Proceedings of the Informing Science + Information Technology Education Conference

An Official Publication of the Informing Science Institute
InformingScience.org

InformingScience.org/Publications

**Online July 6-7, 2022**

# Student Learning Using MongoDB Sharding on a Cluster of Ubuntu Raspberry Pi 4B Servers

| | | |
|---|---|---|
| Robert T. Mason* | Regis University, Denver, CO, USA | rmason@regis.edu |
| William Masters | Regis University, Denver, CO, USA | wmasters@regis.edu |

*Corresponding Author

## Abstract

| | |
|---|---|
| Aim/Purpose | When Data Science students use a Cloud environment such as AWS or Azure, they are not able to have direct hands-on experience with the underlying hardware components. When students create virtual machines in the Cloud, they specify memory size, CPUs, disk space, etc. However, they cannot reach out and touch the underlying hardware directly since it resides in the Cloud. |
| Background | The ability to purchase commodity servers (e.g., $3,000 per Dell server) to create a cluster of multiple machines is cost prohibitive for most faculty and students because it can cost upwards of $30,000 for 10 machines. This cost does not include the other hardware components that are required for the cluster, such as cooling equipment, cables, rack, etc. |
| Methodology | The research methodology leveraged for this research was to build a prototype to evaluate the costs of using inexpensive hardware and software ($1628.82) in comparison to a more expensive cluster of commodity servers ($30,000). |
| Contribution | There is very little research literature about using this approach of using Raspberry Pi servers as an inexpensive replacement for commodity servers. |
| Findings | This paper demonstrates that Raspberry Pi 4b servers (with 8 gig of RAM) can be leveraged to build a cluster of low cost servers to run both Linux Ubuntu 20 and MongoDB Sharding (distributed processing). |
| Recommendations for Practitioners | Practitioners will appreciate this paper because it is a tutorial that describes assembling the cluster components and then installing MongoDB Sharding (distributed processing) on a cluster of 9 Rpi 4b servers |

Accepted by executive review by Editor Eli Cohen │ Received: April 23, 2022 │ Revised: June 5, 2022 │ Accepted: June 6. 2022.
Cite as: Mason, R. T., & Masters, W. (2022). Student learning using MongoDB Sharding on a cluster of Ubuntu Raspberry Pi 4B servers. In M. Jones (Ed.), *Proceedings of InSITE 2022: Informing Science and Information Technology Education Conference*, Article 21. Informing Science Institute. https://doi.org/10.28945/4989

| | |
|---|---|
| Recommendations for Researchers | Researchers will appreciate this paper because it provides a new inexpensive alternative to using a Cloud environment or an expensive cluster of commodity servers to research distributed processing. |
| Impact on Society | Students and faculty now have an inexpensive option of creating a personalized cluster of servers to experiment with distributed processing. |
| Future Research | Future Research can include testing this cluster with other distributed processing tools, such as the Hadoop ecosystem or NoSQL Databases (e.g. such as Cassandra) |
| Keywords | Raspberry Pi 4b, distributed processing, MongoDB Sharding, Linux Ubuntu |

## INTRODUCTION

When Data Engineering students use a Cloud environment such as AWS or Azure, they are not able to have direct hands-on experience with the underlying hardware components. Until recently, the purchase of commodity servers ($3000 per server) to create a cluster of multiple machines was cost prohibitive for most students. As of May 2019, the Raspberry Pi Foundation (2021) has sold over 40 million Raspberry Pi computers to people around the world. A Raspberry Pi is a small single board computer that was developed in association with Broadcom (2021). Broadcom is a company that has been an innovator of technology for 50 years and has an impressive technical heritage with AT&T/Bell Labs, Lucent, and Hewlett-Packard. The Raspberry Pi Foundation has engaged millions of students for STEM education. Raspberry Pi(s) are an inexpensive way to give students a hands-on experience with both hardware and software. Student engagement has been shown to be higher when using Raspberry Pi(s) in comparison to the use of virtual machines for lab work (Hills et al., 2019). An inexpensive way for students to learn about distributed computing is to build a small cluster (e.g., network of servers) using Raspberry Pi(s) (Doucet & Zhang, 2017).

