

Trabajo Práctico 1

Arquitectura del Software (75.73)

2C 2021
Facultad de Ingeniería
Universidad de Buenos Aires

Integrante	Padrón
Aguerre, Nicolás Federico	102145
Colavecchia, Taiel Genaro	102510
Parafati, Mauro	102749

Índice

1. Sección 1	3
1.1. Configuración inicial del servicio	3
1.1.1. Endpoints	3
1.1.2. Infraestructuras	3
1.2. Descripción de casos de análisis	6
1.2.1. Load test: tasa creciente a ritmo constante	6
1.2.2. Spikes test: picos sobre una base constante	6
1.2.3. Endurance test: arribos a tasa constante de larga duracion	6
1.3. Endpoint Ping	7
1.3.1. Load test	7
1.3.1.1. Un nodo single-worker	7
1.3.1.2. Nodos single-worker replicados	8
1.3.1.3. Un nodo multi-worker	9
1.3.1.4. Nodos multi-worker replicados	10
1.3.1.5. Conclusión	10
1.4. Endpoint heavy	12
1.4.1. Load test	12
1.4.1.1. Un nodo single-worker	12
1.4.1.2. Nodos single-worker replicados	13
1.4.1.3. Un nodo multi-worker	14
1.4.1.4. Nodos multi-worker replicados	15
1.4.1.5. Conclusión	15
1.4.2. Endurance test	16
1.4.2.1. Un nodo single-worker	16
1.4.2.2. Nodos single-worker replicados	17
1.4.2.3. Un nodo multi-worker	18
1.4.2.4. Nodos multi-worker replicados	19
1.4.2.5. Conclusión	19
1.5. Endpoint bbox 1	20
1.5.1. Load test	20
1.5.1.1. Un nodo single-worker	20
1.5.1.2. Nodos single-worker replicados	21
1.5.1.3. Un nodo multi-worker	22
1.5.1.4. Nodos multi-worker replicados	23
1.5.1.5. Conclusión	23
1.5.2. Spikes test	24
1.5.2.1. Un nodo single-worker	24
1.5.2.2. Nodos single-worker replicados	25
1.5.2.3. Un nodo multi-worker	26
1.5.2.4. Nodos multi-worker replicados	27
1.5.2.5. Conclusión	27
1.5.3. Endurance test	28
1.5.3.1. Un nodo single-worker	28
1.5.3.2. Nodos single-worker replicados	29
1.5.3.3. Un nodo multi-worker	30
1.5.3.4. Nodos multi-worker replicados	31
1.5.3.5. Conclusión	31
1.6. Endpoint bbox 2	32
1.6.1. Load test	32
1.6.1.1. Un nodo single-worker	32
1.6.1.2. Nodos single-worker replicados	33

1.6.1.3.	Un nodo multi-worker	34
1.6.1.4.	Nodos multi-worker replicados	35
1.6.1.5.	Conclusión	35
1.6.2.	Spikes test	36
1.6.2.1.	Un nodo single-worker	36
1.6.2.2.	Nodos single-worker replicados	37
1.6.2.3.	Un nodo multi-worker	38
1.6.2.4.	Nodos multi-worker replicados	39
1.6.2.5.	Conclusión	39
1.6.3.	Endurance test	40
1.6.3.1.	Un nodo single-worker	40
1.6.3.2.	Nodos single-worker replicados	41
1.6.3.3.	Un nodo multi-worker	42
1.6.3.4.	Nodos multi-worker replicados	43
1.6.3.5.	Conclusión	43
2. Sección 2		44
2.1.	Objetivo	44
2.2.	Determinar la naturaleza de cada servicio	44
2.3.	Cantidad de workers del servicio sincrónico (bbox2)	45
2.4.	Tiempos de demora en la respuesta	48
2.4.1.	Servicio asincrónico	48
2.4.2.	Servicio sincrónico	49

1. Sección 1

1.1. Configuración inicial del servicio

Para desarrollar esta primer sección, se implementó un servicio HTTP en Node.js-Express, a partir del cual se realizarán mediciones ante distintos y variados escenarios de carga, con el objetivo de ver como se adapta nuestro sistema a cada caso.

1.1.1. Endpoints

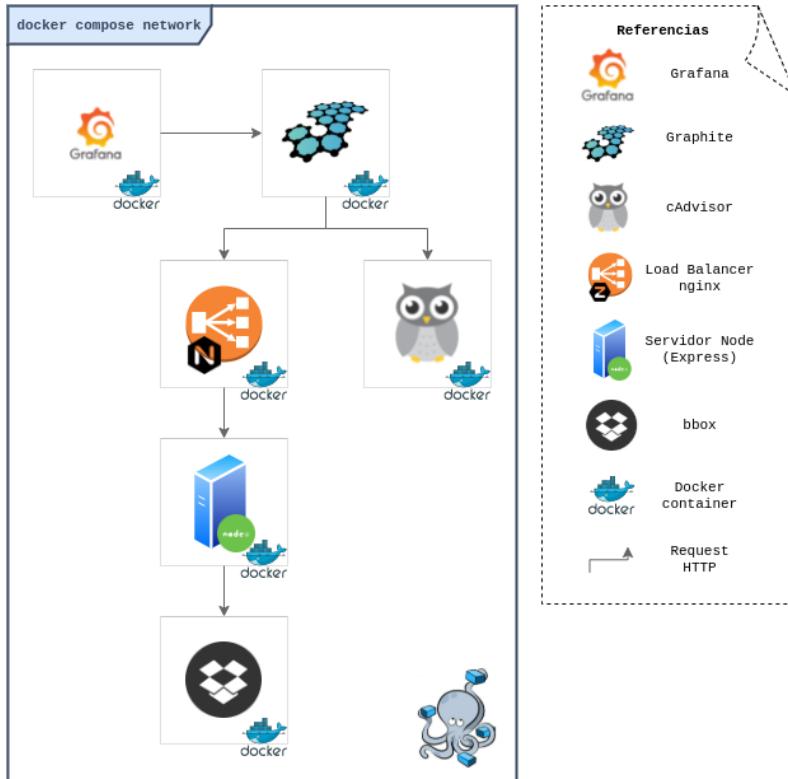
El servicio implementado ofrece los siguientes *endpoints*, cada uno con un objetivo de simulación distinto:

Endpoint	Ruta	Comportamiento
Ping	/ping	Healthcheck básico, respuesta constante inmediata
Intenso	/heavy	Cómputo intenso, alta tasa de utilización de CPU
Servicio 1	/bbox1	Consumo real de servicio de naturaleza desconocida provisto por la cátedra
Servicio 2	/bbox2	Consumo real de servicio de naturaleza desconocida provisto por la cátedra

