DP2 2021-2022

Progress Report

Acme Toolkits

Repositorio:

https://github.com/fracaralb/Acme-Toolkits

Miembros:

- Caro Albarrán, Francisco Andrés (fracaralb@alum.us.es)
- Gallego Huerta, Alberto(albgalhue@alum.us.es)
- Martín Luque, José Manuel (josmarluq@alum.us.es)
- Reyes Madrid, Francisco (frareymad@alum.us.es)
- Sillero Manchón, Jorge (jorsilman@alum.us.es)

GRUPO E3.06

23 de mayo de 2022

Índice

Resumen ejecutivo

El informe de rendimiento a realizar es un documento que agrupa y analiza las métricas correspondientes a este atributo con respecto a nuestro proyecto Acme-Toolkits. Esto es, dos análisis realizados en distintos ordenadores sobre el intervalo de confianza y un contraste de hipótesis que compara los resultados obtenidos en ambos dispositivos

Tabla de revisiones

FECHA	VERSIÓN	DESCRIPCIÓN
2022-05-23	1	Versión inicial

Introducción

Los requisitos estudiados en este informe son requisitos no funcionales, más concretamente centrados en el tiempo que tarda un ordenador en ejecutar una petición o mostrar una vista.

En nuestro caso, se llevará a cabo el análisis del tiempo real medio usado por cada uno de los dos ordenadores para llevar a cabo las tareas mencionadas anteriormente. Se debe declarar cuál de los dispositivos es más eficiente a un nivel de confianza del 95%.

Primero se explicarán y se realizarán todas las pruebas correspondientes al primer ordenador. Más tarde, seguiremos el mismo procedimiento con el segundo, para por último realizar una hipótesis de contraste de los resultados obtenidos para ambos dispositivos

