Performance Tips for container applications

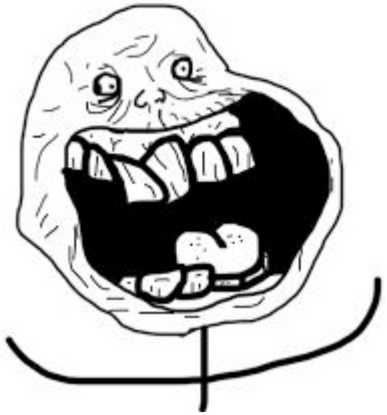
How to test and check performance issues

Schedule

- Infrastructure and Application
- What look?
- How to look?
- Optimization tradeoffs

Infrastructure - Concerns

- Isolated
- One container application doesn’t affect other
- You don’t need to worry about where your container will run
- You can reproduce the same behavior locally



Infrastructure - Concerns

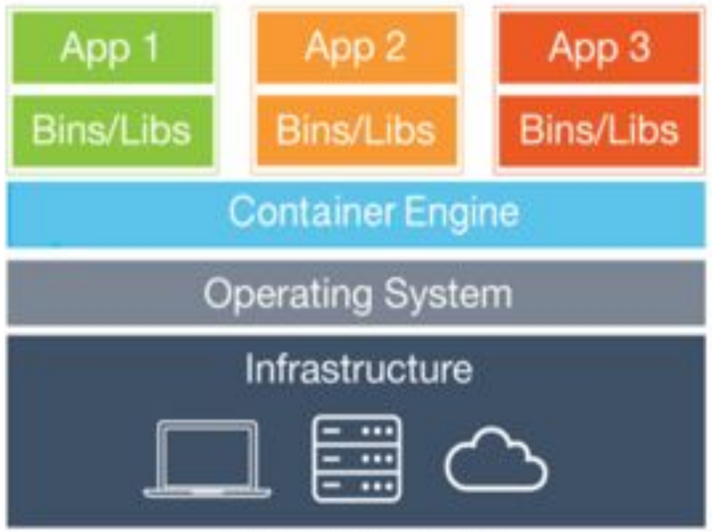
- Your application will share resources with other containers?
  - **Pros**
    - Better use of your physical resources (CPU, Memory, etc)
      - Your application it is not using CPU ALL the time
    - Make sense have more than one container in the same host
    - Save money
  - **Cons**
    - The physical resources available changes over time
    - Application impacts each other
    - Spent more money

Infrastructure - Concerns

- All containers in the same host are yours?
  - If yes
    - The control about sharing resources belongs to the team responsible by the applications
    - The development team needs to know about infrastructure
    - The development team could tune the applications and hardware
    - Could have waste of resources (Ex: batch operations)
  - If not
    - The control about sharing resources belongs to the infrastructure team
    - The development team has fewer options to tune application
    - Possible no waste of resources

Infrastructure - Concerns

- The host will share containers with other applications?
  - Data base
  - Infrastructure resources
    - Load Balancers
    - HAProxy
    - Nginx / Apache
    - Cache Layer
    - Logger App (ex: FluentD)



Infrastructure - Concerns

- The environment will put some limits?
  - Memory
  - CPU usage
  - File Descriptor
  - Network Bandwidth
  - Disk I/O operations

Application - Concerns

- Your application will run inside a Virtual Machine / Server ?
  - Java with JVM (Application Server: Tomcat, Jetty, etc)
  - .NET Kestrel Web Server
- Your container runs more than one application?

What look?

- Memory
- CPU usage
- File Descriptor
    - Sockets
    - Files
- Network Bandwidth
- Disk I/O operations

How to look? - Docker run command

```
docker run -it --ulimit nofile=<file_descriptor_limit>
--memory <memory_limit> --cpus <cpu_limit> --name <app_name>
<app_image>
```

- **Hardware/Infrastructure Limits:**
  - <file_descriptor_limit>
  - <memory_limit>
  - <cpu_limit>

How to look? - Minimum requirements

- **Run your application without limits**
- **Performance Test with status URL (do nothing)**
- **Performance Test with service URL (do the thing)**

How to look? - Example

- Java Application
- Endpoints
  - GET /status : do nothing, return HTTP 200
  - POST /sort : Sort N lists of integer numbers internally using async for each list and with thread sleep with random milliseconds