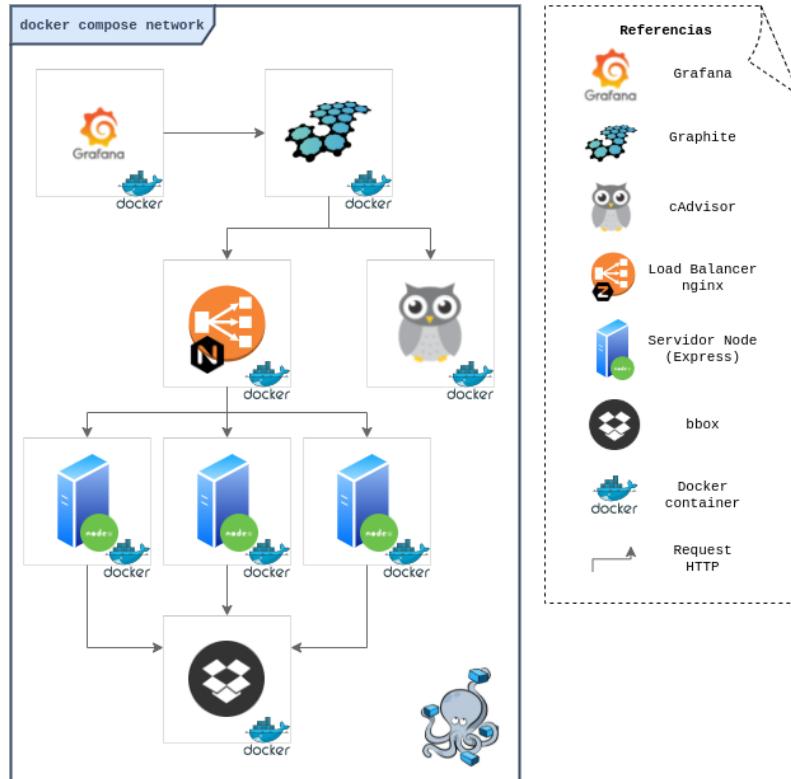
1.1.2. Infraestructuras

El servicio descripto será analizado bajo distintas configuraciones de deployment, con el objetivo de comparar distintos atributos de calidad ante cada escenario de carga.

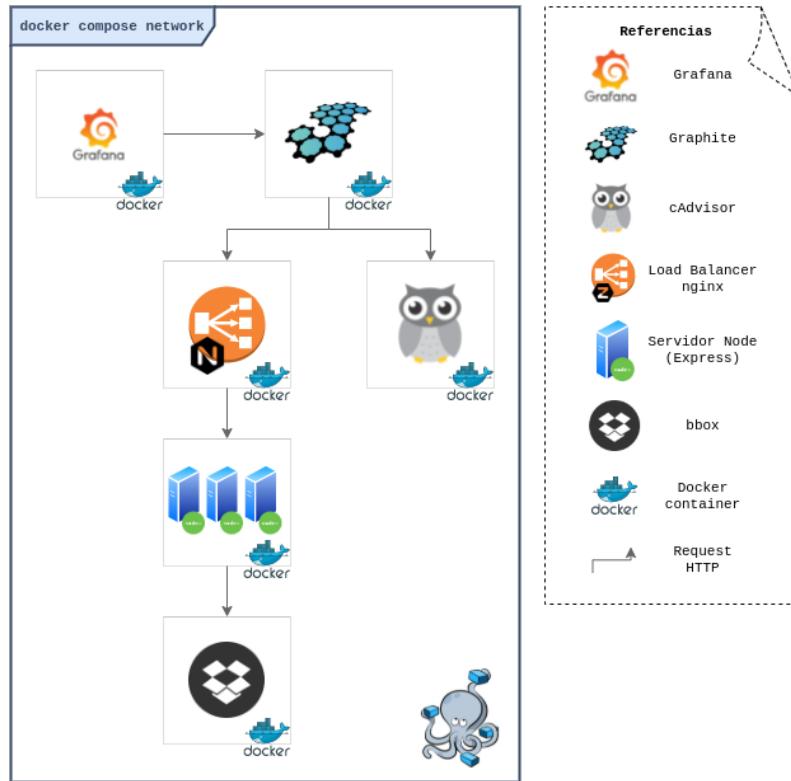
- **Un nodo single-worker:** un sólo container, corriendo un sólo proceso de la aplicación, constituida por un sólo worker.



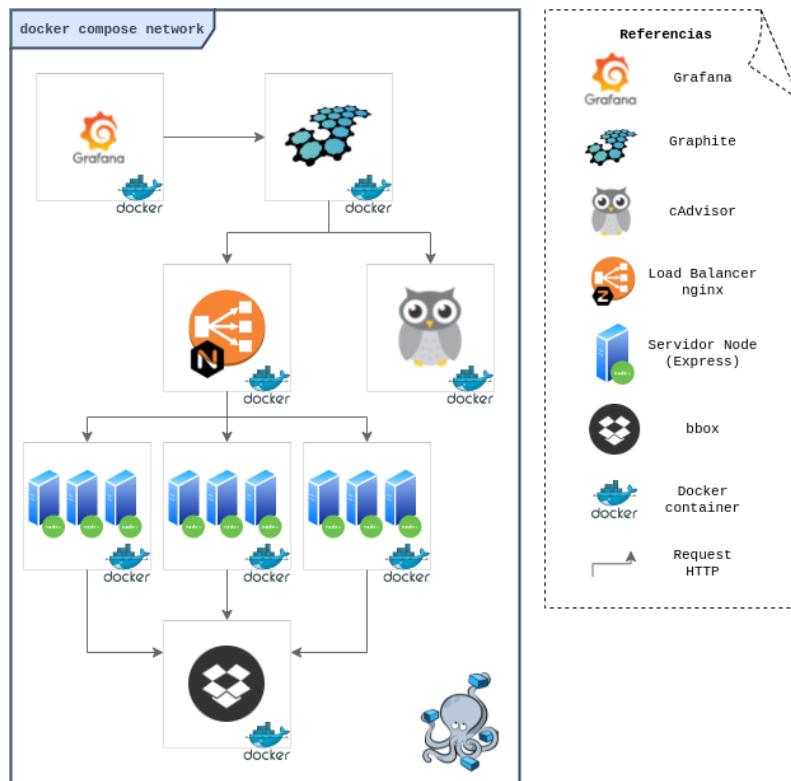
- **Nodos single-worker replicados:** múltiples containers replicando el proceso del servicio Node (que sigue teniendo un único worker), con *load balancing* a nivel *nginx*.



- **Un nodo multi-worker:** un sólo container, corriendo un sólo proceso de la aplicación, constituida por múltiples workers. Se utilizaron 3 workers en cada proceso. (contruido utilizando Node cluster)



- **Nodos multi-worker replicados:** los mismos procesos multi-worker descriptos en el ítem anterior, replicados en múltiples containers con balanceo de carga nuevamente a nivel *nginx*.



En todos los casos como ya se mencionó previamente se utiliza **nginx**, ya sea como balanceador de carga, o simplemente como proxy para el tráfico entre Cliente-Servidor.

1.2. Descripción de casos de análisis

1.2.1. Load test: tasa creciente a ritmo constante

Como primer caso de uso, se utilizó un tráfico creciente partiendo de 0 rq/s hasta llegar a una capacidad adecuada según el endpoint en estudio. De esta forma se podrán detectar los principales parámetros de interés del sistema, y cómo evoluciona el consumo de recursos con la carga del mismo.

1.2.2. Spikes test: picos sobre una base constante

Como segundo caso de estudio se utilizó un tráfico normal base (valores que pueda manejar el endpoint con facilidad) con picos de valores anormales de requests. Estos valores se mantienen durante 15 segundos separados por 45 de tiempo de ejecución normal, repitiendo este ciclo 2 veces. De esta forma se puede evaluar la elasticidad del sistema y su resiliencia contra incrementos repentinos de tráfico.

1.2.3. Endurance test: arribos a tasa constante de larga duración

Para verificar que el sistema no sufre pérdidas de memoria u otras fallas de ejecución ante un período prolongado de funcionamiento esperado, se propone este escenario que consiste en someterlo a una tasa de arribos constante de una duración extendida.

1.3. Endpoint Ping

Para este primer endpoint que, como ya se mencionó, se trata de un health check básico con respuesta constante, se decidió que alcanza con hacerle un **load test** (arribos a tasa creciente en forma de rampa), con el objetivo de poder detectar un posible punto de quiebre en el cual el sistema deje de comportarse de forma esperada y comience a degradar su performance.

1.3.1. Load test

El escenario en este caso particular consistió en comenzar con 0 rq/s hasta llegar a 1000 rq/s, en 120 segundos (es decir, con una tasa de crecimiento: 8,33).

