#MongoDB

Performance Tuning and Monitoring Using MMS

Daniel Coupal

Technical Services Engineer, Palo Alto, CA @ MongoDB

mongoDB

Agenda

1. What and who is MMS for?
2. Performance and monitoring examples
3. Setting it up and getting around
4. Wrapping up
5. Q&A

1. What and who is MMS for?

What is MMS?

The *MongoDB Management Service*

- a free Cloud service for monitoring and managing your MongoDB clusters
- or available to run *On-Prem* for customers with the *Standard* or *Enterprise Subscriptions*
- tool that makes MongoDB easier to use

Who is MMS for?

“For years, MongoDB has been planning to do monitoring right.“

“We want the Ops People to have the same enthusiasm as the Developers.”

- Developers
  - Track bottlenecks
- Ops Team
  - Monitor health of the clusters
  - Backup databases
  - Automate updates and add capacity
- MongoDB Technical Service Team

What is in MMS?

A. Monitoring
   1. Cloud: Sept 2011
   2. On-Prem: July 2013

B. Backups
   1. Cloud: April 2013
   2. On-Prem: April 2014

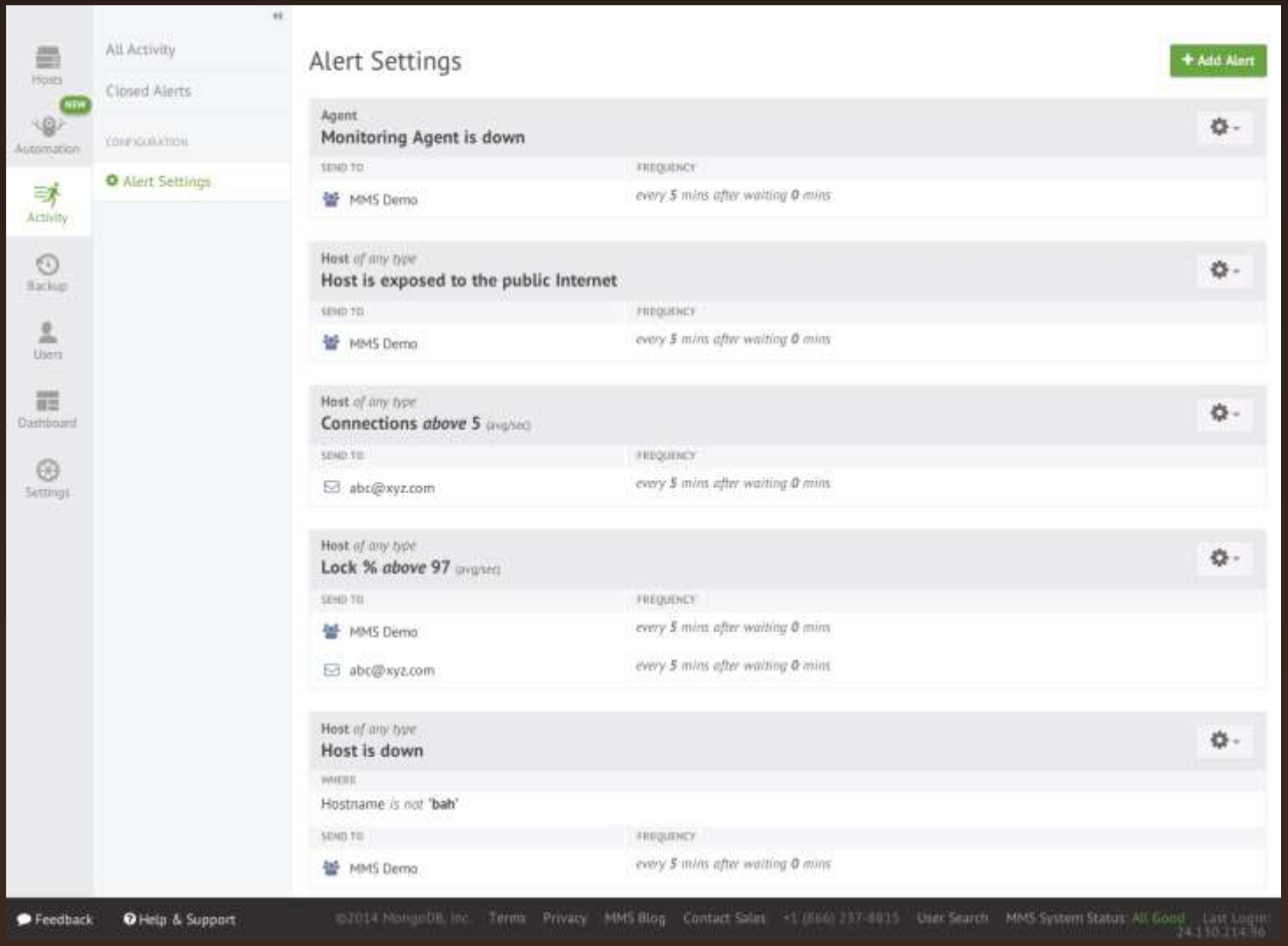
C. Automation
   1. Cloud: April 2014 (Beta)

What is in MMS monitoring?

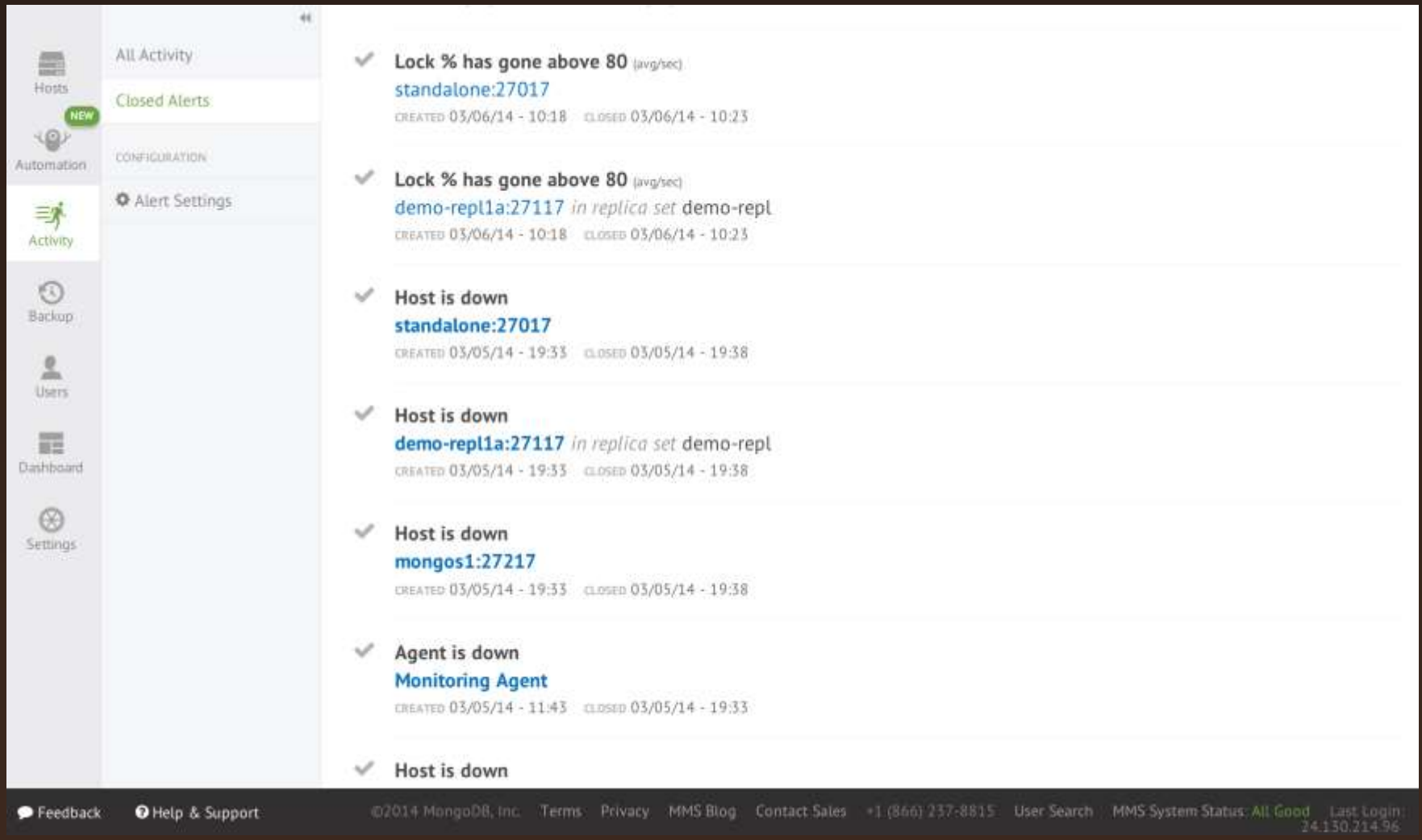
A. Metric Collection and Reporting
B. Alerting
   (Email, SMS, PagerDuty, HipChat, SNMP)
C. Event Tracking
D. Database Stats
E. Hardware Stats
F. Logs and Profile Data