Distributed computing concepts are important for students to grasp since they are utilized across academic research, Cloud computing, and the technology industry. Distributed computing is the main driver for the Big Data movement that began in the 1990s and the supporting distributed data storage products, such as MongoDB, Hadoop, and other NoSQL databases. The goal of the research was to test the feasibility of running MongoDB databases with sharding (e.g., the of storing data across multiple servers based on a key; MongoDB, 2021) on a small cluster of nine Ubuntu Raspberry Pi servers. Sharding involves breaking data into smaller and easily managed pieces that can be distributed across many server machines which can significantly lower the cost of retrieving the data (GeeksforGeeks, 2022).

At the minimum, MongoDB 4.2 requires six servers: three configuration servers, two shard data servers, and a query router server. For this test configuration, five shard data servers were allocated in addition to the four other required servers. Adding additional shards to a cluster is an easy and straight forward process using the addShard() command. Theoretically, this cluster could have supported thousands of more shard servers based on the MongoDB "Limits and 42 Thresholds" (MongoDB, 2022) documentation; however, after reviewing various MongoDB Software Developer Blogs, the realistic limit is probably in the neighborhood of hundreds of additional shard servers. Either way, many more data shards could have been added to the cluster.

## RESEARCH METHODOLOGY

The research methodology leveraged for this research was to build a prototype to evaluate the costs of using inexpensive hardware and software ($1628.82) in comparison to a more expensive cluster of commodity servers ($30,000). This prototype approach was similar to the work by Shafter and Rixner (2007) at Rice University who built a Gigabit Ethernet Network to accurately evaluate new ideas in network server architecture and to use it for education and experimental research. The main research goal was to assess if sophisticated software (Linux Ubuntu 20 and a NoSQL MongoDB 4.2 database) could be run on lower cost Rpi hardware. If feasible, this would allow faculty and students the opportunity to construct their own personal inexpensive hardware and software platform for the testing of distributed computing. If successful, this platform could then be leveraged to test other distributed processing applications such as Hadoop or the Cassandra NoSQL database.

## BACKGROUND

In 2019, the Raspberry Pi Foundation released the Raspberry Pi (Rpi) 4B board that included many significant upgrades to the components. The Rpi 4B is a 64-bit fully functional server with a quad-core Broadcom BCM2711 chip that is capable of running the latest development version of the Ubuntu (20) Linux Operating System (OS). The Rpi with 8Gb of RAM costs $99 on Amazon (2021) excluding the power adapter. "The Pi 4B is a *serious* NAS (Network Attached Storage) contender. Sustained write speeds of over 68 MB/s were obtained, and over 105 MB/s for reading, including saturation of the Gigabit network. Yes, the Pi 4B can push even a 1000 MB/s network to 100%" (Unix Etc., 2019). The Rpi 4B release also offered the capacity of running MongoDB 4.2 on top of Ubuntu Linux OS. MongoDB 4.2 is the version of the NoSQL Database that supports MongoDB sharding (Distributed Processing). Sharding allows for the deployment of very large data sets with very high throughput. Shard servers scale linearly because data is managed by each server independently, and therefore they can provide faster performance with large data sets. As mentioned above, the goal of the research was to test the feasibility of running MongoDB databases that would use sharding on a small cluster of nine Ubuntu Rpi 4B servers.

