Comparación de diferentes implementaciones de servidores http

Autor: Erik Avagyan

Introducción

Se han realizado diferentes implementaciones de un servidor http simple que muestra una página simple en http://127.0.0.1:8080. Lista de todas las versiones:

- Swift, usando el paquete NIO
- Java, usando la clase com.sun.net.httpserver.HttpServer
- C
- PHP
- Node.js
- Rust
- Python
    - usando el framework Flask
    - usando el paquete http.server

El código fuente de todos los servidores y otros ficheros se puede ver en el siguiente repositorio: https://github.com/ea57-ua/http_tests.git.

Medidas de tiempo y memoria

Se han tomado dos medidas, el tiempo que tarda en responder la solicitud y la memoria usada por el servidor para responder una solicitud. Para medir el tiempo se ha usado el comando "time" de Linux acompañado de "curl" para hacer la petición: **time curl http://127.0.0.1:8080**. Este comando devuelve 3 valores: el tiempo real para completar la solicitud, el tiempo de CPU en el modo de usuario y el tiempo de CPU en el modo núcleo. Nos fijaremos más en el primer valor.

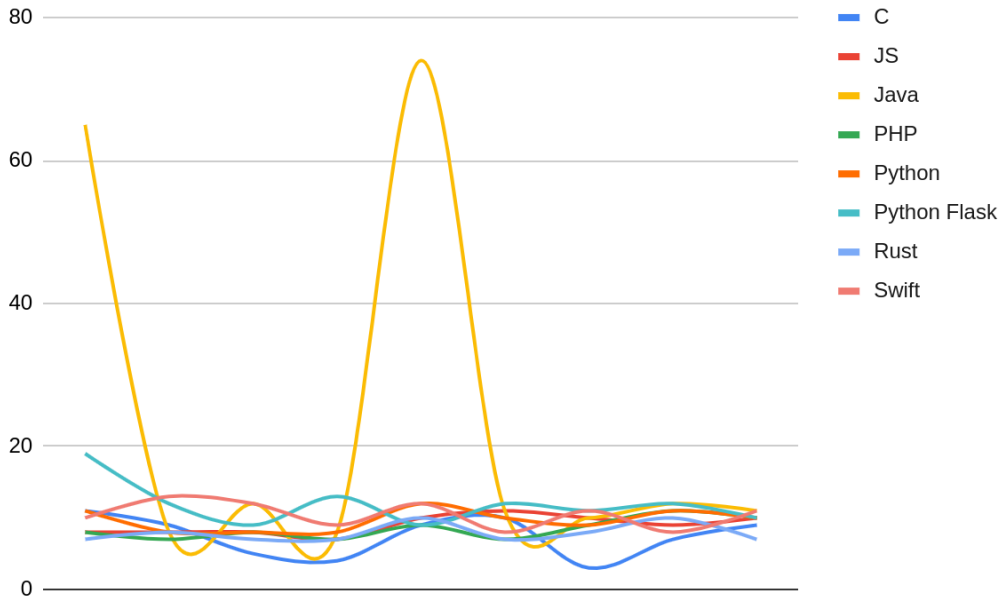
Para medir la memoria utilizada se ha usado el comando "top" de linux. Las peticiones se hacían desde el navegador Firefox y en una terminal se filtraban los datos sobre firefox por top: "**top | grep firefox**". Los resultados son aproximados porque el navegador tiene más procesos en activo pero en rasgos generales muestran diferencias entre las diferentes versiones.

Comparación de tiempos

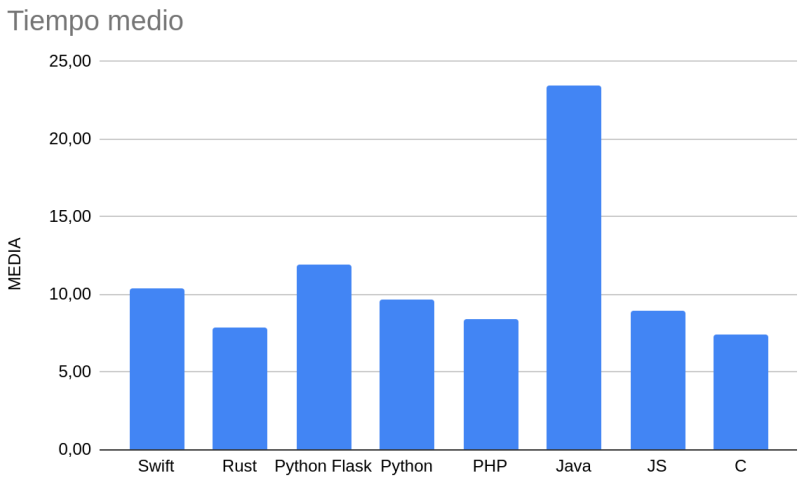
Se han introducido los tiempos medidos en la siguiente tabla. En la última fila vemos la media de tiempo por cada implementación.

	C	JS	Java	PHP	Python	Python Flask	Rust	Swift
	11	8	65	8	11	19	7	10
	9	8	8	7	8	12	8	13
	5	8	12	8	8	9	7	12
	4	7	8	7	8	13	7	9
	9	10	74	9	12	9	10	12
	10	11	11	7	10	12	7	8
	3	10	10	9	9	11	8	11
	7	9	12	11	11	12	10	8
	9	10	11	10	10	10	7	11
MEDIA	7,44	9,00	23,44	8,44	9,67	11,89	7,89	10,44

En la siguiente gráfica vemos una comparación de los datos. La implementación que más se destaca es en Java siendo la más lenta. La siguiente en lentitud sería el framework Flask de Python. Los mejores resultados son de las implementaciones en C y Rust.



