Arquitectura de Software

(75.73/TB054) Curso Calonico

Trabajo Práctico 1

3 de octubre de 2024

Patricio Galvan
106166

Santiago Vaccarelli
106051

Matías Bacigalupo
103715

1. Introducción

El objetivo de éste trabajo práctico es comparar distintas estrategias y escenarios en un servicio HTTP, ver cómo impactan en los atributos de calidad y probar qué cambios se podrían hacer para mejorarlos.

Para ello se creó un servidor local con 4 servicios diferentes:

- **Ping** Un servicio para usar como healthcheck y como baseline para comparar con los demás servicios.
- **Dictionary** Un servicio que devuelve información de diccionario de una palabra en ingles.
- **Spaceflight News** Un servicio que devuelve los títulos de noticias sobre actividad espacial.
- **Useless Facts** Un servicio que devuelve un hecho al azar.

Estos servicios se sometieron a pruebas evaluando las siguientes tácticas:

1. **Caso base** Usado para tomar como referencia y verificar si hay mejoras con las tácticas aplicadas.
2. **Caché** Almacenar información recibida de la API y usarla al responder a un llamado.
3. **Replicación** Escalar el servicio a 3 copias.
4. **Rate Limiting** Limitar la frecuencia de consumo del servicio.

Para construir el servidor, simular tráfico, obtener datos del tráfico simulado en las pruebas, y graficar dichos datos, se usaron diversas tecnologías incluyendo Node.js, Docker, Docker Compose, Nginx, Redis, Artillery, cAdvisor, StatsD, Graphite y Grafana.

A continuación se muestran los resultados de las pruebas de estrategias para cada servicio.

2. Ping

Este servicio se usa como healthcheck y como baseline para comparar con los demás servicios.

2.1. Caso Base

Figura 1: Test de spike

2.2. Caché

Figura 2: Test de spike con caché

Como el tiempo de respuesta es ínfimo, la mejoría que trae almacenarla en caché es esencialmente nula.

2.3. Rate Limiting

Figura 3: Test de spike con rate limiting

Como los tiempos de respuestas son ínifimos, tampoco se nota una diferencia limitando la frecuencia de requests aceptadas.

3. Dictionary

Este servicio devuelve información de diccionario de una palabra en ingles, consultando a la Free Dictionary API.

3.1. Caso Base

Figura 4: Test de escalabilidad

3.2. Caché

Figura 5: Test de escalabilidad con caché

Se almacena en caché cada palabra recibida como request. Por lo tanto, no hay un impacto notable en el tiempo de respuesta si la cantidad de requests no es muy alta, ya que solo se utiliza la caché cuando se recibe una palabra previamente recibida.

