MongoDB Performance issues & Troubleshooting methods

By Vinodh Krishnaswamy

Hello!

I am Vinodh

- Friends call me ***GURUJI***
- MongoDB, MySQL
- Proud Perconian
- I am here because I love to talk & give presentations.

You can find me at @
https://www.linkedin.com/in/vinodhkrish

Performance!

What is Database performance?

What is the expected Performance?

How does a ROCKET go into the space?

“

A performance issue in a database occurs when database operations (such as DML, DDLs) take longer than expected, consume excessive system resources, or cause bottlenecks that degrade overall application performance. These situation can lead to slow responses, high resource usages, replication lag etc

MongoDB Performance Issues

- High CPU / Memory usages
- Slow queries
- Inefficient Indexes & execution plan
- write/read tickets (concurrency)
- Page faults - (working set)
- Locking and Contention
- write/read Latency

Performance Issues

- Replica lag
- Hot shard
- Wrong shard key & Balancing issues
- Chunk migration issues (Balancer window, waitForDelete, secondaryThrottle)
- Disk Saturation
- IOPs bottleneck
- Backup & Restores

1. Slow query related issues

Slow query related issues

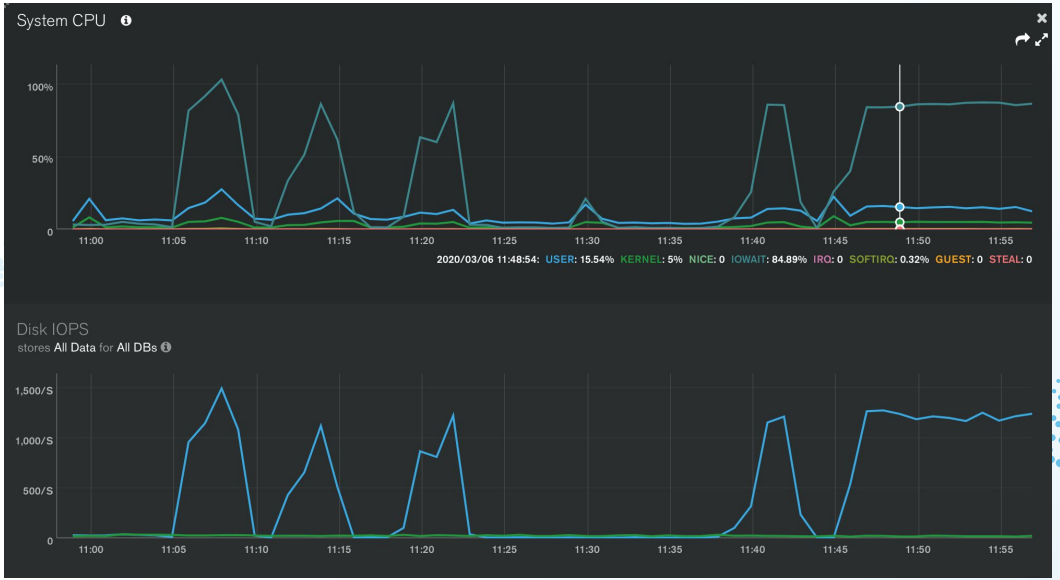
- COLLSCAN queries
- Inefficient Indexes & execution plan
- write/read Latency
- Complex joins / aggregations

Slow query related issues

*{"t":{"$date":"2025-03-01T03:09:58.512-05:00"},"s":"W", "c":"QUERY", "id":23798, "ctx":"conn1769","msg":"Plan executor error during find command","attr":{"error":{"code":50,"codeName":"MaxTimeMSExpired",**"errmsg":"operation exceeded time limit"**}, "stats":{"stage":"SHARDING_FILTER", "nReturned":0,"works":307999, "advanced":0,"needTime":307999,"needYield":0,"saveState":308,"restoreState":307,"isEOF":0,"chunkSkips":0,"inputStage":{**"stage":"COLLSCAN"**,"filter":{"https":{"$eq":"//wwwin-gaurav"}},"nReturned":0,"works":307999,"advanced":0,"needTime":307999,"needYield":0,"saveState":308,"restoreState":307,"isEOF":0,"direction":"forward"**,"docsExamined":307998**}},"cmd":{"find":"collData","filter":{"https":"//wwwin-gaurav"},"shardVersion":[{"$timestamp":{"t":468,"i":1}},{"$oid":"5e2beb7b183382e43fced622"},{"$timestamp":{"t":1723827998,"i":1040}}],"batchSize":1,"singleBatch":true,**"maxTimeMS":1000,**"$readPreference":**{"mode":"secondaryPreferred"}**,"readConcern":{"level":"local"},"$db":"logdata"}}}*

Slow query related issues

- COLLSCAN queries
  - Disk Scan - Collection Scan - (iowait%)



It affects...