How to look? - Example

```
curl -v -X POST http://localhost:8080/sort 'content-type: application/json' \
  -d '{
    "intList": [{
        "list": [8, 9, 4, 12, 56, 78, 90, 56, 34, 74]
    }, {
        "list": [8, 9, 4, 12, 56, 78, 90, 56, 34, 74]
    }, {
        "list": [8, 9, 4, 12, 56, 78, 90, 56, 34, 74]
    }, {
        "list": [8, 9, 4, 12, 56, 78, 90, 56, 34, 74]
    }, {
        "list": [8, 9, 4, 12, 56, 78, 90, 56, 34, 74]
    }]
}'
```

How to look? - Example

Run Application

```
docker run -it -p 8080:8080 --name demo demo:1.0.0
```

Docker Status

```
docker stats demo
```

Open File Descriptors

```
docker exec -it demo /bin/bash
apt-get update
apt-get install procps -y
ps -efl
watch ‘ls -altr /proc/6/fd | wc -l’
```

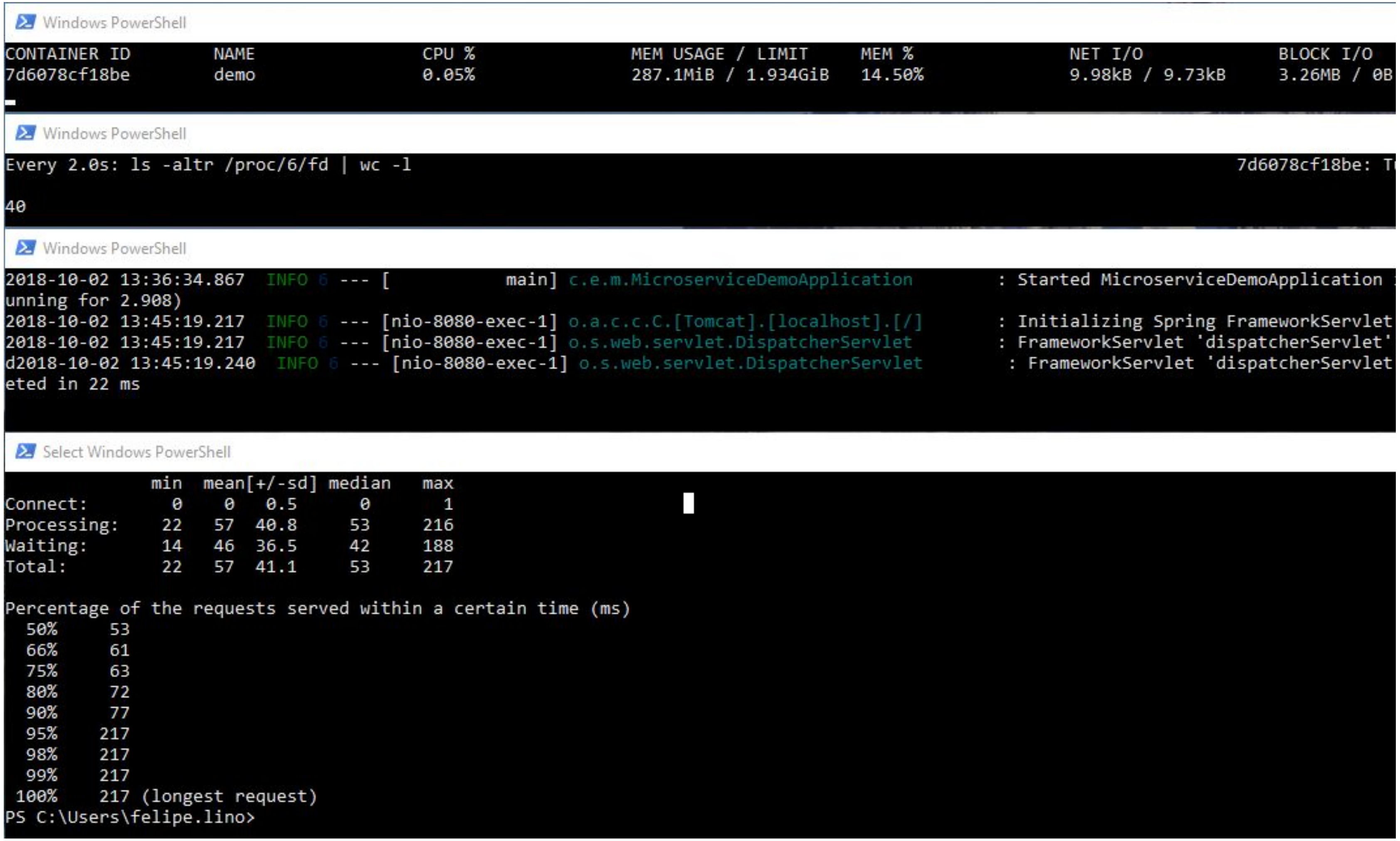
How to look? - Example

Performance Test (Ex: Apache Benchmark Tool)

```
ab -s 60 -c 10 -n 20 http://localhost:8080/status
```

- **-c : Concurrent Users**
- **-n : Number of iterations**
- **-s : timeout (seconds)**

ApacheBench



```
CONTAINER ID        NAME                CPU %               MEM USAGE / LIMIT     MEM %               NET I/O             BLOCK I/O
7d6078cf18be        demo                0.05%               287.1MiB / 1.934GiB   14.50%              9.98kB / 9.73kB     3.26MB / 0B