3.3. Rate Limiting

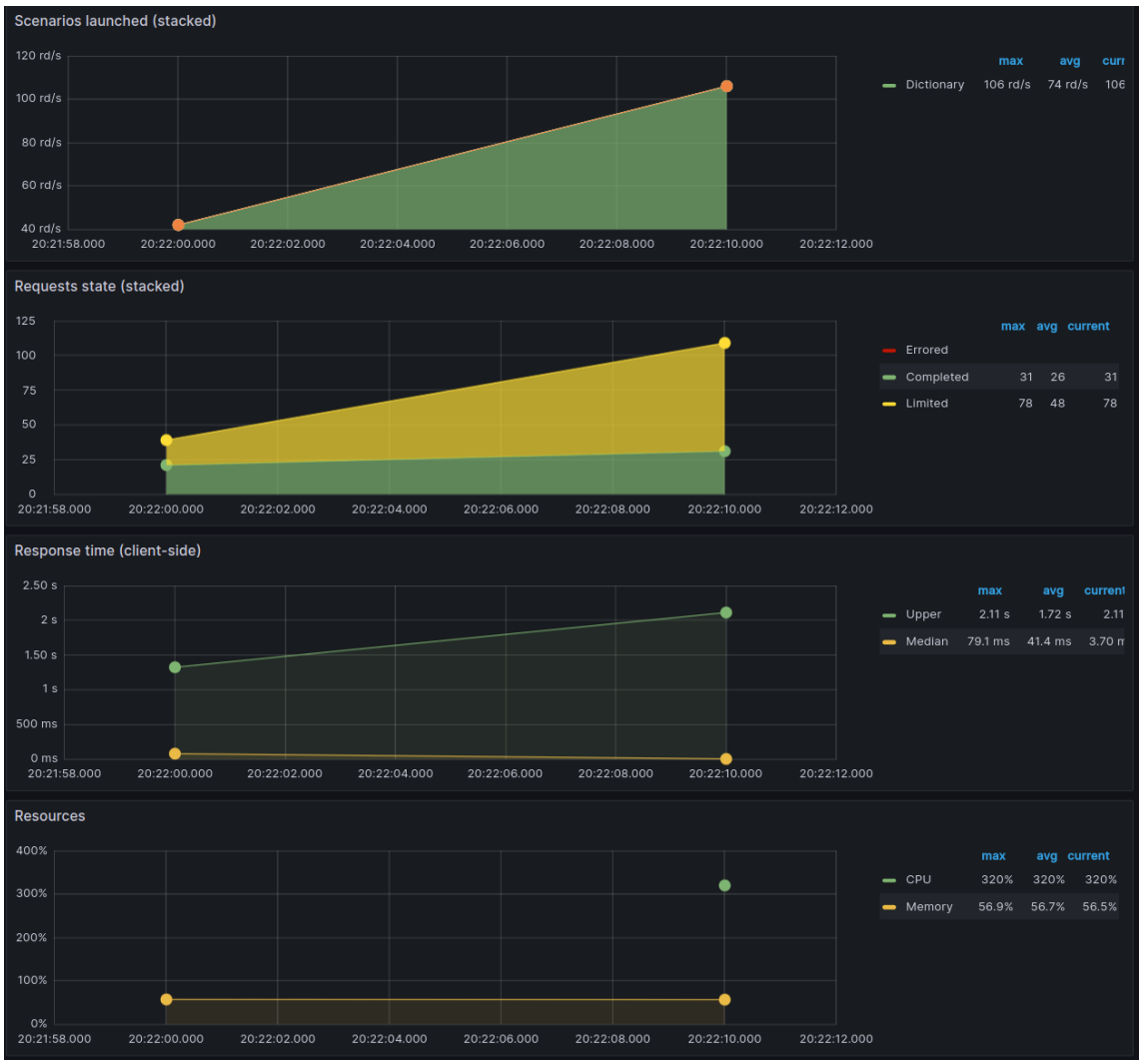

Figura 6: Enter Caption

Limitar la cantidad de requests aceptadas a 3 por segundo mejoró el tiempo de respuesta. Pasó de un promedio de 283ms a 41.4ms con rate limiting.

4. Spaceflight News

Este servicio devuelve los títulos de las 5 últimas noticias sobre actividad espacial, obtenidas desde la Spaceflight News API.

4.1. Caso Base

Figura 7: Test de carga

4.2. Caché

Como la respuesta es la misma para todos los usuarios mientras no haya nuevas noticias, se decidió almacenar la respuesta en el momento en que el primer cliente hace un request (lazy population) y usarla para todos los usuarios siguientes por 1 hora.

