2016

MongoDB AMI Documentation

METHODOLOGY AND USAGE

Table of Contents

1. Introduction

The objective of this document is to inform about different topics regarding the configuration, testing and usage of the Amazon Machine Image (AMI) of a MongoDB running on the Windows Server 2012 R2 operating system developed for Tackle.

First, this document talks about a variety of configuration considerations that were taken into account in the development of the AMI, then the YCSB benchmark results are being presented as a justification of the AMI's performance on the different types of instances. Finally, the usage section contains instructions on several topics about the use of the database.

2. Tuning

2.1 Mongo installation

The database version is MongoDB community edition v3.2.10 64-bit with SSL support for Windows. This is latest stable release of the NoSQL database program.

The installation comes with all the available components, including mongod and mongo components. mongod component is the core database process; it handles data requests, manages data access, and performs background management operations[i]. mongo component is an interactive JavaScript shell interface to MongoDB, which provides a powerful interface for systems administrators as well as a way for developers to test queries and operations directly with the database[ii].

2.2 Configuration file

Mongod service is being configured at startup with the mongod.cfg file located at C:\Program Files\MongoDB\Server\3.2. In this file we specify the path to the log file and the storage path, along with the authentication setting in order to force authentication to use the database.

```
systemLog:
    destination: file
    path: c:\data\log\mongod.log
storage:
    dbPath: c:\data\db
security:
  authorization: enabled
```

The configuration file offers a variety of options which make managing mongod easier[iii]. These settings must be configured depending on the user needs. The mongod component must be restarted when the configuration file is changed so that the changes take effect. In particular the MongoDB service registered in Windows must be stopped and started again.

2.3 MongoDB Authentication

For authentication, the default authentication mechanism of MongoDB is being used. The current version uses the SCRAM-SHA-1 as the default challenge and response authentication mechanism. SCRAM-SHA-1 is an IETF standard, RFC 5802, that defines best practice methods for implementation of challenge-response mechanisms for authenticating users with passwords[iv].

To enable authentication, a user with the name “admin” was created with the built-in role userAdminAnyDatabase which provides superuser access. The userAdminAnyDatabase role also provides the ability to create and modify roles and users. This role applies to all the databases[v].

Since the admin user was configured with the same password, the password must be changed as soon as the instance is launched.

2.4 Network access

By default, the policy in Windows Firewall allows all outbound connections and blocks all incoming connections. The port 27017 is the default port for mongod instances. Using the netsh command line tool we explicitly allow all incoming traffic to port 27017, which allows the application server to connect to the mongod.exe instance.

There are other ports which MongoDB uses by default[vi]. These ports enable different capabilities and depends on the user needs.

2.5 MongoDB as a service

A service with name “MongoDB” was created so that the database service could be started on boot. When the MongoDB is started, it takes into account the settings added to the configuration file. Every change to the configuration file requires the MongoDB service to be stopped and started again so that it can acknowledge these changes.

The net command is used to update, fix, or view the network or network settings. It allows us to stop and start the MongoDB network service.

2.6 TCP/IP-Related Registry Entries

The MongoDB documentation suggest to follow the recommendations regarding the TCP/IP-Related Registry Entries in the Windows Server Technet Article on TCP Configuration[vii]. This registry changes aim to eliminate different kinds of vulnerabilities that malicious users or attackers could take advantage. These registries and their configuration prevention are:

- **DisableIPSourceRouting**: prevents attackers from using source routed packets to obscure their identity and location.

- **KeepAliveTime**: prevents attackers establish numerous connections to attempt to cause a DoS condition.

- **PerformRouterDiscovery**: prevents that an attacker who has gained control of a computer on the same network segment as a router could configure a computer on the network to impersonate the router.

- **TcpMaxDataRetransmissions:** prevents that a malicious user exhaust a target computer's resources

2.7 Additional registry entries

Besides the TCP/IP-related registry entries, two additional entries were considered from the Windows Server Technet Article. These registry entries were chosen since they provided a clear security prevention and also that they may not conflict or cause problems with other services and applications. Other registry entries from the article in which the current default value provided the required security were also excluded. These two items are:

- **NtfsDisable8dot3NameCreation**: prevents that an attacker who has gained access to the file system access data or run applications.