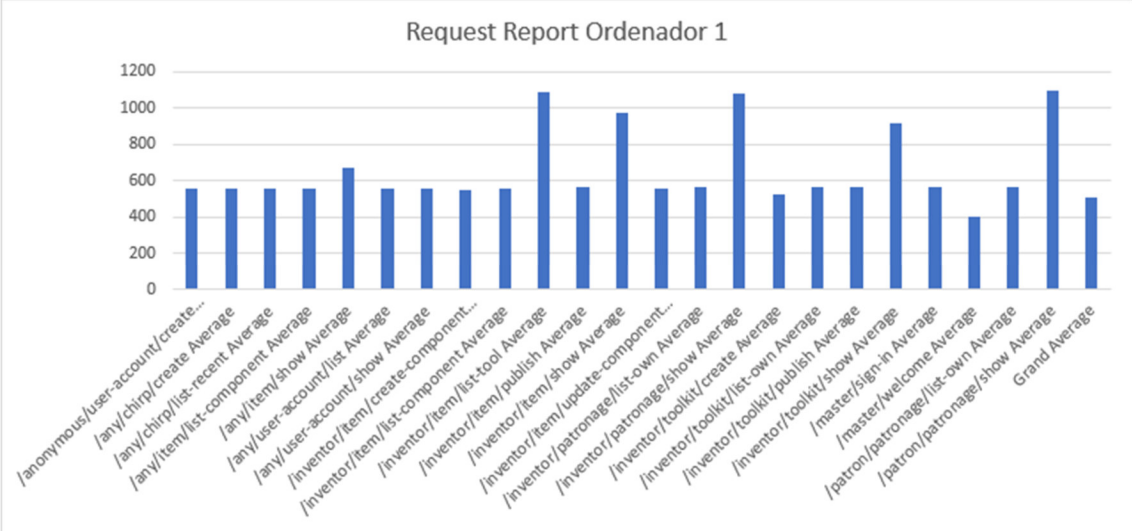
Recopilación de datos

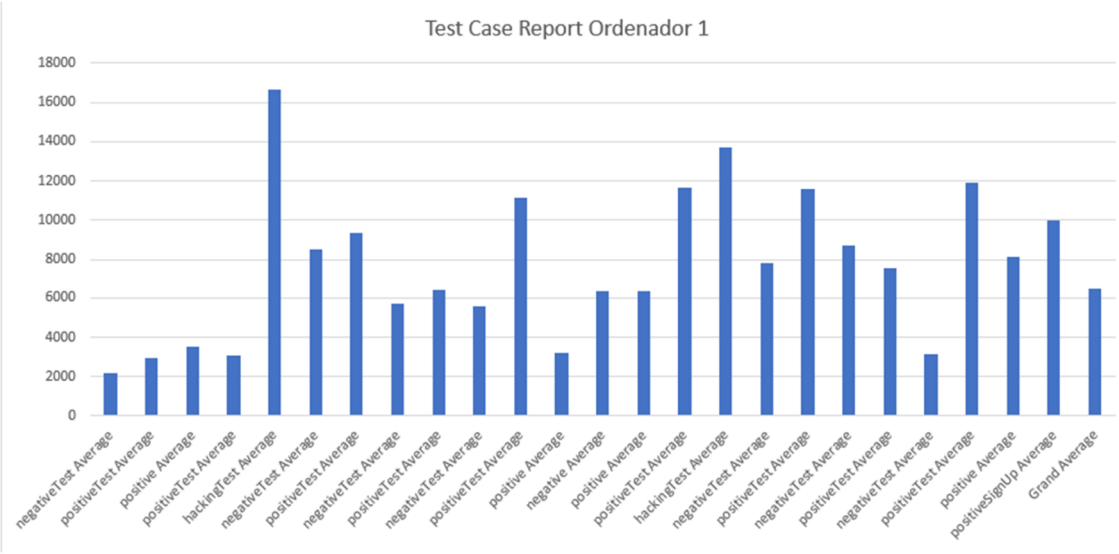
Los datos usados para medir dicho rendimiento gracias a que al ejecutar los tests de Acme-Toolkits, el framework genera automáticamente unos archivos con extensión .csv, los cuáles abrimos con Microsoft Excel. Una vez aquí, podemos aplicarle funciones a los datos para que nos ofrezcan las métricas

Análisis de datos

Ordenador 1

Dicho dispositivo cuenta con un procesador I7-4770, una RAM de 16 GB y Windows 10 como sistema operativo. Tras la ejecución de los tests, se ha generado la siguiente gráfica de tiempos promedio.





Como podemos observar en dicha gráfica, todos los tests tardan en ejecutarse entre 2 y 17 segundos

time		
Mean	511.9620731	
Standard Error	9.014198522	
Median	555	
Mode	566	
Standard Deviation	256.3901753	
Sample Variance	65735.92197	
Kurtosis	102.2026482	
Skewness	6.723353507	
Range	4616	
Minimum	233	
Maximum	4849	
Sum	414177.3171	
Count	809	
Confidence Level(95.0%)	17.69400897	
Confidence intervale	494.2680641	529.6561

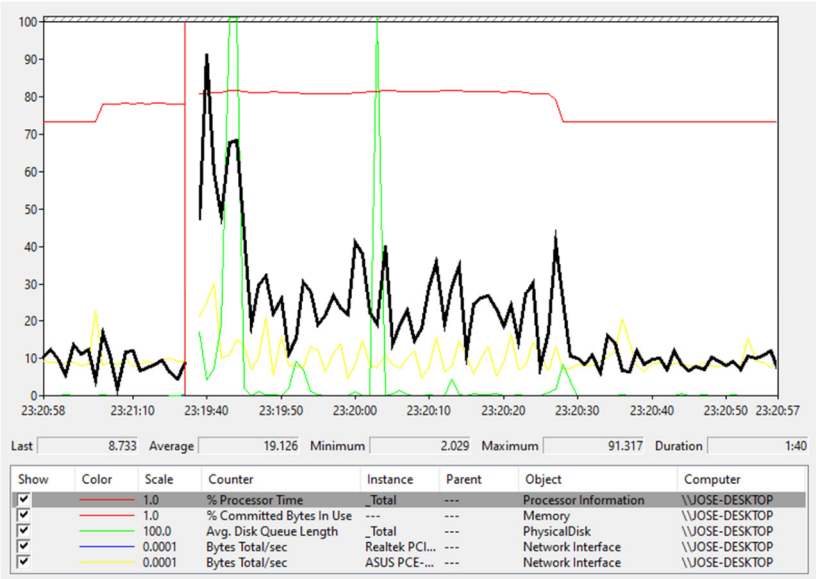
En esta tabla podemos observar que hemos obtenido: el nivel de confianza ha sido del 17.69 milisegundos mientras que el intervalo de confianza oscila entre 497.27 y 529.65. Ambos son menores que los 1000 ms requeridos. Se obtiene por tanto un resultado positivo

