oVirt

Huge Codebases
Application Monitoring with Hystrix

30 Jan. 2016

Roman Mohr
Red Hat

FOSDEM 2016

About Me

oVirt

Roman Mohr

Software Engineer at Red Hat

Member of the SLA team in oVirt

Mail: rmohr@redhat.com

Github: https://github.com/rmohr

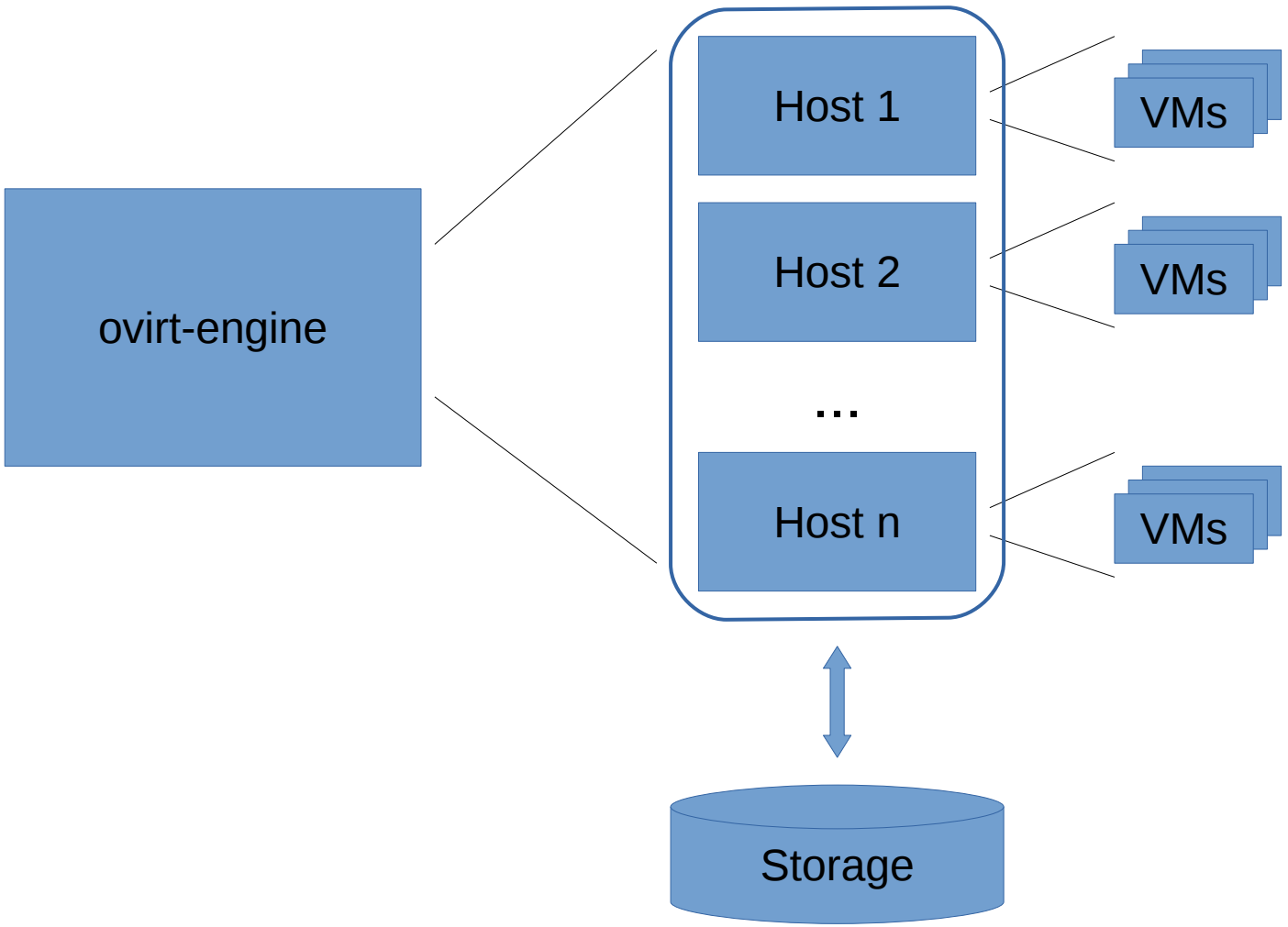
IRC: #ovirt irc.oftc.net

oVirt

oVirt

"oVirt is a powerful virtual machine manager for up to datacenter-class deployments, and provides an awesome KVM management interface for multi-node virtualization." – http://www.ovirt.org