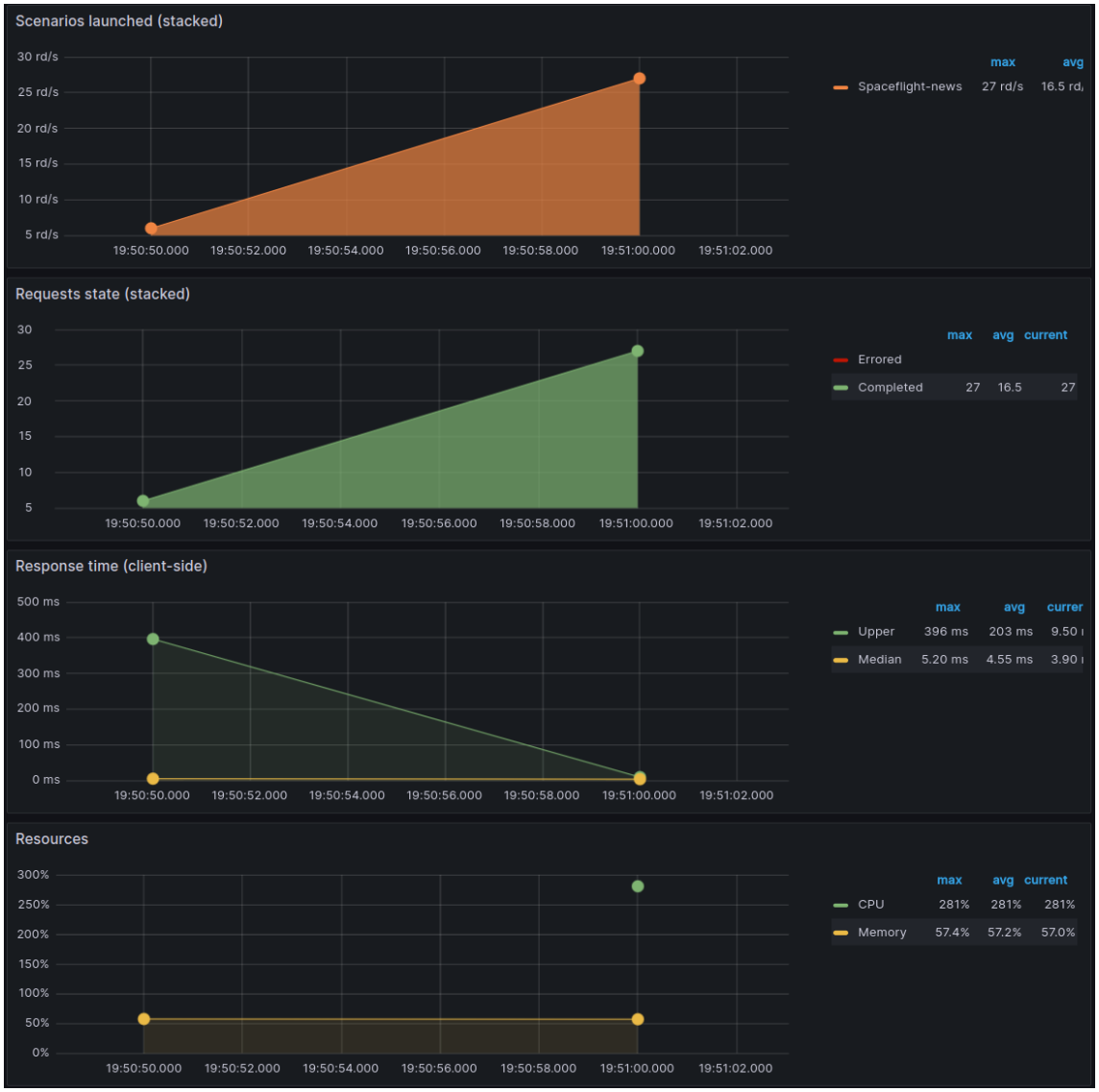

Figura 8: Test de carga con caché

Aquí se nota la diferencia en el response time almacenando la respuesta en el caché. La consecuencia es que el response time promedio es mucho más bajo, pasó de 534ms a 4.55ms con caché.

4.3. Rate Limiting

Figura 9: Test de carga con rate limiting

Limitar el rate de respuestas aceptadas afectó levemente el tiempo de respuesta de las llamadas que más tardaron. En promedio, las respuestas que más tardaron pasaron de 1.43s a 1.12s con rate limiting.

5. Random Useless Facts

Este servicio devuelve un hecho irrelevante al azar, tomado de Useless Facts.

