Huge Codebases
Application Monitoring with Hystrix


Roman Mohr
Red Hat

FOSDEM 2016
Roman Mohr
Software Engineer at Red Hat
Member of the SLA team in oVirt

Mail: rmoehr@redhat.com
Github: https://github.com/rmohr
IRC: #ovirt irc.oftc.net
“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-engine

The Beast

Take 1
Architecture of ovirt-engine

- REST
- GWT
- Commands
- DAL
- Broker
- DB
- VDSM
Git Statistics of ovirt-engine

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


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

- 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?

- 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?

- 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?

- 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?

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

Examples:
- Closed source
- Production only
- License based business model
“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 also provides metrics!
Hystrix Dashboard

Hystrix Stream: http://localhost:8080/ovirt-engine/services/hystrix.stream

Circuit Sort: Error then Volume | Alphabetical | Volume | Error | Mean | Median | 90 | 99 | 99.5

Thread Pools Sort: Alphabetical | Volume |
Hystrix Dashboard

- **Request shape**
- **GetJobsByOffset**
  - Host: 0.2/s
  - Cluster: 0.2/s
  - Circuit Closed
- **Statistics**
  - Hosts Median Mean
    - 1 1ms 0ms
  - 90th 99th 99.5th
    - 1ms 1ms 1ms
- **Error percentage**
  - 0.0 %
- **Requests/s**
  - 9 0 0
<table>
<thead>
<tr>
<th>Category</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Successful</td>
<td>9</td>
</tr>
<tr>
<td>Rejected (Short circuit)</td>
<td>0</td>
</tr>
<tr>
<td>Bad request (exception)</td>
<td>0</td>
</tr>
<tr>
<td>Timeouts (thread isolation)</td>
<td>0</td>
</tr>
<tr>
<td>Rejected (max. concurrent invocations)</td>
<td>0</td>
</tr>
<tr>
<td>Failed executions (exception)</td>
<td>0</td>
</tr>
</tbody>
</table>
Hystrix Dashboard

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

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

- 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

1000 VMs, 1 request

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

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

2000 VMs, 1 request

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

real  0m4.900s
user  0m0.009s
sys   0m0.014s
```
Solution: Let's curl a little bit

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
user 0m0.212s
sys  0m0.438s
```
Solution: We can guess

- “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

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

Huge Codebases – Application Monitoring with Hystrix
Find the Error

- **GetGraphicsDevices**
  - Host: 168.3/s
  - Cluster: 168.3/s
  - Circuit: Closed
  - 10,100 requests
  - 0 failures
  - 0.0 % failure rate

- **GetConfigurationValue**
  - Host: 1.7/s
  - Cluster: 1.7/s
  - Circuit: Closed
  - 100 requests
  - 0 failures
  - 0.0 % failure rate

- **GetVmsInit**
  - Host: 1.7/s
  - Cluster: 1.7/s
  - Circuit: Closed
  - 100 requests
  - 0 failures
  - 0.0 % failure rate

- **ValidateSession**
  - Host: 1.7/s
  - Cluster: 1.7/s
  - Circuit: Closed
  - 100 requests
  - 0 failures
  - 0.0 % failure rate

---

Huge Codebases – Application Monitoring with Hystrix
1000 VMs, 1 request

```
$> time bash rest.sh vms > /dev/null
% Total  % Received % Xferd Average Speed Time    Time     Time  Current
     0     0     0      0            0  0          0 --:--:-- --:--:-- --:--:-- 8516k
real 0m1.008s
user 0m0.091s
sys  0m0.042s
```

2000 VMs, 1 request

```
$> time bash rest.sh vms > /dev/null
% Total  % Received % Xferd Average Speed Time    Time     Time  Current
     0     0     0      0            0  0          0 --:--:-- --:--:-- --:--:-- 8516k
real 0m2.218s
user 0m0.079s
sys  0m0.062s
```
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
user 0m0.211s
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

**GetConfigurationValue**

<table>
<thead>
<tr>
<th>10</th>
<th>0</th>
<th>0.0 %</th>
</tr>
</thead>
</table>

Host: 0.2/s  
Cluster: 0.2/s  
Circuit Closed

**GetGraphicsDevicesMultiple**

<table>
<thead>
<tr>
<th>10</th>
<th>0</th>
<th>0.0 %</th>
</tr>
</thead>
</table>

Host: 0.2/s  
Cluster: 0.2/s  
Circuit Closed

**GetVmsInit**

<table>
<thead>
<tr>
<th>10</th>
<th>0</th>
<th>0.0 %</th>
</tr>
</thead>
</table>

Host: 0.2/s  
Cluster: 0.2/s  
Circuit Closed

**SearchVM**

<table>
<thead>
<tr>
<th>10</th>
<th>0</th>
<th>0.0 %</th>
</tr>
</thead>
</table>

Host: 0.2/s  
Cluster: 0.2/s  
Circuit Closed

---

Huge Codebases – Application Monitoring with Hystrix
Collect data from user systems

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

<table>
<thead>
<tr>
<th>Default Value</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default Property</td>
<td><code>hystrix.command.default.execution.isolation.semaphore.maxConcurrentRequests</code></td>
</tr>
<tr>
<td>Instance Property</td>
<td><code>hystrix.command.HystrixCommandKey.execution.isolation.semaphore.maxConcurrentRequests</code></td>
</tr>
</tbody>
</table>

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

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

Just add it to your config.properties file or set it as system property.
Great OSS Monitoring Ecosystem

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

Huge Codebases – Application Monitoring with Hystrix
Conclusion

- 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