HPE Alletra Storage

Technical white paper

Sizing guide for deploying MongoDB on HPE Alletra Storage Server 4210

HPE GreenLake

Contents

Executive summary

With generative AI and other machine-learning technologies grabbing business headlines, organizations are scrambling to figure out ways to manage the unstructured data that is critical to developing those technologies. A relatively new category of database applications, NoSQL, has emerged from obscurity to run in some of the largest data centers in the world, primarily to manage unstructured data. MongoDB is one of the market leaders in the NoSQL space — a document-style database that is designed to manage a wide variety of datatypes — perfect for machine-learning development.

Properly sizing a MongoDB deployment is essential to ensure that the underlying infrastructure has the compute power to manage both ingest and real-time analytics simultaneously. The HPE Alletra Storage Server 4210 has both the compute capability and massive storage footprint to seamlessly handle the needs of a fast-paced MongoDB configuration. This paper outlines some sizing recommendations when considering a MongoDB deployment using the HPE Alletra Storage Server 4210.

About MongoDB

MongoDB is classified as a document NoSQL database, storing its data in records called “documents.” These documents can store virtually any kind of data: videos, pictures, alpha-numeric data, or even actual documents like PDFs. Queries are directed at indexes that reference the documents, and it is important that the “working set”, or combination of documents and indexes accessed at any given time, reside in RAM as much as possible. Sizing the host hardware properly ensures that there are always sufficient resources so that the working set functions with the ultra-low latency and high-performance that MongoDB users expect.

Production MongoDB environments are deployed in clusters, either as a replica set or a sharded cluster. Replica sets contain between three and seven hosts, with at least four recommended for production environments. One host is the primary that receives all the database writes and usually most of the reads. The rest are secondary hosts, whose main function is to act as disaster-recovery options for the cluster. When more throughput and/or compute power is needed, the database can be divided into shards in a sharded cluster. Each shard is maintained in its own replica set, and additional nodes are needed for traffic management and quorum maintenance, so running an effective high-performing MongoDB environment without having to shard is always preferable.

About the HPE Alletra Storage Server 4210

The HPE Alletra Storage Server 4210 is designed to provide significant compute power, memory capacity, and massive storage space to meet the needs of the most challenging application environments. The HPE Alletra Storage Server 4210 boasts a maximum of over 300 terabytes of storage and 4 terabytes of RAM in a 1U chassis, minimizing rack occupancy and data center power usage. Dual CPUs and network adapters with speeds up to 200 gigabits per second ensure plenty of cores and bandwidth. This combination of power, capacity, speed, and compactness fits perfectly in a MongoDB environment, allowing deployment of either replica sets or sharded clusters in bare-metal or containerized configurations.

Considerations when sizing a MongoDB environment

Replica set or sharded cluster?

When deploying a MongoDB environment, careful consideration is needed as to which type of clustering should be used. As stated earlier, MongoDB deployments use either replica sets or sharded clusters, and there are significant differences between the two.

In a replica set, the primary node handles the vast majority of the throughput traffic. The secondary nodes receive copies of each write in a process known as “eventual consistency.” This means each database transaction is replicated from the primary node to all secondary nodes, but not all secondaries necessarily have to acknowledge the replication immediately. Consistency can be tuned, for example, to have only half of the nodes acknowledge. This facilitates longer-distance replication without sacrificing ingest speed waiting on an acknowledgement from a node in a faraway data center.

With a replica set, it is imperative to consider how much traffic the primary node can handle without impeding performance. One way to mitigate this is to allow some or all the secondary nodes to handle database reads. Still, there might be a point at which the ingest rate is more than the primary node can handle.

It is at this point that sharded clusters could be necessary. Dividing up the database into multiple parts, each hosted by its own replica set, makes more bandwidth and processing power available to the MongoDB instance; however, it does come at a considerable cost.

Figure 1 illustrates the difference in node counts between a replica set and a sharded cluster that divides the database into three shards.