1.3.1.1 Un nodo single-worker



Figura 1: Load test para ping con un nodo single-worker

```

Scenarios launched: 60397
Scenarios completed: 60397
Requests completed: 60397
Mean response/sec: 298.63
Response time (msec):
    min: 0
    max: 97
    median: 4
    p95: 25
    p99: 42
Scenario counts:
    endpoint: 60397 (100%)
Codes:
    200: 60397
  
```

1.3.1.2 Nodos single-worker replicados



Figura 2: Load test para ping con tres nodos single-worker replicados

```

Scenarios launched: 60381
Scenarios completed: 60381
Requests completed: 60381
Mean response/sec: 288.04
Response time (msec):
    min: 0
    max: 65
    median: 2
    p95: 5
    p99: 8
Scenario counts:
    endpoint: 60381 (100%)
Codes:
    200: 60381
  
```

1.3.1.3 Un nodo multi-worker



Figura 3: Load test para ping con un nodo multi-worker

```

Scenarios launched: 60266
Scenarios completed: 60266
Requests completed: 60266
Mean response/sec: 296.16
Response time (msec):
    min: 0
    max: 128
    median: 4
    p95: 26
    p99: 44
Scenario counts:
    endpoint: 60266 (100%)
Codes:
    200: 60266
  
```

1.3.1.4 Nodos multi-worker replicados



Figura 4: Load test para ping con tres nodos multi-worker replicados

```

Scenarios launched: 60455
Scenarios completed: 60455
Requests completed: 60455
Mean response/sec: 297.51
Response time (msec):
    min: 0
    max: 75
    median: 2
    p95: 6
    p99: 9
Scenario counts:
    endpoint: 60455 (100%)
Codes:
    200: 60455
  
```

1.3.1.5 Conclusión

Si se considera el caso de un sólo nodo single-worker, se puede apreciar que en la primer parte de la curva, el tiempo medio de respuesta para las requests realizadas se mantiene prácticamente estable en un valor menor a 50ms. Se cree que esto puede atribuirse al bajo procesamiento que se requiere del lado del servidor para procesar dichas requests. Sin embargo, se observa que aproximadamente al llegar a un acumulado de 4k requests, comienza a notarse un aumento en el response time, lo que entonces marcaría nuestro primer punto de quiebre.

Por otro lado, comparando distintas estrategias de deployment, algo interesante para notar es que los tiempos de respuesta son aproximadamente los mismos para el caso de un nodo con un solo worker y un nodo con multiples workers. Sin embargo, el uso de recursos para un nodo con múltiples workers es mayor. Es posible concluir que el incremento en el uso de recursos no justifica la utilizacion de multiples workers para atender las requests de este endpoint, al menos desde un

punto de vista de performance (ya que en cuanto a availability sí sería conveniente, por ejemplo).

En el caso de multiples nodos replicados (ya sea con uno o multiples workers), el tiempo de respuesta se mantiene aproximadamente constante al aumentar la cantidad de requests, lo que según el contexto podría ser un justificativo interesante para utilizar multiples instancias. Sin embargo, se puede apreciar que el valor mínimo para pocos requests es mayor que trabajando con único nodo lo que indica el overhead que implica hacer *load balancing*.

Con respecto a la utilización de recursos, se debe aclarar que en los casos de nodos replicados se omitieron los gráficos de los nodos 2 y 3 ya que se trataban de un espejo del que se muestra. Dicho esto, se puede notar que la utilización de recursos será más baja en cada nodo lo que nos indica que cada uno de ellos no requiere tanta capacidad de procesamiento (el sistema es más escalable).

1.4. Endpoint heavy

Para este endpoint, que como se describió previamente simula cómputo intensivo en el event loop principal del servicio en Node, se decidió someterlo a dos distintos escenarios de prueba: el **load test** y el **endurance test**.

El primer test (arribos a tasa creciente en forma de rampa) tiene el mismo objetivo que en casos de prueba anteriores: detectar el punto de quiebre del sistema, y analizar la evolución de los distintos parámetros del mismo a medida que aumenta la cantidad de arribos.

El segundo test tiene como objetivo estudiar el comportamiento a largo del sistema cuando se lo somete a una tasa constante de request de procesamiento intensivo, para poder detectar distintos parámetros de interés (tasas de completitud, response times, e incluso síntomas más complejos como memory leaks, detectables gracias al monitoreo de recursos).

1.4.1. Load test

El escenario en este caso particular consistió en comenzar con 0 rq/s hasta llegar a 20 rq/s, en 120 segundos (tasa de crecimiento: 0,16). Se decidió reducir la cantidad de requests a enviar en este caso debido a que se trata de un endpoint de procesamiento intensivo.

1.4.1.1 Un nodo single-worker



Figura 5: Load test para heavy con un nodo single-worker

```

Scenarios launched: 1215
Scenarios completed: 2
Requests completed: 2
Mean response/sec: 9.05
Response time (msec):
    min: 5043
    max: 9702
  
```

```

median: 7372.5
p95: 9702
p99: 9702
Scenario counts:
  endpoint: 1215 (100%)
Codes:
  200: 2
Errors:
  ETIMEDOUT: 1213

```

1.4.1.2 Nodos single-worker replicados



Figura 6: Load test para heavy con tres nodos single-worker replicados

```

Scenarios launched: 1262
Scenarios completed: 7
Requests completed: 7
Mean response/sec: 9.15
Response time (msec):
  min: 5008
  max: 7733
  median: 5032
  p95: 7733
  p99: 7733
Scenario counts:
  endpoint: 1262 (100%)
Codes:
  200: 7
Errors:
  ETIMEDOUT: 1255

```

1.4.1.3 Un nodo multi-worker



Figura 7: Load test para heavy con un nodo multi-worker

```

Scenarios launched: 1238
Scenarios completed: 8
Requests completed: 8
Mean response/sec: 8.78
Response time (msec):
  min: 5014
  max: 9725
  median: 5911
  p95: 9725
  p99: 9725
Scenario counts:
  endpoint: 1238 (100%)
Codes:
  200: 8
Errors:
  ETIMEDOUT: 1230

```

1.4.1.4 Nodos multi-worker replicados



Figura 8: Load test para heavy con tres nodos multi-worker replicados

```

Scenarios launched: 1176
Scenarios completed: 36
Requests completed: 36
Mean response/sec: 8.72
Response time (msec):
    min: 4997
    max: 9772
    median: 6721.5
    p95: 9703.7
    p99: 9772
Scenario counts:
    endpoint: 1176 (100%)
Codes:
    200: 36
Errors:
    ETIMEDOUT: 1140
  
```

1.4.1.5 Conclusión

Debido a la naturaleza sincrónica de este endpoint, es inevitable que se produzca una caída al alcanzar una cierta tasa (baja) de req/s. En todos los casos, la utilización de CPU es del 100 % (notar que es cercano al triple para los casos de multi-workers, esto se debe a que se muestra el trabajo de 3 threads dentro de un solo nodo).

La principal diferencia entre todas las configuraciones de deploy es que en aquellas que tienen mas de un worker y/o mas de un nodo, la cantidad de requests que puede ser servida antes de que

el servicio deje de responder es (levemente) mayor.

Se debe aclarar entonces que un servicio de este tipo **no tiene buena disponibilidad**.

1.4.2. Endurance test

Arribos a tasa constante de 1 rq/s, durante 3 minutos.

