Trabajo Práctico 1

Facultad de Ingeniería, Universidad de Buenos Aires
[75.73] Arquitectura de Software

Grupo “Ladrillo”

100029 del Mazo, Federico
102264 Hojman, Joaquin
97112 Kasman, Lucía
97131 Rombola, Juan Pablo
103409 Schmidt, Agustina

13 de Octubre de 2022

Trabajo Práctico 1

Servicios

En este trabajo vamos a comparar dos servicios distintos, con los mismos endpoints. Ambos servicios se acceden a través de nginx y ambos están implementados en Node.js; su unica diferencia es que el primer servicio tiene solo un proceso, mientras que el segundo esta replicado en múltiples contenedores (configurado en 5 réplicas).

Al levantar la aplicación, tenemos a nginx corriendo en `localhost:5555`. El primer servicio esta sobre `/`, y el segundo en `/many`. Los endpoints que proveen son:

- `/ping`: Un simple healtcheck. Devuelve un número identificador del proceso, para poder confirmar que el servicio replicado esta contestando desde distintos lugares.
- `/work?n=15`: Una manera de representar cálculos pesados. Computa y devuelve los primeros n-mil dígitos de pi, siendo `n` configurable.
- `/sync`: Invoca el servicio sincrónico `bbox` (explicado más adelante)
- `/async`: Invoca el servicio asincrónico `bbox`

```
$ make up # levantamos los múltiples contenedores con docker-compose
Creating network "7573-arqui_default" with the default driver
Creating 7573-arqui_grafana_1  ... done
Creating 7573-arqui_bbox_1     ... done
Creating 7573-arqui_node_1     ... done
Creating 7573-arqui_node_2     ... done
Creating 7573-arqui_node_3     ... done
Creating 7573-arqui_node_4     ... done
Creating 7573-arqui_node_5     ... done
Creating 7573-arqui_graphite_1 ... done
Creating 7573-arqui_nginx_1    ... done
Creating 7573-arqui_cadvisor_1 ... done

# Probemos ambos healthchecks
$ curl "localhost:5555/ping"
```

```
[28] pong
$ curl "localhost:5555/many/ping"
[99] pong

# Chequeemos que se contesta desde distintos procesos (por el número identificador)
$ for run in {1..7}; do curl localhost:5555/ping; done
[28] pong
[28] pong
[28] pong
[28] pong
[28] pong
[28] pong
[28] pong
$ for run in {1..7}; do curl localhost:5555/many/ping; done
[99] pong
[69] pong
[28] pong
[11] pong
[12] pong
[28] pong
[69] pong

# Testeemos todos los endpoints
$ curl "localhost:5555/work"
31415926...
$ curl "localhost:5555/sync"
Hello world!
$ curl "localhost:5555/async"
Hello world!

# Mostremos un poco cuanto trabaja el calculador de Pi
# mil digitos -> menos de un segundo
$ time curl "localhost:5555/work?n=1"
0,01s user 0,01s system 50% cpu 0,037 total
# 10 mil digitos -> menos de un segundo...
$ time curl "localhost:5555/work?n=10"
0,00s user 0,00s system 2% cpu 0,297 total
# 100 mil digitos -> veinte segundos!
$ time curl "localhost:5555/work?n=100"
0,00s user 0,00s system 0% cpu 20,632 total
```

Performance Testing

- https://www.softwaretestingclass.com/what-is-performance-testing/
- Explicar y mostrar ejemplos de artillery
- Pelar grafos de grafana y demas
- Vista componentes y conectos
- Deberán generar sus propias métricas desde la app Node para ser enviadas al daemon de statsd. Como mínimo, deberán generar una métrica con la demora en responder de cada endpoint (vista desde Node). Este métrica deberá ser visualizada en un gráfico adicional, que estará correlacionado con los demás gráficos en el tiempo.

Servicio `bbox`

Análisis y caracterización

Deberán caracterizar cada servicio mirando tres propiedades:

- Sincrónico / Asincrónico: Uno de los servicios se comportará de manera sincrónica, y el otro de mane
- Cantidad de workers (en el caso sincrónico): El servicio sincrónico está implementado con una cantid
- Demora en responder: Cada servicio demora un tiempo en responder, que puede ser igual o distinto ent

Las herramientas para este análisis son las mismas que usaron en la Sección 1. Deben generar carga que p

Caso de estudio: Sistema de inscripciones

Utilizando las herramientas y procedimientos de las secciones anteriores, deberán simular el comportamie

Nos concentraremos en simular la inscripción a una o más materias. Desde el punto de vista del usuario,

- Iniciar sesión
- Seleccionar una carrera
- Inscribirse (n veces)
  - Ver la lista de materias en las que está inscripto
  - Ver la lista de materias disponibles
  - Inscribirse en una materia
- Cerrar sesión

Para implementar este flujo, herramientas como Artillery (usando scenarios) o JMeter nos permiten simula

Deberán establecer ciertas hipótesis respecto de las dimensiones del problema. Por ejemplo, cantidad de

Con el escenario planteado, generar la carga, graficar puntos interesantes y luego analizar el comportam