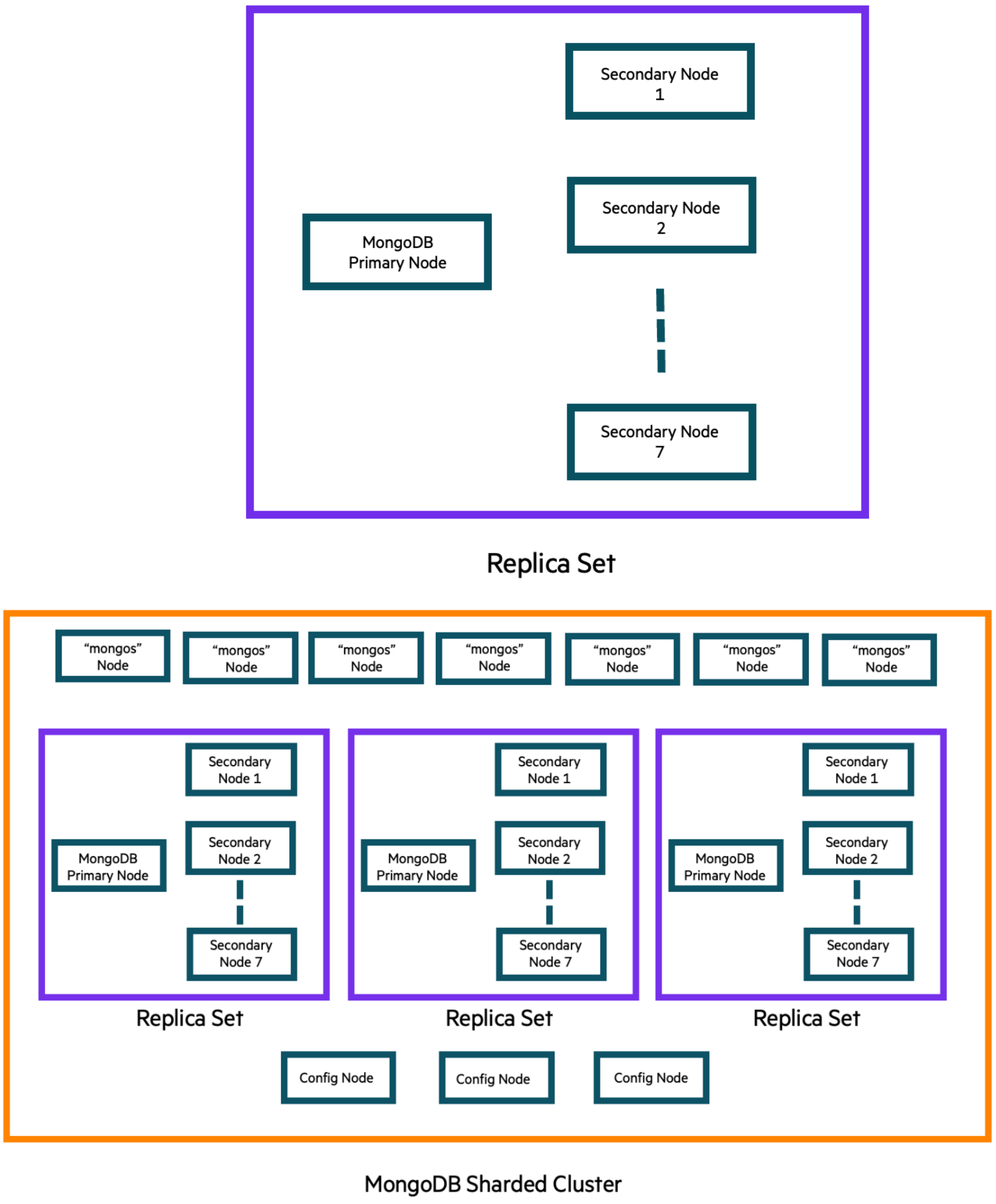

Figure 1. Comparison of MongoDB replica set vs. MongoDB sharded cluster with three shards

Each shard resides in its own replica set, which in the above case at least triples the total number of nodes for the cluster. In addition, because a sharded cluster requires a separate config cluster to monitor the nodes and "mongos" nodes to route traffic to the most correct shard, the node count goes up yet again. Due to this, the vast majority of MongoDB deployments remain at the replica set level. If traffic overwhelms the capability of the primary node of a replica set, though, a sharded cluster might be needed anyway.

