Exercise 6

Simple Benchmarking with wrk

Prior Knowledge
Previous exercises

Objectives
Benchmarking runtimes

Software Requirements

- Java Development Kit 8
- Redis
- wrk - a simple benchmarking tool

Overview
We will look at using a benchmarking tool to call our APIs very fast and see how they react.

Steps

1. Create a directory for this work and clone the git repository:
   mkdir ~/ex6
   cd ~/ex6
   git clone https://github.com/pzfreo/PSBComplete.git

2. Make sure redis is running locally:
   sudo service redis-server start

3. Build the code
   ```
   cd PSBComplete
   gradle build
   ```

4. Run the "Uber JAR" as before

5. We can performance test our app. First lets install wrk, a simple HTTP performance test app:

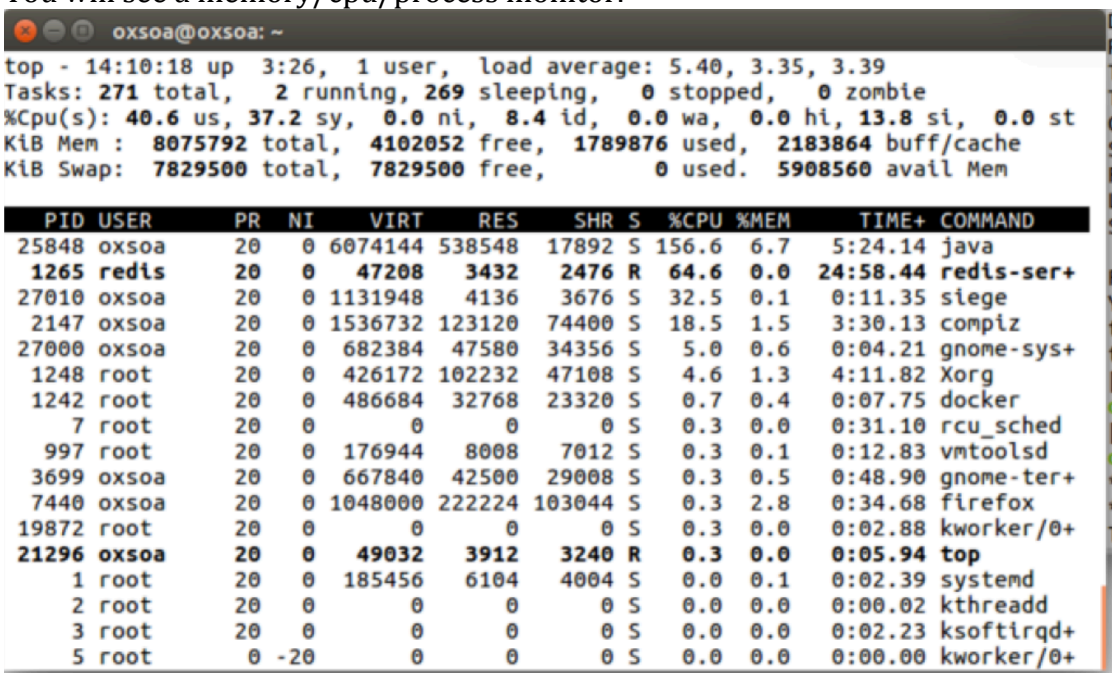
   ```
   sudo apt install -y wrk
   ```

