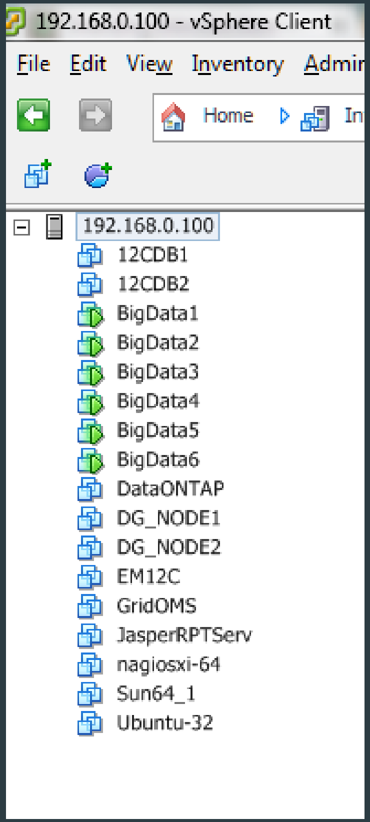
Configuring and Deploying mongoDB Sharded Cluster in 30 minutes

Prepared by Sudheer Kondla

Solutions Architect

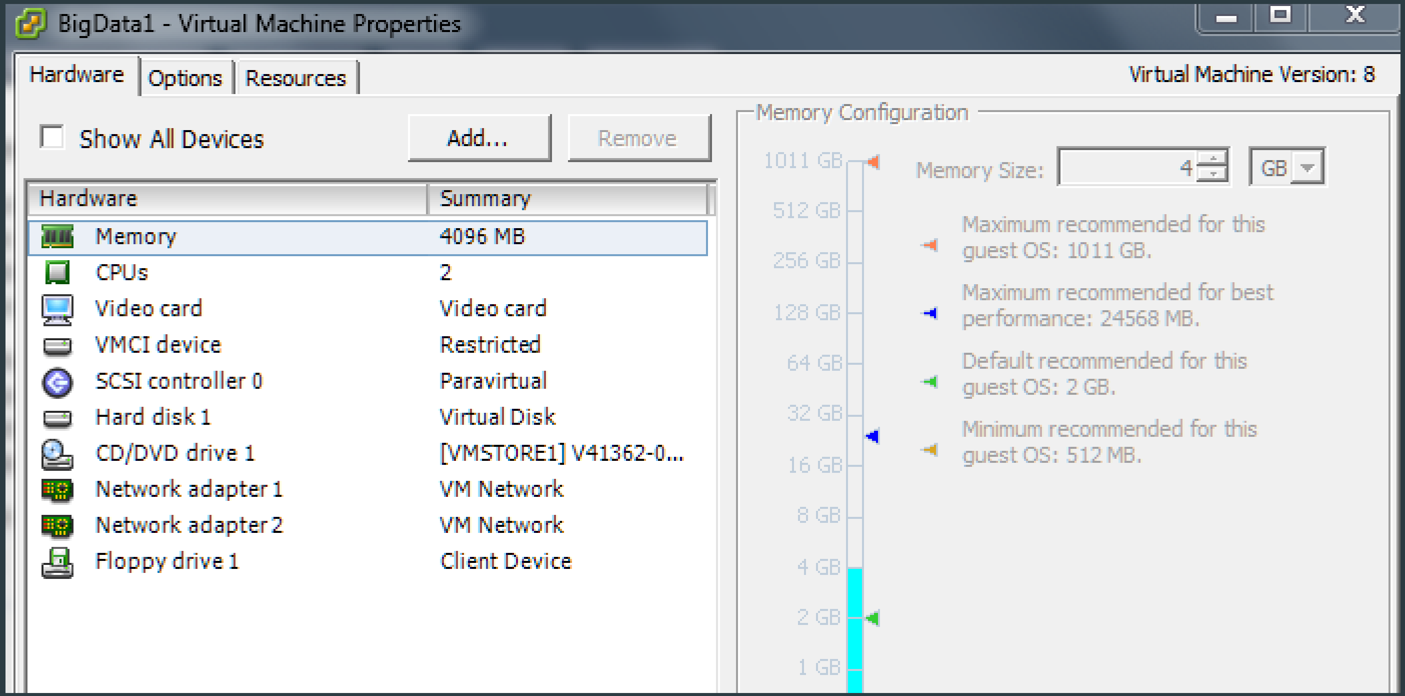
- The presentation is based on VMs created using VMWare’s ESXi/Vsphere and RHEL/Oracle Linux.
- This presentation is based on setting up mongoDB sharded cluster
  - with 6 VMs in the MongoDB cluster (RedHat Linux).
  - Each VM consists of 2 vCPUs and 4 GB of RAM
  - Each VM is created with 80 GB of disk space. No special mounts/ file systems are used.
  - Linux version used: 6.5

Sharded Cluster VIRTUAL MACHINES


192.168.0.100 - vSphere Client
File Edit View Inventory Admir
Home
In

192.168.0.100
12CDB1
12CDB2
BigData1
BigData2
BigData3
BigData4
BigData5
BigData6
DataONTAP
DG_NODE1
DG_NODE2
EM12C
GridOMS
JasperRPTServ
nagiosxi-64
Sun64_1
Ubuntu-32