MongoDB server host considerations

When deciding how many server resources are necessary for each MongoDB host, CPU and networking are rarely bottlenecks when it comes to MongoDB performance. Instead, it is important to focus on three main factors:

- How many IOPS are required
- Available RAM
- Available disk space to house the data

IOPS: Calculating the expected IOPS for a MongoDB deployment depends largely on three factors: the expected query rate, the expected ingest rate, and a well-designed indexing scheme. The expected query rate and ingest rate must also include estimates of growth over time. As this can affect latency, an additional consideration is how much latency can be tolerated as acceptable performance. A poor index design will increase the response time, reducing IOPS performance. It will also decrease the likelihood of that response being complete or useful.

After these factors are known, the server host needs the necessary technology for moving queries quickly to the CPU to ensure that the IOPS rate meets expected levels.

RAM: The amount of memory needed for a MongoDB deployment is primarily determined by the "working set" size. A working set is the number of documents and indexes that can fit entirely in memory at a given time for faster processing. Knowing the number of users at peak times and the amount of data those users are trying to access during their session will also help define the proper working set size. As this size will no doubt grow over time, the RAM availability should be sufficient to match that growth. Ideally, the available RAM should hold all the needed documents and indexes even at peak times. At a minimum, the server host RAM should hold at least the indexes, with some room to pull in documents from disk as required. In that case, the onboard disk storage technology needs to be able to move the documents to RAM as fast as possible.

Storage space: MongoDB users prefer to have the database disk storage onboard if high performance and low latency are a must. Because these are prevailing reasons for choosing MongoDB in the first place, having the fastest onboard storage with significant room for growth is vital. With MongoDB, when onboard storage runs out of available space, this can force a migration to a sharded cluster since each node of a replica set has to hold a copy of the entire database. The optimal solution would be to have server hosts with enough storage density to handle future growth, minimizing the need to ever shard the database strictly for storage capacity reasons.

MongoDB with containers

With a massive storage footprint, combined with powerful dual-CPU and high-capacity RAM options with the HPE Alletra Storage Server 4210, running MongoDB in containers is an intriguing possibility. With less overhead than virtualized environments, deploying a persistent set of containers via Kubernetes for MongoDB can be an economic option to make the most use of the HPE Alletra Storage Server 4210 capacity.

Rather than traditional ephemeral container environments where containers can be spun up and spun down as needed and at will, database deployments are focused on persistent storage and containers that are not meant to be spun down. In this scenario, the MongoDB primary and secondary nodes will be much larger in scope, but multiple nodes can still fit in a single HPE Alletra Storage Server 4210. For example, you could deploy a six-node MongoDB replica set on two separate HPE Alletra Storage Server 4210s, evenly divided at three nodes apiece. They could be placed in different data centers for DR purposes and eventual consistency throttled to allow the off-site nodes of the set to receive transaction copies asynchronously.

Figure 2 shows a containerized MongoDB replica set on an HPE Alletra Storage Server 4210.

