Technical Report

Architecting I/O-Intensive MongoDB Databases on NetApp

Rodrigo Nascimento, NetApp
August 2019 | TR-4788

Abstract

MongoDB is a NoSQL database that is used to support applications needed for the Internet of Things, mobile devices, real-time analytics, and other purposes. MongoDB has become a general-purpose database for web-scale applications. To guarantee the stability, reliability, availability, and performance of data for your business, it's important to understand how to use NetApp® technology to architect a MongoDB cluster.

NetApp®

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 Introduction

This technical report is intended for application owners and developers deploying MongoDB in corporate data centers, in the cloud, and in hybrid cloud environments. It describes how to provision storage subsystems according to NetApp recommendations.

For storage administrators, this report provides an overview of MongoDB and its fundamental concepts to help you work with MongoDB database administrators (DBAs). For MongoDB DBAs, it provides an overview of NetApp storage systems and fundamental concepts that you should be aware of when working with NetApp storage and infrastructure.

The report also discusses database workload characteristics and how to build a database I/O profile from the information gathered during the workload characterization process. Finally, it defines I/O-intensive databases and describes some scenarios for architecting a MongoDB database by using NetApp technology.

2 MongoDB Editions

There are two editions of MongoDB: the Community Edition and the Enterprise Advanced Edition. The recommendations in this document apply to both editions.

The best practices guide from MongoDB discusses MongoDB running on commodity hardware. However, commodity hardware does not offer the same reliability as enterprise-grade arrays. Therefore, the recommendations in this document might differ from those made by MongoDB, based on the high availability and reliability of NetApp systems.

2.1 MongoDB Community Edition

MongoDB Community is the open-source edition of the database. It includes compression with the WiredTiger (WT) storage engine, SCRAM authentication, x.509 certificate authentication, and role-based access control.

2.2 MongoDB Enterprise Advanced Edition

MongoDB Enterprise Advanced is the commercial edition of the database. It includes all the features available in the Community Edition plus the following:

- A management platform (MongoDB Ops Manager)
- Encryption at rest
- An in-memory storage engine
- A BI connector
- Auditing
- Kerberos authentication
- LDAP proxy authentication
- Support 24/7/365 with a 1-hour SLA
- Emergency patches
- Private on-demand training

3 MongoDB Deployment Options

A MongoDB database can be deployed using three different approaches: standalone, a replica set, or a sharded cluster.

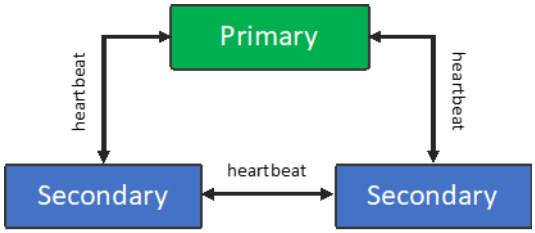
NetApp does not recommend the standalone option for production environments. Rather, you should deploy MongoDB using a replica set or a sharded cluster, as described in the following sections.

3.1 Replica Set

A replica set is a group of three or more mongod processes that are responsible for the same dataset. The main function of a replica set is to provide data availability. In a replica set, each member has one of three possible roles: primary, secondary, or arbiter.

A primary is the member that handles reads and writes to one or more databases in a replica set, and a secondary is a replica of the dataset. An arbiter does not hold a copy of the dataset; it is used to help the cluster elect a new primary, if necessary. Figure 1 shows a three-member primary-secondary-secondary replica set.

Figure 1) Three-member replica set.



Note: By default, the primary is the only member that can read and write to databases. However, MongoDB does allow you to read from secondaries.

The replication between members of a replica set is asynchronous. Each member of the replica set has a replication source, except for the primary. MongoDB supports chainable replication, which means that a secondary can connect to and pull data from either the primary or another secondary.

Members are kept up to date through their own copy of the operation log (oplog). The oplog is a capped collection that is very similar to a circular buffer. As soon as the oplog fills up, it makes room for new documents by overwriting the oldest documents in the collection.

Data availability is guaranteed by the automatic failover feature. In case of any failure that affects the primary, an election is triggered by the secondaries, and a new primary is chosen. Failures can include a server hardware failure, a mongod process crash, or a network partition that isolates the primary from the majority of the members.

Best Practice
For the purpose of high availability, you should never have members of the same replica set running on the same server. In virtual or containerized environments, make sure that virtual machine or container members of the same replica set are running on different physical servers.

A replica set can have up to 50 members, but only a maximum of 7 can be voting members.

3.2 Sharded Cluster

Sharding is the technique chosen by MongoDB to scale out a cluster by creating data partitions based on a shard key. When an application starts demanding more than a single replica set is capable of handling, sharding allows a MongoDB cluster to continue serving more load. Each shard is a replica set that manages a partition as its own database.

