Instituto Politécnico Nacional

Escuela Superior de Cómputo

Cómputo de alto desempeño

Prof. **Benjamín Cruz Torres**

Ejercicios Prácticos No. 01
Identificación de paralelización

Grupo: 4CDV1

Equipo: NetPower

Integrantes:

1. Alcibar Zubillaga Julián
2. De Luna Ocampo Yanina

Fecha:10/02/2022

Ejercicios prácticos – Paralelización de programas parte 1.

Objetivo

Que el alumno o alumna identifique las rutinas del lenguaje de programación que pueden paralelizarse.

Instrucciones

Lee detenidamente cada enunciado que se presenta. Codifica en algún lenguaje de programación el problema (usa computación serial). Una vez que tengas el programa funcionando, identifica cuáles partes del programa se podrían paralelizar. Calcula cuánto tiempo tardaría el programa, usando el valor de *n* como base.

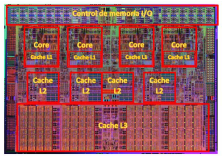
INTRODUCCIÓN

Recordemos que el procesamiento en paralelo es la realización simultánea de una serie de tareas o instrucciones en dos o más unidades de procesamiento para que el resultado del procesamiento sea más rápido. La idea de este es dividir las tareas o instrucciones en subtareas más simples que se conocen como subprocesos. Estos se ejecutan simultáneamente en cada uno de los núcleos de la unidad de procesamiento con el fin de reducir los tiempos de procesamiento. En nuestra casa tenemos procesadores en paralelo, como nuestros teléfonos, nuestras computadoras e incluso algunos de nuestras computadoras. En este tipo de programación, se producen más errores debido a la concurrencia. Asimismo, la sincronización y comunicación entre los subprocesos son algunas desventajas de realizar este tipo de programas.

Hay diversos tipos de programación en paralelo, por mencionar algunos:

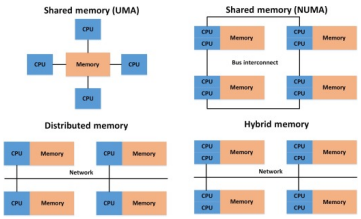
1. Procesadores multinúcleo.

   Aquí tenemos más de una unidad de ejecución, lo que llamamos núcleos o cores. De esta forma un procesador con varios núcleos puede ejecutar más de una tarea en un solo ciclo de reloj. Esto se hace, creando subprocesos, que es un flujo de instrucciones más simple que optimiza los tiempos de espera de cada núcleo.



2. Multiprocesadores simétricos (SMP)

   Se tiene una placa o PCB en donde dos a más procesadores idénticos entre ellos trabajan con recursos compartidos como será una memoria principal y un bus de datos. La ventaja de este método es su escalaridad y las desventaja presente es tener que compartir el bus de datos el cual debe ser controlado y arbitrado para que esté disponible para la CPU que solicite acceso.



3. Procesamiento Paralelo Masivo

   Un MPP será un solo equipo con multitud de procesadores interconectados en red. La interconexión es más especializada que en un clúster y no de consumo general, contando cada procesador con su propia memoria RAM y con una copia del sistema operativo y aplicación para la ejecución en paralelo.

1. Programa que calcula la suma y la resta de dos vectores, cada uno de n dimensiones. Generar los vectores con valores aleatorios de 0 a 100

Gv:= Tiempo que se tarda la computadora generando un número al azar.

Sv:=Tiempo que se tarda la computadora sumando dos números.

Rv:=Tiempo que se tarda la computadora restando dos números.

Tt:=Tiempo total= 2* Gv*n + n * Sv + n * Rv

Hay que notar que los procesos que podemos catalogar como independientes en este caso es al momento de generar los vectores, podemos paralelizar esos procesos ya que no se involucran, por último podemos paralelizar la resta y la suma ya que también son dos procesos independientes.

```
cout << "Arreglo 1" << endl;
imprimirArreglo(arreglo1, x: n);
cout << endl;
cout << "Arreglo 2" << endl;
imprimirArreglo( arreglo1: arreglo2, x: n);
cout << endl;

int sArreglo[n];
int rArreglo[n];

sumaArreglo( sumaArreglo: sArreglo,arreglo1,arreglo2, x: n);

cout << "Suma Arreglo" << endl;
imprimirArreglo( arreglo1: sArreglo, x: n);
cout << endl;

restaArreglo( restaArreglo: rArreglo,arreglo1,arreglo2, x: n);

cout << "Resta Arreglo" << endl;
imprimirArreglo( arreglo1: rArreglo, x: n);
cout << endl;
```

```
int *imprimirArreglo(int *arreglo1, int x){
    for (int i=0;i<x;i++) {
        cout << arreglo1[i];
        cout << " ";
    }
    return arreglo1;
}

int *sumaArreglo(int *sumaArreglo,int *arreglo1, int *arreglo2,int x){
    for (int i=0;i<x;i++) {
        sumaArreglo[i]=arreglo1[i]+arreglo2[i];
    }
}

int *restaArreglo(int *restaArreglo,int *arreglo1, int *arreglo2,int x){
    for (int i=0;i<x;i++) {
        restaArreglo[i]=arreglo1[i]-arreglo2[i];
    }
}
```

```
Arreglo 1
41 67 34 0 69 24 78 58 62 64
Arreglo 2
5 45 81 27 61 91 95 42 27 36
Suma Arreglo
46 112 115 27 130 115 173 100 89 100
Resta Arreglo
36 22 -47 -27 8 -67 -17 16 35 28

Process finished with exit code 0
```