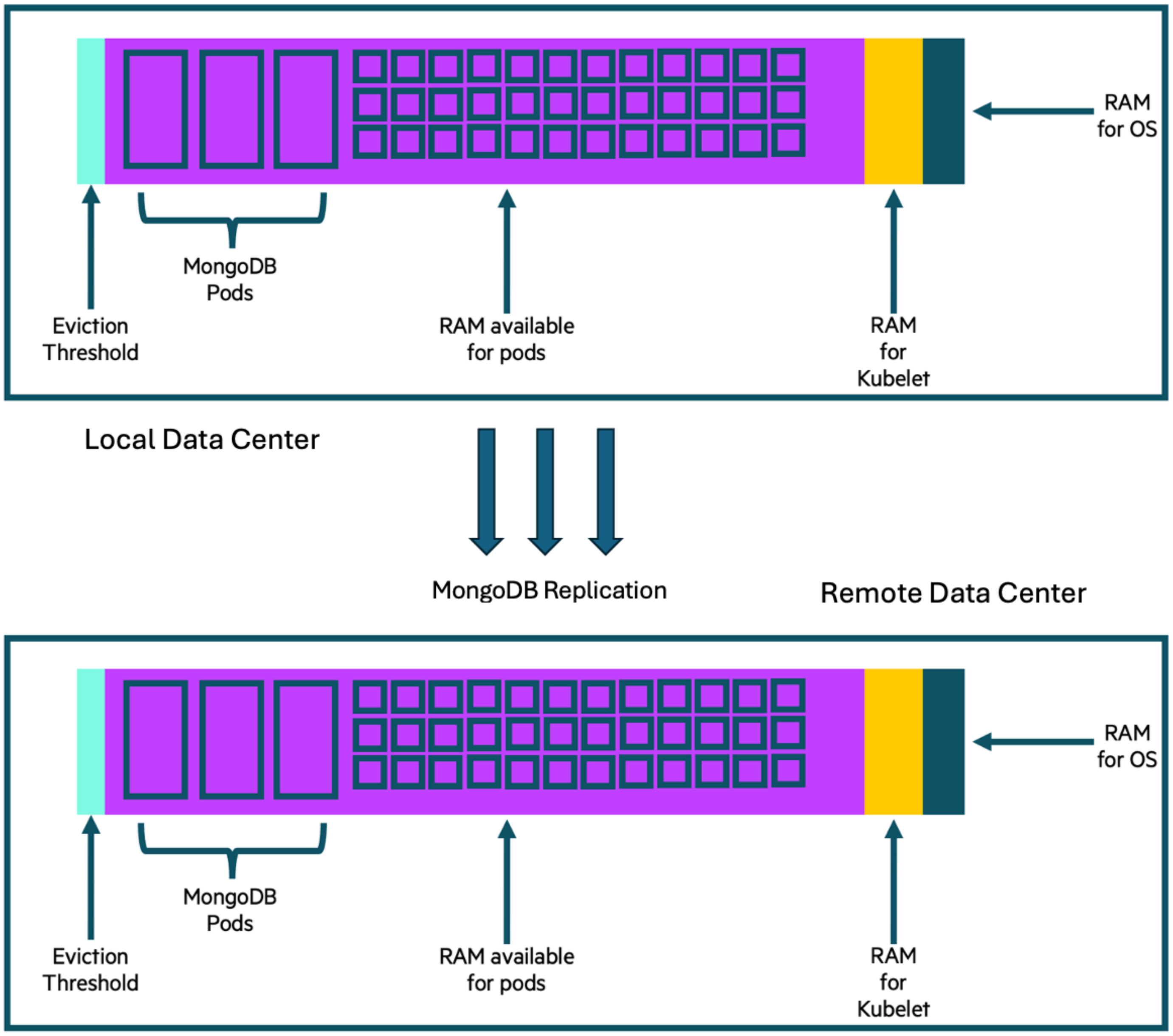

Figure 2. Containerized MongoDB replica set on HPE Alletra Storage Server 4210

However, the more effective use of the HPE Alletra Storage Server 4210 in a MongoDB containerized environment may be in a sharded-cluster deployment. Consider a three-shard MongoDB cluster in which each of three HPE Alletra Storage Server 4210s hosts at least one copy of each shard. In addition, the ephemeral characteristic of a Kubernetes setup could be used to spin up and spin down “mongos” servers, which direct query and ingest traffic to the most appropriate shard cluster. As mongos servers do not actually store any data, it is easy to spin up several when traffic is heavy and spin them back down when traffic eases. These mongos containers could be spread across the three HPE Alletra Storage Server 4210 hosts, ensuring ample bandwidth for the traffic experienced throughout the day.

Figure 3 illustrates a containerized MongoDB sharded cluster hosted by three HPE Alletra Storage Server 4210 systems.