- Disk scans & saturation
- write/read Latency
  - Increased index lookup time
  - Data retrieval time
  - Network latency
  - Disk I/O
  - replicaSet RW concerns

Solutions...

- Tune Inefficient Indexes - **EXPLAIN** plan is your friend
- Execution plan changes after upgrade
- Avoid using $ne, $nin,
- Avoid $or(if not all fields are indexed)
- Use Limit
- Use projections to display only required fields
- **ESR rule** for index creation

ESR Rule

- Equality - $in, $eq
- Sort - $sort
- Range - $gt, $lt, $ne, $nin, $regex

ESR Rule

```
db.testColl.find(
    {
        City: "Hyderabad",
        Count: { $gt: 2000 }
    } ).sort( { Venue: 1 } )
```

ESR Rule

```
db.testColl.find(
    {
        City: "Hyderabad",
        Count: { $gt: 2000 }
    } ).sort( { Venue: 1 } )


db.testColl.createIndex({City:1, Venue:1, Count:1})
```

Problematic query

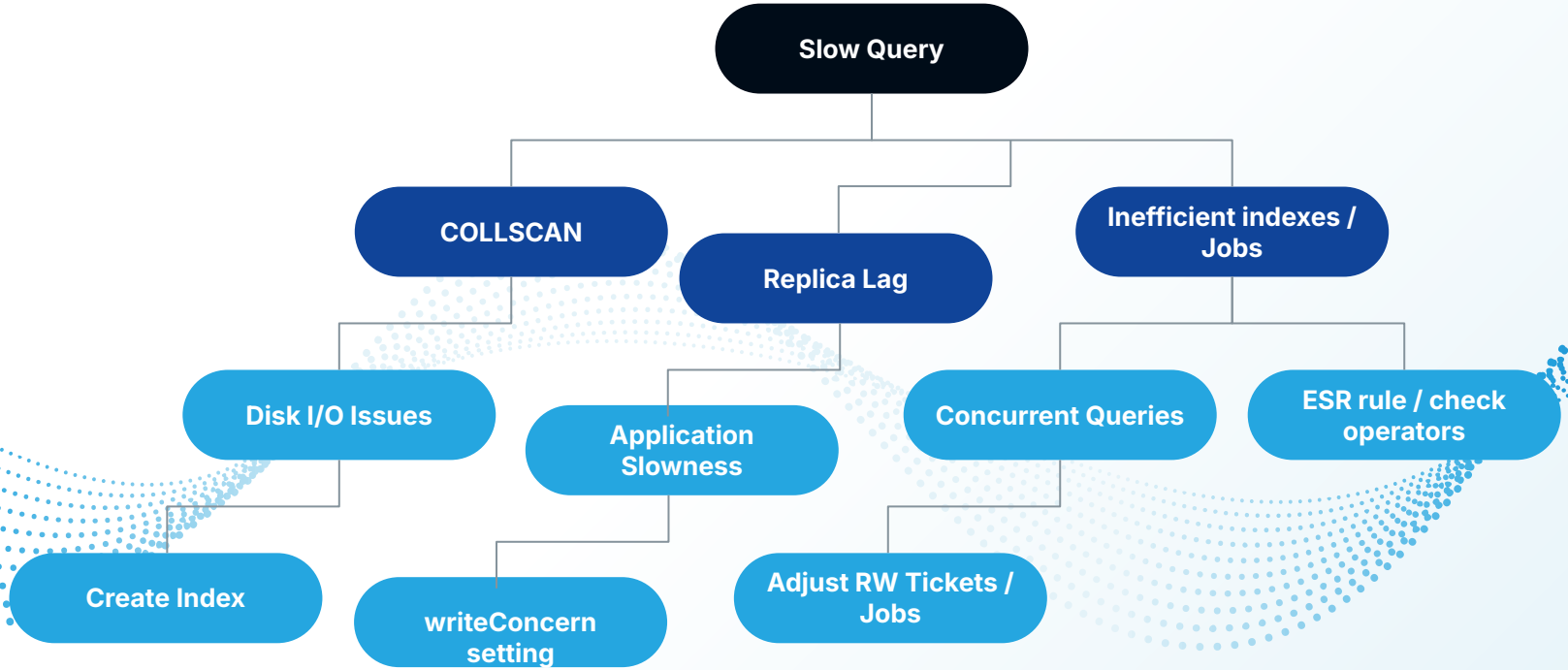
```
replset:PRIMARY> db.testAgg.aggregate([{ $match: {} }, { $group: {_id:1, num: {$sum:1} }}
])
{ "_id" : 1, "num" : 10000 }
```

```
replset:PRIMARY> db.system.profile.find()
{ "op" : "command", "ns" : "vinodh.testAgg", "command" : { "aggregate" : "testAgg", "pipeline" : [
{ "$match" : {  } }, { "$group" : { "_id" : 1, "num" : { "$sum" : 1 } } } ], "cursor" : {  },
"lsid" : { "id" : UUID("98734d8e-c8c3-4f05-92a8-c97ae57ed4a1") }, "$clusterTime" : { "clusterTime"
: Timestamp(1739542682, 1), "signature" : { "hash" : BinData(0,"QFDe0WlVhDXTMRmDandthE+ugSo="),
"keyId" : NumberLong("7471277812494630914") } }, "$db" : "vinodh" }, "keysExamined" : 0,
"docsExamined" : 10000, "cursorExhausted" : true, "numYield" : 10, "nreturned" : 1, "locks" : {
"FeatureCompatibilityVersion" : { "acquireCount" : { "r" : NumberLong(12) } },
"ReplicationStateTransition" : { "acquireCount" : { "w" : NumberLong(12) } }, "Global" : {
"acquireCount" : { "r" : NumberLong(12) } }, "Database" : { "acquireCount" : { "r" : NumberLong(12)
} }, "Collection" : { "acquireCount" : { "r" : NumberLong(12) } }, "Mutex" : { "acquireCount" : {
"r" : NumberLong(2) } } }, "flowControl" : {  }, "responseLength" : 262, "protocol" : "op_msg",
"millis" : 8, "rateLimit" : 1, "planSummary" : "COLLSCAN", "ts" :
ISODate("2025-02-14T14:18:13.952Z"), "client" : "127.0.0.1", "appName" : "MongoDB Shell",
"allUsers" : [ { "user" : "user", "db" : "admin" } ], "user" : "user@admin" }
```

