

# Stanbic IBTC Bank Plc

## MongoDB Engagement Summary

Dharmendra Kumar <dharmendra.kumar@mongodb.com>, MongoDB Inc.

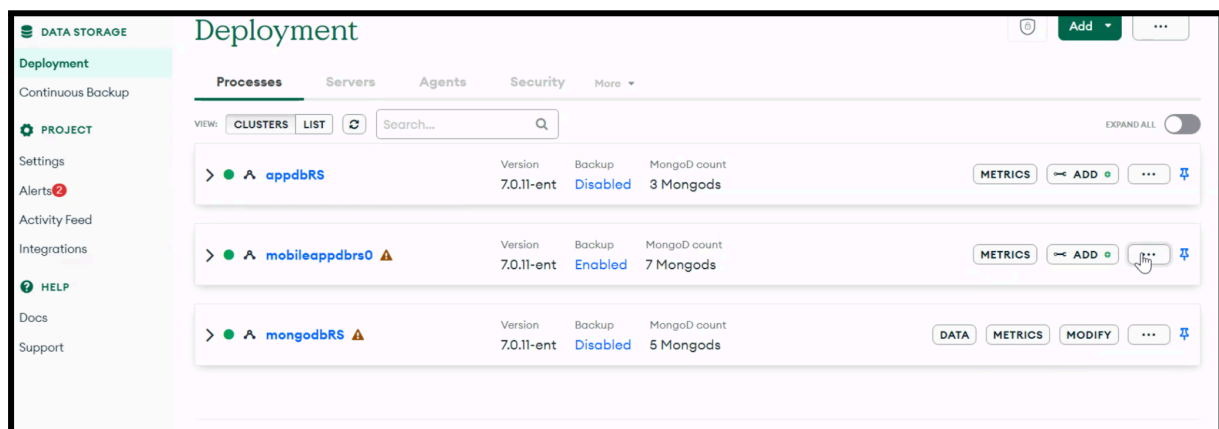
November 2024

### Participants:

- Fazazi Abanikanda, Stanbic IBTC Bank Plc
- Nnamdi Eze, Stanbic IBTC Bank Plc
- MacGodwin Osuji, Stanbic IBTC Bank Plc
- Dharmendra Kumar, Sr Consulting Engineer, MongoDB Inc.

### Environment:

The team is currently utilizing Ops Manager and a cluster that supports mobile banking applications, specifically focusing on transaction activities, The following are the clusters managed by the Ops Manager:



## A. Goals of the Engagement:

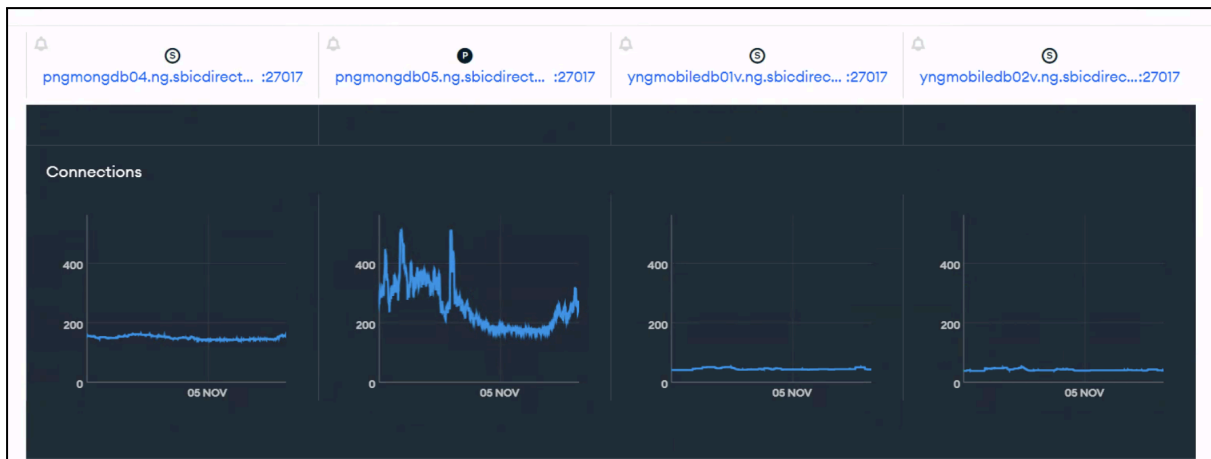
This engagement was aimed at assisting the "Stanbic IBTC Bank Plc" team and the team would like to address two key points:

1. Review and discuss optimization strategies, including cluster metrics analysis to identify performance bottlenecks.
2. Enhance the security of Ops Manager by enabling TLS for secure communication.

