

Exercise 7c

*Trying out WSO2 MSF4J instead of Jetty/Jersey
Benchmarking with Siege*

Prior Knowledge

Previous exercises

Objectives

Testing another JAX-RS implementation for Microservices
Benchmarking runtimes

Software Requirements

- Java Development Kit 8
- MSF4J
- Redis
- siege

Overview

MSF4J is a new framework for running JAX-RS code as a microservice.

It has a very simple model.

We will also 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 ~/ex7c
cd ~/ex7c
git clone https://github.com/pzfereo/POResourceComplete.git`
2. Make sure redis is running locally:
`sudo service redis-server start`



3. Build both the Jersey and MSF4J versions of the same Resource:

```
cd POResourceComplete  
gradle shadowJar  
mvn clean install -Dmaven.test.skip=true
```

Note: We are skipping tests because I didn't add logic into the maven build to start the microservice and stop it before and after tests. The MSF4J recommendation is to start the service in the test itself (since its so quick to start), but I didn't want to change the tests as they are shared with the gradle/jersey build.

Hint: the mvn may take a while as it downloads half the internet to your VM

Hint 2: While you are waiting for the downloads you can take a look at the MSF4J project site to understand it a bit better:

<https://github.com/wso2/msf4j/>

Hint 3: My pom.xml seems horribly more complex than the one on github. There are three simple reasons. Firstly, I'm using the latest build of MSF4J which isn't yet in the main repos. Secondly, I've added my dependencies (json, etc). Thirdly, I've had to exclude the build from looking at the Jersey Main.java class because that has its own set of dependencies.

4. Run the MSF4J version:

```
java -jar target/POResource-1.1.0-SNAPSHOT.jar
```

5. See if it works (its on port 8080).

6. We can performance test our app. First lets install siege:

```
sudo apt install -y siege
```



7. Now we can run a test:

siege -b -c 15 -r 10000 <http://localhost:8080/purchase>

8. This will constantly hit our server with 15 concurrent clients each calling 10k times, in benchmark mode (i.e. each request hits immediately after the one before rather than being random).

9. You should see:

```
oxsoa@oxsoa:~$ siege -b -c 15 -r 10000 http://localhost:8080/purchase
** SIEGE 3.0.8
** Preparing 15 concurrent users for battle.
The server is now under siege.. done.

Transactions:          1500000 hits
Availability:          100.00 %
Elapsed time:           70.15 secs
Data transferred:       33.47 MB
Response time:          0.01 secs
Transaction rate:      2138.28 trans/sec
Throughput:             0.48 MB/sec
Concurrency:            14.80
Successful transactions: 1500000
Failed transactions:     0
Longest transaction:    0.10
Shortest transaction:   0.00

FILE: /var/log/siege.log
You can disable this annoying message by editing
the .siegerc file in your home directory; change
the directive 'show-logfile' to false.
[error] unable to create log file: /var/log/siege.log: Permission denied
```