A. Metric Collection and Reporting

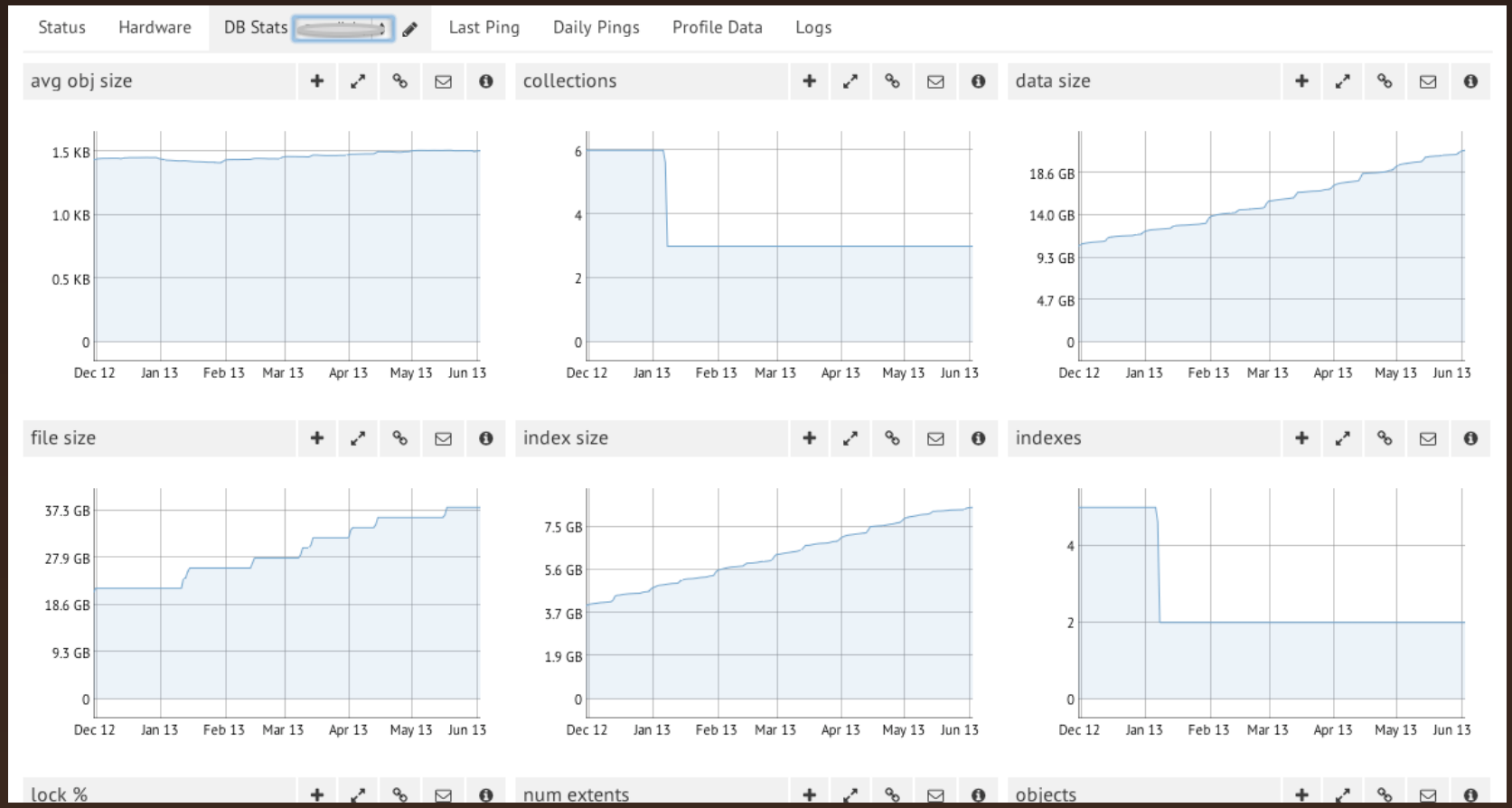
B. Alerting


All Activity
Closed Alerts
CONFIGURATION
Alert Settings
Hosts
Automation
Activity
Backup
Users
Dashboard
Settings
Alert Settings
+ Add Alert
Agent
Monitoring Agent is down
SEND TO
FREQUENCY
MMS Demo
every 5 mins after waiting 0 mins
Host of any type
Host is exposed to the public Internet
SEND TO
FREQUENCY
MMS Demo
every 5 mins after waiting 0 mins
Host of any type
Connections above 5 (avg/sec)
SEND TO
FREQUENCY
abc@xyz.com
every 5 mins after waiting 0 mins
Host of any type
Lock % above 97 (avg/sec)
SEND TO
FREQUENCY
MMS Demo
every 5 mins after waiting 0 mins
abc@xyz.com
every 5 mins after waiting 0 mins
Host of any type
Host is down
WHERE
Hostname is not 'bah'
SEND TO
FREQUENCY
MMS Demo
every 5 mins after waiting 0 mins
Feedback
Help & Support
©2014 MongoDB, Inc.
Terms
Privacy
MMS Blog
Contact Sales
User Search
MMS System Status: All Good
Last Login:

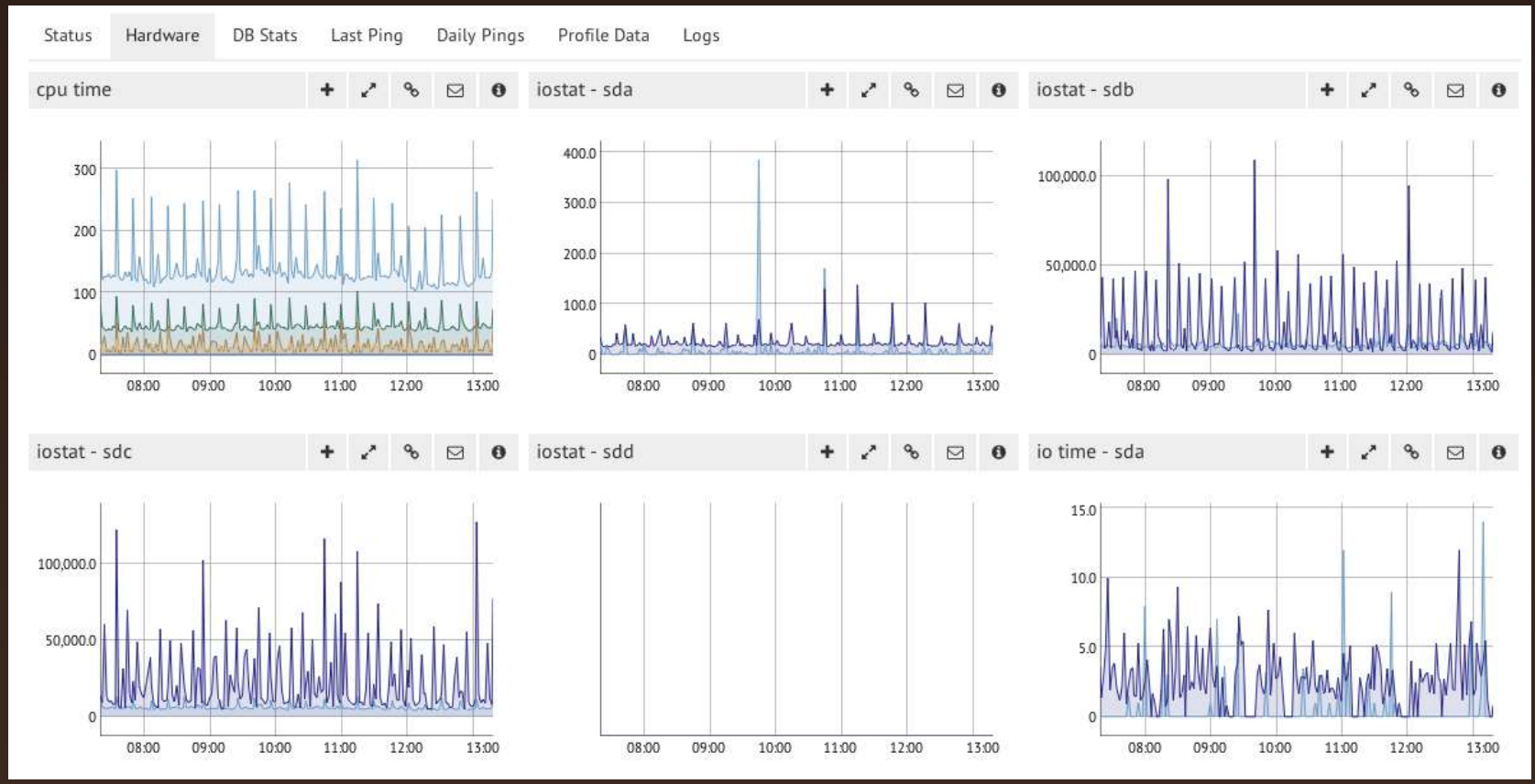
C. Event Tracking



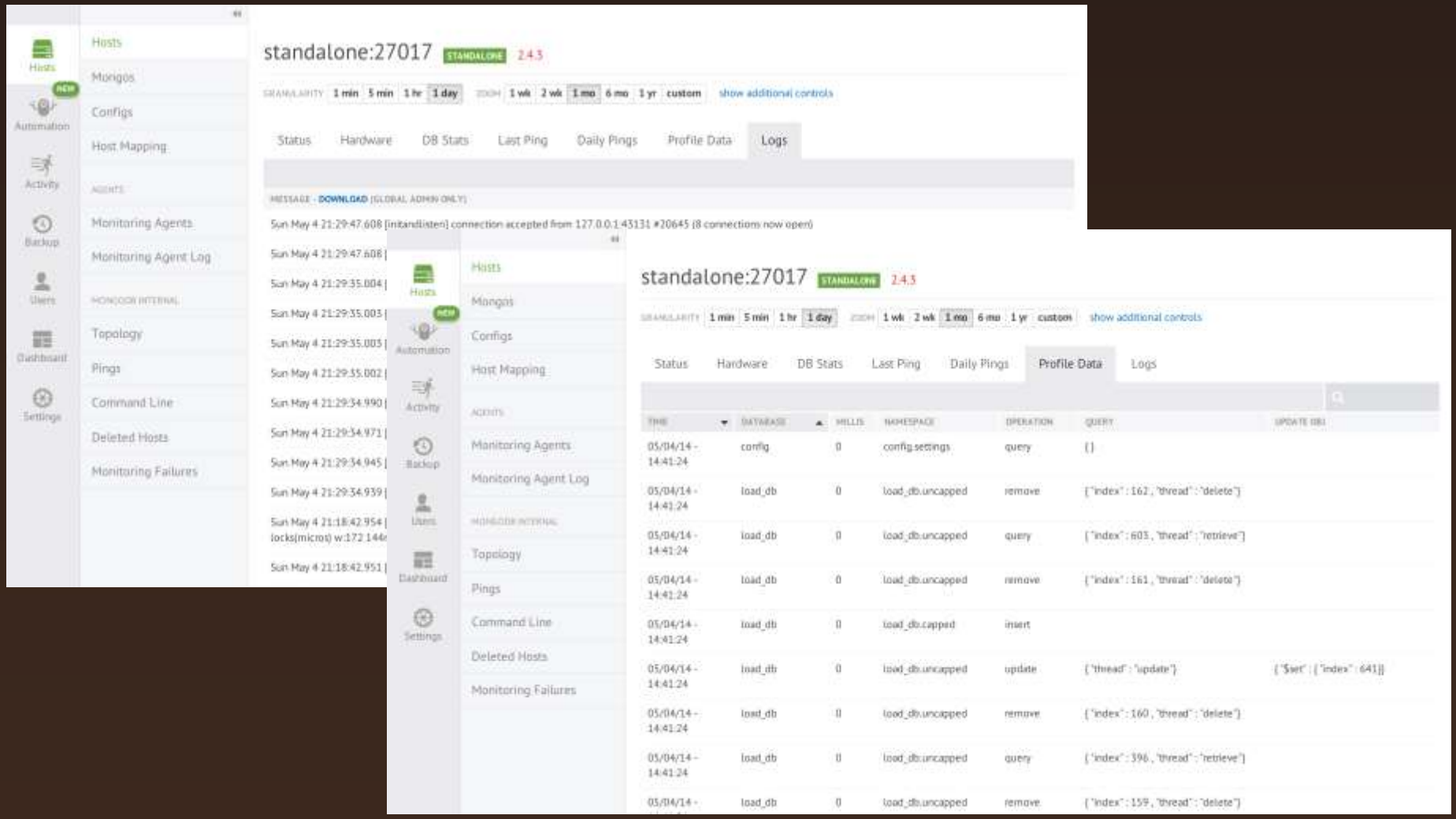
D. Database Stats



E. Hardware Stats (CPU, disk)



F. Logs and Profile Data



2. Performance tuning and monitoring

How to do performance tuning?

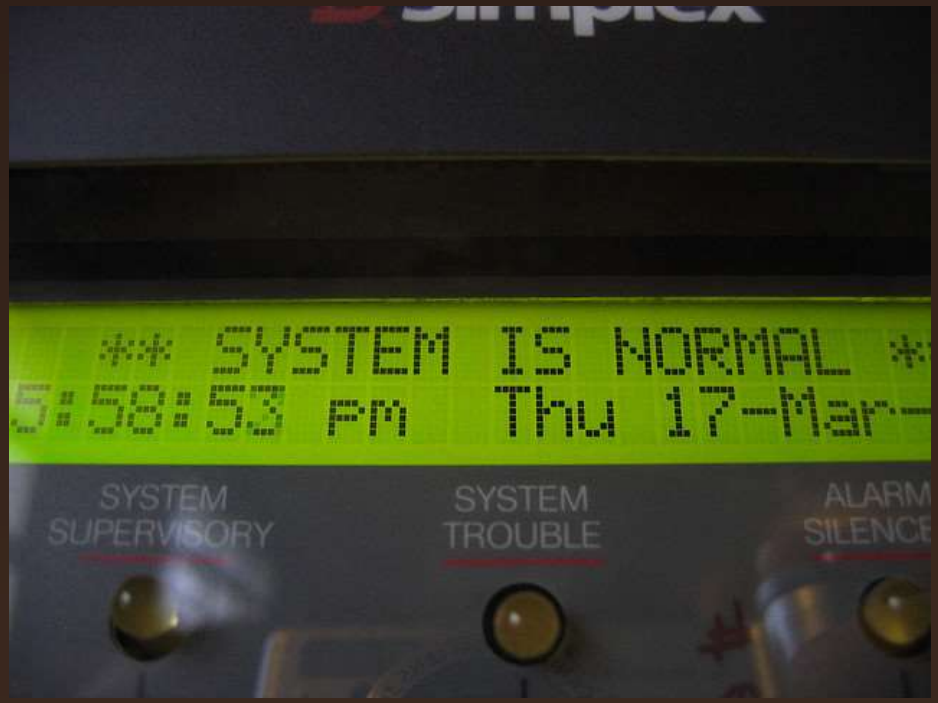
1. Assess the problem and establish acceptable behavior
2. Measure the current performance
3. Find the bottleneck*
4. Remove the bottleneck
5. Re-test to confirm
6. Repeat

* - (This is often the hard part)

(Adapted from http://en.wikipedia.org/wiki/Performance_tuning)

Pro-Tip: know thyself

You have to recognize normal to know when it isn't.



