Laboratorio 8

Sebastian Aristondo 20880
Daniel Gonzlaez 20293

```
import simpy
import random
import numpy as np
import matplotlib.pyplot as plt
import math
```

```
# Parámetros de la simulación
tiempo_simulacion = 3600  # Tiempo de simulación en segundos
tasa_llegada_solicitudes = 40  # Tasa de llegada de solicitudes al sistema (solicitudes por segun
tasa_atencion_servidor = 10  # Tasa de atención del servidor (solicitudes por segundo)

# Variables para almacenar métricas
solicitudes_atendidas_por_servidor = 0  # Inicialmente, el servidor no ha atendido ninguna solici
tiempo_servidor_ocupado = 0  # Tiempo que estuvo el servidor ocupado
tiempo_servidor_desocupado = 0  # Tiempo que estuvo el servidor desocupado
tiempo_total_solicitudes_en_cola = []  # Tiempo total que las solicitudes estuvieron en cola
total_solicitudes_en_cola = 1
ultimo_momento_salida_solicitud = 0
```

```
# Proceso para simular la llegada de solicitudes
def llegada_solicitudes(env, tasa_llegada, servidor):
    solicitud_id = 0
    while True:
        arrival = env.now
        yield env.timeout(np.random.exponential(1.0/tasa_llegada))
        solicitud_id += 1
        env.process(atender_solicitud(env, solicitud_id, servidor, arrival))

# Proceso para simular el proceso de atención de una solicitud por el servidor
def atender_solicitud(env, solicitud_id, servidor, arrival):
    llegada_solicitud = env.now
    # print(f"Llegada de solicitud {solicitud_id} en el tiempo {llegada_solicitud}")
    # Calcular tiempo de atención
    tiempo_atencion = np.random.exponential(1.0/tasa_atencion_servidor)

    # Registrar tiempo en que el servidor se ocupó
    global tiempo_servidor_ocupado
    tiempo_servidor_ocupado += tiempo_atencion

    # Atender la solicitud
```

```
    # print(f"arrival:{arrival}, tiempo_atencion:{tiempo_atencion}, tiempo_servidor_ocupado:{tiem
    with servidor.request() as solicitud:
        yield solicitud
        tiempo_total_solicitudes_en_cola.append(env.now - llegada_solicitud)
        yield env.timeout(tiempo_atencion)


    # Incrementar el contador de solicitudes atendidas por el servidor
    global solicitudes_atendidas_por_servidor
    solicitudes_atendidas_por_servidor += 1
    ultimo_momento_salida_solicitud = env.now
    # print(f"Atendiendo solicitud {solicitud_id} en el tiempo {env.now}")
```

