Universidad del Valle de Guatemala
Facultad de Ingeniería
Departamento de Ciencias de la Computación
Computación Paralela y Distribuida - Sección 20 - 2024 – 2
Ing. Sebastian Galindo, Ing. Juan Luis Garcia

Proyecto # 1

Fecha de Entrega: Semana del 26 al 30 de agosto 2024

Antecedentes: OpenMP es una herramienta que nos facilita la transformación de un programa secuencial en paralelo, así como nos permite la abstracción y programación paralela en alto nivel. Debido a que está pensado como una solución iterativa, podemos realizar cambios graduales en un programa secuencial para aprovechar múltiples recursos mediante ejecución paralela.

Objetivos y Competencias:

- Implementar y diseñar un programa para la paralelización de procesos con memoria compartida usando OpenMP
- Aplicar el método PCAM y los conceptos de patrones de descomposición y programación para modificar un programa secuencial y volverlo paralelo
- Realizar mejoras y modificaciones iterativas al programa para obtener mejores versiones.

Descripción: En equipos de 3 integrantes, deben diseñar e implementar un algoritmo secuencial que resuelve un problema que tiene potencial para ser paralelizado. Luego, cada equipo deberá modificar y mejorar el programa buscando un incremento en el desempeño del mismo (basándonos en los conceptos de speedup y eficiencia estudiados).

Requisitos: Para la implementación de la solución paralela, cada equipo debe cumplir por lo menos con las siguientes condiciones. Incumplir con ellas conlleva penalizaciones en la nota:

A. Código de autoría propia en C/C++. Incluir comentarios explicativos de sus métodos, variables etc...
B. Historial del control de versiones del programa como mínimo iniciando 2 semanas antes de la entrega del proyecto (privado, ponerlo público hasta el día de entrega). El historial debe reflejar que trabajaron con tiempo y no “a última hora” (esto se puede ver en los commits).
C. Uso de OpenMP
D. Versión secuencial y paralela(s) del algoritmo.

Contenido:

La descripción del temario es bastante compacta: **realizar un programa que dibuje en pantalla un "screensaver" y que corra de forma paralela.**

Comenzaran diseñando y programando la versión secuencial. Una vez su versión secuencial esté lista (funcional y corriendo), procederán a buscar acelerarla y mejorarla utilizando OpenMP y sus conocimientos de programación paralela. Como ejemplo, una captura a continuación realizada en C++ con OpenMP y la libreria SDL (https://www.libsdl.org/):

