Vet Software

Hardware and Software specifications

* **Introduction:**

The vet software will be a complete end to end solution for all the needs of veterinary doctors. It will consist of the following modules:

These modules will be more described in the functional specification document.

The main theme of the project will be portability and ease of access.

* **Technology:**
* Database:

Database: Mongo DB (2.6)

Administration:

Administration of Mongo DB contains the following aspects:

1. **Backup :** For initial installation, use Mongo Management Service for backup of the database across the application.
2. **Monitoring:** Similarly for monitoring, use Mongo Management Service for monitoring.
3. **Run time database configuration :**

The runtime configuration of the database is necessary to determine the way in which the database will run in production. The following configuration parameters need to be set properly. This list is not exhaustive and will grow in the future.

List of configurations:

1. **Fork = true**

fork is true, which enables a daemon mode for mongod, which detaches (i.e. “forks”) the MongoDB from the current session and allows you to run the database as a conventional server.

1. **Bind\_ip = 127.0.0.1**

bindIp is 127.0.0.1, which forces the server to only listen for requests on the localhost IP. Only bind to secure interfaces that the application-level systems can access with access control provided by system network filtering (i.e. “firewall”).

1. **Port = 27017**

port is 27017, which is the default MongoDB port for database instances.

1. **Quiet =true**

quiet is true. This disables all but the most critical entries in output/log file, and is not recommended for production systems. If you do set this option, you can use setParameter to modify this setting during run time.

1. **Dbpath = “Path to where the DB is installed”**
2. **Logpath =”Path to where the log file should be saved”**
3. **Logappend=true**

logAppend is true, which ensures that mongod does not overwrite an existing log file following the server start operation.

1. **Journal=true**

storage.journal.enabled is true, which enables journaling. Journaling ensures single instance write-durability. 64-bit builds of mongod enable journaling by default. Thus, this setting may be redundant.

1. **auth = true**

Authorization is true enables the authentication system within MongoDB. If enabled, you will need to log in by connecting over the localhost interface for the first time to create user credentials.

1. **slowms = 50**

slowOpThresholdMs configures the threshold for to consider a query “slow,” for the purpose of the logging system and the database profiler. The default value is 100 milliseconds. Set a lower value if the database profiler does not return useful results, or a higher value to only log the longest running queries.

1. **profile = 3**

mode sets the database profiler level. The profiler is not active by default because of the possible impact on the profiler itself on performance. Unless this setting has a value, queries are not profiled.

1. **verbose = true**

verbosity controls the amount of logging output that mongod write to the log. Only use this option if you are experiencing an issue that is not reflected in the normal logging level.

1. **objcheck = true**

wireObjectCheck forces mongod to validate all requests from clients upon receipt. Use this option to ensure that invalid requests are not causing errors, particularly when running a database with untrusted clients. This option may affect database performance.

1. **Production notes:**

While deploying to production, the following things have to be remembered and followed. The configurations set up using these affect production performance of the database.

1. Packages:
   1. Always use the latest stable release. Keep the instance of MongoDB up to date.
   2. **Always use 64-bit build of MongoDB.** 32 bit build support data only upto 2GB and are used mainly for development and testing.
   3. The operating system that will be used for deployment will be Linux (See server details for more).
2. Concurrency:
   1. The current version of MongoDB has a reader-writer lock that allows concurrent access to database but gives exclusive write access to single operation per database.

The upcoming version has write concurrency. Keep the database build updated.

1. Journaling:
   1. MongoDB uses write ahead logging to an on-disk journal to guarantee that MongoDB can recover the write operations immediately following a crash.
   2. To ensure that mongoDB will be able to recover its data files, **leave journaling enabled**.
2. Networking:
   1. Use Trusted Networking environments:

Always run MongoDB instances in a trusted environment, with network rules that prevent access from all unknown machines, networks and systems.

By default, authorization is not enabled. Use security/Auth mode if needed to set it. More about this in the security section.

* 1. Connection pools: To avoid overloading the connection resources of a single instance, ensure that clients maintain reasonable connection pool sizes.

The connPoolStats command returns information regarding the number of open connections to the current database.

1. Hardware Considerations:

The hardware for the most efficient MongoDB installation have the following properties.

* 1. Allocate sufficient CPU and RAM.

More RAM and faster CPU increases performance.

Databases are not CPU bound. Increasing number of cores can help, but does not give significant return.

* 1. Use SSDs

MongoDB has good price-performance ratio with SSDs

* 1. Avoid remote file systems.

It can create performance issues with database.

* 1. Avoid using NUMA hardware.

It creates low performance and high process usage. If needed, check mongodb docs to find ways to fasten it.

* 1. Assign SWAP space:

Allocating SWAP space can avoid issues with memory contention and can prevent the mongod process from being killed by Linux.

