COMPX341-20A Assignment 5

Instructor: Panos Patros

Due Date: ~~6pm Friday, June 19, 2020~~ 6pm Tuesday, June 23, 2020 (extended due to tech difficulties)

Individual Assignment—Software will track similarities!

Electronic Submission on Moodle—Late submissions will not be accepted!

Important! *Because of technical problems with Docker, you can choose to complete either Option A, which will require that you have access to a system supporting Docker, Docker Compose and Java; or Option B, which only requires access to word processor.*

Option A: Stress-Testing Containerized Microservices

Develop and stress-test using JMeter a simple containerized application-server that implements the following HTTP restful API:

Type	URI	Description	Requirement
GET	/isPrime/<number>	Decides if the input integer is prime and returns "<number> is prime" or "<number> is not prime", accordingly. If the number is prime, it is stored in the connected Redis object-storage service	REQ-1
GET	/primesStored	Returns a list with all the primes stored in the connected Redis service	REQ-2

Follow these instructions to get started:

1. If you are working remotely, install WINSCP and Putty
2. ~~Login to any of the following lab machines: `cms-rg06-XX.cms.waikato.ac.nz`, XX={50-90}~~
   Note: Given the technical problems this year, you'll have to run this on your own machine. If you can't, please complete Option B.
   a. I prefer saving the connection info so that I don't have to type everything again
      i. Particularly useful if you decide to set up a remote display host like Xming (Figure 1)
   b. If connected remotely, first type **`screen`**
   c. Screen allows you to maintain multiple persistent "tabs" and connect to them after you closed down a remote connection
   d. You can create a new tab by pressing ctrl+a followed by c. You can rotate around your open tabs with ctrl-a followed by n or backspace. Look up online for more info
3. Follow **Steps 1-4** of the Docker-Compose tutorial: https://docs.docker.com/compose/gettingstarted/

4. Modify the docker-compose.yml file port line to:
    a. `- "YOURPORT:5000"`
    b. Set YOURPORT to the last five digits of your student id (replace leading zeros with another number) to avoid colliding with each other
    c. This exposes YOURPORT to the host machine and forward it to port 5000 of the container
5. Start a new screen tab (ctrl-a c) and type `curl localhost:YOURPORT`
    a. If you've done the deployment properly, the system should return "`Hello World! I have been seen 1 times.`"

Afterwards:

1. Update the docker-compose.yml file to:

```
version: '2.2'
services:
  web:
    build: .
    ports:
     - "YOURPORT:5000"
    cpus: 0.1
    mem_limit: 128M
    restart: on-failure
  redis:
    image: "redis:alpine"
    cpus: 0.1
    mem_limit: 128M
    restart: on-failure
```

    a. The changes limits the running containers to 0.1 of a VCPU and 128MB of memory
    b. Also, it restarts them automatically on failure
    c. In a new screen, you may use `docker stats` to monitor at runtime the performance of your containers
2. Make the necessary updates to the code to implement the requirements
    a. Run `docker-compose build && docker-compose up to` to rebuild and redeploy after making changes
    b. Read more about Flask here: http://flask.pocoo.org/
    c. Read more about the Python API of Redis here: https://redis-py.readthedocs.io/en/latest/
    d. Read more on Python's math library here: https://docs.python.org/3/library/math.html
3. Come up and document a number of test cases (using black- and white-box coverage), test and debug your application
4. Using Apache JMeter (https://jmeter.apache.org/) conduct a number of stress tests on your application. Run JMeter from the same machine, firing requests at `localhost:YOURPORT`
    a. Write two scenarios that use 50 threads:
        i. Scenario 1 repeatedly decides if the number 2147483647 is prime by invoking the app's isPrime URI