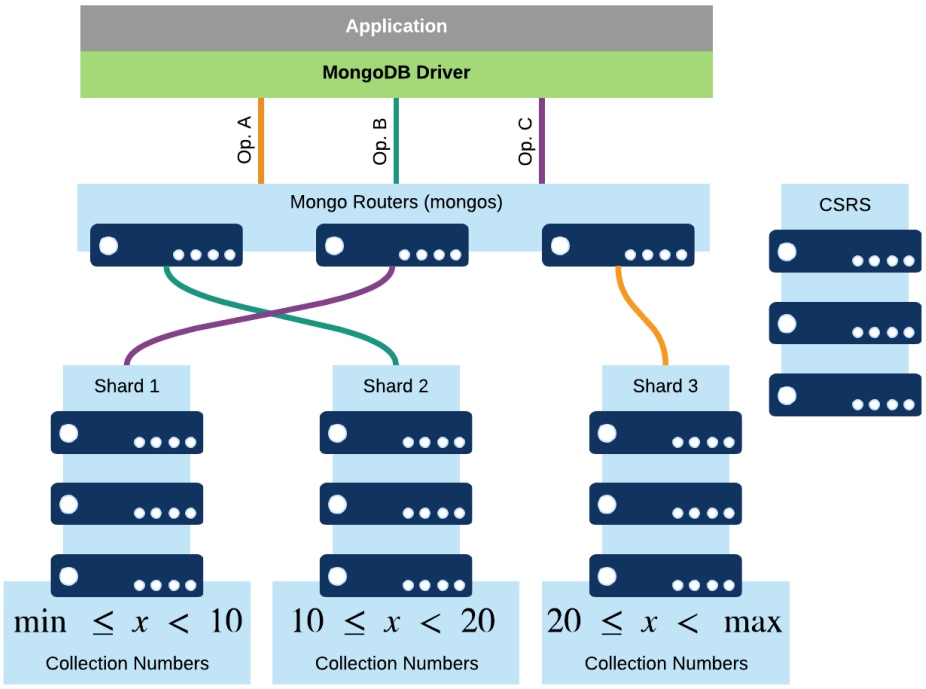
Documents should be balanced across all shards. MongoDB automatically balances documents by using the shard key to partition the data of a given collection. Each sharded collection can have only one shard key. A shard key consists of one or more immutable fields[1] with the following properties: cardinality, frequency, and rate of change.

A good shard key usually presents high cardinality, low frequency, and a nonincremental rate of change. The cardinality determines the maximum number of shards in your cluster. Frequency and rate of change dictate how balanced your documents are across shards. After you enable sharding in a collection, there is no official MongoDB command to reverse sharding.

There are three sharding methods: hashed shard, ranged shard, and zones. Figure 2 shows the collection Numbers sharded by using the field `x` as the shard key.

A ranged, sharded collection behaves in the following way. Your application connects to the sharded cluster by using a MongoDB driver, which opens one or several connections to the mongo routers (mongos). A user issues a query (Op. A) requesting `x: 23` from the collection Numbers. At the same time, two other users also issue queries. (Op. B) requests `x: 13`, and (Op. C) requests `x: 7`. In Figure 2, note that all queries are processed by different servers. This configuration is a reliable and scalable architecture for modern web-scale applications.

Figure 2) Sharded cluster.



3.3 Storage Engine: WiredTiger

Starting with MongoDB 3.2, WiredTiger (WT) is the default storage engine. WT uses a multiversion concurrency control (MVCC) model, which means that it creates a point-in-time snapshot of the data in memory for each operation. To persist data, WT writes the in-memory snapshots to disk every 60 seconds. This process is known as a checkpoint.

[1] In MongoDB 4.2, a shard key can be composed of mutable fields.

WT can also work with journaling enabled, which provides better durability. A write-ahead log is used to capture data between checkpoints. With journaling enabled, a checkpoint is created every 60 seconds or until the journal data reaches 2GB, whichever happens first.

4 NetApp Platforms

NetApp has a broad portfolio of storage solutions. This guide describes best practices for NetApp ONTAP® systems.

In some cases, different systems might have different best practices. Differences are labeled with the following icons.

Icons and Systems		
		
AFF/FAS	ONTAP Select	Cloud Volumes ONTAP

NetApp AFF and FAS systems address enterprise storage requirements with high performance, superior flexibility, and best-in-class data management.

NetApp ONTAP Select offers robust enterprise storage services that are deployed on the customer's choice of commodity hardware. It combines the best of the cloud, in terms of agility and granular capacity scaling, with the flexibility, resilience, and locality of on-premises storage.

NetApp Cloud Volumes ONTAP combines data control with enterprise-class storage features for various use cases. Uses include file shares and block-level storage serving NAS and SAN protocols (NFS, SMB/CIFS, and iSCSI), disaster recovery, backup and archive, DevOps, databases, and all other enterprise workloads.

5 ONTAP Components: Overview

5.1 Storage Virtual Machine

NetApp ONTAP is a clustered storage system composed of physical storage nodes known as HA pairs. An HA pair is a set of two storage nodes. Each storage node has independent CPU, cache, network ports, and disks.

A NAS cluster (using file-based protocols) contains from 1 HA pair up to 12 HA pairs. A SAN cluster (using block-based protocols) contains from 1 HA pair up to 6 HA pairs. A mixed NAS/SAN cluster contains up to 6 HA pairs.

To give access to the underlying resources of a cluster such as disks and network ports, ONTAP provides a layer of abstraction to simplify organization and management of these resources. A storage virtual machine (SVM) provides the abstraction layer that gives you access to all the resources available in an ONTAP cluster.

There are no recommendations for how you should create SVMs. Some customers have only one SVM, while others prefer to create an SVM for each application.

5.2 Aggregates

RAID

MongoDB recommends using RAID 10 for the best reliability and performance. This is a very common recommendation in the database space because RAID 10 consists of mirroring and concatenation not incurred in any parity calculation to protect against disk failures. Other popular RAID types such as 4, 5, and 6 affect write performance because they require parity calculations.

Although MongoDB recommends using RAID 10, with NetApp, you can achieve the best reliability and performance by using the RAID recommended for each specific platform.

For AFF and FAS systems, NetApp recommends using NetApp RAID DP® technology. These systems are equipped with nonvolatile memory cards (NVMEM) that boost write performance. When a client sends writes to the storage system, the operations are received and protected by the NVMEM cards and immediately acknowledge to the client. ONTAP triggers a consistency point (CP) and all data is written from the NVMEM to the disks. RAID parity is calculated as a part of the CP operation, which does not affect application performance during write operations.