VM Configuration



BigData1 - Virtual Machine Properties

Hardware | Options | Resources

Virtual Machine Version: 8

Show All Devices

Hardware	Summary
Memory	4096 MB
CPUs	2
Video card	Video card
VMCI device	Restricted
SCSI controller 0	Paravirtual
Hard disk 1	Virtual Disk
CD/DVD drive 1	[VMSTORE1] V41362-0...
Network adapter 1	VM Network
Network adapter 2	VM Network
Floppy drive 1	Client Device

Memory Configuration

Memory Size: 4 GB

Maximum recommended for this guest OS: 1011 GB.

Maximum recommended for best performance: 24568 MB.

Default recommended for this guest OS: 2 GB.

Minimum recommended for this guest OS: 512 MB.

Installing mongdb

- The following steps guide you through installing mongodb software on Linux.
- set yum repository and download packages using yum package installer.
- [root@bigdata1 ~]# cd /etc/yum.repos.d/
  - [root@bigdata1 yum.repos.d]# cat mongodb.repo

```
[mongodb]
name=MongoDB Repository
baseurl=http://downloads-distro.mongodb.org/repo/redhat/os/x86_64/
gpgcheck=0
enabled=1
```

- Before you run “yum install”, be sure to check internet is working.
- [root@bigdata6 yum.repos.d]# yum install -y mongodb-org-2.6.1 mongodb-org-server-2.6.1 mongodb-org-shell-2.6.1 mongodb-org-mongos-2.6.1 mongodb-org-tools-2.6.1