- **ScreenSaverGracePeriod**: prevents that someone approach the console and attempt to log on to the computer before the lock takes effect.

2.8 CIS guide for configuring Windows server 2012 r2

The Center for Internet Security's (CIS) Microsoft Windows Server 2012 R2 Benchmark is a guide which provides prescriptive guidance for establishing a secure configuration posture. The benchmark was created using a consensus review process comprised of subject matter experts from a diverse set of backgrounds[viii].The benchmark defines different profile definitions depending of the role of the system.

The Level 1 – Member Server configuration profile was taken into account to make the corresponding changes in the Group Policy Editor and the registry. These configurations aim to be practical and prudent, provide a clear security benefit and not inhibit the utility of the technology beyond acceptable means.

Level 2 configuration profiles were avoided since they may negatively inhibit the utility or performance of the technology; only Level 2 configurations regarding the TCP/IP-related registry entries recommended by the MongoDB documentation were taken into account. Level 1 - Domain Controller profile configurations were avoided as well since it is not the intended role of the instance.

Configurations that restricted the accessibility required for an AMI by the Amazon Marketplace were also excluded; this includes configurations which disabled the accessibility through RDP (Remote Desktop Protocol) which is required to allow OS-level administration capabilities to allow for compliance requirements, vulnerability updates and log file access[ix].

2.9 AMI creation and Sysprep

AWS Marketplace requires Windows-based AMIs to use the most recent version of Ec2ConfigService. Ec2Config service starts when the instance boots and performs tasks during startup and each time you stop or start the instance.

Ec2Config performs initial startup tasks when the instance is first started[x]. These are:

- Set a random, encrypted password for the administrator account
- Generate and install the host certificate used for Remote Desktop Connection
- Dynamically extend the operating system partition to include any unpartitioned space
- Execute the specified user data (and Cloud-Init, if it's installed)

These tasks can be enabled manually to run them again, or by running Microsoft System Preparation (Sysprep). The Sysprep tool is recommended to create a standardized Amazon

Machine Image (AMI)[xi]. This way the AMI can comply with the security and accessibility requirements of AWS Marketplace. Once the instance was shutdown with the Shutdown with Sysprep option and the random administrator password, the AMI was created from the EC2 instance manager. When an instance is launch with the new AMI, all the initial startup tasks listed previously will be executed including the random generated administrator password.

2.10 CloudFormation script

AWS CloudFormation is a service that helps you model and set up different types of resources. With a template that describes all the AWS resources needed, CloudFormation takes care of provisioning and configuring those resources[xii].

The CloudFormation script launches the instance and automatically configures its Security Group. This security group enables two ports: 3389 for Remote Desktop Protocol (RDP) and 27017 which is the default port for MongoDB service.

Additionally, the script does the following:

- Sets the available types of instances
- Sets the EC2 Key Pair as a parameter in order to generate the administrator password
- Sets the name on the EC2 instance the same as the stack name
- References the image id of the corresponding AMI in order to launch the EC2 instance

3. YCSB Benchmark

3.1 Configuration

The Yahoo! Cloud Serving Benchmark (YCSB) is an open-source specification and program suite for evaluating retrieval and maintenance capabilities of computer programs and it is often used to compare relative performance of NoSQL database management systems.

The YCSB benchmark test was run against the AMI running in many types of instances in order to guarantee the correct behavior of the database on each one of them.

The test has two phases, the loading phase and the transaction phase. The loading phase inserts the proper kind of records so that the transaction phase can execute transactions against those records. Both phases provide metrics about the system performance.

YCSB includes a set of core workloads. Each workload defines a mix of read, insert, update and scan operations. Each workload defines a basic benchmark for cloud systems. These workloads should be selected depending on the database usage that is expected to have. For this test the Workload A was used, which consists on a 50/50% Mix of Reads/Writes, this way we can have a wide variety of operations along the instances.