En la siguiente imagen se muestra el profiling del proyecto obtenido en el dispositivo 1 tras ejecutar todos los tests

Name	Total Time		Total Time (CPU)	
main	816,105 ms	(100%)	526,449 ms	(100%)
Reference Handler	816,105 ms	(100%)	0.0 ms	(-%)
Finalizer	816,105 ms	(100%)	0.0 ms	(-%)
RMI TCP Accept-0	816,105 ms	(100%)	0.0 ms	(-%)
RMI TCP Connection(1)-192.168.1.15	816,105 ms	(100%)	816,105 ms	(100%)
RMI Scheduler(0)	816,105 ms	(100%)	0.0 ms	(-%)
JMX server connection timeout 19	816,105 ms	(100%)	0.0 ms	(-%)
mysql-cj-abandoned-connection-cleanup	816,105 ms	(100%)	0.0 ms	(-%)
RMI TCP Connection(2)-192.168.1.15	816,105 ms	(100%)	816,105 ms	(100%)
Exec Default Executor	816,105 ms	(100%)	0.0 ms	(-%)
Exec Stream Pumper	816,105 ms	(100%)	816,105 ms	(100%)
Exec Stream Pumper	816,105 ms	(100%)	816,105 ms	(100%)
OkHttp ConnectionPool	816,105 ms	(100%)	0.0 ms	(-%)
Okio Watchdog	816,105 ms	(100%)	0.0 ms	(-%)
HikariPool-1 housekeeper	814,801 ms	(100%)	0.0 ms	(-%)
Catalina-utility-1	814,726 ms	(100%)	186 ms	(100%)
Catalina-utility-2	814,726 ms	(100%)	171 ms	(100%)