```

```
Every 2.0s: ls -altr /proc/6/fd | wc -l                                                              7d6078cf18be: T

40
```

```
2018-10-02 13:36:34.867  INFO 6 --- [           main] c.e.m.MicroserviceDemoApplication        : Started MicroserviceDemoApplication
unning for 2.908)
2018-10-02 13:45:19.217  INFO 6 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring FrameworkServlet
2018-10-02 13:45:19.217  INFO 6 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : FrameworkServlet 'dispatcherServlet'
d2018-10-02 13:45:19.240  INFO 6 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : FrameworkServlet 'dispatcherServlet
eted in 22 ms
```

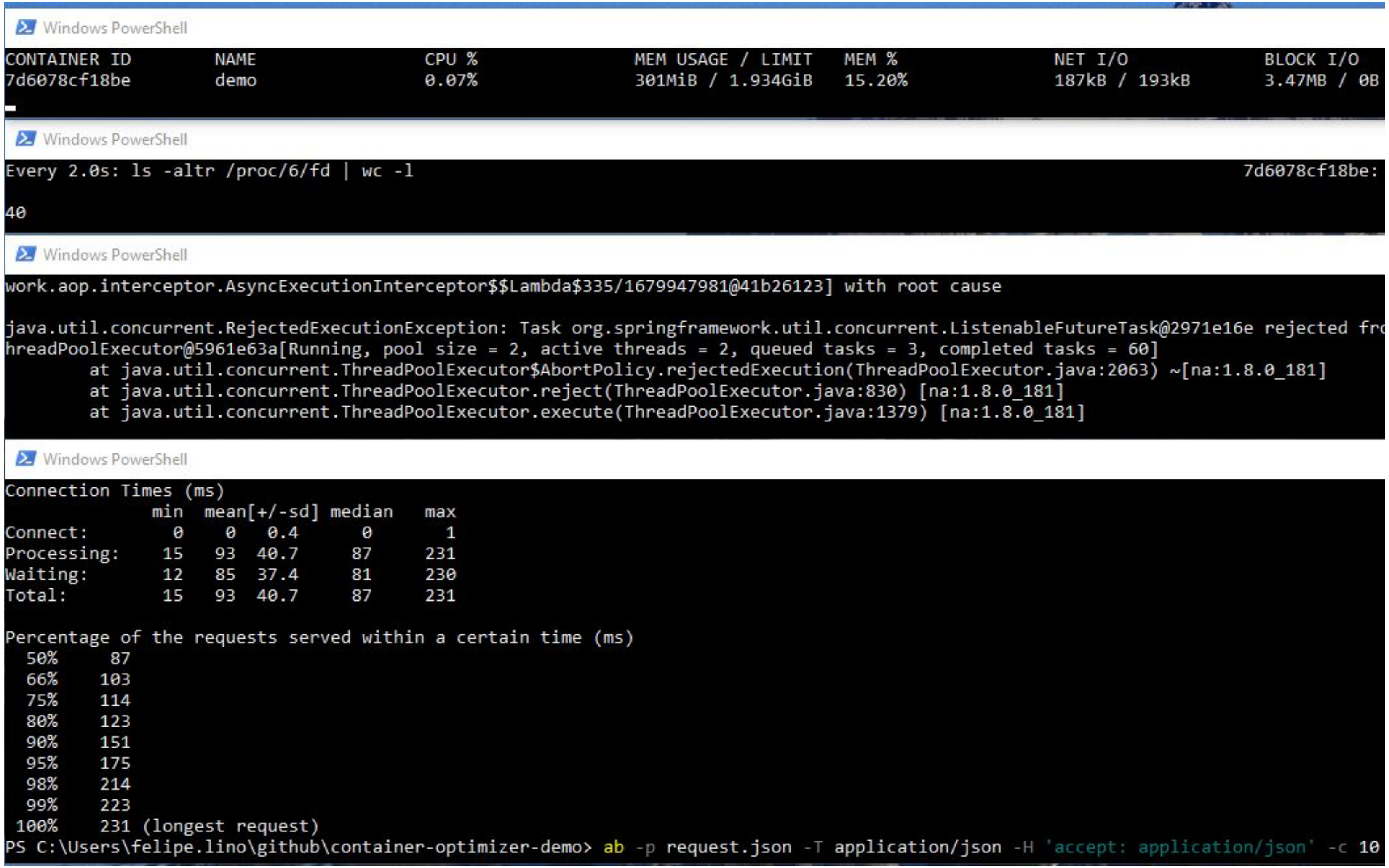
```
              min  mean[+/-sd] median   max
Connect:        0    0   0.5      0       1
Processing:    22   57  40.8     53     216
Waiting:       14   46  36.5     42     188
Total:         22   57  41.1     53     217

Percentage of the requests served within a certain time (ms)
  50%     53
  66%     61
  75%     63
  80%     72
  90%     77
  95%    217
  98%    217
  99%    217
 100%    217 (longest request)
PS C:\Users\felipe.lino>
```