        ii. Scenario 2 first invokes the isPrime API for all numbers between 1 and 100; then, it repeatedly invokes the primesStored URI of the app
        iii. The repeating part of both scenarios should last 60s
    b. Run a number of experiments and collect Response-Time and Throughput data
        i. Try at least three different CPU limits for the web service
        ii. Try at least three different timer delays in JMeter
    c. Note that the visual part of jmeter might not tunnel properly through screen. In that case you can run it directly without screen or create your JMeter files on a different machine.
        i. It is actually preferable to not run the actual tests through the GUI mode of JMeter; instead, use the console
    d. You may create any scripts of your choice to automate the testing process
5. Write a report that graphs and discusses your stress-test results. Use your knowledge from queueing theory, Little's Law and computer systems to interpret the results. In your report, also include your code as well as the discussion about the test cases you used.
6. Create (you can preferably start this at an earlier stage to keep track of your commits) a git repository with all your work, including your JMeter files, result files and reports. Upload it to your personal GitHub account. Make it public so we (and future potential employers) can see it!
7. Create a zip file with all the documents, code and artifacts for submission

Deliverables

Submit your report to Moodle. Ensure it is well-written: it includes title, contents, clearly separated paragraphs, good English, etc. Document any hardware/software specifications you used. Ideally, someone should be able to replicate your results by following the instructions in your report!

Assessment Items

- Implementation of isPrime and isPrimesStored
- Functional testing design and actual test results
- Performance testing and its report
    - Two required jmx files (one for each scenario)
    - Stress testing and documenting two scenarios
        - Good presentation, English, etc.
        - Running the required 6 tests and collecting the data
        - Thorough analysis of performance testing results that includes discussion on Queuing Theory, Little's Law and Software/Computer Systems

Option B: Review Questions

Create and submit a document using a word processor answering all of the following questions. Ensure you properly title each question you answer

1. Suppose that you are given the following javadoc description for a method that was written by a colleague:

   `public static int` **`subString`**`(char[] pattern, char[] line)`

   Returns the index within the character array `line` of the first occurrence of the specified substring `pattern.`

   a) Could you perform an exhaustive test on every possible input parameter value? Explain your answer briefly.
   b) Suggest 5 specific test cases that you would use to test this method and explain why each test case is important.

2. Consider the following Java method:

```
public static long myMethod (long n){
      if (n < 0){
            return -1;
      }
      if (n <= 1){
            return 10;
      }
      long prev = 10;
      long cur = 10;
      for(long i = 2; i<n; i++){
            long temp = cur;
            cur += prev;
            prev = temp;
      }
      return cur;
}
```

a) Propose a test set for myMethod that results in a 100% statement coverage with the minimal number of test cases.
b) Propose a test set for myMethod that results in covering 5 different control-flows of the application with the minimal number of test cases.

3. Consider the following Dockerfile:

```
FROM ubuntu:latest

RUN apt-get update
RUN apt-get install -y python python-pip wget
RUN pip install Flask

ADD app.py /home/app.py

WORKDIR /home
```

   a. Describe the effect of the following command, if it were executed in the same directory as the Dockerfile and assuming it was able to run successfully:

      ```
      docker build -t compx341_19a/flask_app:latest .
      ```

   b. Describe the effect of the following command assuming it was able to run successfully:

      ```
      docker run -p 8081:5000 \
      -m 1g --cpuset-cpus="0,2,4" \
      compx341_19a/flask_app
      ```

   c. What would be the difference if `--cpuset-cpus="0,2,4"` was swapped with `--cpus=2.5` in the previous command?

4. Create an architectural component design that describes all possible elements described in the following docker-compose file:

```
version: "3"

services:

  redis:
    image: redis:3.2-alpine
    ports:
      - "6379"

  db:
    image: postgres:9.4
    volumes:
      - db-data:/var/lib/postgresql/data

  my-app:
    image: compx341_19a/myapp
    ports:
      - 5000:80
    depends_on:
      - redis
      - my-other-app
    deploy:
      mode: replicated
      replicas: 2

  my-other-app:
    image: compx341_19a/myotherapp
    ports:
      - "8080"
    depends_on:
      - db

volumes:
  db-data:
```

End of Assignment 5