Solving Problematic query

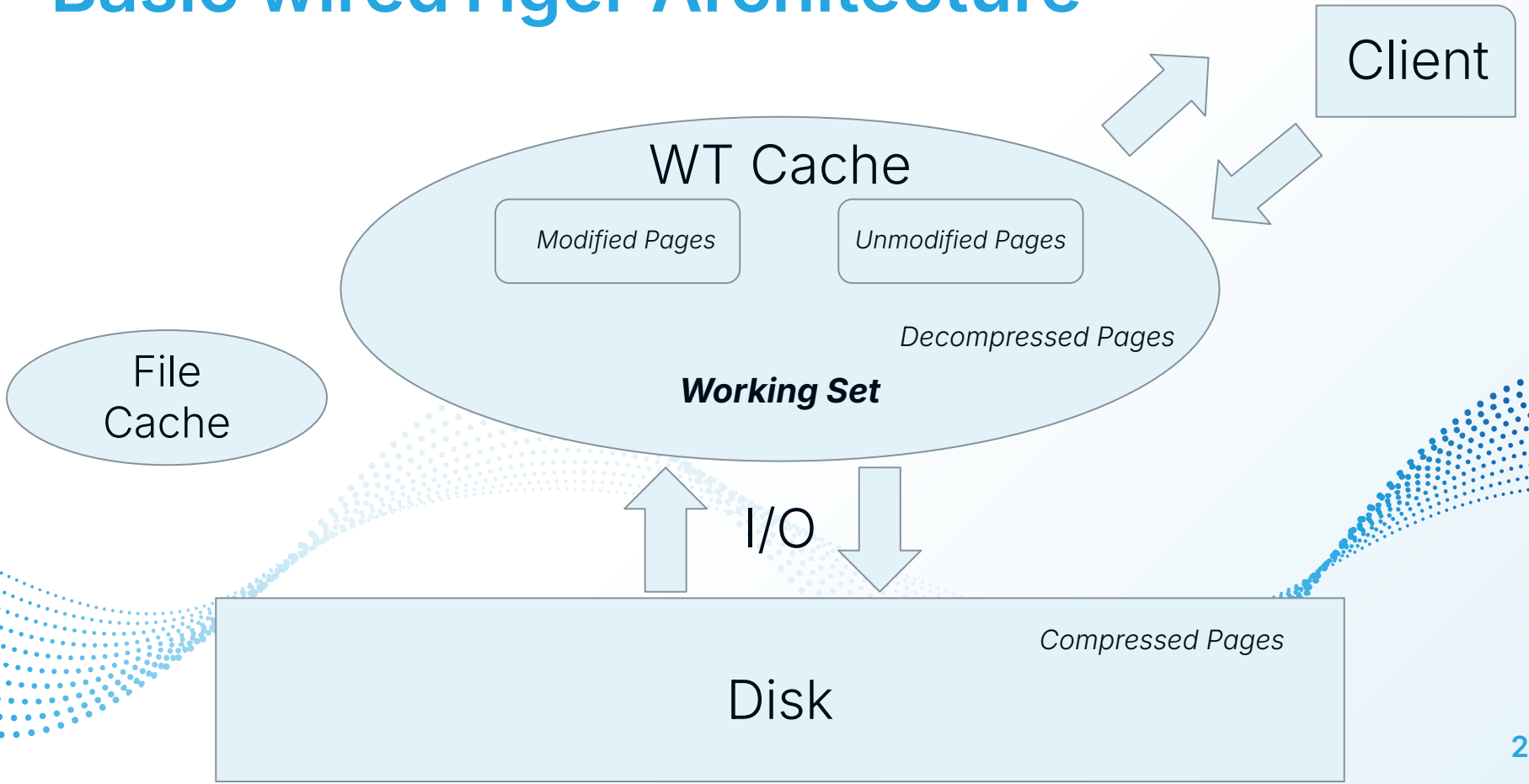
```
replset:PRIMARY> db.testAgg.aggregate([{ $match: {} }, {$sort: {_id:1} }, { $group: {_id:1,
num: {$sum:1} }} ])
{ "_id" : 1, "num" : 10000 }

Plan:
{ "op" : "command", "ns" : "vinodh.testAgg", "command" : { "aggregate" : "testAgg", "pipeline"
: [ { "$match" : {  } }, { "$sort" : { "_id" : 1 } }, { "$group" : { "_id" : 1, "num" : {
"$sum" : 1 } } } ], "cursor" : {  }, "lsid" : { "id" :
UUID("98734d8e-c8c3-4f05-92a8-c97ae57ed4a1") }, "$clusterTime" : { "clusterTime" :
Timestamp(1739542762, 1), "signature" : { "hash" : BinData(0,"tip2cXcaFn0YSthCHS7tqAuFnY4="),
"keyId" : NumberLong("7471277812494630914") } }, "$db" : "vinodh" }, "keysExamined" : 10001,
"docsExamined" : 0, "cursorExhausted" : true, "numYield" : 10, "nreturned" : 1, "queryHash" :
"F44244FE", "planCacheKey" : "F44244FE", "locks" : { "FeatureCompatibilityVersion" : {
"acquireCount" : { "r" : NumberLong(12) } }, "ReplicationStateTransition" : { "acquireCount" :
{ "w" : NumberLong(12) } }, "Global" : { "acquireCount" : { "r" : NumberLong(12) } },
"Database" : { "acquireCount" : { "r" : NumberLong(12) } }, "Collection" : { "acquireCount" : {
"r" : NumberLong(12) } }, "Mutex" : { "acquireCount" : { "r" : NumberLong(2) } } },
"flowControl" : {  }, "responseLength" : 262, "protocol" : "op_msg", "millis" : 5, "rateLimit"
: 1, "planSummary" : "COUNT_SCAN { _id: 1 }", "ts" : ISODate("2025-02-14T14:19:57.787Z"),
"client" : "127.0.0.1", "appName" : "MongoDB Shell", "allUsers" : [ { "user" : "user", "db" :
"admin" } ], "user" : "user@admin" }
```