oVirt Artitecture

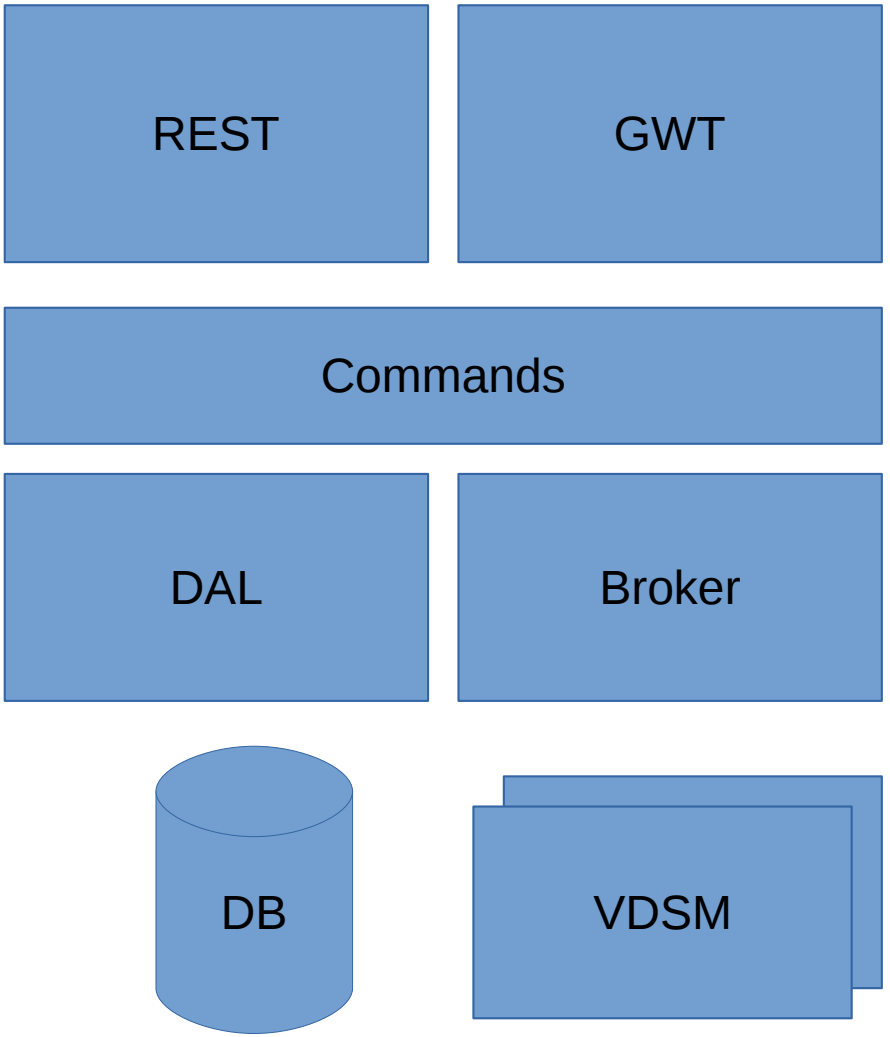


ovirt-engine

The Beast

Take 1

Architecture of ovirt-engine



Git Statistics of ovirt-engine

oVirt

Branch: **master**

Generated: 2016-01-14 14:02:53 (in 370 seconds)

Generator: GitStats (version 2014-12-09), git version 2.4.3, gnuplot 5.0 patchlevel 0

Report Period: 2011-10-04 18:43:09 to 2025-03-31 23:18:53

Age: 4928 days, 1449 active days (29.40%)

Total Files: 10355

Total Lines of Code: **1123168** (**2557376 added**, **1434208 removed**)

Total Commits: **20166** (average 13.9 commits per active day)

Authors: 174 (average 115.9 commits per author)

Issues we have

oVirt

- Many developers
- A lot of code
- No second level cache
- REST performance problems
- The product runs at the user/customer site
- Test coverage
- Hard to configure and run the application

Where to start?

oVirt

- Try to get a high level overview of the architecture
- “Data-mine your Source Control” – Greg Young*
- Gather code metrics (JArchitect, Sonar)
- Monitor your application **before** you change something

* How to get productive in a project in 24h

https://www.youtube.com/watch?v=KaLROwp-VDY

Java and Application Monitoring

Profiler?

oVirt

- You can see where your application spends its time
- Easy to get started. Just connect to the JVM in question and browse the CPU profiling graph.
- Some profilers even support JDBC, JPA, …

But:

- In general no application logic specific insights
- Many are closed source
- Not easy to collect data

XRebel for Monitoring?

oVirt

- Easy to integrate. Just start an additional Java agent
- Every HTTP servlet now contains an additional popup where you can access application metrics.

But:

- Closed source
- Development only

NewRelic for Monitoring?

oVirt

- Excellent visualization
- Supports multi-host applications
- Knows a lot about Java

Examples:

- Closed source
- Production only
- License based business model

Hystrix

oVirt

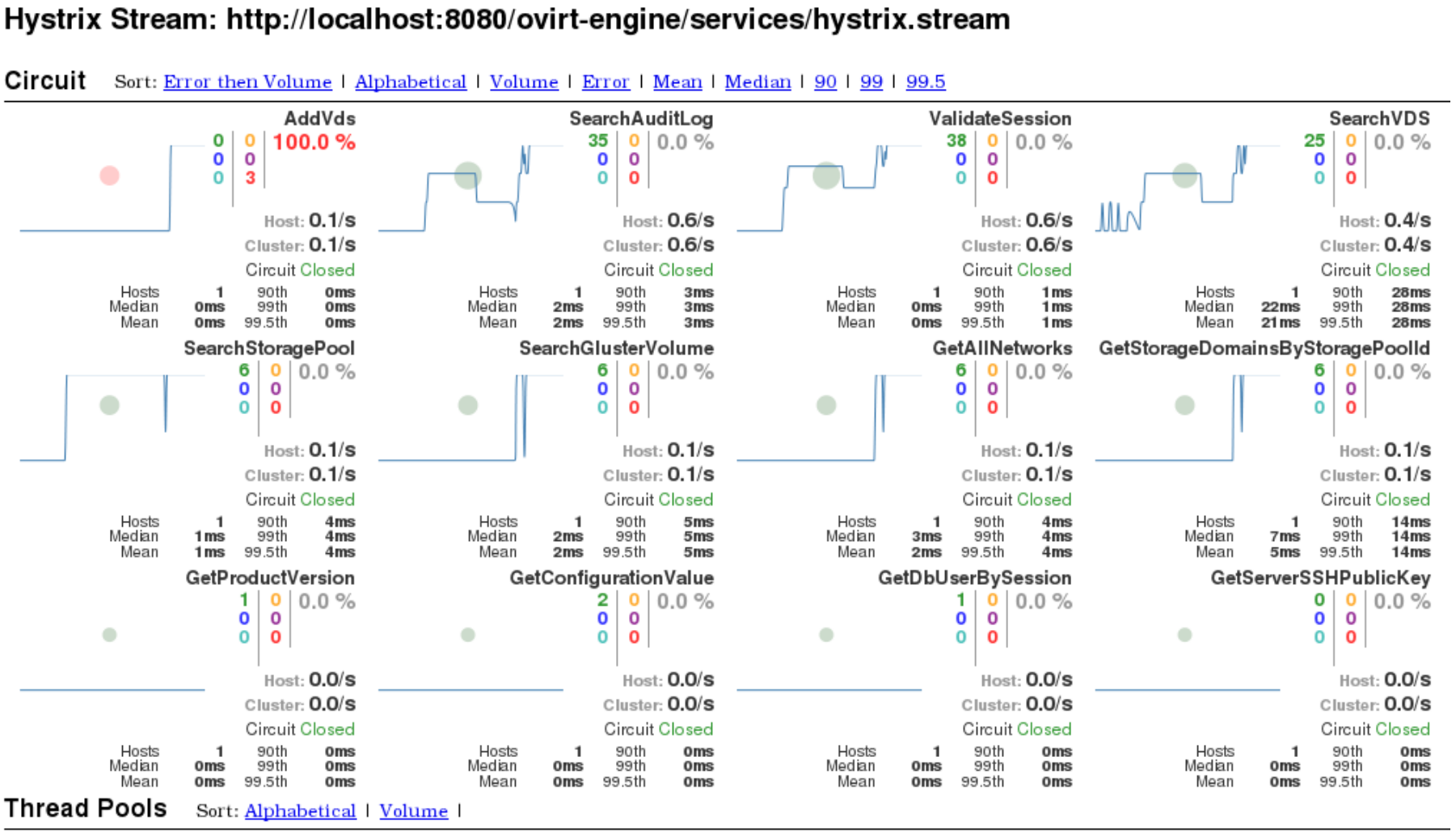
"Hystrix is a latency and fault tolerance library designed to isolate points of access to remote systems, services and 3rd party libraries, stop cascading failure and enable resilience in complex distributed systems where failure is inevitable." – https://github.com/Netflix/Hystrix

Hystrix

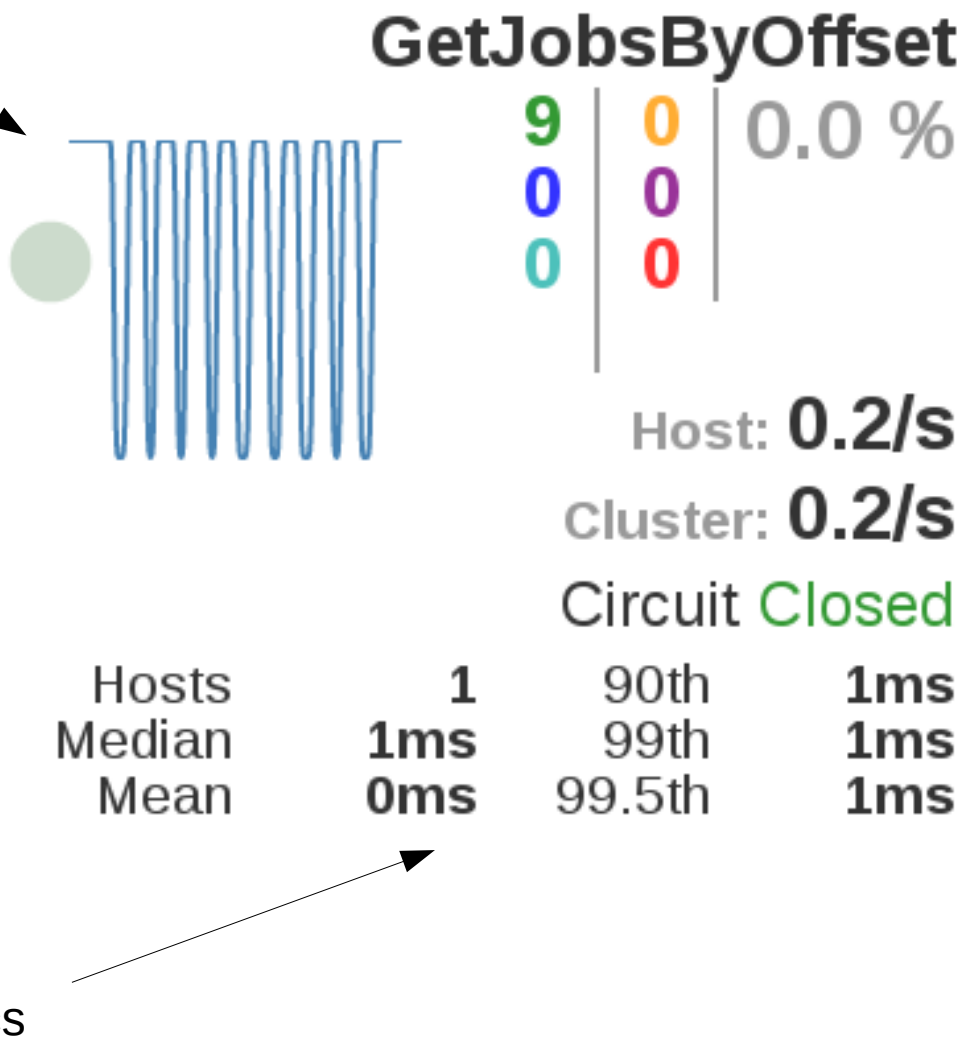
oVirt

Hystrix also provides metrics!