2. Programa que calcula el producto escalar de dos vectores, ambos de n dimensiones. Generar los vectores con valores aleatorios de 0 a 100

Gv:= Tiempo que se tarda la computadora generando un número al azar.

Sv:=Tiempo que se tarda la computadora sumando dos números.

Pv:=Tiempo que se tarda la computadora multiplicando dos números.

Tt:=Tiempo total= 2* Gv*n + n * Sv + n * Pv

El primer proceso que podemos paralelizar es como el inciso anterior, ya que al crear dos vectores aleatorios con números al azar, lo podemos hacer independientemente, por otro lado, podemos paralelizar la multiplicación de las casillas y al finalizar poder sumar todos esos productos.

```
int productoPunto=0;

productoPunto=productoEscalar( productoArreglo: productoPunto,arreglo1,arreglo2, x: n);

cout << "Producto Punto" << endl;
cout << "El producto punto es: " << productoPunto;
cout << endl;
```

```
int productoEscalar(int productoArreglo,int *arreglo1, int *arreglo2,int x){
    for (int i=0;i<x;i++) {
        productoArreglo+=arreglo1[i]*arreglo2[i];
    }
    return productoArreglo;
}
```

```
Arreglo 1
41 67 34 0 69 24 78 58 62 64
Arreglo 2
5 45 81 27 61 91 95 42 27 36
Producto Punto
El producto punto es: 26191

Process finished with exit code 0
```

3. Programa que calcula el promedio de datos de un arreglo de n dimensiones. Generar el vector con valores aleatorios de 0 a 1000

Gv:= Tiempo que se tarda la computadora generando un número al azar.

Sv:=Tiempo que se tarda la computadora sumando dos números.

Pv:=Tiempo que se tarda la computadora dividiendo dos números.

Tt:=Tiempo total= Gv*n + Sv * n + Pv

En el caso del promedio, podemos ir sumando los números a la vez que los generamos, por lo que lo podríamos hacer paralelamente, o incluso podemos dividir el arreglo e ir haciendo que recorra el arreglo una parte de abajo y la otra arriba, dividiéndolo en dos y posteriormente sumarlo.

```
Arreglo 1
41 67 34 0
Arreglo 2
69 24 78 58
Promedio del Arreglo 1 es:
El promedio del arreglo es: 35

Process finished with exit code 0
```

```
int promedioArreglo(int *arreglo1, int x){
    int sumaT=0;
    for (int i=0;i<x;i++) {
        sumaT+=arreglo1[i];
    }
    return sumaT/x;
}
```

```
    int promedio=0;

    promedio=promedioArreglo(arreglo1, x: n);

    cout << "Promedio del Arreglo 1 es:" << endl;
    cout << "El promedio del arreglo es: " << promedio;
    cout << endl;
}
```

4. Programa que calcula la desviación estándar de un arreglo de n dimensiones. Generar el vector con valores aleatorios de 0 a 1000

Gv:= Tiempo que se tarda la computadora generando un número al azar.

Sv:= Tiempo que se tarda la computadora sumando dos números.

Rv:= Tiempo que se tarda la computadora restando dos números

Pv:= Tiempo que se tarda la computadora dividiendo dos números.

Mv:= Tiempo que se tarda la computadora multiplicando dos números.

Tt:=Tiempo total= Gv*n + Sv * n + Rv*n + Mv*n + Pv

Aquí podemos hacer el proceso de restar el valor de la casilla y multiplicarlo al cuadrado paralelamente, ya que estos no se necesitan entre sí.

```
int desviacion=0;

desviacion= desviacionEstandar(arreglo1, x: n);

cout << "Desviacion Estandar del Arreglo 1 es:" << endl;
cout << "La desviacion estandar del arreglo es: " << desviacion;
cout << endl;
```

```
int desviacionEstandar(int *arreglo1, int x){
    int promedio = promedioArreglo(arreglo1,x);
    int sumaT=0;
    for (int i=0;i<x;i++) {
        sumaT+= (arreglo1[i]-promedio)*(arreglo1[i]-promedio);

    }
    return sqrt( x: sumaT/x);
}
```

```
Arreglo 1
41 67 34 0 69 24
Arreglo 2
78 58 62 64 5 45
Desviacion Estandar del Arreglo 1 es:
La desviacion estandar del arreglo es: 24

Process finished with exit code 0
```

5. Programa que calcula la desviación estándar de un conjunto de m vectores cada uno de tamaño n. Generar los vectores con valores aleatorios de 0 a 1000

Gv:= Tiempo que se tarda la computadora generando un número al azar.

