Universidad Nacional de San Agustín de Arequipa

Escuela Profesional de Ciencia de la Computación

Sistemas Operativos

Threads

Presentado por:

Philco Puma, Josue Samuel

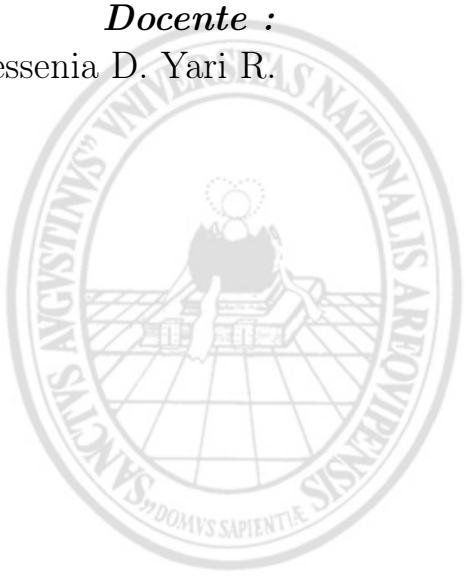
Zavalaga Orozco, Rushell Vanessa

Malcoaccha Díaz, Erick Rubén

Aquino Mamani, Ana Karina

Docente :

Dra. Yessenia D. Yari R.



1. Hilos en GPU

Existen los hilos en GPU? Cómo trabajan? Cómo se asignan el uso de la GPU?

Los hilos en la GPU son fundamentales para el procesamiento paralelo, ya que permiten la ejecución de múltiples tareas simultáneamente.

Existencia y Funcionamiento de los Hilos en GPU

Las GPU están diseñadas para manejar hilos de ejecución que permiten hacer cálculos paralelos de manera eficiente.

- **Modelo de Ejecución**: En el modelo CUDA (Compute Unified Device Architecture) de NVIDIA, los hilos se agrupan en bloques. Cada bloque se organiza en una cuadrícula (grid) y cada hilo dentro del bloque tiene un identificador que le permite acceder a datos específicos.
- **Paralelismo Masivo**: Las GPU están optimizadas para ejecutar miles de hilos simultáneamente. Por ejemplo, una GPU puede tener miles de núcleos y ser capaz de ejecutar decenas de miles o millones de hilos al mismo tiempo. Esto es posible porque los hilos pueden ser programados para realizar tareas independientes y repetitivas, como procesar píxeles en imágenes o realizar cálculos en matrices.

Asignación de los Hilos

Para asignar los hilos se utiliza un modelo de programación que organiza los hilos en bloques y mallas.

- **Organización de Hilos**:
  - **Bloques y Mallas**: Cuando se lanza un *kernel* (una función que se ejecuta en la GPU), el programador especifica el número de bloques y el número de hilos por bloque. Estos hilos se organizan en una estructura llamada malla (grid), donde cada bloque puede contener varios hilos.
  - **Identificación de Hilos**: Cada hilo dentro de un bloque tiene un identificador único accesible a través de la variable `threadIdx`, que permite a cada hilo saber qué datos debe procesar. Similarmente, cada bloque tiene un identificador accesible mediante `blockIdx`, lo que facilita el direccionamiento de memoria.
- **Asignación y Ejecución**:

- **Distribución entre Multiprocesadores**: Una vez que se lanza un *kernel*, la GPU distribuye los bloques entre los multiprocesadores (Streaming Multiprocessors - SM) disponibles. Cada SM puede manejar múltiples bloques simultáneamente, lo que maximiza la utilización del hardware.
- **Limitaciones del Hardware**: La cantidad máxima de hilos por bloque y el número total de bloques que pueden ser lanzados están limitados por las características del hardware específico.

2. Ejemplos de Threads en Windows

La técnica para crear subprocesos mediante la biblioteca de subprocesos de Windows es similar a la técnica Pthreads. Windows utiliza la asignación uno a uno.

Los subprocesos se crean en la API de Windows mediante la función CreateThread() y se pasa un conjunto de atributos para el subproceso a esta función. Estos atributos incluyen información de seguridad, el tamaño de la pila y un indicador que se puede configurar para indicar si el subproceso debe iniciar en un estado suspendido.