Hystrix Dashboard



Hystrix Dashboard



Request shape

Error percentage

Requests/s

Circuit breaker status

Statistics

Hystrix Dashboard

oVirt

Successful	9	0	Timeouts (thread isolation)
Rejected (Short circuit)	0	0	Rejected (max. concurrent invocations)
Bad request (exception)	0	0	Failed executions (exception)

Hystrix Dashboard

oVirt

Easy to run

```
$ git clone https://github.com/Netflix/Hystrix.git
$ cd Hystrix/hystrix-dashboard
$ ../gradlew jettyRun
> Running at http://localhost:7979/hystrix-dashboard
```

Easy to integrate

- Drop the WAR from maven central in your container
- Add the WAR as dependency and serve the resources folder on an endpoint.
- Add the *hystrix-metrics-event-stream* servlet to your application

Hello World Hystrix Command

oVirt

```
Setter setter = Setter.withGroupKey(
        HystrixCommandGroupKey.Factory.asKey("helloWorld")
).andCommandKey(
        HystrixCommandKey.Factory.asKey("helloWorld")
);


HystrixCommand<String> helloWorldCommand =
new HystrixCommand<String>(setter) {
    @Override protected String run() throws Exception {
        return "Hello world!";
    }
};


return helloWorldCommand.execute();
```

ovirt-engine

The Problem

Take 2

Problem description

oVirt

- We have a datacenter with 1000 VMs.
- We query the /api/vms endpoint which returns all VMs.
- We need 2.5 seconds to fetch them with no additional load.

- We have a datacenter with 2000 VMs.
- We query the /api/vms endpoint which returns all VMs.
- We need 5 seconds to fetch them with no additional load.

Solution: Let's curl a little bit

oVirt

1000 VMs, 1 request

```
$> time bash rest.sh vms > /dev/null
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 7043k    0 7043k    0     0  2774k      0 --:--:--  0:00:02 --:--:-- 2775k

real 0m2.547s
user0m0.008s
sys  0m0.011s
```

2000 VMs, 1 request

```
$> time bash rest.sh vms > /dev/null
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 13.7M    0 13.7M    0     0  2876k      0 --:--:--  0:00:04 --:--:-- 3815k

real 0m4.900s
user0m0.009s
sys  0m0.014s
```

Solution: Let's curl a little bit

oVirt

2000 VMs, 10 parallel requests

```
$> time seq 1 10 | parallel -j 10 bash rest.sh vms > /dev/null

  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 13.7M    0 13.7M    0     0   488k      0 --:--:--  0:00:28 --:--:-- 3911k
[...]
100 13.7M    0 13.7M    0     0   487k      0 --:--:--  0:00:28 --:--:-- 3848k

real 0m29.590s
user0m0.212s
sys  0m0.438s
```

Solution: We can guess

oVirt

- “That's because our database is so slow.”
- “The database can cache everything, it is because our REST application code is so slow.”
- “That's because we are keeping the database busy with status updates of Hosts and VMs.”
- “That's because our architecture is not smart enough, it is just an ordinary monolith. That must be solved with streaming and eventbuses.”

Solution: Let's curl a little bit more

oVirt

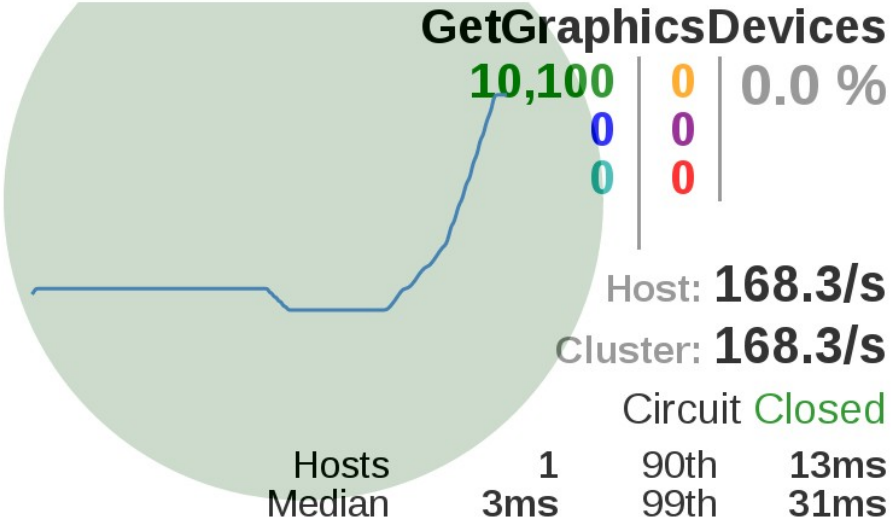
Let us execute the following scenario:

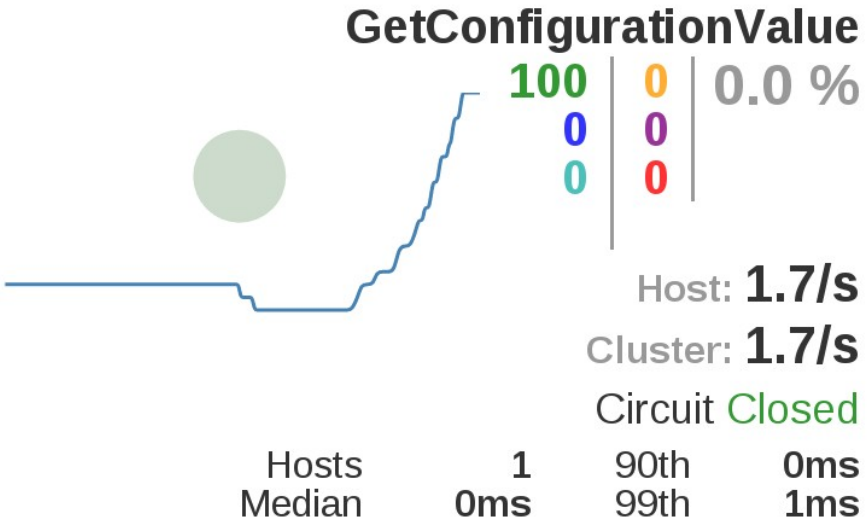
100 Vms, 10 parallel requests, 100 requests total