Task 1

Mountain Mega Computing

```
# Parámetros de la simulación
tiempo_simulacion = 3600  # Tiempo de simulación en segundos
tasa_llegada_solicitudes = 40  # Tasa de llegada de solicitudes al sistema (solicitudes por segun
tasa_atencion_servidor = 100  # Tasa de atención del servidor (solicitudes por segundo)

# Variables para almacenar métricas
solicitudes_atendidas_por_servidor = 0  # Inicialmente, el servidor no ha atendido ninguna solici
tiempo_servidor_ocupado = 0  # Tiempo que estuvo el servidor ocupado
tiempo_servidor_desocupado = 0  # Tiempo que estuvo el servidor desocupado
tiempo_total_solicitudes_en_cola = []  # Tiempo total que las solicitudes estuvieron en cola
total_solicitudes_en_cola = 1


# Inicializar la simulación
env = simpy.Environment()

# Crear el servidor
servidor = simpy.Resource(env, capacity=1)

# Iniciar el proceso de llegada de solicitudes
env.process(llegada_solicitudes(env, tasa_llegada_solicitudes, servidor))

# Ejecutar la simulación
env.run(until=tiempo_simulacion)

# Calcular métricas finales
tiempo_total_simulacion = tiempo_simulacion * 1.0
tiempo_total_solicitudes_en_cola = sum(tiempo_total_solicitudes_en_cola)
promedio_tiempo_en_cola = tiempo_total_solicitudes_en_cola / solicitudes_atendidas_por_servidor
```

```
promedio_solicitudes_en_cola_por_segundo = total_solicitudes_en_cola / tiempo_total_simulacion
ultimo_momento_salida_solicitud = env.now
tiempo_servidor_desocupado = tiempo_total_simulacion - tiempo_servidor_ocupado


# Mostrar métricas finales
print("\nMétricas finales:")
print(f"a. Solicitudes atendidas por el servidor: {solicitudes_atendidas_por_servidor}")
print(f"b. Tiempo total en que el servidor estuvo ocupado: {tiempo_servidor_ocupado}")
print(f"c. Tiempo total en que el servidor estuvo desocupado: {tiempo_servidor_desocupado}")
print(f"d. Tiempo total que las solicitudes estuvieron en cola: {tiempo_total_solicitudes_en_cola
print(f"e. Promedio de tiempo que cada solicitud estuvo en cola: {promedio_tiempo_en_cola}")
print(f"f. Promedio de solicitudes en cola por segundo: {promedio_solicitudes_en_cola_por_segundo
print(f"g. Momento de salida de la última solicitud: {ultimo_momento_salida_solicitud}")
```

```
Métricas finales:
a. Solicitudes atendidas por el servidor: 143788
b. Tiempo total en que el servidor estuvo ocupado: 1443.0704766173947
c. Tiempo total en que el servidor estuvo desocupado: 2156.929523382605
d. Tiempo total que las solicitudes estuvieron en cola: 975.716628006596
e. Promedio de tiempo que cada solicitud estuvo en cola: 0.006785800122448299
f. Promedio de solicitudes en cola por segundo: 0.0002777777777777778
g. Momento de salida de la última solicitud: 3600
```

Pizzita computing

```
# Parámetros de la simulación
tiempo_simulacion = 3600  # Tiempo de simulación en segundos
tasa_llegada_solicitudes = 40  # Tasa de llegada de solicitudes al sistema (solicitudes por segun
tasa_atencion_servidor = 10  # Tasa de atención del servidor (solicitudes por segundo)

# Variables para almacenar métricas
solicitudes_atendidas_por_servidor = 0  # Inicialmente, el servidor no ha atendido ninguna solici
tiempo_servidor_ocupado = 0  # Tiempo que estuvo el servidor ocupado
tiempo_servidor_desocupado = 0  # Tiempo que estuvo el servidor desocupado
tiempo_total_solicitudes_en_cola = []  # Tiempo total que las solicitudes estuvieron en cola
total_solicitudes_en_cola = 1
capacidad = 10
ultimo_momento_salida_solicitud = 0

# Inicializar la simulación
env2 = simpy.Environment()

# Crear el servidor
servidor2 = simpy.Resource(env2, capacity=capacidad)

# Iniciar el proceso de llegada de solicitudes
env2.process(llegada_solicitudes(env2, tasa_llegada_solicitudes, servidor2))
```

```
# Ejecutar la simulación
env2.run(until=tiempo_simulacion)

# Calcular métricas finales
tiempo_total_simulacion = tiempo_simulacion * 1.0
tiempo_total_solicitudes_en_cola = sum(tiempo_total_solicitudes_en_cola) / capacidad
promedio_tiempo_en_cola = tiempo_total_solicitudes_en_cola / solicitudes_atendidas_por_servidor
promedio_solicitudes_en_cola_por_segundo = total_solicitudes_en_cola / tiempo_total_simulacion
ultimo_momento_salida_solicitud = env2.now
tiempo_servidor_ocupado = tiempo_servidor_ocupado / capacidad
tiempo_servidor_desocupado = ultimo_momento_salida_solicitud - tiempo_servidor_ocupado


# Mostrar métricas finales
print("\nMétricas finales:")
print(f"a. Solicitudes atendidas por el servidor: {solicitudes_atendidas_por_servidor}")
print(f"b. Tiempo total en que el servidor estuvo ocupado: {tiempo_servidor_ocupado}")
print(f"c. Tiempo total en que el servidor estuvo desocupado: {tiempo_servidor_desocupado}")
print(f"d. Tiempo total que las solicitudes estuvieron en cola: {tiempo_total_solicitudes_en_cola
print(f"e. Promedio de tiempo que cada solicitud estuvo en cola: {promedio_tiempo_en_cola}")
print(f"f. Promedio de solicitudes en cola por segundo: {promedio_solicitudes_en_cola_por_segundo
print(f"g. Momento de salida de la última solicitud: {ultimo_momento_salida_solicitud}")
```

```
Métricas finales:
a. Solicitudes atendidas por el servidor: 144244
b. Tiempo total en que el servidor estuvo ocupado: 1439.2687704563648
c. Tiempo total en que el servidor estuvo desocupado: 2160.731229543635
d. Tiempo total que las solicitudes estuvieron en cola: 2.03546035592852
e. Promedio de tiempo que cada solicitud estuvo en cola: 1.4111230664211475e-05
f. Promedio de solicitudes en cola por segundo: 0.0002777777777777778
g. Momento de salida de la última solicitud: 3600
```

