PERCONA

TRAINING

Troubleshooting

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

Troubleshooting

INFORMATION SOURCES

© 2011 - 2021 Percona, Inc.

Historic information

- The mongod (and mongos) log file
- Monitoring
  - Example: PMM in combination with mongodb_exporter.
- FTDC
  - Called "Full Time Diagnostic Data Capture" in MongoDB's documentation
- Audit log

© 2011 - 2021 Percona, Inc.

Dynamic information

- MongoDB commands
  - db.serverStatus()
  - db.currentOp()
  - db.stats() a.k.a. dbStats
  - many more
- Linux diagnostics
  - df, lsblk
  - top, sar
  - iostat
  - free, vmstat

© 2011 - 2021 Percona, Inc.

The log files

- Store events, including entries such as incoming connections, commands run, and issues encountered
- Logging API adds four automatic fields:
  - Timestamp
  - Severity
  - Component
  - Thread aka Context

© 2011 - 2021 Percona, Inc.

The log files - Timestamp

- The ISO-8601 timestamp at the start of the line is taken from the server's clock
- Simple

© 2011 - 2021 Percona, Inc.

The log files - Severity

- I(nfo)
- E(rror)
- W(arning)
- F(atal)
- D(ebug)
- Controlled by `db.setLogLevel()`

© 2011 - 2021 Percona, Inc.

The log files - Component

- Added to make it simpler to read
- Can be infered from Context field (see next)
- Examples
  - ACCESS
  - GEO
  - QUERY
  - SHARDING
  - COMMAND
  - INDEX
  - many more...

© 2011 - 2021 Percona, Inc.

The log files - Context

- Name of the class in MongoDB source code
- When a class can be instantiated multiple times it will have a numeric suffix added to it
- Examples
  - [main]
  - [initandlisten]
  - [listener]
  - [ReplCoord-0] (..-1, ..-2, etc.)

© 2011 - 2021 Percona, Inc.

The "old" log file

- Before v4.4, log lines started with the same fixed format for the leading four fields followed by a free text message space
- Regex-based text parsing was the best way to extract and analyze the logs
- Much treasure in huge logs could never be found within a reasonable amount of time

© 2011 - 2021 Percona, Inc.

The LogV2 file

- From v4.4 The format has changed to JSON
- Those same fixed fields in the leading four columns of the old log format are still present
- The message field becomes a "msg" field, plus optionally "attr" and "tags" fields.
- You can parse with `jq`

```
cat mongod.log | jq
```


© 2011 - 2021 Percona, Inc.

The "LogV2" log file

Example

```
{"t":{"$date":"2021-02-16T13:08:35.500+00:00"},"s":"I",  "c":"NETWORK",
"id":4648601, "ctx":"main","msg":"Implicit TCP FastOpen unavailable.
If TCP FastOpen is required, set tcpFastOpenServer, tcpFastOpenClient,
and tcpFastOpenQueueSize."}
```


© 2011 - 2021 Percona, Inc.

Exercises - logs

1. Install `jq`

```
yum install jq
```

2. Check top 10 unique messages sorted by frequency

```
jq -r ".msg" /var/log/mongodb/mongod.log | sort | uniq -c | sort -rn | head -10
```

3. Check the queries slower than 100 ms

```
jq '. | select(.attr.durationMillis>=100)' /var/log/mongo/mongod.log
```


© 2011 - 2021 Percona, Inc.

Exercises - logs

4. Filter logs of REPL

```
jq '. | select(.c=="REPL")' /var/log/mongo/mongod.log
```

5. See all logs except REPL

```
jq '. | select(.c!="REPL")' /var/log/mongo/mongod.log
```

6. See logs of REPL or STORAGE

```
jq '. | select( .c as $c | ["REPL", "STORAGE"] | index($c) )' /var/log/mongo/mongod.log
```

7. See all logs newer than April 15th, 2020

```
jq '. | \
 select(.t["$date"] >= "2020-04-15T00:00:00.000" and .t["$date"] <= "2020-04-15T23:59:59.999")' \
 /var/log/mongo/mongod.log
```


© 2011 - 2021 Percona, Inc.