For each instance multiple runs were made varying the number of threads so we can see how the throughput and latency behaves. Each test consists of 10 million operations which ensures a significant amount of data but without spending too much time on each run.

3.2 Loading phase

Multiple loads were run in each instance in order to check the insert operation performance. Figure 1 show the throughput for each instance with different number of threads.

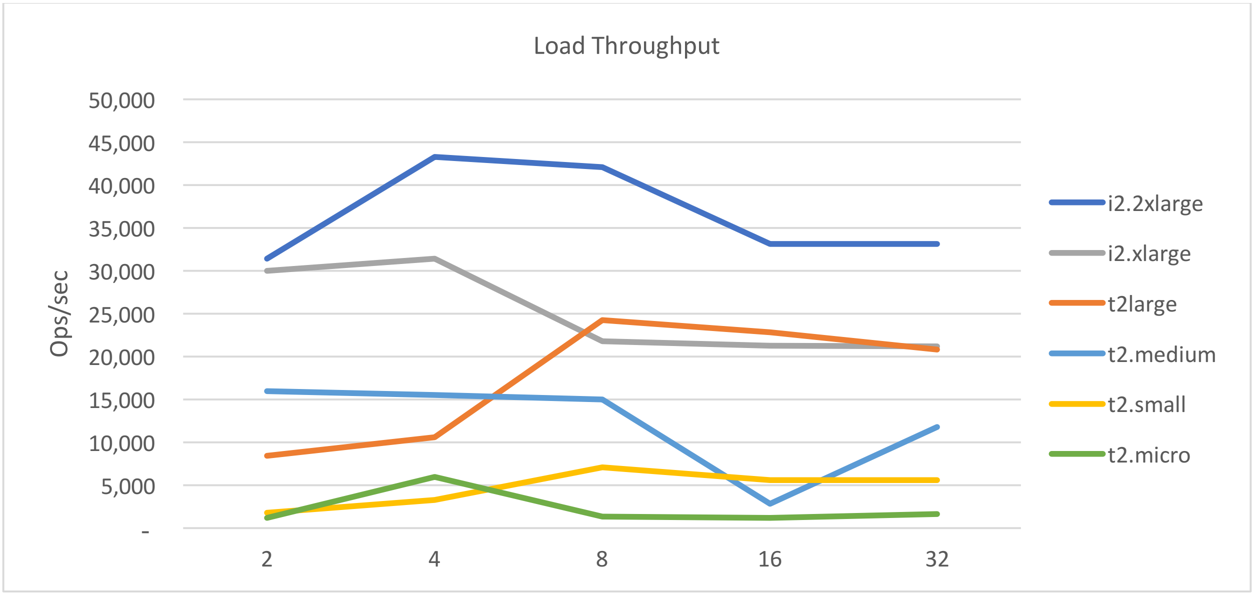

FIGURE 1. LOAD THROUGHPUT

In general, we see a greater throughput on the higher performance instances. The higher performance instances (i2.2xlarge, i2.xlarge and t2.large) provided higher throughput on the higher number of threads. This guarantees the instance's resource capacity is being used more as the number of threads increase.

Figure 2 shows the average insert latency that was obtained in the multiple loads done.