How to look? - Example

Performance Test (Ex: Apache Benchmark Tool)

```
ab -p request.json -T application/json -H 'accept:
application/json' -c 10 -n 200 http://localhost:8080/sort
```



```
CONTAINER ID        NAME                CPU %               MEM USAGE / LIMIT    MEM %               NET I/O             BLOCK I/O
7d6078cf18be        demo                0.07%               301MiB / 1.934GiB    15.20%              187kB / 193kB       3.47MB / 0B
```

```
Every 2.0s: ls -altr /proc/6/fd | wc -l                                                                              7d6078cf18be:

40
```

```
work.aop.interceptor.AsyncExecutionInterceptor$$Lambda$335/1679947981@41b26123] with root cause

java.util.concurrent.RejectedExecutionException: Task org.springframework.util.concurrent.ListenableFutureTask@2971e16e rejected fro
hreadPoolExecutor@5961e63a[Running, pool size = 2, active threads = 2, queued tasks = 3, completed tasks = 60]
        at java.util.concurrent.ThreadPoolExecutor$AbortPolicy.rejectedExecution(ThreadPoolExecutor.java:2063) ~[na:1.8.0_181]
        at java.util.concurrent.ThreadPoolExecutor.reject(ThreadPoolExecutor.java:830) [na:1.8.0_181]
        at java.util.concurrent.ThreadPoolExecutor.execute(ThreadPoolExecutor.java:1379) [na:1.8.0_181]
```

```
Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.4      0       1
Processing:    15   93  40.7     87     231
Waiting:       12   85  37.4     81     230
Total:         15   93  40.7     87     231

Percentage of the requests served within a certain time (ms)
  50%     87
  66%    103
  75%    114
  80%    123
  90%    151
  95%    175
  98%    214
  99%    223
 100%    231 (longest request)
PS C:\Users\felipe.lino\github\container-optimizer-demo> ab -p request.json -T application/json -H 'accept: application/json' -c 10
```

How to look? - Maximum requirements

- **Run your application applying infrastructure limits**
- **Run your application applying server (JVM) parameters**
- **Performance Test with service URL (do the thing)**

How to look? - Example

```
Run Application
docker run -it --ulimit nofile=128 --memory 1000MB --cpus 0.5  -e
"JAVA_OPTS=-Xms312m -Xmx750m -DcorePoolSize=100 -DmaxPoolSize=120
-DqueueCapacity=300" -p 8080:8080  --name demo demo:1.0.0
```

