Lección 1 - Medición de latencias

Tecnologías del sector financiero

Asier Sampietro Alberdi

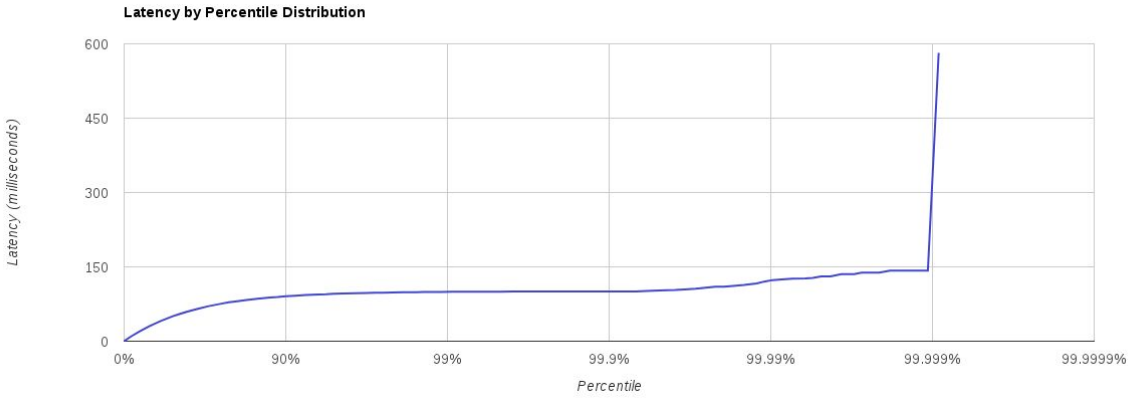
Ejercicio 1

Modificaciones realizadas

Para la realización del siguiente ejercicio, se ha modificado el código de la siguiente manera. Se ha definido el histograma, y también los valores que van a delimitar la configuración de esta. Para asegurarse de que ningún dato fuera de los márgenes vaya a causar un problema en la ejecución del programa, se ha definido el histograma como autoresizable. Después, se ha calculado el tiempo de ejecución usando nanoTime, que calcula el tiempo midiendo los tics del procesador. Esto se ha hecho porque la ejecución es demasiado breve en ciertos casos, y no permite ser medido en milisegundos. Para finalizar, se han mostrado los resultados por consola, añadiendo dos parámetros por mera curiosidad.

Análisis de resultados

La distribución por percentiles se ha almacenado en el fichero “practice1.txt” dentro de la carpeta “txt”, y con ello se ha obtenido el siguiente datagrama.



Se puede observar un comportamiento similar al descrito en los objetivos.
Hasta el percentil 98 se puede contemplar una serie de valores bajo los márgenes, excediéndose un poco a medida que se acerca al 99.
La mínima ha sido de 128 nanosegundos, por lo que asumimos que la carga añadida de la función debe ser de unos 28 nanosegundos, siendo 100 la espera mínima.
Yéndonos al otro extremo, las mediciones se mantienen entre los rangos hasta llegar a las 3 décimas de precisión, de ahí en adelante, se puede apreciar una subida debido a diferentes factores. Se llega a alcanzar el quíntuple de lo normal, debido a algún evento importante en el sistema o en la VM, y desencajando el gráfico llegado a tal punto.
Referente al código, después de hacer algunas mediciones, se han establecido los límites en 120 nanosegundos, que es lo mínimo registrado en una cantidad considerable de ejecuciones, y 4200 milisegundos, que ha sido un maximo muy puntual pero que se repite de vez en cuando. El número significante de dígitos se ha dejado en 2 puesto que no

marcaba mucha diferencia en los resultados, y se puede diferenciar así el primer caso. Por último, se han añadido unos campos por mera curiosidad o para conclusiones, como son Min o Time.