Troubleshooting

EXTERNAL TOOLS

© 2011 - 2021 Percona, Inc.

mongostat

- Displays counts of database operations by type (e.g. insert, query, update, delete, etc.)
- Quickly see load distribution on the server
- Collects server samples every second
  - See current ops, pagefaults, network traffic, etc.
  - Does not give a view into historic performance - use PMM for that

© 2011 - 2021 Percona, Inc.

mongostat - fields

```
insert query update delete getmore command dirty used flushes vsize   res qrw arw net_in net_out conn       set r
    0     0      0      0       0     1|0  0.7%  1.6%     0 1.86G   90.0M 0|0 1|0  313b     43.3k     7    myReplSet
    0     0      0      0       0     3|0  0.7%  1.6%     0 1.86G   90.0M 0|0 1|0  727b     44.7k     7    myReplSet
```

- flushes
  - number of WT checkpoints between each polling interval
- dirty/used
  - WT cache related
- qr/qw
  - queue of clients
- ar/aw
  - active operations

© 2011 - 2021 Percona, Inc.

mongotop

- Reports the current read and write activity of a MongoDB instance, on a per collection basis
- Use `mongotop` to check if your database activity and use match your expectations
- `mongotop` looks at the time spent on reads/writes in each collection

© 2011 - 2021 Percona, Inc.

Exercises - `mongostat` and `mongotop`

1. Open a separate session and connect to the `PRIMARY`

```
mongotop --port 27017
```


© 2011 - 2021 Percona, Inc.

Exercises - `mongostat` and `mongotop`

2. Run the following commands to simulate write activity

```
use training
db.testcol.drop()
for (i=1; i<=10000; i++) {
  arr = [];
  for (j=1; j<=1000; j++) {
    doc = { _id: (1000 * (i-1) + j), a: i, b: j, c: (1000 * (i-1)+ j) };
    arr.push(doc)
  };
  db.testcol.insert(arr);
  var x = db.testcol.find( { b : 255 } );
  x.next();
  var x = db.testcol.find( { _id : 1000 * (i-1) + 255 } );
  x.next();
  var x = "asdf";
  db.testcol.update( { a : i, b : 255 }, { $set : { d : x.pad(1000) } });
  print(i)
}
```


© 2011 - 2021 Percona, Inc.

Exercises - `mongostat` and `mongotop`

3. On another session, run `mongotop`

```
mongotop --port 27017
```

4. Stop `mongotop` and run `mongostat`

```
mongostat --port 27017
```

5. What is the difference when you run this command?

```
mongostat --port 27017 --discover
```


© 2011 - 2021 Percona, Inc.

Troubleshooting

BUILTIN TOOLS/COMMANDS

© 2011 - 2021 Percona, Inc.

Server Level Tools

- `db.setProfilingLevel()`
- `db.currentOp()`
- `db.<COLLECTION>.stats()`
- `db.serverStatus()`

© 2011 - 2021 Percona, Inc.

The profiler

- Collects detailed info about commands
- CRUD and admin commands
- Stores data on `system.profile` in `admin` database (capped at 1 MB)
- Can have a performance impact

© 2011 - 2021 Percona, Inc.

Enabling the Profiler

```
myReplSet:SECONDARY> use training

myReplSet:SECONDARY> db.getProfilingLevel()
0

myReplSet:SECONDARY> db.getProfilingStatus()
{ "was" : 0, "slowms" : 100, "ratelimit" : 1 }

myReplSet:SECONDARY> db.setProfilingLevel(2)
{ "was" : 0, "slowms" : 100, "ratelimit" : 1, "sampleRate" : 1, "ok" : 1 }
```


© 2011 - 2021 Percona, Inc.

Validate Profiler Status

```
myReplSet:SECONDARY> db.getProfilingStatus()
{ "was" : 2, "slowms" : 0, "ratelimit" : 1 }

myReplSet:SECONDARY> db.getProfilingLevel()
2
```


© 2011 - 2021 Percona, Inc.

Test and Inspect Query Profiles

```
myReplSet:SECONDARY> db.movies.find({ "title": "Batman" })
{ "_id" : ObjectId("5bd26f69e07ae9a4b76d0c1a"), "title" : "Batman" }

myReplSet:SECONDARY> db.system.profile.find({ "ns": "training.movies" }).pretty()
{
  "op" : "query",
  "ns" : "training.movies",
  "query" : {
    "find" : "movies",
    "filter" : {
      "title" : "Batman"
    }
  },
  "keysExamined" : 0,
  "docsExamined" : 1,
  "cursorExhausted" : true,
  "numYield" : 0,
  "locks" : {
    ...
  },
  ...
  "execStats" : {
    "stage" : "COLLSCAN",
    "filter" : {
      "title" : {
        "$eq" : "Batman"
      }
    },
    "nReturned" : 1,
    "executionTimeMillisEstimate" : 0,
    ...
    "docsExamined" : 1
  },
  ...
}
```


© 2011 - 2021 Percona, Inc.