Sv:= Tiempo que se tarda la computadora sumando dos números.

Rv:= Tiempo que se tarda la computadora restando dos números

Pv:= Tiempo que se tarda la computadora dividiendo dos números.

Mv:= Tiempo que se tarda la computadora multiplicando dos números.

Tt:=Tiempo total= Gv*n*m + Sv * n*m + Rv*n*m + Mv*n*m + Pv*m

El proceso de crear los vectores en cada casilla de un vector lo podemos hacer paralelamente, también calcular cada desviación estándar es un proceso que podemos hacer paralelamente.

```
cout << "Ponemos numeros random en el arreglo:" << endl;

for (int i = 0; i < m ; i++) {
    randomArreglo( arreglo1: arreglo3[i], x: n);
}

cout << "Imprimir arreglos" << endl;


for (int i = 0; i < m ; i++) {...}

cout << "Aqui va la desviacion estandar" << endl;

int desviacionVector[m];

for (int i = 0; i < m ; i++) {
    desviacionVector[i]=desviacionEstandar( arreglo1: arreglo3[i], x: n);
}

cout << "Vector de la desviacion estandar" << endl;

imprimirArreglo( arreglo1: desviacionVector, x: m);
```

```
Ponemos numeros random en el arreglo:
Imprimir arreglos
41 67 34 0 69 24
78 58 62 64 5 45
81 27 61 91 95 42
27 36 91 4 2 53
92 82 21 16 18 95
47 26 71 38 69 12
Aqui va la desviacion estandar
Vector de la desviacion estandar
24 23 25 30 35 21
Process finished with exit code 0
```

Conclusiones

Como hemos estudiado, el procesamiento en paralelo es una de las implementaciones que más beneficios han traído en la actualidad, creándose verdaderos procesamientos de información a nivel mundial, un ejemplo de esto, es lo visto en clase del TOP500 de los ordenadores más potentes del mundo.

El procesamiento distribuido y en paralelo es una alternativa para gestionar estos grandes volúmenes de datos, los cuales pueden provenir de aplicaciones como chats, blogs, servicios de correo, redes sociales, etc. Uno de los sistemas más usados para este propósito es Hadoop.

No siempre se puede paralelizar una tarea. En principio, cuanto más nodos actúen en paralelo, más rápido irá el sistema, lo común es que haya un límite en que añadir más nodos no aumente la eficiencia, sino que incluso la reduzca. Normalmente las tareas en paralelo tienen una fase secuencial al principio en la que se dividen las tareas y otra al final en la consolidan resultados.

Ejemplo de tareas secuenciales:

- Cajero automático
- Leer una novela
- Ver a alguien

Tareas paralelas:

- Consultar enciclopedia
- Buscar un nombre en una base de datos
- Contar votos en procesos electorales
- Analizar rasgos faciales

El procesamiento paralelo es particularmente beneficioso en áreas como el tiempo y el clima, reacciones químicas y nucleares, exploración de petróleo, medición de datos sísmicos, tecnología espacial, circuitos electrónicos, genoma humano, medicina, gráficos avanzados y realidad virtual y proceso de fabricación.

Bibliografía

Castillo, J. A. (2020, 14 abril). *Procesador en paralelo: que es y para qué sirve*. Profesional Review. https://www.profesionalreview.com/2020/05/10/procesador-en-paralelo/#:%7E:text=aburrir%20al%20personal.-,Qu%C3%A9%20es%20el%20procesamiento%20en%20paralelo,se%20ejecutan%20de%20forma%20simult%C3%A1nea.&text=Cada%20uno%20de%20estos%20subprocesos,espera%20entre%20tarea%20y%20tarea.

Arboleda, F. J. M. (2017). *Procesamiento en paralelo y distribuido en dos SGBDS: un caso de estudio*. Tecnura. https://www.redalyc.org/journal/2570/257051186010/html/

Orbe, A., & Perfil, V. T. M. (2011). *Procesamiento secuencial y paralelo. Tareas, ordenadores y cerebro*. Sinapsis. https://sinapsis-aom.blogspot.com/2011/01/procesamiento-secuencial-y-paralelo.html

Consideraciones finales

Descarga el documento antes de llenarlo.

Este documento se debe llenar en equipo, aunque la actividad la deben hacer TODOS los integrantes del mismo.

Después de llenar el documento, guárdalo como PDF y envíalo a través del tema correspondiente en la plataforma *TEAMS*.

Queda estrictamente prohibido cualquier tipo de plagio. El equipo o equipos involucrados en ello será penalizado con la anulación de su asignación y dos puntos menos a su calificación parcial.