Handling the Situation...



2. wiredTiger related issues

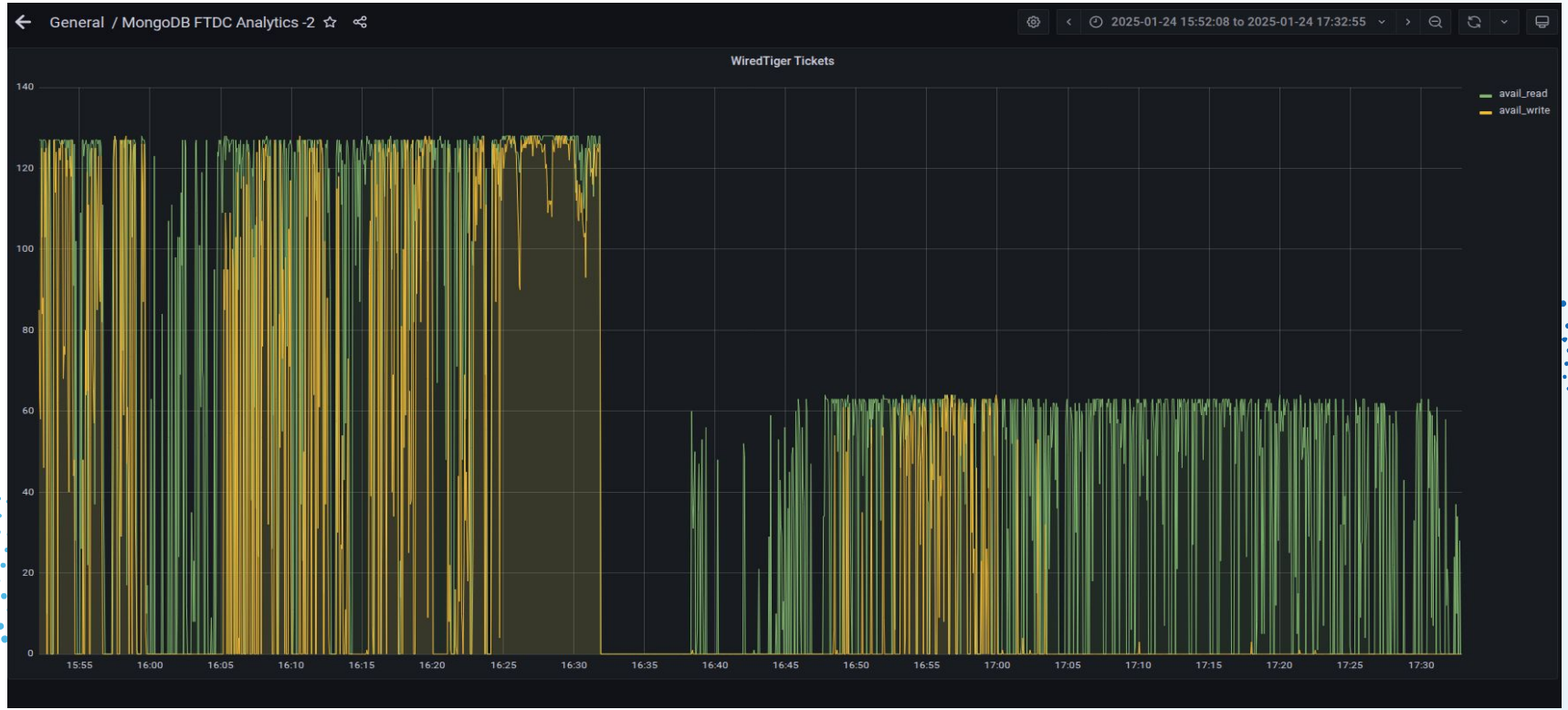
Basic wiredTiger Architecture



wiredTiger related issues

- write/read tickets (concurrency)