## DATA ENGINEERING PRACTICUM

The Anderson College of Business and Computing (ACBC), located in Denver, CO, is a part of Regis University (RU) that has been providing Jesuit education since 1877. The Master of Science (MS) in Data Science program offers graduate students a Specialization in Data Engineering (DE). Data Engineering is an area of Data Science that focuses on building the infrastructure to support complicated analytics used by Data Scientists. The DE practicum provides an opportunity for students to get hands-on experience with technology to build a data infrastructure. Students in the DE specialization learn about Big Data Architecture and gain experience using MongoDB. In the Spring of 2021, three of the DE practicum students decided to use Rpi servers for their research projects. These student projects generated an idea with the DE faculty to test MongoDB sharding on small cluster of Rpi 4B servers.

## CLUSTER CONFIGURATION

Figure 1 shows the physical architecture of the cluster build out. This closely resembles a logical architecture of the Mongodb Query Router, Config server, and Shard server components. Putting each software component on a physical device allows students to experiment with fault tolerance, data redundancy, and networking in a very hands-on and visual way. Multiple shard servers allow Data Engineering students to experiment with the concept of horizontal scaling to understand better the tradeoffs involved. Mongodb deployments are sensitive to the amount of CPU capacity available on each server, network I/O, I/O to disk, and the selection of an appropriate shard key. Since read/writes go through the query router to ensure data integrity, once the platform is up and running, students can experiment with performance scaling.

