PERCONA

TRAINING

Basic Administration

http://www.percona.com/training/

Installing MongoDB

- MongoDB org or Percona repositories
- Packages available for most distros
- Percona distribution normally released within 1 month of upstream https://www.percona.com/services/policies/percona-software-support-lifecycle#mongodb

Installing MongoDB - Debian/Ubuntu

- Install the latest version

```
wget https://repo.percona.com/apt/percona-release_latest.$(lsb_release -sc)_all.deb
dpkg -i percona-release_latest.$(lsb_release -sc)_all.deb
apt-get install percona-server-mongodb
```

- Install a specific version

```
apt-cache madison percona-server-mongodb
apt-get install percona-server-mongodb=4.4.0-1.buster \
percona-server-mongodb-mongos=4.4.0-1.buster \
percona-server-mongodb-shell=4.4.0-1.buster \
percona-server-mongodb-server=4.4.0-1.buster \
percona-server-mongodb-tools=4.4.0-1.buster
```

Installing MongoDB - RHEL/Centos

- Install the latest version

```
yum install https://repo.percona.com/yum/percona-release-latest.noarch.rpm
yum install percona-server-mongodb
```

- Install a specific version

```
yum list percona-server-mongodb --showduplicates
yum install percona-server-mongodb-4.4.0-1.el8
```

Exercises - Setup access

1. Get the private key

```
https://github.com/percona/training-aws/blob/master/Percona-Training.key
```

2. Check the website to get your instance's IP address

```
http://percona-training.s3-website-us-east-1.amazonaws.com/?tag=XXXX
```

3. Connect as `centos` user

```
ssh -i Percona-Training.key centos@public-ip-of-your-server
```

Exercises - Percona release

1. Check installed packages

```
rpm -qa | grep percona
```

2. Check which repos are enabled

```
sudo percona-release show
```

Exercises - Prepare the environment

1. Create the datadir

```
sudo mkdir /var/lib/mongo
```

2. Give proper permissions

```
sudo chown mongod: /var/lib/mongo
```

3. Start the service

```
sudo service mongod start
```

Exercises - Default locations

1. Check Data files

```
ls /var/lib/mongo/
```

2. Configuration file

```
cat /etc/mongod.conf
```

3. Log files

```
ls /var/log/mongo/
sudo tail /var/log/mongo/mongod.log
```

Upgrading MongoDB

1. Stop mongod service
2. Remove the old packages
3. Install the new packages
4. Start mongod service

- Rolling approach to reduce downtime (more on this later)

Exercises - Upgrading

1. Stop the service

```
sudo service mongod stop
```

2. Ensure you have latest percona-release

```
sudo yum update percona-release
```

3. Enable the new version repository

```
sudo percona-release setup psmdb-44
```

4. Upgrade the packages

```
sudo yum update percona-server-mongodb
```

Exercises - Upgrading

5. Start the service

```
sudo service mongod start
```

6. Check the logs

```
sudo less /var/log/mongo/mongod.log
```

7. Enable 4.4 features that persist data incompatible with older versions

```
mongo
> db.adminCommand( { setFeatureCompatibilityVersion: "4.4" } )
exit
```

How Mongo Uses Memory

- Storage engine cache
  - Default 50% of RAM is dedicated to in-heap cache, the remainder can be used by the OS to cache filesystem blocks
- Client connections
  - Each connection allocates up to 1MB RAM for buffers
  - Control the max connections

How Mongo Uses Memory

- Storage Engine Caching
  - No "result" cache, only document and/or block caching
  - Compressed data cached at filesystem layer
  - Uncompressed data cached in WiredTiger cache

How Engines affect a system? (1)

- WiredTiger
  - Increased parallel execution (document vs collection level locks) and CPU utilization vs MMAPv1
  - More CPUs *may* scale throughput in some cases as long as workload is not concentrated on a hotspot
  - Compression adds to CPU usage vs MMAPv1/no compression

Basic Administration

STORAGE AND FILESYSTEMS

SSD vs SAS vs Flash

- MongoDB creates a relatively high amount of random write disk IO
- Non-spinning disks (SSD/Flash) are much better suited for MongoDB workloads due to high random IO efficiency
- The MongoDB Journal is an append-only log with sequential-only IO patterns

SSD vs SAS vs Flash (2)

- The data files should be on SSDs
- The journal is well suited for both spinning/non-spinning disks
- Hybrid low cost approach: SSD/Flash for MongoDB data, RAID 1 SAS/SATA for OS and MongoDB Journal

Network Attached Storage

- Not recommended for high throughput
- Fiber channel, iSCSI generally performs better
- Performance considerations
  - IO Path on Local Disk (simplified): Kernel -> Bus -> Controller -> Disks
  - IO Path on NFS for example, (simplified, *= translation): Kernel* -> Bus -> Network Card -> Cable -> Switch -> Cable -> Network Card -> Bus -> Kernel* -> Bus -> Controller -> Disks

File Systems

- XFS is recommended for MongoDB
- Use of EXT3 is highly discouraged due to poor allocation speed
- EXT4 is ok unless very high transactional rates
- Add `noatime` mount option for the DB data volume in `/etc/fstab`
- Best practice: dedicate a partition/LUN/volume for DB data