- Page faults
- Locking and Contention

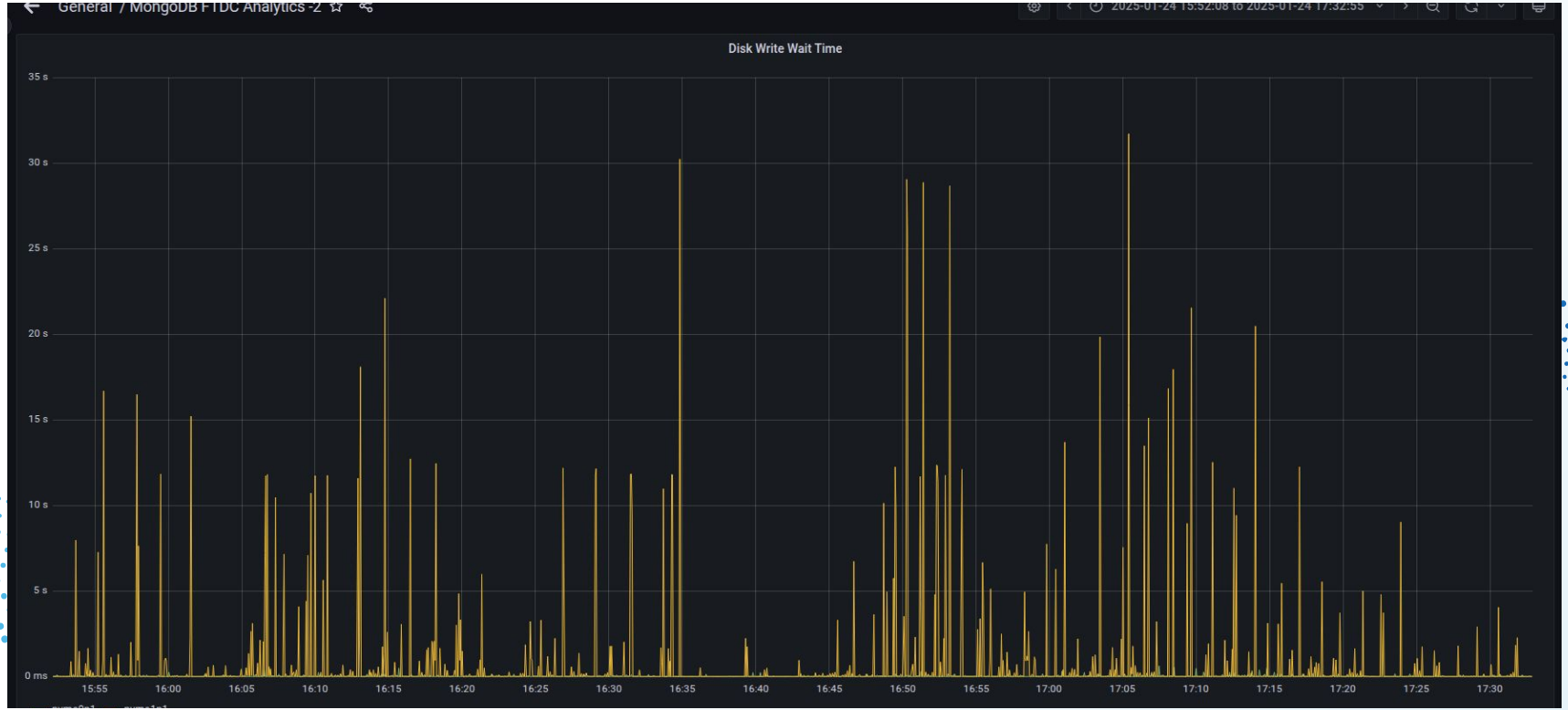
Concurrent Tickets



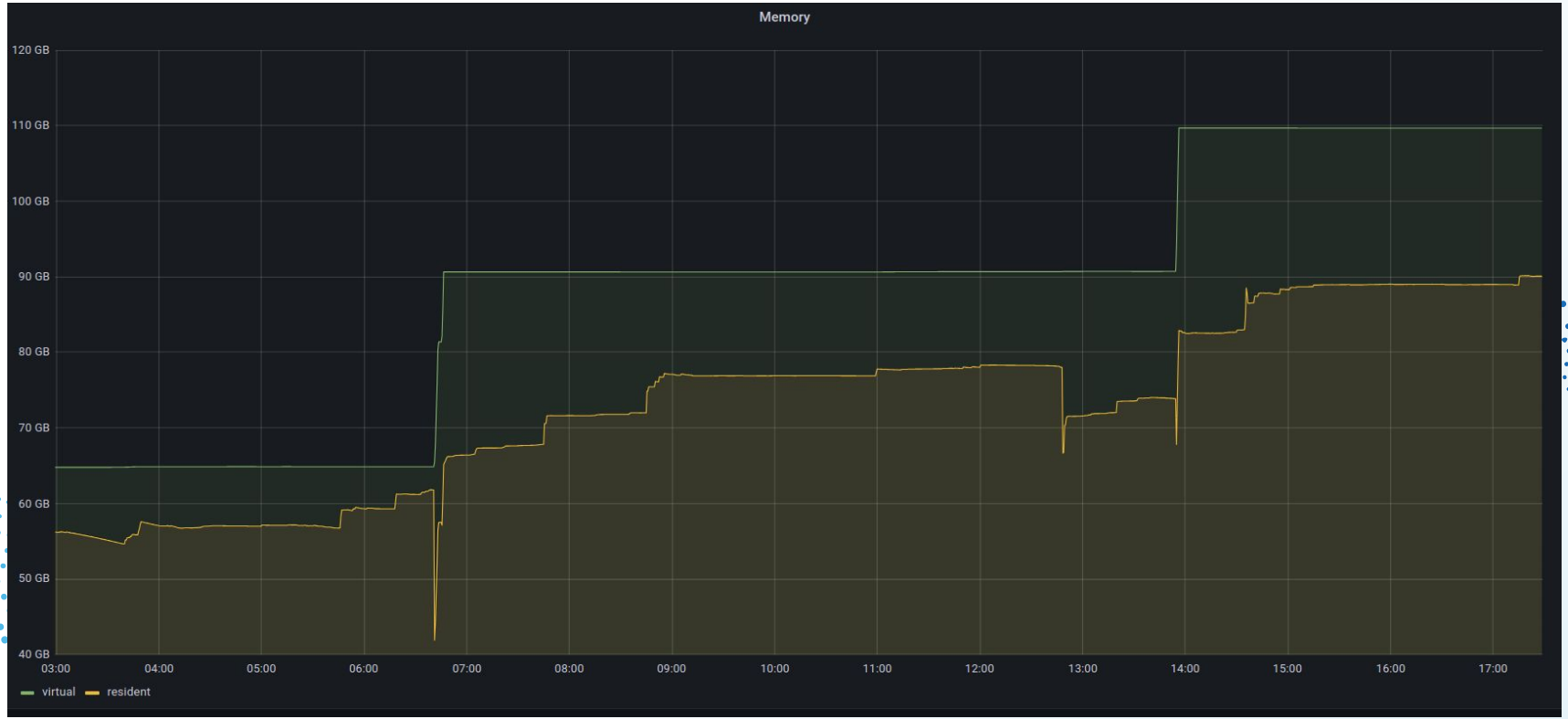
Concurrent Tickets

```
db.serverStatus().wiredTiger.concurrentTransactions
{
"write" : {
"out" : 1,
"available" : 127,
"totalTickets" : 128
},
"read" : {
"out" : 99,
"available" : 29,
"totalTickets" : 128
}
}
```

