PRACTICA 3
Sistema Operativos y Laboratorio
Universidad de Antioquia – Ingeniería de Sistemas

- La palabra t*hread* (hilos) tiene en realidad distintas formas de entenderse dependiendo del "nivel" del que estemos hablando. En la programación de alto nivel hablamos de los *threads* que los programadores podemos crear para habilitar la concurrencia en nuestras aplicaciones; en este punto decimos concurrencia por ser el caso más general, dependiendo del sistema de cómputo sobre el que se ejecute nuestro programa puede haber tanto concurrencia como paralelismo que es el caso típico de los sistemas de cómputo actuales (Si esto no es claro para usted investiguelo). Si nuestra aplicación es web y se despliega a través de un servidor de aplicaciones hablamos entonces de los "thread pools" que definen la cantidad de hilos que pueden ser creados para atender las solicitudes (request) que llegan a nuestra aplicación. Desde el punto de vista del sistema operativo se habla de user-level threads y kernel-level threads. A nivel de hardware se habla por ejemplo de multithreading y simultaneus multithreading.

  Investigue lo que significa la palabra *thread* en cada uno de los "contextos" antes mencionados: Aplicaciones, Servidor de aplicaciones, Sistema operativo y Hardware. Además, explique cual es la relación entre los *threads* de los diferentes niveles ¿es una relación uno a uno? ¿cuál es la relación entre los *threads* que como programador defino dentro de mi aplicación y los *threads* del servidor de aplicaciones?¿cuál es la función del sistema operativo en el mapeo entre los *threads* de la aplicación -y los del servidor de aplicaciones- y los *threads* a nivel de hardware?

- La aplicación que hemos trabajado durante las últimas sesiones de laboratorio se despliega en un contenedor docker. Considerando los conceptos vistos en clase relacionados con la virtualización de la CPU (procesos) y la memoria (espacio de direccionamiento) y el concepto de concurrencia (*threads)* ¿cambia algo en el despliegue con contenedores respecto al despliegue tradicional? Para este caso analice la situación considerando contenedores docker y sistema operativo linux como host.

- Los lenguajes de programación ofrecen la posibilidad de realizar programación concurrente/paralela. Elija un lenguaje como caso de estudio (C, Java, node, Python, etc.) y explique la manera en que dicho lenguaje soporta este estilo de programación ¿Que precauciones debe tomar el programador en un esquema de programación concurrente? Tenga en cuenta que dependiendo del lenguaje que se elija (si es iterpretado como python, o si es compilado como C, o si es intermedio como Java) debe considerar los diferentes niveles que intervienen en la ejecución final del programa; por ejemplo, en el caso de java ¿que papel cumple la JVM? O en el caso de python ¿cuál es el papel del interprete?.

- En el repositorio base de la práctica 3 se compartió un archivo con extensión "jmx" que corresponde a un script de Jmeter para realizar pruebas de performance sobre nuestra aplicación. En dicho script se hacen peticiones a las operaciones *countTutorials* y *findAllAuthors.* Agregue dentro del mismo grupo de hilos peticiones a las operaciones *findAllTutorials* y *countAuthors* (se pueden basar en las peticiones de Postman que también están en el repositorio en un archivo llamado *Prac3_SO.postman_collection.json*).

  Como explicamos en las clases anteriores, en nuestra aplicación también desplegamos varios contenedores que nos permiten observar diversas métricas de rendimiento de nuestra aplicación

(Grafana y Prometheus). Realice los siguientes cambios en la configuración de nuestra aplicación y de nuestras pruebas y analice los resultados de las pruebas con ayuda del monitoreo:

- Configure el tomcat de la aplicación con un pool de 40 threads. Desde el Jmeter lance 200 threads durante 2 minutos.
- Ahora, cambie en Jmeter la cantidad de threads a 500 y repita la prueba.
- Cambie la configuración del tomcat a 250 threads y ejecute nuevamente dos pruebas con 200 y 500 threads en el Jmeter respectivamente.

Entre prueba y prueba deje pasar mínimo dos minutos para dejar que el sistema llegue nuevamente a su estado de "reposo". Céntrese únicamente en las gráficas del tablero de micrometer (el correspondiente a nuestra aplicación java); analice específicamente las siguientes gráficas: *Duration, CPU Usage, JVM Heap, Threads y Threads State.* Considerando las prestaciones de su máquina, las gráficas recien mencionadas y los conceptos aprendidos durante la clase teórica y la investigación solicitada en las preguntas previas ¿como explica usted el comportamiento del sistema al cambiar la cantidad de peticiones en el Jmeter y la cantidad de threads a nivel de servidor de aplicaciones?

Lineamientos para el entregable de la práctica

- Antes de preguntar asegúsere de haber visto todas las grabaciones de las clases que se han dado desde el regreso del paro.
- "Cacharree" la aplicación base y consulte sobre el uso de las métricas que micrometer ofrece para grafana en java ¿como deben interpretarse?
- Investigue tanto en reportes técnicos tipo paper como en foros. Consulte también la documentación oficial de las diferentes tecnologías que manejamos en la práctica (Java, SpringBoot, Tomcat, Jmeter, Docker, Grafana, Prometheus, mongo)
- Esta práctica pretende fomentar la habilidad investigativa que ustedes como ingenieros de sistemas deben desarrollar. Por lo tanto, las respuestas que ustedes den a las preguntas planteadas no se evaluarán teniendo como punto de referencia una "única explicación posible y verdadera"; más bien se evaluará la coherencia de sus argumentos (reflejo del grado de investigación realizado) y la utilización precisa de los conceptos aprendidos durante la materia.
- POR NINGÚN MOTIVO SE RECIBIRAN PRÁCTICAS INDIVIDUALES. Debe trabajar en equipo, mínimo dos personas por equipo, idealmente tres y como máximo cuatro.
- El entregable es un informe de mínimo 5 páginas y máximo 10. En este informe debe consignar las respuestas a las preguntas realizadas. No abuse de los gráficos para llenar espacio, agregue solo los que sean estrictamente necesarios y en un tamaño que permita la legibilidad (sin abusar del tamaño).