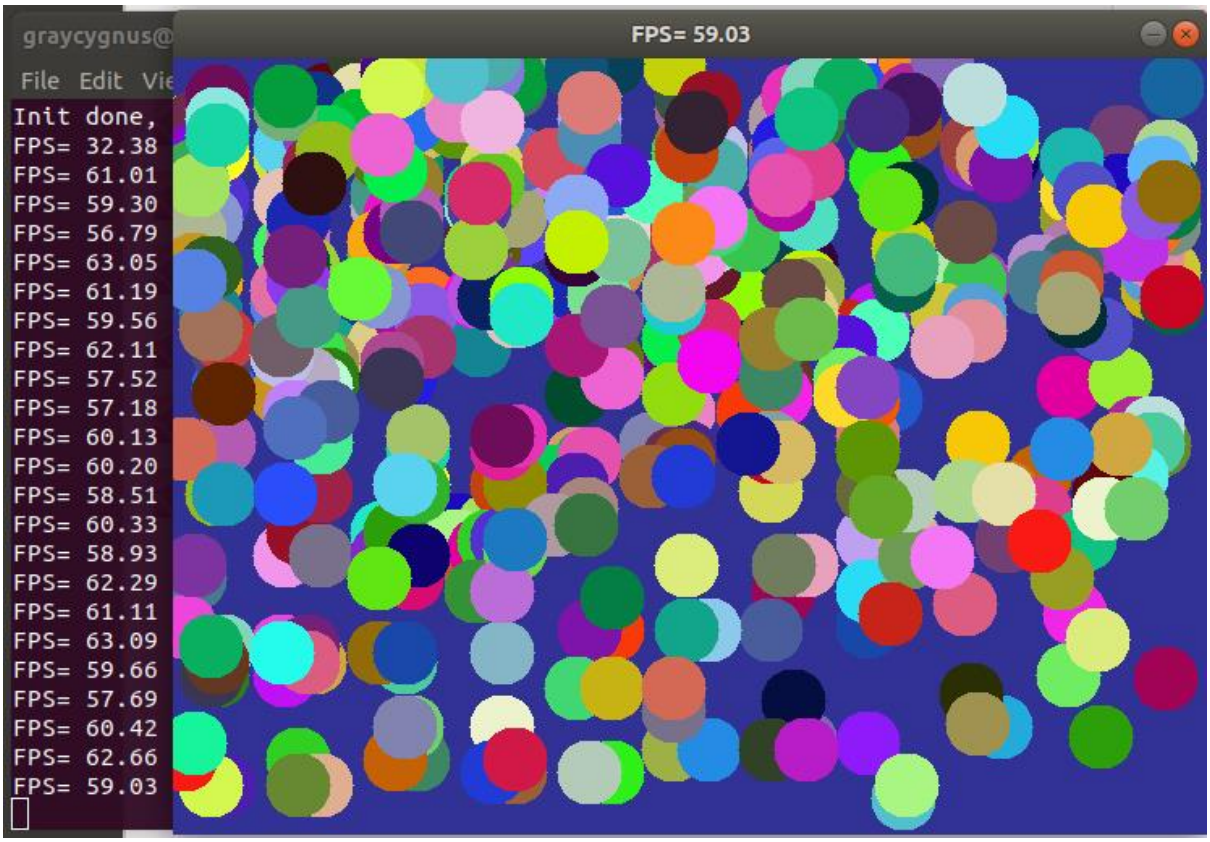

Imagen 1.: Ejemplo de un Screensaver que genera N círculos que se mueven y rebotan.

Como screensavers pueden haber de varios sabores y colores, acá las reglas básicas a seguir para que el proyecto sea de un tamaño manejable:

- Su screensaver debe recibir por lo menos un parámetro (N), el cual indica la cantidad de elementos a renderizar (en el ejemplo N indica la cantidad de circulos a generar).
- Su screensaver debe desplegar varios colores, idealmente generados de forma pseudoaleatoria.
- El tamaño del canvas debe ser de mínimo 640x480 (w,h)
- Su screensaver debe tener movimiento y ser estéticamente interesante (en el ejemplo, los círculos se mueven y rebotan en las paredes).
- Su screensaver debe incorporar algún elemento de física o trigonometría en sus cálculos (por ejemplo, cambio de velocidad al rebotar con los bordes, colisiones, interacciones entre los N elementos, etc.).
- Debe desplegar los FPS (frames per second) en los que corre su programa. Ello a modo de entender la experiencia de usuario y garantizar que no caigan los FPS (<30).

Para librería para el dibujo y gráficos pueden utilizar la que gusten o prefieran. Se sugiere considerar la librería SDL o bien OpenGL.

Entregables:

1. Informe del proyecto de investigación en formato PDF, siguiendo las normas para informes UVG.
2. Código fuente funcional (sus archivos .c y .cpp). **No** entregar ejecutables.
3. Link al repositorio de código utilizado que contiene el código que entregaron. Dejarlo privado hasta el momento de la evaluación.

Todo material debe estar subido antes del periodo de clase del 6 de septiembre.

Evaluación:

	Informe – 25 %	**valor**
1	Formato según guía de informes UVG: carátula, índice, introducción, antecedentes, cuerpo, citas textuales / píe de página, conclusiones / recomendaciones, apéndice con material suplementario y al menos 3 citas bibliográficas relevantes y confiables	10%
3	Anexo 1 - Diagrama de flujo del programa − detalle de todos los pasos incluyendo: ● Captura de argumentos ● Solicitud de ingreso de datos ● Programación defensiva ● Secciones paralelas ● Mecanismos de sincronía ● Despliegue de resultados	5%

4	Anexo 2 − Catálogo de funciones ● Entradas − nombres de variables, tipos y descripción de su uso ● Salidas − nombres de variables, tipos y descripción de su uso ● Descripción − Propósito de la función / clase / subrutina y descripción del funcionamiento	5%
5	Anexo 3 − Bitácora de pruebas, con un mínimo de 10 mediciones por prueba y resultados obtenidos (eficiencia). Incluir captura de las mediciones.	5%

	Programa y presentación – 75 %	**valor**
1	Programa secuencial funcionando correctamente y desplegando el screensaver: Algunas consideraciones: ● Programa compila sin errores. Programa se ejecuta sin fallar. ● Programación defensiva en caso de errores en el ingreso de datos. ● Incluir readme.md ● Evitar hard-coded variables. Parametrizar sus variables leyendo args del comando.	25%
2	Programa paralelo funcionando correctamente y desplegando el screensaver. Algunas consideraciones: ● Programa compila sin errores. Programa se ejecuta sin fallar. ● Programación defensiva en caso de errores en el ingreso de datos. ● Incluir readme.md ● Evitar hard-coded variables. Parametrizar sus variables leyendo args del comando.	35%
3	Documentación y comentarios explicativos sobre las partes importantes del programa	5%
4	Programa ordenado, con nombres de variables de entrada y salida y nombres de rutinas adecuadamente identificadas (nombres nemotécnicos)	5%
5	Inclusión y buen uso de mecanismos de protección de memoria compartida o mecanismo de barrera/sincronía. Manejo adecuado de inicialización y destrucción de objetos y memoria.	5%