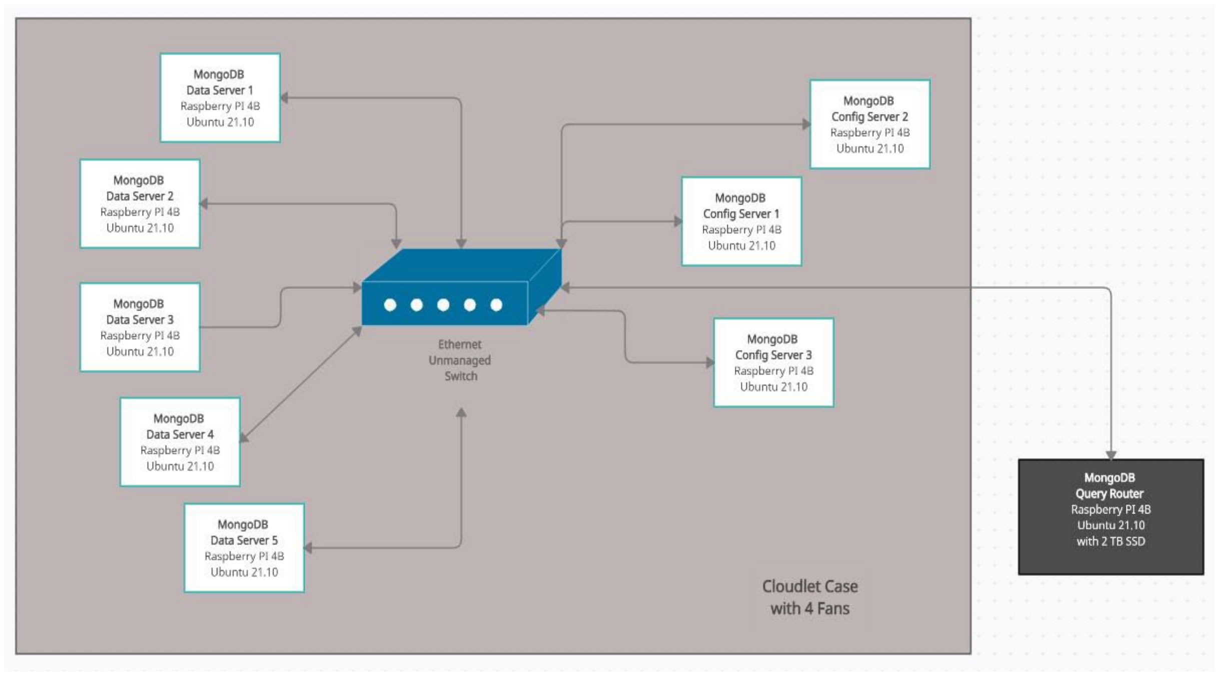

**Figure 1: Rpi Model 4B Cluster Architecture**

Ubuntu 20 OS was installed on nine Rpi 4B servers. The microSD cards hold the Ubuntu Linux OS and are used for booting the servers and storing data. A Cloudlet Cluster Case (similar to a rack for commodity servers in a large data center) with 4 cooling fans was used to hold 8 of the Rpi 4Bs. An inexpensive unmanaged switch was then connected to the Rpi 4Bs with 1 ft ethernet cables. An additional Rpi 4B was paired (using a kit) with a 2 Terabyte solid-state drive (SSD) to provide additional storage for the cluster. Since each Rpi 4B only has the storage available on microSD card of 32 gb, an additional 8 flash drives (128 gb in size) were plugged into each of the USB ports on the Rpi 4Bs. Using one USB port left three unused USB ports on each of the Rpi 4Bs. Shown in Table 1 is a list of the hardware components for the cluster and the overall cost of the cluster was $1628.82.

**Table 1: Hardware Components**

| Quantity | Hardware Components |
|---|---|
| 9 | Rpi 4B with 8 gb RAM - 64-bit with quad-core Broadcom BCM2711 SoC - (9 x $99 = $891) |
| 9 | Rpi Power Adapters - (9 x $8 = $72) |
| 1 | Cloudlet Case with 8 slots and 4 cooling fans (to hold the Rpi 4Bs) – ($59.99) |
| 1 | Solid State Drive (SSD) - 2 Terabyte in size - ($194.99) |
| 1 | SSD Kit for Rpi, includes one cooling fan and case – ($77.99) |
| 1 | Unmanaged Switch with 8 ports (note: one of the 4B servers used WiFi) – ($21.99) |
| 8 | Ethernet Cables - 1 ft length (to plug the Rpi 4Bs into the switch) – ($17.01) |
| 1 | Ethernet Cable 3 ft. length (to plug the switch into the Comcast router) – ($1.04) |
| 9 | Rpi 32 gb microSD cards formatted with Ubuntu 20 – ($8.99 x 9 = $80.91) |
| 8 | Flash Drives with 128 gb of storage – (8 x $18.99 = 151.92) |
| 2 | Power Plugs – (surge protector cords to plug in the power adapters) – (2 x 29.99) |

MongoDB 4.2 server security procedures were followed as suggested by the documentation (Smith, 2020). This involved changing the root password and the default pi user ID/password on the Rpi 4Bs. Also, an alternate port number for connecting to the servers was established. Secure Shell (SSH) was used to connect with and manage the Rpi 4Bs across the cluster. Prior to using the data shards for distributed processing, three types of servers had to be setup: Config Servers (CS), one Query Router (Mongos) and Data Shards (Linode, 2021). The CS stores metadata about the shards and the configuration settings for the cluster. At least one CS is required to test sharding, however as shown in Figure 1, three CSs were allocated for this project (shown on the right side of Figure 1). One CS is designated as the primary and the other two servers are secondary backup servers. If the primary CS fails, the secondary servers can take over control of the sharding process.

Another Rpi 4B server was designated as the Query Router (QR) that acts as an interface between a client application and the cluster shards. Client application queries need to be sent to the appropriate shard where the data will be stored based upon the primary key of the document. This task is accomplished by the QR (Mongos) in conjunction with the CS. For a production environment, multiple QRs can be established, however for this test, only one Rpi 4B server was allocated as a QR. Finally, as shown in Figure 1, five of the Rpi 4Bs were allocated as data shard servers (DSS). The DSS are fundamentally database servers that hold a portion of the data. MongoDB documents can be distributed to the DSS by either a range function or a hash value. For this research, the primary key of the collection was hashed (using a hash algorithm) to distribute the data across the five DSS. Each of the three server types required different mongoDB configuration files. Although the documentation from Linode (2021) was fairly accurate, some of the parameters did not work or were out-of-date for this configuration. Therefore, it required the researcher to experiment with the parameters within the MongoDB configuration files on each Rpi 4B to make the sharding function properly. Each of the revised MongoDB configuration files are shown below with a brief explanation of the parameters.

