Week 9 Graded Questions

Q1: Consider the flask application app.py and an HTML file index.html,

app.py:

```
from flask import Flask, jsonify
from datetime import datetime
import time

app = Flask(__name__)


@app.route('/')
def home():
    time.sleep(10)
    return jsonify(datetime.now().second), 200,
{'Access-Control-Allow-Origin': '*'}


@app.route('/profile')
def profile():
    time.sleep(30)
    return jsonify(datetime.now().second), 200,
{'Access-Control-Allow-Origin': '*'}


if __name__ == "__main__":
    app.run(debug=True)
```

index.html:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>Document</title>
  </head>
  <body>
    <script>
      function test() {
        const res1 = fetch('http://127.0.0.1:5000')
        const res2 = fetch('http://127.0.0.1:5000/profile')
        res1
          .then((res) => {
            return res.json()
          })
          .then((data) => {
            console.log(data)
          })

        res2
          .then((res) => {
            return res.json()
          })
          .then((data) => {
            console.log(data)
          })
      }
      test()
    </script>
  </body>
</html>
```

If a user visits index.html at 10:10:10 AM (in format HH:MM:SS), what will be logged in the console (approximately)?

A. 20, 50
B. 20, 40
C. 40, 40
D. 50, 50

Answer: B

Solution: The flask application is being run in the threaded mode (default). This means that the application will be able to accept and process multiple requests simultaneously. Thus, if 2 requests are sent at the same time (10:10:10), the route “/” will take a minimum of 10 seconds to respond, and route “/profile” will take a minimum of 30 seconds, but both will be processed simultaneously in different threads.

Q2: Consider the flask application app.py and an HTML file index.html,

app.py:

```
from flask import Flask, jsonify
from datetime import datetime
import time

app = Flask(__name__)


@app.route('/')
def home():
    time.sleep(10)
    return jsonify(datetime.now().second), 200,
{'Access-Control-Allow-Origin': '*'}


@app.route('/profile')
def profile():
    time.sleep(30)
    return jsonify(datetime.now().second), 200,
{'Access-Control-Allow-Origin': '*'}


if __name__ == "__main__":
    app.run(threaded=False, debug=True)
```

index.html:

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>Document</title>
  </head>
  <body>
    <script>
      function test() {
        const res1 = fetch('http://127.0.0.1:5000')
        const res2 = fetch('http://127.0.0.1:5000/profile')
        res1
          .then((res) => {
            return res.json()
          })
          .then((data) => {
            console.log(data)
          })

        res2
          .then((res) => {
            return res.json()
          })
          .then((data) => {
            console.log(data)
          })
      }
      test()
    </script>
  </body>
</html>
```

If a user visits index.html at 10:10:10 AM (in format HH:MM:SS), what will be logged in the console (approximately)?

A. 20, 50

B. 20, 40
C. 40, 40
D. 50, 50

Answer: A

Solution: The flask application is being run in the non-threaded mode. This means that the application will not accept any new request before it finishes processing the previous request. Thus, if 2 requests are sent at the same time (10:10:10), firstly, the route “/” will take a minimum of 10 seconds to respond, and then the next request will be accepted, and the route “/profile” will take a minimum of 30 seconds before responding.

Q3: Which of the following statements is/are true about Redis?

A. Data is always manipulated in the main computer memory (RAM).
B. It stores data on the disk by default.
C. Data cannot be reconstructed back.
D. Data is stored as key value pairs.

Answer: A and D

Answer: Redis is an in-memory database, which can be made to store data in the disk, and it stores the data in the form of key-value pairs, but it stores the data in main-memory by default. Even, if the data is stored in the disk, the data needs to be brought to the main memory for manipulating it.

Q4: Which of the following is/are true for Redis?

A. Redis supports multiple data types.
B. It is possible to set expiration time of data.
C. Authentication features are available.
D. TTL value of 1 implies that data will not expire.

Answer: A, B and C

Solution: Redis is an in-memory database, which supports various data types. It is always possible to store data with timeouts. The TTL value of -1 indicates that the data is not meant to be expired. It is also possible to have an authentication layer with Redis.

Q5: Which of the below is true for Celery?

A. Celery systems can consist of multiple brokers.
B. Workers and clients will automatically retry if connection is lost.
C. Redis and RabbitMQ are the only supported brokers with Celery.
D. It has features to monitor and schedule tasks.

Answer: A, B and D

Solution: Celery is a python based task queue, which provides multiples workers to perform heavy computed job asynchronously. It generally retries certain times in case of a connection loss. Celery supports brokers other than Redis and RabbitMQ, like Amazon SQS. It is always helpful for scheduling tasks or jobs.

Q6: Which of the following statements is false regarding concurrency?

A. The concurrency is achieved by executing multiple tasks overlapping in time periods, but not always simultaneously.
B. The concurrency can be achieved in both “single-core” and “multi-core” environments.
C. The concurrency is always achieved via “context switching” in a multicore environment.
D. Concurrency and Parallelism are not mutually exclusive.

Answer: C

Solution: Concurrency is the notion of carrying out multiple tasks or threads overlapping in time periods or even simultaneously (at the same time), not necessarily always. Concurrency is achieved via “context-switching” in a single core environment, whereas, the same can be achieved using “parallelism” in a multicore environment.

Q7: Which of the following statements is false regarding concurrency and parallelism?

A. The parallelism can be achieved in a “multi-core” environment.
B. It is possible to achieve parallelism without concurrency.
C. The parallelism is achieved by executing multiple tasks simultaneously.
D. It is possible to achieve concurrency without parallelism.

Answer: B

Solution: Kindly refer to this thread on discourse: [W9 Graded Assignment : Q7 - Courses / Modern Application Development - II - IITM-PDS]

Q8: Which of the following statement(s) is/are correct regarding message queue?

A. Message queue provides communication between components in a distributed architecture.
B. It provides a buffer which temporarily store messages.
C. In general, a message in message queue is processed more than once in point-to-point messaging.
D. In general, a message in message queue can be processed only once in point-to-point messaging.

Answer: A, B, D

Solution: Message queue provides a means of communication between different components in a distributed network. It does provide a buffer to store temporary messages. A message in a message queue is processed or consumed by only one consumer (in point-to-point messaging).

Q9: Which of the following is/are the true regarding message broker?

A. A message broker helps in reducing the number of connections between servers, if compared with many-to-many connections between servers.
B. It does not allow easier scalability, if compared with many-to-many connections between servers.
C. It can be used for batch processing of many similar types of requests.
D. All of the above

Answer: A and C

Solution: A message brokers provides easier scalability, if compared with many-to-many connections. The simple reason is that the addition of a new component or a server need not be connected to all other servers in the network, and it just needs to connected with the message broker. It is also useful for batch processing, where multiple requests of similar types and collected and processed together.

Q10: Which of the following is/are true regarding message broker?

A. They serve as intermediary between applications, which allow senders to issue a message without knowing the state of the receiver.
B. They serve as intermediary between applications, which allow senders to issue a message only after knowing the state of the receiver.
C. Inter application communication is often asynchronous.
D. Inter application communication is often synchronous.

Answer: A and C

Solution: A sender sending a message need not worry about the present state of the receiver, if the communication is happening using a message broker, as the message broker receives the message from the sender and inform the receiver once it is available to take new requests. This communication is generally asynchronous, as the sender need not wait for the receiver to accept the previous request before sending anew request.