## B. Topics discussed during the engagement

### 1. Tuning Your Connection Pool Settings

As discussed, creating new authenticated connections to the database is expensive. Instead of creating and destroying connections for each request to the database, you want to reuse existing connections as much as possible. This is where connection pooling comes in.



A Connection Pool is a cache of database connections maintained by your driver so that connections can be reused when new requests to the database are required. When properly used, connection pools allow you to minimise the frequency and number of new connections to your database.

Your application can seamlessly get connections from the pool, perform operations, and return connections back to the pool. Connection pools are thread-safe.

**Note:** When connections are not configured optimally, your application can open and close new database connections too often, resulting in “connection churn”. In a high-throughput application, this can result in a constant flood of new connection requests to your database adversely affecting the performance of your database and your application.

## 1.1 Create and Use a Connection Pool

Use one MongoClient instance per application unless the application is connecting to many separate clusters. Each MongoClient instance manages its own connection pool to the MongoDB cluster or node specified when the MongoClient is created. MongoClient objects are thread-safe in most drivers.

Following are a few optimizations that the team can do to optimize the connection pool settings:

Problem	Solution
Slow application-side operation times that are not reflected in the database server logs or the real-time panel.	<p>Use <a href="#">connectTimeoutMS</a> to ensure the driver does not wait indefinitely during the connection phase.</p> <p>Set <b>connectTimeoutMS</b> to a value greater than the longest network latency you have to a member of the set.</p> <p>For example: if a member has a latency of 10000 milliseconds, setting</p>

<p>A misconfigured firewall closes a socket connection incorrectly and the driver cannot detect that the connection closed improperly.</p>	<p><b>connectTimeoutMS</b> to 5000 (milliseconds) prevents the driver from connecting to that member.</p> <p>Use <a href="#">socketTimeoutMS</a> to ensure that sockets are always closed.</p> <p>Set <b>socketTimeoutMS</b> to two or three times the length of the slowest operation that the driver runs.</p>
<p>The server logs or real-time panel show that the application spends too much time creating new connections.</p>	<p>Not enough connections are available at startup. Allocate connections in the pool by setting <a href="#">minPoolSize</a>.</p> <p>Set minPoolSize to the number of connections you want to be available at startup.</p> <p>The MongoClient instance ensures that number of connections exists at all times.</p>
<p>The load on the database is low and there's a small number of active connections at any time. The application performs fewer operations at any one time than expected.</p>	<p>Increase <a href="#">maxPoolSize</a>, or increase the number of active threads in your application or the framework you are using.</p>
<p>Database CPU usage is higher than expected.</p>	<p>Decrease the <a href="#">maxPoolSize</a> or reduce the number of threads in your application. This can reduce load and response times.</p>

Refer: [Calculate Maximum Number of Connections](#)

## 2. Scan and Order: Queries performing an in-memory sort

In MongoDB, sort operations can obtain the sort order by retrieving documents based on the ordering in an index. If the query planner cannot obtain the sort order from an index, it will sort the results in memory. Sort operations that use an index often have better performance than those that do not use an index. In addition, sort operations that do not use an index will abort when they use 100 megabytes for version 4.4 onwards and *to avoid this abortion of query team is using `allowDisk:true` which leads to huge performance degradation, thus it is highly recommended to leverage the compound index to avoid in-memory sort.*

### 2.1 ESR Rule

We quickly discussed the ESR rule for creating the index, As ESR is the thumb rule that should be kept in mind which is also recommended for performance best practices.

### 2.1.1 (E) Equality First

When creating queries that ensure selectivity, we learn that “selectivity” is the ability of a query to narrow results using the index. Effective indexes are more selective and allow MongoDB to use the index for a larger portion of the work associated with fulfilling the query.

Equality fields should always form the prefix for the index to ensure selectivity.

### 2.1.2 (E → S) Equality before Sort

Placing Sort predicates after sequential Equality keys allow for the index to:

- Provide a non-blocking sort.
- Minimize the amount of scanning required.