All of the Rpi 4Bs had a host file named ***hosts*** located in /etc directory. The hosts file enables the network to be aware of the other servers within the cluster. Although the cluster resided on a private network, for the sake of added security, the last two or three values from the IP address listed below have been altered from integers to question marks (?) for the sake of this publication. Also, the Rpi 4b server names have been changed to protect the identity of the cluster servers. The beta1 IP address and name is different on each of the servers. This sample hosts file was copied from the beta1 server. The IP addresses were automatically assigned by the Comcast router when the switch was connected to the router. The unique name for each server was assigned when each Rpi was configured by the researcher.

**Table 2: Hosts File**

| IP Address | Server Name |
|---|---|
| 127.0.0.1 | Localhost |
| 127.0.1.1 | beta1 |
| 10.0.0.?? | beta6 |
| 10.0.0.?? | beta7 |
| 10.0.0.? | beta8 |
| 10.0.0.??? | betadisk1 (2 Terabyte Drive) |
| 10.0.0.?? | beta2 |
| 10.0.0.??? | beta3 |
| 10.0.0.??? | beta4 |
| 10.0.0.??? | Beta5 |

## *Config Server Configuration*

Below is an example of the MongoDB configuration file for a Config Server (CS). Most of the parameters are standard, however there are new entries for replication and sharding. The name of the replication set is **myrs** (my replication set) and cluster role is defined as **configsvr**. This last parameter is how MongoDB knows that this node is a CS versus some other type of server. Note: Some of the integers below have been changed to question marks (?) to protect the cluster. The security key file is omitted from all of the configuration files because it was not used to test the cluster.

```
# Where and how to store data.
storage:
  dbPath: /data/db
  journal:
    enabled: true
#  engine:
#  mmapv1:
#  wiredTiger:

# Where to write logging data.
systemLog:
  destination: file
  logAppend: true
  path: /var/log/mongodb/mongod.log

# network interfaces
net:
  port: 270??
  bindIp: 0.0.0.0,<10.0.0.??>     #IP of this machine

# How the process runs
processManagement:
  timeZoneInfo: /usr/share/zoneinfo

#operationProfiling:

replication:
    replSetName: <myrs>

sharding:
    clusterRole: configsvr
```

After the installation of the CS parameters is successful, the server can be tested by restarting mongod on the CS and then issuing a status command (shown below).

```
sudo systemctl status mongod
[sudo] password:

mongod.service - MongoDB Database Server
     Loaded: loaded (/lib/systemd/system/mongod.service; enabled; vendor preset>
     Active: active (running) since Fri 2021-12-03 21:21:52 MST; 2 days ago
       Docs: https://docs.mongodb.org/manual
   Main PID: 16407 (mongod)
     Memory: 272.9M
     CGroup: /system.slice/mongod.service
             └─16407 /usr/bin/mongod --config /etc/mongod.conf
Dec 03 21:21:52 beta1 systemd[1]: Started MongoDB Database Server.
```

Start the mongo client (mongo) on the CS. If the prompt shows as SECONDARY, then SSH to another CS and repeat the process. On the primary CS, start the client and then run the rs.intiate command to see that the three CS(s) are running.

```
rs.initiate( { _id: "<myrs>", configsvr: true, members: [ { _id: 0, host: "10.0.0.??:270??" }, { _id: 1,
host: "10.0.0.??:270??" }, { _id: 2, host: "10.0.0.???:270??" } ] } )
```

Then, check that the CSs are operational by starting the mongo client (mongo) on a (CS). It will show the following prompt -<myrs>:PRIMARY>. If the prompt shows <myrs>: SECONDARY, then SSH to another CS and repeat the same process until you find the primary CS. The results from the rs.status() are not shown since they are verbose.

```
<myrs>:PRIMARY> rs.status()
```

