Proyecto de investigación - Hilos

Valentina Jaramillo Raquejo
valentina.jaramillor@udea.edu.co

7 de julio de 2020

Para poder responder adecuadamente la pregunta '¿Qué es un hilo?', hay que saber que los hilos o threads son parte constituyente de los núcleos, también llamados cores, en una CPU. Un núcleo es el componente que se encarga de ejecutar todas las instrucciones que se requieren para que los programas funcionen. En una CPU, actualmente, es posible encontrar más de un núcleo, lo cual evidentemente aumentaría la capacidad y simultaneidad en la ejecución de programas en el computador, ya que cada núcleo puede ejecutar una instrucción a la vez.

Ahora bien, un hilo es el flujo o secuencia de instrucciones que se deben realizar en un programa determinado. Si un núcleo tiene 2 hilos, significa que va a estar realizando las instrucciones de cada hilo casi simultáneamente. Esto significa que se va a ir alternando entre las instrucciones de cada hilo, y como esto sucede a gran velocidad, da la sensación de que se ejecutan al mismo tiempo. (García, 2019).

Un hilo se crea a través de un proceso. Cada vez que se abre una aplicación o un programa, se crea un nuevo hilo que se encargará de todas las instrucciones necesarias de esa aplicación específica, esto quiere decir que cuantos más programas se abran, más hilos se van a crear. Estos son creados por el sistema operativo, y siempre se va a encontrar un primer hilo, o hilo primario, que se encargará de las tareas iniciales, y creará el resto de los hilos dependiendo de la información del usuario y de las instrucciones a realizar. (Kumar, s.f.).

Se puede hablar de la historia de los hilos, y esta se remonta a 1965, donde se empezó a hablar del concepto de hilo como un flujo de instrucciones en secuencia, y esto sucedió con el Sistema de Tiempo Compartido de Berkeley, en el cual a los hilos les llamaban procesos. Los procesos realizaban sus interacciones por medio de variables compartidas, semáforos y otros medios similares. En 1970, Max Smith hizo un prototipo de implementación de hilos en Multics; en el cual usó múltiples pilas en un solo proceso pesado para apoyar compilaciones de fondo.

Por otro lado, el lenguaje de programación PL/I contribuyó en gran medida al avance de los hilos, aproximadamente en el año de 1965. El lenguaje, como lo define IBM, proporcionó una CALL XXX (A, B) TASK; un constructor que bifurcaba un hilo para XXX. Al final no hay certeza en si algún compilador de IBM implementó esta característica, pero se sabe que fue examinada detenidamente mientras se diseñaba Multics, lo cual finalmente resultó en la eliminación de esta.

Luego, a principios de los 70, Unix tomó partido con su noción de 'proceso', que se convirtió en un hilo secuencial de control más un espacio de direcciones virtuales (esta noción se derivó directamente del diseño del proceso de Multics). De este modo, los 'procesos', según Unix, son máquinas bastante pesadas. Como no pueden compartir la memoria (cada uno tiene su propio espacio de direcciones), interactúan a través de tuberías, señales, etc. La memoria compartida (que

también es un mecanismo pesado) se añadió mucho más tarde.

Después de algún tiempo, los usuarios de Unix comenzaron a extrañar los viejos procesos que podían compartir la memoria. Esto llevó a la 'invención' de los hilos: procesos de estilo antiguo que compartían el espacio de direcciones de un solo proceso Unix. También fueron llamados 'ligeros', a modo de contraste con los procesos Unix 'pesados'. Esta distinción se remonta a finales de los años 70 o principios de los 80, es decir, a los primeros 'micronúcleos' (Thoth (precursor del núcleo V y QNX), Ameba, Coro, la familia RIG-Accent-Mach, etc.). Por otro lado, los hilos han sido utilizados continuamente en aplicaciones de telecomunicaciones durante mucho tiempo. (O'Sullivan, 2005).

Para hablar ahora de los tipos de hilos, podemos encontrar los hilos de nivel de usuario; que como su nombre lo indica, son gestionados por el usuario, y los hilos a nivel de núcleo o kernel; que son gestionados por el sistema operativo y operan sobre el kernel, que es un núcleo del sistema operativo.

Por parte de los hilos de nivel de usuario, en este caso, el núcleo de gestión de hilos no es consciente de la existencia de los hilos. La biblioteca de hilos contiene código para crear y destruir hilos, para pasar mensajes y datos entre los hilos, para programar la ejecución de los hilos y para guardar y restaurar los contextos de los hilos. La aplicación se inicia con un solo hilo. Las ventajas de este tipo de hilos es que, por ejemplo, el cambio de hilo no requiere privilegios del modo del Kernel, además pueden ejecutarse en cualquier sistema operativo y son rápidos de crear y administrar. Pero también tiene una serie de desventajas, como lo son que en un sistema operativo típico, la mayoría de las llamadas del sistema se bloquean y que la aplicación multihilo no puede aprovechar el multiprocesamiento.