* 1. RAID:

MongoDB deployments should use disks backed by RAID-10.

RAID-0 provides good write performance but gives poor read performance.

* 1. Remote Filesystems:

The network file system is not recommended to be used for MongoDB.

* 1. Separate components onto different storage devices:

For improved performance, consider separating your database’s data, journal and logs onto different storage devices.

1. Architecture:
   1. Write Concern:

Write concern is the guarantee provided by the database that a write operation has been successfully performed. MongoDB offers various levels of write concerns. Consider the right one for the system.

* 1. Replica Sets:

This will be covered in the replication section of the database specifications.

* 1. Sharded Clusters:

This will be covered in the sharding section of the database specifications.

1. Platforms
   1. While running MongoDB in production on Linux, **make sure the Linux kernel version is 2.6.36 or above.**
   2. Use the XFS or Ext4 file format.
   3. Important Considerations:
      1. Turn off atime for the storage volume containing the database files.
      2. Set the file descriptor **–n** and user process limit **–u** to above **20,000**. A low u limit will cause issues in performance for MongoDB.
      3. Disable **transparent huge pages** as MongoDB performs better with 4096 byte virtual memory pages.
      4. Disable Non Uniform Memory Access (NUMA) from the BIOS.
      5. Ensure that readahead settings for the devices that store the database files are appropriate.
      6. Use Network Time Protocol to synchronize time between your hosts. This is essential in sharded clusters.
2. Performance Monitoring
   1. Iostat:

Use the iostat command to see if disk I/O is the bottleneck for your database.

* 1. Bwm-ng:

This is a command line for monitoring network use. If a network based bottleneck is suspected, use this tool to begin the diagnostic process.

Optimization strategies for database

1. **Evaluate the performance of current operations**

The following are few techniques to evaluate operational performance.

Use the database profiler to evaluate operations against the database

* 1. Use **db.currentOp()** to evaluate mongod performance.

This procedure reports on current operations of mongod

* 1. Use **$explain** to evaluate query performance

The explain method returns statistics on a query and the index mongoDB selected to fulfill the query.

1. **Use Capped collections for fast reads and writes**

Capped Collections are fixed size collections which keep the documents well ordered. Capped collections can receive very high-speed writes and sequential reads.

1. **Optimize query performance**
2. Create indexes to support queries

For commonly used queries, create indexes. If a query searches multiple fields, create a compound index. Scanning an index is much faster than scanning a collection.

Indexes also improve efficiency of queries which are routinely sorted.

1. Limit the number of Query results to reduce network demand

MongoDB returns results in groups of multiple documents. If the number of results required is known, set a limit on the number of values to be returned.

1. Use projections to return only necessary data

When only some parts of the data are needed, better results can be achieved by only querying the needed fields

1. Use $hint to select a particular index

The query optimizer chooses the optimal index to be used for a specific operation. However, you can force mongodb to use a specific index using the hint() method.

1. Use the increment operator to perform operations server side

Use the increment operator to increment or decrement values in a document. This helps avoid selecting, modifying and writing the document back just to increment a field.

1. **Design Notes**
2. Schema Considerations
3. Dynamic Schema

Documents have a dynamic schema. Collections don’t enforce document structure.

Consider the following things:

* + - The exact set of collections to be used
    - The indexes to be used
    - Shard key declaration : choosing a good shard key is necessary as it cannot be changed later.

1. Case sensitive strings

MongoDB strings are case sensitive.

Consider the following things:

* + - Storing data in a normalized format
    - Using regular expressions with the i option
    - Using $toLower and $toUpper in the aggregation framework

1. Type sensitive fields

The type of field should be considered while querying a document.

1. General Considerations
2. By default, update affects one document.

In order to update multiple documents, set the update multi option to true.

1. The BSON document size is currently limited to 16MB. Further than that, GridFS should be used.
2. Replica Set Considerations
3. Use an odd number of Replica Set Members

Replica sets perform consensus election. To ensure a proper vote, keep odd number of replica sets

1. Keep replica set members up to date

MongoDB replica sets support Automatic Failover. It is important for the secondaries to be up to date.

* + 1. Use monitoring tools to alert for lag events
    2. Specify appropriate write concern

1. Sharding Considerations

Consider the following points for sharding:

* Shard Keys have to be picked carefully. The shard key cannot be changed for an already sharded database.
* To shard large amounts of data, create a new empty sharded collection, and ingest the data from the source collection using an application level import operation.

1. Analyze Performance

Analysis of performance is necessary while running a database in production

Consider the following things to monitor and analyze performance of the system.

1. Degraded performance of MongoDB is a function of the relationship between quantity of data stored in the database, the amount of system RAM, the number of connections to the database and the amount of time the database spends in locked state.