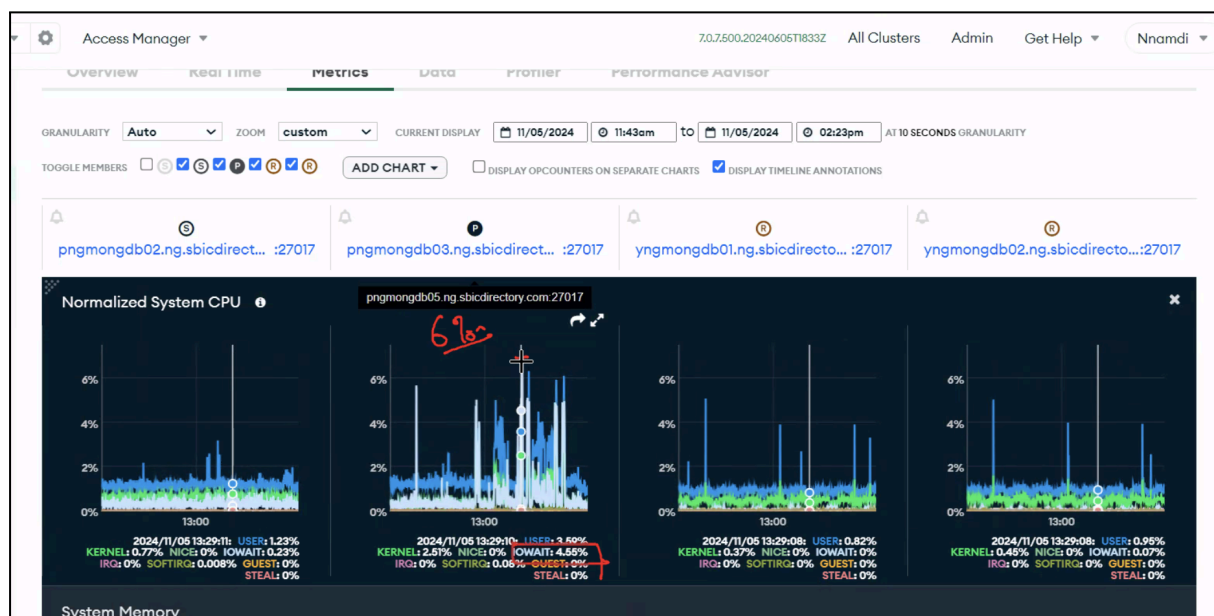
### 2.1.3 (S → R) Sort before Range

Having a Range predicate before the Sort can result in a Blocking (In Memory) Sort being performed as the index cannot be used to satisfy the sort criteria.

Refer to the [Documentation](#) for more details.

## 3. Identifying bottlenecks in the IO

From the performance metrics, a good indication that there is an IO bottleneck is if you see a plateau in the disk IOPS performance metrics graphs. This, coupled with increased **disk queue depth** and high IO Wait (sometimes reaching to ~6% and can be seen in the following graph), all indicate that the disks are saturated with an IO workload that is exceeding its limits.



A high **Disk Queue Depth** value can indicate that the storage system is struggling to keep up with the workload, which may lead to performance issues. However, it's important to note

that what constitutes a "high" value depends on various factors such as your specific workload, hardware configuration, and performance expectations.



Ideally, the OS is responding to disk requests by the mongod process immediately. Disk queuing happens when read or write requests from the mongod process to the OS accumulate and are queued by the disk scheduler. This situation is in itself problematic, specially if it occurs during peaks in the workload or in short bursts, but if it is persistent it might indicate that there is a need to optimize queries or more IOPS are needed.



In general, if you consistently observe a Disk Queue Depth value it could indicate a potential problem. This suggests that there are more pending I/O operations than can be efficiently handled by your storage system.

The IOPS threshold will depend on your current IOPS allocation provisioned for the tier and storage size of the cluster. But allocated IOPS is not a hard cap, with cloud providers allowing some burst capacity, which is why you may regularly see IOPS above the allocated cap. Temporary spikes are normal, but prolonged high IOPS may be concerning.

To address this issue, consider taking the following steps:

1. Monitor other relevant metrics: Check other metrics such as disk latency, throughput, and CPU utilization to get a better understanding of your system's overall performance.
2. Optimize queries: Poorly optimized queries can contribute to increased disk usage and higher Disk Queue Depth values. Review your query patterns and ensure they are properly indexed and optimized for efficient execution.
3. Scale up or out: If your workload consistently generates high Disk Queue Depth values even after optimizing queries, consider scaling up your hardware resources (e.g., increasing CPU or memory) or scaling out by adding more nodes to distribute the load.
4. Evaluate storage configuration: Review your storage configuration settings such as IOPS limits and provisioned capacity in MongoDB Atlas. Adjusting these settings based on your workload requirements can help improve disk performance.