Basic Administration

VIRTUAL MEMORY

Transparent Huge Pages (THP)

- Introduced around 2.6.32 / el6 to optimize high RAM servers
- Not efficient with MongoDB and it's storage engines, disable it
- Add `transparent_hugepages=never` to bootloader and reboot
- If `AnonHugePages:` from `/proc/meminfo` is = 0, it's disabled

vm.swappiness

- Influences the likelihood for the kernel to swap to disk vs RAM
- A setting of `1` is recommended

Basic Administration

READ AHEAD

Read Ahead

- Pre-fetching of file pages when reading blocks from disk
- can waste RAM in some situations
- MongoDB works best with 32 sectors/16kb
  - A lower setting (8 or 16) may help reduce RAM consumption if you have extremely small documents
- Linux default is typically 256 sectors/128kb
- Use `blockdev --setra 32 <block dev>` to change
- Use `blockdev --getra <block dev>` to query

Persisting Read Ahead

- Make udev rule file in `/etc/udev/rules.d` dir to make a permanent change to the readahead setting
  - Example (`/etc/udev/rules.d/60-sda-readahead.rules`):

```
ACTION=="add|change", KERNEL=="sda", ATTR{bdi/read_ahead_kb}="16"
```

  - Replace `sda` with block device name
  - udev runs on Linux startup, reboot to test the change

Basic Administration

OS TUNING

I/O Scheduler

- `cfq` - Avoid like the plague
- `deadline` - Generally best overall performance (Linux 2.6.32+)
- `noop` - Useful on Virtual Machines, AWS, etc

NUMA (Interleaving or Disabling)

- NUMA Interleaving Mode - recommended
- Most MongoDB packages do this already (`numactl --interleave=all mongod...`)
- Confirm nodes are balanced (or only node0) with `numastat`

Configuration tips - Summary

- Use XFS
- Use noatime
- Set up logrotate
- Separate data with `directoryPerDB: true`
- Configure backup from the get-go

Configuration tips - Summary

- Disable transparent huge pages
- Configure user authentication
- Set CPU governor to performance
- Don't use NFS
- Set NUMA interleave

Exercises - tuning the OS

1. Set the `noatime` mount option

```
vi /etc/fstab
UUID=6cd50e51-cfc6-40b9-9ec5-f32fa2e4ff02 / xfs defaults,noatime 0 0
```

2. Remount and verify filesystem and mount options

```
mount -o remount /
mount | grep ' / '
```

Exercises - tuning the OS

3. Set up logrotate

```
tee -a /etc/logrotate.d/mongod << EOF
/var/log/mongod.log {
  daily
  size 100M
  rotate 10
  missingok
  compress
  delaycompress
  notifempty
  create 640 mongod mongod
  sharedscripts
  postrotate
    /bin/kill -SIGUSR1 `cat /var/run/mongod.pid 2>/dev/null` >/dev/null 2>&1
  endscript
}
EOF
```

Exercises - tuning the OS

4. Disable THP

```
tee /etc/systemd/system/disable-transparent-huge-pages.service << EOF
[Unit]
Description=Disable Transparent Huge Pages (THP)
DefaultDependencies=no
After=sysinit.target local-fs.target
Before=mongod.service

[Service]
Type=oneshot
ExecStart=/bin/sh -c 'echo never | tee /sys/kernel/mm/transparent_hugepage/enabled

[Install]
WantedBy=basic.target
EOF


systemctl daemon-reload
systemctl start disable-transparent-huge-pages
systemctl enable disable-transparent-huge-pages
```

Exercises - tuning the OS

5. Disable THP (2)

```
mkdir /etc/tuned/virtual-guest-no-thp

tee /etc/tuned/virtual-guest-no-thp/tuned.conf << EOF
[main]
include=virtual-guest

[vm]
transparent_hugepages=never
EOF

tuned-adm profile virtual-guest-no-thp
```

Exercises - tuning mongod.conf

1. Check the current storage structure in /var/lib/mongo

```
ls -la /var/lib/mongo/
ls -la /var/lib/mongo/admin/
```

2. Separate data and index

```
vi /etc/mongod.conf
storage:
  dbPath: /var/lib/mongo
  directoryPerDB: true
  journal:
    enabled: true
  wiredTiger:
    engineConfig:
          directoryForIndexes: true
```

Exercises - tuning mongod.conf

3. Restart mongod to take effect

```
service mongod restart
```

- What happens?