## *Query Router Configuration*

Below is an example of the configuration file (mongos.conf) for the Query Router (QR) server that is located in the /etc directory. The QR had to be setup to use mongos (not the Mongo client application that is already installed on the QR). Additional information is available on MongoDB website (MongoDB, 2021a) regarding Mongos and how it functions as a query router. Note: Mongos is usually started remotely from another machine in the cluster, such as one of the data shard servers. A new service configuration file is required for the mongos installation. Notice the sharding entry at the bottom of the file. The myrs indicates the config server name and the three IP addresses are for the three CSs.

```
# where to write logging data.
systemLog:
  destination: file
  logAppend: true
  path: /var/log/mongodb/mongos.log

# network interfaces
net:
  port: 270??
  bindIp: 0.0.0.0

sharding:
  configDB: <myrs>/10.0.0.??:270??,10.0.0.??:270??,10.0.0.???:270??
```

Below is an example of the service file that is also required on the Query Router server.

```
-------------------------------------------------------------------
----- mongos service file -/lib/systemd/system/mongos.service file
-------------------------------------------------------------------
[Unit]
Description=Mongo-Cluster-Router
After=network.target
```

```
[Service]
User=mongodb
Group=mongodb
ExecStart=/usr/bin/mongos --config /etc/mongos.conf shown above
# file size
LimitFSIZE=infinity
# cpu time
LimitCPU=infinity
# virtual memory size
LimitAS=infinity
# open files
LimitNOFILE=64000
# processes/threads
LimitNPROC=64000
# locked memory
LimitMEMLOCK=infinity
# total threads (user+kernel)
TasksMax=infinity
TasksAccounting=false
# Recommended limits for mongod as specified in
# https://docs.mongodb.com/manual/reference/ulimit/#recommended-ulimit-settings
[Install]
WantedBy=multi-user.target
```

When the installation of mongos on the Query Router is successful, it can be tested by starting mongos (from a remote server in the cluster) and then issuing a status command on the QR (shown below).

```
sudo systemctl status mongos
[sudo] password:

mongos.service - Mongo-Cluster-Router
     Loaded: loaded (/lib/systemd/system/mongos.service; enabled; vendor preset>
     Active: active (running) since Fri 2021-12-03 20:55:21 MST; 2 days ago
   Main PID: 3076 (mongos)
     Memory: 38.1M
     CGroup: /system.slice/mongos.service
             └─3076 /usr/bin/mongos --config /etc/mongos.conf

Dec 03 20:55:21 betadisk1 systemd[1]: Started Mongo-Cluster-Router.
lines 1-9/9 (END)
```

## *SHARD SERVER CONFIGURATION*

Lastly, the configuration file for the shard data servers is shown below. Most of the parameters are standard; however, note that the replication set is not included in this file, although the Linode (2021) documentation recommended it. The addition of the replication set name in this file resulted in a failure for sharding in the cluster. The cluster role is simply set to shardsvr.

```
 # Where and how to store data.
storage:
  dbPath: /data/db
 journal:
   enabled: true
#  engine:
#  mmapv1:
#  wiredTiger:

# where to write logging data.
systemLog:
  destination: file
  logAppend: true
  path: /var/log/mongodb/mongod.log

# network interfaces
net:
  port: 270??
  bindIp: 0.0.0.0

# how the process runs
processManagement:
  timeZoneInfo: /usr/share/zoneinfo

# notice that name of the replication set name is not included intentionally
sharding:
   clusterRole: shardsvr
```

After the shard servers are configured and restarted, their server status can be verified with a status check of the database server - mongod (as shown below).

```
sudo systemctl status mongod
[sudo] password:

 mongod.service - MongoDB Database Server
     Loaded: loaded (/lib/systemd/system/mongod.service; enabled; vendor preset>
     Active: active (running) since Sat 2021-11-27 16:01:10 MST; 1 weeks 1 days>
       Docs: https://docs.mongodb.org/manual
   Main PID: 1202 (mongod)
     Memory: 233.7M
     CGroup: /system.slice/mongod.service
             └─1202 /usr/bin/mongod --config /etc/mongod.conf

Nov 27 16:01:10 delta6 systemd[1]: Started MongoDB Database Server.
```

At this point the three types of servers have been configured and are operational.