Task 2

```
# Parámetros de la simulación
tiempo_simulacion = 3600  # Tiempo de simulación en segundos
tasa_llegada_solicitudes = 40  # Tasa de llegada de solicitudes al sistema (solicitudes por segun
tasa_atencion_servidor = 10  # Tasa de atención del servidor (solicitudes por segundo)

# Variables para almacenar métricas
solicitudes_atendidas_por_servidor = 0  # Inicialmente, el servidor no ha atendido ninguna solici
tiempo_servidor_ocupado = 0  # Tiempo que estuvo el servidor ocupado
tiempo_servidor_desocupado = 0  # Tiempo que estuvo el servidor desocupado
tiempo_total_solicitudes_en_cola = []  # Tiempo total que las solicitudes estuvieron en cola
total_solicitudes_en_cola = 1
capacidad = 16
```

```
ultimo_momento_salida_solicitud = 0

# Inicializar la simulación
env2 = simpy.Environment()

# Crear el servidor
servidor2 = simpy.Resource(env2, capacity=capacidad)

# Iniciar el proceso de llegada de solicitudes
env2.process(llegada_solicitudes(env2, tasa_llegada_solicitudes, servidor2))

# Ejecutar la simulación
env2.run(until=tiempo_simulacion)

# Calcular métricas finales
tiempo_total_simulacion = tiempo_simulacion * 1.0
tiempo_total_solicitudes_en_cola = sum(tiempo_total_solicitudes_en_cola) / capacidad
promedio_tiempo_en_cola = tiempo_total_solicitudes_en_cola / solicitudes_atendidas_por_servidor
promedio_solicitudes_en_cola_por_segundo = total_solicitudes_en_cola / tiempo_total_simulacion
ultimo_momento_salida_solicitud = env2.now
tiempo_servidor_ocupado = tiempo_servidor_ocupado / capacidad
tiempo_servidor_desocupado = ultimo_momento_salida_solicitud - tiempo_servidor_ocupado


# Mostrar métricas finales
print("\nMétricas finales:")
print(f"a. Solicitudes atendidas por el servidor: {solicitudes_atendidas_por_servidor}")
print(f"b. Tiempo total en que el servidor estuvo ocupado: {tiempo_servidor_ocupado}")
print(f"c. Tiempo total en que el servidor estuvo desocupado: {tiempo_servidor_desocupado}")
print(f"d. Tiempo total que las solicitudes estuvieron en cola: {tiempo_total_solicitudes_en_cola
print(f"e. Promedio de tiempo que cada solicitud estuvo en cola: {promedio_tiempo_en_cola}")
print(f"f. Promedio de solicitudes en cola por segundo: {promedio_solicitudes_en_cola_por_segundo
print(f"g. Momento de salida de la última solicitud: {ultimo_momento_salida_solicitud}")
```

```
Métricas finales:
a. Solicitudes atendidas por el servidor: 144778
b. Tiempo total en que el servidor estuvo ocupado: 906.7595614245647
c. Tiempo total en que el servidor estuvo desocupado: 2693.2404385754353
d. Tiempo total que las solicitudes estuvieron en cola: 0.0
e. Promedio de tiempo que cada solicitud estuvo en cola: 0.0
f. Promedio de solicitudes en cola por segundo: 0.0002777777777777778
g. Momento de salida de la última solicitud: 3600
```