1.4.2.1 Un nodo single-worker



Figura 9: Endurance test para heavy con un nodo single-worker

```

Scenarios launched: 180
Scenarios completed: 7
Requests completed: 7
Mean response/sec: 0.86
Response time (msec):
    min: 4998
    max: 29020
    median: 17030
    p95: 29020
    p99: 29020
Scenario counts:
    endpoint: 180 (100%)
Codes:
    200: 7
Errors:
    ETIMEDOUT: 173
  
```

1.4.2.2 Nodos single-worker replicados



Figura 10: Endurance test para heavy con tres nodos single-worker replicados

```

Scenarios launched: 180
Scenarios completed: 39
Requests completed: 39
Mean response/sec: 0.86
Response time (msec):
    min: 5001
    max: 29031
    median: 17018
    p95: 29024.8
    p99: 29031
Scenario counts:
    endpoint: 180 (100%)
Codes:
    200: 39
Errors:
    ETIMEDOUT: 141
  
```

1.4.2.3 Un nodo multi-worker



Figura 11: Endurance test para heavy con un nodo multi-worker

```

Scenarios launched: 180
Scenarios completed: 39
Requests completed: 39
Mean response/sec: 0.86
Response time (msec):
    min: 4999
    max: 29071
    median: 18025
    p95: 29038.5
    p99: 29071
Scenario counts:
    endpoint: 180 (100%)
Codes:
    200: 39
Errors:
    ETIMEDOUT: 141
  
```

1.4.2.4 Nodos multi-worker replicados



Figura 12: Endurance test para heavy con tres nodos multi-worker replicados

```

Scenarios launched: 180
Scenarios completed: 180
Requests completed: 180
Mean response/sec: 0.9
Response time (msec):
    min: 4996
    max: 5041
    median: 4999
    p95: 5033
    p99: 5040
Scenario counts:
    endpoint: 180 (100%)
Codes:
    200: 180
  
```

1.4.2.5 Conclusión

Es notable que incluso ante la baja tasa de 1 req/s, el servicio no es capaz de soportar la carga excepto en su configuración de múltiples nodos, pudiendo repartir el trabajo en cada uno de sus workers.

Se puede hacer una analogía en la cantidad de respuestas que pueden dar tanto el caso de nodos simples replicados como el caso de múltiples workers en un solo nodo. Ambos tienen la misma capacidad de responder a las consultas antes de quedar deshabilitados por la acumulación de carga.

Por último, puede ser notado que la utilización de CPU sigue siendo alta aunque no se respondan las requests al igual que en el caso de Load Testing anterior.

1.5. Endpoint bbox 1

Dado que este es un endpoint correspondiente a un servicio provisto por la cátedra cuyo funcionamiento interno se desconoce, los tests fueron elegidos de forma que los mismos ayuden a determinar la naturaleza de este. En primer lugar, se corre un **load test** (arribos a tasa creciente en forma de rampa), con el objetivo de observar el comportamiento de los tiempos de respuesta según incrementa la carga, y a partir de esto derivar conclusiones sobre la naturaleza sincrona o asincrona del procesamiento realizado en el endpoint.

1.5.1. Load test

Al igual que en el caso del primer endpoint analizado (ping), se diseñó una rampa desde 0 rq/s hasta llegar a 1000 rq/s, en 120 segundos (tasa de crecimiento: 8,33).

1.5.1.1 Un nodo single-worker



Figura 13: Load test para bbox1 con un nodo single-worker

```

Scenarios launched: 60274
Scenarios completed: 60274
Requests completed: 60274
Mean response/sec: 290.24
Response time (msec):
    min: 1099
    max: 1325
    median: 1111
    p95: 1146
    p99: 1178
Scenario counts:
    endpoint: 60274 (100%)
Codes:
    200: 60274
  
```

1.5.1.2 Nodos single-worker replicados

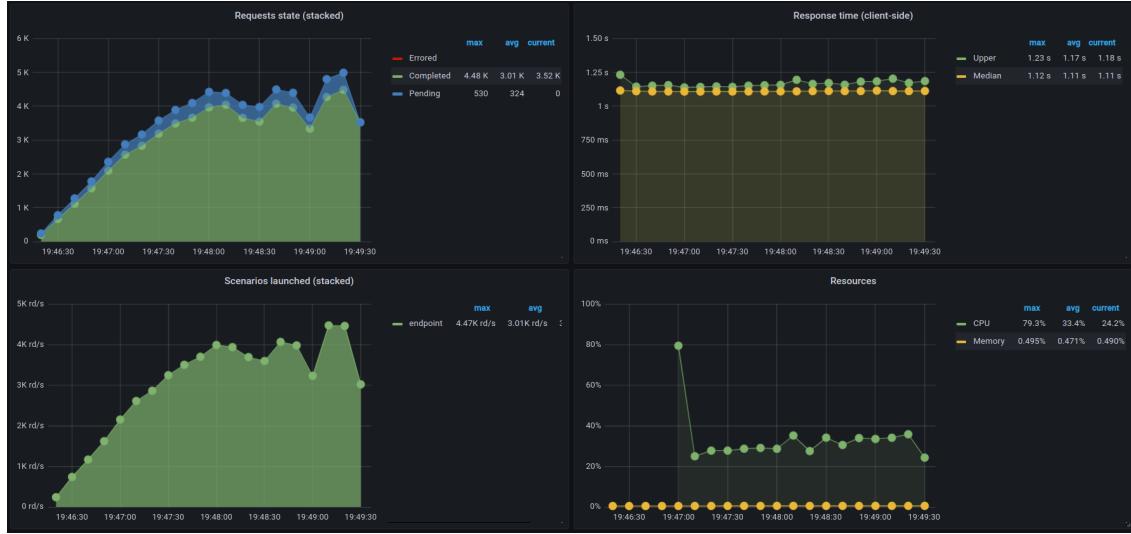


Figura 14: Load test para bbox1 con tres nodos single-worker replicados

```

Scenarios launched: 60196
Scenarios completed: 60196
Requests completed: 60196
Mean response/sec: 277.06
Response time (msec):
    min: 1092
    max: 1207
    median: 1105
    p95: 1112
    p99: 1123
Scenario counts:
    endpoint: 60196 (100%)
Codes:
    200: 60196
  
```

1.5.1.3 Un nodo multi-worker

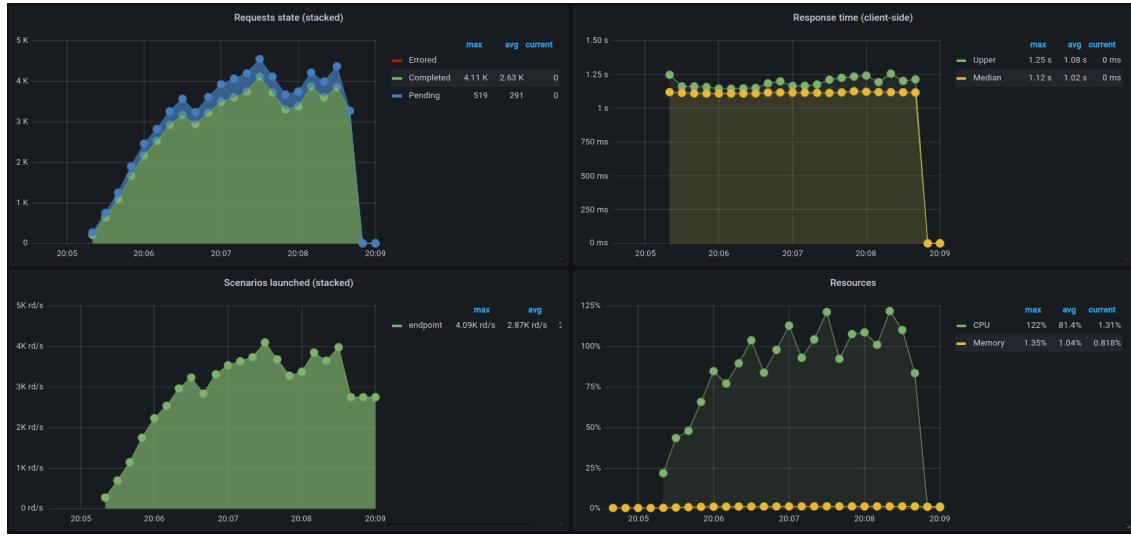


Figura 15: Load test para bbox1 con un nodo multi-worker

```

Scenarios launched: 60415
Scenarios completed: 60415
Requests completed: 60415
Mean response/sec: 265.9
Response time (msec):
    min: 1097
    max: 1244
    median: 1107
    p95: 1130
    p99: 1148
Scenario counts:
    endpoint: 60415 (100%)
Codes:
    200: 60415
  