Source: http://www.flickr.com/photos/skippy/6853920/

Some handy metrics to watch

- Memory usage
- Opcounters
- Lock %
- Queues
- Background flush average
- Replication
  - Replication oplog window
  - Replication lag

Fun fact: oplog idempotency

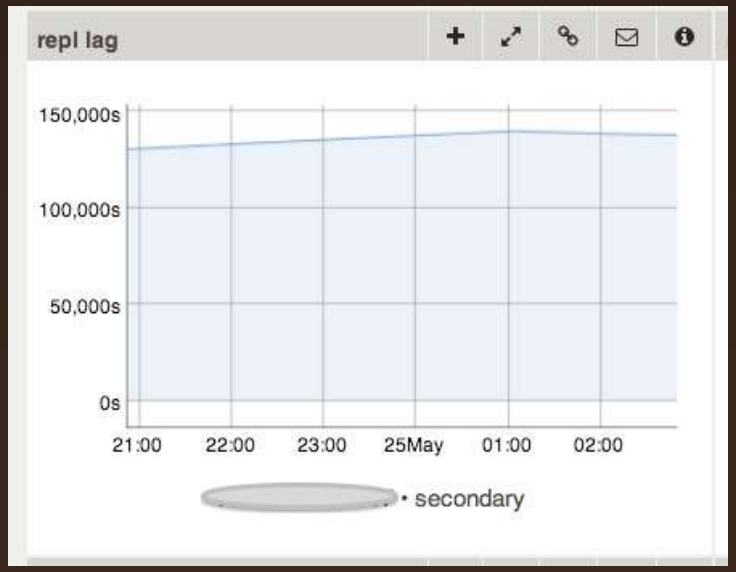
- Operations in the oplog only affect the value once, so they can be run multiple times safely.
- Examples
  - If you increment n from 2 to 3, n = 3 is fine; n + 1 is not.
  - Updating all documents that match a query is not, list of documents to update is.
- Frequent, large updates means a big oplog to sync.
- Updates that change a set mean writing the entire new version of the set to the oplog.

Example 1: replication lag

Scenario:

Customer reports 150,000s of replication lag.

Equals to almost 2 days of lag!



Example 1: replication lag

Some common causes of replication lag:

- Secondaries underspec'd vs primaries
- Access patterns between primary and secondaries
- Insufficient bandwidth
- Foreground index builds on secondaries

"…when you have eliminated the impossible, whatever remains, however improbable, must be the truth…" -- Sherlock Holmes

Sir Arthur Conan Doyle, The Sign of the Four

Example 1: replication lag

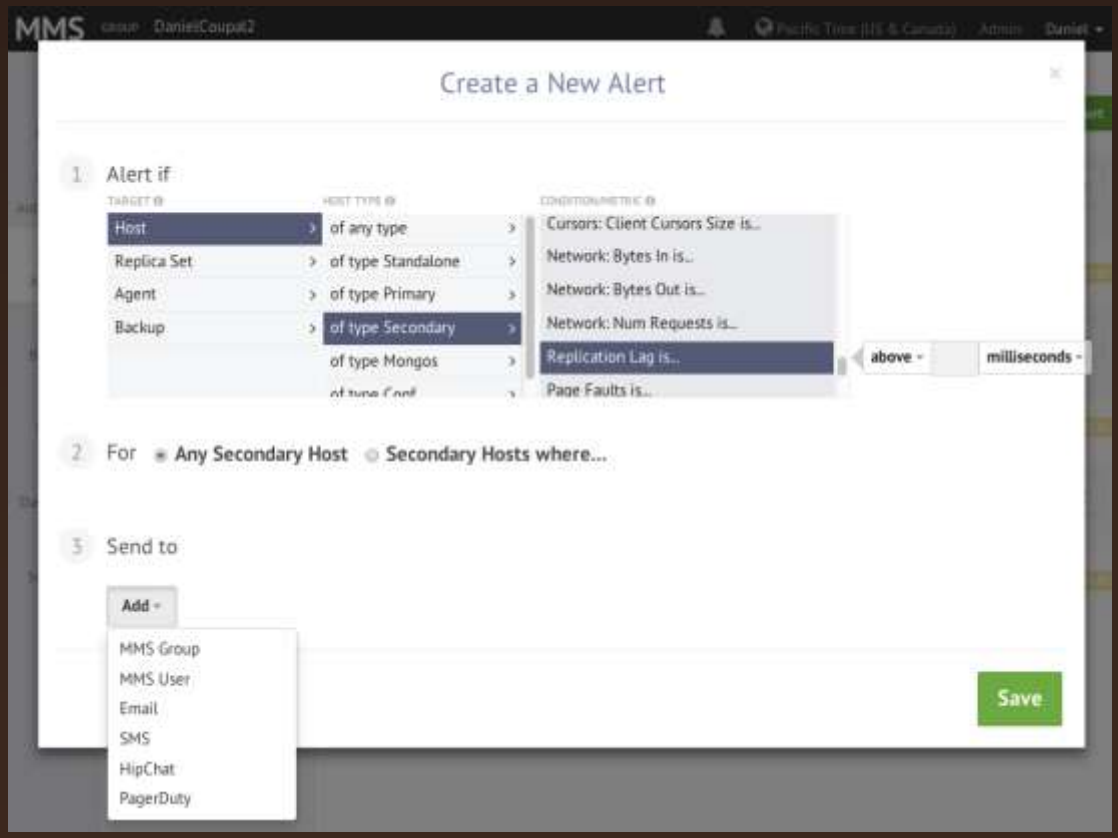
Example:

- ~1500 ops per minute (opcounters)
- 0.1 MB per object (average object size, local db)
- ~1500 ops/min / 60 seconds * 0.1 MB/op * 8b/B
  =~ 20 mbps required bandwidth

➢ Huge updates (oplog is idempotent) translated to 30 mbps, while they only had 10 mbps

Lesson: remember to use alerts!

Don’t wait until your secondaries fall off your oplog!



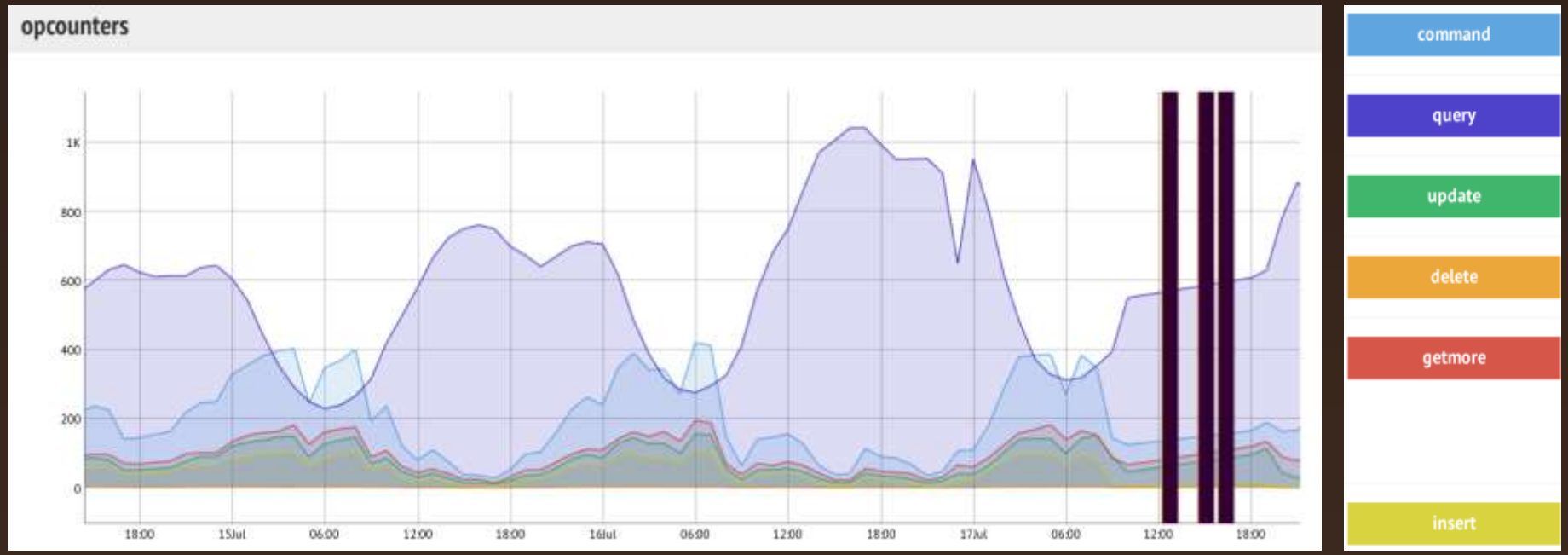
Example 2: slow performance

Scenario:

- User-facing web application with social functionality.
- Customer was seeing significant performance degradation after adding and removing an index from their replicaset.
- Their replicaset had 2 visible data-bearing nodes, each on real hardware, with dedicated 15K RPM disks and a significant amount of RAM.
- Why were things slow?