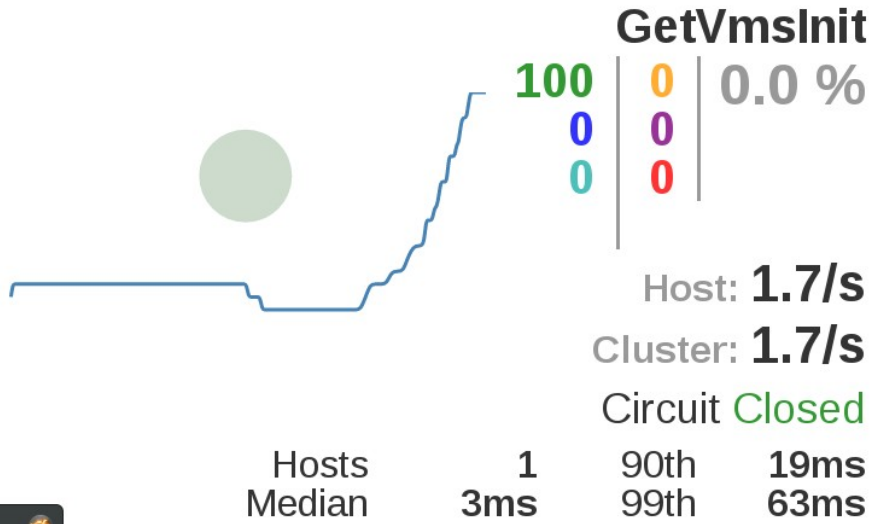
```
$> seq 1 100 | parallel -j 10 bash rest.sh vms > /dev/null
```

