ESCUELA DE CIENCIAS DE LA
COMPUTACIÓN E INFORMÁTICA
UNIVERSIDAD DE COSTA RICA

Proyecto con OpenMP / C++

Objetivos

Que el estudiante profundice en:

1. El diseño de programas “escalables” basados en concurrencia y memoria compartida identificando y combinando los patrones de programación paralela estructurada adecuados tales como: fork, join, map y reduction.
2. La implementación de un programa paralelo identificando y combinando directivas pragma, funciones y tipos de OpenMP en lenguaje C++.
3. La depuración sistemática de un programa hasta lograr el funcionamiento esperado utilizando algún depurador simbólico (asociado al IDE de su preferencia) o haciéndolo desde la consola mediante la interfaz de "gdb" por "comandos".
4. La evaluación del desempeño de un programa utilizando conceptos simples como el tiempo de ejecución y que sea capaz de mejorarlo.

Descripción del problema

Se deberá elaborar un programa paralelo basado en OpenMP/C++ que realice una simulación de un proceso de infección de un virus en un grupo de personas tal como se puede visualizar en: http://netlogoweb.org/launch#http://netlogoweb.org/assets/modelslib/Sample%20Models/Biology/Virus.nlogo.

Los parámetros de entrada de la simulación son:

1. Cantidad de personas (number-people): [0..10,000,000].
2. Potencia infecciosa de virus(infectiousness): [0..1].
3. Probabilidad de recuperación (chance-recover): [0..1].
4. Probabilidad de muerte: [0..1].
5. Porcentaje de personas originalmente infectadas: [0..10].
6. Tamaño del espacio bidimensional: a) 100x100, b) 500x500, c) 1000x1000.
7. Duración de la simulación (en “tics” que representarán días).

Los resultados o salida de la simulación son:

1. Promedio por tic, porcentaje y cantidad actual de personas infectadas.
2. Promedio por tic, porcentaje y cantidad actual de personas susceptibles.
3. Promedio por tic, porcentaje y cantidad actual de personas recuperadas.
4. Promedio por tic, porcentaje y cantidad actual de personas muertas.
5. Promedio final de porcentaje y cantidad de personas infectadas, susceptibles, recuperadas y muertas.

Los supuestos son que:

1. Las personas sólo se pueden morir por la infección.
2. No pueden nacer nuevas personas.
3. Una vez recuperadas, las personas nunca más pueden infectarse.

Las reglas que rigen el comportamiento de las personas son:

1. Deambulan al azar por el espacio bidimensional, trasladándose, por tic, a una posición en cualquier dirección a partir de su posición actual.
2. Una persona infectada sólo puede contagiar a otras que ocupen su misma posición en el mismo tic, de acuerdo con la probabilidad de infección y el estado de las personas con que comparte la posición. Si coincidieran dos o más personas infectadas, junto con otras, en una posición, la probabilidad de infección se incrementa linealmente de acuerdo con la cantidad de personas infectadas que coinciden.
3. Una persona puede morir sólo estando infectada de acuerdo con la probabilidad de muerte.

De acuerdo con lo anterior usted deberá programar las siguientes clases en C++:

Nombre de la clase	**Función que cumple**
Persona	Representa las personas.
Simulador	Ejecuta el proceso de simulación. Debe incluir al menos dos métodos “inicializar(...)” y “ejecutar(int N)” (N tics o días)

Funciones del programa main():

1. Capturar y validar los valores de los parámetros de entrada.
2. Generar mensajes de error para el usuario.
3. Mostrar por consola y generar archivo de las salidas por tic o día.
4. Opcionalmente (para obtener puntos extra) podrá generar una salida gráfica que muestre en colores verde (susceptible), gris (resistente) y rojo (infectado) el estado de las personas. Para optar por estos puntos se deberá implementar una “tarea” graficadora que se ejecutará en otro hilo y se comunicará con los hilos que ejecutan la simulación por medio de una cola de mensajes.

Criterios de evaluación

La calidad de su programa se valorará con base en los siguientes criterios:

1. **Escalabilidad demostrada**: su trabajo será ejecutado en al menos dos arquitecturas diferentes para comprobar que aprovecha la cantidad de núcleos de cada una, suponiendo que la cantidad óptima de hilos por núcleo es de 2 a 5.
2. **Desempeño demostrado**: su trabajo será comparado con aquél que muestre el **mejor desempeño por tic** tomando en cuenta al menos dos diseños de experimento. Entiéndase por “diseño de experimento” un conjunto de valores de parámetros.
3. **Eficacia demostrada**: su trabajo será comparado en las salidas que genera con la versión web referenciada, para lo cual se utilizará al menos dos diseños de experimentos.
4. Eficiencia en el uso de memoria, basado en las estructuras de datos utilizadas.
5. Simplicidad del código.
6. Forma y estilo del código: sangrado de bloques de código, nombres de objetos empiezan en minúsculas, nombres de clases empiezan en mayúsculas, nombres de

métodos también empiezan en minúscula pero se usan mayúsculas para concatenar palabras, comentarios para los atributos y variables de métodos.
7. División de responsabilidades entre main-modelo: el main() sólo se ocupa de la entrada de datos, generar mensajes de error, el despliegue de ciertos resultados por la consola, invocación a los demás objetos para que realicen todos los procesamientos necesarios.

Fecha de entrega: domingo 7 de octubre a las 23:55 por medio del enlace en el sitio del curso. SÓLO DEBERÁ SUBIR LOS ARCHIVOS DE CÓDIGO FUENTE (*.h y *.cpp) los archivos de salida de al menos tres ejecuciones.

Evaluación

Criterio	%Prc
Escalabilidad demostrada	20
Desempeño demostrado	20
Eficacia demostrada	60
Hasta 10/100 puntos extra por tarea graficadora y cola de mensajes	10
Hasta 10/100 puntos extra por un buen reporte de errores cuando NO funcione	10

Notas importantes:
1. Si el programa NO funciona, tendrá cero puntos en todos los rubros.
2. Este proyecto deberá realizarse idealmente y a lo más en parejas. NO SE ACEPTARÁ NINGÚN TRABAJO ELABORADO POR MÁS DE DOS PERSONAS.
3. Cada hora de atraso en la entrega se penalizará con -1/100, lo que se aplicará a la nota obtenida.
4. A TODOS LOS ESTUDIANTES INVOLUCRADOS EN UN FRAUDE SE LES APLICARÁ EL ARTÍCULO #5 INCISO C DEL "Reglamento de Orden y Disciplina de los Estudiantes de la Universidad de Costa Rica".
5. NO SUBA ningún otro archivo que no sea de código fuente (*,h y *.cpp) o de datos para evitar la transmisión de virus.