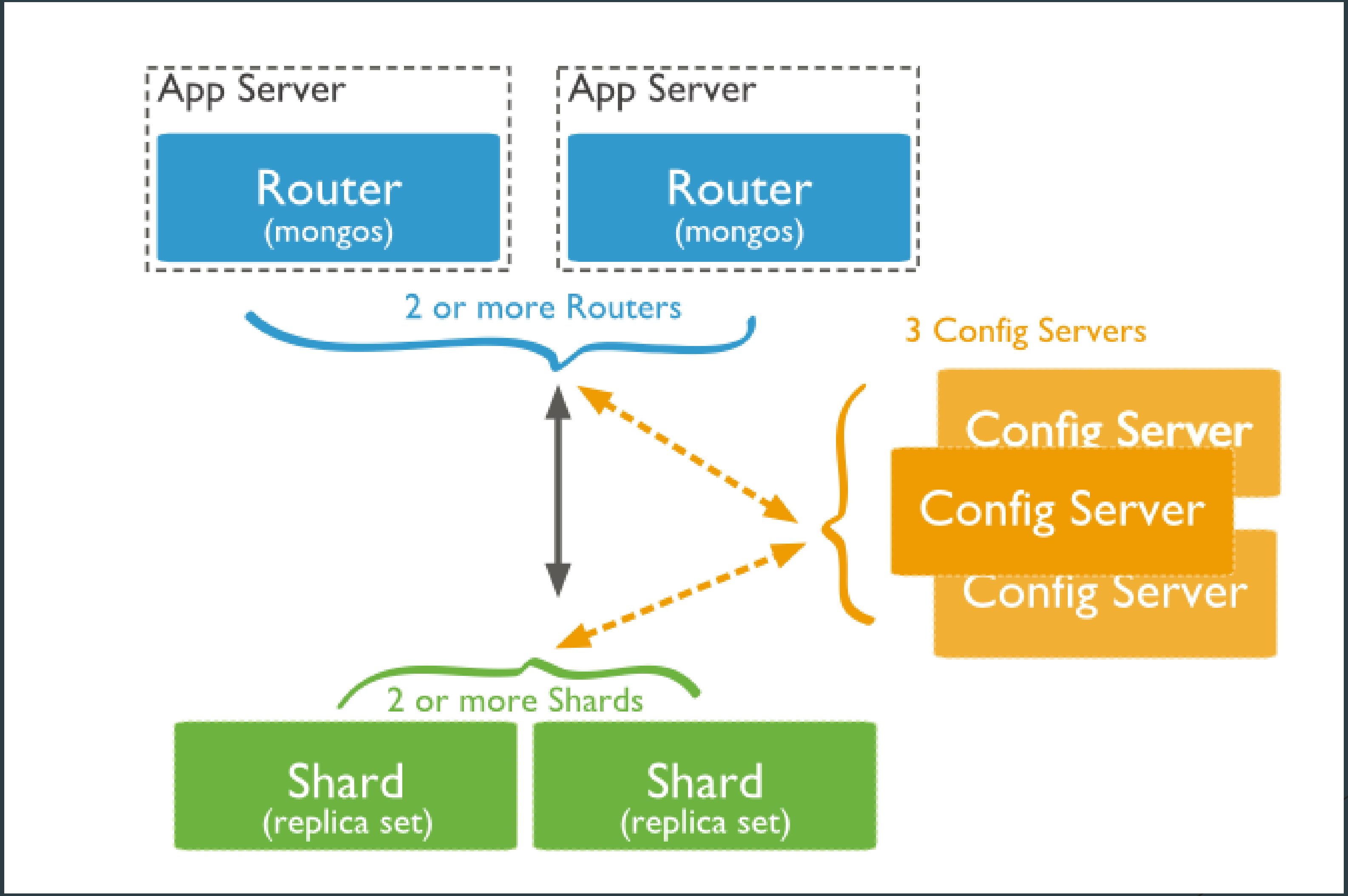
Setting up mongodb

- Make sure to create /data/configdb and /data/db directories on each servers
- The above directories should be owned by mongod user and mongod group.
- Change ownership to mongod
- As a root user run “chown mongod:mongod /data/configdb” and “chown mongod:mongod /data/db”
- Without above directories mongod process will not start
- When you start mongo daemon process it will create “/data/configdb/mongod.lock” file
- You can also start mongod process with service option as root. For example “service mongod start”
- You can also configure mongo daemon to start at system boot.

MongoDB Sharding Architecture

- Sharding is a method for storing data across multiple machines. MongoDB uses sharding to support deployments with very large data sets and high throughput operations.
- Database systems with large data sets and high throughput applications can challenge the capacity of a single server. High query rates can exhaust the CPU capacity of the server. Larger data sets exceed the storage capacity of a single machine. Finally, working set sizes larger than the system’s RAM stress the I/O capacity of disk drives.
- Sharding addresses the challenge of scaling to support high throughput and large data sets:
  - Sharding reduces the number of operations each shard handles. Each shard processes fewer operations as the cluster grows. As a result, a cluster can increase capacity and throughput horizontally.
  - For example, to insert data, the application only needs to access the shard responsible for that record.
  - Sharding reduces the amount of data that each server needs to store. Each shard stores less data as the cluster grows.
  - For example, if a database has a 1 terabyte data set, and there are 4 shards, then each shard might hold only 256GB of data. If there are 40 shards, then each shard might hold only 25GB of data.

Sharding Architecture



Deploying a sharded cluster

- Start the Config Server Database Instances.
  - The config server processes are mongod instances that store the cluster’s metadata.
  - Designate a mongod as a config server using the --configsvr option.

    e.g: mongod --configsvr --dbpath /data/configdb --port 27019 --bind_ip 192.168.0.131 –v
  - Each config server stores a complete copy of the cluster’s metadata.
  - Production deployments require three config server instances, each of them running on different servers to ensure high availability and data safety.
  - All members of a sharded cluster must be able to connect to all other members of a sharded cluster.
  - Make sure to setup proper network connectivity, firewall rules.
  - Start the three config server instances. For example
    - Node1: mongod --configsvr --dbpath /data/configdb --port 27019 --bind_ip 192.168.0.131 -v
    - Node2: mongod --configsvr --dbpath /data/configdb --port 27019 --bind_ip 192.168.0.132 -v
    - Node3: mongod --configsvr --dbpath /data/configdb --port 27019 --bind_ip 192.168.0.133 -v
- Start the mongos Instances:
  - The mongos instances are lightweight and do not require data directories.
  - You can run a mongos instance on a system that runs other cluster components, such as on an application server or a server running a mongod process.
  - By default, a mongos instance runs on port 27017. Specify the hostnames of the three config servers, either in the configuration file or as command line parameters.
  - For example: mongos --configdb bigdata1:27019,bigdata2:27019,bigdata3:27019

Add Shards to the Cluster

- Enable Sharding for a Database
  - Enabling sharding for a database does not redistribute data but make it possible to shard the collections in that database.
  - From a mongo shell, connect to the mongos instance. Issue a command using the following

```
[hdfs@bigdata3 ~]$ mongo --host bigdata2 --port 27017

MongoDB shell version: 2.6.1

connecting to: bigdata2:27017/test

rs0:PRIMARY>
```

    - sh.addShard("rs0/bigdata1:27017")
    - sh.addShard("rs0/bigdata2:27017")
    - sh.addShard("rs0/bigdata3:27017")
    - sh.addShard("rs0/bigdata4:27017")
    - sh.addShard("rs0/bigdata5:27017")
    - sh.addShard("rs0/bigdata6:27017")

Enable Sharding for a Collection

- Before you can shard a collection, you must enable sharding for the collection’s database.
- Enabling sharding for a database does not redistribute data but make it possible to shard the collections in that database.
- Once you enable sharding for a database, MongoDB assigns a primary shard for that database where MongoDB stores all data before sharding begins.
- Issue the sh.enableSharding() method.
  - rs0:PRIMARY> show dbs
  - sh.enableSharding(" customer”) OR
  - db.runCommand({ enableSharding: “customer” })

- Enable sharding on a per-collection basis.
  - sh.shardCollection("records.people", { "zipcode": 1, "name": 1 })
  - sh.shardCollection("people.addresses", { "state": 1, "_id": 1 })
  - sh.shardCollection("assets.chairs", { "type": 1, "_id": 1 })
  - sh.shardCollection("events.alerts", { "_id": "hashed" })