As discussed in general, there are mainly two types of Disk IOPS which are - Disk read IOPS and Disk Write IOPS.

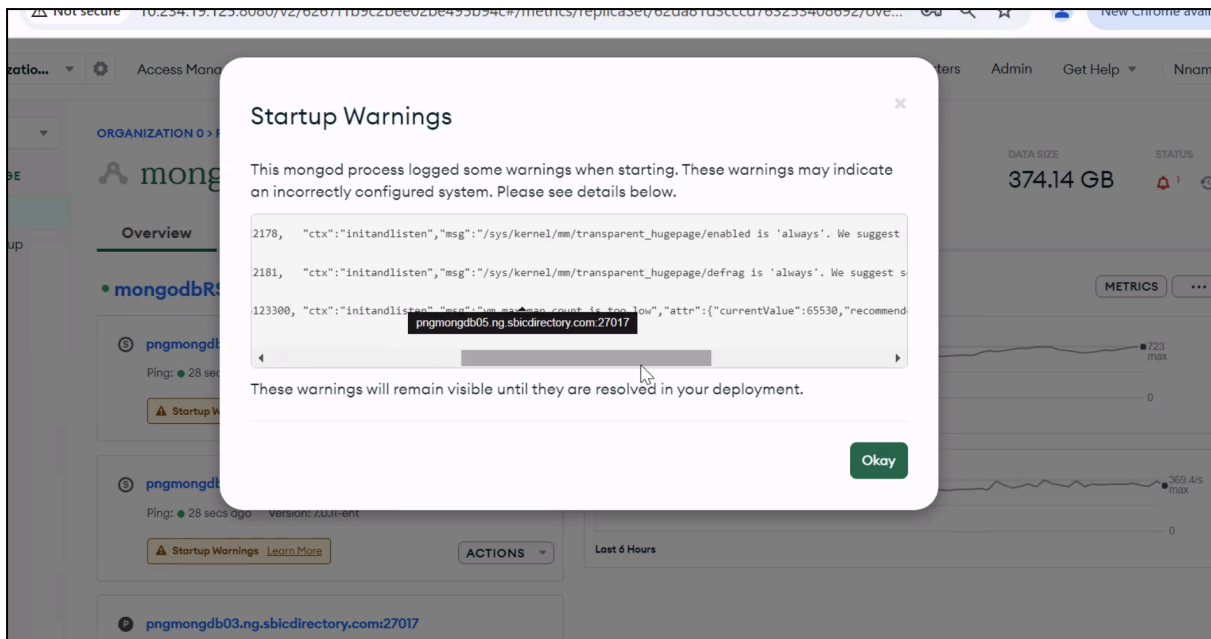
**Read IOPS:** the number of read requests per second for the disk partition running MongoDB.

**Write IOPS:** the number of write requests per second for the disk partition running MongoDB.

Based on the cluster tier and provisioned amount of storage, the amount of available Disk IOPS varies. Also, based on the Cloud Provider there are some extra details to consider.

## 4. Fix Startup Warning

The team is receiving the following startup warnings:



Refer: [transparent-huge-pages](#)

```
JavaScript
[Unit]
Description=Disable Transparent Huge Pages (THP)
DefaultDependencies=no
After=sysinit.target local-fs.target
Before=mongod.service

[Service]
Type=oneshot
ExecStart=/bin/sh -c 'echo never | tee
/sys/kernel/mm/transparent_hugepage/enabled > /dev/null'

[Install]
WantedBy=basic.target
```

Post that team can reload the system files

```
JavaScript
sudo systemctl daemon-reload
sudo systemctl start disable-transparent-huge-pages
sudo systemctl enable disable-transparent-huge-pages

cat /sys/kernel/mm/redhat_transparent_hugepage/enabled
```