## *Remaining Steps to Enable the Testing*

There are a few remaining steps to finish the setup for testing. Create a database (shardDB) and then create a collection (cryptoHistory) using mongos. The cryptocurrency data for testing the sharding was sourced from Kaggle (2021), which is a popular data source for Data Science projects. A collection in MongoDB is similar to a table in a relational database (e.g., Oracle, MySQL, MS SQL Server). The use command shown below will create the database if it does not already exist.

```
mongos> use shardDB        --create a database called shardDB using mongos
```

Add the five shards using mongos

```
mongos> sh.addShard( "shard000?:270??" )

mongos> db.createCollection("cryptoHistory")
{
        "ok" : 1,
        "operationTime" : Timestamp(1638649074, 8),
        "$clusterTime" : {
                "clusterTime" : Timestamp(1638649074, 8),
                "signature" : {    "hash" :
BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)  } } }

mongos> show collections
cryptoHistory

mongos> db.cryptoHistory.ensureIndex( { _id : "hashed" } )    -- create hashed Index on _id
```

Enabled on the shardDB database.

```
mongos> sh.enableSharding("shardDB")  -- enable sharding
{
        "ok" : 1,
        "operationTime" : Timestamp(1638739986, 3),
        "$clusterTime" : {
                "clusterTime" : Timestamp(1638739986, 3),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0) } } }
```

### Test data

Mongos was used to insert 1500 cryptocurrency documents that were downloaded from the Kaggle website and tweaked. Samples of two insert statements are shown below. Note: The MongoImport load utility was tested, however it did not work with sharding and the QR (mongos). Therefore, more research is needed for MongoImport in terms of loading data into shards.

```
--------- sample of 2 crypto Bitcoin documents insert statements downloaded from Kaggle CSV

db.cryptoHistory.insert( {  _id: 20001, Date: "2010-07-18", Price: 0.1, Open:0, High: 0.1, Low: 0.1,
Volume:80, Change  : 0})

db.cryptoHistory.insert( {  _id: 20002, Date: "2010-07-19", Price: 0.1, Open:0.1, High: 0.1, Low: 0.1,
Volume:570, Change  : 0})
```

## RESULTS AND ANALYSIS

Using the getShardDistribution() command shown below, the results indicate that the 1500 documents (rows) were distributed across the five partitions based upon the hashed primary key - key _id.

This configuration could accommodate many more shard servers and much more data as mentioned above.

```
mongos> db.cryptoHistory.getShardDistribution()

Shard shard0000 at 10.0.0.???:270??
 data : 16KiB docs : 271 chunks : 2
 estimated data per chunk : 8KiB
 estimated docs per chunk : 135
…

Totals
 data : 94KiB   docs : 1500   chunks : 10
 Shard shard0000 contains 17.74% data, 18.06% docs in cluster, avg obj size on shard : 63B
 Shard shard0001 contains 21.26% data, 21.33% docs in cluster, avg obj size on shard : 64B
 Shard shard0004 contains 19.95% data, 20.86% docs in cluster, avg obj size on shard : 61B
 Shard shard0003 contains 20.23% data, 19.13% docs in cluster, avg obj size on shard : 68B
 Shard shard0002 contains 20.79% data, 20.6% docs in cluster, avg obj size on shard : 64B
```

The goal of the research was to test the feasibility of running MongoDB databases with sharding on a small cluster of nine Ubuntu Rpi servers. Once configured, the cluster performed as expected loading 1500 documents to the five MongoDB DSS within a few seconds. The totals (above) provide evidence that the data was distributed evenly across the five shard servers based on a hashed key. Shard0000 had the smallest amount of data with 17.74% and shard0001 had the largest amount of data with 21.26%. All of the shard servers received roughly 20% of the data. Future research could include testing with large amounts of data and also testing distributed processing using a Cassandra NoSQL database.