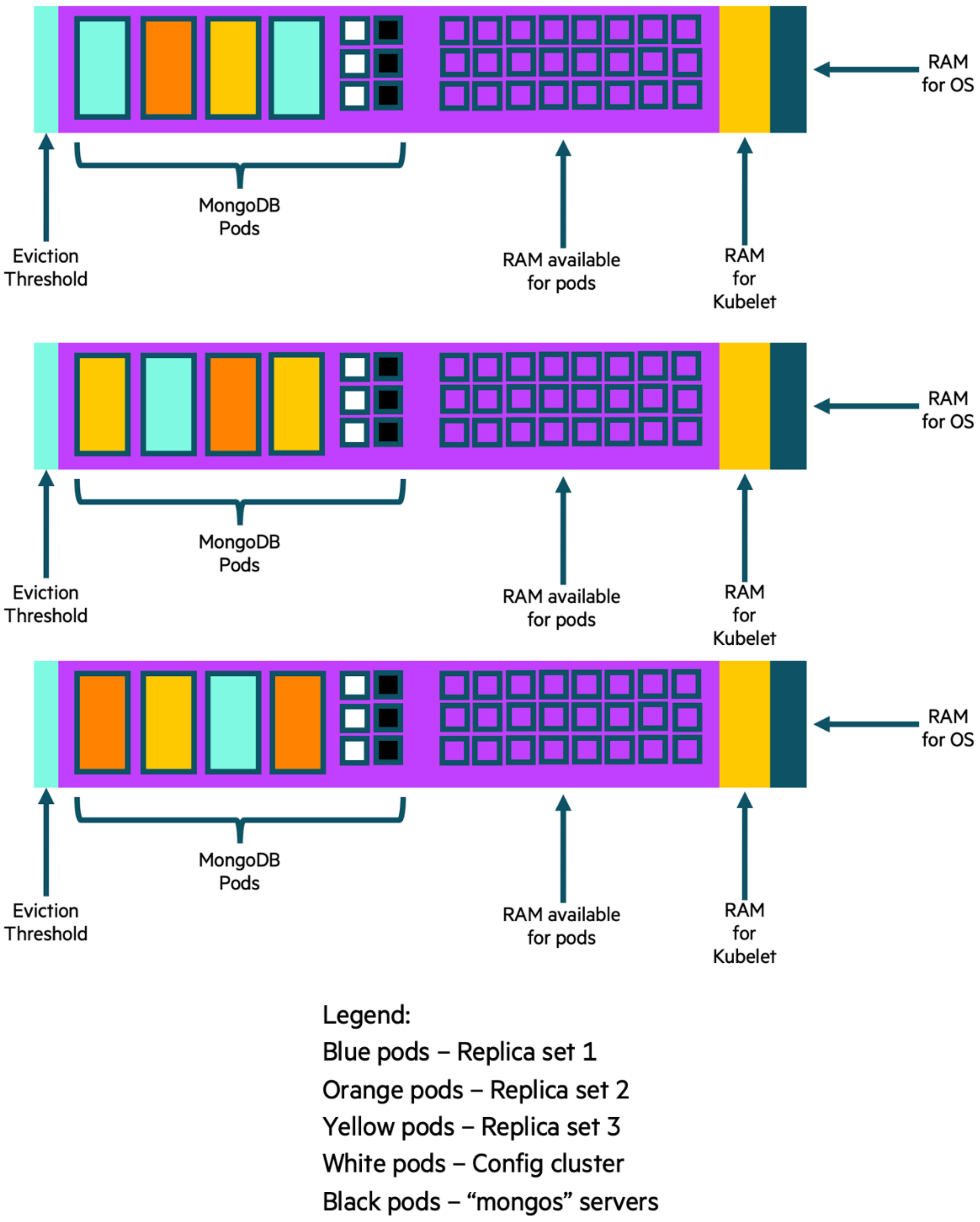

Figure 3. Containerized MongoDB sharded cluster hosted by three HPE Alletra Storage Server 4210 systems

Sizing for MongoDB bare-metal deployments

Keeping in mind the server host considerations discussed earlier, the following graphic is a good starting point for discussion into what levels of compute, memory, and disk storage might be necessary for a MongoDB deployment. One thing to keep in mind is that, even though roughly 80% of MongoDB instances are one terabyte in size or less at this point, MongoDB's focus on its role in AI development will naturally make these instances grow at a substantial rate over time.

The accuracy of AI applications depends on the growth of the data from which it "learns." The wider its potential knowledge base, the greater the chance that it will be able to produce the right results for its users. MongoDB instances must be able to scale to meet these dependencies and its underlying hardware must scale up and scale out to handle this growth. The likelihood is that the 80% figure for instances at or below one terabyte will drop off dramatically in a short period of time.