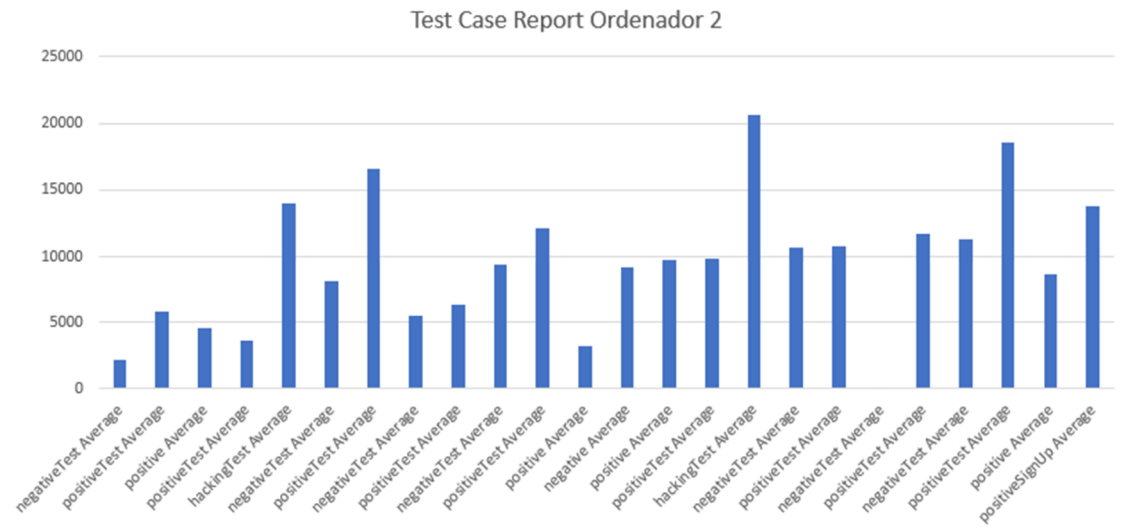
Name	Self Time (CPU)		Total Time (CPU)	
java.net.SocketInputStream.**socketRead0[native]** ()	2,156,630 ms	(44.8%)	2,156,630 ms	(3.1%)
java.io.FileInputStream.**readBytes[native]** ()	1,634,595 ms	(33.9%)	1,634,595 ms	(2.3%)
sun.management.ThreadImpl.**dumpThreads0[native]** ()	816,105 ms	(16.9%)	816,105 ms	(1.2%)
java.io.WinNTFileSystem.**getBooleanAttributes[native]** ()	101,085 ms	(2.1%)	101,085 ms	(0.1%)
java.lang.Throwable.**fillInStackTrace[native]** ()	27,983 ms	(0.6%)	27,983 ms	(0%)
java.util.zip.ZipFile.**getEntry[native]** ()	25,752 ms	(0.5%)	25,752 ms	(0%)
java.lang.Class.**forName0[native]** ()	6,921 ms	(0.1%)	154,314 ms	(0.2%)
java.security.AccessController.**doPrivileged[native]** ()	4,803 ms	(0.1%)	1,750,284 ms	(2.5%)
java.io.FileInputStream.**open0[native]** ()	4,565 ms	(0.1%)	4,565 ms	(0%)
java.net.DualStackPlainSocketImpl.**connect0[native]** ()	4,357 ms	(0.1%)	4,357 ms	(0%)
java.lang.ClassLoader.**findBootstrapClass[native]** ()	2,440 ms	(0.1%)	2,440 ms	(0%)
java.net.SocketOutputStream.**socketWrite0[native]** ()	2,373 ms	(0%)	2,373 ms	(0%)
java.lang.ClassLoader.**findLoadedClass0[native]** ()	2,099 ms	(0%)	2,099 ms	(0%)
java.io.WinNTFileSystem.**canonicalize0[native]** ()	1,744 ms	(0%)	1,744 ms	(0%)
sun.nio.ch.WindowsSelectorImpl.**setWakeupSocket0[native]** ()	1,605 ms	(0%)	1,605 ms	(0%)
java.lang.Throwable.**getStackTraceElement[native]** ()	1,465 ms	(0%)	1,465 ms	(0%)
sun.nio.ch.SocketDispatcher.**write0[native]** ()	1,427 ms	(0%)	1,427 ms	(0%)
java.lang.Class.**isAssignableFrom[native]** ()	876 ms	(0%)	876 ms	(0%)

En esta gráfica observamos que el consumo se ha mantenido medio durante toda la ejecución de los tests del proyecto sin ningún pico alto. (Los picos que se observan, son posteriores a la finalización de la ejecución de los tests). Obtenemos por tanto un resultado también positivo



Ordenador 2

Este segundo dispositivo dispone de un procesador I5-6265U y 8 GB de RAM, usando Windows 10 como sistema operativo. Tras la ejecución de los tests, se ha generado la siguiente gráfica de tiempos promedio.