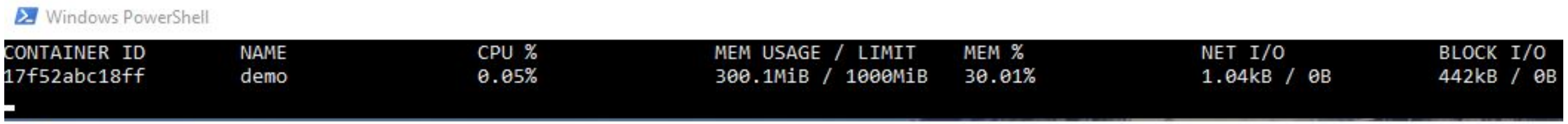
Infrastructure limits:

- --cpus
- --memory
- --ulimit

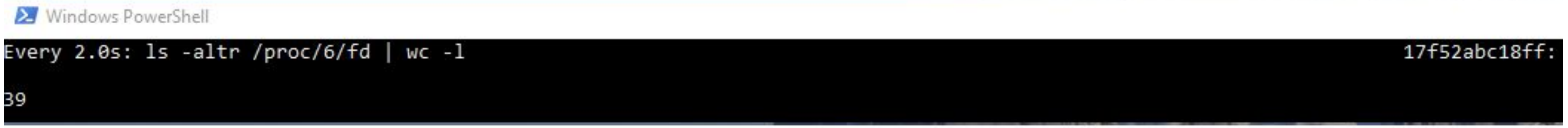
Applications config:

- Threads: corePoolSize (2), maxPoolSize (2), queueCapacity(3)
- Xms / Xmx
- etc.

How to look? Before start again


Windows PowerShell

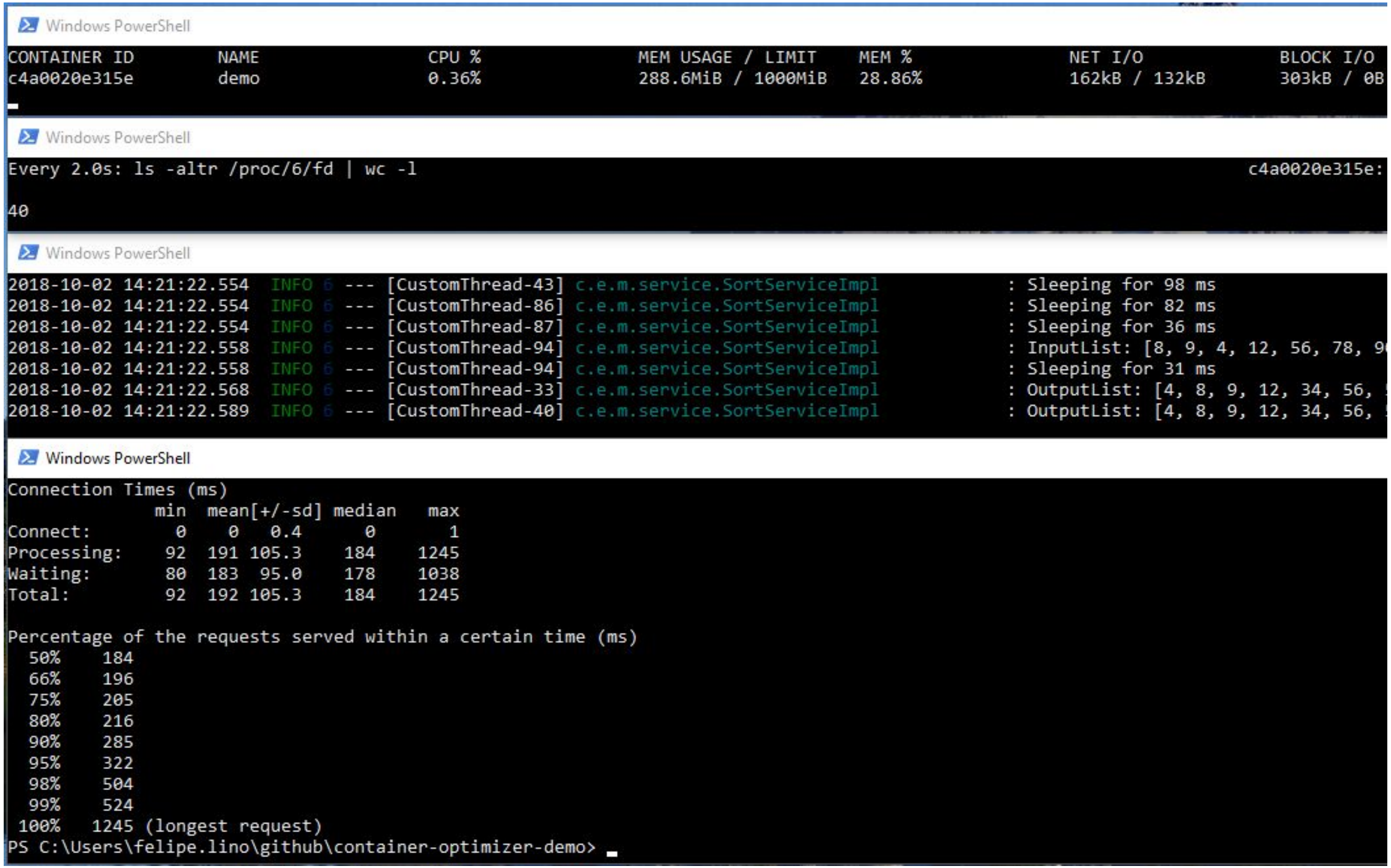
```
CONTAINER ID        NAME                CPU %               MEM USAGE / LIMIT     MEM %               NET I/O             BLOCK I/O
17f52abc18ff        demo                0.05%               300.1MiB / 1000MiB    30.01%              1.04kB / 0B         442kB / 0B
```