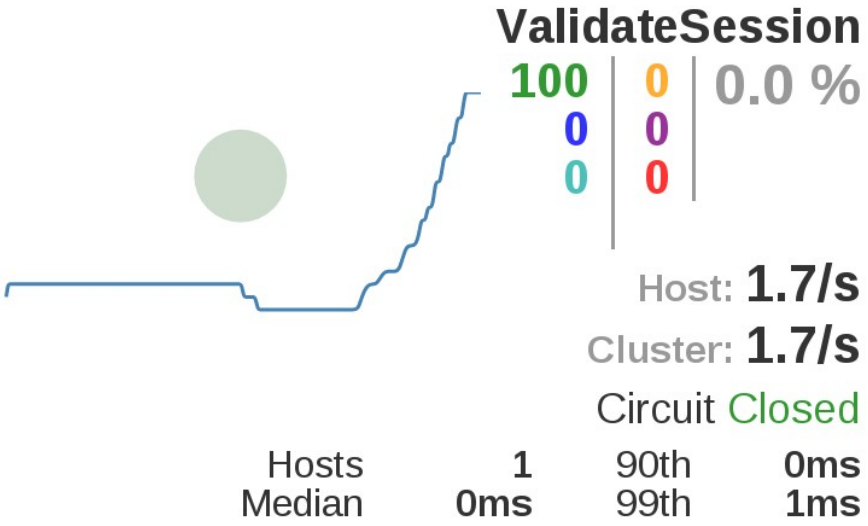
Find the Error

oVirt

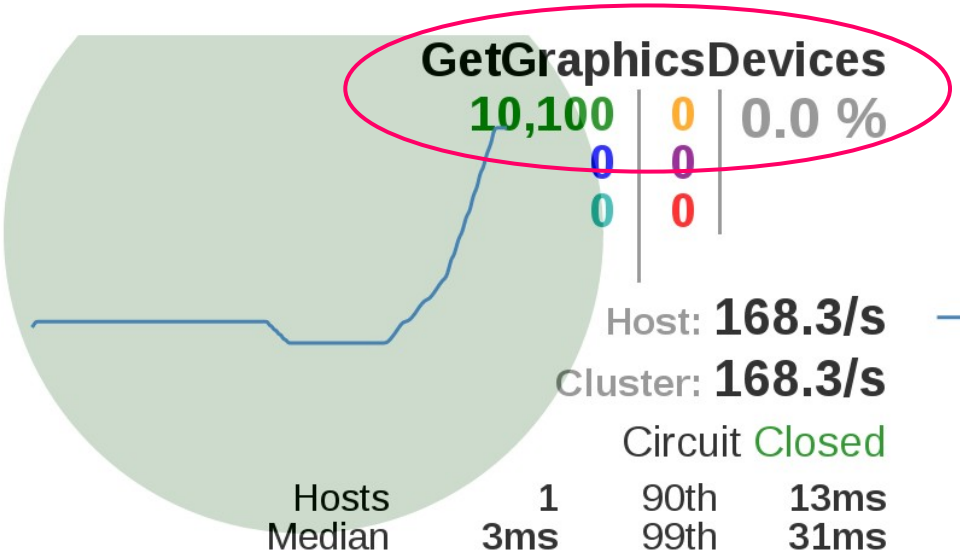


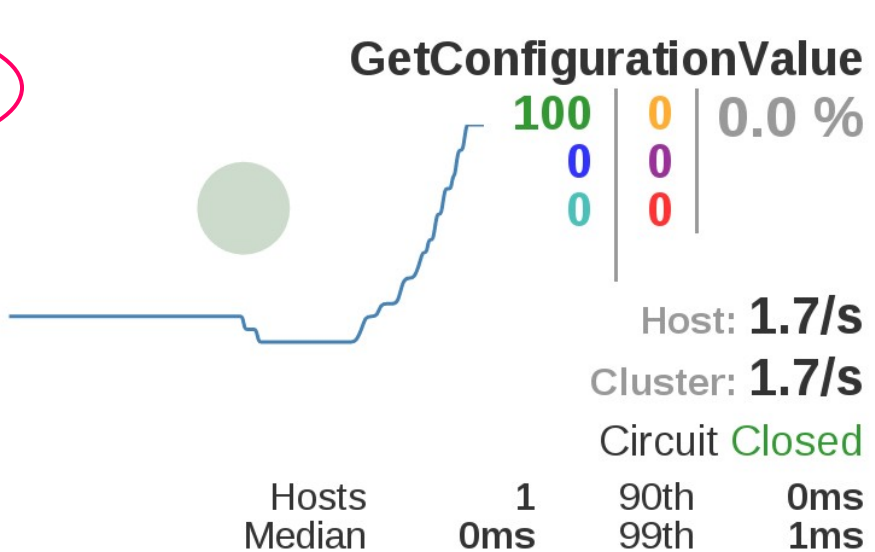


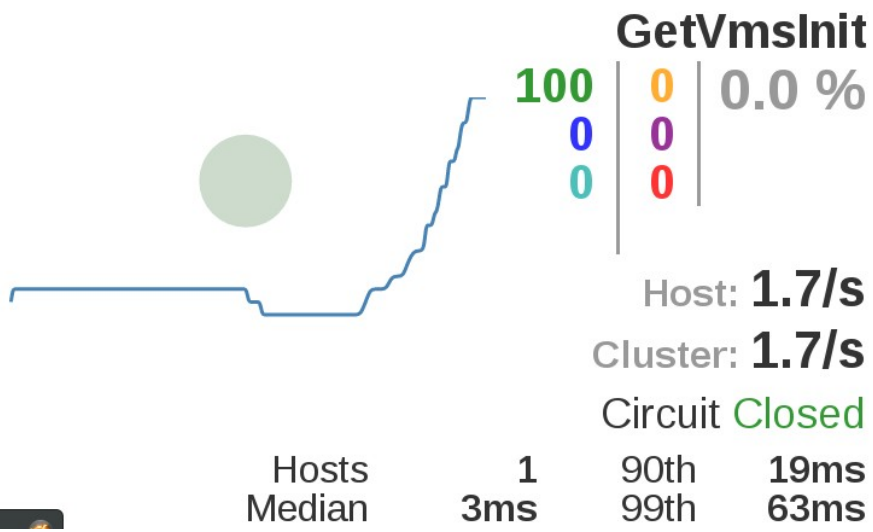


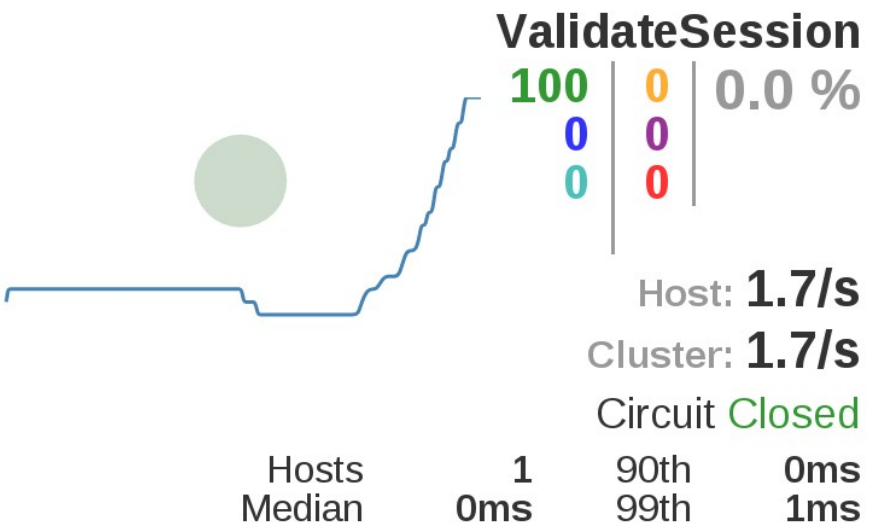


Find the Error









With the Fix

1000 VMs, 1 request

```
$> time bash rest.sh vms > /dev/null
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 7051k    0 7051k    0     0  8521k      0 --:--:-- --:--:-- --:--:-- 8516k

real 0m1.008s
user0m0.091s
sys  0m0.042s
```

2000 VMs, 1 request

```
$> time bash rest.sh vms > /dev/null
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 13.7M    0 13.7M    0     0  7210k      0 --:--:--  0:00:01 --:--:-- 7210k

real 0m2.218s
user 0m0.079s
sys  0m0.062s
```