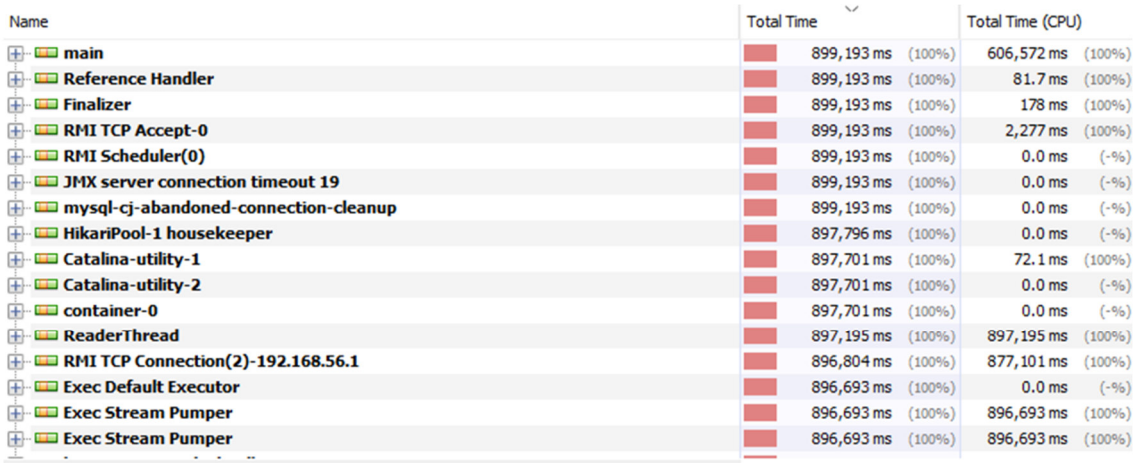


En este segundo dispositivo, observamos que los tests tardan en ejecutarse entre 2 y 21 segundos

En esta tabla, observamos que el nivel de confianza obtenido es de 67.88 ms mientras que el intervalo de confianza oscila entre 430.12 y 565.88 ambos menores que los 1000 ms requeridos. Se ha obtenido por tanto un resultado positivo

Se ha realizado también el profiling del proyecto en busca de posibles cuellos de botella

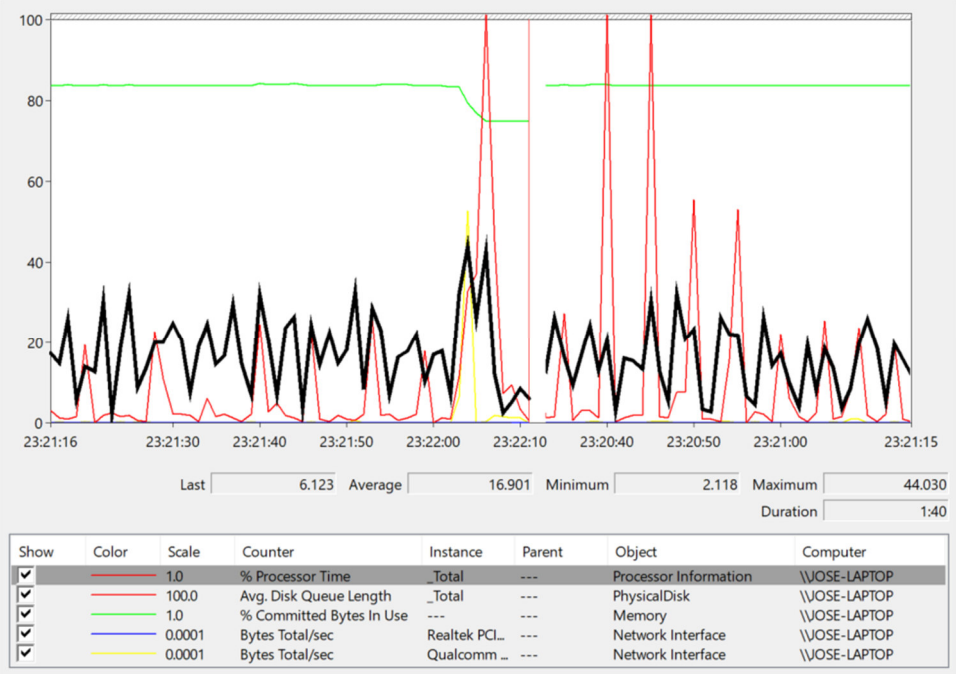
time		
Mean	725.1538164	
Standard Error	16.26865378	
Median	569	
Mode	566	
Standard Deviation	454.6499356	
Sample Variance	206706.5639	
Kurtosis	36.11722415	
Skewness	4.57227802	
Range	5212	
Minimum	262	
Maximum	5474	
Sum	566345.1306	
Count	781	
Confidence Level(95.0%)	31.93553011	
Confidence intervals	693.2182863	757.0893