```

1.5.1.4 Nodos multi-worker replicados



Figura 16: Load test para bbox1 con tres nodos multi-worker replicados

```

Scenarios launched: 60285
Scenarios completed: 60285
Requests completed: 60285
Mean response/sec: 269.31
Response time (msec):
    min: 1094
    max: 1206
    median: 1106
    p95: 1113
    p99: 1123
Scenario counts:
    endpoint: 60285 (100%)
Codes:
    200: 60285
  
```

1.5.1.5 Conclusión

En primer lugar, se puede notar que para cualquiera de las configuraciones de deployment utilizadas, el response time se mantiene prácticamente constante, alrededor de los 1100ms. Esto tiene sentido ya que si bien se aumentó el número de workers o de nodos replicados que llaman al servicio de bbox, este sigue teniendo la misma configuración en todos estos casos, por lo que es esperable que no se modifique el comportamiento observado.

En segundo lugar, resulta interesante que el tiempo de respuesta de las solicitudes es casi constante en función de la tasa de requests. Esto nos da un indicio de que el servicio tiene un funcionamiento asincrónico, favoreciendo la escalabilidad del mismo.

En cuanto al consumo de recursos, se observa que la memoria se mantiene prácticamente al mínimo, mientras que el uso de CPU se reparte entre los distintos workers disponibles en los nodos, aumentando según la cantidad de requests que se recibe.

1.5.2. Spikes test

Sobre una base de 100 rq/s constantes, se realizan dos ráfagas de requests (picos) de 1000 rq/s de 15s de duración, espaciadas entre sí por 45s.

1.5.2.1 Un nodo single-worker



Figura 17: Spikes test para bbox1 con un nodo single-worker

```

Scenarios launched: 43500
Scenarios completed: 41955
Requests completed: 41955
Mean response/sec: 214.15
Response time (msec):
    min: 1100
    max: 1304
    median: 1106
    p95: 1151
    p99: 1183
Scenario counts:
    endpoint: 43500 (100%)
Codes:
    200: 41955
Errors:
    ECONNRESET: 1545
  
```

1.5.2.2 Nodos single-worker replicados



Figura 18: Spikes test para bbox1 con tres nodos single-worker replicados

```

Scenarios launched: 43500
Scenarios completed: 43500
Requests completed: 43500
Mean response/sec: 200.64
Response time (msec):
    min: 1090
    max: 1176
    median: 1105
    p95: 1113
    p99: 1125
Scenario counts:
    endpoint: 43500 (100%)
Codes:
    200: 43500
  
```

1.5.2.3 Un nodo multi-worker



Figura 19: Spikes test para bbox1 con un nodo multi-worker

```

Scenarios launched: 43500
Scenarios completed: 43500
Requests completed: 43500
Mean response/sec: 196.6
Response time (msec):
    min: 1100
    max: 1318
    median: 1106
    p95: 1138
    p99: 1168
Scenario counts:
    endpoint: 43500 (100%)
Codes:
    200: 43500
  
```

1.5.2.4 Nodos multi-worker replicados



Figura 20: Spikes test para bbox1 con tres nodos multi-worker replicados

```

Scenarios launched: 43500
Scenarios completed: 43500
Requests completed: 43500
Mean response/sec: 199.01
Response time (msec):
    min: 1092
    max: 1203
    median: 1106
    p95: 1114
    p99: 1127
Scenario counts:
    endpoint: 43500 (100%)
Codes:
    200: 43500
  
```

1.5.2.5 Conclusión

Si bien en todos los casos el sistema funciona ante los saltos repentinos en carga, claramente se puede notar (por los errores que se generan) que el caso más simple no tiene la flexibilidad para manejar las altas cantidades de requests. Esto sucede al mismo tiempo que los recursos de procesamiento se acaban, no son suficientes en un solo nodo con un solo worker. El sistema se comporta de forma mas estable (sin errores) ante una configuración de multiples workers.

Al igual que en la prueba de carga, el tiempo de respuesta se mantiene aproximadamente constante ante los incrementos repentinos de la tasa de requests (aunque es posible verificar un leve incremento de los valores máximos en los tiempos de respuesta).

Respecto del uso de recursos, como es de esperar, el uso de procesador se dispara mostrando una correlación directa con el incremento en la cantidad de requests.

1.5.3. Endurance test

Arribos a tasa constante de 200 rq/s, durante 3 minutos.

1.5.3.1 Un nodo single-worker



Figura 21: Endurance test para bbox1 con un nodo single-worker

Scenarios launched: 36000

Scenarios completed: 36000

Requests completed: 36000

Mean response/sec: 179.43

Response time (msec):

min: 1100

max: 1141

median: 1103

p95: 1105

p99: 1109

Scenario counts:

endpoint: 36000 (100%)

Codes:

200: 36000

1.5.3.2 Nodos single-worker replicados



Figura 22: Endurance test para bbox1 con un nodo multi-worker

```

Scenarios launched: 36000
Scenarios completed: 36000
Requests completed: 36000
Mean response/sec: 179.43
Response time (msec):
    min: 1101
    max: 1143
    median: 1103
    p95: 1104
    p99: 1107
Scenario counts:
    endpoint: 36000 (100%)
Codes:
    200: 36000
  
```

1.5.3.3 Un nodo multi-worker

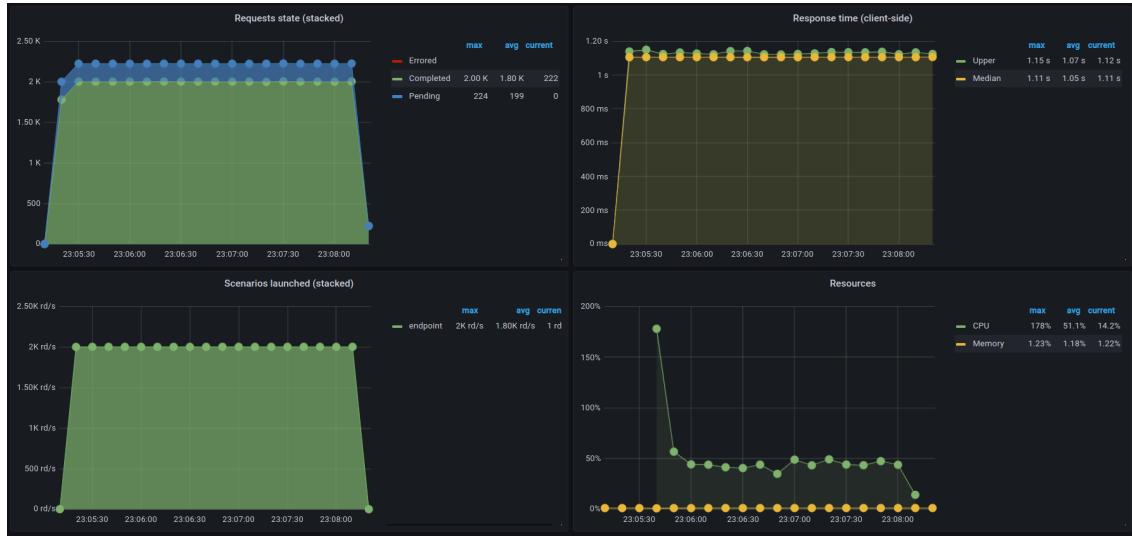


Figura 23: Endurance test para bbox1 con tres nodos single-worker replicados

```

Scenarios launched: 36000
Scenarios completed: 36000
Requests completed: 36000
Mean response/sec: 179.37
Response time (msec):
    min: 1098
    max: 1147
    median: 1103
    p95: 1105
    p99: 1108
Scenario counts:
    endpoint: 36000 (100%)
Codes:
    200: 36000
  