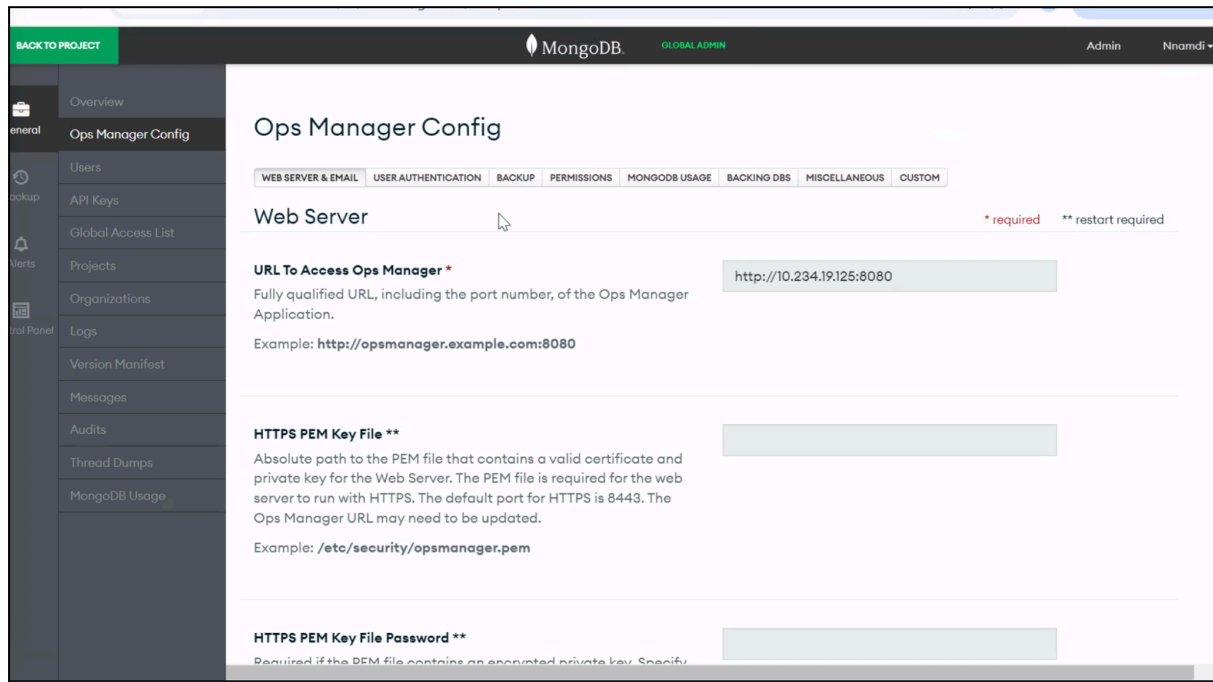
## 5. Configure Ops Manager Application for TLS

The team want to encrypt connections for all MongoDB Agents, website clients accessing the Ops Manager application, and API clients using the REST API.

To enable encryption in Ops Manager you can either:

1. Configure an HTTPS proxy in front of Ops Manager, or
2. Run the Ops Manager application over HTTPS, as detailed on this page.

The following procedure configures Ops Manager with a .pem file that contains the Ops Manager host's TLS certificate.



The screenshot shows the MongoDB Ops Manager configuration interface. The left sidebar contains navigation links: Overview, Ops Manager Config, Users, API Keys, Global Access List, Projects, Organizations, Logs, Version Manifest, Messages, Audits, Thread Dumps, and MongoDB Usage. The main content area is titled 'Ops Manager Config' and has a tabbed interface with 'WEB SERVER & EMAIL' selected. Below the tabs, the 'Web Server' section is active, showing fields for 'URL To Access Ops Manager \*' (with value 'http://10.234.19.125:8080') and 'HTTPS PEM Key File \*\*' (with value '/etc/security/opsmanager.pem'). There are also fields for 'HTTPS PEM Key File Password \*\*'. The interface includes status indicators like '\* required' and '\*\* restart required'.

1. Upload the certificate file to each Ops Manager host.
2. Enable TLS for the Ops Manager Application.
3. Restart each Ops Manager host to enable TLS.
4. Configure MongoDB Agents to use TLS

Refer:

1. [Configure Ops Manager Application for TLS](#)
2. [Configure MongoDB Agents to use TLS](#)

## C Recommended Follow-up Consulting

### 1. Consulting

During the engagement, we reviewed optimization areas using Ops Manager and identified potential improvements. Additionally, we discussed steps to enhance the security of Ops Manager. If further assistance is required, the team can schedule additional consulting days.



## 2. Further training and education in MongoDB

MongoDB offers a comprehensive set of instructor-led training courses covering all aspects of building and running applications with MongoDB. Instructor-led training is the fastest and best way to learn MongoDB in depth. Both public and private training classes are available - for more information or to enroll in classes, please see - [Instructor Led Training](#).