPoolSize

6 Query Targeting

Lower is better. The ratio of number of keys and documents scanned to the number of documents returned, averaged by second. A high level of this metric means that the query subsystem has to perform many inefficient scans in order to fulfill queries. This situation typically arises when there are no appropriate indexes to support queries.

Well performing applications often achieve less than 2. It is possible, but extremely unlikely, to achieve less than 1. Achieving less than 2 does not assure good performance, though much higher numbers do usually indicate bad performance. If this ratio is high the issue is typically caused by a missing index or a design flaw of the queries being run.

Alert Threshold

Expected application behavior needs to be understood before an alert can be set, however, anything above 10 typically indicates poor behavior.

Action

Use the Performance Advisor[5]to observe one or more specific operations that are noticeably contributing to the alert.

The operations that are scanning lots of documents usually come about due to a change in behavior from the application. Either a new query has been introduced, or enough data has been added to the system that a performance threshold has become noticeable. In either case, an efficiency review is needed of the operations the application carries out.

Any sudden or dramatic changes are usually due to application changes that can be rolled back until the behavior can be investigated. If a sudden change is observed without apparent cause, raise a MongoDB support request.

7 Document Metrics

Lower is better. The numbers represent the amount of data being transferred between application and the database, so while these are rarely able to be reduced, they can still be used to flag unexpected changes in the amount of data being handled on the network.

Alert Threshold

None specifically recommended.

Action

If the alert is triggered after an upgrade it may need to be re-evaluated. Consider if the increase in data being transferred is expected before doing so.

[5] https://docs.atlas.mongodb.com/performance-advisor/

Otherwise, consider surrounding factors to why data transfer rates might have increased. Note that this metric represents load purely generated by the application. There is nothing to address at the MongoDB server. The application team must be engaged to be able to address the condition.

8 System CPU (user %)

Lower is better. This is the host operating system measure of CPU utilization.

CPU usage is highly use case dependent. Applications that do only basic CRUD operations rarely use much CPU. Applications that do a lot of aggregations, map-reduce, or in-memory sorting, can utilize much more CPU or even become performance bound by maximum CPU utilization.

Alert Threshold

None specifically recommended unless CPU usage is observed to be important. If CPU alerting is desired, then ensure the level is sustained for several minutes before an alert is generated as short CPU spikes are often a part of normal behavior.

Action

If CPU changes dramatically, it will be due to a change in load patterns, or more likely a change in the application that uses new or revised operations. An application change can be rolled back to review the changes made. A change in load patterns may indicate that either a larger CPU will be needed to handle the extra load, or an efficiency review of the operations that are carried out be undertaken.

9 Disk space used (%)

Lower is better. This is the measure of used disk space for MongoDB related files. A good practice is to set an alert on this at a threshold that would indicate the system is running out of free space.

Alert Threshold

None specifically recommended. Atlas can be set to automatically expand the disk storage to mostly cover normal expansion.

Action

Check if there is any reason why more space is needed in the database. An application change may mean that more data is being stored. Check what the usage is via admin data viewing tools such as MongoDB Compass[6] or the Data Explorer in Atlas[7].

[6] https://www.mongodb.com/products/compass
[7] https://docs.atlas.mongodb.com/data-explorer/
Use the graphs to predict how long it will be before disk space is consumed entirely.

If there is no apparent application related reason for the data size to be increasing, raise a support request with MongoDB.

10 Disk I/O utilization (%)

Lower is better. This is the host operating system measure of disk IO utilization. Most workloads do not expect this value to ever stay at 100%, though it is common for spikes to reach 100% occasionally.

High disk utilization has a few common causes:

- Memory exhaustion
- High write load
- Unusual load churn

The disk is required to write new data. A write operation cannot complete until the disk confirms receipt of the data. This means that a large data import or other batch job will generally be limited by the speed of the disk.

Additionally, the disk is the source of data when memory cache currently does not hold the answer to a query or aggregation. This can be because the query is novel and thus unlikely to be in the cache, or representative query load exceeds the capacity of system memory.

Both reading and writing cause disk IO to increase.

Alert Threshold

Sustained utilization for several minutes of over 80% is typically bad for most workloads.

Action

If a large data import, batch job, or other significant write load event is occurring the alert can be quiesced for re-assessment once the unusual event is complete.

Check the Opcounters and Query Targeting graphs to get some insight into what sort of unexpected load is occurring. Both of these may provide hints towards the profile change that created the load. Most likely, either the application has changed, or load has increased

proportionally and the disk has become a bottleneck.