ONTAP Select supports a hardware RAID controller, and it also offers built-in software RAID. Its physical disks must be placed into RAID groups protected by RAID 5 or RAID 6 in the case of hardware RAID. RAID DP is recommended when using software RAID. Virtual disks are created in these RAID groups and are used by ONTAP Select to create aggregates. The aggregate RAID type for ONTAP Select is RAID 0 for hardware RAID.

Cloud Volumes ONTAP runs on top of Elastic Block Store (EBS) volumes in AWS and Azure Storage Disks in Azure. In both cases, these technologies provide data reliability for Cloud Volumes ONTAP. The aggregate RAID type for Cloud Volumes ONTAP is RAID 0.

5.3 Logical Interfaces

Logical interfaces are virtual network interfaces created by ONTAP on top of the physical network ports. These interfaces are attached to the SVM to provide the connectivity needed by the initiators to communicate with the physical nodes and to provide access to the data stored in their aggregates. They can be easily migrated across physical ports, and they automatically fail over and recover during a failover event.

5.4 Protocols

Protocols are the languages used for communication between hosts and storage subsystems. There are block-based protocols such as FCP, iSCSI, and NVMe/FC and file-based protocols such as NFS and CIFS. NetApp ONTAP is multiprotocol storage software. It supports all the protocols just mentioned running concurrently in the same system.

Protocols supported: FCP, iSCSI, NVMe/FC, NFS, and CIFS

Protocols supported: iSCSI, NFS, and CIFS

Protocols supported: iSCSI, NFS, and CIFS

You select which protocol to use in your environment at the SVM layer. An SVM can support multiple protocols at the same time. By default, all protocols are allowed.

Block-Based Protocols (SAN)

To access the storage capacity in your host, block-based protocols require the creation of initiator groups (igroups) for FCP, iSCSI, or subsystems for NVMe/FC. Regarding storage devices, MongoDB is a shared-nothing architecture. Therefore, each host should have its own igroup (FCP or iSCSI) or subsystem (NVMe/FC).

File-Based Protocols (NAS)

File-based protocols such as NFS and CIFS require a volume junction path and an export policy to provide access to the storage capacity for your host.

5.5 NetApp FlexVol Volumes and Logical Unit Numbers

Volumes

The *ONTAP Introduction and Concepts Guide* from the ONTAP 9 documentation defines a NetApp FlexVol® volume as a logical container used to store data. A volume lives in an aggregate, but, because volumes are loosely coupled with their containing aggregate, they offer flexibility to manage data. FlexVol volumes can be migrated across nodes in a cluster, mirrored to other clusters with NetApp SnapMirror® technology, expanded or reduced in size, and much more.

FlexVol volumes are also a unit of parallelization in ONTAP. On higher-end platforms, using more volumes allows more efficient consumption of resources on the ONTAP system, which means lower latency and higher throughput.

In a SAN environment, volumes contain logical unit numbers (LUNs); in a NAS environment, they contain a file system. The relationship between LUNs and FlexVol volumes can be 1:1 or N:1. Each LUN is always wholly contained in a single FlexVol volume, but multiple LUNs can be contained in a single FlexVol volume.

When multiple LUNs are contained in a single FlexVol volume, the volume acts as a consistency group. A NetApp Snapshot™ copy of the containing FlexVol volume results in a point-in-time copy that can be used to restore all the contained LUNs uniformly.

ONTAP also offers a consistency group Snapshot copy that can be used to sync a Snapshot copy across multiple FlexVol volumes to offer point-in-time recovery.

LUNs

A LUN is an identifier for a device called a *logical unit* that is addressed by a block-based protocol. A Linux host sees a LUN as a disk device (`/dev/sd*`). LUNs are the basic unit of storage allocation in a SAN environment.

LUNs can be accessed in many ways. They can be accessed as raw devices; they can contain a file system such as XFS or ext4; or they can be bundled together using tools such as LVM2. Sections 6 and 7 describe how to lay out and configure LUNs to create consistent and high-performance access for MongoDB.

Snapshot Copies

A Snapshot copy is a read-only, point-in-time image of a volume. By keeping track of block changes since the last Snapshot copy, ONTAP efficiently saves capacity every time a copy is created. You can have up to 1,023 Snapshot copies per volume.

If something goes wrong and you need to restore data from a Snapshot copy, you can restore the entire volume, a single file (NAS), or a specific LUN (SAN).

6 I/O-Intensive Databases

Architecting a storage solution for a database requires an understanding of the database's workload and its I/O profile. Workload characterization is an important tool in performance analysis, and it's also a useful resource for designing storage solutions to support I/O-intensive applications.

Workload Characterization

To architect a storage solution for a database, consider the following characteristics of its workload:

- Buffered I/O or direct I/O
- Operation block size
- Database cache size
- Database cache miss
- Dataset size
- Working set size
- Number of read and write operations
- Read and write randomness

Some databases provide an excellent level of instrumentation, and you might be able to collect all these characteristics (metrics) directly from the database.

MongoDB does not offer this much detail on I/O operations by default. Therefore, you must collect metrics by using database administrator commands and other operating system commands, such as iostat.

- **Buffered I/O or direct I/O.** Buffered I/O happens when an application opens a file and read and write operations go through the file system cache. Direct I/O occurs when an application opens a file using the `O_DIRECT` flag, and its read and write operations bypass the file system cache.