Se probó con distintas configuraciones de servers. Primero se probó muy holgadamente, con 100 servers y se encontró que aquí ya no esperaba ninguna solicitud. Luego, se probó con 50 y 20 y sucedió lo mismo. Con 15 si se esperaba y con 16, dejó de esperar. Así que, de forma empírica, la cantiad óptima de servidores que hacen que no haya congestión en el sistema es de 16.

Task 3

Mountain Mega Computing

```
# Parámetros de la simulación
tiempo_simulacion = 3600  # Tiempo de simulación en segundos
tasa_llegada_solicitudes = 100  # Tasa de llegada de solicitudes al sistema (solicitudes por segu
tasa_atencion_servidor = 100  # Tasa de atención del servidor (solicitudes por segundo)

# Variables para almacenar métricas
solicitudes_atendidas_por_servidor = 0  # Inicialmente, el servidor no ha atendido ninguna solici
tiempo_servidor_ocupado = 0  # Tiempo que estuvo el servidor ocupado
tiempo_servidor_desocupado = 0  # Tiempo que estuvo el servidor desocupado
tiempo_total_solicitudes_en_cola = []  # Tiempo total que las solicitudes estuvieron en cola
total_solicitudes_en_cola = 1
capacidad = 1
ultimo_momento_salida_solicitud = 0

# Inicializar la simulación
env2 = simpy.Environment()

# Crear el servidor
servidor2 = simpy.Resource(env2, capacity=capacidad)

# Iniciar el proceso de llegada de solicitudes
env2.process(llegada_solicitudes(env2, tasa_llegada_solicitudes, servidor2))

# Ejecutar la simulación
env2.run(until=tiempo_simulacion)

# Calcular métricas finales
tiempo_total_simulacion = tiempo_simulacion * 1.0
tiempo_total_solicitudes_en_cola = sum(tiempo_total_solicitudes_en_cola) / capacidad
promedio_tiempo_en_cola = tiempo_total_solicitudes_en_cola / solicitudes_atendidas_por_servidor
promedio_solicitudes_en_cola_por_segundo = total_solicitudes_en_cola / tiempo_total_simulacion
ultimo_momento_salida_solicitud = env2.now
tiempo_servidor_ocupado = tiempo_servidor_ocupado / capacidad
tiempo_servidor_desocupado = ultimo_momento_salida_solicitud - tiempo_servidor_ocupado


# Mostrar métricas finales
print("\nMétricas finales:")
print(f"a. Solicitudes atendidas por el servidor: {solicitudes_atendidas_por_servidor}")
print(f"b. Tiempo total en que el servidor estuvo ocupado: {tiempo_servidor_ocupado}")
print(f"c. Tiempo total en que el servidor estuvo desocupado: {tiempo_servidor_desocupado}")
print(f"d. Tiempo total que las solicitudes estuvieron en cola: {tiempo_total_solicitudes_en_cola
print(f"e. Promedio de tiempo que cada solicitud estuvo en cola: {promedio_tiempo_en_cola}")
```

```
print(f"f. Promedio de solicitudes en cola por segundo: {promedio_solicitudes_en_cola_por_segundo
print(f"g. Momento de salida de la última solicitud: {ultimo_momento_salida_solicitud}")
```