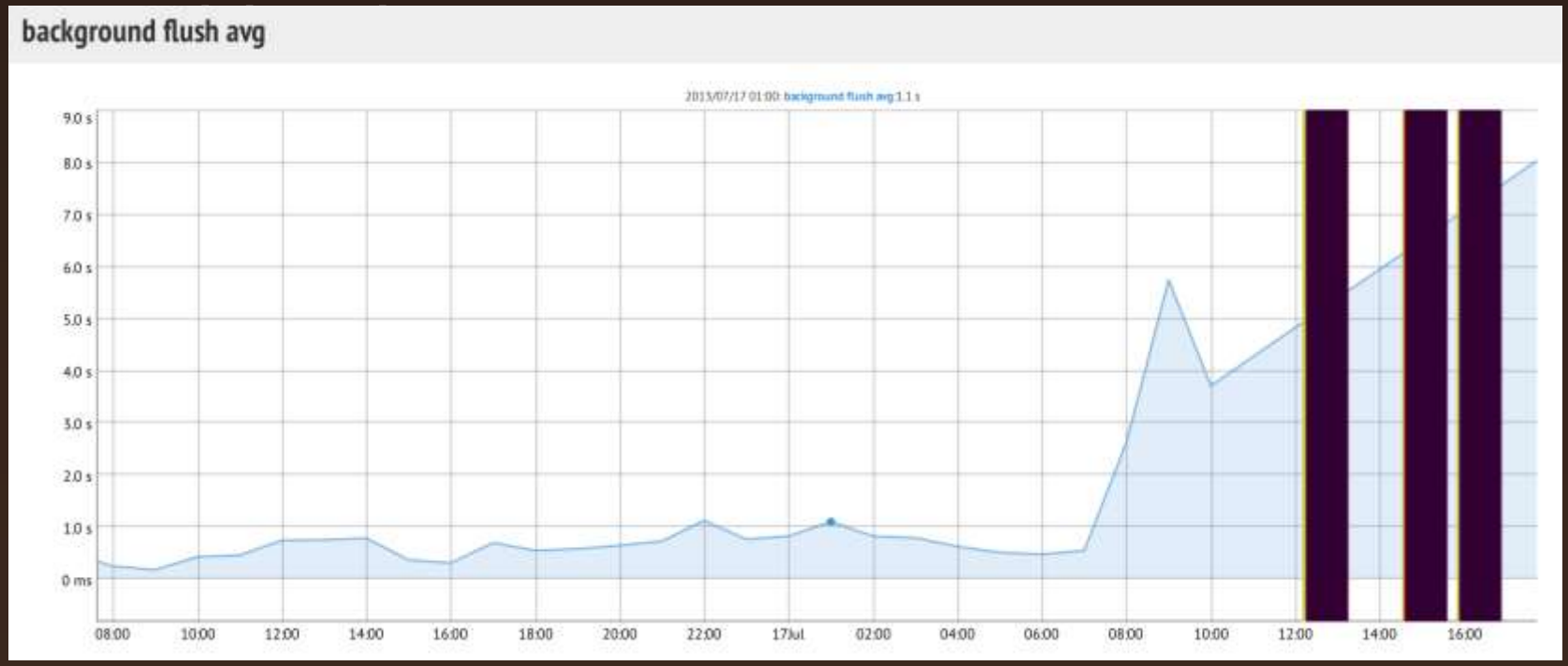
Example 2: slow performance

Opcounters: queries rose a bit, however writes were flat…



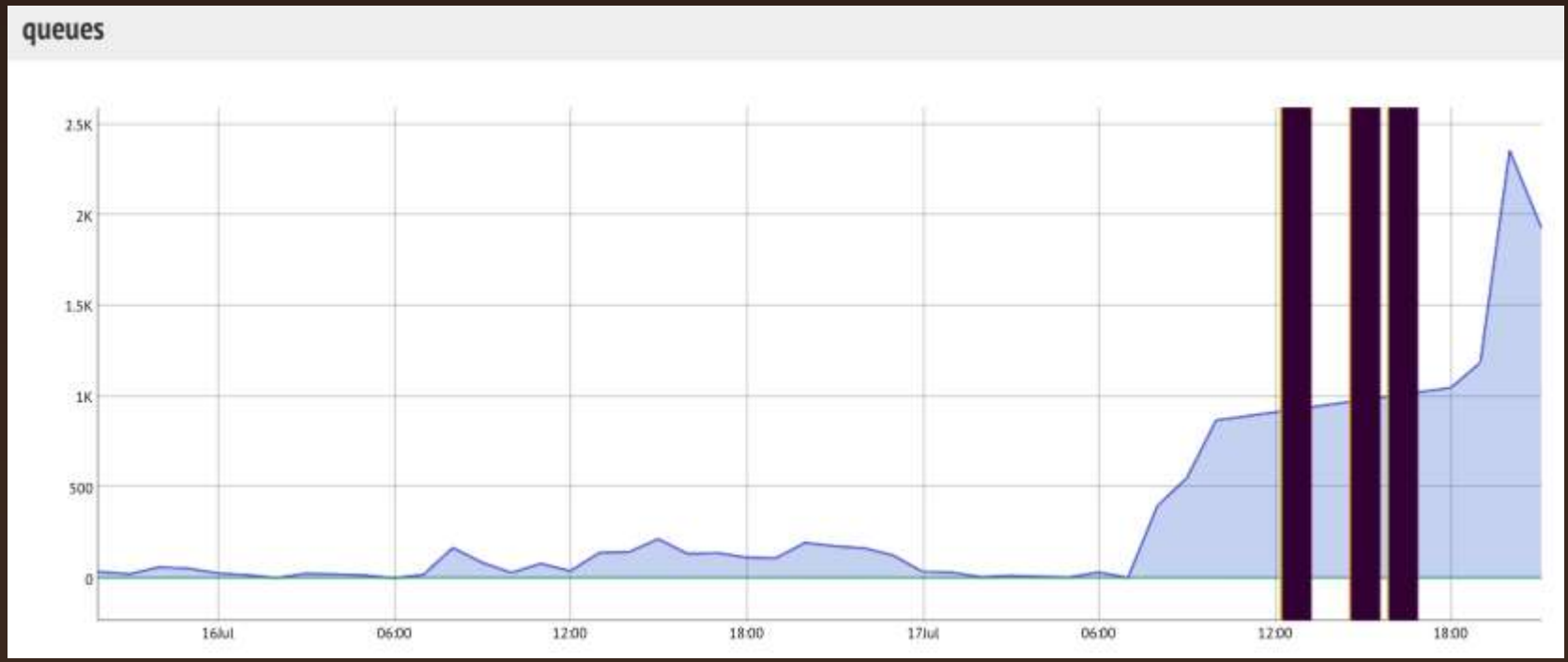
Example 2: slow performance

Background flush average: went up



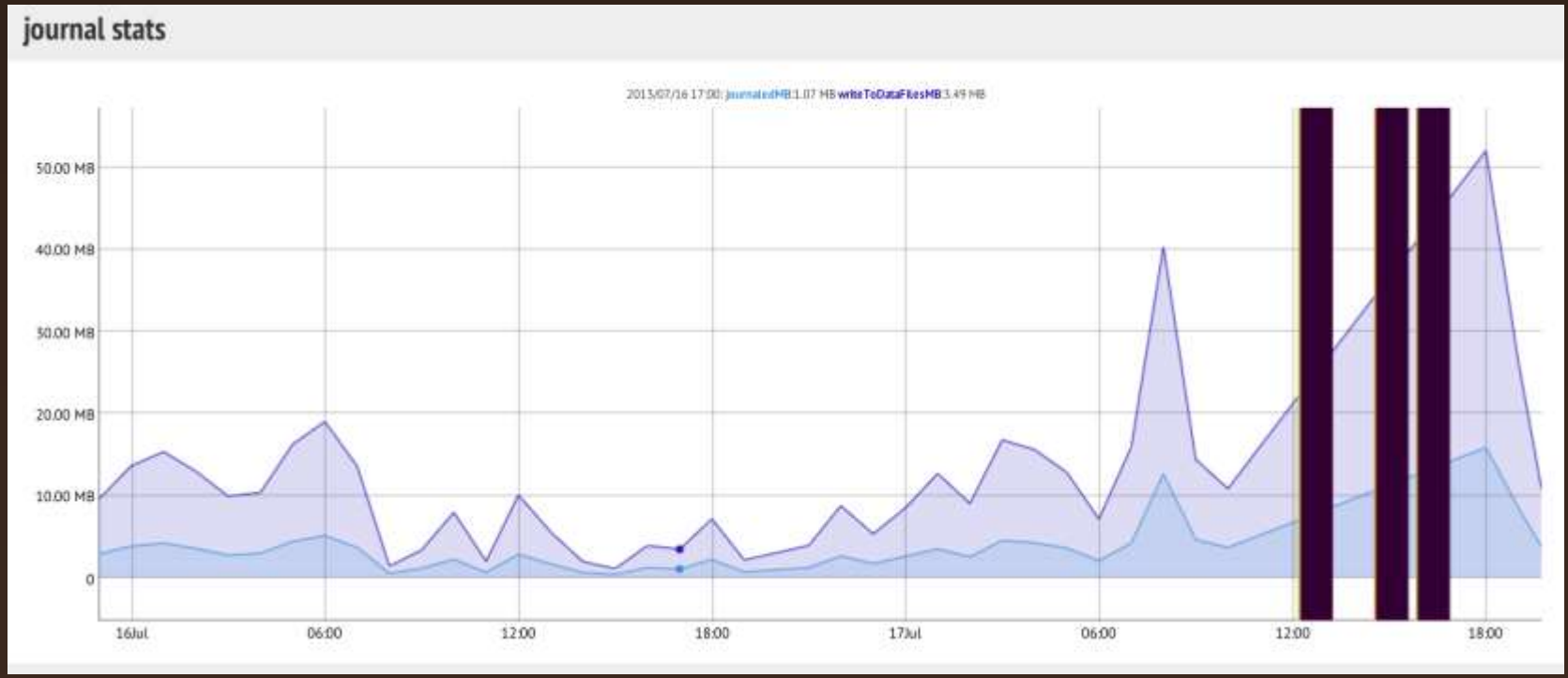
Example 2: slow performance

Queues: also went up considerably!