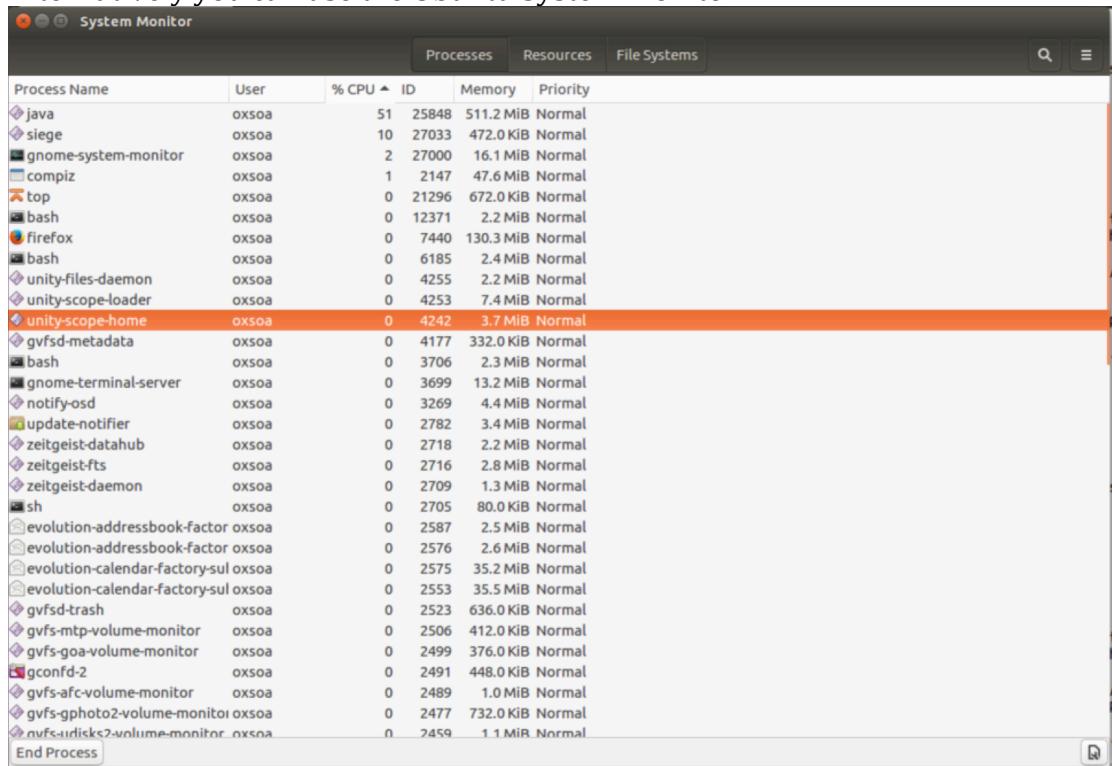
10. Open up a new terminal window and type:

top

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

top - 14:10:18 up 3:26, 1 user, load average: 5.40, 3.35, 3.39										
Tasks: 271 total, 2 running, 269 sleeping, 0 stopped, 0 zombie										
%Cpu(s): 40.6 us, 37.2 sy, 0.0 ni, 8.4 id, 0.0 wa, 0.0 hi, 13.8 si, 0.0 st										
KiB Mem : 8075792 total, 4102052 free, 1789876 used, 2183864 buff/cache										
KiB Swap: 7829500 total, 7829500 free, 0 used. 5908560 avail Mem										
PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+ COMMAND
25848	oxsoa	20	0	6074144	538548	17892	S	156.6	6.7	5:24.14 java
1265	redis	20	0	47208	3432	2476	R	64.6	0.0	24:58.44 redis-ser+
27010	oxsoa	20	0	1131948	4136	3676	S	32.5	0.1	0:11.35 siege
2147	oxsoa	20	0	1536732	123120	74400	S	18.5	1.5	3:30.13 compiz
27000	oxsoa	20	0	682384	47580	34356	S	5.0	0.6	0:04.21 gnome-sys+
1248	root	20	0	426172	102232	47108	S	4.6	1.3	4:11.82 Xorg
1242	root	20	0	486684	32768	23320	S	0.7	0.4	0:07.75 docker
7	root	20	0	0	0	0	S	0.3	0.0	0:31.10 rcu_sched
997	root	20	0	176944	8008	7012	S	0.3	0.1	0:12.83 vmtoolsd
3699	oxsoa	20	0	667840	42500	29008	S	0.3	0.5	0:48.90 gnome-ter+
7440	oxsoa	20	0	1048000	222224	103044	S	0.3	2.8	0:34.68 firefox
19872	root	20	0	0	0	0	S	0.3	0.0	0:02.88 kworker/0+
21296	oxsoa	20	0	49032	3912	3240	R	0.3	0.0	0:05.94 top
1	root	20	0	185456	6104	4004	S	0.0	0.1	0:02.39 systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.02 kthreadd
3	root	20	0	0	0	0	S	0.0	0.0	0:02.23 ksoftirqd+
5	root	0	-20	0	0	0	S	0.0	0.0	0:00.00 kworker/0+