Name	Total Time		Total Time (CPU)	
main	899,193 ms	(100%)	606,572 ms	(100%)
Reference Handler	899,193 ms	(100%)	81.7 ms	(100%)
Finalizer	899,193 ms	(100%)	178 ms	(100%)
RMI TCP Accept-0	899,193 ms	(100%)	2,277 ms	(100%)
RMI Scheduler(0)	899,193 ms	(100%)	0.0 ms	(-%)
JMX server connection timeout 19	899,193 ms	(100%)	0.0 ms	(-%)
mysql-cj-abandoned-connection-cleanup	899,193 ms	(100%)	0.0 ms	(-%)
HikariPool-1 housekeeper	897,796 ms	(100%)	0.0 ms	(-%)
Catalina-utility-1	897,701 ms	(100%)	72.1 ms	(100%)
Catalina-utility-2	897,701 ms	(100%)	0.0 ms	(-%)
container-0	897,701 ms	(100%)	0.0 ms	(-%)
ReaderThread	897,195 ms	(100%)	897,195 ms	(100%)
RMI TCP Connection(2)-192.168.56.1	896,804 ms	(100%)	877,101 ms	(100%)
Exec Default Executor	896,693 ms	(100%)	0.0 ms	(-%)
Exec Stream Pumper	896,693 ms	(100%)	896,693 ms	(100%)
Exec Stream Pumper	896,693 ms	(100%)	896,693 ms	(100%)

Name	Self Time (CPU)		Total Time (CPU)	
java.net.SocketInputStream.**socketRead0[native]** ()	2,804,676 ms	(48.8%)	2,805,097 ms	(3.4%)
java.io.FileInputStream.**readBytes[native]** ()	1,795,143 ms	(31.3%)	1,795,143 ms	(2.2%)
sun.management.ThreadImpl.**dumpThreads0[native]** ()	899,193 ms	(15.7%)	899,193 ms	(1.1%)
java.io.WinNTFileSystem.**getBooleanAttributes[native]** ()	118,155 ms	(2.1%)	118,155 ms	(0.1%)
java.lang.Throwable.**fillInStackTrace[native]** ()	33,428 ms	(0.6%)	33,428 ms	(0%)
java.util.zip.ZipFile.**getEntry[native]** ()	27,978 ms	(0.5%)	27,978 ms	(0%)
java.lang.Class.**forName0[native]** ()	8,281 ms	(0.1%)	177,763 ms	(0.2%)
java.security.AccessController.**doPrivileged[native]** ()	4,236 ms	(0.1%)	2,337,358 ms	(2.8%)
java.lang.ClassLoader.**findBootstrapClass[native]** ()	4,212 ms	(0.1%)	4,212 ms	(0%)
java.io.WinNTFileSystem.**canonicalize0[native]** ()	3,707 ms	(0.1%)	3,707 ms	(0%)
java.net.DualStackPlainSocketImpl.**connect0[native]** ()	3,587 ms	(0.1%)	3,587 ms	(0%)
java.net.NetworkInterface.**getAll[native]** ()	3,177 ms	(0.1%)	3,177 ms	(0%)
sun.nio.ch.WindowsSelectorImpl.**setWakeupSocket0[native]** ()	2,468 ms	(0%)	2,468 ms	(0%)
sun.nio.ch.SocketDispatcher.**write0[native]** ()	2,353 ms	(0%)	2,353 ms	(0%)
java.net.SocketOutputStream.**socketWrite0[native]** ()	2,119 ms	(0%)	2,119 ms	(0%)
java.lang.ClassLoader.**findLoadedClass0[native]** ()	1,727 ms	(0%)	1,727 ms	(0%)
java.lang.Throwable.**getStackTraceElement[native]** ()	1,700 ms	(0%)	1,700 ms	(0%)

Por último, se ha analizado el consumo de recursos del ordenador mientras se ejecuta el proyecto, habiendo obtenido unos resultados muy favorables. El consumo se mantiene medio durante toda la ejecución sin presencia de ningún pico



Conclusión

Como conclusión, podemos sacar que el rendimiento es uno de los atributos más importantes a tener en cuenta a la hora de realizar un proyecto, ya que, si la ejecución de tests o la muestra de vistas necesita grandes tiempos de espera, esto empeora la calidad del producto que estamos creando.

Este informe nos ha servido para comprender todos los factores que pueden afectar al rendimiento de un proyecto software de distintas maneras, como el consumo de recursos provocado por dicho proyecto, los tiempos de espera para la ejecución de los tests o las diferencias entre dispositivos que hemos obtenido según sus características de fábrica.

Hay que tener por tanto mucho cuidado e intentar mantener siempre el rendimiento lo más alto posible.