2.1. Ejemplo 1:

```
#include <windows.h>
#include <stdio.h>

DWORD WINAPI HiloFuncion(LPVOID Param){
    printf("Hola desde el hilo!\n");
    return 0;
}

int main(){
    HANDLE Hilo;        /* Manejador del hilo */
    DWORD ThreadId;     /* Identificador del hilo */

    /* Crear un hilo que ejecutará la función HiloFuncion */
    Hilo = CreateThread(
        NULL,           /* Atributos de seguridad predeterminados */
        0,              /* Tamaño predeterminado de la pila */
        HiloFuncion,    /* Función que ejecutará el hilo */
        NULL,           /* Parámetro de la función del hilo */
        0,              /* Bandera de creación predeterminada */
        &ThreadId);     /* Devuelve el identificador del hilo */

    if (Hilo == NULL){
        printf("Error al crear el hilo.\n");
```

```
        return 1;
    }

    WaitForSingleObject(Hilo, INFINITE);

    CloseHandle(Hilo);

    printf("El hilo ha terminado.\n");
    return 0;
}
```

```
PS C:\CODE\CODE S6\OS\LAB5_THREADS> g++ -o main .\EJ2.c
PS C:\CODE\CODE S6\OS\LAB5_THREADS> ./main
Hola desde el hilo!
El hilo ha terminado.
PS C:\CODE\CODE S6\OS\LAB5_THREADS>
```

2.2. Ejemplo 2:

```
#include <windows.h>
#include <stdio.h>

DWORD Sum; /* Variable global Sum será compartida por el hilo */

/* Esta función será ejecutada por el hilo creado */
DWORD WINAPI Summation(LPVOID Param) {
    // Se convierte el parámetro a un puntero de DWORD y se desreferencia para obtener
    DWORD Upper = *(DWORD*)Param;

    printf("Sumando ", 1);

    for (DWORD i = 1; i <= Upper; i++) {
        printf(" + %lu ", i);
        Sum += i;
    }

    printf("\n");
    return 0;
}

int main(int argc, char *argv[]) {
    DWORD ThreadId; // Identificador del hilo.
    HANDLE ThreadHandle; // Manejador del hilo.
```

```
    int Param;

    // Convierte el argumento de la línea de comandos a un entero.
    Param = atoi(argv[1]);

    /* Se crea el hilo */
    ThreadHandle = CreateThread(
        NULL, /* Atributos de seguridad por defecto */
        0, /* Tamaño de pila por defecto */
        Summation, /* Función que ejecutará el hilo */
        &Param, /* Parámetro que se le pasa a la función del hilo */
        0, /* Flags de creación por defecto */
        &ThreadId); /* Almacena el identificador del hilo */

    /* Se espera a que el hilo termine su ejecución */
    WaitForSingleObject(ThreadHandle, INFINITE);

    /* Se cierra el manejador del hilo */
    CloseHandle(ThreadHandle);

    printf("sum = %d", Sum);
}
```

```
PS C:\CODE\CODE S6\OS\LAB5_THREADS> g++ -o main .\EJ1.c
PS C:\CODE\CODE S6\OS\LAB5_THREADS> ./main 5
Sumando  + 1  + 2  + 3  + 4  + 5
sum = 15
PS C:\CODE\CODE S6\OS\LAB5_THREADS> ./main 10
Sumando  + 1  + 2  + 3  + 4  + 5  + 6  + 7  + 8  + 9  + 10
sum = 55
PS C:\CODE\CODE S6\OS\LAB5_THREADS> ./main 50
Sumando  + 1  + 2  + 3  + 4  + 5  + 6  + 7  + 8  + 9  + 10  + 11  + 12  + 13  + 14  +
  + 30  + 31  + 32  + 33  + 34  + 35  + 36  + 37  + 38  + 39  + 40  + 41  + 42  + 43
sum = 1275
PS C:\CODE\CODE S6\OS\LAB5_THREADS> ./main 100
Sumando  + 1  + 2  + 3  + 4  + 5  + 6  + 7  + 8  + 9  + 10  + 11  + 12  + 13  + 14  +
  + 30  + 31  + 32  + 33  + 34  + 35  + 36  + 37  + 38  + 39  + 40  + 41  + 42  + 43
58  + 59  + 60  + 61  + 62  + 63  + 64  + 65  + 66  + 67  + 68  + 69  + 70  + 71  + 72
+ 87  + 88  + 89  + 90  + 91  + 92  + 93  + 94  + 95  + 96  + 97  + 98  + 99  + 100
sum = 5050
PS C:\CODE\CODE S6\OS\LAB5_THREADS>
```