Windows PowerShell

```
Every 2.0s: ls -altr /proc/6/fd | wc -l                                                                    17f52abc18ff:

39
```

- Memory Limit is 1GB
- You start to usage around 300MB because of JAVA_OPTS with *-Xms312m*


Windows PowerShell

```
CONTAINER ID        NAME                CPU %               MEM USAGE / LIMIT     MEM %               NET I/O             BLOCK I/O
c4a0020e315e        demo                0.36%               288.6MiB / 1000MiB    28.86%              162kB / 132kB       303kB / 0B
```

Windows PowerShell

```
Every 2.0s: ls -altr /proc/6/fd | wc -l                                                                                 c4a0020e315e:

40
```

Windows PowerShell

```
2018-10-02 14:21:22.554  INFO 6 --- [CustomThread-43] c.e.m.service.SortServiceImpl            : Sleeping for 98 ms
2018-10-02 14:21:22.554  INFO 6 --- [CustomThread-86] c.e.m.service.SortServiceImpl            : Sleeping for 82 ms
2018-10-02 14:21:22.554  INFO 6 --- [CustomThread-87] c.e.m.service.SortServiceImpl            : Sleeping for 36 ms
2018-10-02 14:21:22.558  INFO 6 --- [CustomThread-94] c.e.m.service.SortServiceImpl            : InputList: [8, 9, 4, 12, 56, 78, 9
2018-10-02 14:21:22.558  INFO 6 --- [CustomThread-94] c.e.m.service.SortServiceImpl            : Sleeping for 31 ms
2018-10-02 14:21:22.568  INFO 6 --- [CustomThread-33] c.e.m.service.SortServiceImpl            : OutputList: [4, 8, 9, 12, 34, 56,
2018-10-02 14:21:22.589  INFO 6 --- [CustomThread-40] c.e.m.service.SortServiceImpl            : OutputList: [4, 8, 9, 12, 34, 56,
```

Windows PowerShell

```
Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.4      0       1
Processing:    92  191 105.3    184    1245
Waiting:       80  183  95.0    178    1038
Total:         92  192 105.3    184    1245

Percentage of the requests served within a certain time (ms)
  50%    184
  66%    196
  75%    205
  80%    216
  90%    285
  95%    322
  98%    504
  99%    524
 100%   1245 (longest request)
PS C:\Users\felipe.lino\github\container-optimizer-demo>
```

Optimizations tradeoffs

- **Change memory limits**
- **Change CPU limits**
- **File Descriptors and Sockets**
- **Number of containers x Threads (workers) inside application**
- **Specific configurations for your Server/JVM**
  - server.undertow.io-threads
  - KestrelServerOptions.ThreadCount

Optimizations tradeoffs

- **IO Usage**
- **Network Usage**
- **Related dependencies**
  - **APIs**
  - **Database**
  - **File System**

Any doubts?

Source:
https://github.com/felipelino/container-optimizer-demo

Contacts:
Felipe Lino (felipelino44@gmail.com)