Universidad de Buenos Aires

Facultad de Ingeniería

Arquitectura de Software

75.73

TP - Mediciones sobre WebServers

2° Cuatrimestre 2017

<u>Alumno:</u> Santiago Alcalá

Introducción

En el trabajo práctico se propuso realizar mediciones sobre distintos escenarios y web servers diferentes.

Para realizar esta actividad, en un primer momento se planteó la utilización de **ApacheBench** para realizar requests http y una herramienta online de gráficos,al no dar frutos esta forma, se optó por la utilización de **Artillery**. Además, los distintos servidores fueron montados sobre **Docker**, con imágenes hechas de forma propia.

Las herramientas utilizadas para la recolección y muestra de los datos, también fueron montadas sobre la misma plataforma.

Para la recolección se utilizó **StatsD**, una herramienta que permite recibir por un determinado puerto los datos enviados por Artillery, luego está **Graphite**, que actúa de base de datos para estos datos enviados, y también sirve como display de los mismos(aunque es un poco menos intuitivo), como última herramienta se usó **Grafana**, la cual es un dashboard configurable, que puede recolectar datos de distintas bases y mostrarlas, según los criterios que se necesiten.

Web servers

En cuanto a los WebServers utilizados, en este caso tenemos tres variantes, dos en el lenguaje Python y una en JavaScript.

Los del primer grupo son **Flask** y **Gunicorn**, por el otro lado tenemos a **Node**

Con respecto a lo observado en las diferentes métricas logradas, fue que Flask es un servidor mucho menos potente que los otros dos casos. Esto se debió a que las pruebas arrojaron que al punto máximo de exigencia de nuestras mediciones, este arrojo tener muchos más casos de timeouts con respecto a los otros dos, en especial con Node, el mas "potente" de los tres.

En cuanto al tiempo de respuesta, se repite la misma situación que antes, Node mantiene un tiempo de respuesta en el punto crítico más "alto" medido, de una media de 10k ms, contra alrededor de 75k ms de Gunicorn y Flask.

Escenarios planteados

Los escenarios planteados desde la cátedra fueron los siguientes:

- Endpoint liviano(root): solo devuelve un simple valor
- Endpoint lento: Ciclo de un millón de vueltas

- Endpoint con una imagen liviana(~350 KB)
- Endpoint con una imagen pesada(~4 MB)

Aclaraciones

Al realizar las mediciones, no tuvimos la oportunidad de realizar mediciones de CPU y Memoria por el momento, mismo de utilizar Nginx.

En cuanto a los gráficos, hay un error en los gráficos, ya que al empezar y terminar la medición, se muestran los últimos valores medidos y luego, al recibir nuevos, se actualizan, se podría decir que los gráficos continúan en una constante con el último valor medido, hasta recibir uno nuevo. Esto se debe a un fallo en la configuración de StatsD.

Desarrollo

Para los diferentes escenarios, se utilizó un archivo de configuración de Artillery que define diferentes etapas, con diferentes intensidades de peticiones(enviando más requests por segundo, más usuarios en paralelo, etc). Para todas las situaciones, se utilizó la misma configuración.

A continuación podremos presentar los diferentes escenarios con sus respectivas mediciones tomadas.

Endpoint liviano:

Gunicorn

Node

Flask

Análisis:

Con respecto a este endpoint, no se notaron prácticamente diferencias entre los distintos servidores, creemos que es debido a que al ser una request tan básica, la única diferencia que encontramos fue en "picos" con respecto al tiempo de respuesta, sobre todo en el caso de Flask.

Endpoint lento:

Gunicorn

Node

Flask

Análisis

En este caso, al hacer un llamado a un endpoint "lento", el cual realiza un ciclo de un millón de vueltas en todos los casos, se nota una gran diferencia de performance entre Flask con respecto a los otros dos. Esto se ve reflejado no solo el la muy grande diferencia en tiempos de respuesta, sino también en la cantidad de Pending Request que llega a tener en un punto Flask.

En el caso de la diferencia entre Node y Gunicorn, la diferencia en este escenario no es tan notoria, el tiempo de respuesta media de Node es más alto que el de Gunicorn (prácticamente es la mitad), son valores que no modificarían la "usabilidad" de un endpoint de este estilo.