5.1. Caso Base

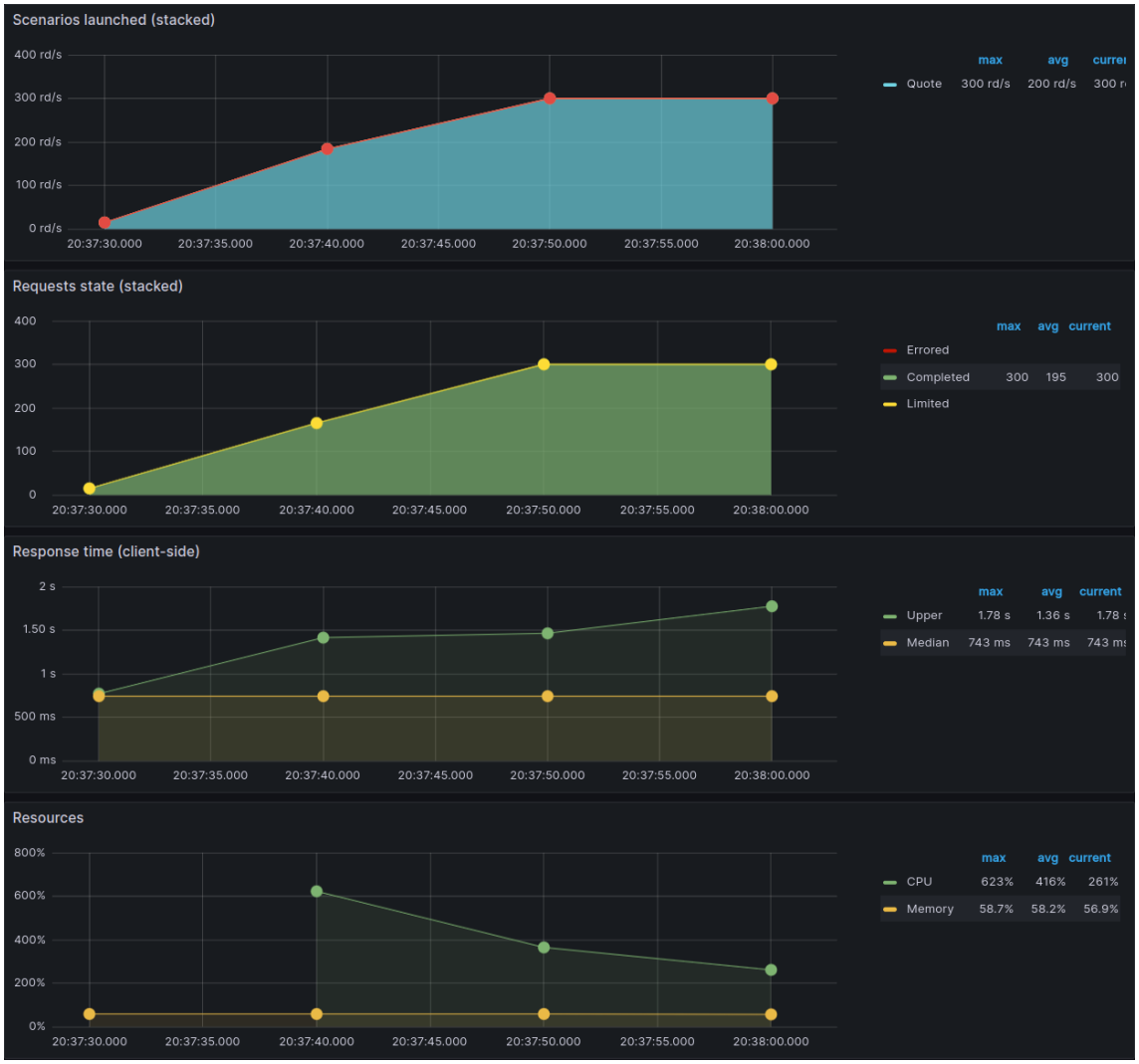

Figura 10: Test de spike

5.2. Caché

No tiene sentido cachear una respuesta aleatoria que cambia con cada solicitud.

5.3. Replicación

Para la replicación se usaron 3 copias del servicio en simultáneo.