Métricas finales:
a. Solicitudes atendidas por el servidor: 359571
b. Tiempo total en que el servidor estuvo ocupado: 3599.732904948749
c. Tiempo total en que el servidor estuvo desocupado: 0.2670950512510899
d. Tiempo total que las solicitudes estuvieron en cola: 895636.9752671174
e. Promedio de tiempo que cada solicitud estuvo en cola: 2.490848748278135
f. Promedio de solicitudes en cola por segundo: 0.0002777777777777778
g. Momento de salida de la última solicitud: 3600

```
# Parámetros de la simulación
tiempo_simulacion = 3600  # Tiempo de simulación en segundos
tasa_llegada_solicitudes = 100  # Tasa de llegada de solicitudes al sistema (solicitudes por segu
tasa_atencion_servidor = 10  # Tasa de atención del servidor (solicitudes por segundo)

# Variables para almacenar métricas
solicitudes_atendidas_por_servidor = 0  # Inicialmente, el servidor no ha atendido ninguna solici
tiempo_servidor_ocupado = 0  # Tiempo que estuvo el servidor ocupado
tiempo_servidor_desocupado = 0  # Tiempo que estuvo el servidor desocupado
tiempo_total_solicitudes_en_cola = []  # Tiempo total que las solicitudes estuvieron en cola
total_solicitudes_en_cola = 1
capacidad = 10
ultimo_momento_salida_solicitud = 0

# Inicializar la simulación
env2 = simpy.Environment()

# Crear el servidor
servidor2 = simpy.Resource(env2, capacity=capacidad)

# Iniciar el proceso de llegada de solicitudes
env2.process(llegada_solicitudes(env2, tasa_llegada_solicitudes, servidor2))

# Ejecutar la simulación
env2.run(until=tiempo_simulacion)

# Calcular métricas finales
tiempo_total_simulacion = tiempo_simulacion * 1.0
tiempo_total_solicitudes_en_cola = sum(tiempo_total_solicitudes_en_cola) / capacidad
promedio_tiempo_en_cola = tiempo_total_solicitudes_en_cola / solicitudes_atendidas_por_servidor
promedio_solicitudes_en_cola_por_segundo = total_solicitudes_en_cola / tiempo_total_simulacion
ultimo_momento_salida_solicitud = env2.now
tiempo_servidor_ocupado = tiempo_servidor_ocupado / capacidad
tiempo_servidor_desocupado = ultimo_momento_salida_solicitud - tiempo_servidor_ocupado
```

```
# Mostrar métricas finales
print("\nMétricas finales:")
print(f"a. Solicitudes atendidas por el servidor: {solicitudes_atendidas_por_servidor}")
print(f"b. Tiempo total en que el servidor estuvo ocupado: {tiempo_servidor_ocupado}")
print(f"c. Tiempo total en que el servidor estuvo desocupado: {tiempo_servidor_desocupado}")
print(f"d. Tiempo total que las solicitudes estuvieron en cola: {tiempo_total_solicitudes_en_cola
print(f"e. Promedio de tiempo que cada solicitud estuvo en cola: {promedio_tiempo_en_cola}")
print(f"f. Promedio de solicitudes en cola por segundo: {promedio_solicitudes_en_cola_por_segundo
print(f"g. Momento de salida de la última solicitud: {ultimo_momento_salida_solicitud}")
```

```
Métricas finales:
a. Solicitudes atendidas por el servidor: 360498
b. Tiempo total en que el servidor estuvo ocupado: 3589.389052065474
c. Tiempo total en que el servidor estuvo desocupado: 10.610947934525939
d. Tiempo total que las solicitudes estuvieron en cola: 66844.52802033506
e. Promedio de tiempo que cada solicitud estuvo en cola: 0.18542274303972575
f. Promedio de solicitudes en cola por segundo: 0.0002777777777777778
g. Momento de salida de la última solicitud: 3600
```