Table 1. MongoDB deployment suggestions on the HPE Alletra Storage Server 4210

"T-shirt" size of MongoDB instance	MongoDB working set size	Storage capacity of MongoDB instance	Suggested RAM capacity per host	Minimum storage capacity per host
Medium	64 GB	350 GB	128 GB	1.6 TB NVMe SSD
Large	128 GB	640 GB	256 GB	3.2 TB NVMe SSD
X-Large	256 GB	1.2 TB	512 GB	6 TB NVMe
XX-Large	512 GB	2.5 TB	1 TB	12.5 TB NVMe SSD

Because the MongoDB WiredTiger storage engine will, by default, allocate 50% of the available RAM minus 1 gigabyte, it will be necessary to double the RAM based on the working set size. For example, the host will need 128 GB for the medium-level instance size because of the 64 GB working set. Also, the minimum storage capacity figures are based on the NVMe SSD sizes at the smaller range of the HPE Alletra Storage Server 4210 and assume a more conservative RAID-0 implementation. If a RAID set with parity is desired, these sizes will be larger to account for the extra SSDs.

The point to keep in mind is that these figures leave a great deal of growth available in a single HPE Alletra Storage Server 4210 for both RAM and disk storage capacities, so an investment in it will allow the MongoDB instance to grow at the scale that their AI focus will demand without adding server count.

Sizing for MongoDB containerized deployments

The sizes for MongoDB Kubernetes pods will naturally skew smaller individually, but the key takeaway from Table 2 is that because of the HPE Alletra Storage Server 4210 capacities, multitenancy on each server is absolutely in play. For example, a data analytics environment with an application such as Spark for data queries would benefit greatly from being close to the data it queries. Traveling down the same PCI bus to the RAM and CPU as the MongoDB data is about as close as it gets, and the performance would likely reflect that.

Table 2. Containerized MongoDB deployment options with the HPE Alletra Storage Server 4210 hosting containers

"T-shirt" size of MongoDB instance	MongoDB working set size	Storage capacity of MongoDB instance	Suggested RAM capacity per pod	Minimum storage capacity per host
X-Small	4 GB	20 GB	8 GB	800 GB NVMe SSD
Small	8 GB	40 GB	16 GB	1.6 TB NVMe SSD
Medium	16 GB	80 GB	32 GB	3.2 TB NVMe
Large	32 GB	160 GB	64 GB	6.4 TB NVMe SSD

Again, the WiredTiger engine in MongoDB will take half of the available RAM minus 1 GB, so the suggested pod memory capacity reflects this. The SSD capacities again reflect a conservative RAID-0 configuration with the smallest drive capacities for the HPE Alletra Storage Server 4210. These numbers emphasize the growth capability for both the MongoDB instance and the availability for deployment of additional apps, especially those that would work in concert with MongoDB to deliver the needed workloads with low latency and high performance.

One other significant point to keep in mind is the CPU and networking capability of the HPE Alletra Storage Server 4210. Because each one is deployed with dual CPUs, and network ports at speeds up to 200 GB/per second are options, the multitenancy capability can go far beyond the MongoDB/Spark scenario. The HPE Alletra Storage Server 4210 provides this in a 1U footprint that saves rack space and reduces the environmental overhead of a multiple-commodity server deployment.

Summary

MongoDB is undergoing a transformation into new markets as technologies such as AI and data analytics find the architecture of MongoDB and other NoSQL applications a perfect fit. As they grow from niche players to the forefront of enterprise business applications, their original homes on commodity servers are no longer sufficient. They need the kind of compute capability and growth potential that the HPE Alletra Storage Server 4210 provides. Whether hosting a MongoDB instance on bare-metal or in a Kubernetes environment, the HPE Alletra Storage Server 4210 can transform the existing MongoDB infrastructure into one that meets current needs and can scale as quickly as these new app environments will require.

Resources

HPE Alletra 4210 data storage system
buy.hpe.com/us/en/storage/disk-storage-systems/storage-servers/alletra-4000/hpe-alletra-storage-server-4210/p/1014699109

MongoDB documentation
mongodb.com/docs/manual/

Learn more at

hpe.com/us/en/storage/alletra-4000.html

Visit **HPE.com**

Chat now (sales)