Endpoint con imagen liviana:

Gunicorn

Node

Flask

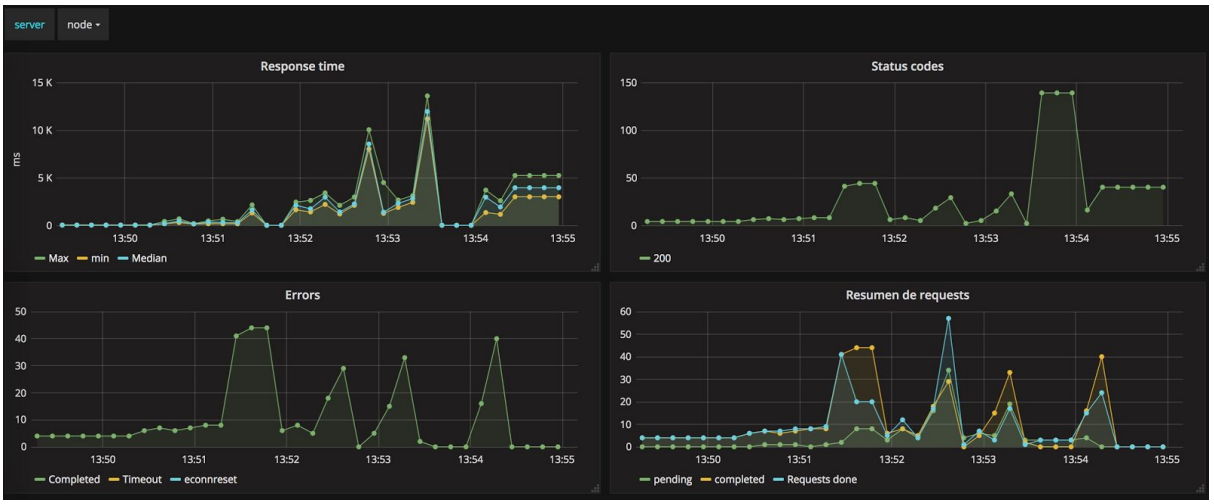
Análisis

Con respecto al endpoint donde se devuelve una imagen liviana, de alrededor de 300 KB, se sigue manteniendo la tendencia notada en los escenarios anteriores, la delantera la sigue llevando Node, si bien se pueden ver algunos picos en los tiempo de respuesta(sobre todo en la de tiempo máximo y la media), por un poco se podría decir que adelanta en velocidad a Gnuicorn.
En el caso de Flask, si bien el tiempo de respuesta fue relativamente aceptable (comparando con respecto a los otros dos), tuvo problemas con respecto a mantener request en estado de pending, cosa que en los otros dos servidores no ocurrió, por eso se podría decir que lo dejaríamos en último lugar.

Endpoint con imagen pesada:

Gunicorn

Node



Flask

Análisis:

Por último, y quizás el escenario donde las diferencias fueron más notorias, es en el caso donde se hace una request para pedir una imagen de alrededor de 4 MB.

Si bien la tendencia marcada en los anteriores puntos se sigue manteniendo, es aquí donde más se ve acentuada la misma.

Por primera vez podemos ver que empiezan a aparecer errores de Timeout y de refuse de conexiones, ambos en los dos webservers que usan Python. Siguiendo con lo visto los anteriores puntos, el servidor de Flask llega a un punto donde los tiempos de respuesta aumentan considerablemente, que también prácticamente coincide con los picos de errores internos, a su vez, antes de llegar a esos picos se puede ver como también acumula importante cantidad de request en estado de pending.

Con respecto a Gunicorn, el comportamiento es relativamente similar, pero difiere en los valores que arroja. También arroja timeouts y negativas a la conexión con el servidor, pero los valores son casi la mitad de veces, y el número de requests en estado de pendiente también es elevado, pero sigue siendo alrededor de 25% menos que en el caso

de Flask. En cuanto a los tiempos de respuesta, se asemejan bastante.

Por último, es aquí donde podemos ver que el Node lleva la ventaja sobre sus competidores, los tiempos de respuesta son considerablemente menores al igual que su acumulación de requests en estado de Pending, pero un dato bastante importante es que no hubo detección de errores, ni por timeout ni por negativa de conexión con el servidor.

Conclusión

Para redondear las ideas surgidas al realizar este trabajo práctico, podríamos decir que en todos los casos, las mediciones que hicimos, Node obtuvo el mejor resultado de todas, seguido en la mayoría de los casos de cerca por Gunicorn(salvo en el último), y por tercer lugar, en varios casos relegado por bastante, Flask.

¿Esto significa que uno sea mejor que otro? Nosotros podríamos decir que no. Simplemente que en estos escenarios planteados, la delantera siempre la llevo Node, luego habría siempre que hacer un análisis del contexto donde estos diferentes tipos de WebServers van a ser utilizados, quizas en un ambiente de desarrollo, donde no se necesita mover grandes volúmenes de datos ni una concurrencia importante, Flask sea una herramienta viable. Luego si lo que uno necesita es montar un servidor en producción con cientos o miles de usuarios, realizando intercambio de datos, quizás sea más prudente la utilización de Node o Gunicorn, dependiendo de las habilidades o capacidades que los desarrolladores posean, o quizás al tipo de aplicación que uno desea montar.