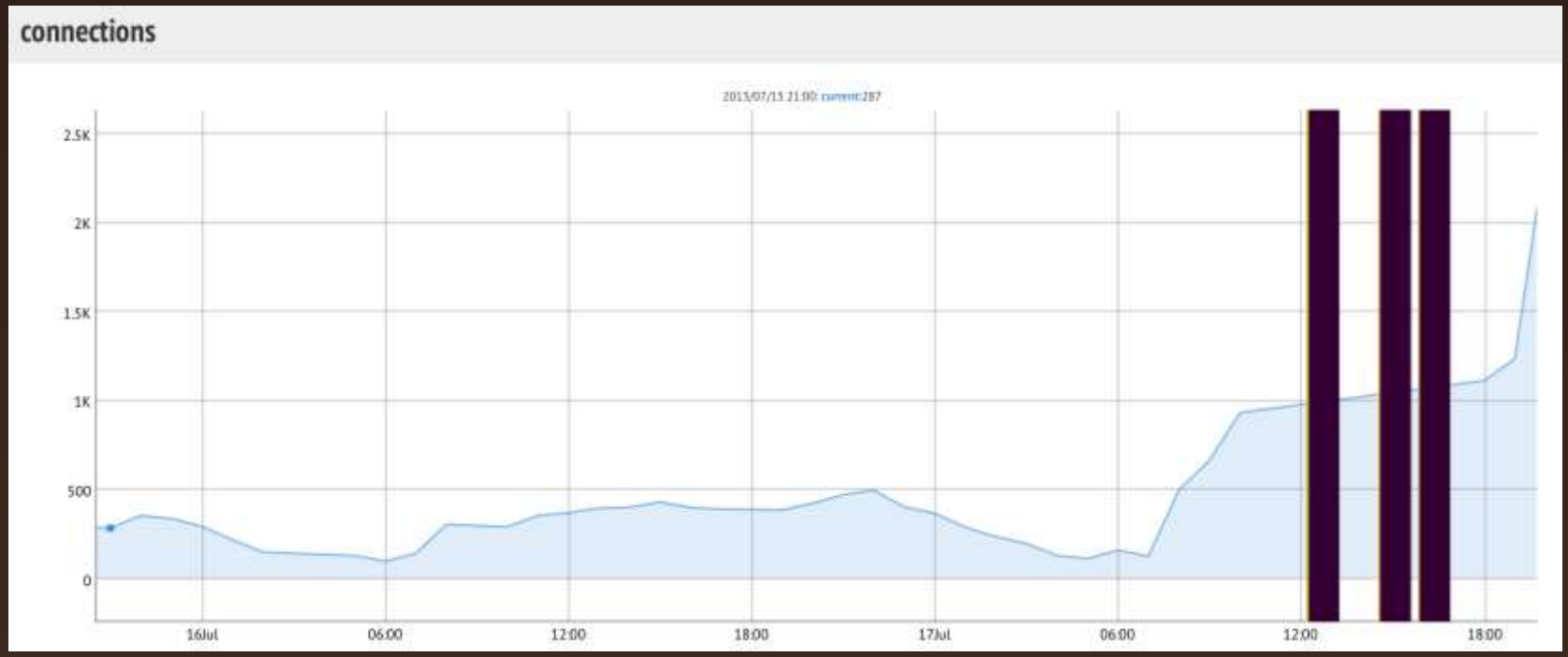
Example 2: slow performance

Journal stats: went up much higher than the ops…



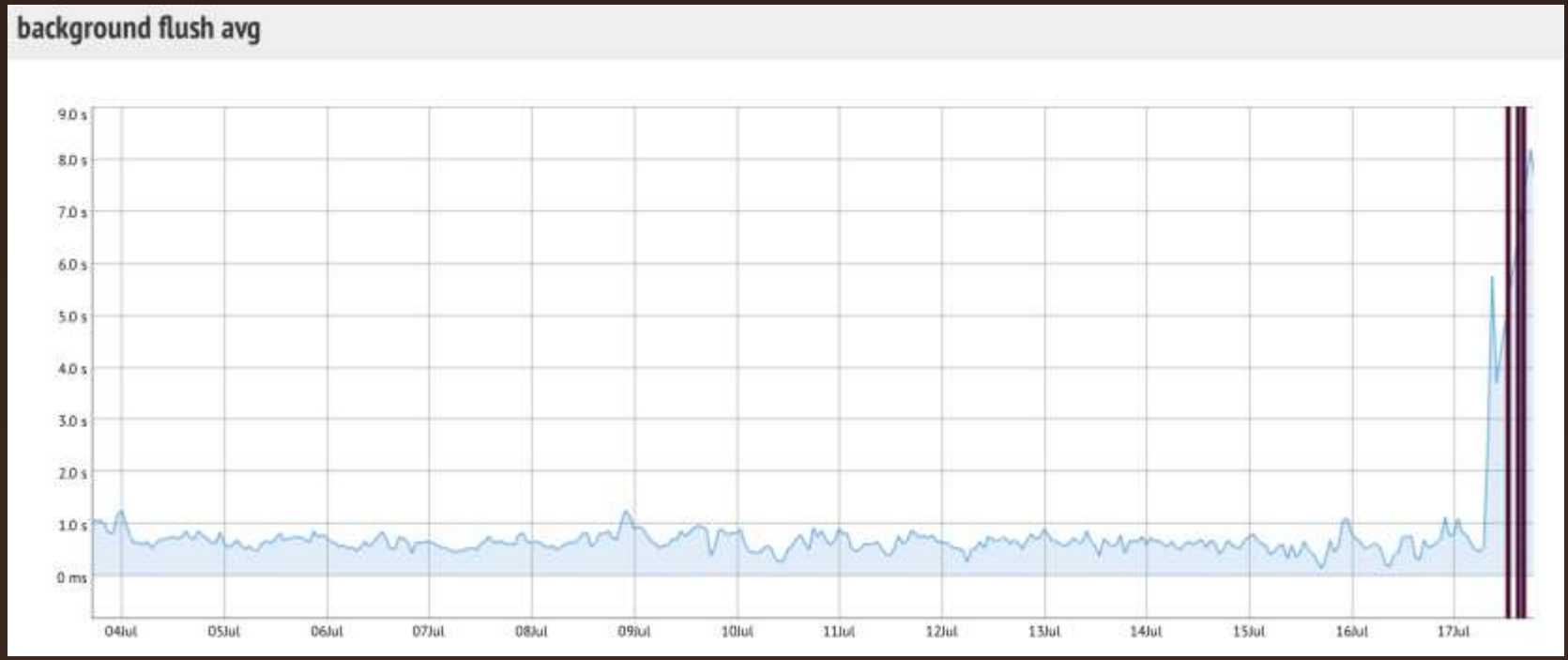
Example 2: slow performance

Connections: also went up…



Example 2: slow performance

Background flush average: consistent until then

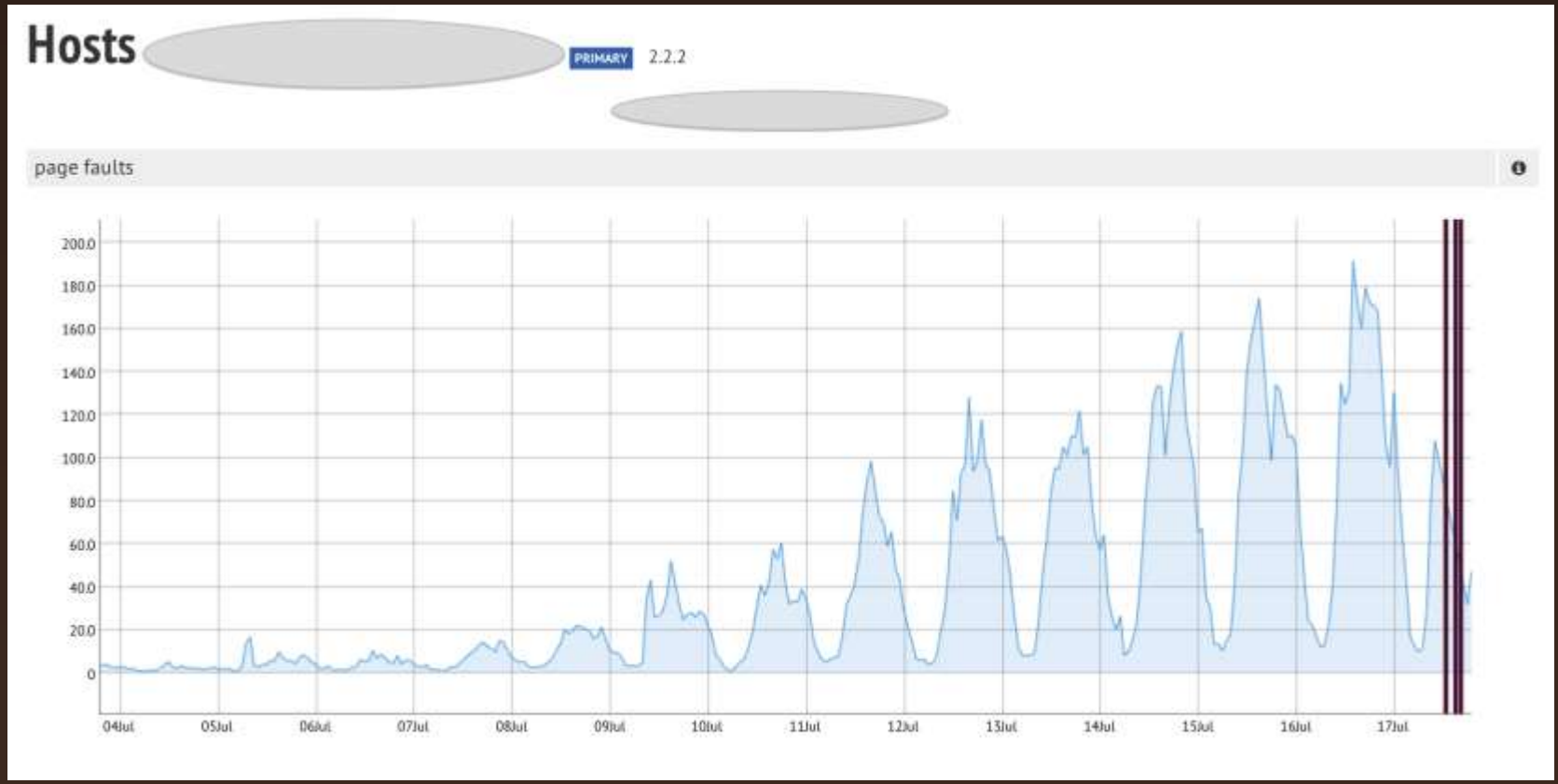


Example 2: slow performance

Opcounters: interesting… around July 9th

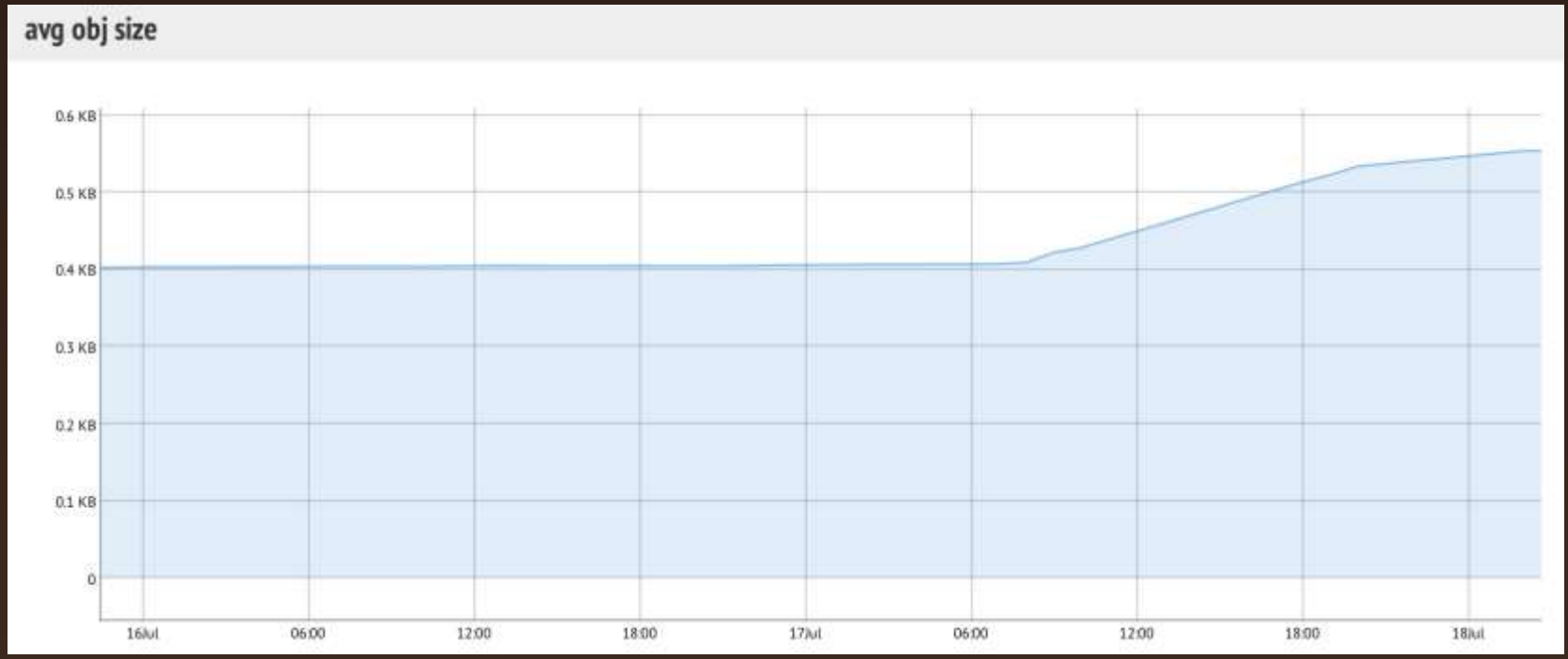
Example 2: slow performance

Page faults: something's going on!



Example 2: slow performance

Local DB average object size: growing!



Example 2: slow performance

- Now what?
- Time to analyze the logs
  - What query or queries were going crazy? And what sort of query would grow in size without growing significantly in volume?
- Remember:
  - Growing disk latency (caused by page faults?) And journal/oplog entries growing even though writes (inserts/updates) were flat.

Example 2: slow performance

Log analysis

The best tools for analyzing MongoDB logs are included in mtools*:

- mlogfilter
  filter logs for slow queries, collection scans, …
- mplotqueries
  graph query response times and volumes

* https://github.com/rueckstiess/mtools

Example 2: slow performance

Log analysis (example syntax)