## CONCLUSION

In conclusion, this research study demonstrated that a small cluster of nine inexpensive Rpi servers could be used by students to learn about distributed processing. This research is significant to the academic community because students often run into limited financial constraints when attempting to build projects. The use of the Ubuntu Linux OS and MongoDB, which are both open-source products, helped to reduce the overall expenses for this project. As mentioned by Hills et al. (2019), students are often more enthusiastic when they are able to conduct hands-on experimentation with hardware and software. It is hoped that future work in this area will inspire students to experiment with NoSQL databases by working on performance scaling, deployment automation, Edge processing and observability.

## REFERENCES

Amazon. (2021). *raspberry pi 4 8gb*. Retrieved February 4, 2022, from https://www.amazon.com/raspberry-pi-4-8gb/s?k=raspberry+pi+4+8gb

Broadcom. (2021). *Company history*. https://www.broadcom.com/company/about-us/company-history

Doucet, K. & Zhang, J. (2017). Learning cluster computing by creating a Raspberry Pi cluster. ACM SE '17: *Proceedings of the SouthEast Conference*, April 13-15, 2017, Kennesaw, GA, USA. (pp. 191–194). https://doi.org/10.1145/3077286.3077324

GeeksforGeeks. (2022). *What is sharding?* https://www.geeksforgeeks.org/what-is-sharding/

Hills, M, Gamrat, C. & Brown, C. (2019). Classroom engagement activities with the Raspberry Pi. *In SIGITE '19: Proceedings of the 20th Annual SIG Conference on Information Technology Education* September 2019. https://doi.org/10.1145/3349266.3351358

Kaggle. (2021). *Datasets*. https://www.kaggle.com/datasets

Linode. (2021). *How to build database clusters with MongoDB*. https://www.linode.com/docs/guides/build-database-clusters-with-mongodb

MongoDB (2021a). *Mongos*. https://docs.mongodb.com/v4.2/core/sharded-cluster-428 query-router/

MongoDB (2021b). *Sharding*. https://docs.mongodb.com/manual/sharding/

MongoDB. (2022). *MongoDB Limits and Thresholds*. https://docs.mongodb.com/v4.2/reference/limits/

Raspberry Pi Foundation. (2021). *About us*. https://www.raspberrypi.org/about/

Shafer, J., & Rixner, S. (2007). RiceNIC: A reconfigurable network interface for experimental research and education. *Experimental Computer Science Conference* (ExpCS, June 2007), San Diego, CA, ACM 978-1-59593-751-3 https://doi.org/10.1145/1281700.1281721

Smith, M. (2020). *Install & configure MongoDB on the Raspberry Pi*. https://www.mongodb.com/developer/how-to/mongodb-on-raspberry-pi/

Unix etc. (2019). *Raspberry Pi 4 Real World Tests*. http://unixetc.co.uk/2019/07/07/raspberry-pi-4-real-world-tests/

## AUTHORS

**Robert T. Mason**. I joined Regis University as a ranked full-time faculty member in January 2011. I am a professor in the Anderson College of Business and Computing. I am now the program director for the MS in Information Systems (MSIS). Prior to accepting this position with RU as a faculty member, I was employed by various Fortune 500 companies for 25 years as a Data Architect, DBA, Manager and Software Engineer.

**William Masters**. I am an experienced team lead, architect, and developer who excels in the rapid creation of products and services using engineering best practices such as Agile, LEAN, DevOps.