```

1.5.3.4 Nodos multi-worker replicados

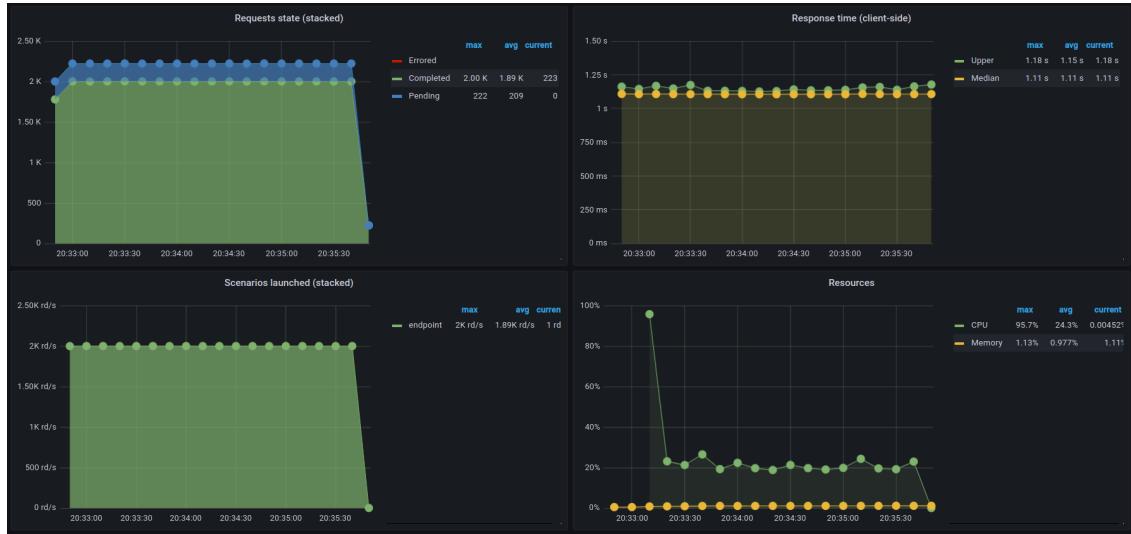


Figura 24: Endurance test para bbox1 con tres nodos multi-worker replicados

```

Scenarios launched: 36000
Scenarios completed: 36000
Requests completed: 36000
Mean response/sec: 179.39
Response time (msec):
    min: 1097
    max: 1171
    median: 1104
    p95: 1107
    p99: 1114
Scenario counts:
    endpoint: 36000 (100%)
Codes:
    200: 36000
  
```

1.5.3.5 Conclusión

Al igual que en los casos de test anteriores, el tiempo de respuesta se mantiene muy estable en la marca de los 1050ms aproximadamente.

Se puede apreciar sin embargo que para el caso de un nodo con un único worker, la cantidad de solicitudes pendientes de respuesta para cualquier momento dado es mucho mayor que para el caso de un nodo con múltiples workers, y para nodos replicados con un único worker. Esto da cuenta de que puede existir un cuello de botella en la aplicación desarrollada (que es la que consume el servicio bbox), debido a que al aparecer más instancias capaces de consumir dicho servicio, la cantidad de requests pendientes disminuye.

El uso de recursos se mantiene estable y bajo durante todos los tests.

1.6. Endpoint bbox 2

Al igual que bbox 1, este es un endpoint correspondiente a un servicio provisto por la cátedra cuyo funcionamiento interno se desconoce, por lo que los tests fueron elegidos de forma que los mismos ayuden a determinar la naturaleza de este. En primer lugar, se corre un **load test** (arribos a tasa creciente en forma de rampa), con el objetivo de observar el comportamiento de los tiempos de respuesta según incrementa la carga, y a partir de esto derivar conclusiones sobre la naturaleza sincronica o asincronica del procesamiento realizado en el endpoint.

1.6.1. Load test

Nuevamente, se diseñó una rampa desde 0 rq/s hasta llegar a 30 rq/s, en 120 segundos (tasa de crecimiento: 8,33).

1.6.1.1 Un nodo single-worker



Figura 25: Load test para bbox2 con un nodo single-worker

```

Scenarios launched: 1821
Scenarios completed: 56
Requests completed: 56
Mean response/sec: 13.36
Response time (msec):
    min: 907
    max: 9885
    median: 3267
    p95: 9299.9
    p99: 9883.7
Scenario counts:
    endpoint: 1821 (100%)
Codes:
    200: 56
  
```

Errors:
ETIMEDOUT: 1765

1.6.1.2 Nodos single-worker replicados



Figura 26: Load test para bbox2 con tres nodos single-worker replicados

```

Scenarios launched: 1819
Scenarios completed: 57
Requests completed: 57
Mean response/sec: 12.9
Response time (msec):
    min: 907
    max: 9872
    median: 3587
    p95: 9659.5
    p99: 9866.3
Scenario counts:
    endpoint: 1819 (100%)
Codes:
    200: 57
Errors:
    ETIMEDOUT: 1762
  
```

1.6.1.3 Un nodo multi-worker



Figura 27: Load test para bbox2 con un nodo multi-worker

```

Scenarios launched: 1839
Scenarios completed: 50
Requests completed: 50
Mean response/sec: 13.41
Response time (msec):
    min: 907
    max: 9541
    median: 4776
    p95: 9232
    p99: 9541
Scenario counts:
    endpoint: 1839 (100%)
Codes:
    200: 50
Errors:
    ETIMEDOUT: 1789
  
```

1.6.1.4 Nodos multi-worker replicados



Figura 28: Load test para bbox2 con tres nodos multi-worker replicados

```

Scenarios launched: 1814
Scenarios completed: 57
Requests completed: 57
Mean response/sec: 13.23
Response time (msec):
    min: 908
    max: 9656
    median: 2448
    p95: 8955.9
    p99: 9651.2
Scenario counts:
    endpoint: 1814 (100%)
Codes:
    200: 57
Errors:
    ETIMEDOUT: 1757
  
```

1.6.1.5 Conclusión

Lo primero a notar es que, al igual que con el primer servicio de bbox, el comportamiento fue prácticamente el mismo en todas las configuraciones de deployment que se probaron. Esto tiene sentido y la razón es la misma: el servicio de bbox mantiene el mismo setup, por lo que por más que se aumente la cantidad de workers que hacen los llamados al mismo, no va a cambiar el procesamiento de los mismos. Esto vale para todos los escenarios corridos para este endpoint.

Sin embargo, se puede notar que en este caso y a diferencia de lo que se observó en el primer servicio, rápidamente los requests comienzan a acumularse como pendientes para finalmente fallar. De hecho, se puede ver como en un punto bastante temprano en la rampa, el servicio directamente deja de responder. Lo que sucedió es que este escenario funcionó como prueba de estrés. En el

reporte es posible ver que solo 56 requests de 1821 se completaron con éxito.