  Although WT allows you to open files using direct I/O, the Enterprise Advanced Edition of MongoDB does not support direct I/O.
- **Operation block size.** The size of the I/O operation when reading or writing data to the file system or disk device.
- **Database cache size.** The amount of system memory dedicated to the database to be used as cache.
- **Database cache miss.** A database cache miss occurs when a database operation cannot be satisfied by the database cache.
- **Dataset size.** The total amount of data stored by a database. All tables and indexes of the database together make the dataset size.

- **Working set size.** Usually a database accesses only a portion of the total stored data (the dataset). The working set size is the amount of active or accessed data in a given window of time.
- **Number of read and write operations.** The number of read and write operations happening at the storage device layer.

  While manipulating data (for example, using the insert, update, or query command) or defining data (for example, using the create, drop, or alter command), the database triggers I/O operations at the storage device layer.
- **Read and write randomness.** After you have determined the number of read and write operations, you must identify how many of these operations are random operations and how many are sequential.

I/O Profile

Determining the characteristics listed in the previous section helps you to understand the database I/O profile. For example, a database with a high rate of database cache misses might have a working set that is larger than its database cache size. If the database cache cannot be expanded, data blocks are read and written to and from the storage layer to satisfy database operations. This process creates greater disk I/O demand compared to a database for which the working set size fits entirely in memory.

Databases with an I/O-intensive profile might need a different approach for laying out storage devices due to limits imposed at the operating-system layer (for example, the SCSI disk device queue).

The NVMeoF protocol simplifies this step by providing increased parallelism compared to the SCSI protocol. While the industry waits for the NVMe protocol to mature and gain market penetration, it is possible to employ simple optimizations for SCSI-based deployments by combining two or more disks (LUNs) to achieve high performance.

To illustrate the single disk versus multiple disk situation described above, an I/O-intensive workload was simulated on a file system using Flexible I/O (FIO) to create four 100GB files in an XFS file system.

Table 1 shows a sample of iostat data collected when FIO created files on a file system using a single LUN. The output is sorted in descending order by the IOPS column.

Table 1) iostat output for a single LUN.

IOPS	GB/s	Latency (ms)	I/O Size KB	Avg. Queue Size
23,543	1.44	5.83	64	137.28
23,244	1.42	5.98	64	138.67
23,239	1.42	5.83	64	135.42
22,920	1.40	6.09	64	139.68
22,913	1.40	5.88	64	135.04
22,369	1.37	6.30	64	140.42
22,345	1.36	5.82	64	130
22,319	1.36	6.28	64	140.1

Table 2 shows a sample of iostat data collected while FIO created files on a file system using two LUNs. The output is sorted in descending order by the IOPS column.

Table 2) iostat output for two LUNs.

IOPS	GB/s	Latency (ms)	I/O Size KB	Avg. Queue Size
43,444	2.65	4.37	64	189.81
40,244	2.46	4.37	64	170.57
38,836	2.37	4.90	64	190.62

38,789	2.37	5.00	64	195.02
38,583	2.35	5.15	64	199.21
38,565	2.35	5.33	64	205.94
38,406	2.34	5.39	64	207.61
38,243	2.33	4.87	64	186.26

By using two LUNs, the server was able to push almost 2 times more IOPS at a lower latency compared to the single LUN deployment.

An I/O-intensive database benefits from having more than a single LUN to write and/or read data from disk when necessary. On Linux systems, you can use LVM2 to aggregate LUNs together at the host level. You can also create a striped logical volume instead of the default concatenated logical volume. A striped volume provides the best performance by balancing the I/O across LUNs.

Definition of an I/O-Intensive Database

Observing organizations that have deployed MongoDB to support critical parts of their business frequently reveals two database performance profiles. (These are not the only possible database performance profiles, but they are examples that help to define high performance as a combination of throughput and latency.)

- A sharded cluster generating hundreds of thousands of IOPS (read and write operations) with an average response time of single-digit milliseconds
- A replica set producing a few thousand IOPS with an aggressive average response time of 3-digit microseconds

Given these observations, it's clear that a performance profile cannot be defined only by considering IOPS (read and write operations) or only response time. This document uses the term I/O-intensive to mean a combination of disk throughput (IOPS) and response time (latency) driven by a database.

Simply put, when MongoDB performs a read operation, it first goes to the WT cache to get the data block. If the data block is not there, it looks for it in the file system cache. If the block is not available in the file system cache, the database operation is translated into a disk I/O operation. After all, a database is characterized as I/O-intensive when it must rely on a certain amount of disk I/O operations at a given latency to be able to meet the business requirements defined by the organization.

7 Scenarios and Examples

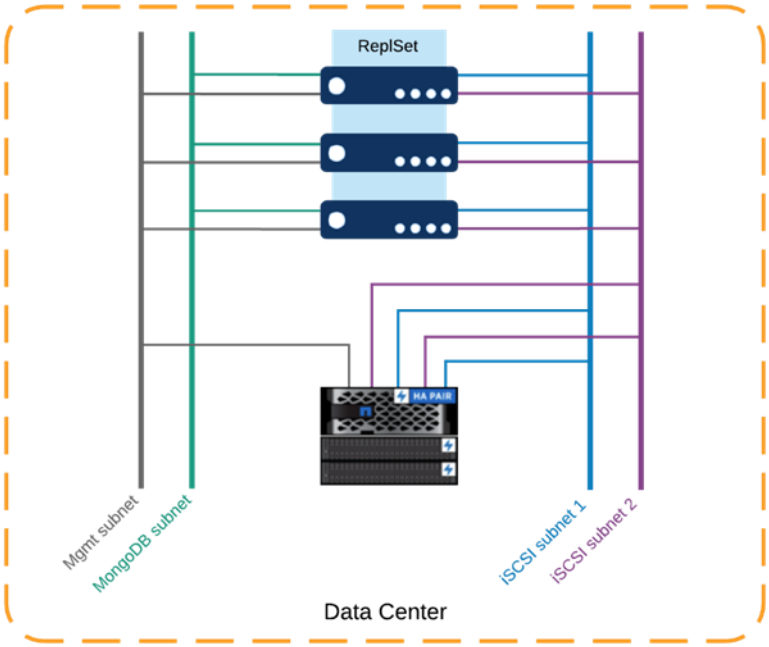
7.1 Single Data Center

Considering an I/O-intensive MongoDB replica set, the configuration for this scenario includes a three-member replica set, LAN switches, and a NetApp AFF system.