12. Alternatively you can use the Ubuntu System Monitor:



The screenshot shows the Ubuntu System Monitor application window. The title bar says "System Monitor". Below it is a toolbar with three tabs: "Processes" (selected), "Resources", and "File Systems". On the right side of the toolbar are search and filter icons. The main area is a table with columns: "Process Name", "User", "% CPU", "ID", "Memory", and "Priority". The table lists numerous system processes, including "java", "siege", "gnome-system-monitor", "compiz", "top", "bash", "firefox", "unity-files-daemon", "unity-scope-loader", "unity-scope-home" (which is highlighted with a red border), "gvfsd-metadata", "bash", "gnome-terminal-server", "notify-osd", "update-notifier", "zeitgeist-databus", "zeitgeist-fts", "zeitgeist-daemon", "sh", "evolution-addressbook-factor", "evolution-addressbook-factor", "evolution-calendar-factory-sul", "evolution-calendar-factory-sul", "gvfsd-trash", "gvfs-mtp-volume-monitor", "gvfs-goa-volume-monitor", "gconfd-2", "gvfs-afc-volume-monitor", "gvfs-gphoto2-volume-monitor", and "gvfs-udisks2-volume-monitor". The "Memory" column shows values like 511.2 MiB, 472.0 KiB, etc.

Process Name	User	% CPU	ID	Memory	Priority
java	oxsoa	51	25848	511.2 MiB	Normal
siege	oxsoa	10	27033	472.0 KiB	Normal
gnome-system-monitor	oxsoa	2	27000	16.1 MiB	Normal
compiz	oxsoa	1	2147	47.6 MiB	Normal
top	oxsoa	0	21296	672.0 KiB	Normal
bash	oxsoa	0	12371	2.2 MiB	Normal
firefox	oxsoa	0	7440	130.3 MiB	Normal
bash	oxsoa	0	6185	2.4 MiB	Normal
unity-files-daemon	oxsoa	0	4255	2.2 MiB	Normal
unity-scope-loader	oxsoa	0	4253	7.4 MiB	Normal
unity-scope-home	oxsoa	0	4242	3.7 MiB	Normal
gvfsd-metadata	oxsoa	0	4177	332.0 KiB	Normal
bash	oxsoa	0	3706	2.3 MiB	Normal
gnome-terminal-server	oxsoa	0	3699	13.2 MiB	Normal
notify-osd	oxsoa	0	3269	4.4 MiB	Normal
update-notifier	oxsoa	0	2782	3.4 MiB	Normal
zeitgeist-databus	oxsoa	0	2718	2.2 MiB	Normal
zeitgeist-fts	oxsoa	0	2716	2.8 MiB	Normal
zeitgeist-daemon	oxsoa	0	2709	1.3 MiB	Normal
sh	oxsoa	0	2705	80.0 KiB	Normal
evolution-addressbook-factor	oxsoa	0	2587	2.5 MiB	Normal
evolution-addressbook-factor	oxsoa	0	2576	2.6 MiB	Normal
evolution-calendar-factory-sul	oxsoa	0	2575	35.2 MiB	Normal
evolution-calendar-factory-sul	oxsoa	0	2553	35.5 MiB	Normal
gvfsd-trash	oxsoa	0	2523	636.0 KiB	Normal
gvfs-mtp-volume-monitor	oxsoa	0	2506	412.0 KiB	Normal
gvfs-goa-volume-monitor	oxsoa	0	2499	376.0 KiB	Normal
gconfd-2	oxsoa	0	2491	448.0 KiB	Normal
gvfs-afc-volume-monitor	oxsoa	0	2489	1.0 MiB	Normal
gvfs-gphoto2-volume-monitor	oxsoa	0	2477	732.0 KiB	Normal
gvfs-udisks2-volume-monitor	oxsoa	0	2459	1.1 MiB	Normal
End Process					

13. Re-run the siege and check on the system performance as the system is under load.

14. Now run the Jersey / Jetty server instead
`java -jar build/libs/POResourceComplete-all.jar`

15. Re-run the siege and compare the memory and tps performance.

16. 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.

17. That's all