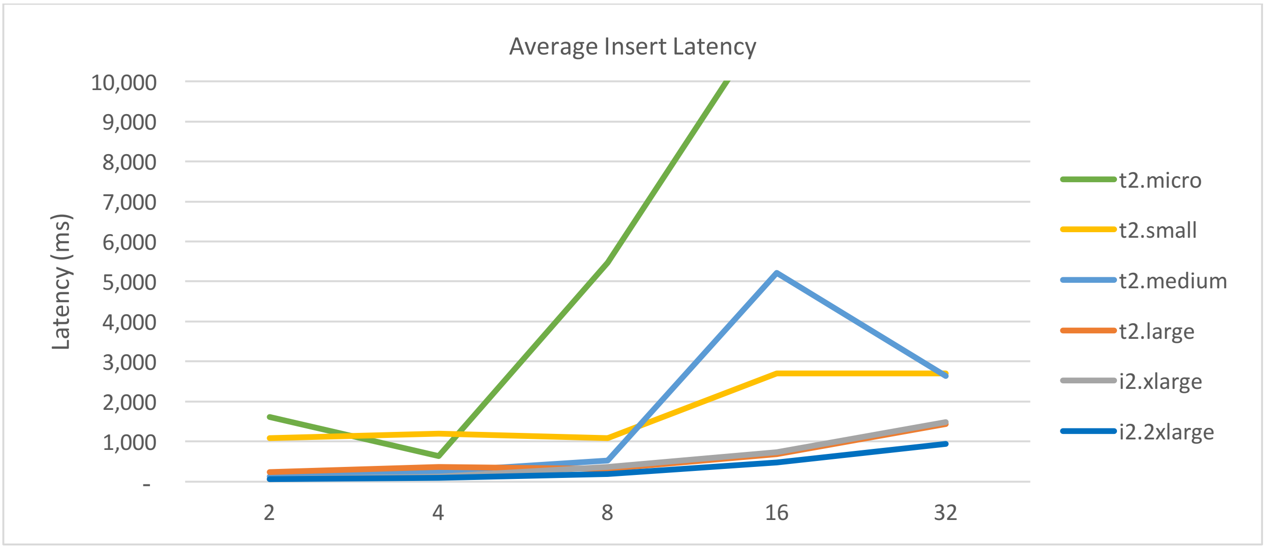

FIGURE 2. AVERAGE INSERT LATENCY

As expected, as more connections are, the latency increases. In general, instances with higher resources have lower latency as the threads increase in comparison to instances with lower resources. In particular, t2.large, i2.xlarge and i2.2xlarge besides having the lower latency of all instances, their latency doesn't change as abruptly as the others.

3.3 Transaction phase

Running the workload, we check the read and update operations performance. Figure 3 shows the throughput of each instance with different number of threads that was obtained during the transaction phase.

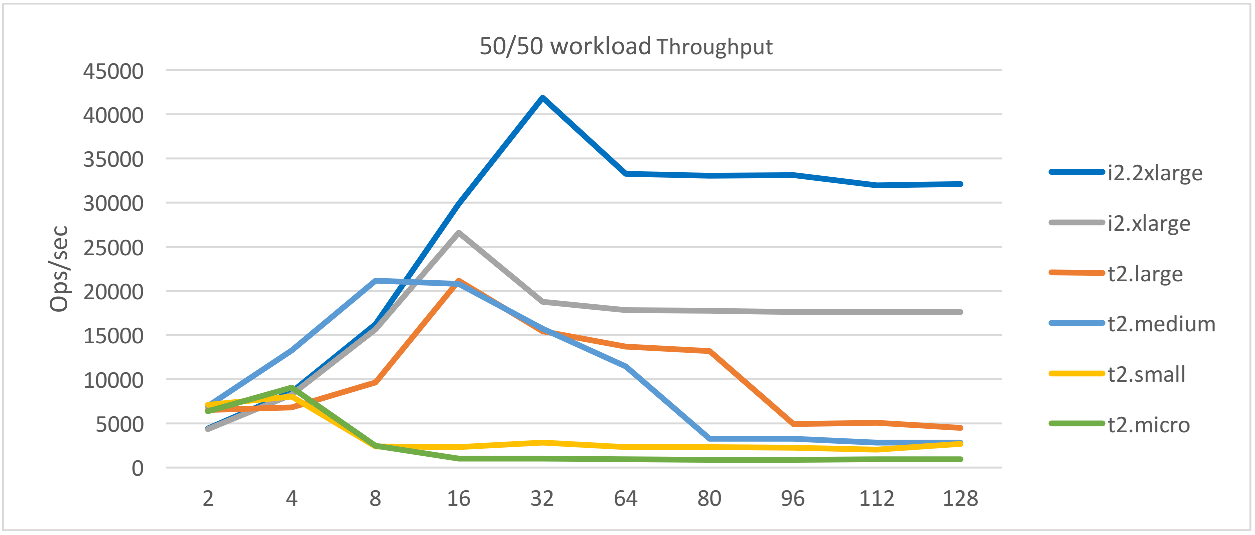

FIGURE 3. 50/50 WORKLOAD THROUGHPUT

Higher resource instances handle greater throughput as we can see in Figure 3. At first, as the number of threads goes up, the throughput increases till the processing power of the instance is reached; from there the throughput starts to decrease. We can see in the figure that instances with higher resources reach the throughput peak at higher number of threads.