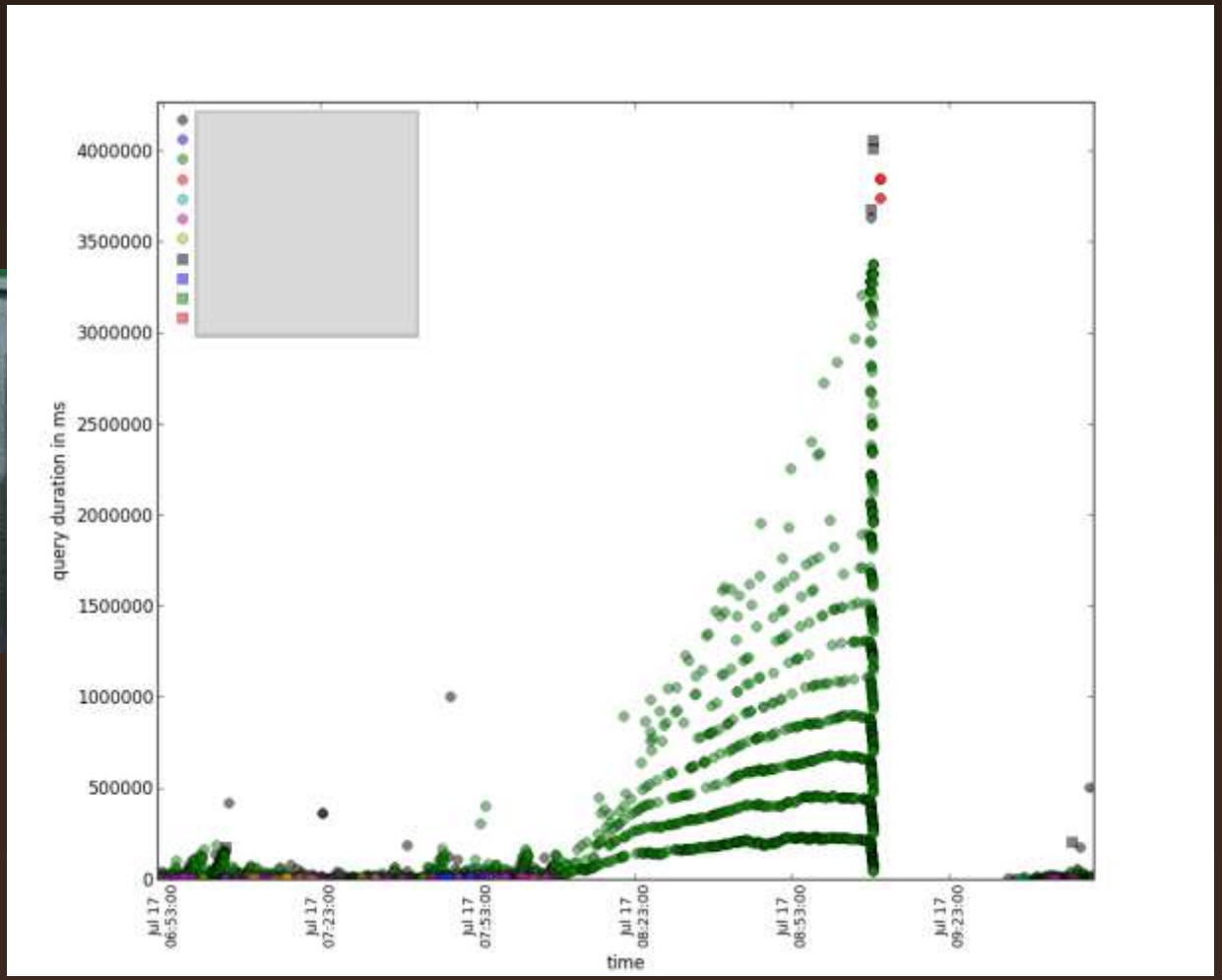
Show me queries that took more than 1000 ms from 6 am to 6 pm:

```
mlogfilter mongodb.log --from 06:00 --to 18:00 --slow 1000 > mongodb-filtered.log
```

Now, graph those queries:

```
mplotqueries --logscale mongodb-filtered.log
```

Example 2: slow performance



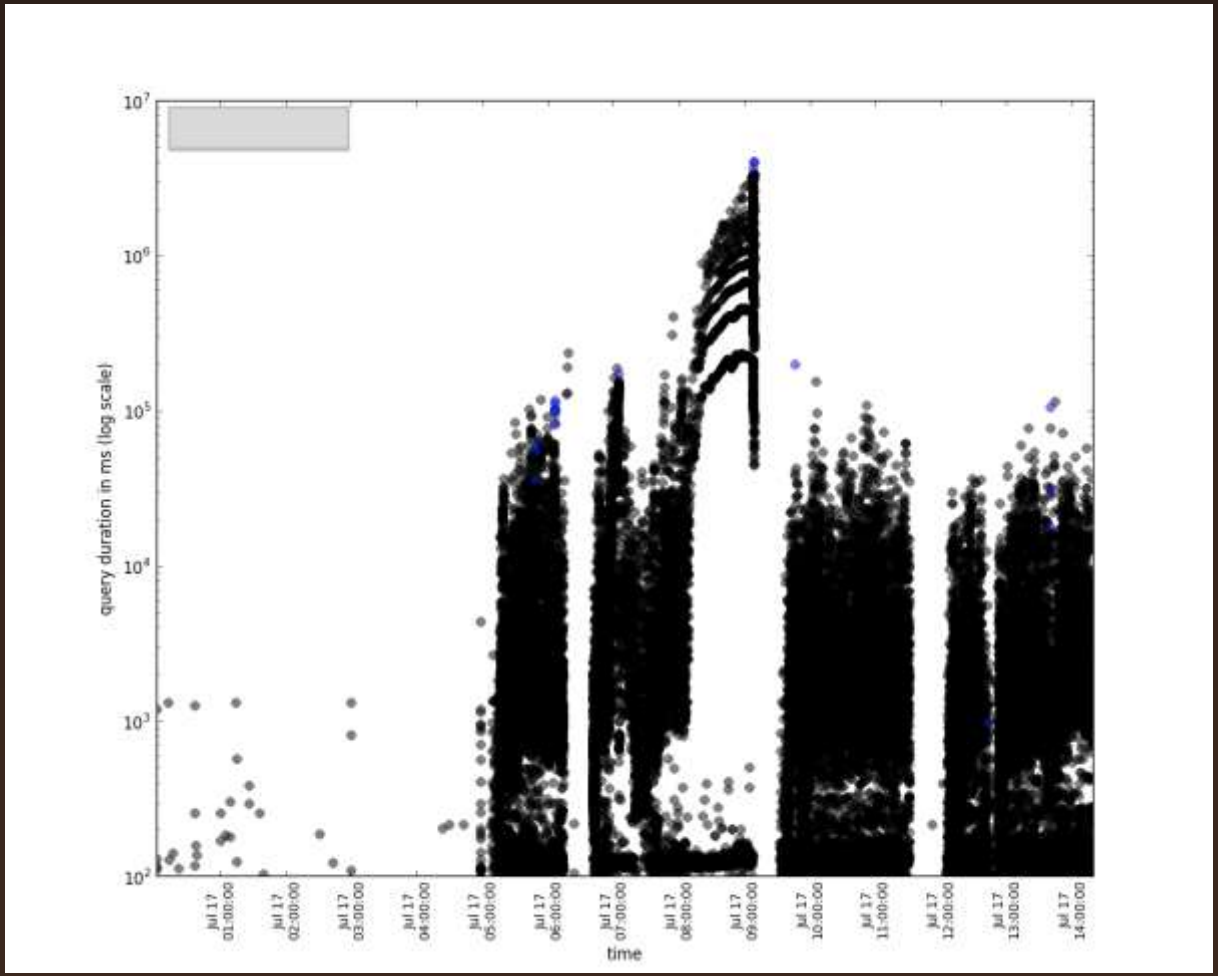
Example 2: slow performance

Filter more!

`--operation`

Logarithmic!

`--logscale`



Example 2: slow performance

Sample query

```
Wed Jul 17 14:16:44 [conn60560] update x.y query: { e:
”[id1]" } update: { $addToSet: { fr: ”[id2]" } }
nscanned:1 nupdated:1 keyUpdates:1 locks(micros) w:889
6504ms
```

6.5 seconds to add a single value to a set!

Example 2: slow performance

http://docs.mongodb.org/manual/reference/operator/addToSet/

The $addToSet operator adds a value to an array only if the value is not in the array already. If the value is in the array, $addToSet returns without modifying the array.

Consider the following example:

```
db.collection.update({field:value}, {$addToSet: {field:value1}});
```

Here, $addToSet appends value1 to the array stored in field, only if value1 is not already a member of this array.

Example 2: slow performance

https://jira.mongodb.org/browse/SERVER-8192

"`IndexSpec::getKeys()` finds the set of index keys for a given document and index key spec. It's used when inserting / updating / deleting a document to update the index entries, and also for performing in memory sorts, deduping `$or` clauses and for other purposes.

Right now extracting 10k elements from a nested object field within an array takes on the order of seconds on a decently fast machine. We could see how much we can optimize the implementation."

Example 2: slow performance

What else?!

- `Wed Jul 17 14:11:59 [conn56541] update x.y query: { e: ”[id1]" } update: { $addToSet: { fr: ”[id2]" } } nscanned:1 nmoved:1 nupdated:1 keyUpdates:0 locks(micros) w:85145 11768ms`

- Almost 12 seconds!
  This time, there’s “`nmoved:1`”, too.
  This means a document was moved on disk, it outgrew the space allocated for it.

Example 2: slow performance

Wait, there's more!

- `Wed Jul 17 13:40:14 [conn28600] query x.y [snip] ntoreturn:16 ntoskip:0 nscanned:16779 scanAndOrder:1 keyUpdates:0 numYields: 906 locks(micros) r:46877422 nreturned:16 reslen:6948 38172ms`

- 38 seconds! Scanned 17k documents, returned 16.

Example 2: slow performance

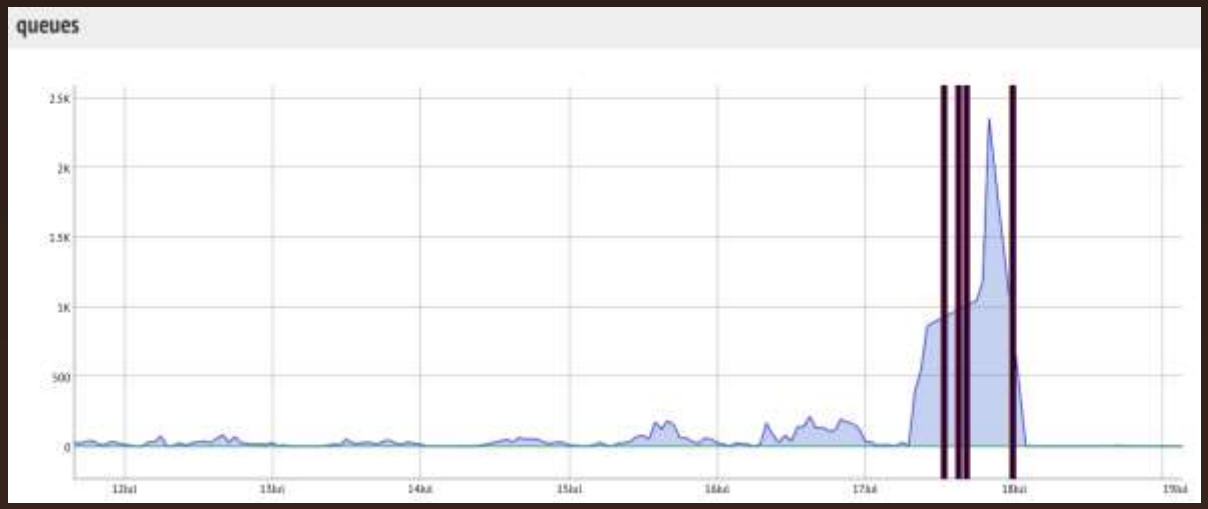
What next?

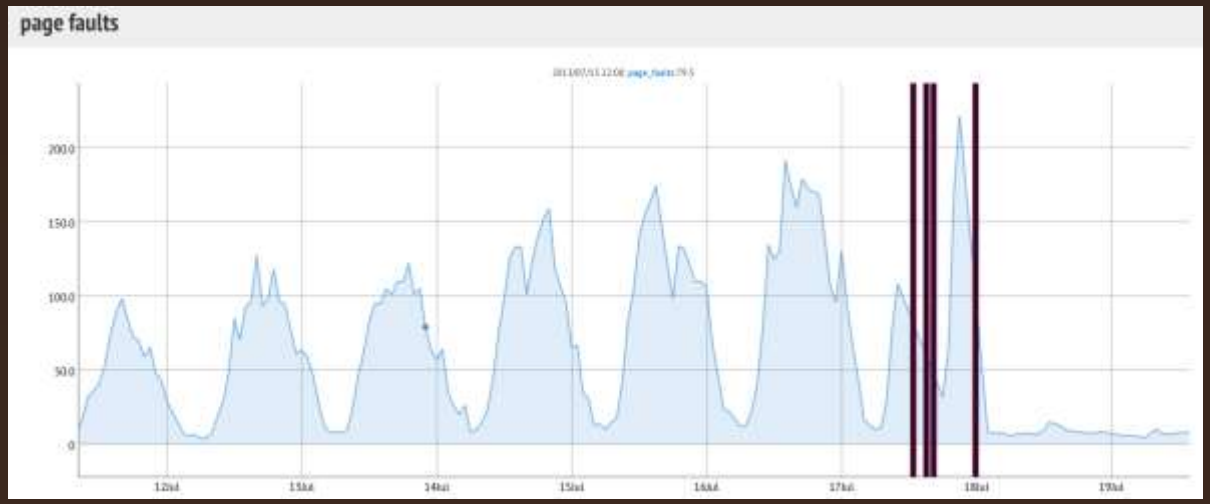
Short term fix: disable the new feature for the heaviest users! After that:

- rework the code to avoid `$addToSet`
- add indexes for queries scanning collections
- use powerOf2Sizes (*http://docs.mongodb.org/manual/reference/command/collMod/*) to reduce fragmentation and document moves

Example 2: slow performance

Did it work?