With the Fix

oVirt

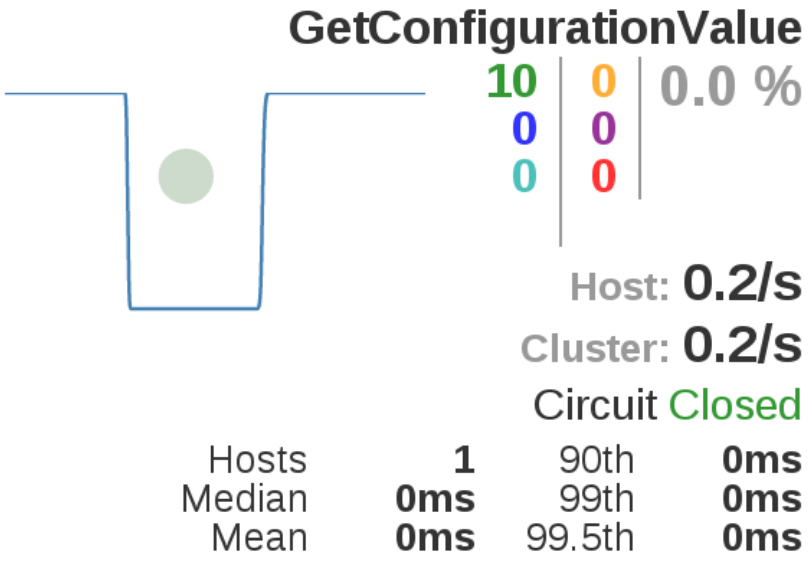
2000 VMs, 10 parallel requests

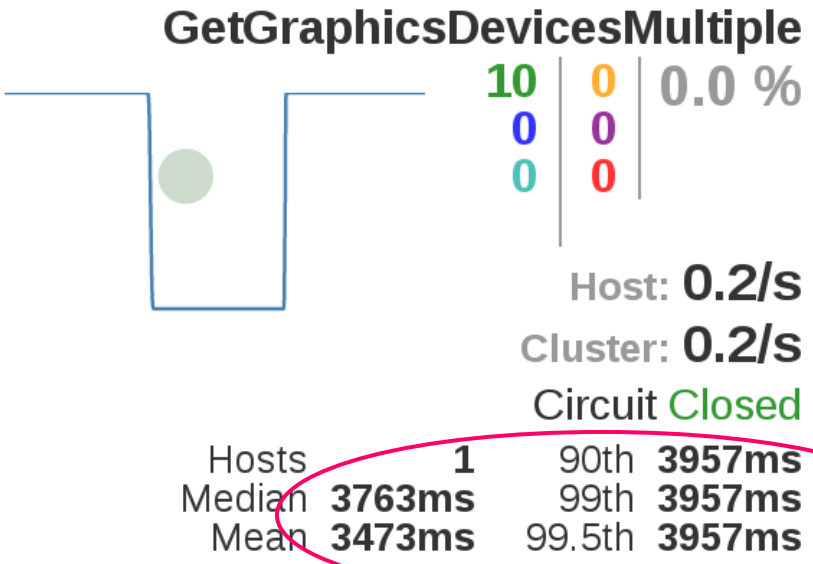
```
time seq 1 10 | parallel -j 10 bash rest.sh vms > /dev/null

  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 13.7M    0 13.7M    0     0  1473k      0 --:--:--  0:00:09 --:--:-- 3249k
[...]
100 13.7M    0 13.7M    0     0  1534k      0 --:--:--  0:00:09 --:--:-- 3566k

real 0m10.228s
user0m0.211s
sys  0m0.436s
```

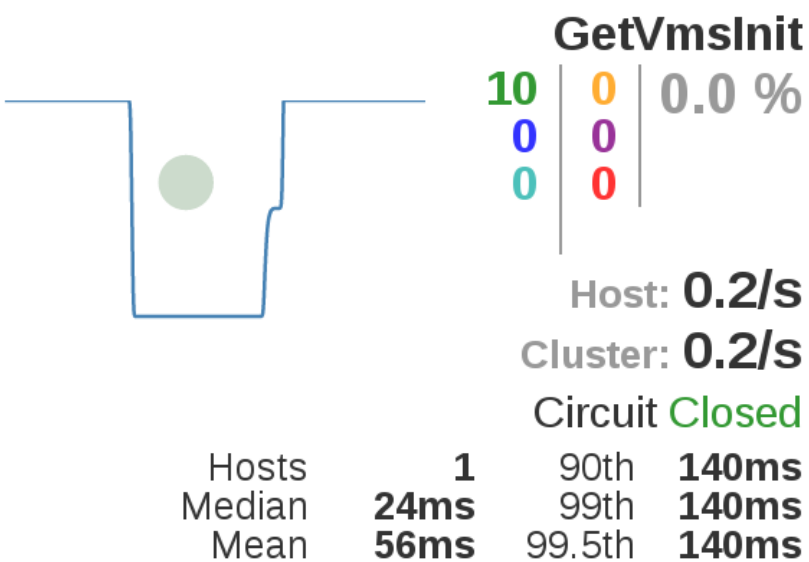
Much better but still too slow. We will see later how to avoid being overwhelmed by too much expensive calls.

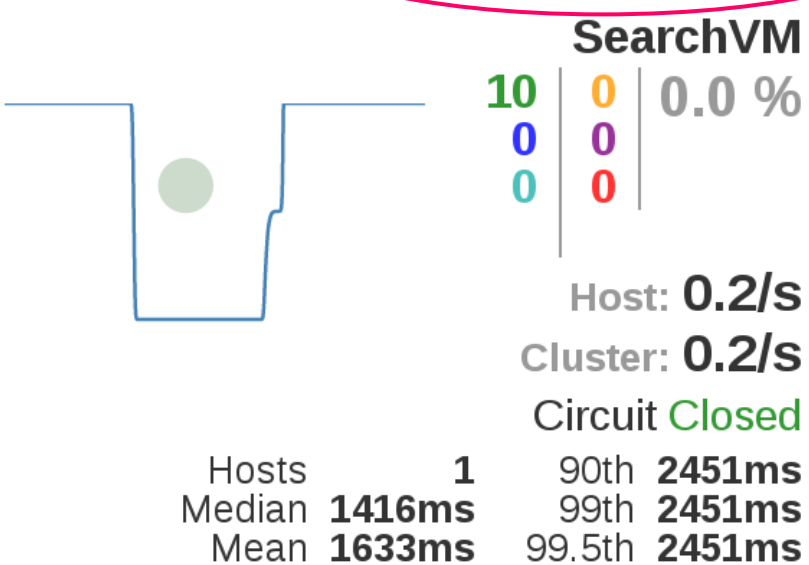
With the Fix

oVirt









Collect data from user systems

oVirt

Collect as much streaming data as you want:

```
$> curl -H "Accept: application/json"
       -H "Content-type: application/json" -X GET
       --user admin@internal:engine
       http://localhost:8080/ovirt-engine/services/hystrix.stream

ping:

data: {"type":"HystrixCommand","name":"GetVmsInit","group":"GetVmsInit", [...] }
data: {"type":"HystrixCommand","name":"VdsHostDevListByCaps", [...] }
```

Import it later in your favourite analysis tool or send the data directly to it by using Hystrix plugins.