Disk IO

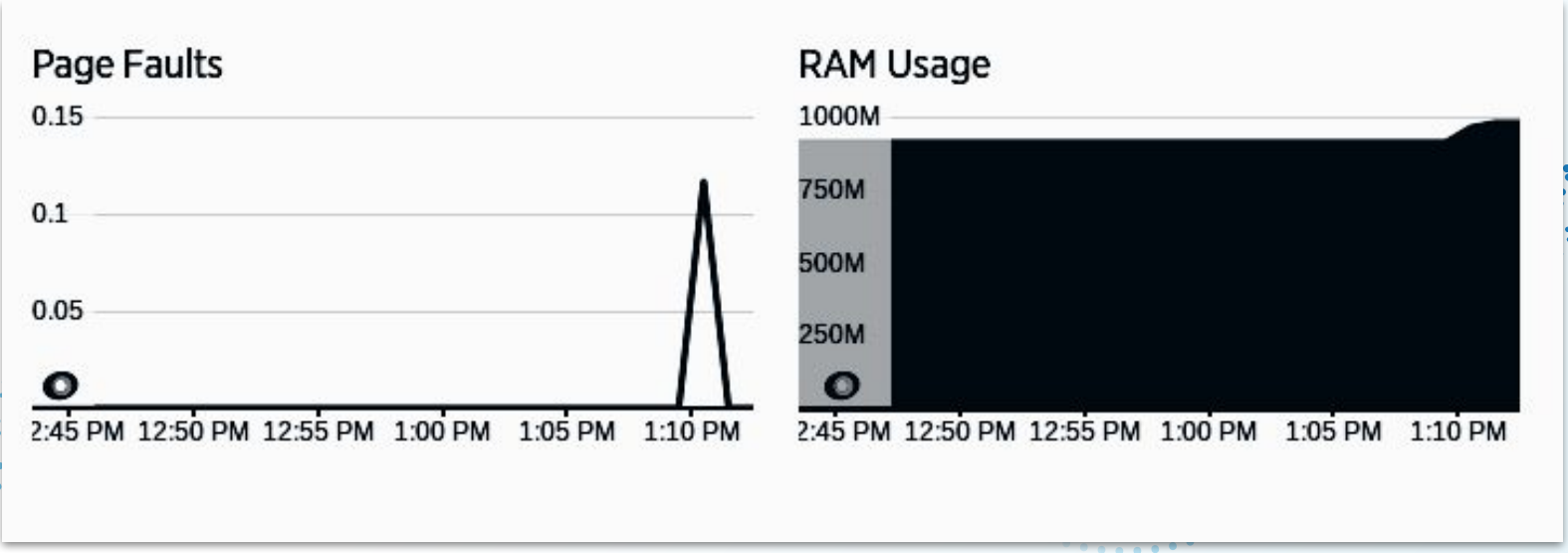

General / MongoDB FTDC Analytics -2
2025-01-24 15:52:08 to 2025-01-24 17:32:55
Disk Write Wait Time
35 s
30 s
25 s
20 s
15 s
10 s
5 s
0 ms
15:55
16:00
16:05
16:10
16:15
16:20
16:25
16:30
16:35
16:40
16:45
16:50
16:55
17:00
17:05
17:10
17:15
17:20
17:25
17:30