El tiempo de respuesta aumentó de forma exponencial mientras el servicio estuvo activo, llegando hasta los 10 segundos antes de caerse.

El comportamiento observado lleva a concluir que se trata de un servicio pobremente escalable, que se satura ante una cantidad de requests muy pequeña, por lo que se puede intuir que su implementación es sincrónica e inefficiente.

En ninguno de los casos el uso de recursos llega a ser notorio o determinante, manteniéndose por debajo de límites razonables.

1.6.2. Spikes test

Sobre una base de 1 rq/s constante, se realizan dos ráfagas de requests (picos) de 4 rq/s de 15s de duración, espaciadas entre sí por 45s.

1.6.2.1 Un nodo single-worker



Figura 29: Spikes test para bbox2 con un nodo single-worker

```

Scenarios launched: 255
Scenarios completed: 217
Requests completed: 217
Mean response/sec: 1.46
Response time (msec):
    min: 903
    max: 9963
    median: 2105
    p95: 9309
    p99: 9792.2
Scenario counts:
    endpoint: 255 (100%)
Codes:

```

```
200: 217
Errors:
ETIMEDOUT: 38
```

1.6.2.2 Nodos single-worker replicados



Figura 30: Spikes test para bbox2 con tres nodos single-worker replicados

```
Scenarios launched: 255
Scenarios completed: 217
Requests completed: 217
Mean response/sec: 1.46
Response time (msec):
  min: 903
  max: 9961
  median: 2105
  p95: 9307
  p99: 9792.8
Scenario counts:
  endpoint: 255 (100%)
Codes:
  200: 217
Errors:
  ETIMEDOUT: 38
```

1.6.2.3 Un nodo multi-worker



Figura 31: Spikes test para bbox2 con un nodo multi-worker

```

Scenarios launched: 255
Scenarios completed: 217
Requests completed: 217
Mean response/sec: 1.46
Response time (msec):
    min: 904
    max: 9970
    median: 2111
    p95: 9311
    p99: 9800.2
Scenario counts:
    endpoint: 255 (100%)
Codes:
    200: 217
Errors:
    ETIMEDOUT: 38
  
```

1.6.2.4 Nodos multi-worker replicados



Figura 32: Spikes test para bbox2 con tres nodos multi-worker replicados

```

Scenarios launched: 255
Scenarios completed: 217
Requests completed: 217
Mean response/sec: 1.46
Response time (msec):
    min: 904
    max: 9968
    median: 2107
    p95: 9311.7
    p99: 9798.8
Scenario counts:
    endpoint: 255 (100%)
Codes:
    200: 217
Errors:
    ETIMEDOUT: 38
  
```

1.6.2.5 Conclusión

Nuevamente, el comportamiento fue aproximadamente el mismo para todas las configuraciones de deployment, debido a que existe una única instancia del servicio bbox siendo consumido.

Se puede observar que el servicio funciona de forma correcta con el arribo de requests a tasa constante, hasta que llegan los picos de requests, momento en el cual se comienzan a registrar requests que retornan con error. El punto positivo a remarcar, es que una vez que el pico pasó, el servicio vuelve a estabilizarse, logrando bajar el response time al esperado (aproximadamente 1s).

El uso de recursos y el response time entonces, acompañan de forma prácticamente simétrica a los picos de requests recibidos, lo cual se consideró un punto positivo en cuanto a recuperabilidad del sistema.

1.6.3. Endurance test

Arribos a tasa constante de 2 rq/s, durante 3 minutos.

1.6.3.1 Un nodo single-worker



Figura 33: Endurance test para bbox2 con un nodo single-worker

```

Scenarios launched: 360
Scenarios completed: 360
Requests completed: 360
Mean response/sec: 1.8
Response time (msec):
    min: 903
    max: 924
    median: 906
    p95: 910
    p99: 911
Scenario counts:
    endpoint: 360 (100%)
Codes:
    200: 360
  
```

1.6.3.2 Nodos single-worker replicados



Figura 34: Endurance test para bbox2 con tres nodos single-worker replicados

```

Scenarios launched: 360
Scenarios completed: 360
Requests completed: 360
Mean response/sec: 1.8
Response time (msec):
    min: 904
    max: 924
    median: 906
    p95: 910
    p99: 911
Scenario counts:
    endpoint: 360 (100%)
Codes:
    200: 360
  
```

1.6.3.3 Un nodo multi-worker



Figura 35: Endurance test para bbox2 con un nodo multi-worker

```

Scenarios launched: 360
Scenarios completed: 360
Requests completed: 360
Mean response/sec: 1.8
Response time (msec):
    min: 904
    max: 924
    median: 906
    p95: 910
    p99: 913
Scenario counts:
    endpoint: 360 (100%)
Codes:
    200: 360
  
```

1.6.3.4 Nodos multi-worker replicados



Figura 36: Endurance test para bbox2 con tres nodos multi-worker replicados

```

Scenarios launched: 360
Scenarios completed: 360
Requests completed: 360
Mean response/sec: 1.8
Response time (msec):
    min: 904
    max: 1071
    median: 907
    p95: 912
    p99: 926.8
Scenario counts:
    endpoint: 360 (100%)
Codes:
    200: 360
  
```

1.6.3.5 Conclusión

El endpoint parece mantenerse estable ante una carga constante, manteniendo un response time promedio aproximado de 920ms. Se observa una alta tasa de completitud de requests (no hay requests que fallen), y un consumo prácticamente nulo de CPU.

Sin embargo, si bien el uso de la memoria siempre se mantiene por debajo del 0.3 %, se puede observar una leve pendiente positiva en el gráfico, lo que podría ser un indicio de que existe alguna ineficiencia en el manejo de la misma.

2. Sección 2

2.1. Objetivo

Esta sección del trabajo consiste en caracterizar los dos servicios de caja negra que fueron provistos por la cátedra para la realización del presente trabajo de análisis.

Esta caracterización se basa en distintas propiedades que se busca encontrar:

- **Sincrónico / Asincrónico:** Uno de los servicios se comportará de manera sincrónica, y el otro de manera asincrónica. Deberán detectar de qué tipo es cada uno.
- **Cantidad de workers (en el caso sincrónico):** Se sabe que el servicio sincrónico está implementado con una cantidad de workers. Se deberá buscar algún indicio sobre cuál es esta cantidad.
- **Demora en responder:** Se sabe que cada servicio demora un tiempo en responder, que puede ser igual o distinto entre ellos. El objetivo es conocer este valor para cada uno.

2.2. Determinar la naturaleza de cada servicio

En la primer parte, cuando se realizaron los tests de carga para cada servicio, se observó que el primero respondió de forma satisfactoria a la creciente cantidad de requests, logrando una alta disponibilidad, sin errores, y con un response time prácticamente constante. Se incluye nuevamente a continuación el gráfico correspondiente al primer test de carga realizado sobre este (con un nodo single-worker):



Figura 37: Load test para bbox1 con un nodo single-worker

Sin embargo, al repetir este test sobre el servicio bbox 2, se observó que rápidamente la tasa de completitud cae, hasta que el servidor deja de responder. Se repite el gráfico correspondiente a este comportamiento:



Figura 38: Load test para bbox2 con un nodo single-worker

Como se concluyó en la sección anterior, el primer servicio expone una naturaleza asincrónica (favoreciendo la **escalabilidad**), mientras que el segundo por el contrario es sincrónico y no escala, soportando un número de requests mucho menor.

2.3. Cantidad de workers del servicio sincrónico (bbox2)

Para determinar la cantidad de workers utilizados en la configuración de deploy de este servicio, se utilizó la herramienta [Apache Bench](#), que permite determinar la performance de un servidor que atiende requests HTTP mediante el envío concurrente de dicha clase de requests.