Archaius: config.properties

oVirt

How to limit concurrent invocations of a command protected by Hystrix:

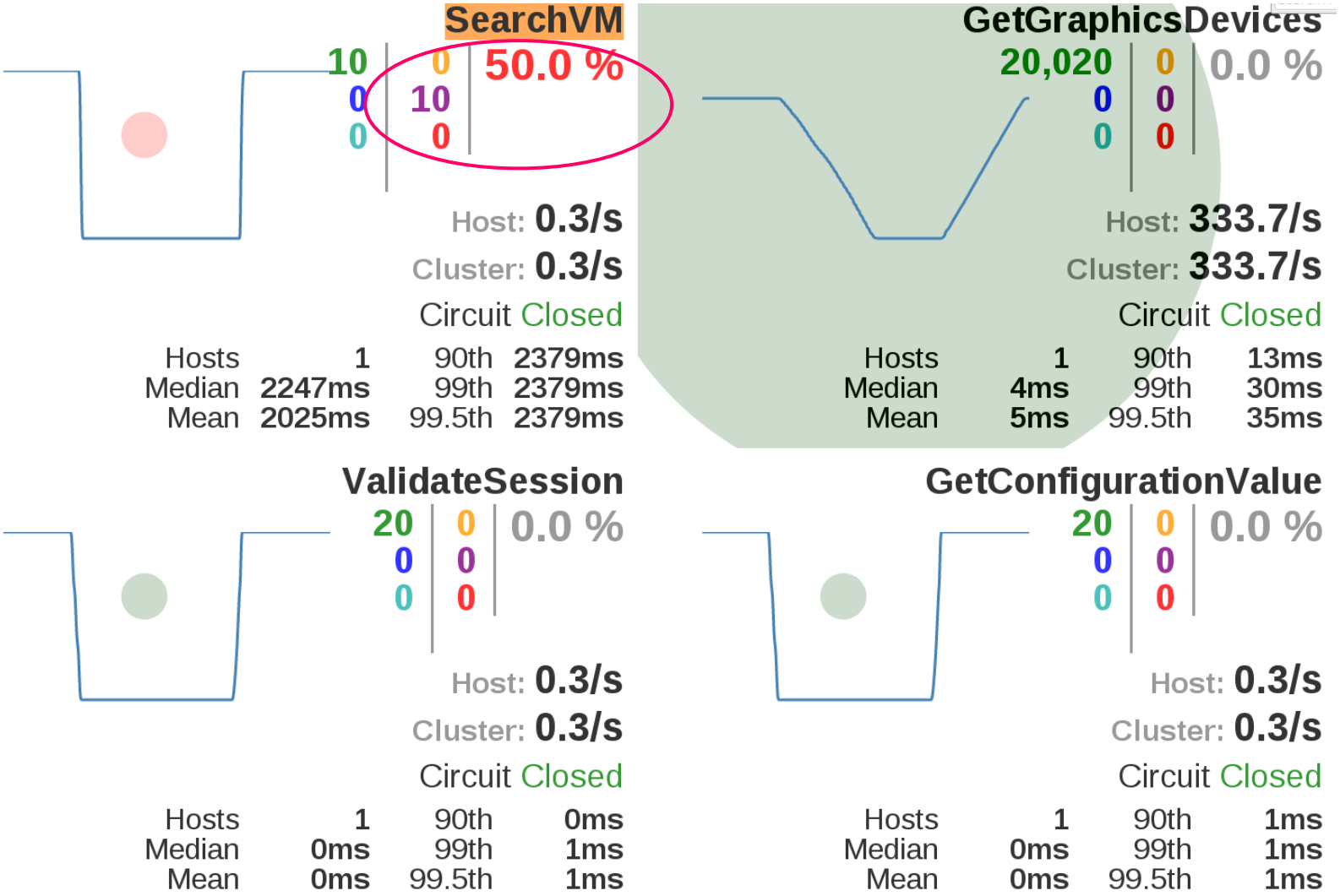
Default Value	`10`
Default Property	`hystrix.command.default.execution.isolation.semaphore.maxConcurrentRequests`
Instance Property	`hystrix.command.`*`HystrixCommandKey`*`.execution.isolation.semaphore.maxConcurrentRequests`
How to Set Instance Default	HystrixCommandProperties.Setter() .withExecutionIsolationSemaphoreMaxConcurrentRequests(int value)

This will override the default configuration of the **SearchVM** command:

```
hystrix.command.SearchVM.execution.isolation.semaphore.maxConcurrentRequests=10
```

Just add it to your config.properties file or set it as system property.

Protected VMs Endpoint



Great OSS Monitoring Ecosystem

oVirt

- Hystrix
- Servo
- Dropwizard Metrics
- Cockpit
- Thermostat
- Grafana, Graphite
- Hawkular
- Prometheus
- ...

Conclusion

oVirt

- Hystrix provides an easy way to monitor and protect your monolith or your microservices
- An interesting open source ecosystem around monitoring exists
- From some closed source projects we can learn a lot about usability

Questions?

Thank you!

Mail: rmohr@redhat.com

Github: https://github.com/rmohr

IRC: #ovirt irc.oftc.net