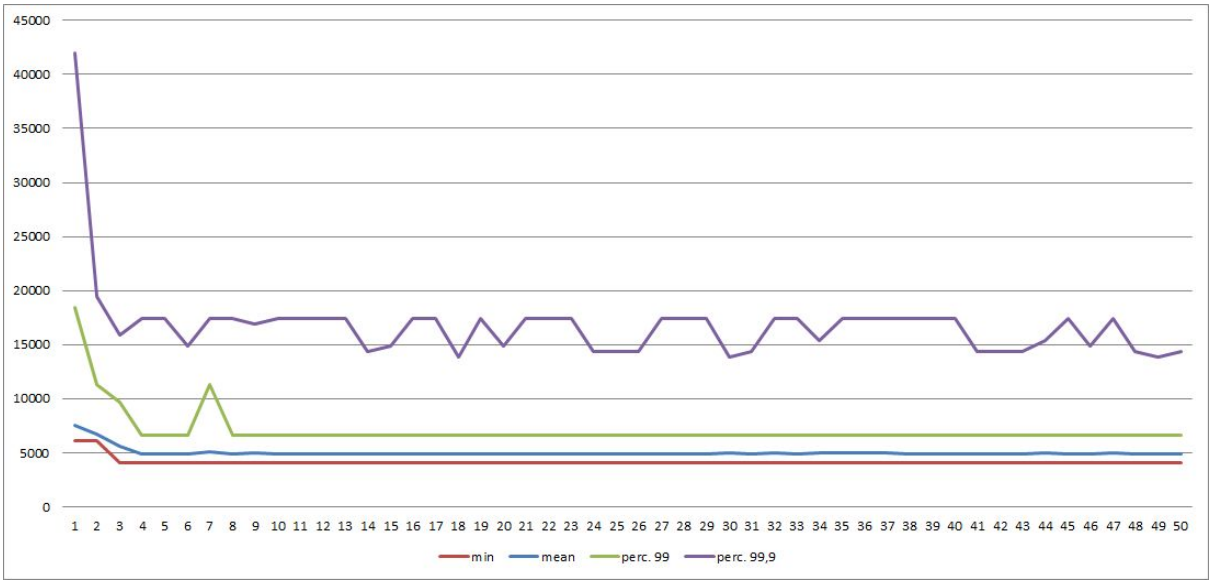
Ejercicio 2

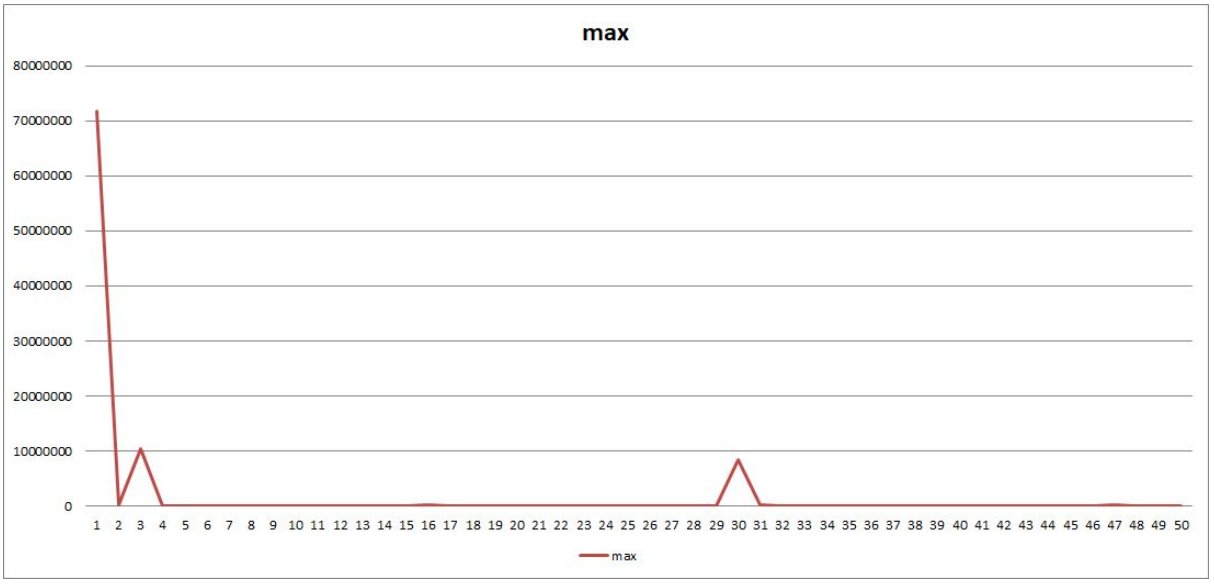
Modificaciones realizadas

La configuración ha sido la misma que en el anterior ejercicio, pero para optimizar el uso de memoria y evitar latencias mayores, se ha creado una única instancia de Histogram y se ha reseteado después de cada iteración del bucle principal. Se han impreso los resultados en pantalla antes del reseteo para no perder los datos.

Análisis de resultados

El output obtenido tras la ejecución del programa está almacenado en el documento “practice2.txt” dentro del directorio txt del proyecto. Con ello, se han obtenido los siguientes dos gráficos: uno que recoge el valor mínimo, el valor medio, el delimitante del percentil 99 y el delimitante del percentil 99,9; y otro que recoge los máximos. Se ha distribuido en dos gráficas porque las diferentes escalas dificultan la visualización.