Por último, vemos en la siguiente gráfica de columnas los tiempos medios que da otro punto de vista de lo mismo.



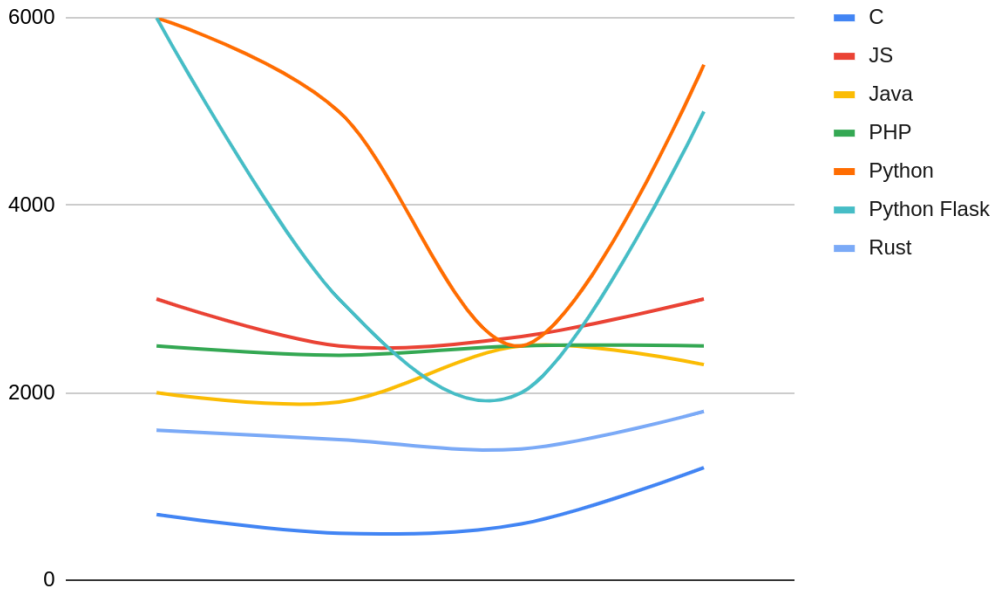
Comparación de memoria

Los siguientes resultados no son muy precisos ya que se han tomado viendo la memoria en kilobytes que usa Firefox y quedándose con la diferencia de antes y después de hacer la solicitud al servidor http. Algunos valores son aproximados.

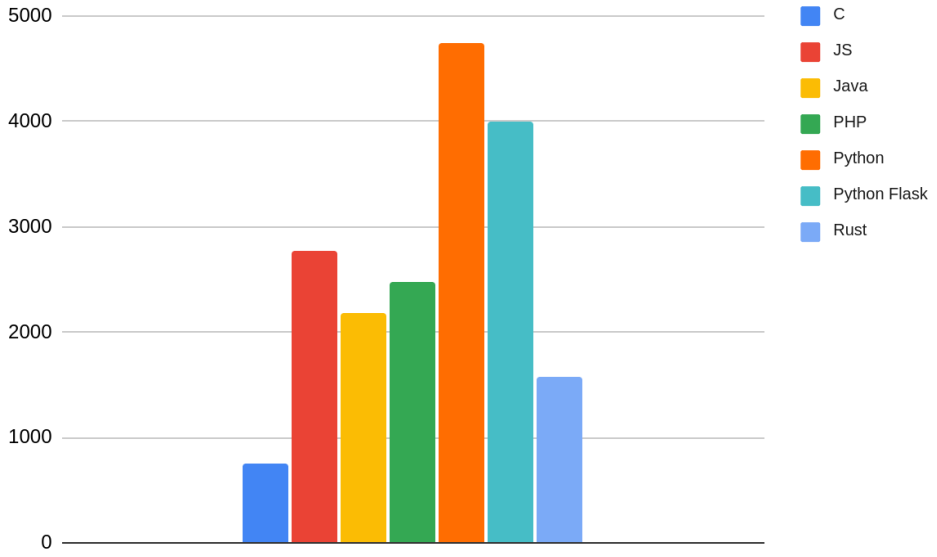
	C	JS	Java	PHP	Python	Python Flask	Rust	Swift
	700	3000	2000	2500	6000	6000	1600	
	500	2500	1900	2400	5000	3000	1500	
	600	2600	2500	2500	2500	2000	1400	
	1200	3000	2300	2500	5500	5000	1800	
MEDIA	750	2775	2175	2475	4750	4000	1575	

Nota: La columna de Swift está vacía porque el código ha empezado a dar problemas y está en proceso de revisión.

En la tabla anterior vemos el número de kilobytes aproximados que ha usado cada servidor en responder una solicitud hecha desde Firefox. En la siguiente gráfica vemos más clara la comparación. Como era esperable la implementación en C gasta muy poca memoria, la versión en Rust también es muy eficiente en cuanto a la RAM usada. Las implementaciones que más consumen son en Python.



Podemos comparar las medias aritméticas de las medidas en la siguiente gráfica de columnas. La implementación en Python con el paquete http.server usa 6 veces más memoria que la versión en C.


5000
4000
3000
2000
1000
0
C
JS
Java
PHP
Python
Python Flask
Rust