En primer lugar, se envió una única request y se midió el tiempo de respuesta:

```

Server Software:                               Apache/2.4.18 (Ubuntu)
Server Hostname:                             localhost
Server Port:                                9091

Document Path:                               /
Document Length:                            12 bytes

Concurrency Level:                          1
Time taken for tests:                      0.907 seconds
Complete requests:                           1
Failed requests:                            0
Total transferred:                         77 bytes
HTML transferred:                          12 bytes
Requests per second:                       1.10 [#/sec] (mean)
Time per request:                          907.316 [ms] (mean)
Time per request:                          907.316 [ms] (mean, across all concurrent requests)
Transfer rate:                             0.08 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
        (ms)

```

Waiting:	906	906	0.0	906	906
Connect:	0	0	0.0	0	0
Processing:	907	907	0.0	907	907
Total:	907	907	0.0	907	907

Luego, se enviaron dos requests de forma concurrente:

```

Server Software:
Server Hostname:      localhost
Server Port:          9091

Document Path:         /
Document Length:      12 bytes

Concurrency Level:    2
Time taken for tests: 1.815 seconds
Complete requests:    2
Failed requests:      0
Total transferred:    154 bytes
HTML transferred:     24 bytes
Requests per second:  1.10 [#/sec] (mean)
Time per request:     1814.744 [ms] (mean)
Time per request:     907.372 [ms] (mean, across all concurrent requests)
Transfer rate:        0.08 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:       0    0.0    0.0      0    0
Processing:   907  908    1.2    908    908
Waiting:      906  907    1.2    907    907
Total:        907  908    1.2    909    909

Percentage of the requests served within a certain time (ms)
  50%    909
  66%    909
  75%    909
  80%    909
  90%    909
  95%    909
  98%    909
  99%    909
100%    909 (longest request)

```

Se puede apreciar que 2 requests concurrentes son respondidas en el mismo tiempo que 1 sola request. Esto quiere decir que hay *al menos* 2 workers.

Se prueban 3 requests concurrentes

```

Server Software:
Server Hostname:      localhost
Server Port:          9091

Document Path:         /

```

```

Document Length:           12 bytes
Concurrency Level:        3
Time taken for tests:    1.812 seconds
Complete requests:        3
Failed requests:          0
Total transferred:        231 bytes
HTML transferred:         36 bytes
Requests per second:      1.66 [#/sec] (mean)
Time per request:         1811.815 [ms] (mean)
Time per request:         603.938 [ms] (mean, across all concurrent requests)
Transfer rate:            0.12 [Kbytes/sec] received

Connection Times (ms)
                  min  mean[+/-sd] median   max
Connect:          0    0.1      0.1     0       0
Processing:      906   906     0.9     906    907
Waiting:         904   905     0.7     905    906
Total:           906   906     0.8     907    907
WARNING: The median and mean for the total time are not within a normal deviation
These results are probably not that reliable.

Percentage of the requests served within a certain time (ms)
 50%    906
 66%    906
 75%    907
 80%    907
 90%    907
 95%    907
 98%    907
 99%    907
100%   907 (longest request)

```

El tiempo de respuesta es nuevamente el mismo. Se prueba con 4 requests concurrentes:

```

Server Software:
Server Hostname:        localhost
Server Port:             9091

Document Path:           /
Document Length:         12 bytes

Concurrency Level:       4
Time taken for tests:   2.716 seconds
Complete requests:       4
Failed requests:         0
Total transferred:       308 bytes
HTML transferred:        48 bytes
Requests per second:     1.47 [#/sec] (mean)
Time per request:        2715.711 [ms] (mean)
Time per request:        678.928 [ms] (mean, across all concurrent requests)
Transfer rate:            0.11 [Kbytes/sec] received

```

```

Connection Times (ms)
    min   mean[+/-sd] median   max
Connect:      0   0.0  0.0      0      0
Processing:  905 1133 451.5   909   1811
Waiting:     905 1132 451.4   907   1809
Total:       905 1133 451.6   909   1811

Percentage of the requests served within a certain time (ms)
 50%   909
 66%   909
 75% 1811
 80% 1811
 90% 1811
 95% 1811
 98% 1811
 99% 1811
100% 1811 (longest request)

```

Aquí se puede notar que el tiempo de respuesta se duplica. Es en este punto donde todas las requests ya no pueden ser respondidas de forma concurrente, por lo que es posible concluir que el servicio dispone de exactamente 3 workers.

2.4. Tiempos de demora en la respuesta

Para conocer estos tiempos de demora, basta con hacer una cantidad de requests concurrentes que no excedan la capacidad básica del servidor (en el caso del servicio sincrónico, dada por el número de workers, mientras que para el servicio asíncrono, el límite es lo suficientemente alto como para no preocuparnos).

Gracias al análisis anterior, se sabe que el servicio sincrónico cuenta con **3 workers**, por lo que se realizarán **3 requests concurrentes** a cada uno de los servicios para ver el tiempo medio de respuesta, lo que nos dice en nuestro caso el tiempo de demora.

Se incluyen a continuación ambas pruebas realizadas.

2.4.1. Servicio asíncrono

```

$ ab -n 3 http://localhost:9090/
[...]

Server Software:
Server Hostname:      localhost
Server Port:          9090

Document Path:         /
Document Length:      12 bytes

Concurrency Level:    1
Time taken for tests: 3.305 seconds
Complete requests:    3
Failed requests:      0
Total transferred:    231 bytes

```

```

HTML transferred:      36 bytes
Requests per second: 0.91 [#/sec] (mean)
Time per request:    1101.612 [ms] (mean)
Time per request:    1101.612 [ms] (mean, across all concurrent requests)
Transfer rate:       0.07 [Kbytes/sec] received

Connection Times (ms)
                  min   mean[+/-sd] median   max
Connect:          0     0.0    0.0      0
Processing:      1101  1101    0.7    1102    1102
Waiting:         1101  1101    0.7    1102    1102
Total:           1101  1102    0.7    1102    1102

Percentage of the requests served within a certain time (ms)
  50%   1102
  66%   1102
  75%   1102
  80%   1102
  90%   1102
  95%   1102
  98%   1102
  99%   1102
100%   1102 (longest request)

```

Se observa que el tiempo medio de respuesta coincide con el mínimo, ya que en todos los casos es de aproximadamente **1100ms**, que representa el tiempo de demora de este servicio.

2.4.2. Servicio sincrónico

```

$ ab -n 3 http://localhost:9091/
[...]
Server Software:
Server Hostname:      localhost
Server Port:          9091

Document Path:        /
Document Length:     12 bytes

Concurrency Level:   1
Time taken for tests: 2.759 seconds
Complete requests:   3
Failed requests:     0
Total transferred:   231 bytes
HTML transferred:    36 bytes
Requests per second: 1.09 [#/sec] (mean)
Time per request:    919.749 [ms] (mean)
Time per request:    919.749 [ms] (mean, across all concurrent requests)
Transfer rate:       0.08 [Kbytes/sec] received

Connection Times (ms)
                  min   mean[+/-sd] median   max
Connect:          0     0.1    0.1      0

```

Processing:	905	919	20.0	927	942
Waiting:	905	919	19.9	926	942
Total:	905	920	20.0	927	942

Percentage of the requests served within a certain time (ms)

50%	911
66%	911
75%	942
80%	942
90%	942
95%	942
98%	942
99%	942
100%	942 (longest request)

Se observa que el tiempo medio de respuesta es de 920ms, sin embargo, en este caso nos interesa el mínimo observado, ya que el resto es necesariamente mayor a este valor. El valor mínimo observado es de **911ms**, por lo que se concluye que esta puede ser una buena aproximación del tiempo de demora de este servicio.