Connectivity

When architecting the connectivity between your storage cluster and MongoDB, you should consider performance and data availability. Figure 3 shows two iSCSI subnets that provide iSCSI connectivity from the AFF system to the MongoDB replica set members.

Figure 3) Back-end connectivity between storage nodes and replica set members and front-end connectivity.

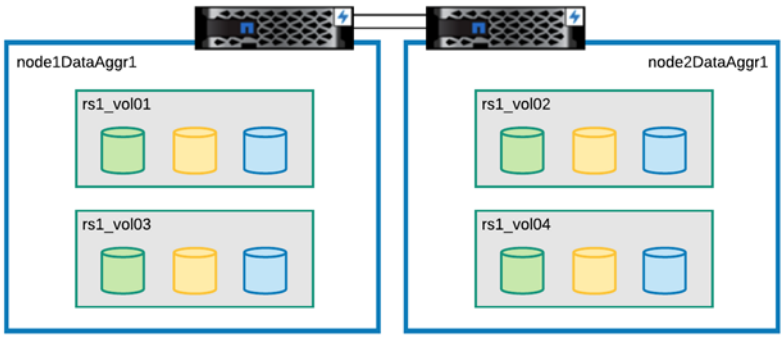


NetApp recommends creating at least one iSCSI LIF for each ONTAP storage node. In this example, there are a total of four iSCSI LIFs. There are two iSCSI LIFS on node1, with each LIF on top of a distinct physical network port on node1. There are another two iSCSI LIFS on node2, with each LIF on top of a distinct physical network port on node2.

Volume and LUN Layout

Defining the number of volumes and LUNs is an important step when architecting a database in a storage system. The number of volumes and LUNs is a critical factor for performance. Also, if you plan to use NetApp Snapshot copies as part of your backup strategy, you must capture the whole state of a replica set.

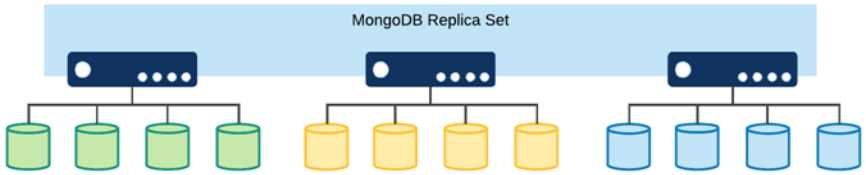
Figure 4) Volume and LUN layout for a three-member replica set.



NetApp recommends creating a set of volumes and distributing the LUNs of each member across these volumes. Figure 4 shows a set of four volumes, with two volumes per storage node and each volume containing one LUN for each replica set member.

Your LUN mapping should resemble the diagram in Figure 5.

Figure 5) LUN mapping for a three-member replica set.



You must adopt a naming convention that helps you to identify the resources created for a replica set. In Figure 4, the naming convention for the volumes is `<replica_set_name>_vol_<incremental_number>` and the convention for LUNs is `<replica_set_name>_<hostname>_lun_<incremental_number>`.

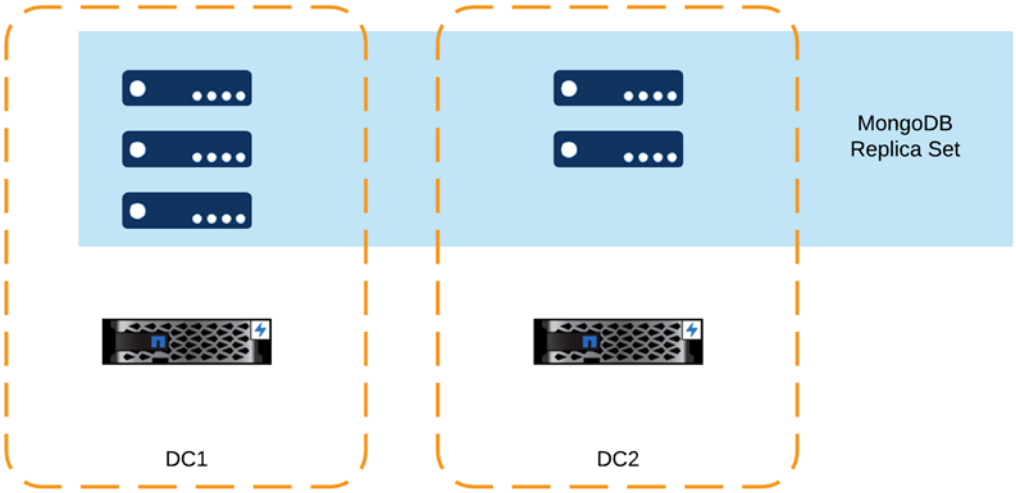
7.2 Multiple Data Centers

MongoDB is a distributed database system that is typically deployed across different locations. In this scenario, considering two data centers, each data center follows the design described in section 7.1, “Single Data Center.”

