Nicolas Arturo Alvarado 201630444

Javier David Peniche Calderón 201716772

Descripción de los monitores

Para monitorear el tiempo de ejecución cada Thread tiene una variable long con el nombre stIE, esta variable representa el tiempo en que comienza a correr el algoritmo. La forma en que se hace esto es inicializando la variable en System.currentTimeMillis(); justo en la primera linea del run() del thread.

Cuando el thread encuentra la cadena deseada se crea otra variable long y también se le asigna System.currentTimeMillis();, se calcula la diferencia entre las dos variables y el resultado es el tiempo que tardo en encontrar la entrada el thread.

Finalmente se convierte de mili segundos a minutos y segundos para tener un formato mas comprensible.

Para monitorear el CPU se tiene un thread especifico que se encarga de esto, este calcula el porcentaje de CPU que se usa en el momento con el método propuesto en el enunciado. Después con TimeUnit.MINUTES.sleep(n); el thread duerme n minutos. Finalmente cuando pasan los n minutos el thread empieza de nuevo su siclo.

Plataforma

Arquitectura: 64 bits

Cores: 4

Velocidad del procesador: 4.2 Ghz

RAM: 16 GB

Memoria JVM: 260 MB

Punto 3

Según las graficas realizadas con los datos obtenidos con las pruebas, podemos observar varias cosas. Primero vemos que el tiempo para calcular la peor entrada disminuye entre 1 thread y 4 threads, pero para 8 threads aumenta. Esto se pude deber a que la maquina en que se corrieron las pruebas tiene 4 cores físicos y cada core tiene un thread, entonces cuando uno crea mas de 4 cuatro threads el sistema operativo tiene que encargarse de turnar cada thread en los cpus y esto tiene como consecuencia que cada thread tiene que esperar mas tiempo para ejecutar.

Por la forma de la primera grafica también notamos que 4 threads es la capacidad utilizable de la maquina, ya que es acá donde se ve el “codo” a partir del cual el rendimiento comienza a disminuir. Esto concuerda con las características de la maquina ya que esta cuanta con 4 cores físicos.

Los porcentajes de CPU son los esperados ya que: para un thread utiliza entre 26.4% y 26.5%; para dos threads aproximadamente 53% y de cuatro threads en adelante es muy cercano el 100%. Estos son los resultados esperados ya que como el sistema operativo esta corriendo junto con otros servicios en el fondo, para un thread se esperaba un poco mas del 25% para dos thread un poco mas del 50% y de cuarto en adelante se esperaban valores cercanos al 100%. Estos valores se obtienen teniendo en cuanta el numero de cores de la maquina y el numero de threads en ejecución (para un thread ¼ = 25%, para dos 2/4 = 50%, etc).

También podemos notar que el uso de CPU con 8 threads es menor que con 4 threads. Esto creemos que se puede deber a la cantidad de interrupciones que tenia que hacer el sistema operativo para intercalar los threads en los cores. Esto ya que cada que se cambia el thread que corre en un core toca cargar a la memoria del core los datos específicos de ese thread de la RAM y en este tiempo el core no hace muchas operaciones.

Finalmente con respecto a los algoritmos podemos ver que SHA512 se demora mas que SHA256 y este a su vez se demora mas que MD5, esto era de esperarse ya que el tamaño de hash de SHA512 es mas grande que el de SHA256 y mucho mas grande que el de MD5. Por esto se demora un poco mas comparando los hashes, esto se muestra en los resultados finales con el cambio en el tiempo de cada algoritmo.

Graficas

MD5:

SHA256

SHA512