Resident Memory



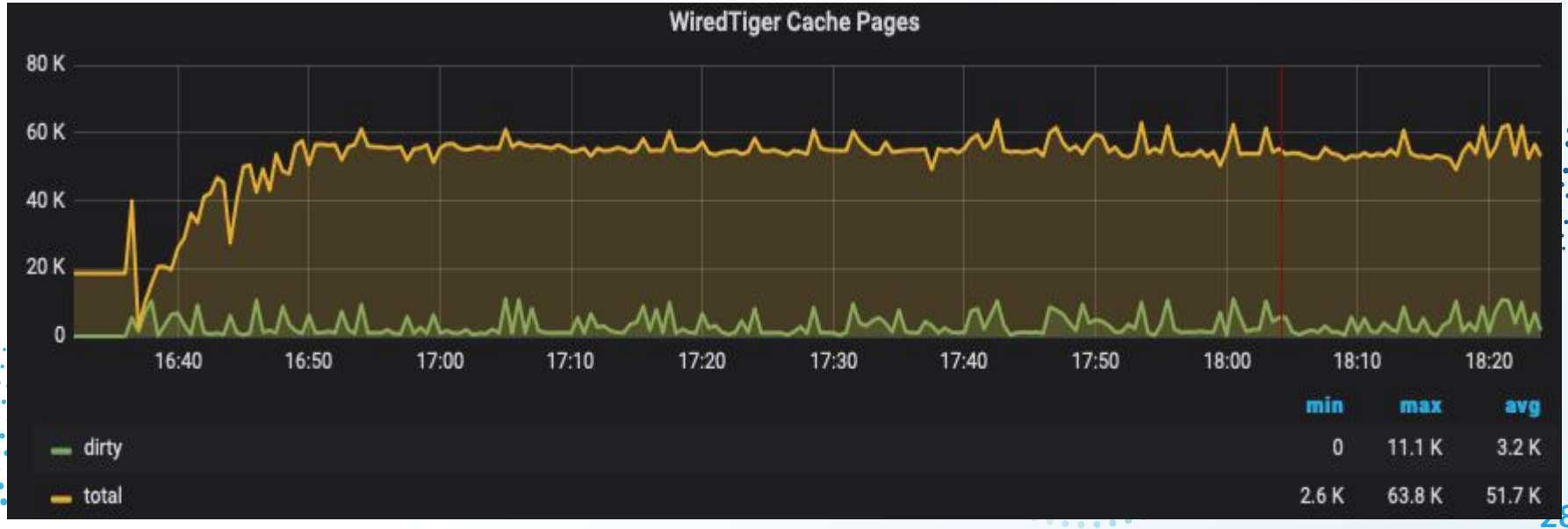
PageFault

- Page faults - (working set to Memory)



WT Cache usages

- Checkpoint / Flushes - (eviction)



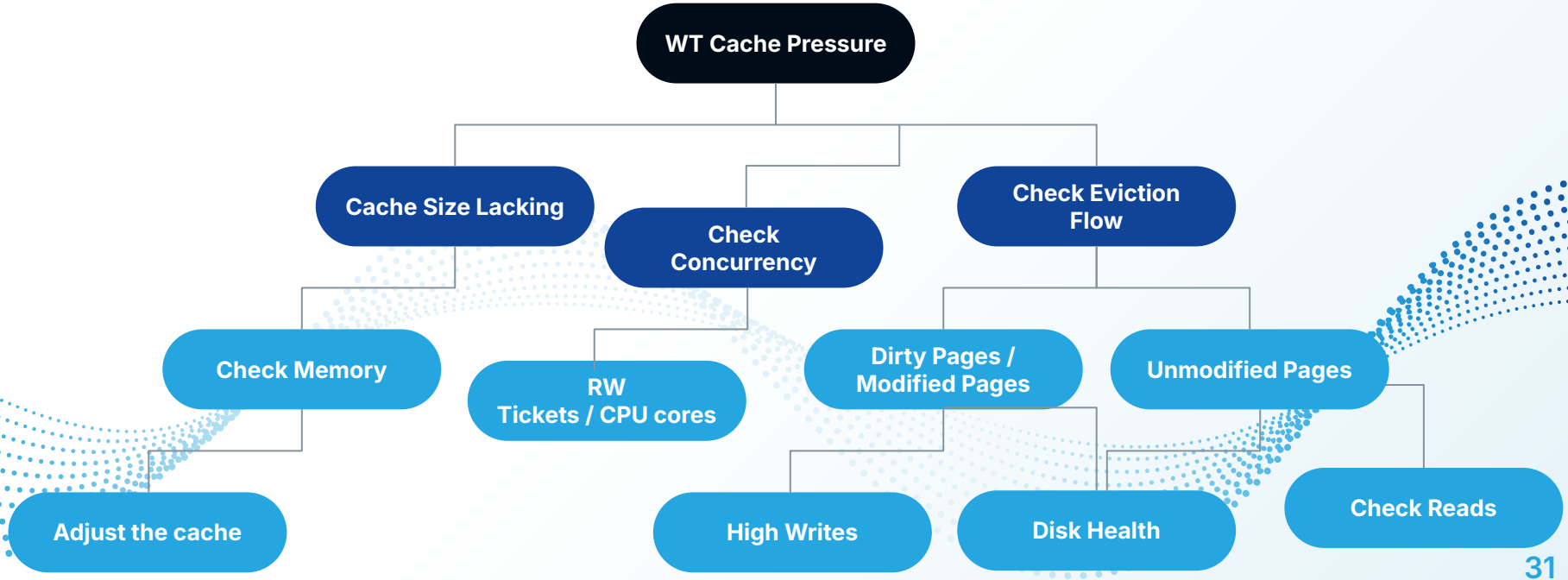
wiredTiger tuning

- Tune eviction process
- Limit Dirty Pages
- eviction_target , eviction_trigger
- eviction_dirty_target, eviction_dirty_trigger

wiredTiger tuning

```
db.adminCommand( { "setParameter": 1,
"wiredTigerEngineRuntimeConfig":
"eviction=(threads_min=20,threads_max=20),checkpoint=(wait=60),e
viction_dirty_trigger=5,eviction_dirty_target=1,eviction_trigger
=95,eviction_target=80"})
```

Handling the Situation...



Our Projects.

- PSMDB
- PSMDB Operator
- PBM
- PMM
- Everest


WE ARE HIRING
Percona is growing.
See our open positions.
PERCONA
Reach out to us!

Thank You!

Any questions?

You can find me at

- LinkedIN @vinodhkrish