The Profiler - options

- Levels
  - 0: don't collect any data (default)
  - 1: collect operations slower than `slowms`
  - 2: collect all operations
- Sample rate
  - Specifies the fraction of data to be logged
  - Ignores operations faster than `slowMs`
  - Numbers between 0 and 1

© 2011 - 2021 Percona, Inc.

The Profiler vs. Diagnostic Log

- `slowms` and `slowOpSampleRate` also affect which operations are recorded to the diagnostic log
- By default slow operations are logged to diagnostic log even if profiler is disabled
- This is controlled by `LogLevel` setting
  - Default `LogLevel` is 0 (Informational)

© 2011 - 2021 Percona, Inc.

Exercises - The profiler

1. Check profiler settings

```
> use training
> db.getProfilingStatus()
```

2. Review the logs and see if you can spot any slow operations

```
2021-02-11T14:49:10.056+0000 I  WRITE    [conn21] update training.testcol appName:
```


© 2011 - 2021 Percona, Inc.

Exercises - The profiler (2)

3. Enable the profiler for the training database

```
> use training
> db.setProfilingLevel(2)
```

4. Query some data

```
> db.testcol.find()
```

4. Check the profile collection

```
db.system.profile.find().limit(3).sort( { ts : -1 } ).pretty()
```


© 2011 - 2021 Percona, Inc.

db.currentOp()

- Returns a document that contains information on in-progress operations for the database instance
- Parameters accepted:
  - document with filtering conditions or
  - a boolean parameter to include/excluded idle and system operations

© 2011 - 2021 Percona, Inc.

db.currentOp()

- Examples
  - All operations including system/idle

```
db.currentOp(true)
```

  - Operations on db1 running longer than 3 sec:

```
db.currentOp(
{
"active" : true,
"secs_running" : { "$gt" : 3 },
"ns" : /^db1\./
}
)
```


© 2011 - 2021 Percona, Inc.

Exercises - db.currentOp()

1. Run the command and inspect the output

```
db.currentOp()
```

2. Count the number of connections per host

```
db.currentOp(true).inprog.reduce((accumulator, connection) => { ipaddress = connection.client ?
connection.client.split(":")[0] : "Internal"; accumulator[ipaddress] = (accumulator[ipaddress]
|| 0) + 1; accumulator["TOTAL_CONNECTION_COUNT"]++; return accumulator; },
 { TOTAL_CONNECTION_COUNT: 0 })
```


© 2011 - 2021 Percona, Inc.

db.<COLLECTION>.stats()

- Returns statistics that reflect the use state of a single database
- Optional parameter: `scale`
  - `db.stats( 1024*1024 )` to see in MB

© 2011 - 2021 Percona, Inc.

db.<COLLECTION>.stats() Example

```
> db.stats( 1024*1024 )
{
  "db" : "admin",
  "collections" : 4,
  "views" : 0,
  "objects" : 8,
  "avgObjSize" : 276.25,
  "dataSize" : 0.0021076202392578125,
  "storageSize" : 0.125,
  "indexes" : 6,
  "indexSize" : 0.1796875,
  "totalSize" : 0.3046875,
  "scaleFactor" : 1048576,
  "fsUsedSize" : 3414.44140625,
  "fsTotalSize" : 102388.98046875,
  "ok" : 1,
  "$clusterTime" : {
    "clusterTime" : Timestamp(1613575031, 1),
    "signature" : {
      "hash" : BinData(0,"XJ3QxsXghqDIr8tFVNPfASA3oyU="),
      "keyId" : NumberLong("6929873004167954436")
    }
  },
  "operationTime" : Timestamp(1613575031, 1)
}
```


© 2011 - 2021 Percona, Inc.

db.serverStatus()

- Returns a document that provides an overview of the database process's state
- Statistics are reset on server start
- Returns a lot of info
- Filter with
  - `db.serverStatus( { repl: 0, metrics: 0, locks: 0 } )`

© 2011 - 2021 Percona, Inc.

Exercise `db.serverStatus()`

1. Get the uptime

```
> db.serverStatus().uptime
```

2. Get the WT cache stats

```
> db.serverStatus().wiredTiger.cache
```

3. Calculate the WT cache evictions per second, defined as: `( "unmodified pages evicted" + "modified pages evicted" ) / uptime`

```
> (3459 + 4324 ) / db.serverStatus().uptime
0.09061273910563143
```


© 2011 - 2021 Percona, Inc.

PERCONA
TRAINING

Questions

© 2011 - 2021 Percona, Inc.