Figure 4 shows the average read latency obtained during the transaction phase.

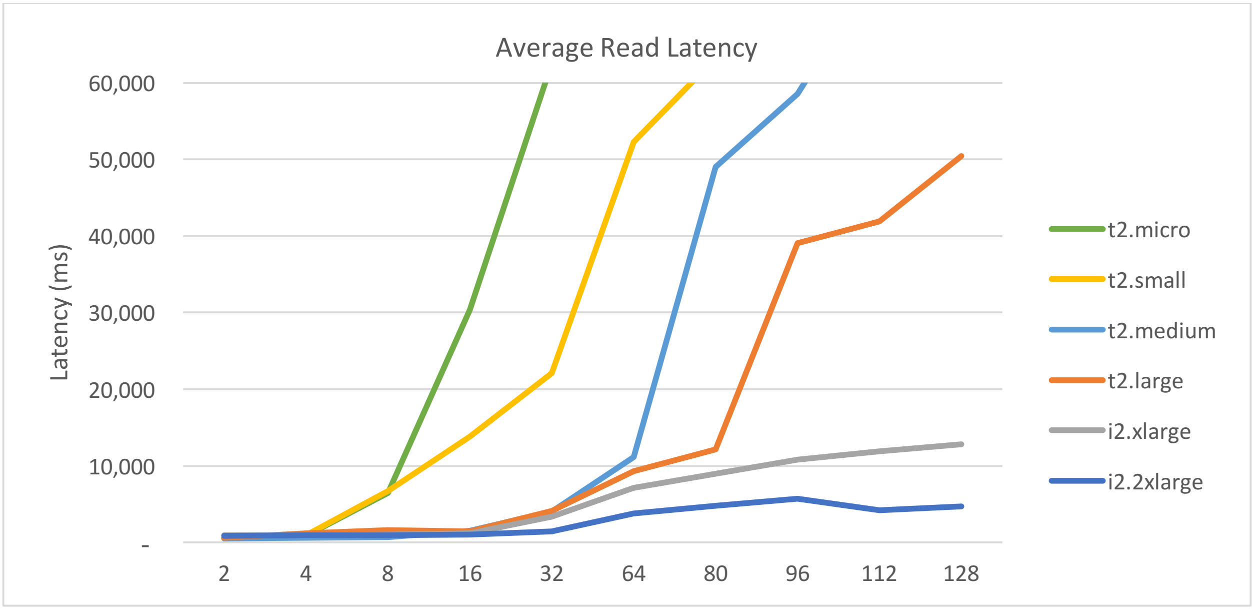

FIGURE 4. AVERAGE READ LATENCY

As expected, the higher the specifications the instance has, the lower the average read latency it gets.

3.4 Conclusion

Different situations, like the number of active connections (threads) and the way the operations are executed, determine the system performance. With the help of YCSB benchmark we can simulate different kind of configurations which enable us to corroborate the database is performing as intended on different types of instances. In this particular case we checked that higher performance instances performed better in the different metrics the benchmark provided.

4. Usage

4.1 Connecting to the shell

The mongo component provides a powerful interface for systems administrators as well as a way for developers to test queries and operations directly with the database. mongo also provides a fully functional JavaScript environment for use with a MongoDB.