For this example, a five-member replica set is used. The main data center is called DC1 and the other one is called DC2. The primary role for the replica set is always assigned to servers in DC1. Therefore, the replica set member’s priority must be adjusted to reflect that requirement.

Figure 6 is a high-level diagram of this example.

Figure 6) Five-member replica set distributed across two data centers.



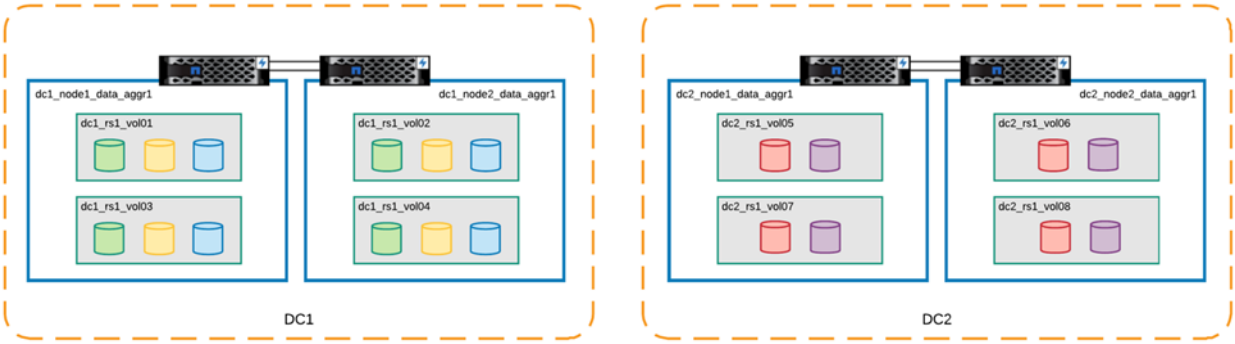
Connectivity

The connectivity follows the same pattern as shown in Figure 3. Make sure that your MongoDB replica set members can communicate with each other across the data center. In other words, one member of the replica set on data center 1 can connect to another member on data center 2 by using the IP and port of the mongodb service. The connectivity should be through the network dedicated for your MongoDB database, not through the server management network.

Volume and LUN Layout

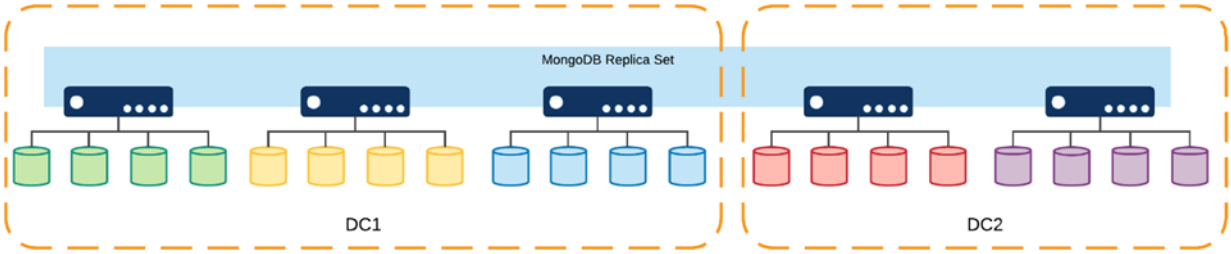
Regarding the volume and LUN layout, NetApp recommends that you follow the same principle used in the previous section. Figure 7 shows the volume and LUN layout for this scenario.

Figure 7) Volume and LUN layout for a five-member replica set distributed across two data centers.



After you have completed LUN mapping, it should resemble Figure 8.

Figure 8) LUN mapping for a five-member replica set distributed across two data centers.



7.3 Hybrid Cloud

It is very common to send data from a corporate data center to the cloud. For example, your production environment might be on the premises but you want to run development and QA in the cloud.

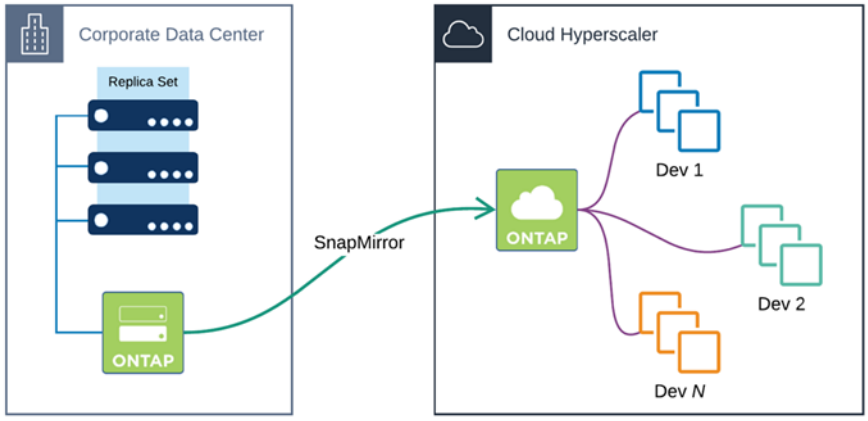
If your dataset is small enough that you can dump your database into a file without causing any interruption in the application, you should use mongodump. If your database is too large for mongodump, you might spin up a new MongoDB instance in the cloud and then add it to your on-premises replica set.

You might encounter a number of different issues with that approach. Your database might not be able to complete an initial sync, or your secondary member in the cloud might get out of sync because it can't keep up with the changes in your dataset. To reduce your chances of experiencing these issues, or to avoid them entirely, you should consider factors such as the round-trip time between your replica set members, the oplog size, and the voting settings.