Búsqueda empírica

```
# Parámetros de la simulación
tiempo_simulacion = 3600  # Tiempo de simulación en segundos
tasa_llegada_solicitudes = 100  # Tasa de llegada de solicitudes al sistema (solicitudes por segu
tasa_atencion_servidor = 10  # Tasa de atención del servidor (solicitudes por segundo)

# Variables para almacenar métricas
solicitudes_atendidas_por_servidor = 0  # Inicialmente, el servidor no ha atendido ninguna solici
tiempo_servidor_ocupado = 0  # Tiempo que estuvo el servidor ocupado
tiempo_servidor_desocupado = 0  # Tiempo que estuvo el servidor desocupado
tiempo_total_solicitudes_en_cola = []  # Tiempo total que las solicitudes estuvieron en cola
total_solicitudes_en_cola = 1
capacidad = 29
ultimo_momento_salida_solicitud = 0

# Inicializar la simulación
env2 = simpy.Environment()

# Crear el servidor
servidor2 = simpy.Resource(env2, capacity=capacidad)

# Iniciar el proceso de llegada de solicitudes
env2.process(llegada_solicitudes(env2, tasa_llegada_solicitudes, servidor2))

# Ejecutar la simulación
env2.run(until=tiempo_simulacion)

# Calcular métricas finales
```

```
tiempo_total_simulacion = tiempo_simulacion * 1.0
tiempo_total_solicitudes_en_cola = sum(tiempo_total_solicitudes_en_cola) / capacidad
promedio_tiempo_en_cola = tiempo_total_solicitudes_en_cola / solicitudes_atendidas_por_servidor
promedio_solicitudes_en_cola_por_segundo = total_solicitudes_en_cola / tiempo_total_simulacion
ultimo_momento_salida_solicitud = env2.now
tiempo_servidor_ocupado = tiempo_servidor_ocupado / capacidad
tiempo_servidor_desocupado = ultimo_momento_salida_solicitud - tiempo_servidor_ocupado


# Mostrar métricas finales
print("\nMétricas finales:")
print(f"a. Solicitudes atendidas por el servidor: {solicitudes_atendidas_por_servidor}")
print(f"b. Tiempo total en que el servidor estuvo ocupado: {tiempo_servidor_ocupado}")
print(f"c. Tiempo total en que el servidor estuvo desocupado: {tiempo_servidor_desocupado}")
print(f"d. Tiempo total que las solicitudes estuvieron en cola: {tiempo_total_solicitudes_en_cola
print(f"e. Promedio de tiempo que cada solicitud estuvo en cola: {promedio_tiempo_en_cola}")
print(f"f. Promedio de solicitudes en cola por segundo: {promedio_solicitudes_en_cola_por_segundo
print(f"g. Momento de salida de la última solicitud: {ultimo_momento_salida_solicitud}")
```

```
Métricas finales:
a. Solicitudes atendidas por el servidor: 359453
b. Tiempo total en que el servidor estuvo ocupado: 1240.0631749947258
c. Tiempo total en que el servidor estuvo desocupado: 2359.936825005274
d. Tiempo total que las solicitudes estuvieron en cola: 0.0
e. Promedio de tiempo que cada solicitud estuvo en cola: 0.0
f. Promedio de solicitudes en cola por segundo: 0.0002777777777777778
g. Momento de salida de la última solicitud: 3600
```

Se puede observar que con 29 servidores se pueden manejar las 6000 solicitudes por minuto.

Task 4

Podemos observar que la usar Pizzita computing, si se aumentan las solicitudes por minuto de una manera importante, la cantidad de servidores va a aumentar casi de forma lineal, si se quiere lograr que las solicitudes no esperen. Por lo tanto, es necesario considerar adecuadamente cuál es el precio de los servidores y cuál es la carga esperada en los años venideros. De forma que puede ser que no sea beneficioso de forma económica contratar a esta empresa. Sin embargo, si no es imprescindible que las solicitudes no esperen, probablmente sea mejor usar a Pizzita que a Mountain, porque la cantidad de requests que manejan es similar, pero el tiempo de espera en cola es mucho menor en Pizzita que en mountain. Por esto, es posible que aunque Pizzita tenga menor capacidad computacional, se compensa con el tiempo de espera.

Además como esta planificando que en 2 años se llegara a esa cantidad de solicitudes, es mejor usar Pizzita, ya que se puede ir aumentando la cantidad de servidores de forma lineal, y no se tiene que pagar por servidores que no se estan usando. Eso permite que el crecimiento sea organico, sumandole que los tiempos de espera no aumentaran conforme aumenten las solicitudes.