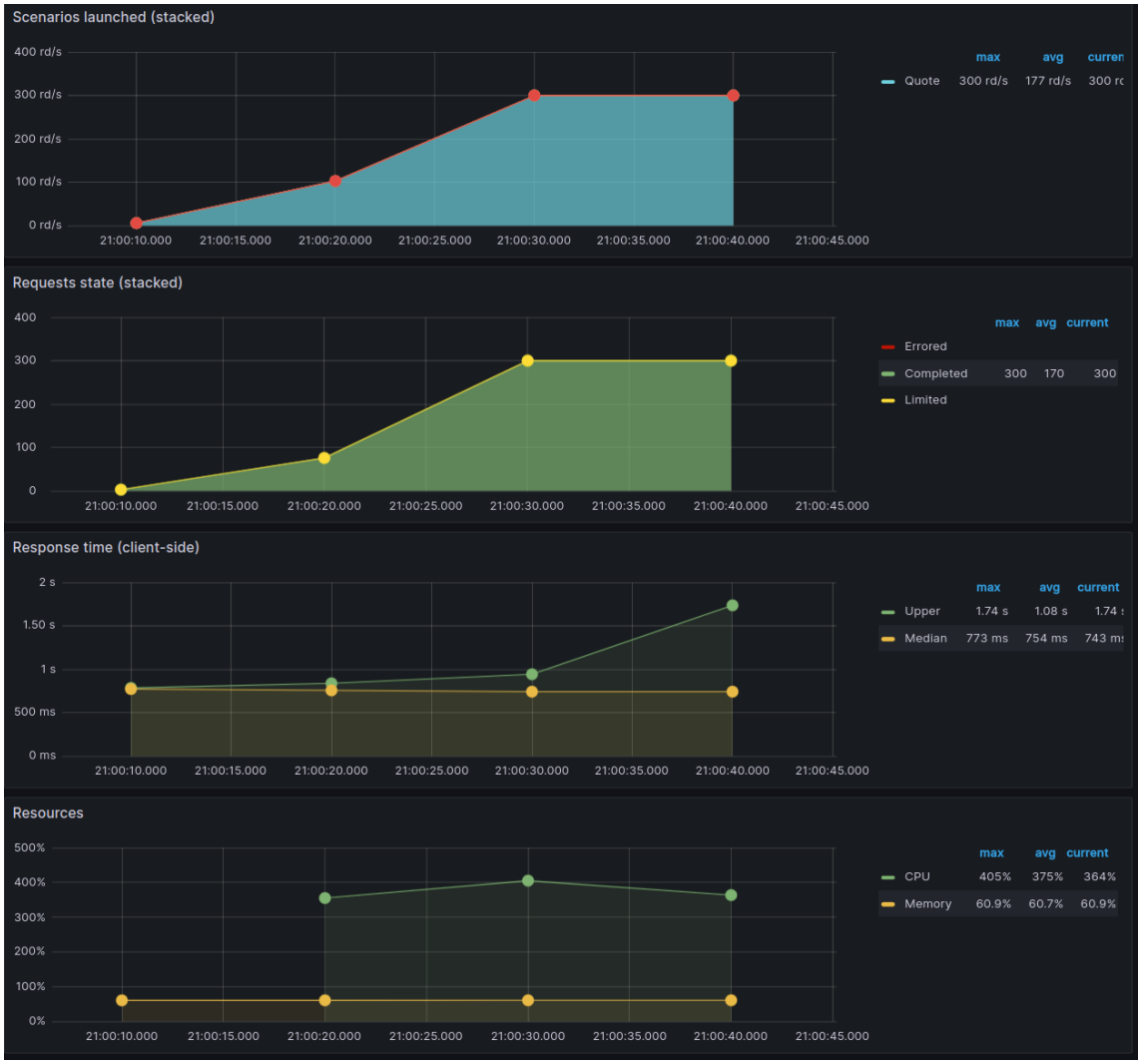

Figura 11: Test de spike con balanceo

Replicar el servicio causó que disminuyera el upper response time promedio ya que la carga se distribuye entre las 3 copias del servicio. Pasó de 1.36s a 1.08s.

5.4. Rate Limiting

Se limitó el rate de respuestas aceptadas a 3 por segundo.

Figura 12: Test de spike con rate limiting

Limitar las respuestas aceptadas disminuyó el tiempo de respuesta. El promedio bajó de 743ms a 190ms.

Figura 13: Test de spike con rate limiting con balanceo

Limitando las respuestas aceptadas por segundo, al replicar el servicio a 3 copias aumentó la cantidad de requests aceptadas, ya que hay más copias del servicio para aceptar las requests. Pasaron de un promedio de 24.8 requests aceptadas a 40.3.

6. Conclusiones

Analizando los gráficos de cada servicio se concluye lo siguiente sobre las estrategias testeadas:

La estrategia de caché aumenta la performance cuando se trata de una respuesta que no cambia muy frecuentemente, por lo que es útil para el servicio de Spaceflight News. No vale la pena aplicarla cuando los datos varían mucho en cada solicitud.

La estrategia de rate limiting mejoró el tiempo de respuesta en los casos testeados, pero se debería usar con criterio ya que puede empeorar la experiencia de usuario.

La estrategia de replicación disminuye el tiempo de respuesta y el consumo de recursos (por cada servidor). Además, resultó muy exitoso usarla en conjunto con rate limiting para reducir el impacto de esa estrategia.