With ONTAP, storage can take care of data mobility for you. And ONTAP can alleviate concerns about having to alter the MongoDB settings.

By spinning up a Cloud Volumes ONTAP instance in a hyperscale cloud provider of your choice, a SnapMirror relationship between your on-premises ONTAP system and your Cloud Volumes ONTAP instance can easily replicate your dataset into a hyperscaler cloud. Then you can take advantage of ONTAP FlexClone® technology to create multiple virtual copies of your replica set for development and/or testing environments in the cloud (Figure 9).

Figure 9) Replicating a dataset by using storage replication and cloning it to create dev/test instances in the cloud.



This approach can be handy when your MongoDB data files live natively on ONTAP volumes or LUNs without any disk virtualization layer such as VMDK or VHD.

7.4 What If My Database Is Not I/O Intensive?

If your MongoDB database doesn't have an I/O-intensive profile, you can still take advantage of a cloud-connected platform such as ONTAP.

Here are some recommendations you can follow to deploy your MongoDB instance on NetApp:

- Providing multiple connections between your MongoDB servers and your storage system provides high availability if a network port fails. Therefore, it is a best practice to employ the connectivity design described in section 7.1 (Figure 3).
- Rules for initiator groups are the same as stated Block-Based Protocols (SAN) in section 5.4. The MongoDB shared-nothing architecture mandates that LUNs must not be shared across MongoDB cluster members.
- Regarding volumes and LUN layouts, by not having an I/O-intensive profile, your database might not push beyond the performance boundaries of a single volume. Considering one MongoDB replica set, you could create one LUN for each member inside of a single volume. If your database trespasses on the performance boundaries of a volume, you can nondisruptively move any LUN to another volume.

8 Storage Efficiencies

8.1 In-Line Storage Efficiencies

In-line storage efficiencies are enabled by default on AFF and ONTAP Select with an all-flash configuration. Cloud Volumes ONTAP does not support an AFF-like personality. ONTAP is optimized to efficiently and seamlessly provide data reduction savings by applying several in-line data reduction algorithms such as in-line compression, in-line deduplication, and data compaction before committing writes to disk.

In-Line Compression

By default, MongoDB provides data compression for its collections and indexes. WT, MongoDB's storage engine, provides two compression libraries, snappy and zlib. Snappy, the default compression library for MongoDB, can be used with collections and indexes. Zlib can be used only for collections.

Because the database is already compressing data before it writes data down to the file system, in-line data compression in ONTAP might yield minimal savings. However, you don't need to disable in-line compression at the volume level. The in-line compression algorithm senses that there is not enough savings in the compression groups and ramps down compression automatically.

In-Line Deduplication

WT does not offer block deduplication. You should not expect to see deduplication savings for MongoDB. Although the data is the same across all replicas in a replica set, each replica committing data to disk has a unique signature.

A WT block has a page header and a block header. The page header is the first 28 bytes of a WT block, and the block header is the next 12 bytes. The page header has a unique ID called write_gen. Even though the data written is the same for all members, each member has its own write_gen ID.

You do not have to disable deduplication at the volume level.

8.2 Storage Savings

There are other NetApp ONTAP features that can help you to improve your MongoDB operation while saving disk capacity. Snapshot copies can be used as backups of your MongoDB cluster, and FlexClone volumes can help you efficiently create development and testing environments while saving disk space.

NetApp Snapshot Copies

NetApp Snapshot copies play a key role in data protection by providing points of recovery for your database, and they also play an important role in storage efficiency.

A Snapshot copy is a read-only version of your data at a specific moment in time, but it does not consume the same amount of storage as the protected data. For instance, a Snapshot copy of a 1TB database consumes only a few gigabytes of disk space. ONTAP implements a redirect-on-write type of copy, so there is no performance effect when writing or reading data from the system regardless of the number of Snapshot copies taken.

NetApp FlexClone Volumes

NetApp FlexClone is a Snapshot copy-based technology that allows you to create virtual copies of your data. A FlexClone volume is a readable and writable version of a Snapshot copy that consumes only a few gigabytes of space upon creation for its metadata. After the volume is accessed, additional space is consumed only when data is overwritten or when new data is inserted.

For instance, if you have a production database of 1TB and you need five copies of your data for development and testing, without FlexClone you would consume 5TB of additional space. By using FlexClone volumes, you would consume approximately 500GB of additional space for the same five copies (assuming a 10% rate of change per clone).

9 Linux

In the field, some OS-level disk parameters and configurations have been observed that affect the performance behavior of a MongoDB deployment when doing disk I/O. Based on these observations, you should apply the following recommendations to your deployment.

Disk Tuning

Read Ahead

MongoDB's I/O profile is generally random except for the journal (write-ahead log), which is sequential. Disabling read ahead for devices serving your database can save some memory resources.

There are many ways to set up read ahead for a device. One way is to use the `blockdev` command. For example, your MongoDB dbpath might point to `/mongodb/data`, which is mounted on top of `/dev/sdb1`. To disable read ahead on `/dev/sdb1`, run the following command:

```
# blockdev --setra 0 /dev/sdb
```

If you are using LVM, you should also set read ahead to 0 for the logical volume device.

The `blockdev` command is not persistent across reboots. Therefore, you might want to create a udev rule file to change the read-ahead setting persistently across reboots.

I/O Scheduler

I/O schedulers are used to order block I/O categories, and schedulers are categorized as multiqueue and non-multiqueue. Non-multiqueue schedulers are available on Linux 3.x kernels and Linux 4.x kernels, but multiqueue schedulers are enabled by default only on Linux 4.x kernels.