Con respecto a los hilos a nivel de kernel, la gestión de los hilos la hace el núcleo y no hay ningún código de gestión de hilos en el área de aplicación. Los hilos del kernel son respaldados directamente por el sistema operativo. De esta forma, cualquier aplicación puede ser programada para ser multi-hilo y todos los hilos dentro de una aplicación son administrados dentro de un solo proceso.

El kernel mantiene la información de contexto para el proceso como un todo y también para los hilos individuales dentro del proceso. Por otro lado, la programación del kernel se hace en base a los hilos, y así es como el kernel realiza la creación de hilos, la programación y la gestión de estos en su propio espacio. Generalmente, los hilos del kernel se crean y administran más lentamente que los hilos del usuario.

Los hilos a nivel de kernel tienen una serie de ventajas que constan en que el núcleo puede programar simultáneamente múltiples hilos del mismo proceso en múltiples procesos y si se bloquea un hilo de un proceso, el núcleo puede programar otro hilo del mismo proceso. Además de esto, las propias rutinas del kernel pueden ser multi-hilos. Sin embargo, existen algunas desventajas como que los hilos del núcleo son más lentos de crear y administrar que los del usuario, y que la transferencia de control de un hilo a otro dentro del mismo proceso requiere un cambio de modo del kernel. (*Operating System - Multi-Threading*, s.f.).

En otro enfoque, para entender cómo funcionan o se implementan los hilos a nivel del hardware y del software, y poder diferenciarlos, es necesario entender algunos conceptos básicos. Primero, un solo componente de computación que tiene más de una unidad central de procesamiento independiente (núcleo) es un proce-

sador multinúcleo, estos núcleos leen y ejecutan las instrucciones de cada programa. De esta forma es posible ejecutar instrucciones al mismo tiempo, en paralelo.

Por esto es por lo que, para tomar ventaja de los múltiples núcleos físicos, se necesita ejecutar muchos procesos o ejecutar más de un hilo en un solo proceso, creando así un código multi-hilo. Sin embargo, cada núcleo físico puede ofrecer más de un hilo de hardware, llamado también núcleo o procesador lógicos.

Por ejemplo, cada programa que se ejecuta en Windows es un proceso. Cada proceso crea y ejecuta uno o más hilos, llamados hilos de software, para diferenciarlos de los hilos de hardware anteriormente mencionados.

Cada proceso tiene al menos un hilo, llamado el hilo principal. Un programador interno del sistema operativo reparte equitativamente los recursos de procesamiento disponibles entre todos los procesos e hilos que tienen que ser ejecutados. Este programador asigna tiempo de procesamiento a cada hilo de software. Cuando el programador de Windows se ejecuta en un microprocesador de varios núcleos, tiene que asignar el tiempo de un hilo de hardware, apoyado por un núcleo físico, a cada hilo de software que necesita ejecutar instrucciones. A modo de analogía, se puede pensar en cada hilo de hardware como una pista de natación y en un hilo de software como un nadador. (*Hardware Threads vs Software Threads*, 2012).

El sistema operativo de Windows reconoce cada hilo de hardware como un procesador lógico programable. Cada procesador lógico puede ejecutar el código de un hilo de software. Un proceso que ejecuta código en múltiples hilos de software puede aprovechar los hilos de hardware y los núcleos físicos para ejecutar instrucciones en paralelo. El programador de Windows puede decidir reasignar un hilo de software a otro hilo de hardware para equilibrar el trabajo realizado por cada hilo de hardware. Puesto que normalmente hay muchos otros hilos de software esperando obtener tiempo de procesamiento, el equilibrio de carga hará posible que estos otros hilos ejecuten sus instrucciones organizando los recursos disponibles. (*Hardware Threads vs Software Threads*, 2012).

Todas estas características no siempre están disponibles para ser programadas en todos los lenguajes de programación, ya que cada uno tiene sus particularidades, y algunos pueden o no soportar la función de multi-hilo. Un ejemplo de un lenguaje de programación que puede usar librerías con funcionalidades de multi-hilo, es el lenguaje C, en el cual, con la librería pthreads o POSIX Threads, es posible crear programas que tengan más de un flujo de ejecución de instrucciones, lo cual permite que se obtenga una mayor velocidad en los programas.

Referencias

García, J. (2019, 10). *Núcleos e hilos de un procesador.* Descargado de `https://jesgargardon.com/blog/nucleos-e-hilos-de-un-procesador/`

Hardware Threads vs Software Threads. (2012, 10). Descargado de `http://paxcel.net/blog/hardware-threads-vs-software-threads/`

Kumar, R. (s.f.). *What are Threads in Computer Processor or CPU?* Descargado de `https://www.geeksforgeeks.org/what-are-threads-in-computer-processor-or-cpu/`

Operating System - Multi-Threading. (s.f.). Descargado de `https://www`

.tutorialspoint.com/operating_system/os_multi_threading.htm

O'Sullivan, B. (2005, 9). *The history of threads.* Descargado de http://www.serpentine.com/blog/threads-faq/the-history-of-threads/#:~:text=The%20notion%20of%20a%20thread,%2C%20semaphores%2C%20and%20similar%20means.