De las ejecuciones realizadas, se ha podido observar una mejora a partir de la tercera ejecución. Se ha obtenido una diferencia de 2500 nanosegundos entre las medias de estas. Aunque los máximos también se han visto afectados, pero en una medida más irregular. Esto se debe a que por cualquier acción del sistema, una ejecución puede haber tardado más de lo normal, independientemente de la optimización en tiempo real. El reflejo mas claro esta en la media del tiempo de ejecuciones, porque realmente se ve cuando empieza a afectar el JIT.

Para evitar sobrecargar la función de objetos, se ha utilizado el mismo histograma para todas las medidas, vaciandola entre ejecuciones. También se ha intentado recoger los datos de la ejecución en una lista dinámica, pero se ha observado un impacto en el rendimiento de la tarea, por lo que se ha descartado. Lo más eficaz ha sido presentar los resultados a medida que se obtengan, o almacenarlos en un espacio fijo.

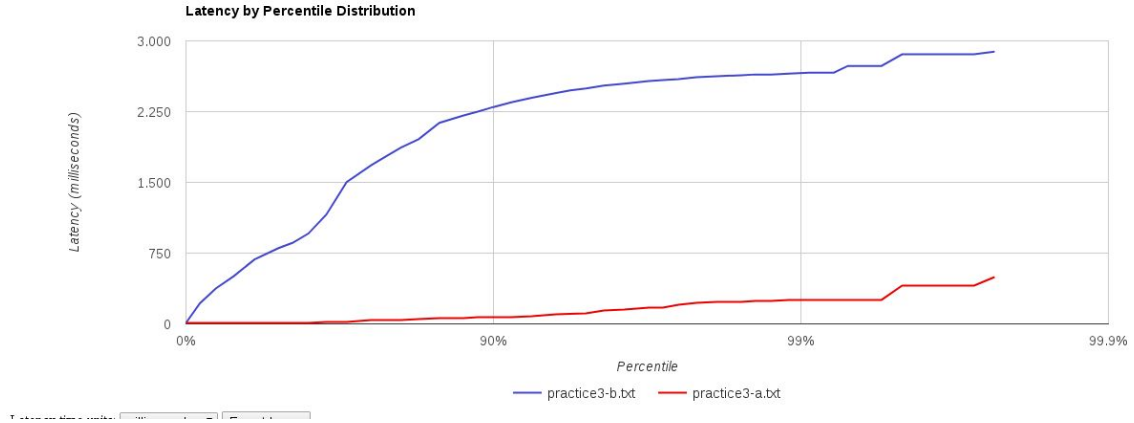
Ejercicio 3

Modificaciones realizadas

Este ejercicio ha requerido más modificaciones que el resto, ya que para hacerlo de manera más óptima, se han tenido que editar constructores y llamadas a funciones. Cabe destacar que el análisis de resultados se ha hecho sobre una ejecución con 4 hilos.
Como había que medir dos datos diferentes, se ha optado por usar dos instancias de histogramas. Además, se ha usado la clase ConcurrentHistrogram, porque es el mejor optimizado para sistemas concurrentes (según la documentación de la librería).
Como todas las ejecuciones paralelas deben almacenar los datos en el mismo histograma (en ambos, tanto en la absoluta como en la cumulativa), estos se han pasado en el constructor del Runner para asegurarse que todos los hilos escriban en los mismos histogramas.
Dentro de la ejecución del hilo, se ha modificado todo añadiendo cálculo de tiempos y un sistema para recoger el tiempo acumulado.
Por último, se ha cambiado el sistema de mostrar los resultados para obtener los datos por pares y hacer una comparación más sencilla con ellos.

Análisis de resultados

Se ha recogido lo que se muestra en consola en el fichero “practice3.txt” dentro del directorio “txt”. Con ello se ha obtenido la siguiente gráfica comparativa.



En la gráfica se aprecia, de color rojo, el conjunto de latencias sin tener en cuenta el retardo (referido en adelante como latencia roja). De color azul, se ve el conjunto de latencias acumuladas (referido en adelante como latencia azul), que claramente tiene una curva más acentuada.
De esta imagen se pueden comentar 2 cosas. Primero, a medida que la latencia “roja” escala ligeramente, la latencia azul obtiene un impacto drástico. Con esto se ve que el programa no es capaz de recuperar el tiempo “a deber” que tiene con facilidad. Por último, desde el percentil 99 hasta el 100 no ha habido apenas cambios. La estructura del

programa, un tiempo en reposo tan lineal, no permite muchos comportamientos inesperados.