6. Now we can run a test:
   ```
   wrk -c 100 -d 2m -t 10 http://localhost:8080/purchase
   ```

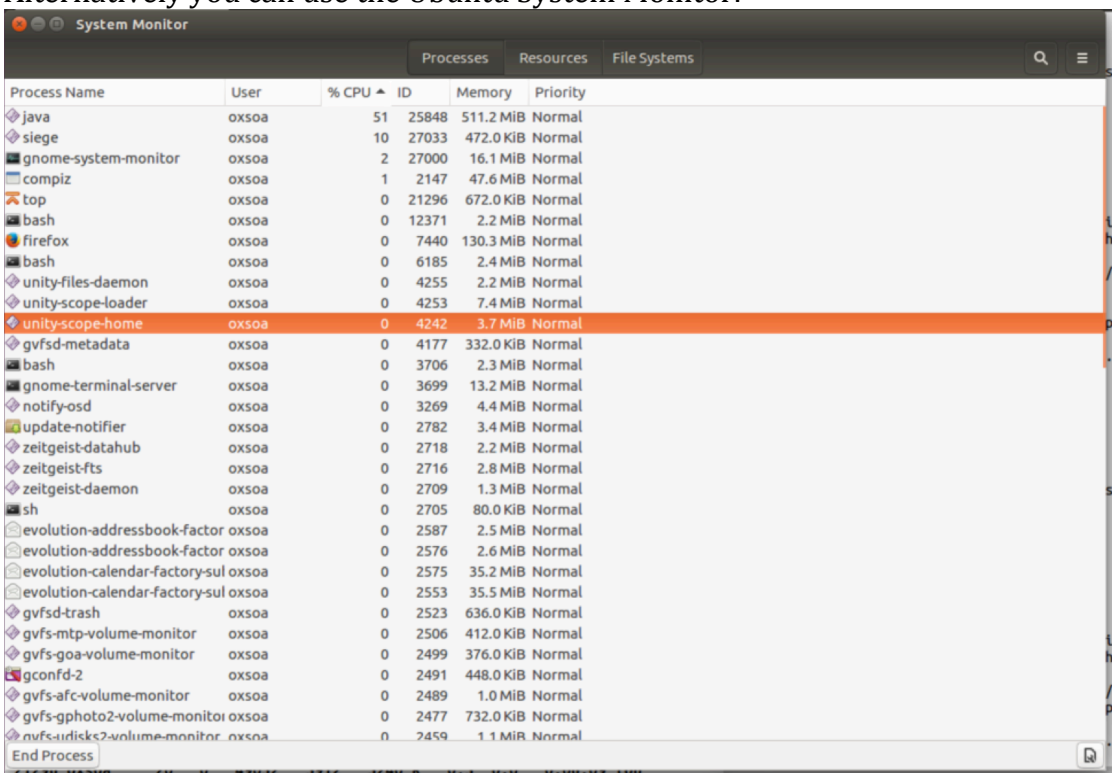
7. This will constantly hit our server with 100 concurrent clients calling over 2 minute (using 10 threads).

8. While it is running you can monitor the CPU.
   Open up a new terminal window and type:
   `top`

9. You will see a memory/cpu/process monitor.



10. Alternatively you can use the Ubuntu System Monitor:



11. Back in your wrk terminal window you should see it complete:

```
oxsoa@oxsoa:~/PSBComplete$ wrk -c 1000 -t 10 -d 1m http://localhost:8080/purchase
Running 1m test @ http://localhost:8080/purchase
  10 threads and 1000 connections
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency   540.59ms  389.74ms   2.00s    60.82%
    Req/Sec   187.00     74.79   797.00     70.07%
  111104 requests in 1.00m, 53.52MB read
  Socket errors: connect 0, read 0, write 0, timeout 708
Requests/sec:   1849.88
Transfer/sec:      0.89MB
```

12. You might want to re-run this again as the JVM will warm up (JIT compilation).

13. By default Spring Boot uses the Tomcat servlet engine as the server. However, that is pretty slow. There is a better engine called Undertow.

    We can enable that just by changing the gradle build file.

    We need to exclude the Tomcat and include undertow. The lines you need to change are commented out in the build gradle or you can see them below:

```
configurations {
    // exclude Tomcat
    compile.exclude module: 'spring-boot-starter-tomcat'
}


dependencies {
    implementation('org.springframework.boot:spring-boot-starter-undertow')
    implementation('org.springframework.boot:spring-boot-starter-jersey')
```

14. Rebuild and then rerun the performance test. Run it twice to allow the JVM to warm up. How do they compare?

15. Note that this is not a real performance analysis. Ideally the servers would be on a separate machine from the client load drivers (siege engines!). Also, microservices are designed to be run in parallel in multiple containers with load balancing across them, so this model is not the recommended way of running either deployment.

16. That's all