(Yes.)

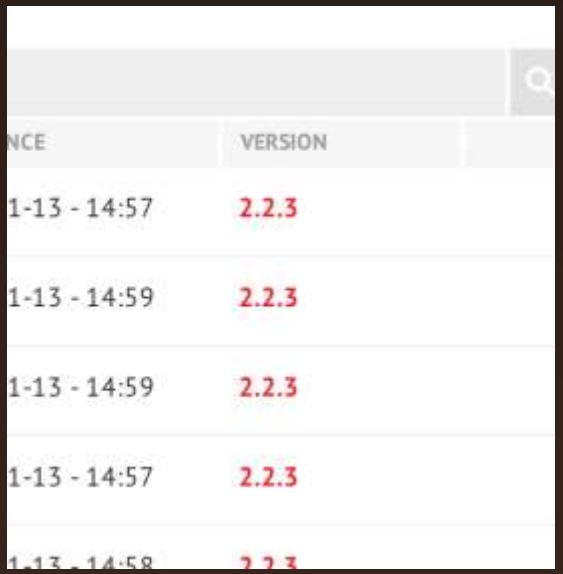
(So far. ;))

Monitoring: watch for warnings

MMS warns you if your systems

a) are running outdated versions
b) have startup warnings
c) if a *mongod* is publicly visible

Don’t ignore these warnings!



3. Setting it up and getting around

Setting up monitoring for the Cloud

http://mms.mongodb.com/help/monitoring/tutorial/

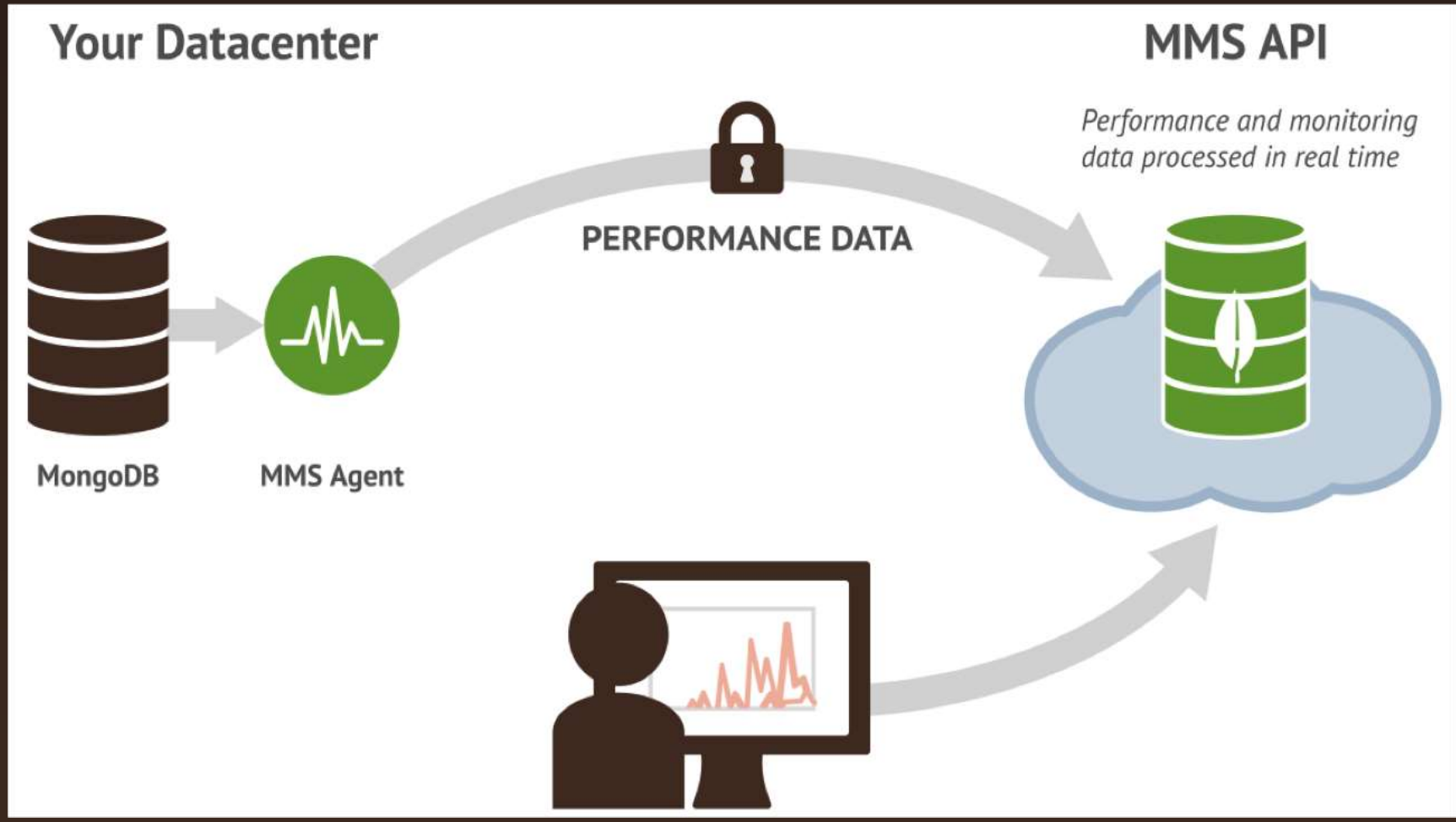
- Setup an account (Free)
- Install the agent
  - Download the binary
  - Set the API key in the configuration file
- Add your hosts
  - Add a seed host, MMS will discover the cluster
- Optional: hardware stats through Munin-node
- Optional: enable logging and profiling

Setting up monitoring “On-Prem”

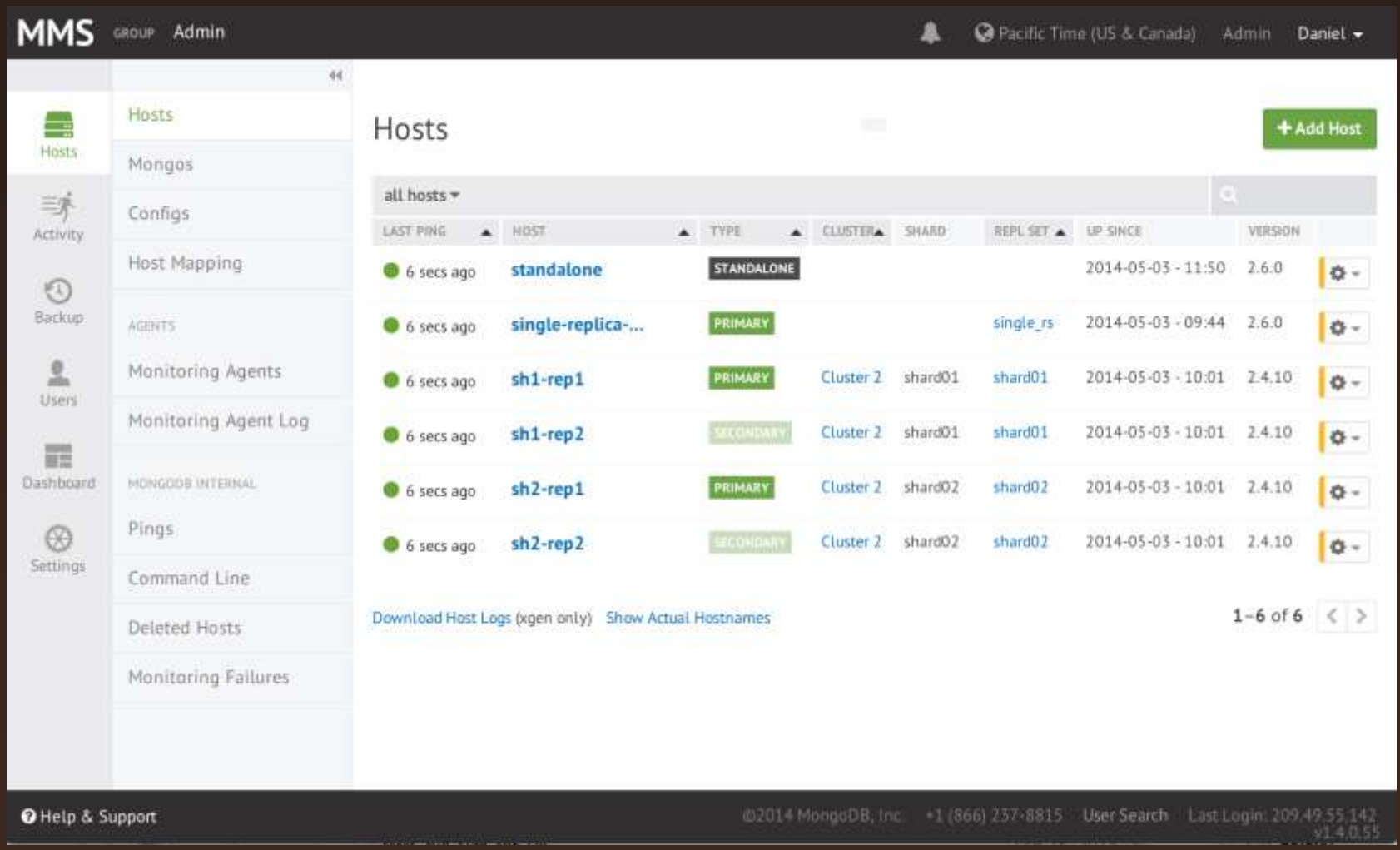
- Available (included) to Customers with
  - Enterprise subscription
  - OR Standard subscription
- One host to run the server for up to 400 hosts
- One RPM/Deb package to install

- Installing the backup requires more planning and more resources (hosts, disks, …)
  - The alternative is use MMS Backups in the Cloud, we removed the complexity for you.

MMS Monitoring Architecture



List of monitored hosts



Notes

- Agent written in Go
- Failover: run multiple agents (1 primary)
- Hosts: use CNAMEs, especially on AWS!
- You can use a group per environment (each needs an agent)
- Connections are over SSL
- On-Prem solution for Enterprise or Standard customers that don’t want to use the hosted service
- Makes it easier for the technical services to help you!

4. Wrapping up

The Technical Services Team

- We are here to make you successful, open tickets
  - We support 1000s of customers
  - Another customer may had the same issue as you
- Throw any problems to the Technical Services Team

What's next for MMS?