To start the mongo shell from the Windows Server instance run the command shell as an administrator and enter the following command:

```
& "C:\Program Files\MongoDB\Server\3.2\bin\mongo.exe"
```

Since authentication is required to take action, you must first authenticate. Switch to the admin database and then use the db.auth() method to authenticate:

```
use admin

db.auth("admin", "e9ju0g" )
```

The user can connect from a Linux based instance by entering in the terminal the following command:

```
mongo [DNS of your instance]:27017 -u admin -p e9ju0g --
authenticationDatabase admin
```

4.2 Changing MongoDB password

The first thing the user must do is change the authorization password of the admin user. To do this, first the user must run the command shell as an administrator, then start the mongo shell and use the admin database:

```
use admin
```

Update the admin user with the password of your choice:

```
db.updateUser(
   "admin",
   {
      pwd: "myNewPassword"
   }
)
```

4.3 Migrate from a MongoDB version to another in the same instance

A series of recommendations must be considered before migrating to a new instance like ensuring the application and deployments are compatible with the version the user wants to migrate, or check if the current version can be updated to the desired version, if not, the user must change to a version that is applicable to the update.

Depending on the current version and the version the user wants to migrate, different procedures must be taken. All these procedures are in the MongoDB documentation. Here is a list of procedure examples:

Upgrade MongoDB to 3.2

Upgrade MongoDB to 3.0

4.4 Migrate the data from one instance type to another using the same AMI

One way to ensure the integrity of your MongoDB databases when migrating is to use the mongodump and mongorestore components. mongodump is a utility for creating a binary export of the contents of a database. It can be used to make backups of your data, syncing from production to staging or development environments, or changing the storage engine of a standalone[xiii]. The mongorestore program writes data from a binary database dump created by mongodump to a MongoDB instance. By making a backup of your database, copying the binary export to the new instance and restoring it with the mongorestore program the user can accomplish a migration ensuring the integrity of the data[xiv].

With the following command you create the binary export specifying the name of the database and the folder in which you wish to place the binary file:

```
& "C:\Program Files\MongoDB\Server\3.2\bin\mongodumb.exe" --db
myDatabase --out "C:\path\to\backup\folder"
```

Then, the binary folder generated by mongodumb is transferred to the new instance and restored with the following command, specifying the name of the database and the folder with the corresponding binaries:

```
& "C:\Program Files\MongoDB\Server\3.2\bin\mongorestore.exe" --db
ycsb --drop "C:\pathc\to\data\binaries\folder"
```

mongorestore performs inserts only and does not perform updates. That is, if restoring documents to an existing database and collection and existing documents have the same value _id field as the to-be-restored documents, mongorestore will not overwrite those documents.

4.5 Cluster the system

Detailed information about sharding and replication methods can be found in the MongoDB documentation:

Sharding: https://docs.mongodb.com/manual/sharding/

Replication: https://docs.mongodb.com/manual/replication/

5. References

[i] *"MongoDB Manual 3.2 mongod". MongoDB.*
[ii] *"MongoDB Manual 3.2 mongo". MongoDB.*
[iii] *"MongoDB Manual 3.2 Configuration File Options". MongoDB.*
[iv] *"SCRAM-SHA-1 — MongoDB Manual 3.2". MongoDB.*
[v] *"MongoDB Manual 3.2 Built-In Roles". MongoDB.*
[vi] *"MongoDB Manual 3.2 Default MongoDB Port". MongoDB.*
[vii] *"Additional Registry Entries". Microsoft.*
[viii] *"CIS Microsoft Windows Server 2012 R2 Benchmark". Center for Internet Security. v2.2.0 - 04-28-2016*
[ix] *"AWS Marketplace Seller Guide. Product and AMI Policies (pp 18)". Amazon Web Services Marketplace. v 3.3 October 31, 2016*
[x] *"Configuring a Windows Instance Using the EC2Config Service - Amazon Elastic Compute Cloud". Amazon Web Services.*
[xi] *"Create a Standard Amazon Machine Image Using Sysprep - Amazon Elastic Compute Cloud". Amazon Web Services.*
[xii] *"What is AWS CloudFormation? - AWS CloudFormation". Amazon Web Services.*
[xiii] *"MongoDB Manual 3.2 mongodump". MongoDB.*
[xiv] *"MongoDB Manual 3.2 mongorestore". MongoDB.*