4. Fix the problem by removing the current files and starting with empty datadir

```
service mongod stop
rm -rf /var/lib/mongo/*
service mongod start
```

5. Check how the storage structure was affected

```
ls -la /var/lib/mongo/admin/
```

Exercises - The Mongo Shell

1. Connect to mongo shell

```
# mongo

Percona Server for MongoDB shell version v3.4.5-1.5
connecting to: mongodb://127.0.0.1:27017
Percona Server for MongoDB server version: v3.4.5-1.5
Server has startup warnings:
2017-07-19T13:05:18.369+0000 I CONTROL  [initandlisten]
  ** WARNING: Access control is not enabled for the database.
2017-07-19T13:05:18.369+0000 I CONTROL  [initandlisten]
  **          Read and write access to data and configuration is unrestricted.
2017-07-19T13:05:18.369+0000 I CONTROL  [initandlisten]
  **          You can use percona-server-mongodb-enable-auth.sh to fix it.
2017-07-19T13:05:18.369+0000 I CONTROL  [initandlisten]
```

- Connections using the localhost exception only have access to create the first user on the `admin` database

Exercises - Mongo Shell Help

- Auto completion or looking up functions

```
db.<TAB><TAB>

db.adminCommand(                db.getCollectionInfos(          db.groupcmd(                    db.runCommandWithMetadata(
db.auth(                        db.getCollectionNames(          db.groupeval(                   db.runReadCommand(
db.changeUserPassword(          db.getLastError(                db.hasOwnProperty               db.serverBits(
db.cloneCollection(             db.getLastErrorCmd(             db.help(                        db.serverBuildInfo(
db.cloneDatabase(               db.getLastErrorObj(             db.hostInfo(                    db.serverCmdLineOpts(
db.commandHelp(                 db.getLogComponents(            db.isMaster(                    db.serverStatus(
db.constructor                  db.getMongo(                    db.killOP(                      db.setLogLevel(
db.copyDatabase(                db.getName(                     db.killOp(                      db.setProfilingLevel(
db.createCollection(            db.getPrevError(                db.listCommands(                db.setSlaveOk(
db.createRole(                  db.getProfilingLevel(           db.loadServerScripts(           db.setWriteConcern(
db.createUser(                  db.getProfilingStatus(          db.logout(                      db.shutdownServer(
db.createView(                  db.getQueryOptions(             db.movies                       db.stats(
db.currentOP(                   db.getReplicationInfo(          db.printCollectionStats(        db.system.profile
db.currentOp(                   db.getRole(                     db.printReplicationInfo(        db.toLocaleString
db.dbEval(                      db.getRoles(                    db.printShardingStatus(         db.toString(
db.dropAllRoles(                db.getSiblingDB(                db.printSlaveReplicationInfo(   db.tojson(
db.dropAllUsers(                db.getSisterDB(                 db.propertyIsEnumerable         db.unsetWriteConcern(
db.dropDatabase(                db.getSlaveOk(                  db.prototype                    db.updateRole(
db.dropRole(                    db.getUser(                     db.removeUser(                  db.updateUser(
db.dropUser(                    db.getUsers(                    db.repairDatabase(              db.upgradeCheck(
db.eval(                        db.getWriteConcern(             db.resetError(                  db.upgradeCheckAllDBs(
db.forceError(                  db.grantPrivilegesToRole(       db.revokePrivilegesFromRole(    db.valueOf(
db.fsyncLock(                   db.grantRolesToRole(            db.revokeRolesFromRole(         db.version(
db.fsyncUnlock(                 db.grantRolesToUser(            db.revokeRolesFromUser(
db.getCollection(               db.group(                       db.runCommand(
```

Exercises - Mongo Shell Help

- Analyzing a specific function

```
db.movies.findOne
function (query, fields, options, readConcern, collation) {
    var cursor = this.find(query, fields, -1 /* limit */, 0 /* skip*/, 0 /* batchSize */, options);

    if (readConcern) {
        cursor = cursor.readConcern(readConcern);
    }

    if (collation) {
        cursor = cursor.collation(collation);
    }

    if (!cursor.hasNext())
        return null;
    var ret = cursor.next();
    if (cursor.hasNext())
        throw Error("findOne has more than 1 result!");
    if (ret.$err)
        throw _getErrorWithCode(ret, "error " + tojson(ret));
    return ret;
}
```

Exercises - Enable authentication

1. Exit the shell and run the following command

```
# percona-server-mongodb-enable-auth.sh
We have detected authentication is not enabled.
Would you like help creating your first user?
Please note that mongodb service could be restarted during this action
Would you like to proceed?(Y/n)y
Percona Server for MongoDB shell version v3.4.5-1.5
connecting to: mongodb://localhost/admin
Percona Server for MongoDB server version: v3.4.5-1.5
Successfully added user: { "user" : "dba", "roles" : [ "root" ] }
bye
User has been created successfully!
User:dba
Password:MG7IvqFk47t1VnRnGbqMgSjjxGNZINSj
```

2. Restart the service

```
service mongod restart
```

3. Authenticate with the generated credentials

```
mongo -u dba -p MG7IvqFk47t1VnRnGbqMgSjjxGNZINSj --authenticationDatabase admin
```

Exercises - Test dataset

1. Get a test dataset from the internet

```
wget https://raw.githubusercontent.com/mongodb/docs-assets/primer-dataset/primer-dataset.json
```

2. Import the dataset

```
mongoimport --db test --collection restaurants --drop \
 --file primer-dataset.json
```

3. Check the imported data

```
> use test
> db.restaurants.findOne()
```

PERCONA

TRAINING

Questions