Choosing an I/O scheduler depends on your database I/O profile. For example, “deadline” offers better performance for read-intensive workloads because its implementation favors read operations over write operations. Generally speaking, if you are running MongoDB on a virtual machine, you should use “noop.” If you are running it on a bare-metal system, then “deadline” typically offers better performance.

You should spend some time testing which I/O scheduler fits your database workload better.

Logical Volume Manager

Logical Volume Manager (LVM) is a tool that allows you to combine disk devices into a group, which provides better capacity and data availability from the OS layer. With LVM, you can concatenate, stripe, mirror, and resize logical volumes.

Usually a disk or a disk partition is configured as a physical volume (PV). PVs are grouped together into a volume group (VG), and then you can create one or more logical volumes (LV) for your database.

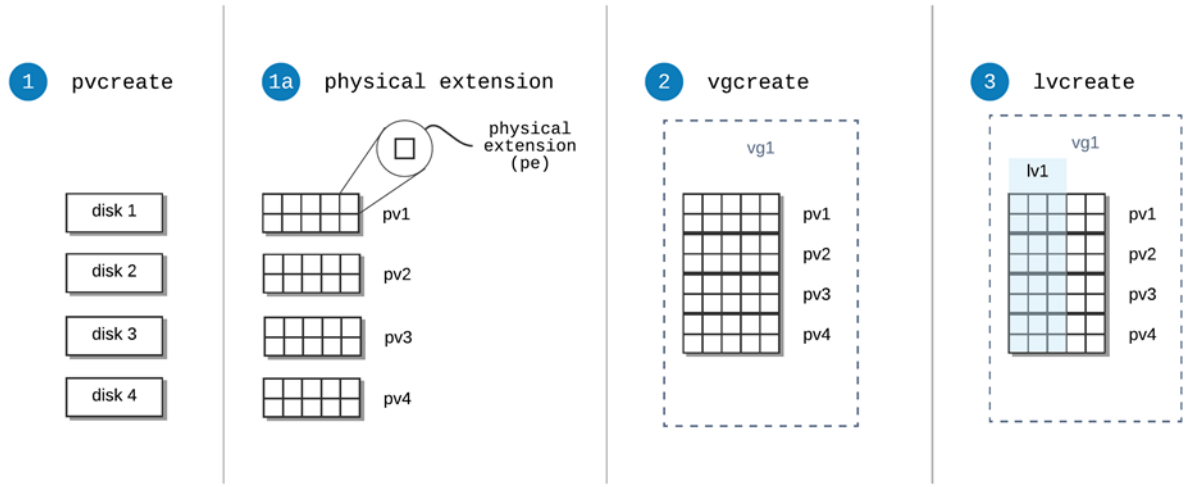
When you issue the `pvcreate` command for a device, LVM chops your disk into 4MB chunks known as physical extensions (PE). Although you can customize the PE size, 4MB is the default.

When you create a VG, specify a list of PVs (for example, pv1, pv2, pv3, and pv4). PEs are then grouped together, and you are ready to create an LV.

By default, when you create an LV, its type is linear. This means that LVM allocates PEs from the first PV; when this PV becomes full, LVM allocates PEs from the next device. From a performance perspective, a linear LV type might cause a bottleneck for an I/O-intensive database.

NetApp recommends creating a striped LV (Figure 10). In a striped LV, LVM allocates one PE of each PV in a round-robin process. In doing so, the I/O in the underlying disk devices (for example, disk 1, disk 2, disk 3, and disk 4) is balanced across all the disk devices.

Figure 10) LVM from physical volume to logical volume.



For example, to create a striped LV using the example shown in Figure 10, run the following command:

```
# lvcreate --stripes 4 --size 10g --name lv1 vg1
```

File Systems

For information about file systems, always check the MongoDB production notes. MongoDB currently recommends XFS or ext4. They strongly recommend XFS for WT to avoid performance issues that might occur with ext4.

10 MongoDB Configuration File Options

NetApp provides the following recommendations for the following MongoDB configuration parameters. These parameters are not turned on by default in MongoDB, but they are very helpful for day-to-day administration.

- `storage.directoryPerDB`. Setting this parameter to true causes mongod to create a directory for each database created. This option is helpful when you need to identify the group of files that are part of a database.
- `wiredTiger.engineConfig.cacheSizeGB`. By default, mongod allocates almost 50% of DRAM for the WT cache. To increase the WT cache, you must add this option to your `mongod.conf` file.
- `security.authorization`. Always enable authorization to protect your database against unauthorized access.

Acknowledgments

Putting together high-quality, relevant content for a technical report is not an easy task and demands much effort. I would like to thank all the people who helped me, with special thanks to:

- Neto from Brazil. Principal CPOC Architect at CPOC Labs.
- Joe Scott. Workload Performance Engineer at Workload Engineering.

Where to Find Additional Information

To learn more about the information described in this document, see the following documents:

- MongoDB Enterprise Advanced Datasheet
  https://resources.mongodb.com/featured-content/datasheet-mongodb-ent-advanced
- NetApp AFF Datasheet:
  https://www.netapp.com/us/media/ds-3582.pdf
- NetApp ONTAP 9 Documentation Center: Introduction and Concepts Guide
  https://docs.netapp.com/ontap-9/index.jsp

Version History

Version	Date	Document Version History
Version 1.0	July 2019	Initial version

Refer to the Interoperability Matrix Tool (IMT) on the NetApp Support site to validate that the exact product and feature versions described in this document are supported for your specific environment. The NetApp IMT defines the product components and versions that can be used to construct configurations that are supported by NetApp. Specific results depend on each customer's installation in accordance with published specifications.

Copyright Information

Trademark Information