- Continuous updates
  - Cloud: ~every 3 weeks
  - On-Prem: ~every 3 months
- Automation service
- A lot of more information at: MongoDB World 2014 @ NYC on June 23-25

Summary

- MMS is a great, free service
- Setup is easy
- Metrics and graphs are awesome
- Preventing failures even more awesome
- Using MMS makes it easier for the Technical Services Team to help you
- Monitor your DBs like a Pro, use MMS

5. Questions?

Appendix

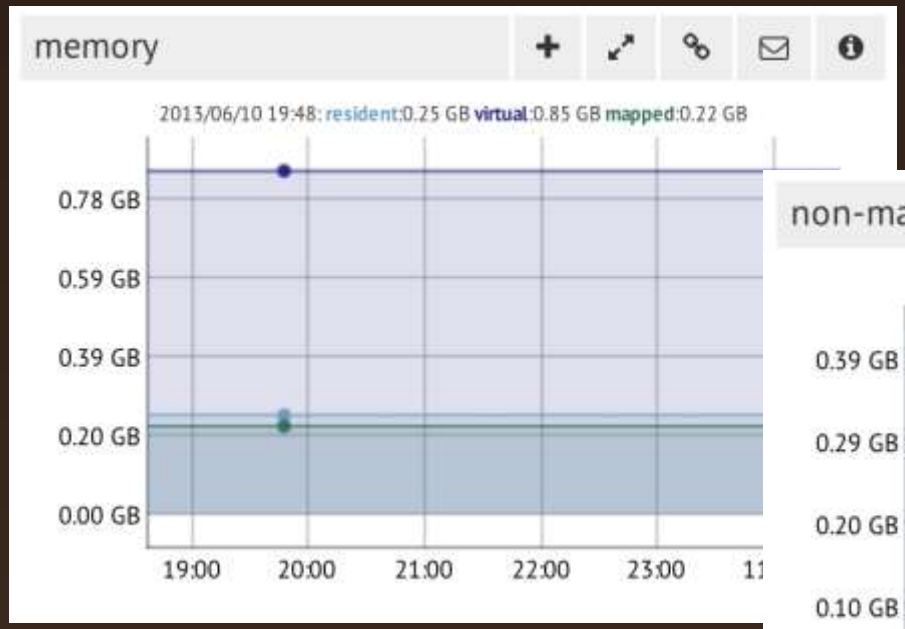
Other MMS References

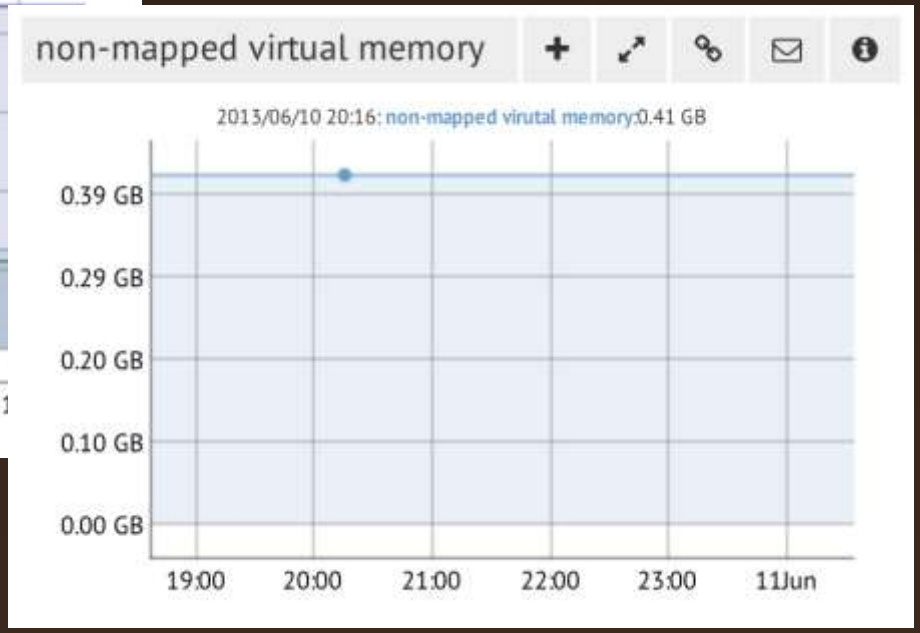
- Tons of webinars and other presentations
  http://www.mongodb.com/presentations

- Five MMS Monitoring Alerts to Keep Your MongoDB Deployment on Track
  http://www.mongodb.com/blog/post/five-mms-monitoring-alerts-keep-your-mongodb-deployment-track
- Digging into the meaning of some metrics
  - Lock % : http://blog.mms.mongodb.com/post/78650784046/learn-about-lock-percentage-concurrency-in-mongodb
  - Replication Oplog Window : http://blog.mms.mongodb.com/post/77279440964/replica-set-health-is-more-than-just-replication-lag
- How to fix some warnings reported by MMS
  - http://docs.mongodb.org/manual/administration/production-notes/
    (includes readahead settings)

Examining memory and disk

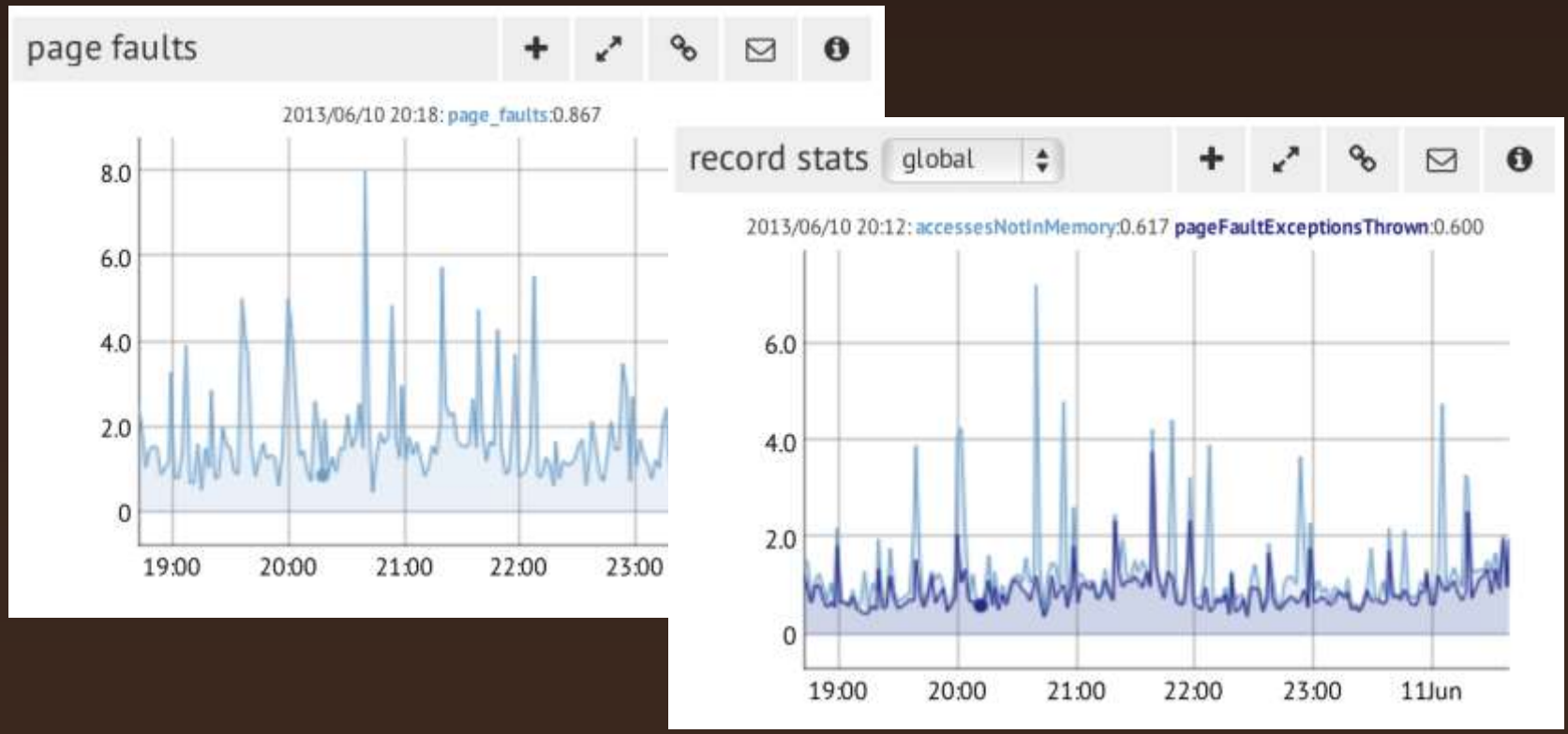
Memory: resident vs virtual vs non-mapped





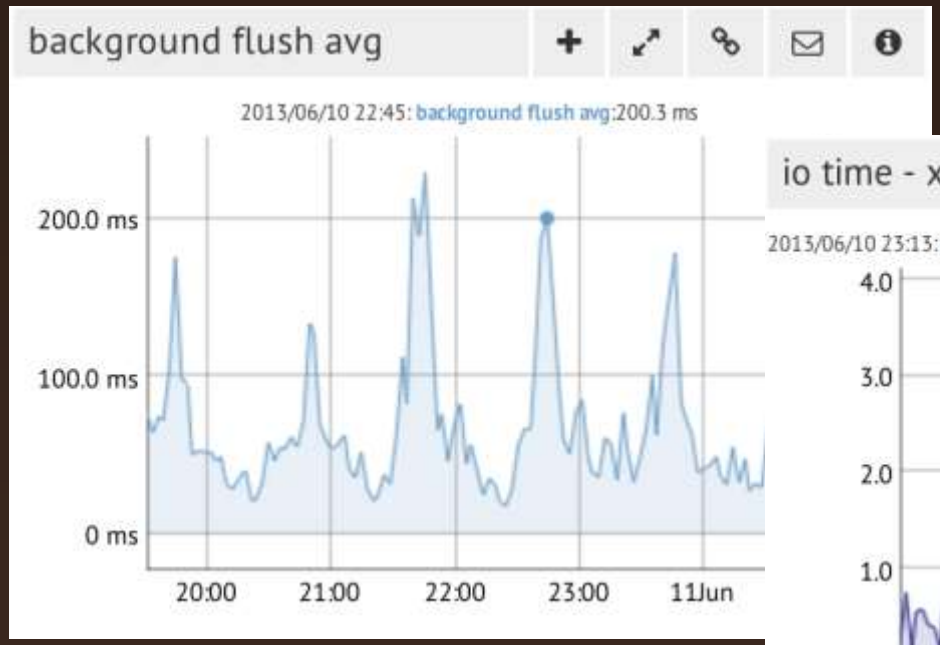
Examining memory and disk

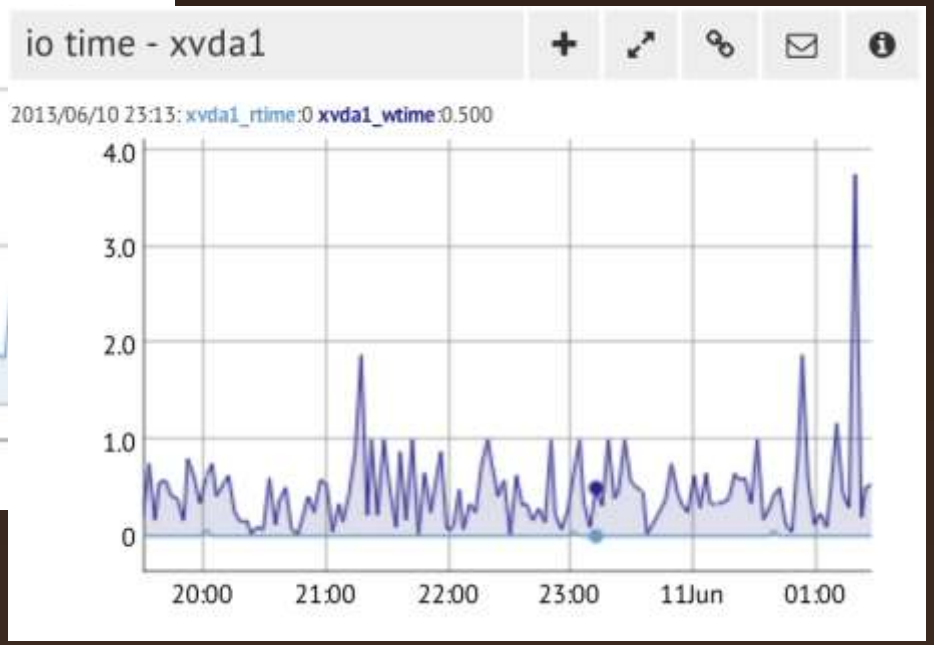
Page faults and Record Stats



Examining memory and disk

Background flush and Disk IO





Thank you for using